IBM Systems - iSeries

Application Display Programming

Version 5

SC41-5715-01

IBM Systems - iSeries

Application Display Programming

Version 5

SC41-5715-01

Note

| Before using this information and the product it supports, read the information in “Notices” on
| page 711 and the manual *IBM eServer Safety Information,*, G229-9054.

Second Edition (February 2006)

This edition applies to version 5, release 4, modification 0 of IBM i5/OS (product number 5722-SS1) and to all subsequent releases and modifications until otherwise indicated in new editions. This version does not run on all reduced instruction set computer (RISC) models nor does it run on CISC models.

This edition replaces SC41-5715-00.

Contents

Figures xiii

Tables xv

About Application Display Programming (SC41-5715) xvii

Summary of Changes to Application Display Programming xix

Part 1. Building a Sample Display with Online Help Information 1

Chapter 1. Building a Sample Display with Online Help Information 3

The Application Display 3
The Online Help Information 3

Part 2. Programming Application Displays Using Display Files 9

Chapter 2. Defining Your Display in a Display File 15

Establishing a Display File 15
  Determining File Descriptions 16
  Deciding Whether to Describe Data Inside or Outside Your Program 17
  Creating a Display File and Description 18
  Changing the File Description 19
  Detecting File Description Changes 20
Defining Display Fields and Functions in a Record Format 21
  DDS for Display File 21
  Record Format Used by the Program 21
  Record Format on the Display 22
  Understanding the Field Attribute Characters . . 22
  Understanding How Record Format Fields Can Be Used 22
  Defining Function Keys 24
  Defining Command Attention (CAnn) and Command Function (CFnn) Keys 25
  Specifying Alternative Keys 26
  Passing Information via Indicators. 27
  Inserting Constant Field Text from a Message Description 28
  Allowing for Right-to-Left Cursor Movement . . 28
  Defining Cursor Movement to Input-Capable Positions Only 29
  Defining Cursor Progression for Entry Fields . . 30
  Defining Attributes for Entry Fields 31
  Protecting Entry Fields Using Edit Masks . . . 31
  Specifying Right-to-Left Display Processing. . . 31
  Specifying Word Wrap for Fields 32
  Emphasizing Fields. 33
  Adding Color. 34
  Editing Output Fields 34
  Defining Your Own Edit Codes. 35
  Specifying Valid Screen Sizes 36
  Enabling Your Display to Be Printed 37
  Defining Windows 38
Using Program-Described Data 38
  Defining Input-Only Files 39
  Defining Output-Only Files 39
  Defining Input and Output Files 39

Chapter 3. Working with Display Files in an Application 41

Understanding How the System Allocates Resources 41
Opening Display Files 41
Acquiring a Display Station for I/O Operations . . 42
Obtaining Information about Display Files and Display Stations 43
  Obtaining Information about Open and I/O Operations 43
  Obtaining Attribute Information about Display Stations. 43
Sending and Receiving Data 44
  Determining Which Record Formats Are Active on a Display 44
  Writing Output to the Display 44
  Inviting Input to the Display 67
  Reading Invited Input from the Display 68
  Reading Input from the Display 70
  Writing Output and Reading Input at the Same Time. 76
  Canceling Input That Was Not Waited For . . . 77
  Locking the Keyboard and Positioning the Cursor During I/O Operations 77
  Saving Previously Displayed Information . . . 78
  Understanding the Effects of I/O Operations on Command Keys 80
  Avoiding Record Format Problems on the 5250 Display Station 80
Releasing an Acquired Display Station from I/O Operations 83
Closing Display Files 83
Mapping Display Operations to High-Level Language Operations 84
Sharing Display Files in the Same Job 85
  Understanding the Open Operation for Files Shared in a Job 85
  Understanding the Input/Output Operation for Files Shared in a Job 85
  Understanding the Close Operation for Files Shared in a Job 86

Chapter 4. Displaying Groups of Records Using Subfiles 87
Recognizing Subfile Uses 87
Describing Subfiles in Your DDS Source 89
Using a Subfile in a Program 94
Requesting I/O Operations for a Subfile 96
Requesting I/O Operations for a Subfile Record Format 96
Requesting I/O Operations for a Subfile Control Record Format 98
Recognizing Subfile I/O Requests in High-Level Languages 98
Controlling the Appearance of Subfiles 99
Displaying Horizontal Subfiles with Display Modes 100
Specifying Subfile Size Equal to Page Size 102
Specifying Subfile Size Not Equal to Page Size . . 103
Checking Validity on Subfile Data 104
Displaying Error Messages from Subfiles 105
Positioning the Cursor on the Displayed Subfile 105
Positioning the Cursor Initially 106
Positioning the Cursor When a Roll Key Is Used 106
Positioning the Cursor When a Fold or Truncate Key Is Used 109
Positioning the Cursor and Rolling When Two or More Records Are Displayed 109
Understanding Subfile DDS and Program Logic-Example 113

Chapter 5. Defining Windows with Display Files 115
Window Terminology. 115
DDS Window Keywords. 116
Window Representation and Hardware Configuration 116
Creating Windows. 116
Window Definition Records 117
Window Reference Records. 117
Window Size and Location 118
Cursor Position. 119
Error Messages 119
Subfiles 120
DDS Help Records 120
Defining Window Borders 120
Border Defaults. 121
Multiple Border Definitions 121
UIM Help Window Borders 122
Defining a Window Title 122
DDS for a Window Title-Example 123
Reading Data from Windows 124
Changing Window Borders and Contents 124
Moving and Duplicating Windows 124
Making Two Windows Seem Active at Once . . . 124
Making One Window in a Series Stand Out . . . 125
Removing Windows 126
Removing All Windows 126
Removing More Recent Windows 126
Improving Application Performance. 126
System Save and Restore Operations 126
Bypassing System Save and Restore Operations 127
Programming Examples 128
Using Basic Window Functions 128
Defining Windows in a Separate Display File 134
RPG Program Source 134
RPG Program Source for WINPGM 135

Chapter 6. Creating a Graphical Look for Displays 139
Factors Affecting the Graphical Look 139
Hardware Configuration 139
Enhanced Display Parameter 141
DDS Keywords. 142
Creating Menu Bars 144
Defining the Menu-Bar Choices 145
Suppressing the Menu-Bar Separator 145
Defining the Menu-Bar Separator. 145
Selection Fields-Overview 147
DDS for Selection Fields-Example 148
Creating a Vertical Single-Choice Selection Field 149
Creating a Vertical Multiple-Choice Selection Field 150
Creating a Horizontal Selection Field 150
Cursor Movement in a Vertical Selection Field 151
Cursor Movement in a Horizontal Selection Field 151
Controlling the Selection Indicators in a Selection Field 152
Creating Pull-Down Menus Using Single-Choice Selection Fields. 154
Controlling the Selection Indicators in a Pull-Down Menu 155
Defining Accelerator Keys 156
Defining a Menu-Bar Switch Key. 157
Defining a Cancel Key 158
Limiting Function When Cursor is Outside a Pull-Down Menu 158
Selection Lists-Overview 159
DDS for Selection Lists-Example 160
Creating Selection Lists 161
Controlling the Selection Indicators in a Selection List 162
Scroll Bars-Overview 163
Creating a Scroll Bar 164
DDS for Scroll Bars-Example 165
Scroll Bar Operation 166
Push Buttons-Overview 166
DDS for Push Buttons-Example 167
Creating Push Buttons 167
Controlling the Availability of Choices 168
Auto-Selection in Single-Choice Selection Fields 169
Auto-Enter in Single-Choice Selection Fields . . . 169
Defining Mnemonics 170
Defining Choice Colors and Attributes 171
Continued-Entry Fields-Overview 174
Specifying Word Wrap on Continued-Entry Fields 175
DBCS Considerations with Continued-Entry Fields 175
How DBCS Data is Returned for Continued-Entry Fields 175
Keyboard Functions with Continued-Entry Fields 176

Forward Field-Exit Processing 179
Backward Field-Exit Processing 180
How the Menu Bar Interacts with the Application 180
Defining the MNUBARDSP Keyword on the Application Record 180
Defining the MNUBARDSP Keyword on the Menu-Bar Record 181
Receiving Input from the Pull-Down Menus . . 182
Removing a Pull-Down Menu after Receiving Input 184
Updating a Pull-Down Menu before Displaying 184
Defining Application Help 185
Defining Choice-Level Help 185
Defining Help for a Field 187
Key Interaction for Menu Bars and Pull-Down Menus. 188
Cursor Movement 189
Pressing the Tab Key 189
Pressing the Cursor Keys 190
Programming Examples 190
Using the MNUBARDSP Keyword on the Application Record 190
Using the MNUBARDSP Keyword on the Menu-Bar Record 193
How the Displays Look 194
Simple Hotspots 195
Command Key Emulation 196
Page Up and Page Down Key Emulation . . . 196
Programmable Mouse Buttons-Overview 196
Pointer Device Events 197
AID Codes to be Returned 197
Programmable Mouse Buttons-Benefits 198
Programmable Mouse Buttons Operation . . . 198
Programmable Mouse Buttons-NWS Considerations 199
Programmable Mouse Buttons-Event Processing States 199
Programmable Mouse Buttons-Event Processing Priority 200
Grid Line Structures-Overview 205
DDS for Grid Line Structures-Example 205
Grid Line Structures and Windows 206
Hardware Requirements for Grid Line Structures 207
Inserting HTML Tags. 207
Resolving HTML Field Overlap 208
Programming Examples 209

Chapter 7. Overriding Display Files and Display File Attributes. 213

Determining Whether or Not to Use Overrides . . 213
Overriding File Attributes in HLL Programs . . . 213
Example 213
Overriding File Names in HLL Programs 214
Example 214
Overriding Both File Names and Attributes in HLL Programs. 215
Example 215
Applying Overrides When Compiling a Program 216
Example 216
Deleting Overrides 217
Displaying Overrides. 217
Using File Redirection to Override File Names and Libraries or File Types 217
Overriding Files with the Same File Types. . . 217
Overriding Files with Different File Types . . . 218
Recognizing Commands That Ignore or Restrict Overrides 220

Chapter 8. Handling Messages and Errors for Display Files 223

Creating and Displaying Your Own Messages . . 223
Displaying a Message on the Message Line . . 224
Displaying a Message on the Message Line When a Subfile Control Record is Written . . . 224
Displaying a Message on the Message Line Using a Message Field 224
Priorities for Displaying Messages on a Message Line 224
Displaying Messages in a Field on the Display 225
Displaying Messages on a Program Message Queue. 225
Displaying Error Messages through a Subfile 225
Sounding an Alarm for Messages. 228
Automatically Handling Permanent I/O Errors on Display Stations 228
Analyzing Error Messages Sent from the System 229
Understanding Messages and Message Monitors 229
Understanding Major/Minor Return Codes . . 230
Recovering from Errors 231

Chapter 9. Creating and Accessing Menus Using Display Files 235

Running System and User-Defined Menus. . . . 235
Returning to a Menu after Running the GO command 235
Determining the Previous Menu 235
Using the Cancel and Exit Keys on Menus . . 235
Choosing the Menu That Is Shown at Sign-On Time 237
Defining Your Own Display File Menus 237
Understanding DDS and Display File Considerations for Menus 238
Describing Menu Actions in a Message File . . 239
Naming Help Formats for Menus 239
Building a Display File Menu 240
Defining Your Own Program Menus. 244
Passing Parameters for Program Menus . . . 244
Building a Program Menu 244
Exiting from a Program Menu without Returning to the Previous Menu 246
Avoiding Menu Name Conflict 248
Naming Your Menus 248
Placing Your Menu in a Higher Library in the Library List 248
Specifying the Library That Contains the Menu 248
Using the Generic Menu Specification 248
Changing the Command Default after Duplicating a Command 249
Displaying Menu Attributes 249
Changing Menu Attributes 249

Deleting Menus 249

Chapter 10. Using User-Defined Data Streams. 251
Understanding Display Station Differences . . . 251
Understanding User-Defined Data Stream Limitations 252

Chapter 11. Passing Data between Programs 255
Passing Data in the Same Routing Step in a Job 255
Passing Data between Routing Steps in a Job . . . 256

Chapter 12. Waiting for Input from a Display File, an ICF File, and a Data Queue 259

Chapter 13. Using Alternative Character Sets and Code Pages . . . 261
System Has Characters Not Normally Displayed on the Device 261
Device Passes Characters Not Displayed on the System 262
Specifying Character Translation for Fields . . . 262
Determining the Character Identifier (CHRID) Value for Your Display 264

Chapter 14. Improving System Performance with Displays 267
Deferring the Write Operation for a Display File 267
Designating the Primary Screen Size for a Display File. 267
Writing Only One Page of Subfile Records at a Time 267
Sharing an Open Data Path (ODP) for the Same Job 268
Sending Records with Input Fields to the Display in Order 268
Overlapping and Not Deleting Repeatedly Sent Records 268
Restoring the Display 268
Defining Command Attention Keys Rather Than Command Function Keys 269
Using the Invite Operation 269
Using Windows 269

Part 3. Programming Application Displays Using Panel Groups . . . 271

Chapter 15. Improving Productivity with User Interface Manager 273
Increasing User Productivity 273
Increasing Application Programmer Productivity 273
What to Consider before Using UIM Instead of Data Description Specifications (DDS) 273

Chapter 16. Introduction to the User Interface Manager 275
Overview of UIM 275
What the UIM Supports 276
What Is a Panel Group 276
What Is a Menu 276
Creating Objects 277
Elements Within a Panel Group 277
Using the UIM Language Tags 277
Using Dialog Commands 278
Using Control Language (CL) Commands 279
Using an Application Programming Interface (API) 279
Defining a Menu Object Using UIM 280
Creating a Menu Panel 280
  Required Tags for a Menu Panel 282
  Source for Example Menu 282
Defining a Panel Group Object Using UIM . . . 287
Creating a List Panel 287
  Required Tags for a List Panel. 289
  Source for Example List Panel. 290
  Application Programming for a List Panel. . . 298
Creating a Confirmation List Panel 299
  Required Tags for a Confirmation List Panel . . 300
  Source for Example Confirmation Panel . . . 300
  Automatic Confirmation Processing 303
  Application Programming for Confirmation Processing 303
Creating a Data Presentation Panel 303
  Required Tags for a Data Presentation Panel . . 306
  Source for Example Data Presentation Panel . . 306
  Application Programming for a Data Presentation Panel. 317
  Data Entry Panel 320
Creating a Panel with a Menu Bar 320
  Required Tags for a Panel with a Menu Bar . . 322
  Source for Example Panel with a Menu Bar . . 323
  Application Programming for a Menu Bar Panel 332

Chapter 17. Details of Using User Interface Manager 335
Opening a UIM Application 335
  Defining Dialog Variables 335
  Restrictions on Using Dialog Variables 336
  Dialog Variable Error Messages 337
  Providing Field Values for a Display Panel Using Dialog Variables 337
  Using Variable Pool Services 338
| Dialog Variables and Special Values 338
  Character Set and Code Page Considerations 338
Managing a List 340
  Defining a List 341
  Initializing a List 341
  Displaying a List 341
  Updating a List. 342
  Incomplete List Processing 342
  Removing and Inserting an Entry from a List 342
  Controlling List Entries on a List Display . . . 343
  Improving Interactive Response Time for a List Display 343
  Using Action Lists and Selection Lists 344

Using Selection Characters 345
Managing Panel Functions 345
Enabling Conversion to a GUI. 346
Scrolling Support 346
Defining Scrollable Areas 346
Defining Function Key Scrolling 347
Scrolling and Error Conditions 347
Scrolling a List Area 347
Scrolling a Menu Area 347
Scrolling an Information Area 348
Scrolling Data Item Groups. 348
Scrolling a Text Area 348
Defining Contextual Help 348
Command Line Restrictions 350
Command Line Interpretation 350
Entering Commands That Are Too Long . . . 351
Defining Function Keys 351
Formatting Function Keys 351
Handling Function Keys and VARUPD Value 351
Panel Formatting Concepts 352
When Panel Formatting Is Performed 353
Application Control of Panel Formatting . . . 353
Limits of the Panel Formatter 354
Folding Up Multiple Panels When EXIT Is Requested 354
Folding Up a List Panel 355
Adding a Pop-Up Window over Another Panel 356
Using Menu Bars 357
Differences Between Pull-Down Menus and Pop-Up Windows 358
Using Pop-Up Windows. 359
Defining Application Windows 359
Adding and Removing Windows. 360
Using the Command Line in a Window . . . 361
UIM as a Request Processor Program When Displaying a Panel 361
Printing Concepts 362
Printing a Print Head Panel 364
Printing a Print Panel 364
Using Blank Lines for Separating. 364
Fonts and Highlighting 364
Printing the Trailer 364
Defining Prolog Areas 364
Defining Header Areas 365
Using the Page-Eject Function During Printing 365
Sharing and Overriding Printer Files 365
Printing Double-Byte Character Set (DBCS) Considerations 365
Commonly Asked UIM Questions 365

Part 4. Programming Help Displays 367

Chapter 18. Making Online Help Information Accessible for Your Display File 369

Enabling the Help Key 370
Choosing between Panel Groups and Records for Help 370
Defining Which Areas of Your Display Need Online Help Information 371
Specifying Panel Groups for Help in Your Display File. 373
Defining Panel Groups with Option Indicators 375
Copying QUSRTOOL Examples That Specify Help Using Panel Groups 376
Specifying Records in Your Display File 376
Defining Records with Option Indicators . . . 377
Entering the Records That Contain the Help Information 378
Using Records and Documents for Help in the Same Display File 378
Understanding the Restrictions on Records . . 379
Paging between Help Displays That Use Records 379
Returning Control to Your Program after Pressing the Help Key 383
Returning Control to Your Program after Showing the Help Display 383
Returning Control to Your Program without Showing the Help Display 384

Chapter 19. Making Online Help Accessible for Your Panel Group . . . 385

Definitions and Explanations 385
Giving Help Panel Groups Access to Index Search. 387
Giving Help Panel Groups Access to A User-Defined Panel Group 387
Removing Access to F18=More Indexes. . . . 388
Help in a List Area 388
Coding Help 389
Help in a Menu Area. 389
Coding Help 390
Help in a Data Area 391
Coding Help 392
Help in a Menu Bar Area 394
Coding Help 395
Help in a Function Key Area 396
Coding Help 397

Chapter 20. Defining Online Help Information 399

Defining Online Help Information in a Panel Group. 399
Entering the UIM Source for a Panel Group for Help 399
Creating and Deleting Panel Groups. 410
Assigning Panel Groups as Help for Commands 411
Using Panel Groups in a Search Index 411
Copying QUSRTOOL Examples That Define Help in a Panel Group 412
Defining Online Help Information in a DDS Record 412

Part 5. Guidelines for IBM i5/OS-Style Displays 413

Chapter 21. Designing IBM i5/OS-Style Displays . . . 415

- Using the Displays Example in the QUSRTOOL Library . . . 415
  - Recognizing the Example Objects . . . 415
  - Installing the Example Objects . . . 417
  - Viewing the Sample Displays, Command, and Online Help Information . . . 417
  - Copying the Source for the Example Objects for Your Own Use . . . 420
- Defining Special Functions and Attributes for All Displays . . . 421
- Designing the Single-Choice Menu Display . . . 421
  - Title . . . 422
  - Instruction Line . . . 422
  - Menu Options . . . 422
  - Menu Selection Entry Field . . . 423
  - Function Keys . . . 423
  - Online Help Information . . . 423
  - General Menu Display Operation . . . 423
- Designing the Entry Display . . . 423
  - Title . . . 424
  - Instruction Line . . . 424
  - Prompt Area . . . 425
  - Function Keys . . . 428
  - Online Help Information . . . 428
  - General Entry Display Operation . . . 429
- Designing the Information Display . . . 429
  - Title . . . 430
  - Location Information . . . 430
  - Prompt Area 1 . . . 431
  - Prompt Area 2 . . . 431
  - Instruction Line . . . 432
  - Function Keys . . . 432
  - Online Help Information . . . 432
  - General Information Display Operation . . . 432
- Designing the List Display . . . 432
  - Title . . . 433
  - Prompt Area . . . 433
  - Instruction Line . . . 434
  - Options Line . . . 434
  - Column Headings . . . 435
  - Extended Action Entry Area . . . 435
  - List Fields . . . 435
  - Paging Location Information . . . 436
  - Function Keys . . . 436
  - Online Help Information . . . 436
  - General List Display Operation . . . 437
- Defining the Function Key Area for All Displays 443
  - Optional Command Line and Identifier Field 443
  - Common Key Assignments . . . 446
- Defining Help Information for All Displays . . . 450
  - Help for the Menu Display . . . 450
  - Help for the Entry Display . . . 451
  - Help for the Information Display . . . 452
  - Help for the List Display . . . 453
- Defining and Presenting Messages . . . 454
- Designing Common User Access (CUA) Entry Level Models . . . 456
  - Entry Dialog Actions . . . 456
  - Single-Choice Selection (Menu) . . . 457
  - Entry Display . . . 459
  - Information Display . . . 460
  - List Display . . . 460
  - Help Information . . . 461

Part 6. Appendixes . . . 463

Appendix A. UIM Panel Group Definition Language . . . 465

- Tag Content Formatted as Paragraphs . . . 466
- Panel Areas . . . 467
- Panels . . . 467
- Panel Group Objects . . . 467
- Help on Panels . . . 468
  - Panel Group Organization . . . 468
  - Name Syntax . . . 469
  - Symbols . . . 469
  - Comments . . . 471
  - Imbeds . . . 471
  - DBCS Graphic Literals . . . 471
  - Hexadecimal Literals . . . 472
- APPFMT (Application Formatted Area) . . . 472
  - Required Attributes . . . 473
  - Optional Attribute . . . 473
  - Application Formatted Data . . . 473
  - Example: Application Formatted Area . . . 474
- BOTINST (Bottom Instruction) . . . 474
  - Optional Attribute . . . 475
  - Optional Text . . . 475
- CHECK (Validity Checking) . . . 475
  - Required Attribute . . . 476
  - Optional Attributes . . . 476
  - Example: Validity Checking . . . 477
- CIT (Title Citation) . . . 477
  - Optional Text . . . 477
  - Example: Title Citations . . . 477
- CLASS (Class Definition) . . . 478
  - Required Attributes . . . 479
  - Optional Attributes . . . 483
  - Example: Class Definitions . . . 488
  - Display Forms of Numeric Values . . . 489
  - Display Forms of Character, Date, and Time Values . . . 490
- CMDLINE (Command Line) . . . 491
  - Required Attribute . . . 491
  - Optional Attribute . . . 491
  - Optional Text . . . 491
- COND (Condition Definition) . . . 492
  - Required Attributes . . . 492
  - Optional Attribute . . . 495
  - Example: Conditioning an Option . . . 495
- COPYR (Copyright) . . . 496
  - Required Text . . . 496
- DATA (Data Presentation Area) . . . 496
  - Required Attribute . . . 497
  - Optional Attributes . . . 498
  - Optional Text . . . 499
  - Print Formatting Considerations . . . 499
  - Example 1: Data Entry Panel . . . 500

Example 2: Two-Column Format in a Data Entry Panel 501
Example 3: Two Presentation Areas for Data Items 503
Example 4: Data Presentation Area with a Menu Area 504
Example 5: Data Entry Panel with a Nested Data Group 505
DATAC (Data Item Choices) 506
Optional Attribute 506
Optional Text 507
DATACOL (Data Column) 507
Required Attribute 507
Optional Text 507
DATAGRP (Data Group) 508
Optional Attributes 508
Optional Text 509
DATAI (Data Item) 510
Required Attributes 511
Optional Attributes 511
Optional Text 514
DATAIX (Data Item Extender) 515
Required Attributes 516
Optional Attributes 516
DATASLT (Data Selection Field) 518
Required Attributes 519
Optional Attributes 519
Optional Text 520
Example 1: Data Entry Panel 520
Example 2: Multiple-Selection Field 522
DATASLTC (Data Selection Field Choice) 523
Optional Attributes 523
Optional Text 525
DL (Definition List) 525
Optional Attribute. 526
Required Tags 526
Optional Tags 526
Example 1: Definition List 526
Example 2: Compact Definition List 527
FIG (Figure) 527
Optional Attribute. 528
Optional Tag 528
Example: Sample Figure 528
HELP (Help Module). 529
Required Attribute 529
Optional Attributes 529
Optional Text 530
Example: Help Panel Definition 531
HP0 through HP9 (Highlighted Phrase) 532
Optional Text 532
H1 through H4 (Heading) 533
Required Text 533
Example: Heading Tags 534
IMHELP (Imbed Help) 534
Required Attribute 534
Example: Imbedded Help 535
IMPORT (Import) 536
Required Attributes 536
Optional Attributes 536
INFO (Information Area) 537
Required Attribute 537
Optional Attributes 537
Optional Text 538
Print Formatting Considerations 538
ISCH (Index Search) 538
Required Attribute 538
Required Text 538
Example: Index Search 539
ISCHSUBT (Index Search Subtopic) 539
Required Attribute 539
Example: Index Search Hierarchy. 539
ISCHSYN (Index Search Synonym) 540
Required Attribute 541
Required Text 541
Example: Index Search Synonyms 541
KEYI (Key List Item) 542
Required Attributes 542
Optional Attributes 543
Optional Text 544
Example: Key Definitions 544
KEYL (Key List) 545
Required Attribute 545
Optional Attribute. 545
Example: Key List 545
LINES (Unformatted Lines). 546
Optional Text 546
Example: Unformatted Lines 546
LINK (Hypertext Link Definition) 547
Required Attribute 548
Optional Attributes 548
Conditional Expressions 548
Bidirectional Considerations 550
Example: Hypertext Link 550
LIST (List Area) 552
Required Attributes 553
Optional Attributes 553
Optional Text 556
Print Formatting Considerations 556
Example 1: List Area 557
Example 2: List Area with Three Layout Columns 558
Example 3: List Area with List Column Groups 559
Example 4: Dynamic List Column Heading Formatting 560
LISTACT (List Action) 562
Required Attributes 563
Optional Attributes 563
Optional Text 566
Confirmation Panel Requirements 566
Confirmation Panel Conventions 567
Example: List Actions 567
LISTCOL (List Column) 568
Required Attributes 569
Optional Attributes 569
Optional Text 571
Formatting Considerations 572
LISTDEF (List Definition) 573
Required Attributes 573
Optional Attributes 573
LISTGRP (List Column Group) 575
Required Attributes 576
Optional Attribute. 576

Optional Text 576
Example: List Column Group 577
LISTVIEW (List View) 578
Required Attribute 578
Optional Attribute. 578
LP (List Part) 579
Example: List Part. 580
MBAR (Menu Bar) 580
Required Attribute 581
Optional Attributes 581
Example: Menu Bar 581
MBARC (Menu Bar Choice) 583
Required Attributes 583
Required Text 583
MENU (Menu Area) 584
Required Attribute 584
Optional Attribute. 585
Optional Text 585
Example 1: Simple Menu Area. 585
Example 2: Menu Area with Groups. 586
MENUGRP (Menu Group) 587
Optional Attribute. 588
Optional Text 588
MENUI or MI (Menu Item). 588
Required Attributes 588
Optional Attributes 589
Optional Text 590
NT or NOTE (Note) 590
Optional Text 591
Example: Using a Note 591
OL (Ordered List) 591
Optional Attribute. 592
Required Tag 592
Example: Ordered List 592
OPTLINE (Option Line) 593
Optional Attribute. 593
Optional Text 593
P (Paragraph) 593
Optional Text 594
Example: Paragraph Tag. 594
PANEL (Display Panel) 595
Required Attributes 596
Optional Attributes 596
Optional Text 601
Example: Panel Definition 601
PARML (Parameter List). 602
Required Tags 603
Example: Parameter List. 603
PC (Paragraph Continuation) 603
Example: Paragraph Continuation 603
PDACCEL (Pull-Down Accelerator) 604
Required Text 604
PDFLD (Pull-Down Field) 604
Optional Attribute. 605
PDFLDC (Pull-Down Field Choice) 605
Required Attributes 605
Optional Attributes 606
Optional Text 608
Confirmation Panel Requirements 608
Confirmation Panel Conventions 608
PK (Programming Keyword) 609
Optional Attribute. 609
Required Text 609
PNLGRP (Panel Group) 609
Optional Attributes 610
PRTHEAD (Print Head Panel). 613
Required Attribute 613
Optional Attributes 613
Optional Text 615
Layout of the Title Lines. 615
Example: Print Title Line 617
PRTPNL (Print Panel) 618
Required Attribute 618
Optional Attributes 618
Optional Text 619
PRTTRAIL (Print Trailer Message) 619
Required Text 619
Example: Trailer Message 619
PV (Programming Variable) 619
Required Text 620
RT (Reverse Text) 620
Example 1: Left-to-Right Formatting on a Right-to-Left Panel 621
Example 2: Left-to-Right Formatting on a Left-to-Right Panel 621
SL (Simple List) 622
Optional Attribute. 623
Required Tag 623
Example: Simple Lists 623
TEXT (Text Area) 623
Required Attribute 624
Optional Attributes 625
Cursor positioning 625
Text Data. 625
Example: Text area 628
TI (Translation List Item) 628
Optional Attribute. 628
Optional Text 629
TL (Translation List) 629
Optional Attributes 629
Examples: Translation List 630
TOPINST (Top Instruction) 630
Optional Attribute. 631
Optional Text 631
TT (Truth Table) 631
Required Attributes 632
Example: Truth Table. 632
TTROW (Truth Table Row) 632
Required Attribute 632
UL (Unordered List) 633
Optional Attribute. 634
Required Tag 634
Example: Unordered Lists 634
VAR (Variable Definition) 634
Required Attribute 635
Optional Attributes 635
Dialog Variables Defined by UIM. 635
VARRCD (Variable Record Definition) 637
Required Attributes 637
Optional Attributes 638
XH1 through XH4 (Extended Help Headings) . . 638
Required Text 638

Formatting Rules 639
Example: Sample Headings. 639
XMP (Example). 639
Example: Formatting an Example. 640

Appendix B. UIM Dialog Commands 641

The VARUPD Attribute 643
ACTIONS (Menu Bar Cursor Action) 644
Messages 644
CALL (Call Program). 644
Required Parameter 645
CANCEL 646
Optional Parameter 646
CHGVIEW (Change View) 647
Parameters 647
VARUPD Value. 647
Messages 647
CMD (System Command) 647
Parameter 648
VARUPD Value. 648
Messages 648
Hint 648
CMDLINE (Command Line) 648
Parameters 648
Messages 648
DSPHELP (Display Help) 648
Required Parameter 649
Optional Parameter 649
ENTER 649
Parameters 649
VARUPD Value. 649
Messages 649
Considerations for Using the ENTER, HELP, and PROMPT Dialog Commands. 649
EXIT (Exit Display) 650
Optional Parameter 650
EXTHELP (Extended Help). 650
Parameters 650
HELP 651
Parameters 651
VARUPD Value. 651
Messages 651
HELPHELP 651
Parameters 651
Messages 651
HELPIDX 651
Parameters 652
Messages 652
HOME (Display Home Menu). 652
Parameters 652
VARUPD Value. 652
Messages 652
KEYSHELP 652
Messages 652
MENU 653
Required Parameter 653
Optional Parameter 653
MOREKEYS (Display More Function Keys) . . . 653
Parameters 654
Messages 654
MOVETOP (Move to Top) 654
Parameters 654
VARUPD Value. 654
Messages 654
MSG (Display Message) 654
Required Parameter 655
Optional Parameter 655
PAGEDOWN 655
Parameters 655
Messages 655
PAGEUP 655
Parameters 656
VARUPD Value. 656
Messages 656
PRINT (Print Display) 656
Parameters 656
VARUPD Value. 656
Messages 656
PROMPT 656
Parameters 657
VARUPD Value. 657
Messages 657
Prompting an Entry Field 657
Prompting an Action List Option or Command 657
PULLDOWN (Display Pull-Down Menu) 658
Parameters 658
RETRIEVE (Retrieve Command String) 658
Parameters 659
VARUPD Value. 659
Messages 659
RETURN (Return Control to Application) . . . 659
Required Parameter 659

Appendix C. Feedback Area Layouts for Display Files 661

Open Feedback Area 661
Device Definition List 664
I/O Feedback Area 667
Common I/O Feedback Area 667
I/O Feedback Area for Display Files. . . . 670
Get Attributes 673

Appendix D. Display File Return Codes 677

Major Code 00 677
Major Code 02 677
Major Code 03 678
Major Code 04 679
Major Codes 08–11 679
Major Code 34 680
Major Code 80 680
Major Code 81 683
Major Code 82 684
Major Code 83 688

Appendix E. Edit Codes 693

i5/OS Edit Codes 693
Examples of Editing Using i5/OS Edit Codes 694
User-Defined Edit Codes 695
Using User-Defined Edit Codes 696
Example of a User-Defined Edit Code 697

Appendix F. System/36-Compatible Display Data Management 699
Clearing Lines on the Display 700
Input Data for Display File Records 700
Input Data from the Work Station Controller . . . 700
Self-Check 701
Return Input 701
Erase Input Fields 701
Display Attributes 702
Positioning the Cursor 702
Displaying Messages 703
Put Override 705
Handling Signed Numeric Data 705
Function Keys 706
Help Key Considerations 706
Using Command Keys to Exit Application Help 707
Cancel-Invite Operation 707
Retain Command and Function Keys 708
System/36 Functions Not Supported 709
Restricted DDS Keywords/Functions 709

Notices 711
Programming Interface Information 712
Trademarks 712
Terms and conditions. 713

Bibliography 715
System Use 715
Systems Management 715
Application Development 715
Communications and Connectivity 715
Program Enablers 716
Program Interfaces 716

Index 717

Figures

1. Sample DDS Source for a Display File . . . 21
2. Record Formats in the Program and on the Display 22
3. DDS for Field-Level Cursor Progression 30
4. DDS for Subfile-Level Cursor Progression 30
5. Sample DDS for Right-to-Left Display Processing 31
6. Sample DDS for Program-to-System Fields 34
7. Sample DDS for Two Display Sizes 36
8. Sample DDS for Subfiles for Two Display Sizes 37
9. Valid Placement of Records on a Screen When the CLRL Keyword Is Not Used 46
10. Wrong Placement of Records on Screen When CLRL Keyword Not Used 47
11. Replacing Record Formats 49
12. Sample DDS Source Showing Use of the SLNO(*VAR) Keyword 52
13. Sample DDS Source Showing Difference between CLRL and OVERLAY 55
14. Sample DDS Source Showing Use of the PUTOVR Keyword 58
15. Sample DDS Source Showing Efficient Use of PUTOVR Keyword 59
16. Sample DDS Source Showing Another Use of PUTOVR Keyword 61
17. Sample DDS Source Showing Use of the PUTRETAIN Keyword 62
18. Sample DDS Source Showing Use of the PUTRETAIN Keyword 63
19. Sample DDS Source Showing Use of the DSPMOD Keyword 65
20. Sample DDS to Show Record Format Problems 82
21. DDS Keyword Processing Order for Subfile Control 94
22. Vertically Displayed Subfile 99
23. Horizontally Displayed Subfile 99
24. Horizontally and Vertically Displayed Subfiles Displayed at the Same Time 100
25. Sample DDS Using DSPMOD with Subfiles 101
26. Sample DDS Using SFLSIZ, SFLPAG, and ROLLUP Keywords 102
27. Sample DDS for a Variable-Length Record 103
28. Sample DDS Using the SFLNXTCHG Keyword 104
29. Sample DDS for a Message Subfile 105
30. Sample DDS Using the DSPATR(PC) Keyword 107
31. Sample DDS Showing Customer Name Search Subfile 113
32. Window Title-Display Example 123
33. DDS for a Window Title 123
34. Radio Buttons and Check Boxes 139
35. Example of a Menu Bar 144
36. DDS for a Menu Bar 146
37. Menu Bar on a Graphical Display Station with Enhanced Interface 146
38. Menu Bar on a Nongraphical Display Station with Underline Capability 147
39. Menu Bar on a Nongraphical Display Station without Underline Capability 147
40. Menu Bar on a Display Station without Enhanced Interface. 147
41. Selection Fields on a Graphical Display Station with Enhanced Interface 148
42. Selection Fields on a Nongraphical Display Station 148
43. DDS for Single-Choice and Multiple-Choice Selection Fields 149
44. Example of DDS for Horizontal Selection Field 150
45. DDS for Suppressing Selection Indicators in a Selection Field 153
46. Suppressed Selection Indicators in Selection Field 153
47. Example of a Pull-Down Menu 154
48. DDS for a Pull-Down Menu 154
49. Pull-Down Menu on a Graphical Display Station with Enhanced Interface 155
50. Pull-Down Menu on a Nongraphical Display Station with Underline Capability. 155
51. Pull-Down Menu on a Nongraphical Display Station without Underline Capability . . . 155
52. Pull-Down Menu on a Display without Enhanced Interface. 155
53. DDS for Suppressing Selection Indicators in a Pull-Down Menu 156
54. Suppressed Selection Indicators on Graphical Display Station 156
55. Suppressed Selection Indicators on Nongraphical Display Station 156
56. DDS for Accelerator Keys 157
57. Accelerators in a Pull-Down Menu 157
58. DDS for Menu-Bar Switch Key and Cancel Key 158
59. Selection Lists on a Graphical Display Station with Enhanced Interface 159
60. Selection Lists on a Nongraphical Display Station with Underline Capability. 160
61. DDS for Selection Lists-Example 160
62. DDS for Enabling Selection Indicators in a Selection List. 162
63. Selection Indicators on Graphical Display Station. 163
64. Scroll Bar on a Graphical Display Station with Enhanced Interface. 164
65. Scroll Bar on a Nongraphical Display Station with Underline Capability 164
66. DDS for Scroll Bars-Example 166
67. Push Buttons on a Graphical Display Station with Enhanced Interface 167
68. Push Buttons on a Nongraphical Display Station with Underline Capability. 167

69. Push Buttons on a Nongraphical Display Station without Underline Capability . . . 167
70. Push Buttons on a Display Station without Enhanced Interface. 167
71. DDS for Push Buttons-Example 167
72. Control Values for the CHCCTL Keyword 168
73. DDS to Control the Availability of Choices 169
74. Single-Choice Selection Field with an Unavailable Choice 169
75. Examples of Valid DDS for Mnemonics 170
76. Examples of DDS Not Valid for Mnemonics 170
77. DDS Using CHCAVAIL and CHCSLT for Menu-Bar Choices 172
78. DDS Using CHCAVAIL and CHCUNAVAIL for Selection Fields. 173
79. DDS Using CHCAVAIL, CHCUNAVAIL, and CHCSLT for Selection Fields 173
80. DDS Using CHCAVAIL, CHCUNAVAIL, and CHCSLT for Single Choice Selection List Choices 173
81. DDS Using CHCAVAIL, CHCUNAVAIL, and CHCSLT for Multiple Choice Selection List Choices 174
82. Continued-Entry Fields in Rectangular Arrangement. 174
83. DDS Using MNUBARDSP on the Application Record 180
84. DDS Using MNUBARDSP on the Menu-Bar Record 181
85. DDS for Pull-Down Input (PULLINPUT) Parameter. 183
86. DDS for Return-Field Parameter 185
87. DDS for Menu-Bar Choice Help 186
88. DDS for Single-Selection Field Choice Help 186
89. A help list for a menu bar 187
90. Help for a Named Field 188
91. Help for a Constant Field 188
92. Cursor Locations 188
93. Grid line structures 205
94. DDS for Grid Line Structures-Example 206
95. DDS Coding Before Adding HTML Keyword 209
96. DDS Coding After Adding HTML Keyword 210
97. Graphic Image on an i5/OS 5250 Gateway Display 210
98. Sample DDS Source for ERRSFL Keyword 226
99. Sample DDS Source for SFLMSGID Keyword 227
100. DDS Source for Sample Menu Called PERSMENU 242
101. DDS Source for Program Menu Example 245
102. An Example Menu. 281
103. Required UIM tags for a menu panel 282
104. Example List Panel 287
105. Example of Alternate View of List 288
106. Required tags for a list panel 290
107. Example Confirmation List Panel 300
108. Example Data Presentation Panel 304
109. Example Data Presentation Panel after Scrolling 305
110. Required UIM tags for a data presentation panel 306
111. Example Panel with a Menu Bar 321
112. Required UIM tags for a panel with a menu bar 323
113. Example of Job Exit Flag. 355
114. Example of Printout 363
115. Sample H Specification in DDS Source 372
116. Sample DDS Source Showing HLPPNLGRP 375
117. Sample DDS Source Showing HLPPNLGRP and Option Indicators. 376
118. Sample DDS Source Showing HLPRCD 377
119. Sample DDS Source Showing HLPRCD and Option Indicators 378
120. Sample DDS Source with HLPRCD 378
121. Sample DDS Source to Show Secondary Help 382
122. Sample DDS Source to Show HLPCMDKEY 383
123. Sample DDS Source to Show HLPCMDKEY and Response Indicators 384
124. Index Search Display 387
125. Sample Menu in QUSRTOOL 418
126. Sample Entry Display in QUSRTOOL 418
127. Sample Information Display (Two Pages) in QUSRTOOL 419
128. Sample List Display in QUSRTOOL 420
129. Sample Application Menu 422
130. Sample Entry Display. 424
131. Sample Information Display (Two Pages) 430
132. Sample List Display 433
133. Layout of Display with Location Information 436
134. Example of Processing Priority with List Display 439
135. Create User Profile Entry Display 444
136. Create Command Display with Additional Parameters Selected 445
137. Second Display of Additional Parameters 445
138. Work with Members Using PDM Entry Display 446
139. Help Areas for Entry Displays 452
140. Help Areas for Information Displays 453
141. Help Areas for List Displays 454
142. Example of an Application Menu 458
143. Entry Display 459
144. Example of an Information Display 460
145. Example of a List Display 461
146. Highlighting Classes Allowed in TEXT Area 626
147. Circumventing the Save Command 708

Tables

1. Names Used in Steps for Creating Sample Displays 3
2. More Ways to Create Application Displays 7
3. More Ways to Create Online Help Information 8
4. Column positions for sample DDS 21
5. DDS for Emphasizing Fields 33
6. PRINT Keyword Results Using Print Key 37
7. Display Stations Implicitly Acquired When Display Files Are Opened 42
8. Information Available from the Get-Attributes Operation 43
9. Results of SLNO(*VAR) Values 50
10. Results from CLRL Example 55
11. Keywords Ignored If Display Modes Are Changed 65
12. Display File Operations Supported by the Operating System and the Equivalent High-Level Language Commands 84
13. Optional Functions for Subfiles 90
14. Subfile Operations Supported by the System and Equivalent HLL Commands 98
15. Functions Supported by Hardware Configurations A, B, and C 139
16. Functions Supported by Hardware Configurations D, E, and F 140
17. How a Scroll Bar is Sized 165
18. Scroll Bar Operation 166
19. Keywords Used to Define Colors and Display Attributes 171
20. Values Returned in MNUCHOICE and PULLINPUT 183
21. Actions Performed at Different Cursor Locations 189
22. File Redirections 218
23. File Redirection Combinations 219
24. System Message Number Ranges 230
25. Major Return Code Definitions. 231
26. Restrictions for Display File Menus 238
27. Suggestions for Display File Menus 239
28. Display File and ICF File Entry Field Attributes 259
29. CHRID Values 264
30. Initial Values of Dialog Variable 336
31. UIM CCSID/CHRID Conversions for Display 339
32. UIM CCSID/CHRID Conversions for Print 340
33. Cursor-Sensitive Function Keys Assigned to Dialog Commands 358
34. Different Ways to Define Online Help Information 370
35. Characteristics of Different Methods of Online Help Information 370
36. Help for Sample Display 372
37. Other DDS Keywords for UIM Help 373
38. Help for Sample Display Using Panel Groups 374
39. Help for Sample Display Using HLPRCD 377
40. Source Members for Displays Example in QUSRTOOL (Install, Create, and Delete) 415
41. Source Members for Displays Example in QUSRTOOL (Sample Displays, Command, and Online Help Information) 416
42. Objects Created When Creating Example Objects. 417
43. Required Functions and Attributes of All i5/OS-Style Displays 421
44. Optional Functions and Attributes of All i5/OS-Style Displays 421
45. Function Key Assignments 447
46. Type of Help for Each Help Area-Menu Display 450
47. DDS Considerations-Help on Menu Displays 451
48. Type of Help for Each Help Area-Entry Display 451
49. DDS Considerations-Help on Entry Displays 452
50. Type of Help for Each Help Area-Information Display 452
51. DDS Considerations-Help on Information Displays 453
52. Type of Help for Each Help Area-List Display 453
53. DDS Considerations-Help on List Displays 454
54. CUA Entry Dialog Actions 456
55. Tag Attributes That Can Be Continued 466
56. Restrictions Associated With Mixed Panels 467
57. Tags Allowed Between the CLASS and ECLASS Tag 479
58. Attribute Summary for Each BASETYPE 483
59. Tags Allowed Between the DATA and EDATA Tags 497
60. Tag Allowed Between the DATASLT and EDATASLT Tags 519
61. Tags Allowed Between the DL and EDL Tags 526
62. Tag Allowed Between the KEYL and EKEYL Tags 545
63. Tags Allowed Between the LIST and ELIST Tags 553
64. NOCMD and NOEXT Attribute Interaction 565
65. Emphasis Values 574
66. Layout Values for Width=80 579
67. Layout Values for WIDTH=132 579
68. Layout Values for WIDTH=132 579
69. Tag Allowed Between the MBAR and EMBAR Tags 580
70. Tags Allowed Between the MBARC and EMBARC Tags 583
71. Tags Allowed Between the MENU and EMENU Tags 584
72. Tag Allowed Between the MENUGRP and EMENUGRP Tags 588
73. Tags Allowed Between the NOTE and ENOTE Tags 590
74. Tags Allowed Between the OL and EOL Tags 592

75. Tags Allowed Between the PANEL and EPANEL Tag 595
76. Layout of UIM finger print 598
77. Tags Allowed Between the PARML and EPARML Tags 602
78. Tags Allowed Between the PDFLD and EPDFLD Tags 605
79. Valid Action Text for ACTION Values 606
80. Tags Allowed Between the PNLGRP and EPNLGRP Tags 610
81. Tags Allowed Between the PRTHEAD and EPRTHEAD Tags 613
82. First Line of Heading with Print Width 132 615
83. First Line of Heading with Print Width 80 615
84. Second Line of Heading with Print Width 132 615
85. Second Line of Heading with Print Width 132 and time zone 616
86. Second Line of Heading with Print Width 80 617
87. Second Line of Heading with Print Width 80 and time zone 617
88. Tags Allowed Between the PRTPNL and EPRTPNL Tags 618
89. Tags Allowed Between the SL and ESL Tags 622
90. Tag Allowed Between the TL and ETL Tags 629
91. Tag Allowed Between the TT and ETT Tags 632
92. Tags Allowed Between the UL and EUL Tags 633
93. Attributes of UIM-Defined Variables (Z-Variables) 637
94. Summary of the Valid Uses of Dialog Commands 641
95. Summary of the Effects of Dialog Commands 642
96. Open Feedback Area 661
97. Device Definition List 664
98. Common I/O Feedback Area 667
99. I/O Feedback Area for Display Files 670
100. Get Attributes 673
101. Summary Chart for i5/OS Edit Codes 693
102. Valid Edit Codes, Source Data, and Edited Output 695
103. IBM-Supplied Edit Descriptions 695
104. Message Files for MSGID 703
105. Message Files for MSGID 704

About Application Display Programming (SC41-5715)

This book contains information about the following topics:

- Using DDS to create and maintain displays for an application
- Creating and working with display files on the system
- Creating online help information
- Using UIM to define panels and dialogs for an application

Use this book to program for application and help displays. This book does not describe all the DDS keywords or the configuration of display stations. You may need to refer to other IBM books for more specific information about a particular topic. For a list of related publications, see "Bibliography" on page 715.

This book should be used by application programmers who create or work with application and help displays. You should also have knowledge of the source entry utility (SEU) and data description specifications (DDS).

This book assumes that a device description already exists to describe your display station to the system.

Summary of Changes to Application Display Programming

This edition of the Application Display Programming book incorporates numerous technical updates that occurred after the book was last published. These changes are marked with a vertical bar on the margin beside the changed information. In addition, some references to UIM keywords that have been removed from the operating system have been deleted.

Part 1. Building a Sample Display with Online Help Information

Chapter 1. Building a Sample Display with Online Help Information 3
The Application Display 3
The Online Help Information 3

Chapter 1. Building a Sample Display with Online Help Information

This chapter outlines the steps you need to do to create a sample display with online help information on i5/OS. If you are not sure how to do one or more of the steps, see the additional information referred to in each step.

The Application Display

The sample display is created using a display file (also known as a display device file). A **display file** is an object, or named storage space, created by the user that contains the file description. The **file description** identifies the display station used and, optionally, the record formats used by the display station. **Record formats** describe the characteristics and arrangement of the fields on a display. Record formats are defined using **data description specifications (DDS)**, which describe data attributes outside the application program that processes the data.

The Online Help Information

The online help information for the sample display is defined using help panel groups. A **panel group** is an object, or named storage space, that contains text to be used as online help information by the user interface manager. The **user interface manager (UIM)** is a function of the operating system that provides online help information for displays, including help for part or all of a display, help for commands, the index search function (selectable help topics), and hypertext (the capability to link different units of online help information).

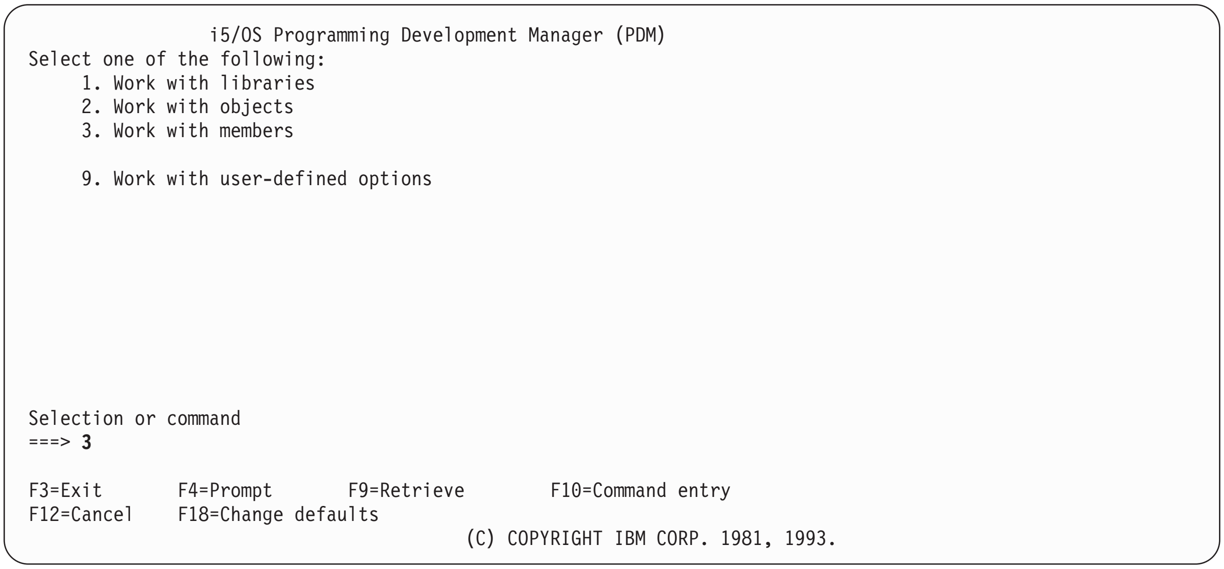
The following table lists the sample names used in the steps:

Table 1. Names Used in Steps for Creating Sample Displays

Name	What It Is
SRCSAM	Sample source file
ADMSAM	Sample source member for application display
DSPSAM	Sample display file
HDMSAM	Sample source member for help display
PNLSAM	Sample panel group
LIBSAM	Sample library that contains source file SRCSAM, display file DSPSAM, and panel group PNLSAM

The steps show only one way to create a sample display with online help information. Other methods are discussed at the end of this chapter.

1. Create the source file SRCSAM using the Create Source Physical File (CRTSRCPF) command; create the library LIBSAM using the Create Library (CRTLIB) command.
2. Enter the Start Programming Development Manager (STRPDM) command to begin using the programming development manager (PDM). When the display appears, select option 3 (Work with members).



```
                 i5/OS Programming Development Manager (PDM)
Select one of the following:
     1. Work with libraries
     2. Work with objects
     3. Work with members

     9. Work with user-defined options










Selection or command
===> 3

F3=Exit       F4=Prompt       F9=Retrieve        F10=Command entry
F12=Cancel    F18=Change defaults
                                         (C) COPYRIGHT IBM CORP. 1981, 1993.
```

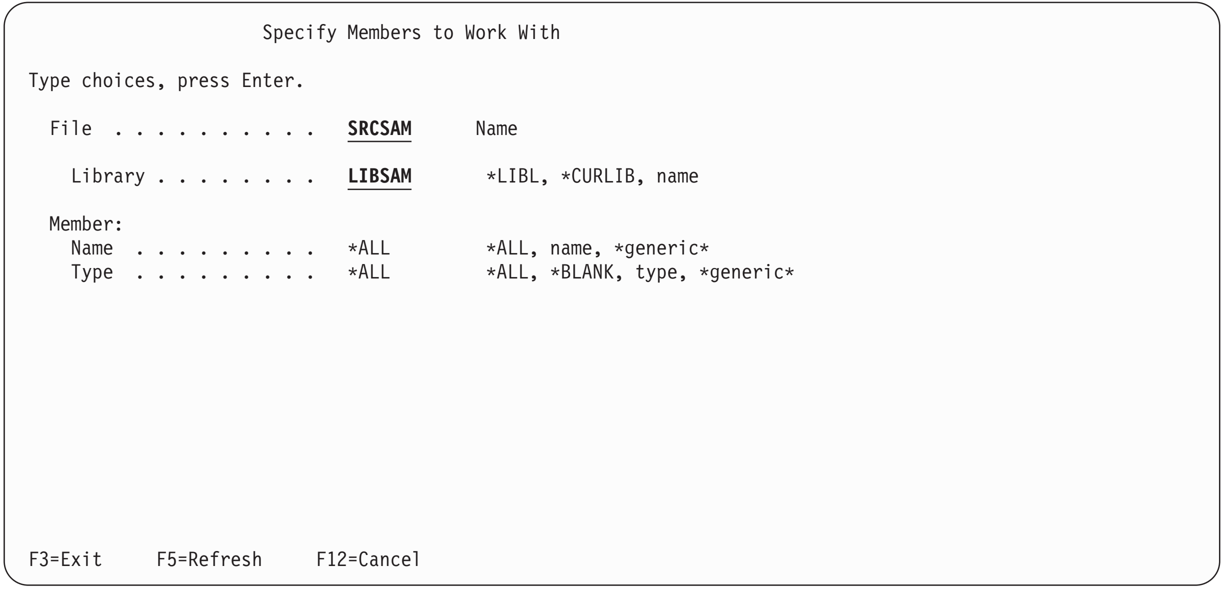
Additional Information

The **programming development manager (PDM)** is the part of the Application Development ToolSet licensed program that allows users to perform several operations (such as copy, delete, and rename) from lists of libraries, objects, and members.

Since the display file (which will be created in step 9) does not actually contain any data, the DDS source for a display file is entered in a source file. A **source file** is an object that is made up of one or more **source members**, which are the different sets of data that make up your DDS source.

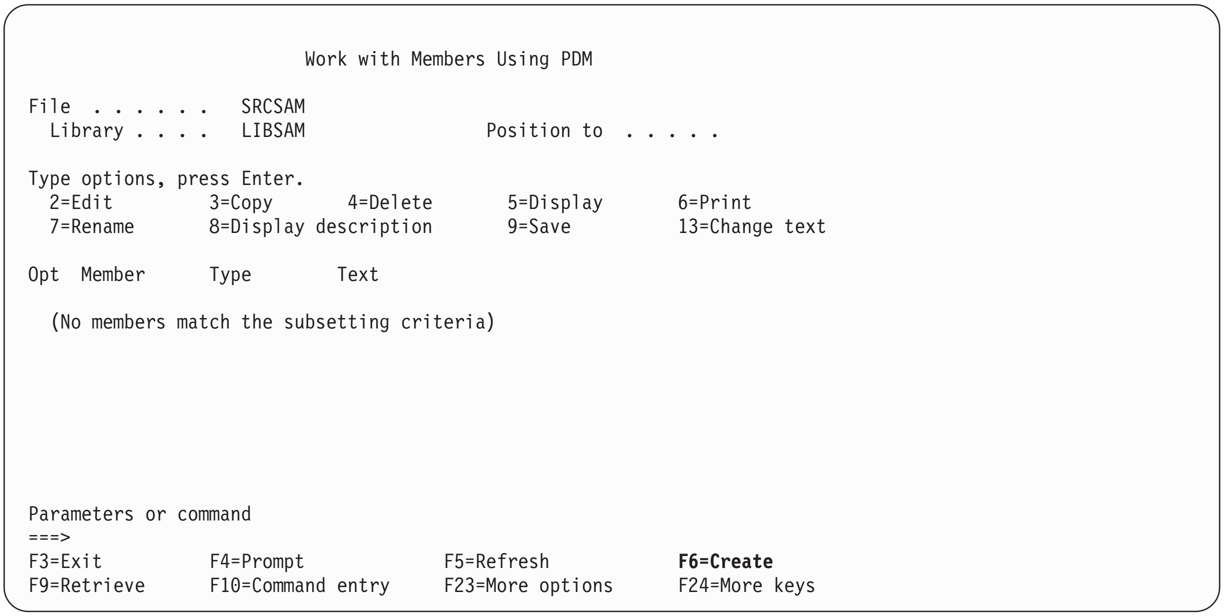
More information about the programming development manager is available in the *ADTS/400: Programming Development Manager* book.

3. When the Specify Members to Work With display appears, complete the file and library information and press the Enter key.



```
                      Specify Members to Work With

Type choices, press Enter.

  File  . . . . . . . . . .   SRCSAM      Name

    Library . . . . . . . .   LIBSAM       *LIBL, *CURLIB, name

  Member:
    Name  . . . . . . . . .   *ALL         *ALL, name, *generic*
    Type  . . . . . . . . .   *ALL         *ALL, *BLANK, type, *generic*











F3=Exit     F5=Refresh     F12=Cancel
```

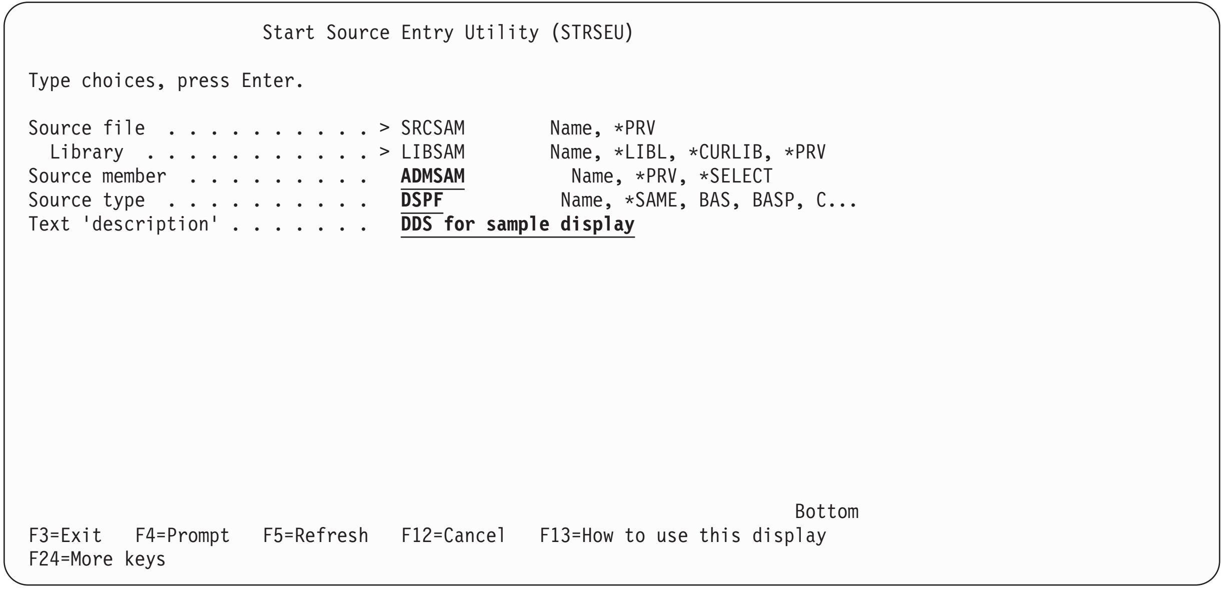
4. The Work with Members Using PDM display appears.



```
                          Work with Members Using PDM

File  . . . . . .   SRCSAM
  Library . . . .   LIBSAM                Position to  . . . . .

Type options, press Enter.
  2=Edit        3=Copy       4=Delete       5=Display       6=Print
  7=Rename      8=Display description       9=Save          13=Change text

Opt  Member      Type        Text

  (No members match the subsetting criteria)









Parameters or command
===>
F3=Exit          F4=Prompt             F5=Refresh           F6=Create
F9=Retrieve      F10=Command entry     F23=More options     F24=More keys
```

Press F6 on this display to create a new member.

5. The Start Source Entry Utility display appears. Complete the information on this display and press the Enter key.

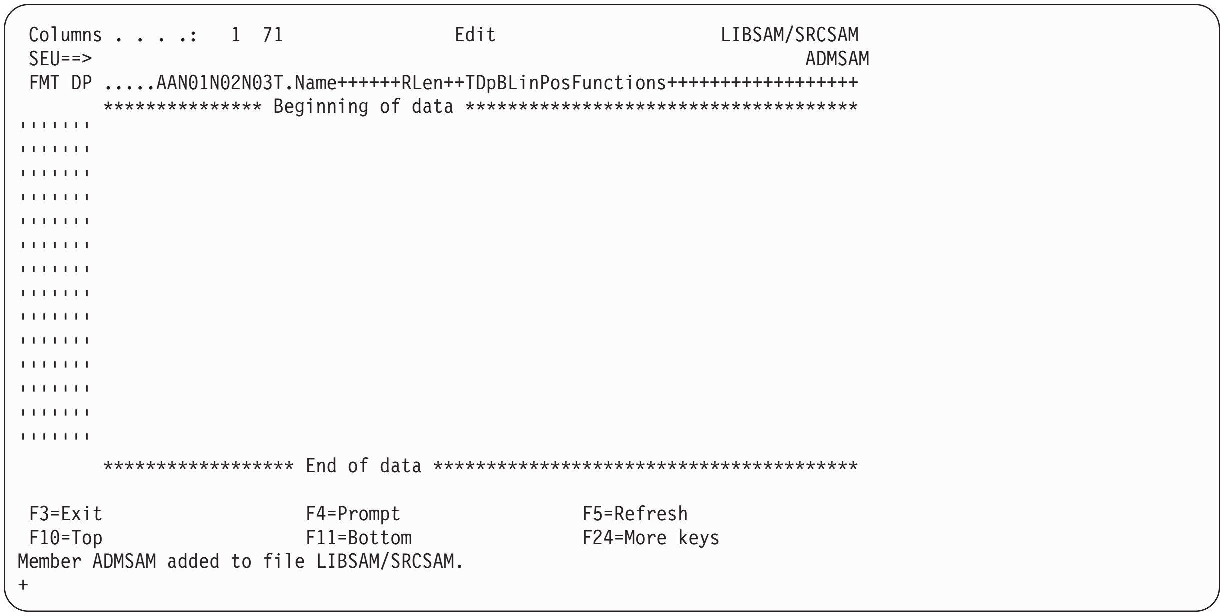


```
                      Start Source Entry Utility (STRSEU)

Type choices, press Enter.

Source file  . . . . . . . . . . . > SRCSAM        Name, *PRV
  Library  . . . . . . . . . . . . > LIBSAM        Name, *LIBL, *CURLIB, *PRV
Source member  . . . . . . . . .    ADMSAM          Name, *PRV, *SELECT
Source type  . . . . . . . . . .    DSPF           Name, *SAME, BAS, BASP, C...
Text 'description' . . . . . . .    DDS for sample display













                                                                      Bottom
F3=Exit   F4=Prompt   F5=Refresh   F12=Cancel   F13=How to use this display
F24=More keys
```

Additional Information

The **source entry utility (SEU)** is a function of the Application Development ToolSet licensed program that is used to create and change source members. More information about the source entry utility is available in the *ADTS for AS/400: Source Entry Utility* book.

6. Because you are creating a new member, the SEU Edit display appears with a screen of blank lines. The text on the last line of the display, *Member ADMSAM added to file SRCSAM* indicates that SEU added the new member to the file you specified.



```
 Columns . . . .:   1  71               Edit                    LIBSAM/SRCSAM
 SEU==>                                                                 ADMSAM
 FMT DP .....AAN01N02N03T.Name++++++RLen++TDpBLinPosFunctions+++++++++++++++++
        *************** Beginning of data ************************************
'''''''
'''''''
'''''''
'''''''
'''''''
'''''''
'''''''
'''''''
'''''''
'''''''
'''''''
'''''''
'''''''
'''''''
        ****************** End of data ***************************************

 F3=Exit                F4=Prompt                F5=Refresh
 F10=Top                F11=Bottom               F24=More keys
Member ADMSAM added to file LIBSAM/SRCSAM.
+
```

7. On the SEU edit display, enter the DDS source statements for the new member. Since the instructions in this chapter allow you to provide online help information for the sample display, make sure your DDS source includes the necessary DDS keywords to enable and access help.

Additional Information

SEU has many functions available to help you enter your DDS. More information about the functions of SEU is available in the the *ADTS for AS/400: Source Entry Utility* book.

For information about describing your display using DDS, see “Defining Display Fields and Functions in a Record Format” on page 21 in this guide. For more information about the DDS keywords needed for online help information, see Chapter 18, “Making Online Help Information Accessible for Your Display File.”

8. When you are finished entering your DDS source, press F3 and complete the information on the Exit display. Press the Enter key.
9. To create the new display file, enter the Create Display File (CRTDSPF) command on any command line. Make sure you specify the source file and member that contains your DDS source:

   ```
   CRTDSPF FILE(LIBSAM/DSPSAM) SRCFILE(LIBSAM/SRCSAM)  SRCMBR(ADMSAM)
   ```

   Press the Enter key.

Additional Information

For more information about creating display files, see “Creating a Display File and Description” on page 18.

10. To add online help information to the sample display, create a second source file member for the UIM source. You are not required to create the second source member in the same source file that you created your DDS source in; however, do not use the same source member for both the DDS and UIM source.

    To create the second source file member, repeat steps 2 through 4. When the Start Source Entry Utility display appears again, continue with step 11.
11. Complete the information on the Start Source Entry Utility display by specifying the following:

For this prompt...
Enter this...

Source member
`HDMSAM`

Source type `PNLGRP`

Text description
`UIM help for sample display`

12. On the SEU edit display, enter the UIM source statements for the new member.

Additional Information

For information about defining your online help information using UIM, see "Defining Online Help Information in a Panel Group" on page 399.

13. When you are finished entering your UIM source, press F3 and complete the information on the Exit display. Press the Enter key.
14. Enter the Create Panel Group (CRTPNLGRP) command on any command line:

```
CRTPNLGRP  PNLGRP(LIBSAM/PNLSAM)  SRCFILE(LIBSAM/SRCSAM)  SRCMBR(HDMSAM)
```

Press the Enter key.

Additional Information

For more information about panel groups, see "Creating and Deleting Panel Groups" on page 410.

Although one library was used in the previous steps, you are not required to create the different objects for your displays in the same library.

Other ways of creating your own application displays on the system follow:

Table 2. More Ways to Create Application Displays

Method	Description	Where to Find More Information
Screen design aid (SDA)	A function of the Application Development ToolSet licensed program that helps you design, create, and maintain displays and menus. SDA allows you to design your displays as you want them to look and then the system creates the DDS source and a display file for you.	*ADTS for AS/400: Screen Design Aid* book.
QUSRTOOL	A library, which is optionally installable on your system, that provides you access to examples of various tools and programming techniques that may help you with application development and management of your system. The display example in QUSRTOOL provides four sample displays with online help information. You can copy the source for these displays into a library of your choosing and then tailor them for your own use.	"Using the Displays Example in the QUSRTOOL Library" on page 415
UIM panel groups	A part of the system that allows you to define panels and dialogs for your application. The UIM controls the panel's appearance and assures consistency with panels developed by IBM®.	Chapter 16, "Introduction to the User Interface Manager," on page 275

Other ways of creating online help information for your displays follow:

Table 3. More Ways to Create Online Help Information

Method	Description	Where to Find More Information
Help records	DDS keywords that allow you to create your online help information in the same or a different member as the DDS source for your application display. This can be accessed from display files and not the UIM application panels.	"Defining Online Help Information in a DDS Record" on page 412 and "Specifying Records in Your Display File" on page 376

To compare and contrast the different ways to create online help information, see "Choosing between Panel Groups and Records for Help" on page 370.

Part 2. Programming Application Displays Using Display Files

Chapter 2. Defining Your Display in a Display File 15
Establishing a Display File 15
Determining File Descriptions 16
Field-Level Descriptions 16
Record-Level Descriptions 16
File-Level Descriptions 17
Deciding Whether to Describe Data Inside or Outside Your Program. 17
Externally Described Data 17
Program-Described Data 17
Creating a Display File and Description 18
Changing the File Description 19
Detecting File Description Changes 20
Defining Display Fields and Functions in a Record Format 21
DDS for Display File 21
Record Format Used by the Program 21
Record Format on the Display 22
Understanding the Field Attribute Characters . . 22
Understanding How Record Format Fields Can Be Used 22
Defining Function Keys 24
Defining Command Attention (CAnn) and Command Function (CFnn) Keys 25
Specifying Alternative Keys 26
Passing Information via Indicators. 27
Removing Option and Response Indicators from the Record Area 27
Enabling Different Response Indicators Simultaneously 28
Setting an Indicator Off 28
Inserting Constant Field Text from a Message Description 28
Allowing for Right-to-Left Cursor Movement . . 28
Defining Cursor Movement to Input-Capable Positions Only 29
Defining Cursor Progression for Entry Fields . . 30
Defining Attributes for Entry Fields 31
Protecting Entry Fields Using Edit Masks . . . 31
Specifying Right-to-Left Display Processing. . . 31
Specifying Word Wrap for Fields 32
Specifying Word Wrap for Fields—Tips . . . 33
Emphasizing Fields. 33
Adding Color. 34
Editing Output Fields 34
Defining Your Own Edit Codes. 35
Specifying Valid Screen Sizes 36
Enabling Your Display to Be Printed 37
Defining Windows 38
Using Program-Described Data 38
Defining Input-Only Files 39
Defining Output-Only Files 39
Defining Input and Output Files 39

Chapter 3. Working with Display Files in an Application 41
Understanding How the System Allocates Resources 41
Opening Display Files 41
Acquiring a Display Station for I/O Operations . . 42
Obtaining Information about Display Files and Display Stations 43
Obtaining Information about Open and I/O Operations 43
Obtaining Attribute Information about Display Stations. 43
Sending and Receiving Data 44
Determining Which Record Formats Are Active on a Display 44
Writing Output to the Display 44
Placing Records on the Display. 45
Understanding Which Records Do Not Occupy Space on the Display 47
Changing Record Formats on a Display . . . 48
Deciding the Order of Record Formats Written to the Display 49
Overlaying and Erasing Record Formats on a Display. 49
Starting Your Record Format on a Specific Line 50
Clearing a Specified Number of Lines . . . 53
Rolling Data between Two Lines on a Display 55
Overriding the Attributes or the Content of a Field. 57
Erasing All Unprotected Input and Output/Input Fields on the Display 61
Resetting Modified Data Tags Associated with Records on the Display 62
Keeping a Record or Field on a Display . . . 62
Deferring the Write Operation Until a Read Request is Made. 64
Specifying Default Values for Fields 64
Indicating Which Mode to Display Records. . 64
Positioning the Cursor after an Output Operation 66
Returning the Cursor Position to an Application 66
Returning the Cursor Position Within a Subfile to an Application 67
Returning the Mode of a Subfile to an Application 67
Initializing Output/Input Fields 67
Inviting Input to the Display 67
Inviting Input from CL Programs 68
Reading Invited Input from the Display 68
Understanding the Read-From-Invited-Devices 69
Reading Input from the Display 70
Unlocking the Keyboard while the Program Is Processing Data 71
Keeping Input Data 72
Setting an Indicator When Data Is Changed 72

Initializing Records and Unlocking the Keyboard-Diagram 72
Specifying Validity-Checking Functions . . . 73
Understanding the Limitations on the Number of Input-Capable Fields 75
Handling Negative Numeric Input Data . . . 75
Understanding How the System Reads Input from the Display 76
Writing Output and Reading Input at the Same Time. 76
Canceling Input That Was Not Waited For . . . 77
Locking the Keyboard and Positioning the Cursor During I/O Operations 77
Saving Previously Displayed Information . . . 78
Understanding the Effects of I/O Operations on Command Keys 80
Avoiding Record Format Problems on the 5250 Display Station 80
Releasing an Acquired Display Station from I/O Operations 83
Closing Display Files 83
Mapping Display Operations to High-Level Language Operations 84
Sharing Display Files in the Same Job 85
Understanding the Open Operation for Files Shared in a Job 85
Understanding the Input/Output Operation for Files Shared in a Job 85
Understanding the Close Operation for Files Shared in a Job 86

Chapter 4. Displaying Groups of Records Using Subfiles 87
Recognizing Subfile Uses 87
Describing Subfiles in Your DDS Source 89
Using a Subfile in a Program 94
Requesting I/O Operations for a Subfile. 96
Requesting I/O Operations for a Subfile Record Format 96
Adding a Record at a Specified Location in a Subfile 96
Updating an Active Record in the Subfile . . 96
Reading an Active Record at a Specified Location in the Subfile. 97
Reading the Next Changed Record in a Subfile 97
Requesting I/O Operations for a Subfile Control Record Format 98
Displaying Subfile Records 98
Placing Subfile Records on the Display for Processing 98
Displaying and Processing Subfile Records at the Same Time 98
Recognizing Subfile I/O Requests in High-Level Languages. 98
Controlling the Appearance of Subfiles 99
Displaying Horizontal Subfiles with Display Modes 100
Specifying Subfile Size Equal to Page Size 102
Specifying Subfile Size Not Equal to Page Size . . 103
Checking Validity on Subfile Data 104
Displaying Error Messages from Subfiles 105
Positioning the Cursor on the Displayed Subfile 105
Positioning the Cursor Initially 106
Positioning the Cursor When a Roll Key Is Used 106
Positioning the Cursor When a Fold or Truncate Key Is Used 109
Positioning the Cursor and Rolling When Two or More Records Are Displayed 109
Understanding Subfile DDS and Program Logic-Example 113

Chapter 5. Defining Windows with Display Files 115
Window Terminology. 115
DDS Window Keywords. 116
Window Representation and Hardware Configuration 116
Creating Windows. 116
Window Definition Records 117
Window Reference Records. 117
Window Size and Location 118
Cursor Position. 119
Error Messages 119
Subfiles 120
DDS Help Records 120
Defining Window Borders 120
Border Defaults. 121
Multiple Border Definitions 121
UIM Help Window Borders 122
Defining a Window Title 122
DDS for a Window Title-Example 123
Reading Data from Windows 124
Changing Window Borders and Contents 124
Moving and Duplicating Windows 124
Making Two Windows Seem Active at Once . . . 124
Making One Window in a Series Stand Out . . . 125
Removing Windows 126
Removing All Windows 126
Removing More Recent Windows 126
Improving Application Performance. 126
System Save and Restore Operations 126
Response Time 126
Bypassing System Save and Restore Operations 127
USRRSTDSP Keyword Processing and Interactions 127
Programming Examples 128
Using Basic Window Functions 128
DDS Full-Screen Display and Window Definitions 128
RPG Display Program 130
Step 1: Display Initial Display 132
Step 2: Display Window 1 132
Step 3: Display Window 2 132
Step 4: Restore Window 1 133
Step 5: Display Initial Display 133
Defining Windows in a Separate Display File 134
DDS Full-Screen Display and Window Definitions 134
RPG Program Source 134
RPG Program Source for WINPGM 135
Step 1: Display Initial Display 135
Step 2: Display a Window 136
Step 3: Return to the Initial Display 136

Chapter 6. Creating a Graphical Look for Displays 139
Factors Affecting the Graphical Look 139
Hardware Configuration 139
Enhanced Display Parameter 141
DDS Keywords 142
Creating Menu Bars 144
Defining the Menu-Bar Choices 145
Suppressing the Menu-Bar Separator 145
Defining the Menu-Bar Separator 145
Selection Fields-Overview 147
DDS for Selection Fields-Example 148
Creating a Vertical Single-Choice Selection Field 149
Creating a Vertical Multiple-Choice Selection Field 150
Creating a Horizontal Selection Field 150
Cursor Movement in a Vertical Selection Field 151
Cursor Movement in a Horizontal Selection Field 151
Controlling the Selection Indicators in a Selection Field 152
Creating Pull-Down Menus Using Single-Choice Selection Fields 154
Controlling the Selection Indicators in a Pull-Down Menu 155
Defining Accelerator Keys 156
Defining a Menu-Bar Switch Key 157
Defining a Cancel Key 158
Limiting Function When Cursor is Outside a Pull-Down Menu 158
Selection Lists-Overview 159
DDS for Selection Lists-Example 160
Creating Selection Lists 161
Controlling the Selection Indicators in a Selection List 162
Scroll Bars-Overview 163
Creating a Scroll Bar 164
DDS for Scroll Bars-Example 165
Scroll Bar Operation 166
Push Buttons-Overview 166
DDS for Push Buttons-Example 167
Creating Push Buttons 167
Controlling the Availability of Choices 168
Auto-Selection in Single-Choice Selection Fields 169
Auto-Enter in Single-Choice Selection Fields . . . 169
Defining Mnemonics 170
Defining Choice Colors and Attributes 171
Continued-Entry Fields-Overview 174
Specifying Word Wrap on Continued-Entry Fields 175
DBCS Considerations with Continued-Entry Fields 175
How DBCS Data is Returned for Continued-Entry Fields 175
Keyboard Functions with Continued-Entry Fields 176
Character data 176
Field Mark 177
Automatic Shape Determination (ASD) Processing 177
Delete 177
Erase EOF 177
Erase Input 177
Reverse 177
Close 177
Field Exit 178
Field Plus 178
Field Minus 178
Dup 178
Kanji 178
Character Backspace 178
Character Advance 179
New Line 179
Field Advance 179
Field Backspace 179
Forward Field-Exit Processing 179
Backward Field-Exit Processing 180
How the Menu Bar Interacts with the Application 180
Defining the MNUBARDSP Keyword on the Application Record 180
Defining the MNUBARDSP Keyword on the Menu-Bar Record 181
Receiving Input from the Pull-Down Menus . . 182
Receiving Input from Pull-Down Menus Using the Pull-Down Input Parameter . . . 182
Removing a Pull-Down Menu after Receiving Input 184
Updating a Pull-Down Menu before Displaying 184
Defining Application Help 185
Defining Choice-Level Help 185
Defining Help for a Field 187
Key Interaction for Menu Bars and Pull-Down Menus. 188
Cursor Movement 189
Pressing the Tab Key 189
Pressing the Cursor Keys 190
Programming Examples 190
Using the MNUBARDSP Keyword on the Application Record 190
Description 192
Using the MNUBARDSP Keyword on the Menu-Bar Record 193
Description 194
How the Displays Look 194
Simple Hotspots 195
Command Key Emulation 196
Page Up and Page Down Key Emulation . . . 196
Programmable Mouse Buttons-Overview 196
Pointer Device Events 197
AID Codes to be Returned 197
Programmable Mouse Buttons-Benefits 198
Programmable Mouse Buttons Operation . . . 198
Programmable Mouse Buttons-NWS Considerations 199
Programmable Mouse Buttons-Event Processing States 199
Programmable Mouse Buttons-Event Processing Priority 200
Unshifted Left Button Pressed Event Processing 200
Unshifted Left Button Released Event Processing 202

Unshifted Left Button Double Click Event Processing 202
Shifted Left Button Pressed Event Processing 202
Shifted Left Button Released Event Processing 203
Shifted Right Button Pressed Event Processing 204
Any Other Pointer Device Event Processing 205
Grid Line Structures-Overview 205
DDS for Grid Line Structures-Example 205
Grid Line Structures and Windows 206
Hardware Requirements for Grid Line Structures 207
Inserting HTML Tags 207
Resolving HTML Field Overlap 208
Programming Examples 209

Chapter 7. Overriding Display Files and Display File Attributes. 213
Determining Whether or Not to Use Overrides . . 213
Overriding File Attributes in HLL Programs . . . 213
Example 213
Overriding File Names in HLL Programs 214
Example 214
Overriding Both File Names and Attributes in HLL Programs. 215
Example 215
Applying Overrides When Compiling a Program 216
Example 216
Deleting Overrides 217
Displaying Overrides. 217
Using File Redirection to Override File Names and Libraries or File Types 217
Overriding Files with the Same File Types. . . 217
Overriding Files with Different File Types . . . 218
Recognizing Commands That Ignore or Restrict Overrides 220

Chapter 8. Handling Messages and Errors for Display Files 223
Creating and Displaying Your Own Messages . . 223
Displaying a Message on the Message Line . . 224
Displaying a Message on the Message Line When a Subfile Control Record is Written . . . 224
Displaying a Message on the Message Line Using a Message Field 224
Priorities for Displaying Messages on a Message Line 224
Displaying Messages in a Field on the Display 225
Displaying Messages on a Program Message Queue. 225
Displaying Error Messages through a Subfile 225
Sounding an Alarm for Messages. 228
Automatically Handling Permanent I/O Errors on Display Stations 228
Analyzing Error Messages Sent from the System 229
Understanding Messages and Message Monitors 229
Understanding Major/Minor Return Codes . . 230
Recovering from Errors 231
Normal Completion 231
Completion with Exceptions 231
Permanent System or File Error 232
Permanent Device or Session Error on I/O Operation 232
Device or Session Error on Open or Acquire Operation 233
Recoverable Device or Session Errors on I/O Operation 233

Chapter 9. Creating and Accessing Menus Using Display Files 235
Running System and User-Defined Menus. . . . 235
Returning to a Menu after Running the GO command 235
Determining the Previous Menu 235
Using the Cancel and Exit Keys on Menus . . 235
Choosing the Menu That Is Shown at Sign-On Time 237
Defining Your Own Display File Menus 237
Understanding DDS and Display File Considerations for Menus 238
Describing Menu Actions in a Message File . . 239
Naming Help Formats for Menus 239
Building a Display File Menu 240
Describing the Menu and Menu Help Information 241
Creating the Display File 242
Creating the Message File 243
Adding Messages to the Message File . . . 243
Creating the Menu Object 243
Running the Menu 243
Defining Your Own Program Menus. 244
Passing Parameters for Program Menus . . . 244
Building a Program Menu 244
Describing the Menu 245
Creating the Display File 245
Entering the Source and Creating a CL Program 245
Creating the Menu 246
Running the Menu 246
Exiting from a Program Menu without Returning to the Previous Menu 246
Program 1 247
Program 2 247
Avoiding Menu Name Conflict 248
Naming Your Menus 248
Placing Your Menu in a Higher Library in the Library List 248
Specifying the Library That Contains the Menu 248
Using the Generic Menu Specification 248
Changing the Command Default after Duplicating a Command 249
Displaying Menu Attributes 249
Changing Menu Attributes 249
Deleting Menus 249

Chapter 10. Using User-Defined Data Streams 251
Understanding Display Station Differences . . . 251
Understanding User-Defined Data Stream Limitations 252

Chapter 11. Passing Data between Programs 255

Passing Data in the Same Routing Step in a Job 255
Passing Data between Routing Steps in a Job . . . 256

Chapter 12. Waiting for Input from a Display File, an ICF File, and a Data Queue 259

Chapter 13. Using Alternative Character Sets and Code Pages 261
System Has Characters Not Normally Displayed on the Device 261
Device Passes Characters Not Displayed on the System 262
Specifying Character Translation for Fields . . . 262
Determining the Character Identifier (CHRID) Value for Your Display 264

Chapter 14. Improving System Performance with Displays 267
Deferring the Write Operation for a Display File 267
Designating the Primary Screen Size for a Display File 267
Writing Only One Page of Subfile Records at a Time 267
Sharing an Open Data Path (ODP) for the Same Job 268
Sending Records with Input Fields to the Display in Order 268
Overlapping and Not Deleting Repeatedly Sent Records 268
Restoring the Display 268
Defining Command Attention Keys Rather Than Command Function Keys 269
Using the Invite Operation 269
Using Windows 269

Chapter 2. Defining Your Display in a Display File

A display file defines the format of the information to be presented on a display station, and how that information is processed by the system on its way to and from the display station. Data description specifications (DDS) describe the data referred to by a display file.

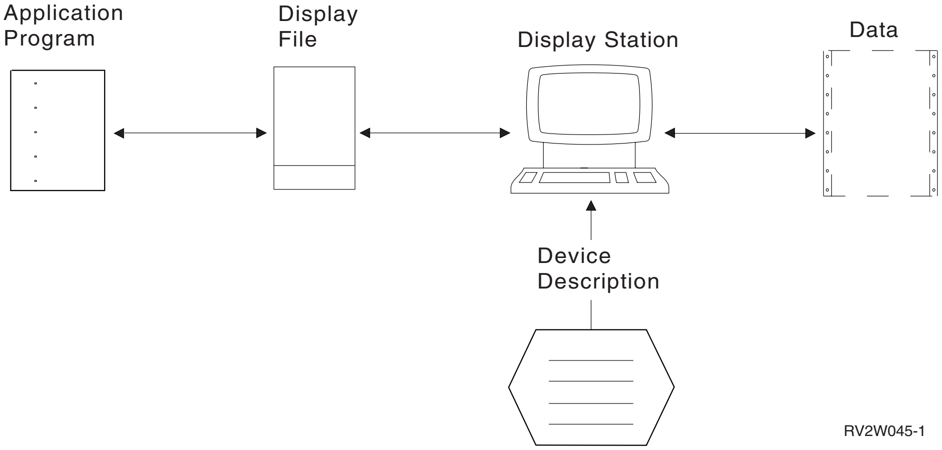
This chapter tells you about display files, including how to create them and how to provide DDS source for them to describe your display.

Establishing a Display File

A display file is an object on the system. An **object** is a named storage space that consists of a set of characteristics that describe itself and, in the case of a display file, the data. Like other objects on the system, display files have the following characteristics:

- A display file is named and placed in a library when it is created. The file name and library name allow you to refer to the display file in your applications.
- Once a display file is created, it can be changed, secured, saved, restored, or deleted.

Before an application program can work with a display station, a display file must be opened to allow data to flow between the program and the display station.



A **device description**, which is a system object that describes the display station to the system, must also exist for the display station. A device description contains information such as device address, device type, model number, and features. Device descriptions are usually created by system personnel or, for locally attached devices, can be created during the automatic configuration of the system.

A program may work with more than one display station at a time by doing one of the following:

- Opening more than one display file
- Opening a display file that allows more than one display station to be attached to an open file

Since a display file does not have a set of data uniquely associated with it, the relationship between the data and the display file is established when the display file is opened and ends when the display file is closed.

Determining File Descriptions

The **file description**, which is created at the same time the display file is created, describes the characteristics of the display file and determines how the display file does the following:

- Controls the display station
- Formats output data from the program for presentation at the display station
- Formats input data from the display station for presentation to the program

A file description determines how a program is able to use the file. If a program attempts to perform an operation that is inconsistent with the display file description, the system does not allow the operation.

The file description is created and deleted at the same time as the display file it describes. Some parts of a file description may be changed, either permanently with the Change Display File (CHGDSPF) command or temporarily with the Override with Display File (OVRDSPF) command.

A file description describes data at three different levels:

- Field level
- Record level
- File level

The following sections describe these levels.

Field-Level Descriptions

A **field** is the smallest unit of data that is recognized and handled by the data management support of the system. A **field-level description** allows you to give the system detailed characteristics of a field, such as:

- Where on the screen the field is to appear
- What type of data is valid for the field
- Whether the field should be highlighted in some way
- How it will be presented from the program to the system on output and from the system to the program on input.
- Where each field is relative to the start of a record
- What the characteristics of each field will be while in the system
- Where the data for each field should be acquired from for output
- Where and how input from the display station should be placed so the program can use it
- Whether the field is an input-capable field or output-capable field only

Only field-level descriptions can determine that valid data is specified for individual fields on a display.

Record-Level Descriptions

A **record** is an ordered set of one or more fields. A **record-level description** allows you to tell the system what a particular record looks like, or its **record format**.

A record-level description is given in one of two ways:

- *If field-level descriptions are also used,* you identify what fields make up the record format and the order of these fields within the record format. The system can then perform separate operations on each field described with a field-level description in the record-level description. For example, one field can be highlighted while another is not.
- *If field-level descriptions are not used,* the record format is given by specifying the length of the record. The system handles the entire record as a unit and cannot perform operations on one part of the record one way and another part a different way.

Since records are used to transfer data between the system and the application program, a record-level description is required for display files.

File-Level Descriptions

A **file** is an organized set of zero or more records (a file with zero records is empty). A **file-level description** is a description that applies to the file as a whole. For a display file, you can specify the following in the file-level description:

- What record formats are valid for the display file
- What display station should be usable with the display file
- What graphic character set is to be assumed for the data that will be entered through the display file

Deciding Whether to Describe Data Inside or Outside Your Program

When the detailed description of the display file and the data it refers to is contained in a display file rather than imbedded in a program, the data is called **externally described data**. When the data is described within the source program, the data is called **program described data**.

Externally Described Data

Externally described data exists independently of any program that uses the file. Using externally described data, you can produce a detailed and standard description of both the display file and any data that can be processed through the display file.

To use externally described data, you need to declare that the display file is to be used as an externally described file. The language compiler or interpreter extracts the file description from the display and then incorporates it into the program.

There are several advantages to using externally described data:

- *Increased programmer productivity.* The language automatically describes the record layouts for you without additional coding. You need to describe records and fields only once (when the file is created). You can then refer to these fields within the program.
- *Ease of file and program maintenance.* When fields are added, deleted, or changed, it can be done in one place instead of maintaining the record layout in each program that uses the file.
- *Increased data integrity.* Since the fields and records are described in one central location, there is less chance of programming errors when describing the data in the file to the program. All application programs using the file will have the same view of the data. Moreover, the system view of the data becomes the same as the application program view.
- *Level checking provided.* Level checking is an automatic method used when the program is run that determines if the file description has changed since the program was last compiled. Depending on the type of change, the program may only need to be recompiled without modification. This allows better control over program maintenance. There is more information on the level-checking function in “Detecting File Description Changes” on page 20.

Program-Described Data

You are not required to use external descriptions to describe your displays in your program. If you do not use externally described data, you must declare variables in your source program that define to the compiler or interpreter what the data looks like.

When program-described data is used, the program and the system may not have the same view of the data:

- If the file does not have any field-level descriptions, the system must operate at the record level. The only concern in this case is that the record length the program is using is the same as what the system is using. It need not be, but the system always operates with the record length it has. If the record length that the system is using differs from the record length the program is using, the system truncates or pads as necessary.

- If the file has field-level descriptions but the program does not use them, the system uses the field-level descriptions even though the program does not. The system expects the program to present data according to the file description and, conversely, provides data to the program according to the description.

More information about program-described data is found in "Using Program-Described Data" on page 38.

Creating a Display File and Description

Display files are created using the Create Display File (CRTDSPF) command.

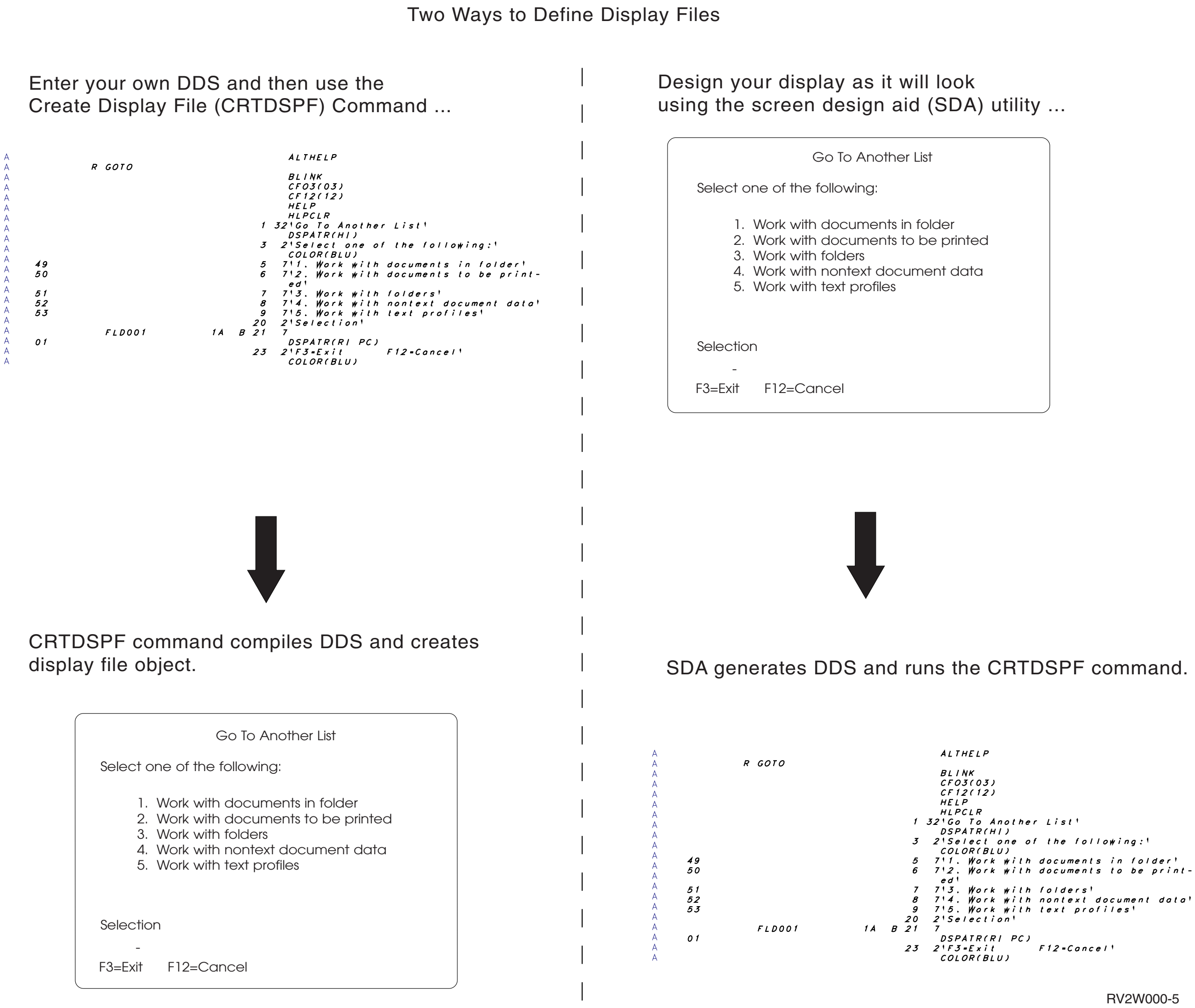
You can define the DDS for your display file in one of two ways:

- Using the screen design aid (SDA) utility
- Entering your own DDS source

You can specify certain attributes about your display file. Information about these attributes is found in the following:

- "Deferring the Write Operation Until a Read Request is Made" on page 64
- "Saving Previously Displayed Information" on page 78
- **CL** topic in the iSeries Information Center
- *iSeries Security Reference*

The following illustration compares the two ways to create display files:



Two Ways to Define Display Files

Enter your own DDS and then use the Create Display File (CRTDSPF) Command ...

```
A                                           ALTHELP
A          R GOTO
A                                           BLINK
A                                           CF03(03)
A                                           CF12(12)
A                                           HELP
A                                           HLPCLR
A                                       1 32'Go To Another List'
A                                           DSPATR(HI)
A                                       3  2'Select one of the following:'
A                                           COLOR(BLU)
A  49                                   5  7'1. Work with documents in folder'
A  50                                   6  7'2. Work with documents to be print-
A                                           ed'
A  51                                   7  7'3. Work with folders'
A  52                                   8  7'4. Work with nontext document data'
A  53                                   9  7'5. Work with text profiles'
A                                      20  2'Selection'
A            FLD001             1A   B 21  7
A  01                                       DSPATR(RI PC)
A                                      23  2'F3=Exit      F12=Cancel'
A                                           COLOR(BLU)
```

CRTDSPF command compiles DDS and creates display file object.

```
                  Go To Another List

Select one of the following:

      1. Work with documents in folder
      2. Work with documents to be printed
      3. Work with folders
      4. Work with nontext document data
      5. Work with text profiles


Selection
      -
F3=Exit     F12=Cancel
```

Design your display as it will look using the screen design aid (SDA) utility ...

```
                  Go To Another List

Select one of the following:

      1. Work with documents in folder
      2. Work with documents to be printed
      3. Work with folders
      4. Work with nontext document data
      5. Work with text profiles


Selection
      -
F3=Exit     F12=Cancel
```

SDA generates DDS and runs the CRTDSPF command.

```
A                                           ALTHELP
A          R GOTO
A                                           BLINK
A                                           CF03(03)
A                                           CF12(12)
A                                           HELP
A                                           HLPCLR
A                                       1 32'Go To Another List'
A                                           DSPATR(HI)
A                                       3  2'Select one of the following:'
A                                           COLOR(BLU)
A  49                                   5  7'1. Work with documents in folder'
A  50                                   6  7'2. Work with documents to be print-
A                                           ed'
A  51                                   7  7'3. Work with folders'
A  52                                   8  7'4. Work with nontext document data'
A  53                                   9  7'5. Work with text profiles'
A                                      20  2'Selection'
A            FLD001             1A   B 21  7
A  01                                       DSPATR(RI PC)
A                                      23  2'F3=Exit      F12=Cancel'
A                                           COLOR(BLU)
```

RV2W000-5

You may also combine the two methods, creating an initial display using SDA and then tailoring the generated DDS.

Changing the File Description

After a display file has been created, the file description can be changed:

- To change the file description that was originally specified on the CRTDSPF command, use the Change Display File (CHGDSPF) command.
- To change the file-level, record-level, or field-level information contained in the DDS source, you must first update the DDS source and then create the display file again using the CRTDSPF command. A new display file can be created without deleting the existing display file by specifying REPLACE(YES) on the CRTDSPF command.
- To change both the CL command file-level descriptions and the DDS source, specify the new values when you create the display file again.

Changes to display file descriptions are applied according to the following:

- If the file-level description was changed with a CL command, any program that uses the file will automatically use those new descriptions.
- If the DDS descriptions were changed and the program uses the file as a program-described file, the system uses the new file-level description. However, if the DDS descriptions were changed and the

program uses the file as an externally described file, then the record-level and field-level descriptions used when the program was compiled may not match the changed file. If the system detects a mismatch when the program opens the file, an error occurs. See "Detecting File Description Changes."

You may also temporarily change a file-level description when a display file is opened. More information about these temporary changes is found in Chapter 7, "Overriding Display Files and Display File Attributes," on page 213.

Detecting File Description Changes

When a program that uses externally described files is compiled, the high-level language compiler extracts the record-level and field-level descriptions for the files referred to in the program and makes those descriptions part of the compiled program. When you run the program, you can verify that the descriptions with which the program was compiled are the current descriptions.

The system assigns a unique **level identifier** for each record format when the file it is associated with is created. The system uses the following information to determine the level identifier:

- Record format name
- Field name
- Total length of the record format
- Number of fields in the record format
- Field attributes (for example, length and decimal positions)
- Order of the fields in the record format

Display files may also use the number of and order of special fields called indicators to determine the level identifier.

If you change the DDS for a record format and change any of the items in the preceding list, the level identifier changes.

To check for changes in the level identifiers when you run your program, specify *YES for the LVLCHK parameter on the CRTDSPF or CHGDSPF command. When the display file is opened, the level identifiers of the display file and the file description that is part of the compiled program are compared format-by-format. If the identifiers differ or if any of the formats specified in the program do not exist in the file, a message is sent to the program to identify the condition.

If the identifiers differ, either the formats have been changed or your program does not use the changed formats. If the changed format does affect your program, you may decide to do the following:

- Compile the program again so that the changes are included
- Determine if the changes affect your program before deciding what action to take.

To check the changes to the record format, run one of the following commands:

- Display File Field Description (DSPFFD) command to display the record-level and field-level descriptions
- Start Source Entry Utility (STRSEU) command to display the source file containing the DDS for the file
- Display File Description (DSPFD) or the DSPFFD command to display the format level identifier defined in the file
- Display Program References (DSPPGMREF) command to display the format level identifier that was used when the program was created

Defining Display Fields and Functions in a Record Format

A record format in a display file describes both the format of the record as it is used in the application program and the format of the record when it is displayed (see Figure 2 on page 22).

A record format contains field descriptions, which are defined using data description specifications (DDS). For each field in a record format, you describe the following:

- Location of the field on the display
- Length of the field
- Type of data contained in the field (character, zoned decimal, or floating point)
- Field type (output, input, or output/input).

Information about DDS keywords

This section describes how DDS keywords are used to describe the information on your display. For more information about specific DDS keywords, see the **DDS** topic in the iSeries Information Center.

DDS for Display File

The following source shows the DDS for a sample display file:

```
|...+....1....+....2....+....3....+....:833.+....5....+....6....+....7...
     A          R RECORD
     A                                  3  2'Customer Number:'
     A            CUST           5  0   3 20
     A                                  3 27'Customer Name:'
     A            NAME          20      3 44
     A                                  4 27'Address:'
     A            ADDR          20      4 44
     A            CITY          20      5 44
     A            STATE          2      5 66
     A            ZIP            5  0   5 70
```

Figure 1. Sample DDS Source for a Display File

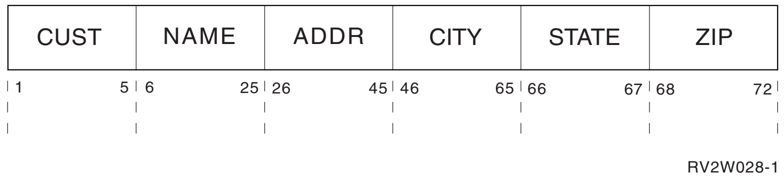
Table 4 shows the column positions and descriptions for the DDS specifications.

Table 4. Column positions for sample DDS

Column	Definition	Starting Position
17	Type of name	17
19 - 28	Field name	19
30 - 34	Length	34
36 - 37	Decimal positions	37
39 - 41	Line location	41
42 - 44	Position location	44
45 - 80	Function	45

Record Format Used by the Program

The program passes the fields in the record in the same order that you described them in the DDS source.



Record Format on the Display

The fields are displayed according to the display positions you assigned them in the DDS source.

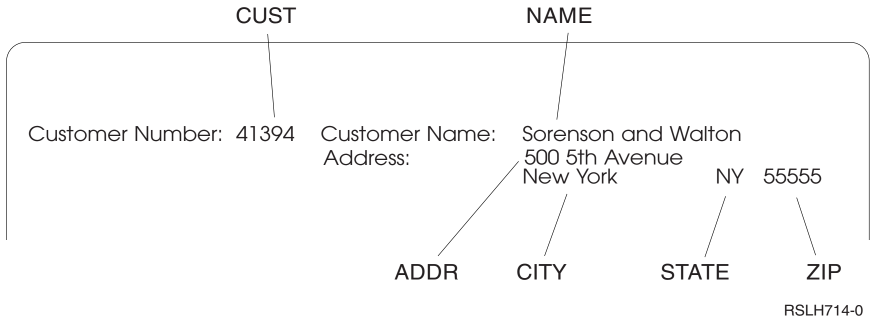


Figure 2. Record Formats in the Program and on the Display

Understanding the Field Attribute Characters

Each field displayed has a **beginning attribute character** and an **ending attribute character** associated with it that define the displayed field. The beginning character precedes the first character of a field and is displayed as a blank. The ending attribute character follows the last character of a field and is also displayed as a blank. For example, if you specify a field for positions 2 through 8, the beginning attribute character is in position 1 and the ending attribute character is in position 9. These characters are not included in the field length you specify in DDS. A beginning attribute character can overlap an ending attribute character; that is, they can occupy the same position on the display. However, nothing else can overlap the beginning attribute character. Therefore, when you design a display, you must allow one space for the beginning attribute character of each field. You can use the blank attribute character to space between fields when they are displayed.

If field-level descriptions are not used, the entire record is treated as a field with a beginning attribute character and an ending attribute character.

When a record is displayed so that the last field in the record ends in the last position on the line, the ending attribute character for that field is in the first position of the next line. The beginning attribute character of the first field in the next record can be superimposed on the ending attribute character. For example, if the ending attribute character for the last field in record 1 is in position 1 of line 5, the beginning attribute character for record 2 can also be in position 1 of line 5. In this case, the first record is not considered to be overlapped. However, if the first field in a record begins in position 1, which means that the beginning attribute character is in the last position of the preceding line, the previous record is overlapped and is cleared from the display.

To see the locations of fields in the input records and output records used by the program, see the printed DDS output produced when you created your display file using the CRTDSPF command.

Understanding How Record Format Fields Can Be Used

The fields you describe in the record format can be used in the following ways:

Note: To see the location of positions on a DDS form, see Table 4 on page 21.

- **Input fields** are fields that are passed from the display station to the program when the program reads a record. Input fields can be initialized with a default value (specified in the record format for the display file). If the user does not change the field and the field is selected for input, the default value is passed to the program. Input fields that are not initialized are displayed as blanks into which the user can enter data. By default, input fields are underlined on the display.

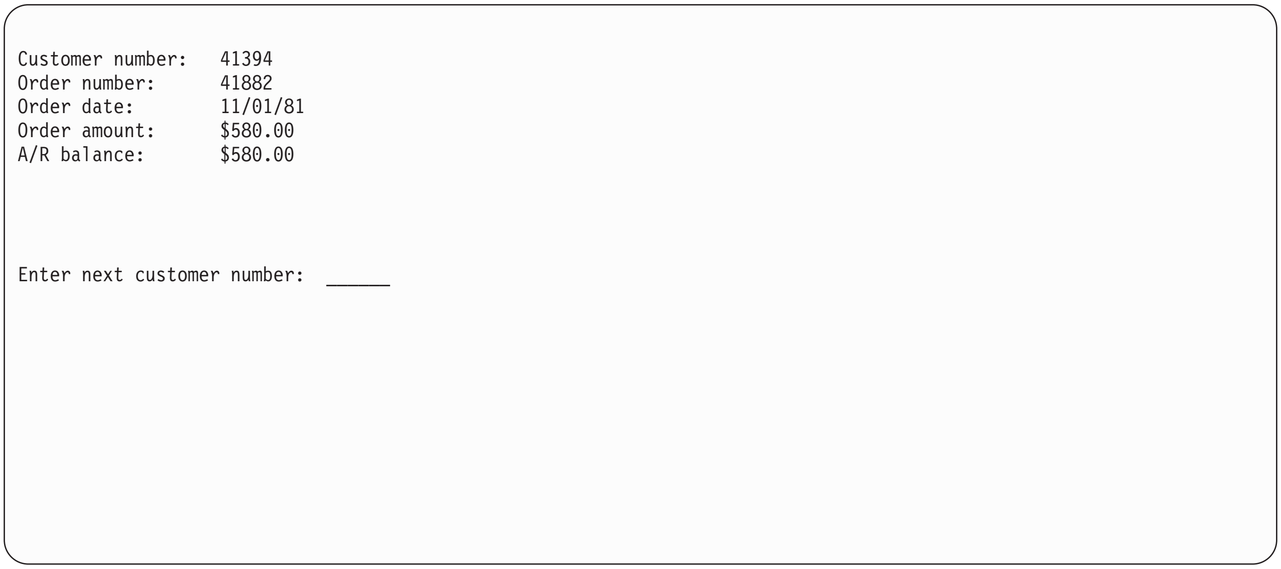
  Note: Trailing blanks on input fields are replaced by null and not blank characters; therefore, the Insert key can be used to insert characters in all input fields that end in blanks.
- **Output fields** are fields that are passed from the program to the display station when the program writes a record to a display. Output fields contain data provided by the program, not by the user. To specify an initial value for a named output field, see "Specifying Default Values for Fields" on page 64.

  In the case of **subfiles**, which are special records used to display lists of information, output fields are returned to the program as if they were output/input fields.
- **Output/input fields** are fields that are passed from the program when the program writes a record to a display and are passed to the program when the program reads a record from the display and the field is selected for input. By default, these fields are underlined on the display. Output/input fields are usually used when the program displays data that can be changed by a user. To specify an initial value for a named output field, see "Specifying Default Values for Fields" on page 64.
- **Hidden fields** are fields that are passed from and to the program but are not sent to the display. Hidden fields are useful in applications involving subfiles. For example, a subfile record can contain record key information in a hidden field. The hidden field cannot be seen by the user, but is returned to the program with the subfile record so that the program can return the record to the database.
- **Constant fields** are fields that are passed to the display but are unknown to the program. These fields are unnamed and have their constant values defined in the DDS for the file. DATE, TIME, and MSGCON are examples of keywords that are allowed only on constant fields and whose constant values are determined during program run time (DATE and TIME) or DDS compile time (MSGCON).
- **Message lines** are output fields that are treated as messages.
- **Program-to-system fields** are output-only fields that are named, numeric or alphanumeric. They are used to communicate between an application program and the system. Program-to-system fields do not appear on the display. That is, your program can place data in these fields and the system will use that data to control its processing on an output operation, but the user cannot see the contents of these fields.

A field is input-capable if it is an input field or an output/input field. Each input-capable field has a special attribute called a **modified data tag (MDT)**. The MDT is set on by the display station when any data is typed into the field. It can also be set on and cleared by the application program.

The maximum number of fields that you can specify for each record format is 32 763. (See "Understanding the Limitations on the Number of Input-Capable Fields" on page 75 for information on the number of input-capable fields that can be specified.) The maximum combined length for all fields and indicators in a record format is 32 763.

The following display shows output fields and input fields displayed in response to a request (in the form of entering a customer number in an input field) from a user.



```
Customer number:   41394
Order number:      41882
Order date:        11/01/81
Order amount:      $580.00
A/R balance:       $580.00




Enter next customer number:  ______
```

The prompts, *Customer number:*, *Order number:*, *Order date:*, *Order amount:*, *A/R balance:*, and *Enter next customer number:* are constants. The data associated with these fields (41394, 41882, 11/01/81, $580.00, and $580.00) is displayed in output fields. The data is passed from the application program to the system, and the system displays it. The field following the constant *Enter next customer number:* is an input field. The user must enter data into this field (the cursor is positioned at the beginning of the input field). Input fields are underlined by default. Editing of the field is normally defined within DDS.

You must specify the location for each field except when the field is a hidden field, a message line, or a program-to-system field, or when the field is in a subfile message record format. You cannot specify line 1, position 1 for location, except when you define a record that can start in any line.

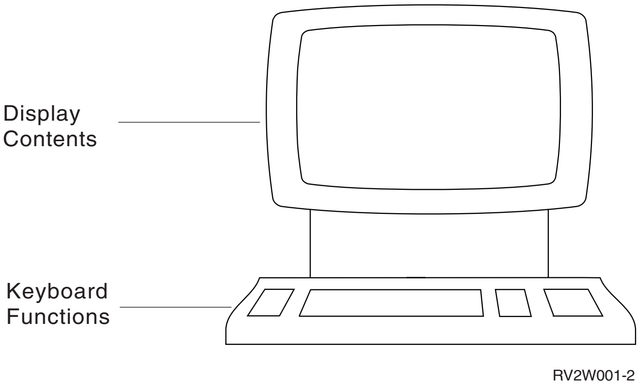
The maximum length of a character field or numeric (zoned decimal) field is the number of positions remaining (relative to the start location of the field) on the display minus 1. Another restriction of the numeric (zoned decimal) field is that it can be no longer than 31, even if more than 31 positions are remaining on the display.

Specifications for the fields you describe can be retrieved from a previously described field. The previously described field can be either in a database file or already defined in the DDS source for the display file. When you use field-level descriptions from a database file, binary and packed decimal fields are changed to zoned decimal fields. These fields that you use to define other fields are called **reference fields**.

You can define two fields to occupy the same positions on the display, and use option indicators to select which of the overlapping fields is to be displayed. If more than one overlapping field is selected on the same output operation, only the first field selected is displayed.

Defining Function Keys

To write an application using a display station, you have to control both the functions of the keys at the keyboard, and the contents of the display.



The Enter key can always be used by the user. So that the user can use the other function keys, you must specify the following DDS keywords to enable the corresponding function key:

- CAnn, where nn is 1-24
- CFnn, where nn is 1-24
- CLEAR
- HELP (not required if you only need the Help key to retrieve the message help on the display)
- HOME
- PRINT
- ROLLDOWN or PAGEUP (not required to be able to roll a subfile when the subfile page is not equal to the subfile size)
- ROLLUP or PAGEDOWN (not required to be able to roll a subfile when the subfile page is not equal to the subfile size)
- MOUBTN (Programmable Mouse Button) allows attention identifiers to be associated with various pointer device events.
- PSHBTNFLD (Push Button Field) allows an attention ID to be associated with a push button.

To tell which function key is pressed when you perform the read operation, you need to define your function keys using one of the following:

- Define a response indicator for the function key. A **response indicator** is an indicator that returns information back to an application. There are 99 response indicators available to you.

  Note: Response indicators are used for more than function keys. For example, you can use them to tell when the data in a field on the display has changed.
- Examine the input/output feedback area. The **input/output feedback area** is status information provided by the system about the operations performed on an opened file. To find out how to get information from the input/output feedback area, see the manual for the programming language you are using. See Appendix C, "Feedback Area Layouts for Display Files," on page 661 for a description of the information available from the feedback areas.

Defining Command Attention (CAnn) and Command Function (CFnn) Keys

The command function (CFnn) keys and command attention (CAnn) keys are numbered 1 through 24 and are the same physical set of keys on the keyboard. These keyboard keys are usually labeled *Cmdnn* or *PFnn* or *Fnn*, where nn is the associated key number. They can be used to set a response indicator or to perform a certain function.

The different command keys do the following:

Command function
A record containing changed fields is returned to the program.

Command attention
A record is returned to the program but the record does not contain the data entered by the user and no field validation is performed.

If a key is specified as a CFnn key in a file, it cannot also be specified as a CAnn key in the file. Likewise, if it is specified as a CAnn key, it cannot also be a CFnn key. For example, if function key 01 is specified as a CAnn key (CA01), you cannot specify CF01 anywhere in the same file.

If a response indicator is specified for a CFnn key and the key is pressed, the response indicator is set on and passed to the program with the input data. If a response indicator is not specified for a CFnn key, only the input data is passed.

Note: The input/output feedback area contains the 1-character attention identifier (AID), which also identifies the key pressed. See Appendix C, "Feedback Area Layouts for Display Files," on page 661 for a description of the input/output feedback area.

If a response indicator is specified for a CAnn key and the key is pressed, the response indicator is set on and passed to the program. Fields sent to the display and hidden fields are returned to the program. If a CFnn key or the Enter key was previously pressed, the input-only field is returned as previously typed data. If data was never entered into an input-only field, the field is returned as blanks (character field) or zeros (numeric field). Fields changed by the user since the last time a CFnn key or Enter key was pressed are not returned.

The use of CAnn keys can cause the input buffer of the program to contain user-entered data that does not meet the validation specified in the display file. For example, the user enters data and presses a CFnn key or the Enter key, and the data is validated as defined in your DDS. Input data is processed one field at a time with data manipulation taking place before the validity checking. If a validity-checking error occurs, a message is selected and all the other input data is processed. After all input data is processed and one or more errors have occurred, a message is sent to the user. Then, if the user presses a valid CAnn key, no changed data is sent from the display. The data is moved from the input buffer save area to the input buffer. The input buffer now contains the data that is in error. If your program is not going to process this data when the CAnn key is pressed, you do not have a problem. If this is a problem, avoid using CAnn keys; only use CFnn keys so that data that is not valid can be detected.

If you want to use CAnn keys, you should not specify the following validity-checking DDS keywords:

CHECK(M10)
CHECK(M11)
CHECK(VN)
CHECK(VNE)
CHKMSGID
COMP/CMP
RANGE
VALUES

The Print, Help, Clear, and Home keys operate in the same manner as the CAnn keys. The Roll Up, Roll Down, Page Up, and Page Down keys operate in the same manner as CFnn keys.

Specifying Alternative Keys

You can also define command attention or command function keys to perform the functions of the Help, Page Up (or Roll Down) and Page Down (or Roll Up) keys. The function key specified on the keyword identifies the alternative key to be used.

The DDS keywords are:

- **ALTHELP**: Indicates that the help function will be started when either the Help key or the key specified on the ALTHELP keyword is pressed. If the ALTHELP keyword is specified but an alternative key is not specified, the default is CA01. Note that the Help key is an attention key, not a function key, because it does not return input.
- **ALTPAGEUP** and **ALTPAGEDWN**: Indicate that the paging functions will be started when the page keys or the keys specified on the keywords are pressed. If alternative keys are not specified on the ALTPAGEUP or ALTPAGEDWN keywords, the defaults are CF07 and CF08, respectively. Note that the page keys are function keys, because they return input.

The alternative keys specified on the ALTHELP, ALTPAGEUP, and ALTPAGEDWN keywords provide the same function as the actual keys. For example, if pressing the Help key starts the help function, then pressing the alternative key defined by the ALTHELP keyword will also start the help function. Likewise, if pressing the Page Up or Page Down key returns control to the application program, then pressing the alternative key will also return control to the application program. Both of these examples appear to the program as if the actual key was pressed.

The user profile option for paging (USROPT(*ROLLKEY)) applies to the PAGEUP, PAGEDOWN, ALTPAGEUP, ALTPAGEDWN, ROLLUP, and ROLLDOWN keywords.

The alternative help key function does not work when the keyboard is locked. For example, if you type information into a field that is not input-capable, a controller-detected error occurs and flashing numbers appear. The Help key can be used to get more information about the error. The function key specified as the alternative help key will not be valid until the Reset key is pressed, and then the help information will no longer be available.

Passing Information via Indicators

Indicators are one-character fields that exist either in the input records and output records used by the program or in a special indicator area. An indicator is on if it has the value 1 and off if it has the value 0. You can use indicators to pass information from a program to the system or from the system to a program. You specify how indicators are to be used through the DDS for the display file.

There are two types of indicators for display files:

Option indicators:
Pass information from an application program to the system. These typically are used to control the processing of a particular record format by the system.

Response indicators:
Pass information from the system to an application program when an input request completes. Response indicators can inform the program which function keys were pressed by the user or whether data was changed by the user.

Both option and response indicators can be specified at the file level, record format level, and field level. Indicators specified at the file level apply to all record formats within the file.

Removing Option and Response Indicators from the Record Area

You can use the Indicator Area (INDARA) keyword to separate the option and response indicators from the input and output records used by the program. If you use the INDARA keyword, the indicators are placed in a separate 99-character area; see your appropriate high-level language manual for information on how this 99-character area is defined.

If you use the same indicator number as both a response indicator and as an option indicator, you can use the status of the response indicator to set the option indicator for a subsequent output operation. For example, indicator 15 is used as both a response indicator and an option indicator. If the response indicator is on when an input operation is performed on the record format, option indicator 15 is set on and will be on when an output operation is performed for that record format.

The maximum number of record formats that you can define for a display file is 1024. If you do not use the INDARA keyword, the maximum number of fields that you can specify depends on the number of indicators (1 character each) you use and the length of each field you describe. The total combined length of all fields and indicators in a record format cannot exceed 32 763 characters. If you use the INDARA keyword to specify a separate indicator area, the maximum number of fields that you can specify depends only on the length of each field. The total number of all fields cannot exceed 32 763.

Enabling Different Response Indicators Simultaneously

It is possible to have different response indicators for the ROLLUP/ROLLDOWN keywords on record formats displayed at the same time. For example, record A has specified a roll-up indicator of 52 and record B has specified a roll-up indicator of 25 and both records are displayed. When a read operation is requested to record A in your program, the operator presses the Roll Up key which returns control to your program. Record A is passed to your program with response indicator 52 set on; response indicator 25 is not set. Your program can then do a read operation to record B. When record B is passed to your program, response indicator 25 is set on; response indicator 52 is not set. Only the response indicator specified on the record format for which the read operation is done is set. The record format in which the cursor was located when the Roll Up key was pressed does not affect the setting of the response indicator associated with the ROLL keyword.

Setting an Indicator Off

An indicator specified on the SETOF or SETOFF keyword becomes a response indicator that is set off and returned to the program. The indicator is not set off until an input operation is performed. If the same indicator is specified elsewhere in the record format as a response indicator, the indicator is returned to the program based on the status of the associated keyword condition. For example, if response indicator 01 is specified both for the SETOF/SETOFF keyword and the CF5 key, indicator 01 is returned in the on condition when the CF5 key is pressed. If the indicator is specified elsewhere as a response indicator, there is no need to use the SETOF/SETOFF keyword.

Inserting Constant Field Text from a Message Description

You can specify that the text for constant fields is contained in a message description using the Message Constant (MSGCON) keyword.

If the message description used for the constant text is shorter than the field on the display, the remaining portion of the field is padded with blanks. If the message description is longer than the field, the message description is truncated.

If the messages description does not exist when the DDS is compiled, the file is not created. If you change the message description, you will have to create the file again if you want the display file to contain the updated messages.

Allowing for Right-to-Left Cursor Movement

The cursor can be made to move from right to left on the display between fields and in input fields. Two parameters for the DDS CHECK keyword can be used to do this:

- CHECK (RL): Moves the cursor from right to left in specified nonnumeric input fields or all nonnumeric input fields on the display.
- CHECK (RLTB): Moves the cursor from right to left between fields.

When using these parameters, remember the following:

- Modulus check digit verification is supported, but the check digit is still the byte to the extreme right of the field.

- A field for which right-to-left cursor movement is specified can occupy more than one line on the display. However, the cursor still moves from the top of the display to the bottom.
- You cannot use right-to-left cursor movement with user-defined data streams.

Note: If no cursor positioning is specified in the display file or by the program, the cursor is placed in the input-capable field to the extreme left of the top line.

Defining Cursor Movement to Input-Capable Positions Only

Use the cursor input only (CSRINPONLY) keyword to restrict cursor movement to input-capable positions only. This keyword affects only the cursor arrow keys. This function moves the cursor to the first input-capable position on a display in the direction of the arrow key. The user needs to press the cursor key only once in the appropriate direction to have the cursor move to the input-capable position.

Specify this keyword at the file or record level.

The input-capable positions to which the cursor can move include the following:

- Input field (except protected fields)
- Selection-field choice (except those on which you cannot place the cursor because of its choice control (CHCCTL) value)
- Selection-list choice (except those on which you cannot place the cursor because of its choice control (CHCCTL) value)
- Message line (if a message is displayed and the keyboard is not locked).
- Message subfile defined with the subfile message record (SFLMSGRCD) keyword.

Several DDS keywords (such as DSPATR(PC) and CSRLOC) can be used to position the cursor at any display position. This is true even if the CSRINPONLY keyword is specified. The first subsequent cursor movement keystroke will move the cursor to a cursorable location. If no cursorable position exists on the display, the cursor will be positioned to row 1, column 1. Once the cursor has been moved from this position, pressing the home key repositions the cursor back to its initial position.

If a window is displayed with no input fields, the cursor is positioned at row 1, column 1 of the window. If a cursor movement key is pressed, the cursor moves to row 1, column 1 of the full display (outside the window). If the window is defined with *RSTCSR, command keys are not valid outside the window. Pressing the home key returns the cursor to the window. Pressing any command key or the Enter key sounds an alarm and returns the cursor to the window. To avoid this problem, consider specifying an input inhibited input field in the upper corner of the window or specifying *NORSTCSR on the window keyword.

Notes:

1. If a message subfile is defined with a SFLPAG keyword greater than 1 and the CSRINPONLY keyword is in effect, any fields that have been turned to reverse image because of an error, will be turned to unreverse image if the message subfile is rolled to a partial page of messages.
2. Fields that have been turned to reverse image because of an error are turned to unreverse image when the following conditions are true:
   - A message subfile is defined with a SFLPAG keyword greater than 1
   - The CSRINPONLY keyword is in effect
   - The message subfile is rolled to a partial page of messages
3. When a record is written with the PUTOVR, ERRMSG, or ERRMSGID keywords in effect, the state of the CSRINPONLY keyword is not changed. If the CSRINPONLY keyword is in effect prior to the write operation with the PUTOVR, ERRMSG, or ERRMSGID keywords, the CSRINPONLY remains in effect. This is true regardless of the optioning of the CSRINPONLY keyword on the record assigned the PUTOVR, ERRMSG, or ERRMSGID keywords. This is also true regardless of the optioning of the PUTOVR, ERRMSG, or ERRMSGID keywords on the record assigned the CSRINPONLY keyword.

4. Writing a record with the PROTECT keyword does not affect the input fields associated with messages when the CSRINPONLY keyword is in effect. Any messages displayed are not protected. Therefore, the cursor may still be moved the messages.
5. The CSRINPONLY keyword is valid only for display stations attached to a controller that supports an enhanced interface for nonprogrammable work stations. It is ignored on display stations attached to other controllers.

Defining Cursor Progression for Entry Fields

The FLDCSRPRG keyword lets the user specify the next field the cursor should move to when the cursor leaves a field.

The DDS for the field looks like this:

```
|...+....1....+....2....+....3....+....4....+....5....+....6....+....7...
     A            F1            10A  B  3  4FLDCSRPRG(F3)
     A            F2            10A  B 13  4FLDCSRPRG(F1)
     A            F3            10A  B 16  4FLDCSRPRG(F2)
```

Figure 3. DDS for Field-Level Cursor Progression

The parameter for the FLDCSRPRG keyword is the name of the field the cursor will go to when forward field-exit processing is performed. When the cursor leaves F1 because of a field exit key, it goes to F3. If the field named with this keyword is optioned off, cursor progression for this field is ignored.

Note: When the cursor leaves a field using backward field-exit processing, the cursor moves to the first field on the display that has the exited-field name specified on the FLDCSRPRG keyword. For the DDS in Figure 3, if backward field-exit processing is used to leave field 2, the cursor moves to field 3.

SFLCSRPRG is the keyword used for subfile cursor progression. The DDS for subfiles looks like this:

```
|...+....1....+....2....+....3....+....4....+....5....+....6....+....7...
     A          R SFL01                     SFL
     A            S1            10A  B  5  5SFLCSRPRG
     A            S2            10A  B  5 25
     A          R CTL01                     SFLCTL(SFL01)
     A                                      SFLPAG(5) SFLSIZ(20) SFLDSP
```

Figure 4. DDS for Subfile-Level Cursor Progression

The SFLCSRPRG keyword causes the cursor to move from a field in a subfile record to the same field in the next displayed subfile record. Without SFLCSRPRG, the cursor moves from a field in a record to the next field in the same record. When the cursor leaves field S1 of the first record of the subfile, it goes to S1 of the second record of the subfile. Without the SFLCSRPRG keyword, the cursor goes to field S2 of the first record. When the cursor leaves S2, it goes to S1 of the next record because S2 does not have the SFLCSRPRG keyword. This keyword is not allowed with subfiles that use field selection. It cannot be used with horizontal subfiles. When the cursor is on S1 of the last SFL record displayed, the cursor moves to the next input field below the last SFL record. If there are no remaining SFL fields, the cursor moves to the top of the display.

Note: The FLDCSRPRG keyword and the SFLCSRPRG keyword are ignored on displays that are attached to a controller that does not support an enhanced interface for nonprogrammable work stations.

Defining Attributes for Entry Fields

An entry field's leading field attribute is changed to a specified attribute when the cursor enters the field.

The DDS for the field looks like this:

```
|...+....1....+....2....+....3....+....4....+....5....+....6....+....7...
     A            F1            10A  B  3  4ENTFLDATR(*CURSOR (*DSPATR CS))
     A            F2            10A  B 13  4ENTFLDATR(*NOCURSOR (*COLOR RED))
```

ENTFLDATR tells the system to change the attribute of the field when the cursor enters the field. *CURSOR and *NOCURSOR are used to specify whether the cursor is visible when it enters the field. If the *NOCURSOR option is specified, the cursor row and column values in the input-output feedback area indicate the first position in the field. You can also specify a color or an attribute.

Note: The ENTFLDATR keyword is ignored on displays that are attached to a controller that does not support an enhanced interface for nonprogrammable work stations.

Protecting Entry Fields Using Edit Masks

The EDTMSK keyword is for fields with EDTCDE or EDTWRD keywords. When the field is displayed, certain areas of the field will be protected. You define which areas to protect.

The DDS for the field looks like this:

```
|...+....1....+....2....+....3....+....4....+....5....+....6....+....7...
     A            F1             6  0B  3  4EDTWRD('  /  /  ')
     A                                      EDTMSK('  &  &  ')
     A            F2             6  0B  4  4EDTCDE(Y)
     A                                      EDTMSK('  &  &  ')
```

The ampersand (&) represents a protected part of the field. A blank represents an unprotected part of the field. The length of the edit mask must equal the display length of the field. The number of unprotected positions must at least equal the program length of the field. You must only protect nonnumeric data because protected data is not returned if the field is changed. Wherever there is an &,; that part is protected no matter what data is in the field.

The first field has the slash (/) characters protected in a date. For the second field, the / in the date is always protected.

Keyboard functions on displays attached to a controller that supports an enhanced interface for nonprogrammable work stations are the same for edit-mask fields as they are for continued-entry fields.

Note: The EDTMSK keyword is ignored on display stations attached to a controller that does not support an enhanced interface for nonprogrammable work stations.

Specifying Right-to-Left Display Processing

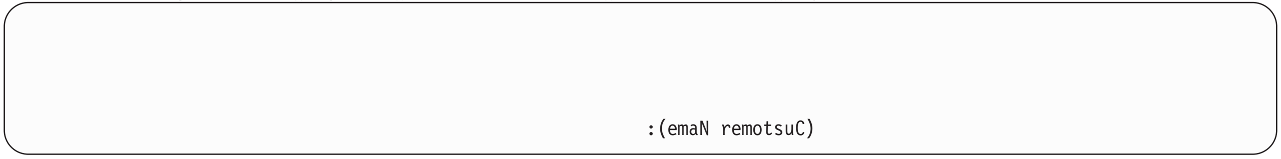
You can specify that records in a display file be written in the right-to-left direction by using the Display Right-to-Left (DSPRL) keyword. This keyword is allowed only at the file level.

Figure 5 shows an example of the DDS coding.

```
|...+....1....+....2....+....3....+....4....+....5....+....6....+....7...
     A                                      DSPRL
     A          R RECORD
     A            FLD1          20A     5  5')Customer Name(:'
```

Figure 5. Sample DDS for Right-to-Left Display Processing

The DDS in Figure 5 on page 31 produces this output on the display:



```
                                                      :(emaN remotsuC)
```

Notice that the left and right parentheses in the DDS are reversed; this is so they appear correctly on the display. All symmetrical characters have to be specified in this way.

If your application program uses one display file with the DSPRL keyword and another display file with the WINDOW keyword, make sure that the display file with the WINDOW keyword also specifies the DSPRL keyword. Otherwise, the display assumes the orientation of the display file that has the WINDOW keyword.

If you specify the DSPRL keyword, the cursor moves from right to left when you enter data. Therefore, it is not necessary to use the CHECK(RL) keyword. If you specify CHECK(RL) and DSPRL, the CHECK(RL) keyword is ignored.

The DSPRL keyword causes all records in a display file to be written in the right-to-left direction. You cannot specify that individual records be written in the left-to-right direction.

If you specify the ERRMSG or the ERRMSGID keywords with the DSPRL keyword, the messages associated with these keywords display in the left-to-right direction.

Specifying Word Wrap for Fields

Word wrap is the function that automatically moves the last word in a field down to the next line in the field if the word runs beyond the right margin of the field. To specify the word wrap function for a named field, use the word wrap (WRDWRAP) keyword. This keyword can be used at the file, record, or field level. It can only be used with input-only (I) or output/input (B) fields.

Notes:

1. This function is available only for display stations attached to a controller that supports an enhanced interface for nonprogrammable work stations.
2. The Reverse key and the Close key cannot be used in a word wrap field.
3. When word wrap is used and the keyboard is in insert mode, null characters are not shifted to the right; they are replaced.

Word wrap is not allowed for these fields:

- DBCS-only fields
- Pure fields
- Either fields (with double byte)
- Open fields with SBCS data

Word wrap is not allowed with the following field types or features:

- Signed numeric
- Numeric only
- Digits only
- Magnetic Stripe Reader (DSPATR(OID))
- Light Pen (DSPATR(SP))
- Right-justify
- Mandatory fill

- Self-check (M10F/M11F)
- Dup allowed
- Right-to-left cursor movement (CHECK(RL))
- Right-to-left, top-to-bottom cursor movement (CHECK(RLTB))

If all the data cannot fit within a word wrap field without splitting words, the word wrap function for that field is ignored. The data is written as if word wrap had not been specified. The subsequent operation of the field is also as if word wrap were not specified.

Word wrap may be specified on fields that are contained on a single line. In this case, when the keyboard is in insert mode, null characters are not shifted to the right; they are replaced.

Specifying Word Wrap for Fields—Tips

Some things to consider when using the word wrap function:

- The total length of the input field should allow for character positions at the ends of lines or segments to be used for padding when a wrap occurs. If a field is too short, errors will occur or word wrap will be turned off.
- The length of each line or segment should be as large, or larger than the longest word that may be entered in the field. If a line or segment is too short, errors will occur or the data may be shifted down to the last line or segment.
- Extra blanks that are inserted to make a wrap occur are removed when data is returned to your program.

Emphasizing Fields

You can emphasize a field of a record on the display by specifying the following in the DDS for the file:

Note: Any function not supported for your display station is ignored.

Table 5. DDS for Emphasizing Fields

Type of Emphasizing	DDS keyword
Underlining a field (the default for input fields)	DSPATR(UL)
Placing vertical separators between the characters in a field	DSPATR(CS)
Highlighting a field by displaying it with greater intensity than is normally used on the display	DSPATR(HI)
Reversing the image of a field from light on dark to dark on light or from dark on light to light on dark	DSPATR(RI)
Making the data in the field invisible to the display station user	DSPATR(ND)
Placing the cursor at a specific field	DSPATR(PC)
Blinking a field when it is displayed	DSPATR(BL)

Another way of specifying attributes for a field is by using a program-to-system field parameter on the DSPATR keyword. Your application program uses the program-to-system field to set the display attributes or protection attribute for the field to which the DSPATR keyword applies.

Figure 6 on page 34 shows an example of the DDS coding for program-to-system fields:

```
|...+....1....+....2....+....3....+....4....+....5....+....6....+....7...
     A          R RECORD
     A            FLD1           5A     2  2DSPATR(&PFLD1)
     A            FLD2           5A     2  9DSPATR(&PFLD2)
     A            PFLD1          1A  P
     A            PFLD2          1A  P
```

Figure 6. Sample DDS for Program-to-System Fields

One program-to-system field may be used for multiple fields within a record. Only one program-to-system field can be used per field. You cannot specify the following attributes using the program-to-system field:

MDT Set changed data tag when displayed

OID Operator identification

PC Position cursor

SP Select by light pen

For the valid hexadecimal values that your program can pass to the program-to-system field, see the **DDS** topic in the iSeries Information Center.

Adding Color

You can design your displays for use on display stations that show color. The DDS keyword COLOR allows you to specify the following colors for fields: green, white, red, turquoise, yellow, pink, and blue. This keyword is ignored if it is selected for a field displayed on a display station that does not support color.

If the COLOR keyword is not specified in the DDS for the display file but the display station is specified in the display station description as a color display station, displays that you have designed for display stations that do not support color can also be used for the color display station. The keywords DSPATR(UL) and DSPATR(RI), if specified on separate fields, function the same as they do for the 5250 display station. However, the keywords DSPATR(CS), DSPATR(HI), and DSPATR(BL) produce the following colors on a color display station (the specified display attributes CS, HI, and BL are suppressed):

Color Produced on the Color Display Station when No COLOR Keyword is Specified	Display Attribute Selected: DSPATR(CS)	Display Attribute Selected: DSPATR(HI)	Display Attribute Selected: DSPATR(BL)
Green (normal)			
Turquoise	X		
White		X	
Red, no blinking			X
Red, blinking		X	X
Yellow	X	X	
Pink	X		X
Blue	X	X	X

Editing Output Fields

The system provides editing support that makes fields more readable when they are displayed. With the system editing support, you can do the following:

- Suppress leading zeros

- Punctuate a field with commas and periods to show decimal column and to group digits by threes
- Show negative values with a minus sign to the left or right
- Show negative values with the letters CR (credit) to the right
- Show zero values as zeros or blanks
- Show asterisks to the left of significant digits to provide asterisk protection
- Show a currency symbol corresponding to the system value QCURSYM

The system provides this editing support with edit codes and edit words. **Edit codes** are a defined set of editing patterns. In addition to those provided by the system, you may also define your own edit codes. You identify edit codes by name, and the system edits a field according to the pattern defined by the named edit code. **Edit words** are edit patterns that you define to produce the desired results. Edit codes cover most commonly used editing requirements. You need to use the edit word support only for those editing needs not covered by edit codes.

Edit codes are used as follows:

- If your application is using program-described data, your high-level language may allow you to identify edit codes or create your own edit words.
- If your application is using externally described data, the edit code (EDTCDE) DDS keyword allows you to identify an edit code, and the edit word (EDTWRD) DDS keyword allows you to define your own editing pattern.

The system provides several edit codes. The editing patterns defined by these codes are contained in Appendix E, “Edit Codes.”

Defining Your Own Edit Codes

You can define five edit codes to provide more editing function than is available with the i5/OS edit codes, and to handle common editing functions that would otherwise require the use of an edit word. These are called user-defined edit codes. For example, you may need to edit numbers that include hyphens (like some telephone numbers), or more than one decimal point. You can use user-defined edit codes for these functions. These edit codes are named QEDIT5, QEDIT6, QEDIT7, QEDIT8, and QEDIT9 and can be referred to in DDS or a high-level language program by number (5, 6, 7, 8, or 9).

These edit codes are created by using the Create Edit Description (CRTEDTD) command. Edit descriptions are always placed in library QSYS. They cannot be moved or renamed; only one occurrence of each is allowed. Edit descriptions have an object type of *EDTD.

Even though they are user-defined edit codes, your system comes with a version of each one of them. You can use these edit descriptions as they are, or you can delete them and create your own. The editing performed by the IBM-supplied versions of these edit descriptions as well as a definition of the contents of and the rules for using any user-defined edit code are described in Appendix E, “Edit Codes.”

Before using any of the user-defined edit codes, you should check its contents on your system, since it may have been changed from the IBM-supplied version. The Display Edit Description (DSPEDTD) command will display the contents of a user-defined edit code.

Changing a user-defined edit code description does not affect any application or display file that has already been created using that edit description. If you want your application to use the changed edit description, you must either create the high-level language program again (if the edit code is referred to in the program) or create the file again (if the application is using an externally described file that contains EDTCDE keywords).

Specifying Valid Screen Sizes

In some cases, you can use the following screen size condition names to select keywords and display locations based on screen size:

- *DS3, 24 by 80 (5251 Models 11 and 12, 5291, 5292, 3179 Model 2, 3180-2, 3196, and 3197)
- *DS4, 27 by 132 (3180-2; 3197 Models D1, D2, W1, W2; 3477 Models FA, FC, FD, FE, FG, FW; 3487 Models HA, HG, HW, HC)

Note: The capability to display in 27 by 132 mode is allowed on 3180-2, 3197, 3477 Models FA, FC, FD, FE, FG, FW, and 3487 Models HE, HD, HW, HC display stations attached to a 6040 or 6041 or 2638 local display station controller, or remotely attached to a 5294 or 5394 controller. The display size for 27 by 132 mode is ignored for the DSPSIZ keyword unless these controllers are specified.

These condition names can be used to place fields in different locations on different sizes of screens. However, the fields must still be specified in the same order on each size of screen. For example, the following DDS formats a display for both a 24 by 80 screen and a 27 by 132 screen:

```
|...+....1....+....2....+....3....+....4....+....5....+....6....+....7...
     A                                      DSPSIZ(*DS3 *DS4)
     A                                       .
     A                                       .
     A                                       .
     A          R RECORD
     A            NAME          20      5  2
     A  *DS4                            3112
     A            ADDR          30      6  2
     A  *DS4                            4102
```

Figure 7. Sample DDS for Two Display Sizes

Normally, the display files are set up for a 24 by 80 screen (default size). The DSPSIZ keyword specifies which display sizes are valid for a file and indicates which sizes are the primary and secondary screen sizes. (The primary screen size is the first or only DSPSIZ value.) On the DSPSIZ keyword, the screen size can be specified as *DS3, *DS4, 24 80, or 27 132. For example, DSPSIZ (24 80) specifies a screen size of 24 by 80. When primary and secondary display sizes are specified, the display file is validated for both sizes.

The screen size designated as the primary screen size should be the one with which the display file will most often be used. A performance benefit will be realized by coding the DSPSIZ keyword in this manner. Additional processing is performed when the actual screen size is the secondary screen size.

The screen size condition names let you improve the use of a single display file for any size screen. For example, when you are using subfiles, you can specify 22 records per page for a 24 by 80 screen or 25 records per page for a 27 by 132 screen:

```
|...+....1....+....2....+....3....+....4....+....5....+....6....+....7...
     A                                  DSPSIZ(*DS4 *DS3)
     A                                   .
     A                                   .
     A                                   .
     A                                  SFLPAG(25)
     A  *DS3                            SFLPAG(22)
     A                                   .
     A                                   .
     A                                   .
```

Figure 8. Sample DDS for Subfiles for Two Display Sizes

You can also define your own screen size condition names.

Enabling Your Display to Be Printed

If the Print key is enabled for your display, the user can print the current display by pressing the Print key. The parameter (or lack of parameter) you specify for the DDS PRINT keyword controls how your display is printed:

Table 6. PRINT Keyword Results Using Print Key

Parameter	Action
No parameter	The output goes to the display station printer associated with this display (the PRINTER parameter on the Create Device Description for Display (CRTDEVDSP) or Change Device Description for Display (CHGDEVDSP)). If the operation to the display station printer fails or if there is no display station printer specified, the output goes to the printer file specified on the PRTFILE parameter on the display station description. The default for the PRTFILE parameter is QSYSPRT.
File name and, optionally, library name	The print operation is directed to the specified printer file. If the operation fails, it is directed to the default printer file, which is specified on the PRTFILE parameter on the CRTDEVDSP or CHGDEVDSP command.
Response indicator	Control returns to the program with the response indicator set on.
*PGM	Control returns to the program which must check the attention identifier in the input/output feedback area to determine which key was pressed.

The PRINT keyword can be used at the file level and also at the record level. When PRINT is specified at the record level, several records with different forms of the PRINT keyword (or with no PRINT keyword) may be displayed on the screen at the same time. The last record format written to the screen controls the use of the Print key for the entire screen.

If you specify the PRINT keyword in any form, the user can print a display containing the message help.

The PRTKEYFMT parameter allows you to control what information is included when you print your display:

- Output only
- Output with header information (rows and columns)
- Output with border information (title lines, which include the system name, date and time, and formatted user and display device name)
- Output with both border and header information

The PRTKEYFMT parameter on the Change Job (CHGJOB) and Retrieve Job Attributes (RTVJOBA) commands allows you to select how you want your Print key output to look.

When you change the device description of a display station printer (by using the CHGDEVPRT command or the DLTDEVD and CRTDEVPRT commands), you should also change the device description of the associated display station, using the PRINTER parameter on the CHGDEVDSP command. You should do this even if the name of the printer, whose device description you changed, remains the same.

Defining Windows

There are applications that could make use of a window on the display to assist the user in entering data.

A **window** is information that overlays part of the current display and allows the user to read the information inside the window. The remainder of the display is not overlaid by the window and can still be read by the user.

You can create windows by using standard DDS or by using user-defined data streams. To use standard DDS, see Chapter 5, “Defining Windows with Display Files,” on page 115. Examples are also available in the QUSRTOOL library.

Using Program-Described Data

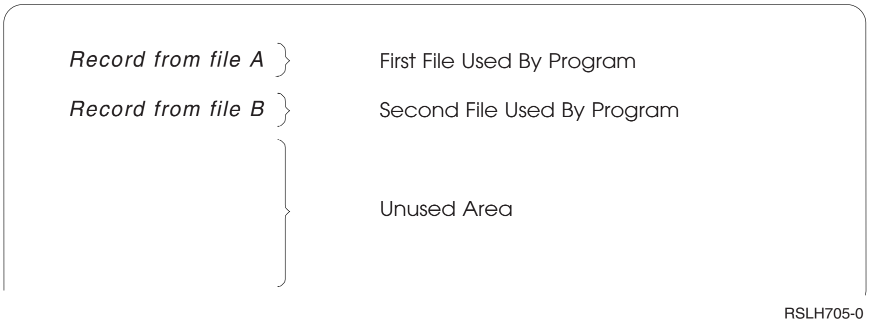
You can create a display file without using data description specifications. Such a display file then uses program-described data, and has no record or field level descriptions of its own.

When you are using program-described data with a display file to communicate with one or more display stations, only simple display formatting can be performed, and that formatting must be specified in the high-level language program that is using the file. All field descriptions are defined and all processing is performed in the program that uses the file. More than one display file can be opened to the same display station at the same time within the same program, but only two can be used on the same display station at the same time: one for input and one for output.

When a display file that uses program-described data is opened, the system treats the area on the display as a single field. That is, the field length is the same as the record length. The record length is defined by the program that is using the file, and stays the same from the time the file is opened until it is closed. Indicators cannot be passed when records are passed from the program to a display station, or from the display station to a program. Also, command keys cannot be used for program-described display files.

The space on the display is assigned to program-described display files as shown in the following example.

Records for the first file used by the program appear on the first (top) part of the display. Records for the second file appear on the display immediately following the area used by records in the first file.



The record from file B starts at the beginning of the first full line after the last line of the record from file A. If the record from file A does not completely fill its last line, the space is used by neither record and must be accounted for when calculating the maximum record lengths. In program-described display files, the maximum record lengths are:

- For an input file, the screen size minus 2

- For an output file, the screen size minus 2
- For an output/input file, the screen size minus 2
- For two files (one output and one input), the screen size minus 3

When a program-described display file is opened, it can be defined as:

- Input only
- Output only
- Input and output

Defining Input-Only Files

When an input-only file is opened, the record is initialized to a single blank field on the screen. The cursor is positioned at the first position of the field and the user can type in any type of data.

When the program reads the record, the input is passed to the program. The record is not erased from the screen. The cursor is again positioned at the first position of the record (field) and the keyboard is unlocked when the program reads the next record. The user can then type in the next record over the previous record.

Defining Output-Only Files

When an output-only file is opened, the record is initialized to a single blank field on the screen. When the program writes a record to the file, the record is displayed and the keyboard is locked. The user must press the Enter key before another record can be written to the file. Subsequent records written to the file erase the currently displayed record because only one record can be displayed for the output file.

Defining Input and Output Files

When an input and output file is opened, the record is initialized to a single blank field on the screen. The cursor is positioned at the first position of the file and the user can type in any kind of data.

The program that is using the file can read records or write records in any sequence. Whenever a record is written to the file, the modified data tag is set off (to indicate that data was not entered into the field) and the keyboard is unlocked. If the user then enters data into the field, the modified data tag is set on. If the next operation is a write operation instead of a read operation, the data typed in by the user is written over and the modified data tag is set off again.

Chapter 3. Working with Display Files in an Application

After you define your display file, you can use it in an application. This chapter discusses the operations performed when a display file is used in an application.

Understanding How the System Allocates Resources

When a high-level language program uses a display file, several operations require that the system allocate the resources needed to perform that operation. Allocating file resources causes the system to obtain a lock on a display file when it is opened. Multiple users then cannot use the same display file in conflicting ways. For example, the system will not allow you to delete a file while any application program is using it.

File resources are allocated in two ways:

- The system performs the allocation whenever an operation is requested that requires it. The following operations for display files require allocation:
  - *Open*: The file resources include the file description and the display station. More information about the open operation is found in "Opening Display Files."
  - *Acquire*: The display station is allocated as a resource. More information about the acquire operation is found in "Acquiring a Display Station for I/O Operations" on page 42.
  - *Starting a program on a remote system*: The file resources include session resources needed for APPC/APPN.
- If you prefer to ensure that all the resources that are needed by a program are available before the program is run, you may use the Allocate Object (ALCOBJ) CL command in the job prior to running the program.

When allocating resources, the system waits for a predefined time if the resources are not immediately available. If the resources do not become available within the time limit, the following happens:

- If you are using the ALCOBJ command, the command fails.
- If your program is performing a file operation, that operation fails and an error message is sent to the program message queue.

You may attempt to use the error handling functions of your high-level language to try the operation again. For example, if an open operation fails because another job is using the display station associated with the file, you could retry the open operation later when the display station is no longer being used.

The length of time that the system waits when allocating resources is specified on the ALCOBJ command and on the WAITFILE parameter of the CRTDSPF command. If the ALCOBJ command is used prior to running a program, then the value of the WAITFILE parameter is ignored because resources are available. If your application has error handling procedures for display station errors occurring on display files, you should specify a value other than *IMMED to allow the system to recover from the display station error. The system recovery procedures for the display station must be completed before your program can recover from errors due to the allocation of resources.

Opening Display Files

The **open operation** connects a display file to a program for processing.

When a display file is opened, usually one or more display stations are implicitly acquired, or automatically prepared for I/O operations, for the display file:

Table 7. Display Stations Implicitly Acquired When Display Files Are Opened

How the Display File is Defined When Opened	Other Specifications	Display Station Implicitly Acquired
Defined with a single display station	*REQUESTER specified	Display station at which the user requested the program
	*REQUESTER not specified	Display station specified on the DEV parameter of the CRTDSPF, CHGDSPF, or OVRDSPF command
Defined with multiple display stations	Opened by a CL program	All display stations specified on the DEV parameter of the CRTDSPF, CHGDSPF, or OVRDSPF command
	Opened by any high-level language other than CL	The first display station specified on the DEV parameter of the CRTDSPF, CHGDSPF, or OVRDSPF command
Defined with no display stations	-	None

The value specified on the WAITFILE parameter for the CRTDSPF, CHGDSPF, or OVRDSPF command is used to determine how long the open operation should wait for file resources to become available so they can be allocated. If a file resource, such as a display station, does not become available and the wait time specified ends, the open operation fails.

Implicitly acquiring a display station when the file is opened results in the following:

- The screen is cleared completely and the cursor is placed in the upper-left corner of the display.
- The keyboard is unlocked.

Any display station to be implicitly acquired at the open operation must be varied on. Switched-line display stations can also be acquired if they are in a vary-on-pending state. Also, a display station other than the *REQUESTER display station cannot be acquired if it is signed on.

Acquiring a Display Station for I/O Operations

The system implicitly acquires, or automatically assigns, a display station to a display file when the display file is opened. However, you may also acquire additional display stations for your program using the **acquire operation**. The acquire operation is used in multiple display file applications or if you are performing error recovery in your application.

A successful acquire operation results in the following:

- The screen is cleared completely and the cursor is placed in the upper-left corner of the display.
- The keyboard is unlocked.

The value specified for the WAITFILE parameter on the CRTDSPF, CHGDSPF, or OVRDSPF command is used to determine how long the acquire operation should wait for a display station to become available so it can be allocated. If a display station does not become available and the wait time ends, no display is acquired.

A display station cannot be allocated unless it is varied on. Switched-line display stations can be allocated if they are in a vary-on-pending state. Also, a display station other than the *REQUESTER display station cannot be allocated if it is signed on.

The system allows only one *REQUESTER display station to be acquired to any display file, including a multiple-device display file.

If an acquire operation is not successful, the release operation is the only valid operation to the display station.

Obtaining Information about Display Files and Display Stations

You can obtain information about the open and I/O operations performed on an open display file, and attribute information about a display station you are using.

Obtaining Information about Open and I/O Operations

After a display file is successfully opened, the system keeps track of the status of the file in feedback areas.

The **open feedback area** contains information about the display file after it has been successfully opened, including:

- Name and library of the display file
- Number of rows and columns on the display
- Name and library of the file after overrides have been applied
- Information about the display station defined for the file

The **device definition list** part of the open feedback area contains information about each device attached to the display file.

The **I/O feedback area** contains information about the I/O operations performed on the display file after it has been successfully opened. The I/O feedback area has two sections:

- The **common feedback area** contains information about I/O operations that were performed on the file. This includes the number of operations and the last operation.
- The **file-dependent feedback area** contains file-specific information for display files, such as the major/minor return code and the amount of data received from a display station.

As operations are performed on the display file, the feedback area is updated to reflect the latest status.

There is one feedback area for each open file. An exception is for shared files, which share feedback areas as well as the data path between the program and the file. For more information on shared files, see "Sharing Display Files in the Same Job" on page 85.

Feedback areas can be used to provide information when errors occur. For example, when an error occurs with a display file, the program could determine predefined error handling operations based on the major/minor return code in the file-dependent feedback area. More information on major/minor return codes is available in Chapter 8, "Handling Messages and Errors for Display Files."

Some high-level languages on the system allow you to access the status and other information about the file against which operations are being performed.

See Appendix C, "Feedback Area Layouts for Display Files," for a description of all the information available from the feedback areas.

Obtaining Attribute Information about Display Stations

The **get-attributes operation** allows you to obtain the following information about a specific display station:

Table 8. Information Available from the Get-Attributes Operation

Information	Details
The specific model of the display station	Valid only when the display station is acquired to the file
The screen size of the display station	Valid only when the display station is acquired to the file

Table 8. Information Available from the Get-Attributes Operation (continued)

Information	Details
Device acquired indicator	Indicates whether or not the display station is currently acquired to the file
Device invite status	Indicates whether or not the display station is invited, and if so whether or not the display station has data available; valid only when the display station is acquired to the file
*REQUESTER display station indicator	Indicates whether or not the display station is the *REQUESTER display station

Since the information supplied is also available in the open and input/output feedback areas for the display station that is implicitly acquired when the file is opened, the get-attributes operation is most useful for multiple display file applications. For more information on how to perform the get-attributes operation, see the appropriate high-level language manual.

See Appendix C, "Feedback Area Layouts for Display Files," for a description of all the information available from the get-attributes operation.

Sending and Receiving Data

All data written to and read from the display by the application program is done with records. Records consist of fields, which are individual items of data. The high-level languages in which application programs are written have I/O statements that give data to the system to be written to the display and receive data from the system that it read from the display in the form of records.

The I/O statements of the high-level languages also refer to record formats, which are defined using DDS. On output, a record format describes how the data given by the program is to be presented on the display as well as how the data is to be processed before presenting it. On input, the record format is used to control some display functions, to extract the program data from all the data which is on the display, and to present that data back to the application program.

Information about DDS keywords

This section uses DDS keywords that control sending and receiving information to the display. For more information about specific DDS keywords, see the **DDS** topic in the iSeries Information Center.

Determining Which Record Formats Are Active on a Display

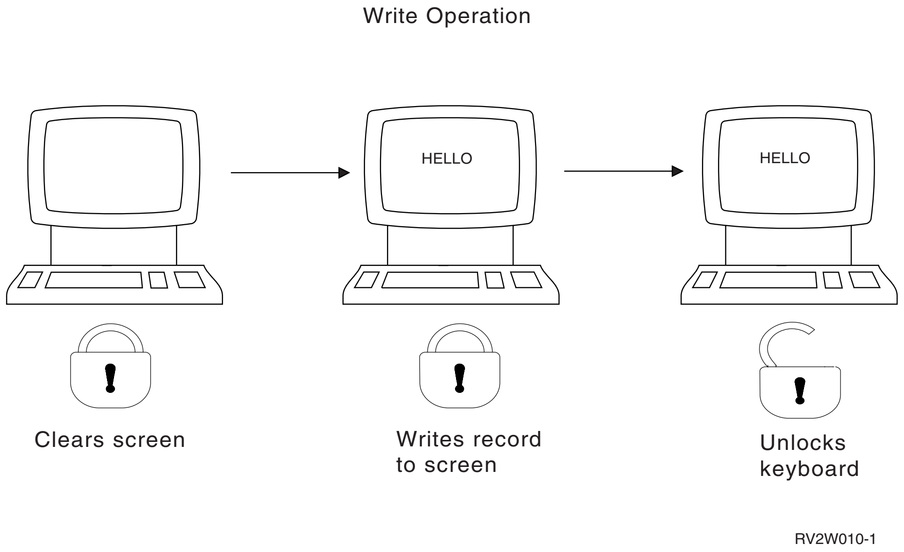
The system maintains the **active record format table**, a table of all record formats that are currently on the display. Read operations may take place against only those record formats that are in the active record format table. Certain DDS keywords cause records in the table to be altered.

A record format is added to the table when a write operation that contains the record format is performed. A record format is removed from the table when a read operation can no longer be correctly performed for this record.

The active record format table is cleared whenever the display is cleared.

Writing Output to the Display

A **write operation** passes a record from the program to the system. The record format in the display file contains the information necessary for the system to handle the record. The write operation results in the following:



Placing Records on the Display

One record format can occupy an entire screen or the screen can be divided to display more than one record format.

If a record is displayed on more than one line, the following rules apply:

- The lines must be consecutive lines on the display. For example, if one record occupies two lines and the record begins on line 2, the record must continue on line 3.
- Only a beginning attribute character can occupy line 1, position 1.
- If only the ending attribute character for the last field in a record is in position 1 of the next line, another record format can begin on that same line.
- Only one record can occupy a line on the display at a time. If you want to display a record format that overlaps one or more lines of a record format already on the display, specify *NO for the clear lines (CLRL) keyword. CLRL(*NO) clears the common lines before displaying the new record format.

The following three figures show how screens can be divided when the CLRL keyword is not specified:

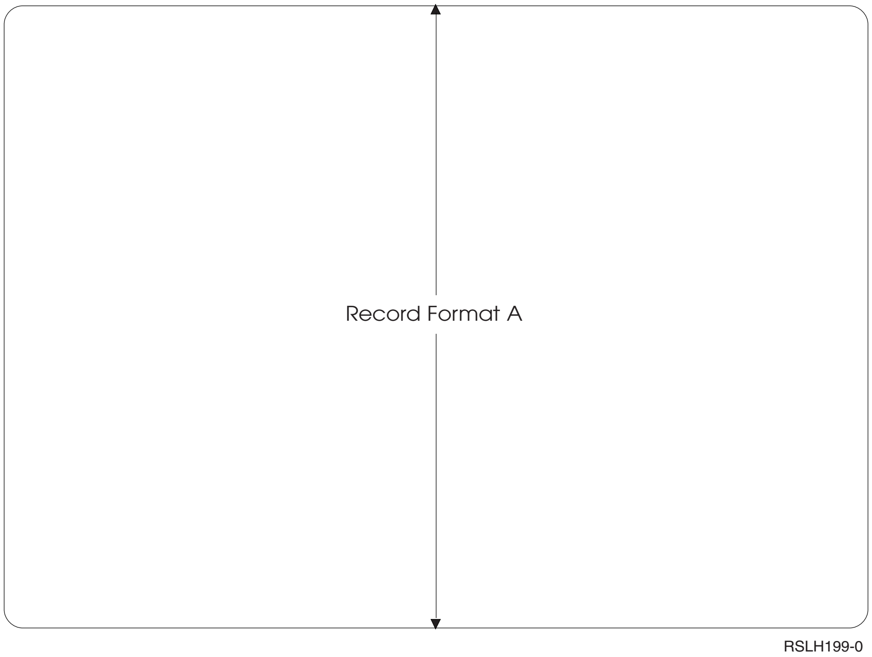


Figure 9. Valid Placement of Records on a Screen When the CLRL Keyword Is Not Used (Part 1 of 3)

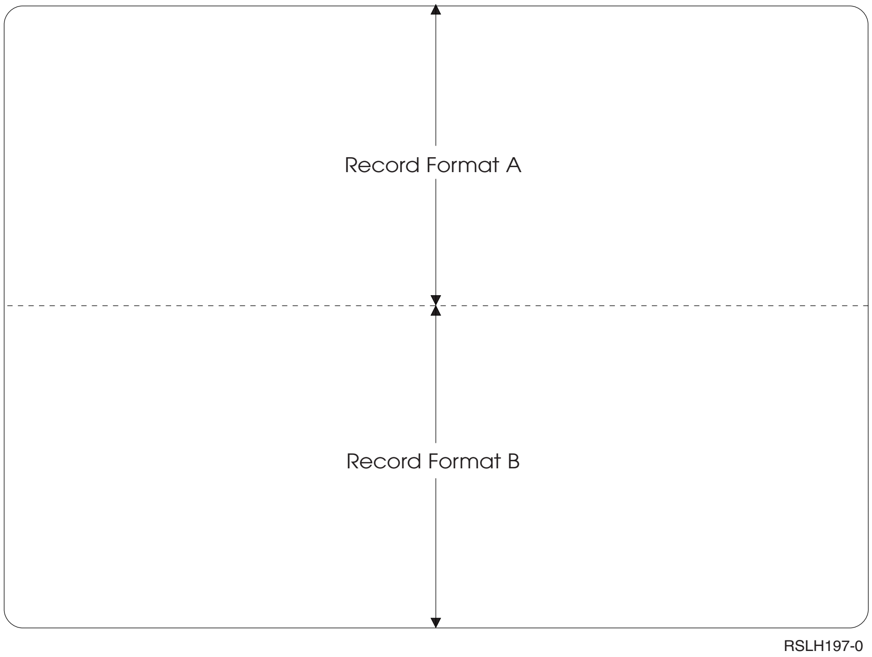


Figure 9. Valid Placement of Records on a Screen When the CLRL Keyword Is Not Used (Part 2 of 3)

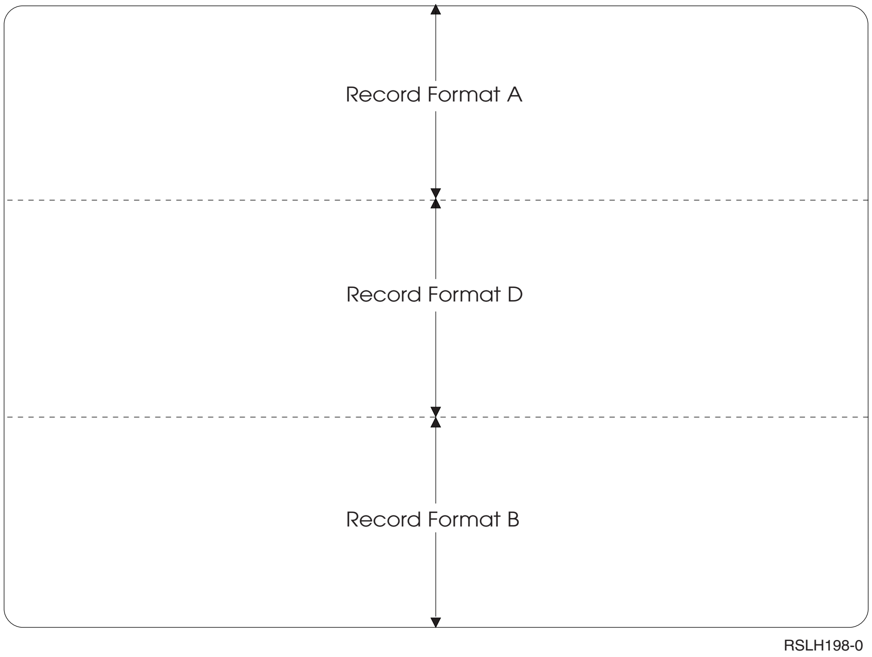


Figure 9. Valid Placement of Records on a Screen When the CLRL Keyword Is Not Used (Part 3 of 3)

The following figure shows how screens *cannot* be divided when the CLRL keywords is not specified:

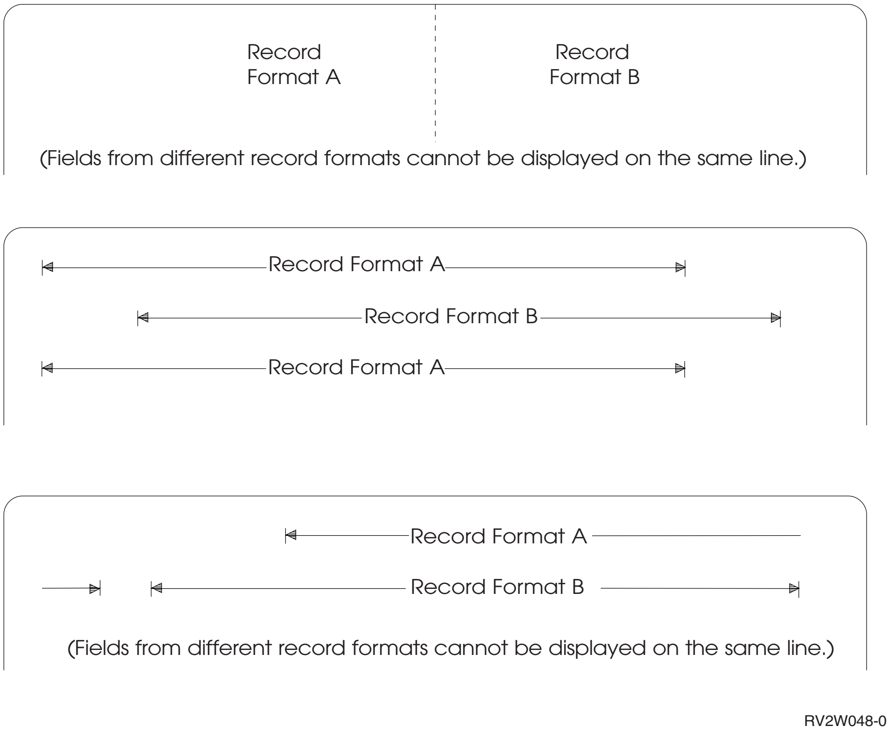

Record Format A
Record Format B
(Fields from different record formats cannot be displayed on the same line.)
Record Format A
Record Format B
Record Format A
Record Format A
Record Format B
(Fields from different record formats cannot be displayed on the same line.)
RV2W048-0

Figure 10. Wrong Placement of Records on Screen When CLRL Keyword Not Used

Understanding Which Records Do Not Occupy Space on the Display

The following types of records do not occupy space on a display but are assumed to be at line 0:

- Records with no fields defined
- Records with only hidden, program-to-system, or message fields

- Records with the CLRL keyword specified and with no input-capable fields

The system keeps track of only one of these records at a time. If an output operation for a record assumed to be at line 0 replaces another record assumed to be at line 0, you can no longer issue an input operation for the replaced record.

Changing Record Formats on a Display

The formats displayed can change while a file is being processed because information on a display can be deleted when new formats are displayed.

When your program displays a new record format for output or to allow input, the existing display is usually erased before the new record format is displayed. For example, if three record formats are on the display at the same time and you write another record to the display, the three record formats on the display are erased. Several DDS keywords, such as the OVERLAY keyword, let you control the displaying of records and input fields on input and output operations. For more information about these DDS keywords, see "Overlaying and Erasing Record Formats on a Display" on page 49 and "Clearing a Specified Number of Lines" on page 53.

In the following example, the fields are defined for a record format as follows:

- Fields from record format A occupy lines 1 through 4.
- Fields from record format B occupy lines 5 through 7.
- Fields from record format C occupy lines 8 through 10.
- Fields from record format D, which has the CLRL keyword specified for it, occupy lines 5 through 9.

In the following illustration, record formats A, B, and C are displayed first. When record format D is displayed, it replaces record formats B and C.

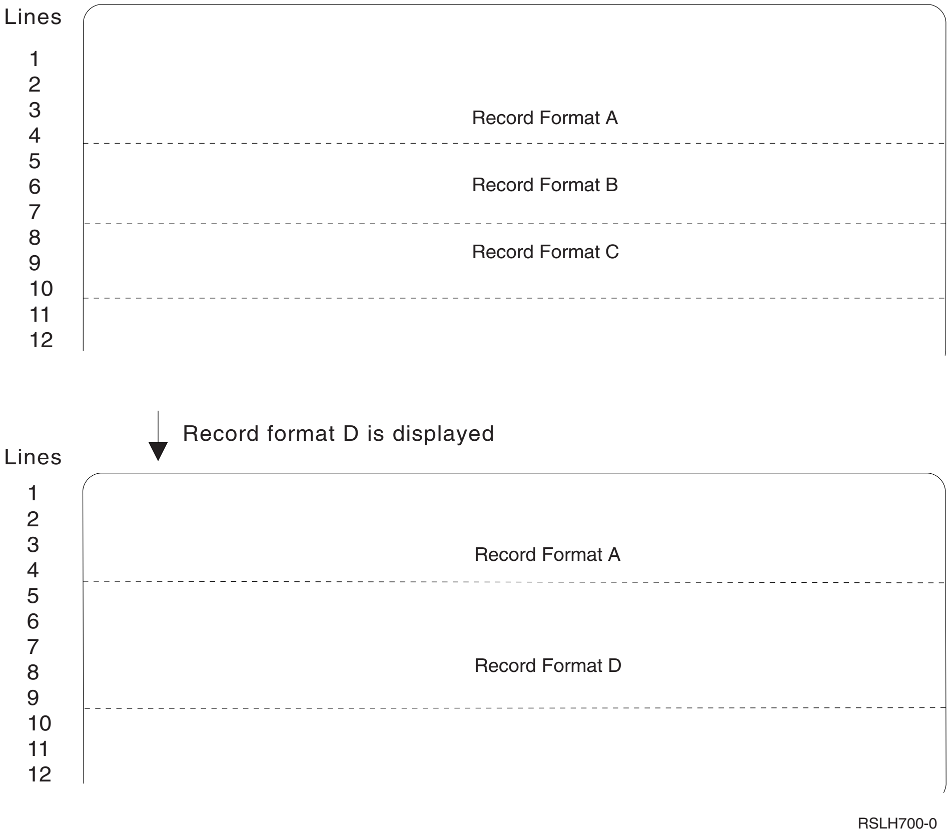


Figure 11. Replacing Record Formats

If record format D did not have the OVERLAY keyword specified for it, the following would have happened in the previous example:

- Record format A would also have been deleted.
- Lines 5 through 7 of record format B would have remained on the display. The data in any fields of record format B overlaid by record format D would have been changed. (see "Clearing a Specified Number of Lines" on page 53 for more information).

Deciding the Order of Record Formats Written to the Display

To improve performance, records containing input fields should be sent to the display station in the order in which they appear on the display.

In Figure 11, if record formats A and B both contain input fields and appear on the same display, record format A should be sent to the display first.

Overlaying and Erasing Record Formats on a Display

To avoid erasing the existing display when your program displays a new record format for output or to allow input, you can specify the OVERLAY keyword. The OVERLAY keyword causes only those records that are completely or partially overlapping to be erased; all other records remain on the display.

Note: The OVERLAY keyword does not prevent the screen from being erased if it is in effect for the first write operation after a file is opened unless the DDS keyword ASSUME is specified for any record format in the display file.

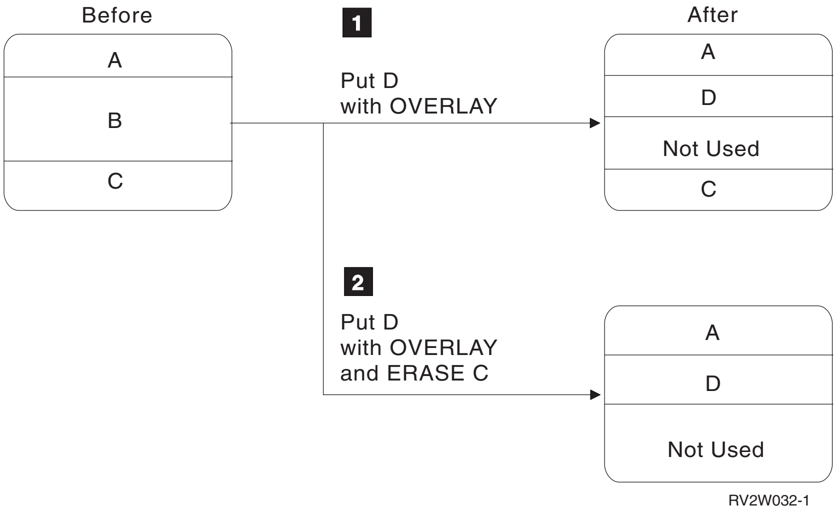
You can use the OVERLAY keyword to display information from your application that needs to be presented together but naturally falls into two or more pieces. For example, you could use one record format in your application to present information for a state at the top of a display and another record format to provide the information for a particular region within that state.

To place two or more records on the display at the same time, separate the write operations for your display from the read operations. Then, when you perform each write operation, the system takes the data from the record that you have provided it, combines it with the information specified in the record format, and places it on the display. You can lock the keyboard until the display is ready to perform the read operations by doing one of the following:

- Specify the LOCK keyword on all the record formats
- Specify *YES for the Defer Write (DFRWRT) parameter for the display file

You can use multiple record formats when you want to present lists of information in a subfile to the user. A subfile is a group of records that have the same record format and are read from and written to a display station in one operation. For more information about subfiles, see Chapter 4, “Displaying Groups of Records Using Subfiles,” on page 87.

To erase certain records from the display when you overlay records, use the ERASE keyword with the OVERLAY keyword. The following diagram shows the effect of the OVERLAY and ERASE keywords on an output operation:



1 Record format B is erased because record format D overlaps it, and record format D is displayed. Record format D did not use all of the space record format B previously used so it does not overlap record format C.

2 Record format B is erased because record format D overlaps it and record format C is erased because ERASE C is specified. Record format D is displayed, and part of the display is no longer in use.

Starting Your Record Format on a Specific Line

To start your record format on a specific line, use the starting line number (SLNO) keyword. On the SLNO keyword, you can specify one of the following:

- *The actual starting line number for the record format (a value from 1 to 27).* When you specify an actual line number, the system adjusts the line numbers for all fields in a record by the specified value minus 1.
- *A variable starting line number (*VAR)*, which allows you to specify a starting line number value in your high-level language program at run time. Depending on the value specified in your program, the following occurs:

*Table 9. Results of SLNO(*VAR) Values*

Value Specified	Results
0 or no value specified	A starting line number of 1 is assumed.

*Table 9. Results of SLNO(*VAR) Values (continued)*

Value Specified	Results
Value exceeds the number of lines on the screen or is a negative value	The system sends a message to the program and the I/O request is not performed.
The starting location for the field for at least one display size is row 1, column 1	A warning message (severity 10) is issued at file creation time. At run time, an error message is issued if the screen size being displayed contains a field starting in row 1, column 1, and the variable starting line number is set to 1 by the program.

Each programming language provides a different way to set and add to the starting line number. See the appropriate manual for the language you are using.

The system adjusts the line numbers for each field in the record format by the specified value minus 1. If the resulting line number exceeds the screen size, the field is not displayed. In addition, if any part of a field goes beyond the last line on the screen, the field is not displayed.

The SLNO keyword cannot be used in a record format that contains the record-level keywords ASSUME, KEEP, USRDFN, SFL, or SFLCTL, or in a display file that contains the file level keyword PASSRCD.

However, the SLNO keyword may be used with several other DDS keywords:

- If the CLRL keyword is used with the SLNO keyword and the CLRL keyword specifies a number of lines to clear, clearing starts with the starting line number on the SLNO keyword.
- If you use the SLNO(*VAR) keyword with the OVERLAY keyword but without the CLRL keyword and then write the record several times, each time with a different starting line number, the previous record is deleted before the new record is displayed.
- If the SLNO keyword is used with the PUTOVR, PUTRETAIN, ERRMSG, or ERRMSGID keyword in effect, the system checks the starting line number to determine if the previous output operation to the record had the same starting line number:
  - If the starting line number is the same, the action specified by the PUTOVR, PUTRETAIN, ERRMSG, or ERRMSGID keyword is performed.
  - If the starting line numbers are not the same, the PUTOVR, PUTRETAIN, ERRMSG, or ERRMSGID keyword is ignored, and the record format is displayed with the lines adjusted by the new value.

The following DDS shows an example of using the SLNO(*VAR) keyword:

```
|...+....1....+....2....+....3....+....4....+....5....+....6....+....7...
     A          R ORDENT
     A                                  1 36'ORDER ENTRY'
     A                                  3  2'Enter customer number:'
     A            CUST           5   B    +2
     A                                  5  2'Enter order number:'
     A            ORDNBR         6   B    +2
     A                                  7  2'ITEM NUMBER'
     A                                  7 18'DESCRIPTION'
     A                                  7 44'QUANTITY'
     A          R LINITM                    OVERLAY
     A                                      SLNO(*VAR)
     A                                      CLRL(*NO)
     A            ITEM           6 00   9  2
     A            DESCRP        20   0  9 18
     A            QTYORD         3 00   9 44
     A          R INPFMT                    OVERLAY
     A                                 23  2'Enter item number:'
     A            ITMNBR         6 0I     +2
     A                                    +5'Enter Qty:'
     A            QTY            3 0I     +2
```

*Figure 12. Sample DDS Source Showing Use of the SLNO(*VAR) Keyword*

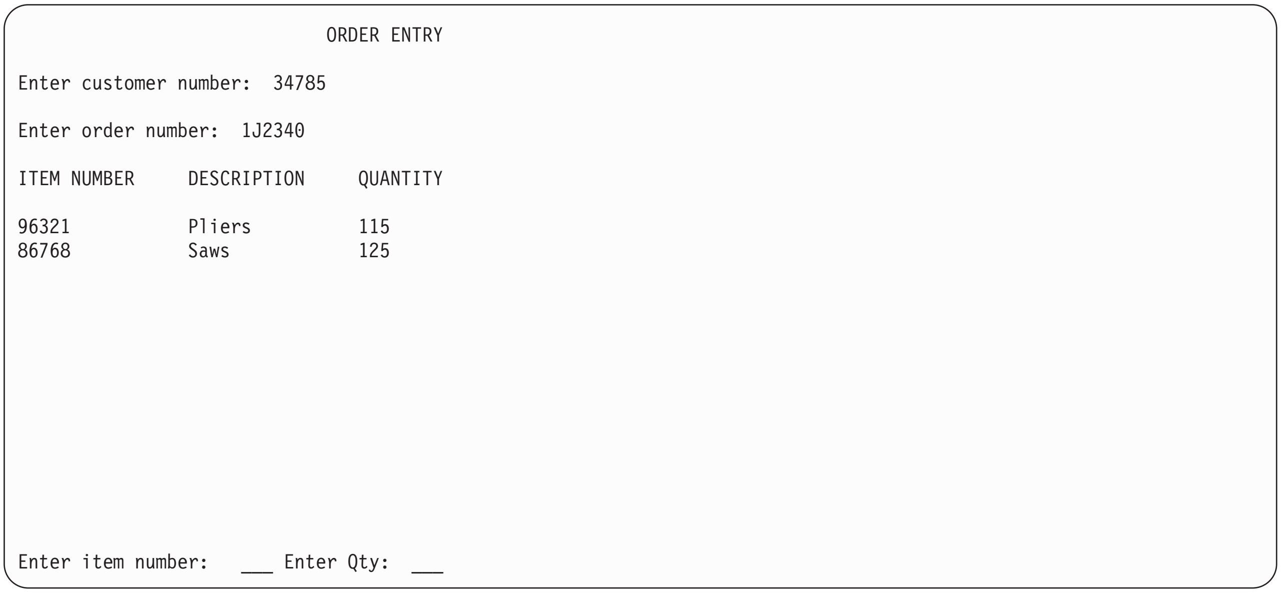
In this example, the record format ORDENT contains the prompt for an Order Entry display. When the user enters a customer number and an order number, the following occurs:

1. The program writes the record format INPFMT to the display, which allows the user to enter an item number and quantity ordered.
2. After the user enters the item number and the quantity, the program retrieves the description of the item from a file and writes the record format LINITM to the display.
3. The program writes the INPFMT record format to the display to allow the user to enter another item number.

The design of this display allows the user to enter the item number and quantity on the same line. As a line item is entered, the program uses the LINITM record format to build the order on the display. The SLNO(*VAR) keyword is used so the program can add a line to the display each time the LINITM record format is written. The CLRL(*NO) keyword has to be specified on the LINITM record format so that the previous record is not deleted when a new record is written.

When the LINITM record format is first written to the display, the value of *VAR is 1 so the fields are displayed on line 9. On each successive output operation to this record format, the program adds 1 to the starting line number so that a new line item is added to the display.

After the user enters two item numbers and quantities, the display looks like this:



```
                             ORDER ENTRY

Enter customer number:  34785

Enter order number:  1J2340

ITEM NUMBER     DESCRIPTION     QUANTITY

96321           Pliers          115
86768           Saws            125










Enter item number:   ___ Enter Qty:  ___
```

The SLNO keyword is most efficient when you want the user to always enter data on the same line and yet build a display of previously entered records, as shown in the preceding example. However, for a typical inquiry function where you want to display more than one record at a time, the use of a subfile is more efficient.

Clearing a Specified Number of Lines

To clear a certain number of lines on the screen before you write a record format to the screen, use the clear lines (CLRL) keyword. You can specify the CLRL keyword even when the record contains no fields that are displayed. Clearing begins with the starting line number, and the value specified on the CLRL keyword determines the number of lines to be cleared (any value from 1 to 27). The starting line number is determined as follows:

- If the SLNO keyword is not specified, the field locations determine the starting line number.
- If the SLNO(nn) keyword is specified, nn is the starting line number.
- If the SLNO(*VAR) keyword is specified, the starting line number defaults to 1 at the time the display file is created and can be changed by the application program at the time it is run.

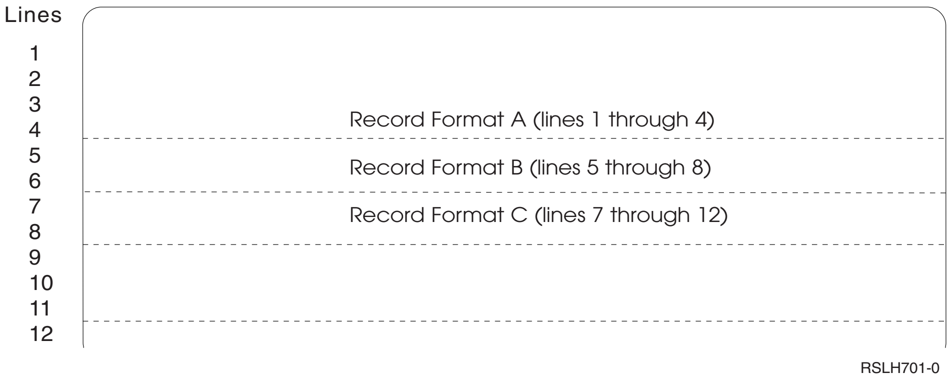
You can also specify the following values to clear specific lines:

Value	**Lines Cleared**
***END**	All lines from the starting line to the end of the display
***NO**	Only the lines of the display that are used by the overlapping record format
***ALL**	All lines of the display. Since the default action is to clear all the lines of the screen, you do not normally have to specify CLRL(*ALL) unless you also specify a DDS keyword, such as USRDSPMGT, that changes this default.

Note: When you use the CLRL keyword, you should specify *YES for the RSTDSP parameter on the CRTDSPF or CHGDSPF command; otherwise, data on the display may be lost if the file is suspended.

You can use the CLRL(*NO) keyword to prevent an overlapped record from being deleted when the overlapping record is written to the display. If you use this keyword, any records being displayed that are to be overlapped are not deleted from the screen; the new record overlays them entirely or partially. There is a performance advantage to using CLRL(*NO) if you have a display that contains constants and data that is repeatedly sent to the screen. By sending the constants as a separate format and by using

CLRL(*NO) for the format containing only the data, you can reduce the time required to send the record format to the display. For example:


Lines
1
2
3
4
5
6
7
8
9
10
11
12
Record Format A (lines 1 through 4)
Record Format B (lines 5 through 8)
Record Format C (lines 7 through 12)
RSLH701-0

If CLRL(*NO) is specified on record format C, all fields of record format B not overlapped by C remain on the screen when record format C is written to the screen. If the OVERLAY or PUTOVR keyword were used for this same situation, record format B would be deleted when record format C is written to the screen because record format C overlaps record format B.

The following considerations apply to the CLRL keyword when used with other DDS keywords:

- If the CLRL keyword is specified in a record format with input-capable fields, any input-capable fields in the overlapped records are no longer input-capable. Fields in all other record formats that are not overlapped remain input-capable. If you do not want these fields to remain input-capable, you should use the PROTECT keyword on the record format along with the CLRL(nn) keyword.
- Records with the CLRL keyword and with no input-capable fields are assumed to be at line 0. Thus, if the CLRL(nn) keyword is specified in a record format that has no input-capable fields, all records already on the display remain on the display and their input-capable fields remain input-capable. Because records that start at line 0 are not known to the system, the ROLLUP and ROLLDOWN keywords do not work for these records. Also, these records may not be cleared completely when they are overlapped by other records that have the OVERLAY keyword specified. The lines needed for the overlapping record are cleared whereas the lines not needed for the overlapping record remain on the screen.
- The CLRL(nn) keyword is not allowed in a record format with the record-level keywords ASSUME, KEEP, USRDFN, SFL, or SFLCTL, or in a display file with the file level keyword PASSRCD.
- The CLRL(nn) keyword is ignored if either the ERRMSG or ERRMSGID keyword is in effect.
- If the CLRL(nn) keyword is used and the PUTOVR or PUTRETAIN keyword is in effect, the clearing of lines may conflict with the PUTOVR or PUTRETAIN function. The PUTOVR or PUTRETAIN keyword requires that the fields being overridden be on the display whereas the CLRL(nn) keyword may clear those fields first. If a record becomes unavailable for input because of the CLRL(nn) keyword, the input-capable fields remain input-capable if the PUTOVR keyword is in effect. However, the system issues a message if the program attempts to read such a record. Although the CLRL(nn), CLRL(*NO), and CLRL(*END) keywords imply the OVERLAY keyword, the following example illustrates the differences between the CLRL and OVERLAY keywords:

```
|...+....1....+....2....+....3....+....4....+....5....+....6....+....7...
     A          R RECORD1
     A            FLD1          10   I  4  5
     A                                  5 21'Enter employee number'
     A          R RECORD2A                  OVERLAY
     A            FLD3           1   B  5  2
     A                                  6  2'Required field'
     A                                  7  2'Enter 1, 2, or N'
     A            FLD4          19   O  8  2
     A          R RECORD2B                  CLRL(4)
     A            FLD5           1   B  5  2
     A                                  6  2'Required field'
     A                                  7  2'Enter 1, 2, or N'
     A            FLD6          19   O  8  2
     A          R RECORD3                   OVERLAY
     A            FLD7          10   O  8 15
     A             .             .
     A             .             .
     A             .             .
     A            FLD8          10   B 12  4
     A          R RECORD4                   CLRL(*NO)
     A            FLD9          42   I 11  2
```

Figure 13. Sample DDS Source Showing Difference between CLRL and OVERLAY

The following results occur if the program performs the output operations on the record format in the following order:

Table 10. Results from CLRL Example

Order of Record Formats	Results
RECORD1 RECORD3 RECORD2A	Lines 4 through 12 are deleted when RECORD2A is written to the display because RECORD2A overlaps RECORD1 and RECORD3, and only the OVERLAY keyword is specified for RECORD2A.
RECORD1 RECORD3 RECORD2B	Lines 5 through 8 are cleared before RECORD2B is written to the display because the CLRL(4) keyword is specified. FLD1 in RECORD1 and any input-capable fields in RECORD3 (lines 9 through 12) remain on the screen but are no longer input-capable because part of RECORD1 and RECORD3 is overlapped by RECORD2B.
RECORD1 RECORD3 RECORD4 RECORD2A	RECORD1 remains on the screen when RECORD3 is written to the screen because the OVERLAY keyword is specified in RECORD3. When RECORD4 is written to the screen, it uses part of line 11, which is also used by RECORD3, and because CLRL(*NO) is specified in RECORD4, RECORD3 remains on the screen. However, the system is no longer aware that RECORD3 is on the screen so when RECORD2A is written, only lines 4 through 8 are cleared; the part of RECORD3 below line 8 remains on the screen.

Rolling Data between Two Lines on a Display

If you are using a high-level language program, you can roll the data between two lines on the display up or down by specifying the allow roll (ALWROL) keyword. The lines vacated by the rolled data are set to nulls and another record format can be written to those lines.

In your program, you must specify the following:

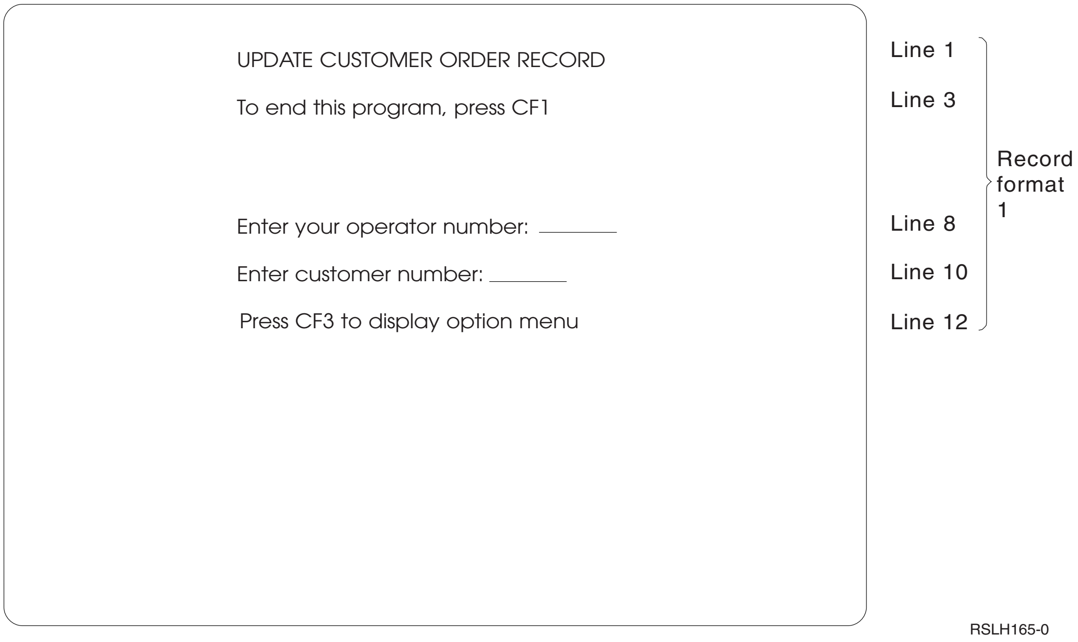
- *The starting line number and the ending line number of the lines to be rolled.* The start and end line numbers define a window on the screen.
- *The number of lines to be rolled.* If the number of lines to be rolled is positive, the data is rolled up. If the number of lines to be rolled is negative, the data is rolled down.
- *Whether the roll is to be up or down.*

In the window, the lines of data are rolled up (or down) by the number of lines you specified in your program. The data rolled off the window is gone. The input-capable fields of any record format partially or completely within the window are no longer input-capable. After the roll, your program cannot issue an input operation to any record format within the window.

The following example shows a display before a program-controlled roll occurs, and the same display after a program-controlled roll occurs. The following is specified in the program:

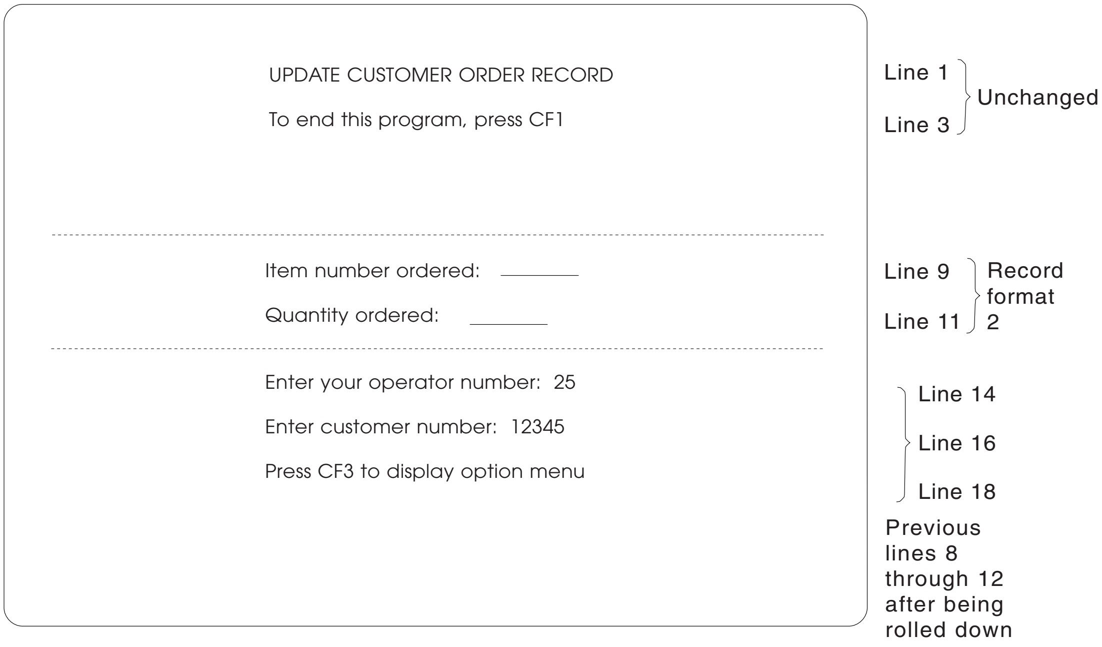
- The starting line number is 8
- The ending line number is 18
- The number of lines to be rolled down is 6.

Display before the Roll Operation



```
UPDATE CUSTOMER ORDER RECORD

To end this program, press CF1




Enter your operator number: ________

Enter customer number: ________

Press CF3 to display option menu
```

Display after the Roll Down Operation



UPDATE CUSTOMER ORDER RECORD

To end this program, press CF1

Item number ordered: ________

Quantity ordered: ________

Enter your operator number: 25

Enter customer number: 12345

Press CF3 to display option menu

The ALWROL keyword cannot be used with the file level keyword PASSRCD or with the following record-level keywords: KEEP, ASSUME, USRDFN, SFL, or SFLCTL.

If the ERRMSG, ERRMSGID, PUTOVR, or PUTRETAIN keyword is in effect for the same output operation in which the ALWROL keyword is in effect, the system issues message CPF5014. If an ERRMSG, ERRMSGID, PUTOVR, or record level PUTRETAIN keyword is not in effect, the message is not issued. However, if the PUTRETAIN keyword is specified at the field level with option indicators, the message (CPF5014) is issued if the option indicators for the PUTRETAIN keyword are on or off.

Overriding the Attributes or the Content of a Field

To send only some of the data and attributes of a record to the display, use the following keywords:

- Put with explicit override (PUTOVR)
- Override data (OVRDTA)
- Override attribute (OVRATR)

By sending less data or attributes, you can shorten the response time at the display, especially for remotely attached displays.

When the PUTOVR keyword is specified, the following occurs:

- The display attributes are overridden for those fields with the OVRATR keyword in effect.
- The data content is overridden for those fields with the OVRDTA keyword in effect.
- The output operation functions as if the OVERLAY keyword were also in effect, even if the OVERLAY keyword is not specified.

The PUTOVR keyword cannot be specified in a record format that contains the PUTRETAIN keyword nor can it be used for subfile records.

The display attributes that can be overridden by the OVRATR keyword are:

CHECK(ER) End of Record

CHECK(ME) Mandatory enter

DSPATR(MDT)
Set on modified data tag

DSPATR(PR) Protect

DSPATR(BL) Blink

DSPATR(CS) Column separator

DSPATR(HI) High intensity

DSPATR(ND) Nondisplay

DSPATR(PC) Position cursor

DSPATR(RI) Reverse image

DSPATR(UL) Underline

DUP Dup key capable

The following is an example of the PUTOVR keyword.

```
|...+....1....+....2....+....3....+....4....+....5....+....6....+....7...
     A          R ITMRVW
     A                                      PUTOVR
     A                                  1 35'ITEM REVIEW'
     A                                  3  2'Item number:'
     A            ITMNBR         5   B    +2
     A                                  5  2'Item description:'
     A            ITMDSC        20        +2
     A  10                                  OVRDTA
     A                                  7  2'Item price:'
     A            ITMPRC         8  2     +2
     A  15                                  OVRDTA
     A                                  9  2'Warehouse location:'
     A            WHSLOC         3        +2
     A  20                                  OVRDTA
     A                                 11  2'Quantity on hand:'
     A            QTYOH          5  0     +2
     A  25                                  OVRDTA
     A                                      OVRATR
     A N25 30                               DSPATR(HI)
```

Figure 14. Sample DDS Source Showing Use of the PUTOVR Keyword

The DDS describes a display that allows the user to enter an item number, and to review the item description, the item price, the warehouse location, and the quantity on hand:

1. On the first output operation, all fields are sent to the display, and all option indicators are off. The PUTOVR keyword is ignored because the record is not already on the screen. On the first output operation, the current field values in the program are displayed for the output fields. If your program has not set any of these fields, the values will be whatever the high-level language used to initialize the output buffer.

   If an output-capable field must always have a specific value on the first output operation, you can use the DFT or DFTVAL keywords to initialize the field to that value. When used on an output-capable field with the PUTOVR and OVRDTA keywords the DFT keyword causes the system to place the default value rather than the program value on the display when the record is first placed on the display.
2. The user enters an item number. The program sets on indicators 10, 15, 20, and 25 and issues a write-read operation to display the output fields. On the write operation, the PUTOVR keyword is in effect because the record is already on the screen. Because the OVRDTA keyword is specified on the ITMDSC, ITMPRC, WHSLOC, and QTYOH fields and because their option indicators are on, these fields are the only data sent to the display.

If the user enters another item number and the data for a field already displayed does not change, the program sets off the option indicator and does not display that field again. For example, assume that for the second item number, the WHSLOC is the same as for the first item number. On the output operation to display the information for the second item number, the program sets off indicator 20. Therefore, the only fields sent to the display are ITMDSC, ITMPRC, and QTYOH because indicators 10, 15, and 25 are on.

For the QTYOH field, the program can change the attributes for the field without changing the data by setting off indicator 25 and setting on indicator 30 before the output operation.

You can use the option indicators on the OVRDTA keyword to control which fields are sent to the display. If no option indicators are used, all fields with the OVRDTA keyword specified are sent to the display on each output operation because the OVRDTA keyword is in effect when the PUTOVR keyword is in effect. In the preceding example, if no option indicators were used, all four fields would be sent to the display on each output operation. You can also use the same indicator to control more than one field.

An alternative design for this same application is to use two record formats and send the constants to the display in one record format and the variables in the other record format. You would have to use the CLRL(*NO) keyword to prevent the record format containing the constants from being erased. However, if the record format is already on the display, the use of the PUTOVR keyword provides the most efficient approach.

The following examples illustrate how to use the PUTOVR keyword for efficient coding:

```
|...+....1....+....2....+....3....+....4....+....5....+....6....+....7...
     A          R PROMPT
     A                                      CF03(91 'Return')
     A                                      PUTOVR
     A                                      ERASEINP
     A                                      OVERLAY
     A                                  1 28'Efficient Coding Example'
     A                                  3  2'FLD1'
     A            FLD1           5   I  3  7
     A                                  5  2'FLD2'
     A            FLD2           5      5  7
     A                                      OVRDTA
     A                                  7  2'FLD3'
     A            FLD3           5      7  7
     A  15                                  OVRDTA
     A                                  9  2'FLD4'
     A            FLD4           5      9  7
     A                                      OVRDTA
     A  16                                  DSPATR(HI)
     A                                  11 2'FLD5'
     A            FLD5           5      11 7DFT('ABCDE')
     A                                      OVRDTA
     A  17                                  DSPATR(HI)
     A                                 13  2'Constant 1'
     A                                      OVRATR
     A  18                                  DSPATR(BL)
     A                                 15  2'Constant 2'
     A                                      OVRATR
     A N19                                  DSPATR(ND)
     A                                 17  2'Constant 3'
     A  20                                  OVRATR DSPATR(RI)
```

Figure 15. Sample DDS Source Showing Efficient Use of PUTOVR Keyword

In the preceding example, the following happens:

1. If the record format is not currently on the display, the PUTOVR, OVRATR, and OVRDTA keywords are ignored when the record format is displayed. On subsequent output operations when the record format is already on the display and the PUTOVR keyword is in effect, only the fields or constants defined with the OVRATR or OVRDTA keyword are sent to the display. The ERASEINP keyword is used because it is the most efficient way to clear all input fields, and the OVERLAY keyword is used because it is required with the ERASEINP keyword.
2. FLD1 is an input field that is cleared each time the record format is displayed.
3. FLD2 is sent to the display each time the record format is displayed because its associated OVRDTA keyword is unconditionally specified.
4. FLD3 is sent to the display on the first output operation. On subsequent output operations, FLD3 is not sent to the display unless indicator 15, which is used to condition the OVRDTA keyword, is on.
5. FLD4 is sent to the display on each output operation because its associated OVRDTA keyword is unconditionally specified. When the OVRDTA keyword is in effect, the attributes for the field are always sent to the display. Indicator 16 is used to control the DSPATR(HI) keyword for FLD4.
6. On the first output operation, the default value of ABCDE appears in FLD5. On subsequent output operations, a value from the program is displayed in FLD5 because its associated OVRDTA keyword is unconditionally specified. Indicator 17 is used to control the DSPATR(HI) keyword for FLD5.
7. *Constant 1* is always displayed, but it is only sent to the display on the first output operation. However, the attributes for the field are sent to the display each time the record format is written, and option indicator 18 is used to control whether the field blinks.
8. *Constant 2* is sent to the display only on the first output operation. However, the attributes for the field are sent to the display each time the record format is written, and if option indicator 19 is off, *Constant 2* will not be displayed.
9. *Constant 3* is sent to the display only on the first output operation. However, the attributes for this field are not sent to the display on subsequent output operations unless indicator 20 is on. If option indicator 20 is on when an output operation is done, *Constant 3* is displayed in reverse image, and it will continue to appear in reverse image regardless of the status of indicator 20 on subsequent output operations.

The following example shows how the PUTOVR keyword can be used for an application in which the user enters some information common to a group of records and then repeatedly enters detailed information relating to specific records in the group.

```
|...+....1....+....2....+....3....+....4....+....5....+....6....+....7...
     A          R HEADING                   TEXT('Header Display')
     A                                      SETOF(88 'ERASEINP CTL')
     A                                      CF03(91 'Return')
     A                                  1  2'HEADING INFORMATION'
     A                                      DSPATR(HI)
     A                                  1 60'CF3-End of Program'
     A                                  2 60'CF2-New heading'
     A            HDING          5   I    +2
     A          R DETAIL                    TEXT('Detail display')
     A                                      OVERLAY
     A                                      PUTOVR
     A                                      PROTECT
     A  88                                  ERASEINP
     A                                      CF02(92 'New header')
     A                                  8  2'DETAIL DISPLAY'
     A                                      DSPATR(HI)
     A                                 10  2'Input'
     A            FLDA           5   I    +2
     A                                 12  2'Output'
     A            FLDB           5        +2DFT(' ')
     A                                      OVRDTA
```

Figure 16. Sample DDS Source Showing Another Use of PUTOVR Keyword

In the preceding example, the following happens:

1. The program displays the HEADING record format, and then performs an input operation to the record format to receive the HDING field as input. The SETOF keyword in the HEADING record format sets off indicator 88, which is used to condition the ERASEINP keyword in the DETAIL record format.
2. The program then displays the DETAIL record format. Because the OVERLAY keyword is in effect, the HEADING record format remains on the display. The PROTECT keyword is also in effect so the input field (HDING) in the HEADING record format is protected. Therefore, the user cannot change this field when the DETAIL record format is displayed.
3. The ERASEINP keyword is conditioned by option indicator 88. Because indicator 88 is off the first time the DETAIL record format is displayed, the ERASEINP keyword is not in effect. On subsequent output operations, indicator 88 is set on and the ERASEINP keyword is in effect. Therefore, FLDA is cleared on subsequent output operations. The option indicator is used on the ERASEINP keyword so that it is not in effect the first time the DETAIL record format is displayed. Because the ERASEINP keyword is processed before the PROTECT keyword, it would clear the HDING field in the HEADING record format if it were in effect the first time the DETAIL record format is written.
4. FLDB is an output field that is sent to the display on each output operation because the OVRDTA keyword is specified unconditionally. The DFT keyword with a value of blanks is used so the field will not contain any data the first time the DETAIL record is displayed for a group.

Erasing All Unprotected Input and Output/Input Fields on the Display

To erase all unprotected input-capable fields, use the erase input (ERASEINP) keyword. The ERASEINP keyword can only be used with the OVERLAY keyword.

To erase all unprotected input-capable fields that have their modified data tags on, specify *MDTON for the ERASEINP keyword. To erase all unprotected input-capable fields whether their modified data tags are on or not, specify *ALL for the ERASEINP keyword.

The ERASEINP keyword can improve response time because it clears fields rather than sends blanks to the display. If the fields erased at the display do not have their modified data tags set on for the next read operation, data is returned for those fields from the input save area. This is data saved by the system from the previous return of the field from the display station.

You can use the INZINP keyword at the record level with ERASEINP(*ALL) and PUTOVR to initialize the input save area without sending data for the cleared fields to the display.

Resetting Modified Data Tags Associated with Records on the Display

To reset the modified data tags, use the modified data tag off (MDTOFF) keyword. The MDTOFF keyword, which can only be used with the OVERLAY keyword, is processed before the next record is displayed.

To reset only the modified data tags of the unprotected fields, specify *UNPR for the MDTOFF keyword. To reset the modified data tags of all input-capable fields, specify *ALL for the MDTOFF keyword.

Keeping a Record or Field on a Display

The PUTRETAIN keyword is used to reduce the number of characters sent to the display. This keyword can only be used with the OVERLAY keyword and can be used to change only the display attributes of a field. Except for not sending data, all other functions are supported when the PUTRETAIN keyword is specified.

Using the PUTRETAIN keyword at either the record format level or the field level can cause fields from this record which were previously written to the display to remain on the display even if they are not selected for this write operation. To avoid this, you can use the PUTRETAIN keyword at the field level and define the field twice: once with option indicators as you want it to appear in the display, and once with no option indicators and as a constant with a value of blanks. If the first field is not selected, the second field is. The second field is displayed so the blanks erase the contents of the field that is not selected.

Note: The ERRMSG and ERRMSGID keywords function as if the PUTRETAIN keyword were specified at the record format level. That is, no fields are sent to the display, no field attributes for other fields are changed, and no command keys are changed when the ERRMSG and ERRMSGID keywords are in effect.

The following is an example of the PUTRETAIN keyword used at the record format level. The following DDS describes a student search menu having three options. The option selected is highlighted. For example, if option 1 is selected, the character string *1. By number* is highlighted.

```
|...+....1....+....2....+....3....+....4....+....5....+....6....+....7...
     A          R SELECT                    OVERLAY
     A                                      PUTRETAIN ERASEINP
     A N44                                1  2'STUDENT SEARCH MENU'
     A N44                                3 10'1. By number'
     A  10                                  DSPATR(HI)
     A N44                                4 10'2. By name'
     A  11                                  DSPATR(HI)
     A N44                                5 10'3. By address'
     A  12                                  DSPATR(HI)
     A N44                               10  2'Select the number of the item to  +
     A                                      search by:'
     A            INPUT          1   I 10 47
     A  44                                  DSPATR(RI)
```

Figure 17. Sample DDS Source Showing Use of the PUTRETAIN Keyword

The following happens:

1. On the first output operation, all fields are sent to the display, and all option indicators are off. The PUTRETAIN keyword is ignored because the record is not already on the display.
2. The user selects item 1, 2, or 3. When the program receives the input, it sets on indicator 10, 11, or 12, depending on which item is chosen. If anything other than item 1, 2, or 3 is chosen, the program sets on indicator 44.

On the next output operation, field 1, 2, or 3 is highlighted, or the input field is in reverse image, depending on which indicator is on.

The data for all fields is not resent to the display but the field attributes are resent. No data is sent for constants. To resend attributes for each output field or constant, 4 bytes are needed. To resend attributes for each input-capable field, 9 bytes are needed. By using the PUTRETAIN keyword, you reduce the number of characters sent to the display by 96, from 138 to 42. (These numbers do not include protocol control characters needed to frame data.)

The ERASEINP keyword causes the user’s selection to be erased.

The following is an example of the PUTRETAIN keyword used at the field level. Here, the PUTRETAIN keyword is used to keep input that is not valid and to reduce the number of characters sent to the display. The following DDS describes a display containing an item’s name, color, shape, and size, and asks for quantity. The user can change the values for color, shape, and size.

```
|...+....1....+....2....+....3....+....4....+....5....+....6....+....7...
     A          R CHANGE                    OVERLAY
     A                                  1  2'CHANGE MENU'
     A N43                              3  2'Change the underlined fields to   +
     A                                      change the description.'
     A                                  4  2'Item:'
     A            ITEM          20     O  4 12
     A  44                                  PUTRETAIN
     A                                  5  2'Quantity:'
     A            QTY            4Y 0I  5 12
     A  44                                  PUTRETAIN
     A  09                                  DSPATR(BL PC)
     A                                  6  2'Color:'
     A            COLOR         10     B  6 12
     A  44                                  PUTRETAIN
     A  10                                  DSPATR(BL PC)
     A                                  7  2'Shape:'
     A            SHAPE         10     B  7 12
     A  44                                  PUTRETAIN
     A  11                                  DSPATR(BL PC)
     A                                  8  2'Size:'
     A            SIZE          10     B  8 12
     A  44                                  PUTRETAIN
     A  12                                  DSPATR(BL PC)
     A  44                              9  2'Choice:'
     A  44        CHOICE        20     O  9 12
     A  15                              9 12'                             '
```

Figure 18. Sample DDS Source Showing Use of the PUTRETAIN Keyword

The following happens:

1. On the first output operation, all indicators are off, so all the constants and the fields except CHOICE and the constant field following CHOICE are sent to the display.
2. The user enters a quantity. The program sets on indicator 43. When the next output operation occurs, indicator 43 prevents the second constant field from being resent.
3. When the user is to enter the quantity for another item, the program issues another output operation. The attributes for the fields QTY, ITEM, COLOR, SHAPE, and SIZE are sent to the display. Field selection prevents the CHOICE field from being sent to the display.

   At least one field, in this case QTY, must be kept to prevent the entire record area from being erased.
4. If the user enters a quantity, color, shape, or size that is not valid, indicator 44 is set on so that the input fields (QTY, COLOR, SHAPE, and SIZE) are not erased and so that the output field CHOICE is sent to the display. In addition, the appropriate indicator, 9, 10, 11, or 12, is set on so that the input field in error blinks and the cursor position is below the field. (The CHOICE field would show the user valid choices for the field in error.)

5. The CHOICE field and a constant field of blanks are defined for the same location. After the user enters valid data, indicator 15 is set on, indicator 44 is set off, and the constant field initializes the CHOICE field to all blanks.

Deferring the Write Operation Until a Read Request is Made

The DFRWRT parameter on the Create Display File (CRTDSPF) or Change Display File (CHGDSPF) command allows you to specify how the system is to handle write operations. If you specify DFRWRT(*NO), the program does not regain control until the write operation has displayed the data and updated the input/output feedback area.

If you specify the default of DFRWRT(*YES) for the file, the program regains control after the output record is processed. The program can then use the record area where the output was stored to start processing the next write or write-read operation. The data is actually sent to the display only when a read or write-read operation is issued or when the FRCDTA DDS keyword is in effect for a write-only operation.

Using DFRWRT(*YES) on a display file improves systems performance; however, DFRWRT(*YES) should not be used in the following circumstances:

- If you want to find out immediately if the write operation was successful. An error associated with a write operation for a file with DFRWRT(*YES) specified is issued only when the data is actually sent to the display.
- If the time between the write operation and the read or write-read is long. For example, if the program does several database operations after the write operation (before it issues a read or write-read operation), the user will not see the data while the database operations are performed.
- If the file is closed after the write-only operation and the KEEP keyword is not specified. If the display file has the DDS keyword KEEP specified in any of its records, the data accumulated from the write-only operation is displayed when the file is closed. However, if the KEEP keyword is not specified, the data may never be displayed.

The DFRWRT parameter has no effect on the following:

- Write operations using user-defined data streams
- Write operations to display files that use program-described data
- Record formats for which the FRCDTA DDS keyword is in effect

Specifying Default Values for Fields

Both DFT and DFTVAL keywords are used to specify the default values to be displayed for fields. However, there are differences between the way the two are used.

The DFT keyword can be used with constant, input, output, and output/input fields and cannot be optioned. When it is used with output or output/input fields the OVRDTA and PUTOVR keywords must also be specified. If the record is not on the display, this combination of keywords will cause the default value to be placed on the screen. If the record is already on the display, the PUTOVR keyword is in effect and the data from the program appears on the display rather than the default value.

The DFTVAL keyword can be used only on output and output/input fields and can be optioned. If it is in effect on an output operation, the value from the keyword is placed in the field, rather than the value from the program. If the record is on the display and the PUTOVR and OVRDTA keywords are in effect, the program value is used rather than the default value.

The DFT and DFTVAL keywords may not be specified on the same field.

Indicating Which Mode to Display Records

Some display stations, for example the 3180-2 display station, support an alternate screen size. You can specify this alternate size using the DSPMOD keyword. The DSPMOD keyword indicates, for a particular record, which mode is used to display the record. Any record that does not have the DSPMOD keyword specified for it is displayed in the default display mode. The default display mode is the first of the *DS3 or *DS4 display sizes on the DSPSIZ keyword.

The DSPMOD keyword is only valid when both *DS3 and *DS4 are specified on the DSPSIZ keyword. This keyword is valid only at the record level. Option indicators are allowed. The DSPMOD keyword may not be duplicated in a record.

Note: The capability to display in 27 by 132 mode is allowed on 3180-2, 3197, 3477 Models FA, FC, FD, FE, FG, FW, and 3487 Models HE, HD, HW, HC display stations attached to a local display station controller, or remotely attached to a 5294 or 5394 controller. The DSPMOD keyword is ignored unless these controllers are used.

For example, the following DDS would display RECORD1 in 27 by 132 mode, and RECORD2 in 24 by 80 mode (the default mode set up by the DSPSIZ keyword). RECORD3 will be displayed in 27 by 132 mode if option indicator 03 is on, or in 24 by 80 mode if option indicator 03 is not on.

```
|...+....1....+....2....+....3....+....4....+....5....+....6....+....7...
     A                                      DSPSIZ(*DS3 *DS4)
     A          R RECORD1                   DSPMOD(*DS4)
     A          R RECORD2
     A          R RECORD3
     A  03                                  DSPMOD(*DS4)
```

Figure 19. Sample DDS Source Showing Use of the DSPMOD Keyword

The use of the DSPMOD keyword can cause the display mode to be changed dramatically. Caution should be used when specifying the DSPMOD keyword. When a record with DSPMOD active causes the mode to be changed, all records currently on the display are cleared and deleted from the active record table. The record with DSPMOD active is then sent to the display. The mode for this record is maintained on the display as long as the DSPMOD keyword is active. Setting DSPMOD off or a write operation to another record without DSPMOD causes the display mode to be placed back in the primary display screen size for the display station.

Using the previous sample DDS source, the DSPMOD keyword gives the following results if records are written to the screen in the following order:

- RECORD1 is displayed in *DS4 mode.
- The display screen is cleared and RECORD2 is displayed in *DS3 mode.
- If indicator 03 is off, RECORD3 is displayed in *DS3 mode. RECORD2 remains on the display if the OVERLAY keyword is specified.
- If indicator 03 is on, RECORD2 is cleared and RECORD3 is displayed in *DS4 mode.

Note: When changing display modes, the displayed subfile data is removed from the display. However, the subfile data is not cleared from the subfile table.

The following keywords are ignored if the display modes have changed:

Table 11. Keywords Ignored If Display Modes Are Changed

Keywords	Additional Information
ALWROL	When a record is not on the screen, it cannot be rolled.
ASSUME	The records with the ASSUME keyword remain on the screen when the file is opened. When the display modes change, the records on the screen are cleared. This is similar to specifying the ASSUME keyword without the OVERLAY keyword. The display size of the file with the KEEP keyword must equal the display size of the file with the ASSUME keyword.

Table 11. Keywords Ignored If Display Modes Are Changed (continued)

Keywords	Additional Information
CLRL	All lines will be cleared by a change in display mode.
ERASEINP/INZINP ERRMSG ERRMSGID KEEP OVERLAY PROTECT PUTOVR	When the display modes change, the record is displayed with PUTOVR not in effect, even if the record was on the screen before the display modes changed.
PUTRETAIN SFLMSG SFLMSGID	-

Positioning the Cursor after an Output Operation

You can specify where you want the cursor positioned after an output operation by using the CSRLOC or DSPATR(PC) keyword.

On the record-level keyword CSRLOC, you can specify the names of two 3-byte zoned decimal hidden fields that contain the exact line and position for the cursor location. With the CSRLOC keyword, you can position the cursor outside the record you are displaying.

The field-level keyword DSPATR(PC) positions the cursor at the first position of the field after the record is written. However, if the OVERLAY keyword is specified at the record level, the cursor position may be lost after subsequent write operations.

Note: The cursor is not positioned if the keyboard is unlocked before the output operation.

If both the CSRLOC and DSPATR(PC) keywords are specified, the cursor is positioned by the CSRLOC keyword. If several fields have DSPATR(PC) keyword specified, the cursor is positioned at the first field for which the DSPATR(PC) keyword is specified.

If the CSRLOC and DSPATR(PC) keywords are not specified, the cursor is positioned at the first input-capable field on the display. If there is no input-capable field, the cursor is positioned in the upper-left corner of the display. However, if the CSRLOC and DSPATR(PC) keywords are not specified for records containing input-capable fields, the cursor position may be lost if the record is suspended and then restored. For example, the cursor position may be lost if the F1 (Help) key is pressed after the record is displayed.

Other DDS functions can affect the write operation. For a write operation to a user-defined data stream (USRDFN keyword), the functions performed are determined by the user-supplied controls.

Returning the Cursor Position to an Application

You can determine where the cursor was positioned on input by using the RTNCSRLOC (return cursor location) keyword.

This keyword may be specified in either of two formats:

- Return the name of the record and field in which the cursor is currently positioned. Optionally, a third parameter may be specified that will contain the relative cursor position within the field.
- Return the row and column position of the cursor relative to the display. Optionally, two additional parameters return either of the following:
  - The row and column position of the cursor relative to the active window (if one exists)
  - The location of the cursor at the beginning of a two event mouse button call.

The parameters of these formats are described in the **DDS** topic in the iSeries Information Center.

Returning the Cursor Position Within a Subfile to an Application

On input, you can determine where the cursor is located in a subfile by using the SFLCSRRRN (subfile cursor relative record number) keyword. The relative record number on which the cursor is positioned is returned in the hidden field specified as the parameter on the keyword. The field must be defined in the record format as a signed numeric (S in position 35) with a length of 5 with zero decimal places. Also, it must be a hidden field (H in position 38).

Returning the Mode of a Subfile to an Application

You can use the SFLMODE (subfile mode) keyword to determine whether the subfile was in folded or truncated mode on input. The mode parameter is required. The SFLMODE keyword is only valid for subfile control records and the SFLCTL keyword must be specified.

The field specified for the mode parameter is defined in the record format as a 1 character (A in position 35) hidden field (H in position 38). The field is returned with a value of 0 if the subfile is folded and with a value of 1 if the subfile is truncated. If SFLDROP (subfile drop) or SFLFOLD (subfile fold) is not specified on the SFLCTL (subfile control) record, the value is always returned as 0.

Initializing Output/Input Fields

Device support saves all data read from input-capable fields for records currently on the display in a save area. The output/input fields within this save area are updated on output operations.

For output operations, the following happens:

- Input-only fields are initialized to zeros (numeric fields), blanks (character fields), or a default value (DFT keyword) from the display file.
- Output/input fields, hidden fields, and program-to-system fields are initialized to the contents of the output buffer. If this output operation is caused by the initialize record function (INZRCD keyword), no output buffer is available. Output/input fields and hidden fields are initialized similar to input-only fields. Output/input fields are input and output capable.
- Output-only fields are not part of the input buffer unless they are part of a subfile record, in which case they are saved as if they were output/input fields.
- All response indicators for this record are set off.

Note: For input-capable fields, if the PUTRETAIN or ERASEINP keyword is in effect, the save area for the field remains unchanged.

Neither the input nor the output buffer is changed during write operations.

Inviting Input to the Display

The **invite operation** is used to send a request for input to a display station and return to the program without waiting for the input to arrive. This allows a program to request input from one or more display stations but continue processing without waiting for any of the display stations to respond. When the program is ready to process the input, the data can be received from any of the invited display stations by performing a read-from-invited-devices operation.

The invite operation is done by performing a write operation using a record format with the INVITE DDS keyword in effect. Refer to the appropriate high-level language manual to determine how to perform a write operation and how to use indicators to control the INVITE DDS keyword.

Once a display station is invited, the valid operations to receive data from the display station are the read-from-invited-devices operation and the read(wait) operation directed to a specific display station. Cancel invite is also a valid operation to an invited display station.

Before a display station can be used for I/O operations in a multiple-device display file, it must be acquired to the file. A program can direct the invite operation to any display station currently acquired for the file.

If the multiple-device display file was created with DFRWRT(*YES) specified, an output operation with the INVITE DDS keyword optioned on will cause the output that has been postponed to be displayed on the screen before the display station is invited.

If you want to invite a display station but have no data to send to it, perform the output operation with a record format which contains the INVITE DDS keyword optioned on but has no output-capable fields.

Multiple-display station display files are supported in ILE RPG, ILE COBOL, ILE C/C++, and CL.

Inviting Input from CL Programs

The invite operation is available directly to CL programs through CL commands:

- *WAIT(*NO) on RCVF and SNDRCVF Commands*:

  WAIT(*NO) allows overlapping of I/O operations and the running program, requests for input from more than one display, and receiving input as it is available. This provides support equivalent to the invite operation.

  On a read operation with the no-wait option, the system sends the request to the display and returns to the program. However, the requested record is not available when control returns. The purpose of this operation is to make the display station eligible to send input data while the program performs other work.

  To retrieve the record, issue a WAIT command. The WAIT command issues a read-from-invited-devices operation. The program waits until data is available from the display station or the WAITRCD time elapses. Then, the display station name and any input data are passed to the user program. If more than one read-with-no-wait operation has been issued (each to a different display) and more than one completes, the WAIT command processes only the first read-with-no-wait operation that is completed.

  A WAIT command can be issued to process each of the other read-with-no-wait operations. They are processed in the order of completion.

  When a record containing the INVITE keyword is sent to the display, the operation is handled as a write-read operation with a no-wait option. The INVITE DDS keyword is ignored on the write-read operation.

  A write-read operation with a no-wait option is the same as a write followed by a read-with-no-wait.

- *ENDRCV Command*:

  The ENDRCV command is used to end a request for input made with the WAIT(*NO) option. The ENDRCV command ends the input request even if data is available from the display station. If data is being sent by the display station when the ENDRCV operation is performed, the data is lost. If the display station is not invited, the application program is signaled with an error.

Reading Invited Input from the Display

The read-from-invited-devices operation provides a means of waiting for and receiving data from any one of the invited display stations. This method of inviting a display station and then reading from the invited display station is useful when the application must control the amount of time spent waiting for the user to respond. When the read-from-invited-devices operation is performed, the program waits for the time interval specified on the WAITRCD keyword of the CRTDSPF, CHGDSPF, or the OVRDSPF command. The wait can be ended in the following ways:

- *Data becomes available from an invited display station.* The display station name, the results of the operation, and any input data are passed to the program. When data has been received, the display station is no longer invited and must be invited again by an invite operation if more data is to be received from the display station by a read-from-invited-devices operation.

- *No-display station-invited signal.* Indicates that none of the display stations associated with the file are in the invited condition. Refer to the appropriate high-level language manual for information on how this will be communicated to the program.
- *Job-ended-controlled signal.* Indicates that the job that the program is running in is being ended with the controlled option through the End Job (ENDJOB), End System (ENDSYS), Power Down System (PWRDWNSYS), or End Subsystem (ENDSBS) command. Refer to the appropriate high-level language manual for information on how this will be communicated to the program. This occurs only once in a process no matter how many multiple-device display files are in use. All invited display stations remain invited.
- *No-invited-devices-have-data-available signal.* This occurs when no display stations associated with the file have data available, the WAITRCD time is *IMMED, and none of the previous conditions apply. The invited display stations remain invited. Refer to the appropriate high-level language manual for information on how this will be communicated to the program.
- *Time-out-on-wait-for-data-from-invited-devices signal.* This occurs when the WAITRCD value is a finite number of seconds, no data became available during that interval, and none of the previous conditions apply. Refer to the appropriate high-level language manual for information on how this will be communicated to the program. The invited display stations remain invited.

Also, ILE COBOL provides a means of performing the read-from-invited-devices operation as if WAITRCD(*IMMED) had been specified. See the ILE COBOL books for information on the NODATA phrase and its effect on the read-from-invited-devices operation.

Understanding the Read-From-Invited-Devices

When the program is ready to process input from one of the invited display stations, it can issue a read-from-invited-devices operation. This operation waits for a specified time for input to arrive from one of the invited display stations. The time limit can be specified when the display file is created and can subsequently be changed or overridden. If no invited display stations respond within the time limit, the program receives an indication that the time limit expired and can continue processing. If an invited display station responds within the time limit, the program can determine which display station responded and the record format used to process the data. The other invited display stations remain invited and can be sending data. The responding display station can also be invited again by another invite operation.

A read operation can also be directed to a specific display station. This operation will not complete until the specified display station responds with data. The display station need not be invited for the read operation, but, if it is, the program will wait for input and the display station is no longer invited.

- The read-from-invited-devices operation only accepts data from display stations which are currently invited.
- If more than one display station acquired to the display file has an invite outstanding, a read-from-invited-devices operation will return the next available record from one of the invited display stations. If records are received from more than one display station before the read-from-invited-devices operation, the other records will be kept for a subsequent read-from-invited-devices operation or for a subsequent read(wait) operation directed to a specific display station.
- When a display station has responded and the input is received by the read-from-invited-devices operation, that display station is no longer invited. It can be invited again by another invite operation but this should not be done until all the record formats on the display with input-capable fields have been read.
- A record format cannot be specified on the read-from-invited-devices operation. The record format returned from a display is the same as the last record format written to the display station.
- The timing function associated with the WAITRCD parameter may not force an end to the wait if the system is processing the Help key. In the following cases, the read-from-invited-devices function will not end until the user exits from the help information:
  - The system is displaying help that is defined by H specifications in the DDS for the display file.

- The system is displaying help for a message when the display station is the requester display station for the job and the display file specifies MAXDEV(1).

You can force message help to end when the WAITRCD time ends by specifying a value greater than 1 for the MAXDEV parameter on the CRTDSPF or CHGDSPF command.

Reading-From-Invited-Devices from CL Programs: The read-from-invited-devices operation is available directly to CL programs through CL commands. To retrieve the record, issue a WAIT command. The WAIT command issues a read-from-invited-devices operation. The program waits until data is available from the display station or the WAITRCD time elapses. Then, the display station name and any input data are passed to the user program. If more than one read-with-no-wait operation has been issued (each to a different display) and more than one completes, the WAIT command processes only the first read-with-no-wait operation that is completed.

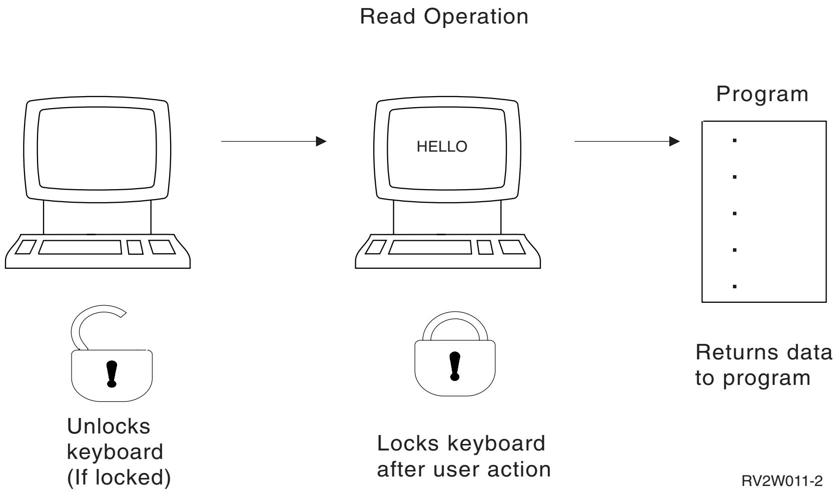
A WAIT command can be issued to process each of the other read-with-no-wait operations. They are processed in the order of completion.

When a record containing the INVITE keyword is sent to the display, the operation is handled as a write-read operation with a no-wait option. The INVITE DDS keyword is ignored on the write-read operation.

A write-read operation with a no-wait option is the same as a write followed by a read-with-no-wait.

Reading Input from the Display

A **read operation** passes a record from the system to the program. The display file record format contains the information necessary for the system to handle the record. The user must perform a required action such as pressing the Enter key or a function key to pass the data to the system. The read operation results in the following:


Read Operation
HELLO
Program
Unlocks keyboard (If locked)
Locks keyboard after user action
Returns data to program
RV2W011-2

For input operations, the following happens in the order given:

1. For an input-only operation, all response indicators for this record are set off and the read operation is issued.
2. Character fields received from the display are right- or left-justified and padded with blanks or truncated as necessary. The default is left-justify, which can be overridden using the AUTO or CHECK keyword.
3. Numeric fields received from the display have the following done to them:
   a. If the field is negative, the zone portion of the units position is set to a D (see “Handling Negative Numeric Input Data” on page 75).

b. All nonnumeric characters are removed and the numeric characters are compressed.
c. Signed numeric fields are right-justified and numeric-only fields are decimal aligned.
d. The field is padded with zeros or truncated as necessary.
e. Field validation is performed.

All fields received from the display whether they are part of the selected record or not are handled in this way.

If any field validation errors are detected, a message is sent to the user so that the error can be corrected. This process is repeated until there are no longer any errors. The save area for the requested record is then copied into the input buffer.

Note: To process input data for a read operation with no record format name, display station support uses the last record written to the display that contains at least one of the following:

- Input-only fields
- Output/input fields
- Hidden fields

If no such format is on the display, display station support uses the last format written to the display that did not contain these kinds of fields, for example, an output-only record that specifies valid command keys. If no such record exists on the display, an error message is returned to the program.

A record does not have to be written to the display before it can be read by the program with the INZRCD keyword. The system does this the same way an application program performs an output operation with the exception of the following:

- For an output-only field, no user data is available so the field is initialized to blanks. If the field is edited, the editing is ignored. If the BLKFOLD keyword is specified, it is ignored.
- For an output/input field, no user data is available so the field is initialized to blanks. If the field is edited, the editing is ignored. The field actually contains null characters (hexadecimal zeros), which appear as blanks.
- For a constant or input-only field, the data does not normally come from the output buffer so the field appears the same as when the program displays it using a write operation.
- For a hidden field, the field is returned on a read operation as blanks (hex 40) if the field is a character field or zeros (hex F0) if the field is a numeric field.
- For a message, there is no message data so the field is ignored.
- The LOGOUT keyword is ignored.
- The ERRMSG and ERRMSGID keywords are ignored because the record is not already on the display.
- The SFLMSG and SFLMSGID keywords are ignored.

All other fields or keywords are processed as if they were selected on an output operation.

Unlocking the Keyboard while the Program Is Processing Data

The keyboard can be unlocked so that data can be entered into input fields while the program is processing previously entered input data with the UNLOCK keyword.

Normally, input fields are not erased until after the keyboard is unlocked. On a read operation, input fields are erased after the keyboard is unlocked only if the UNLOCK keyword is specified and the GETRETAIN keyword is not specified.

For the 5250 display station, the read operation with the UNLOCK keyword in effect results in the following:

1. The 5250 display station does a hardware validity check on the fields. If no errors are found, the following is done:
   a. If the UNLOCK keyword is specified without the GETRETAIN keyword or if the UNLOCK(*ERASE) keyword is specified, all input-capable fields that are changed are cleared.
   b. If the UNLOCK keyword is specified with the GETRETAIN keyword or if the UNLOCK(*MDTOFF) keyword is specified, all modified data tags (MDTs) are reset.
   c. If the UNLOCK(*ERASE *MDTOFF) keyword is specified, all input-capable fields that are changed are cleared and their MDTs are reset.
   d. The cursor is repositioned to the field where the user can enter the next record.
   e. The keyboard is unlocked.
2. The system validity checks all the fields for all records on the display. If errors are detected, normal error retry is performed. A user could be typing into the next record when an error message is displayed.

   Note: The error message could refer to data that is no longer on the display because the data was erased.
3. Control returns to the program.

Notes:

1. If an application program detects input errors and sends error messages to the display, the messages may refer to input that has been typed over.
2. If the CHANGE keyword is specified and either the UNLOCK keyword is specified without the GETRETAIN keyword or with the UNLOCK(*ERASE) keyword is specified, the associated response indicator is set on for the next input record.
3. When a read operation with the UNLOCK keyword (and without the GETRETAIN keyword) or the UNLOCK(*ERASE) keyword is used for a record while a subfile is on the screen, subfile records may be returned to the program on a subsequent get-next-changed operation to the subfile even though the user did not enter data into the subfile record. It is recommended that you use the UNLOCK(*ERASE *MDTOFF) keyword instead of the UNLOCK keyword (without the GETRETAIN keyword) or the UNLOCK(*ERASE) keyword. If you must use either of the latter, you should make sure that your high-level language program compares for blanks to handle the possibility that an unmodified field containing all blanks is returned to the program.

Keeping Input Data

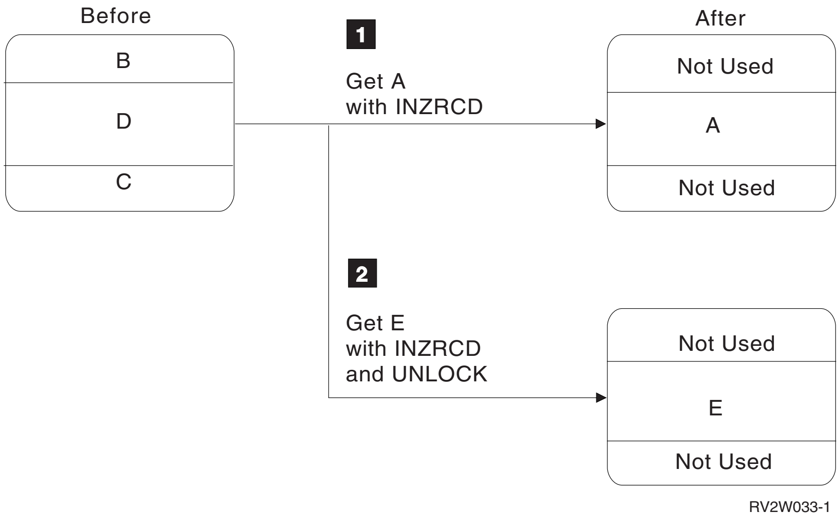
Input data on a display can be kept after the user presses the Enter key with the GETRETAIN keyword. The GETRETAIN keyword can only be used with the UNLOCK keyword.

Setting an Indicator When Data Is Changed

A response indicator can be set on when data is entered into an input field or when data is changed in an output/input field with the CHANGE keyword.

Initializing Records and Unlocking the Keyboard-Diagram

The following diagram shows the effect of INZRCD and UNLOCK keywords on an input operation:



Note: Record formats A, D, and E occupy the same lines.

1 Record formats B, D, and C are erased if the OVERLAY keyword is not specified for record format A. Record format A is displayed with constants and initialized input fields. The keyboard is unlocked. The keyboard is locked after the user satisfies the get operation.

2 Record formats B, D, and C are erased if the OVERLAY keyword is not specified for record format E. Record format E is displayed with constants and initialized input fields. The keyboard is unlocked. After the user satisfies the read operation, the contents of the input fields are erased and the keyboard is unlocked again.

Note: Even though the UNLOCK keyword is specified, field validity checking, if specified, and command key verification are performed. Therefore, a user could be typing into the next record when an error message is sent to the display.

Specifying Validity-Checking Functions

Two methods can be used to check the validity of data entered by the user:

- Have the system check the data before it is passed to the application program.
- Have all the input data passed to the application program, which checks the validity of the data.

In either case, if errors are detected, a message is displayed informing the user of the error so that it can be corrected. If you choose the second method for detecting errors, see "Creating and Displaying Your Own Messages" on page 223 for information on how your program can display error messages. The rest of this section gives more information on the first method, when the system detects the errors before passing the data to your program.

The validity-checking functions you can specify in DDS are:

- Detecting fields in which at least one character must be entered (CHECK(ME) keyword). Blanks are valid characters. This is referred to as *mandatory enter*.
- Detecting fields in which every position must contain a character (CHECK(MF) keyword). Blanks are valid characters. This is referred to as *mandatory fill*.
- Detecting incorrect data types where character, numeric, or signed numeric data is required.
- Detecting data that is not in the range specified for the field (RANGE keyword).
- Performing comparison checking between data entered and specified constant value (COMP keyword).
- Comparing the data entered to a specific list of valid entries (VALUES keyword).
- Detecting if a valid field or record name was entered in a character field (CHECK(VN) keyword).
- Detecting if a valid object name was entered in a character field (CHECK(VNE) keyword).

- Performing modulus 10 or 11 check digit verification (CHECK(M10) or CHECK(M11) keyword). (Only one can be specified.)
- Allowing blank-key entries to be processed as if no entry had been made (CHECK(AB) keyword). CHECK(AB)-Allow Blanks-is ignored if the subfile keyword SFLROLVAL or SFLRCDNBR is also specified for the field.
- Detecting if a space, a plus sign, or a minus sign is embedded between numeric digits in a numeric field. Also, detecting if a plus sign or minus sign precede a numeric digit in a numeric field. To detect such cases, use the Validate Numeric (VALNUM) keyword.

The ERRSFL keyword can be used in addition to the validity checking keywords CHECK(M10 M11 VN VNE), COMP, RANGE, and VALUES to allow more than one of the error messages associated with the keywords to be displayed at one time.

When you specify the RANGE, COMP/CMP, VALUES, CHECK(VN), CHECK(VNE), CHECK(M10), or CHECK(M11) keyword for validity checking and an error is detected by one of these validity checking functions, the following happens:

1. The keyboard is locked.
2. All fields in error are displayed in reverse image. If a field in error has both the underline (UL) display attribute and the highlight attribute (HI), its image is not reversed, as this combination of attributes has the same effect as DSPATR(ND).
3. The cursor is positioned at the beginning of the first field in error.
4. A system-supplied error message for the first field in error is displayed on the error line,

   or,

   If you have chosen to provide your own error message for a field using the CHKMSGID keyword and this is the first field in error, then your error message is displayed on the error line.

If your controller is installed with the self-check feature (see the *5250 Functions Reference*), the controller performs validity checking for the CHECK(M10F) and CHECK(M11F) keywords. Errors are detected when you attempt to move the cursor from the input field rather than when you press the Enter key or a Command Attention key. The Operator Error Code 00115, rather than a system-supplied or user-specified message, is displayed in the lower left corner of the display. If the USRDSPMGT keyword is also specified, CHECK(M10) and CHECK(M11) function as CHECK(M10F) and CHECK(M11F).

If the RANGE, COMP, VALUES, CHECK(VN), or CHECK(VNE) keyword is specified for a field, and data is entered into that field, the field indicates that it has been changed regardless of attempts by the user to restore the field after an error. If blanks (for character fields) or zeros (for numeric fields) will fail the validity checking function, use the CHECK(AB) keyword. This will satisfy the validity checking function.

When you specify validity checking for records that are part of a subfile, each field in the record is validity checked before it is placed in the subfile from the display. You cannot roll the records until all fields in error are corrected.

The system only performs validity checking on a field if the field is changed by the user or if its modified data tag (MDT) is set on using DSPATR(MDT).

Notes:

1. If the user presses the Dup key, any validity checking for a field is ignored. The DUP keyword lets the user use the Dup key.
2. The value for a numeric field for which the COMP, VALUES, or RANGE keyword is specified is aligned based on the decimal positions specified for the field and filled with zeros where necessary. If decimal positions were not entered for the field, the decimal point is assumed to be to the right of the digit to the extreme right in the value. For example, for a numeric field with length of 5 and decimal positions of 2, the value 1.2 is interpreted as 001.20 and the value 100 is interpreted as 100.00.

3. When you use the RANGE keyword for validity checking an input field and blanks are entered in the input field, the value for the input field may not meet the range requirements. Blanks are converted to zeros for numeric fields and are passed as blanks for character fields. Use the field level keyword BLANKS to determine when a field is displayed as all blanks. The response indicator on the BLANKS keyword is set on if the user enters blanks.

Understanding the Limitations on the Number of Input-Capable Fields

For a remote 5250 display station (a display station attached through a remote controller), you can specify as many as 126 or 256 input fields on one display, depending on the controller model. (The 5294 controller supports 126 input fields; the 5394 controller supports 256 input fields.) Additionally, if either DSPATR(OID) or DSPATR(SP) is specified, this maximum is reduced by 1 for each three instances of these keywords. If fewer than three instances occur, it is still reduced by one.

For a local 5250 display station (a display station attached through the local display station controller), you can specify as many as 256 input fields. Also, if either DSPATR(OID) or DSPATR(SP) is specified, this maximum is reduced by 1 for each three instances of these keywords. In addition, any use of the magnetic stripe reader on a local 5250 display station also reduces the maximum number of fields. The maximum number of fields is calculated as follows:

$$256 - \left[\frac{3+B}{6} + \frac{A}{3} \right] \text{ rounded up to the next whole number}$$

RSLH131-2

A is the number of DSPATR(OID) and DSPATR(SP) fields on the display and B is the length of the longest expected magnetic stripe input where 125 data characters is the maximum allowed. Magnetic stripe data not specified as DSPATR(OID) can be entered into any input field.

If the maximum number of input fields is exceeded in any of the preceding cases, message CPF5192 is issued to the using program.

No maximum-number-of-fields diagnostic is provided during display file creation because the number of fields and record formats is not known until the program is run.

When a subfile record is displayed, the actual number of input-capable fields sent to the display is the number defined in the record multiplied by the number of subfile records that are displayed.

For remotely attached 3270 displays, the limitation is 126 input fields.

For ASCII displays attached through a protocol converter, the limitations are the same as the controller to which they are attached.

Handling Negative Numeric Input Data

The negative sign in numeric input data can appear in three forms:

- Hex 60 if the sign is entered using the - (minus) key
- Hex D if the sign is entered using the Field Minus key
- Hex Dn if the sign is entered as an alphanumeric character with a D zone

The hex 60 is treated as a true minus sign if it is to the right of the least significant digit.

The hex D zone is treated as a minus sign if it is the least significant digit. In addition, it is treated as a significant digit with a value equal to the numeric portion.

Imbedded blanks (between significant digits) are changed to zeros before decimal alignment.

Understanding How the System Reads Input from the Display

When a read operation is issued, the system reads all the records on a display. However, only one record is passed to the program for each read operation. The system saves all the other records in anticipation of more read operations.

If each read operation refers to a different record on the display, no action is required of the user. However, if each read refers not to a different record on the display but to the same record and if the RTNDTA keyword is not specified, the user must perform an action such as pressing the Enter key or a CFnn key to start the next read operation because each record entered is passed to the program only once. If the RTNDTA keyword is specified, the user does not have to perform any action because the same input buffer that was returned to the program on the previous read operation for the record is returned again.

The system saves the contents of input-capable fields for records that are active on the display. This saved data is passed to the user program and can be altered by:

- Initializing the data with a constant on a write operation. A field can be initialized with the value specified in a DFT keyword.
- Entering data through directly typing the data in or using a light pen to select data. (The MDT for a field can be set on to simulate user input.)
- Entering data from a program on a write operation. This applies to output/input fields (and output-only fields for subfiles).
- Initializing the data with blanks (character fields) or zeros (numeric fields) on an output operation for the same record unless the PUTRETAIN keyword is specified. This applies to input-only fields.

Writing Output and Reading Input at the Same Time

The write-read operation is a combination of a write operation and a read operation to the same record format in one high-level language statement like the SNDRCVF command in a CL program. It behaves as if you had specified a read operation immediately following a write operation.

Some high-level languages have a write-read operation which writes information on the display and reads the user response in one statement. For example, ILE RPG has the execute format (EXFMT) operation. This kind of operation is useful if you need to both present new information on the display and request information from the user at the same time. You can also use a write operation followed by a read operation to the same record format to simulate this operation in languages that do not support a combined write-read operation.

When this operation is performed, the following happens:

1. The program calls the system display support giving it the data to show on the display and the record format to use when writing and reading that data.
2. The system combines that data with the information it finds in the record format and constructs the data stream to be sent to the display.
3. The data stream is then sent to the display and the keyboard is unlocked.
4. The user types the data in the fields which allow input and presses the Enter key or some other function key.
5. The data is then sent from the display to the system. The system decodes it and extracts only the information that the application program needs to know and returns that data to the application program.

When you work with only one record format, this write-read style of working with it is the most common. On the write portion of the operation, you provide the data that the user will see. On the read portion, you receive data back that the user has entered or changed.

Canceling Input That Was Not Waited For

The cancel-invite operation is used to cancel the input request issued to a display station that was previously invited through the invite operation. The input request is canceled by performing a write operation to the invited display station. One of the following occurs:

- If the write request is received before the user responds to the input request from the invite operation, the input request is canceled and the record format specified on the write operation is sent to the display station. If the record format has the option indicator set on for the DDS keyword INVITE, the display station is invited again.
- If the write request is received after the user responds to the input request from the invite operation, the input request is not canceled and the write operation fails. The read-from-invited-devices operation or a read(wait) operation must be issued to receive the available data.

Releasing a display station also implicitly cancels any input requests directed to the display station. If the display station has data available, the data is lost.

Locking the Keyboard and Positioning the Cursor During I/O Operations

The following lists what happens to the keyboard when a write, write-read, or read operation is run:

Operation	Keyboard
Write	The keyboard is unlocked by default. If the LOCK keyword is specified, the keyboard is not unlocked.
Write-Read	The keyboard is unlocked.
Read	The keyboard is unlocked (if locked) before display station user action. After user action, the keyboard is locked by default. If the UNLOCK keyword is specified, the keyboard is left unlocked.

Every time the keyboard is unlocked, the cursor is repositioned. In some cases, many write operations between read operations can cause erratic cursor movement. If the user starts typing before the last write operation, the cursor is repositioned when the keyboard is unlocked and this can cause confusion for the user. You can prevent this by using the LOCK keyword. By using the LOCK keyword on each write operation but the last, the keyboard remains locked until the last write operation. This avoids erratic cursor movement, but prevents the user from starting to type data.

Normally, a user action, such as pressing a valid command key, locks the keyboard.

To specify that the system unlock of the keyboard on the next input operation should not occur, specify the retain lock status (RETLCKSTS) keyword. This keyword prevents the loss of data when the input operation is started and data is already being transmitted from the keyboard.

Note: Use the RETLCKSTS keyword only when the keyboard is already unlocked.

To position the cursor with the DSPATR(PC), CSRLOC, or SFLRCDNBR(CURSOR) keyword, the keyboard must be locked. Only the following conditions on an output operation cause the keyboard to be locked and must be present for the display station to position the cursor. (An output operation normally unlocks the keyboard before it ends unless the LOCK keyword is specified so these conditions lock the keyboard only momentarily.

- Input-capable fields are erased (ERASEINP keyword).
- Modified data tags are reset (MDTOFF keyword).
- Any input-capable field is written to the display.
- The complete display is erased (a write operation without an OVERLAY keyword).
- The 5250 format buffer is reset, which can be the result of:

– A record format with an input-capable field is overlaid or erased.
– A record format with a cursor location specification is overlaid or erased.
– The PROTECT keyword is specified on the record being written.

For the cursor positioning keyword to take effect, the keyboard must go from the lock condition to an unlocked condition. That is, if the keyboard is unlocked prior to the write operation, the cursor positioning keyword does not take effect immediately on the write operation. However, there is one exception. If the keyboard is temporarily locked during an output operation, the cursor positioning keyword will be in effect if the output operation unlocked the keyboard at the end.

In addition, if any of the preceding conditions happens on a write operation, the keyboard must be unlocked before any user action either by the same operation or by a following operation (it should be the last write operation).

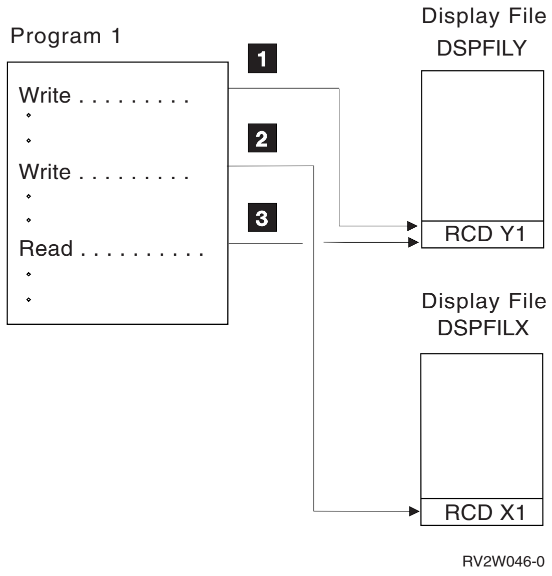
A write operation to a subfile never unlocks the keyboard because no input or output is sent to the display station.

Saving Previously Displayed Information

A display file may be opened to a display station even when another display file is already using that display station. When an I/O operation is performed to the second display file, the first display file is *suspended*.

When a display file is suspended, the information on the display can be saved automatically by the system if you specify *YES for the RSTDSP parameter on the Create Display File (CRTDSPF) or Change Display File (CHGDSPF) command. The contents of a display file can then be restored when an I/O operation is later performed to that display file. If *NO is specified for the RSTDSP parameter, the application program needs to rewrite the display to show it again.

The RSTDSP parameter lets you overlap a program call, keyboard input, and file I/O processing, as shown in the following example:



1 Program 1 issues a write operation to record format RCD Y1 in display file DSPFILY, which activates display file DSPFILY.

2 Program 1 issues a write operation to record format RCD X1 in display file DSPFILX, which suspends display file DSPFILY and activates display file DSPFILX. If RSTDSP(*YES) is specified for display file DSPFILY, the data displayed on the display station is saved. If RSTDSP(*NO) is specified instead, the data is lost and the program needs to write the information in RCD Y1 again to show it.

3 Program 1 issues a read operation to record format RCD Y1, which suspends display file DSPFILX and restores display file DSPFILY. If the RSTDSP(*YES) parameter is specified for DSPFILY, then the data displayed on the display station when DSPFILY is suspended can be restored.

When RSTDSP(*YES) is specified for a display file, and you are suspending and restoring that display file because of operations to another display file, some displays may appear to flash on the screen briefly. If a display file has a record on the display and an I/O operation is done to a second display file, the first file is suspended and its screen contents are saved. When returning to the first display file, the display file and its screen contents are restored. If a write operation is done to a different record format in the display file, the restored display will flash briefly before the output operation is complete. If you are going to completely rewrite the display contents from your program when going back to the first file, use RSTDSP(*NO).

You should specify *YES for the RSTDSP parameter in the following situations:

- When you are writing a record that has the following keywords in effect:
  - CLRL
  - OVERLAY
  - PUTOVR
  - PUTRETAIN
  - ERRMSG
  - ERRMSGID

  You must ensure that the records that are on the display are the records that these keywords apply to. If the display file is suspended, the data must be restored to the screen so that the write operations to the record formats that use these keywords are valid.
- When you perform multiple read operations to a record format on the display without intervening write operations. If you should call a program while processing the data that has been read and that program presents a display of its own, the subsequent read operation done by your program restores the display properly.

Saving and displaying data again requires significant system and data transmission overhead. For a 1920-character 5250 display station, approximately 3000 characters are transmitted each time the display data is saved displayed again. To avoid this overhead, write your application programs to do the following:

- Make the programs in the application share the same copy of the display file among themselves by specifying SHARE(*YES) on the display file.
- Perform complete display rewrites each time the programs in the application write to the display. A complete display rewrite occurs when a record is written to the screen and the OVERLAY keyword is not used or implied.

  Note: If complete display rewrites are not performed and if new input fields, occupying positions on the screen above the currently displayed fields, are sent to the display, the program receives a message (CPF5192). This occurs because the 5250 display station requires that new input fields sent to the display appear in lower positions than input fields currently on the display. In normal operations, data management performs field processing to satisfy the 5250 requirement. See “Avoiding Record Format Problems on the 5250 Display Station” on page 80.

When one program that uses a display file with the SHARE(*YES) parameter specified calls another program that uses the same display file, the display file is not suspended even though both programs have opened the file. If the display file is not shared, the system maintains separate copies of the display file for each program and suspends and restores the display files separately.

Since system programs do not specify file sharing, you should specify RSTDSP(*YES) on the CRTDSPF or CHGDSPF command if your program contains a display file and calls system functions that present displays. System functions that break into the normal path of an application, however, such as the System Request Menu or the presentation of break messages, restore the display without RSTDSP(*YES) specified.

To display saved display data again after a close operation is issued to a suspended file, specify the KEEP keyword for a record format in the saved display data.

Understanding the Effects of I/O Operations on Command Keys

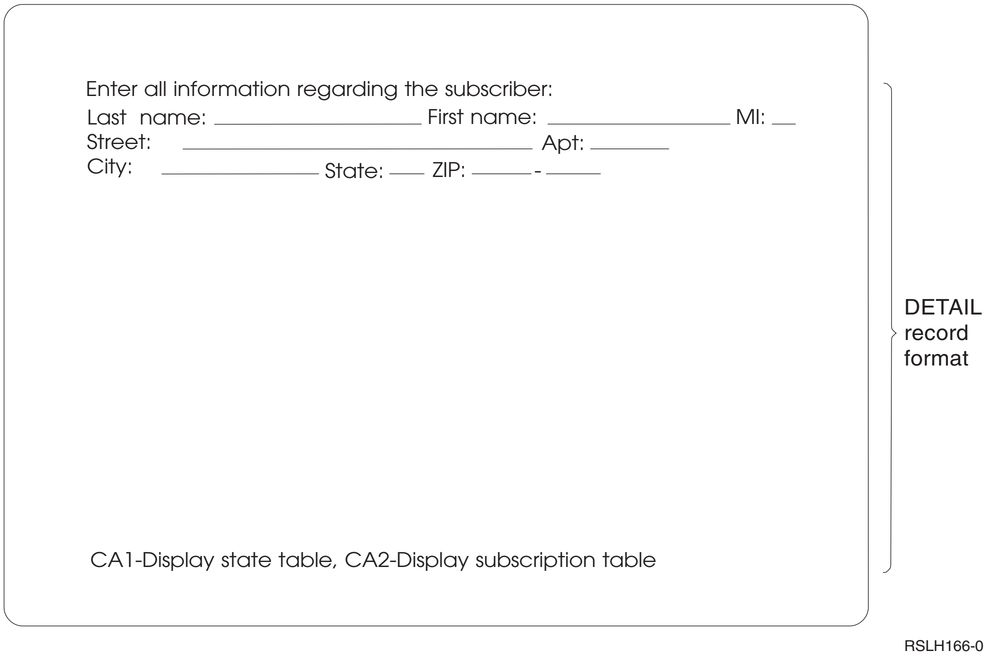
Read and write operations may or may not affect how the function keys work:

- A write or write-read operation for which no input or output is sent to the display does not affect which keys are valid. Examples of such operations are a write operation or an update operation to a subfile record format.
- If a write or write-read operation displays a message by selecting either the ERRMSG or ERRMSGID keyword, the command keys in effect on that output operation are valid. Therefore, you can specify a different set of command keys to be valid if an error occurs.
- If only one subfile record format is displayed and the subfile control record format specifies a CAnn or CFnn key for the SFLDROP keyword, that key remains valid for that function as long as the subfile is still on the display. In addition, the key specified for the SFLENTER keyword remains valid until another write or write-read operation is done. At the next output operation, the specifications for that record apply.
- If two subfile record formats are displayed and both specify the SFLDROP keyword, only the last SFLDROP keyword is used. There can only be one drop key at a time.

Avoiding Record Format Problems on the 5250 Display Station

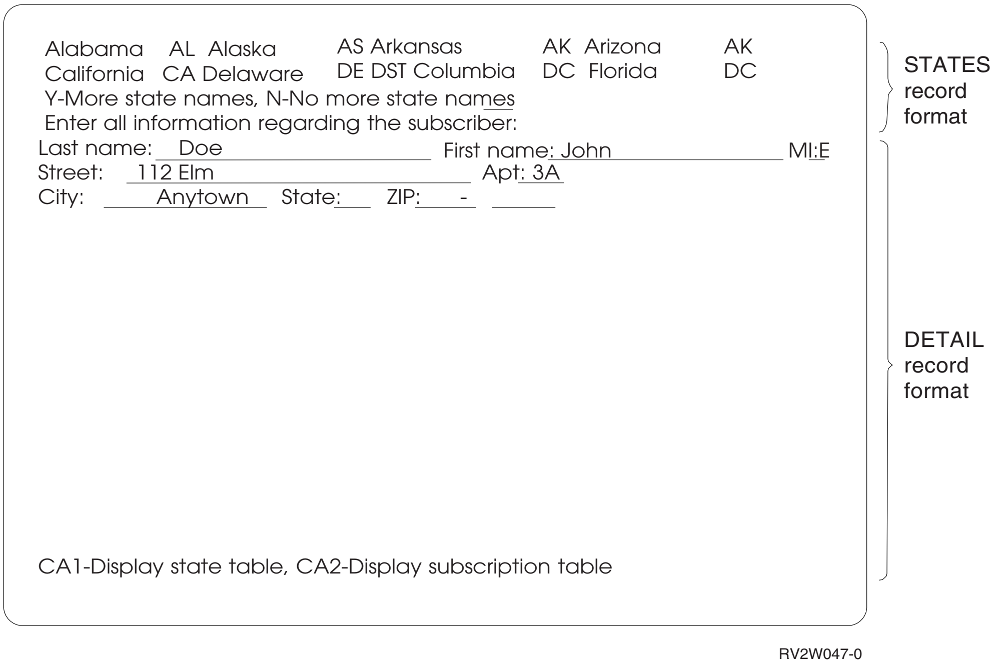
Because of the characteristics of the 5250 display stations, certain record format positioning and operational combinations can produce undesirable results. The following example illustrates a combination that can cause undesirable results and explains how to avoid these results.

The displays produced and the DDS for the record formats follow:



Enter all information regarding the subscriber:
Last name: ____________ First name: ____________ MI: __
Street: ____________________ Apt: ______
City: ____________ State: __ ZIP: _____ - ____

CA1-Display state table, CA2-Display subscription table



Alabama AL Alaska AS Arkansas AK Arizona AK
California CA Delaware DE DST Columbia DC Florida DC
Y-More state names, N-No more state names
Enter all information regarding the subscriber:
Last name: Doe First name: John MI:E
Street: 112 Elm Apt: 3A
City: Anytown State: ZIP: -

CA1-Display state table, CA2-Display subscription table

```
|...+....1....+....2....+....3....+....4....+....5....+....6....+....7...
     A          R RCDA
     A            FLD1           1   I  2  4DSPATR(PC)
     A* STATES record format follows
     A          R STATES                    OVERLAY
     A            SNAME1        12   O  1  2
     A            SCODE1         2   O  1 15
     A            SNAME2        12   O  1 18
     A            SCODE2         2   O  1 31
     A            SNAME3        12   O  1 34
     A            SCODE3         2   O  1 47
     A            SNAME4        12   O  1 50
     A            SCODE4         2   O  1 63
     A             .             .
     A             .             .
     A             .             .
     A            SNAME8        12   O  2 50
     A            SCODE8         2   O  2 63
     A                                   3  2'Y-More state names, N-No more +
     A                                      state names'
     A            MORESNAM       1   I  3 48VALUES('Y' 'N')
     A* DETAIL record format follows
     A          R DETAIL                    OVERLAY CA01(11) CA02(12)
     A                                   4  2'Enter all information regarding +
     A                                      the subscriber:'
     A                                   5  1'Last name:' DSPATR(HI)
     A            NAMEL         20   I  5 12
     A                                   5 33'First name:' DSPATR(HI)
     A            NAMEF         13   I  5 45
     A                                   5 59'MI:' DSPATR(HI)
     A            MI             1   I  5 63
     A                                   6  1'Street:' DSPATR(HI)
     A
     A            STREET        45   I  6  9
     A                                   6 55'Apt:' DSPATR(HI)
     A            APT            4   I  6 60
     A                                   7  1'City:' DSPATR(HI)
     A            CITY          15   I  7  7
     A                                   7 23'State:' DSPATR(HI)
     A            SCODE          2   I  7 30
     A                                   7 33'Zip:' DSPATR(HI)
     A            ZIP1           5   I  7 38
     A                                   7 44'-'
     A            ZIP2           4   I  7 46
     A             .                   .
     A             .                   .
     A             .                   .
     A                                  10  1'CA1-Display state table, +
     A                                      CA2-Display subscription table'
```

Figure 20. Sample DDS to Show Record Format Problems

Assume that the DETAIL record format is on the screen, and the user is entering data for a subscriber. Because the user does not know the state code for the state to be entered, he or she presses the CA01 key.

Because CA01 is defined as a CAnn key, no data is transmitted to the system when the CA01 key is pressed. The data, however, remains on the screen. The program detects that the CAnn key was pressed because response indicator 11 is set on. The program then displays the STATES record format.

Because the STATES record format is physically above the DETAIL record format, the system must resend the field formats for the input fields in the DETAIL record format. (The system would also resend the field formats if the STATES record format contained only output-only fields and was replacing the RCDA record format. In this case, the field formats are resent because a record format (RCDA) with a specific cursor location is being removed.)

The following problems occur because the system resends the field formats for the DETAIL record format:

- All the input fields in the DETAIL record format lose their modified data tags (MDTs). When the program does the next read to the DETAIL record format (for example, when the user presses the Enter key), none of the fields typed in before the user pressed the CA01 key are returned to the program. The program cannot retrieve that typed data even though the data still remains on the screen.

  To avoid this problem:

  - Avoid using a CAnn key.
  - If you must use a CAnn key, avoid writing the format containing the CAnn key to the screen and then writing another format that is physically placed above the first format if both formats contain input-capable fields.
  - Avoid writing a format to the screen that causes the removal of a format containing a cursor location specification.
- The highlight attribute for the constant fields (except Last name) is lost. The system does not resend the field format for output-only fields. However, if an output-only field immediately follows an input-capable field so that the leading attribute character for the output field is in the same position as the ending attribute character for the input-capable field, the attribute of the output field reverts to normal.

  To avoid this problem:

  - Do not specify an output-only field with special display attributes immediately following an input-capable field.
  - If you must specify an output-only field with special display attributes immediately following an input-capable field, avoid writing that format to the screen and then writing another format that is physically placed above the first format if both formats contain input-capable fields.
  - Avoid writing a format to the screen that causes the removal of a format containing a cursor location specification.

Note: The system needs to resend the attributes for the input-capable fields when a subfile is rolled from a full page to a partial page, a partial page to a full page, or a partial page to a partial page. The two problems mentioned above may also occur when resending the field attributes.

Releasing an Acquired Display Station from I/O Operations

The **release operation** makes a display station ineligible for any further I/O operations through a file. This operation is used in multiple display file applications or if you are performing error recovery in your program. If the display station being released is invited, the invite is ended. If the invited display station had data available, the data is lost. The release operation can only be performed on display stations that are currently acquired to the file.

The release operation can also be used to recover from errors from acquire, I/O, and release operations. After a display station is released, it must be acquired again with another acquire operation before any I/O operations can be directed to it. If a program is written to recover from errors by releasing a display station and then acquiring it again, a value other than *IMMED should be specified on the WAITFILE keyword. This is because it takes the system a short time to transfer the allocation of a display station description from a job, to the subsystem, and back again.

Closing Display Files

The **close operation** makes the display file ineligible for any further I/O operations between the program and the system. Refer to the appropriate high-level language manual for information on how to start the close operation.

If the display file is not being shared, the close operation also implicitly releases all the display stations acquired to the file and deallocates any file resources allocated by the open operation or the acquire operation.

If the close operation is successful, the only valid operation to the file is open. If the close operation fails, the program should issue the close operation a second time.

Mapping Display Operations to High-Level Language Operations

The following shows the I/O requests supported by the operating system and the equivalent high-level language operations:

Table 12. Display File Operations Supported by the Operating System and the Equivalent High-Level Language Commands

Operation	BASIC Statements	ILE C/C++ Functions	CL Commands	ILE COBOL Statements	ILE RPG Operations
Open	OPEN	fopen, _Ropen		OPEN	OPEN
Acquire		_Racquire		ACQUIRE	ACQ
Release		_Rrelease		DROP	REL
Get Attributes		_Rdevatr		ACCEPT	POST
Write	WRITE	fwrite, _Rformat, _Rpgmdev, _Rwrite	SNDF	WRITE	WRITE, output specifications
Read(wait)	READ	fread, _Rformat, _Rpgmdev, _Rreadn	RCVF WAIT(*YES)	READ	Primary or secondary file input, READ
Read	READ		RCVF WAIT(*NO)	READ	
Cancel Read			ENDRCV		
Wait			WAIT		
Invite		fwrite[1], _Rformat, _Rpgmdev, _Rwrite[1]	SNDF[1]	WRITE[1]	WRITE[1]
Read from Invited Device		_Rreadindv		READ	READ
Cancel Invite		fwrite, _Rformat, _Rpgmdev, _Rwrite	SNDF	WRITE	WRITE
Write-Read(wait)		_Rwriterd, _Rformat, _Rpgmdev	SNDRCVF WAIT(*YES)		EXFMT
Write-Read(no-wait)			SNDRCVF WAIT(*NO)		
Close	CLOSE, END	fclose, _Rclose	RETURN, RCLRSC	CLOSE, CANCEL, STOP RUN	CLOSE, RETRN

1. This is the write operation of a record format with the INVITE DDS keyword selected.

If an error occurs during an I/O operation to a display file, the major/minor return code field in the file dependent I/O feedback area may be used to help diagnose the error and determine the error recovery action needed.

Sharing Display Files in the Same Job

By specifying the SHARE parameter on the CRTDSPF, CHGDSPF, and OVRDSPF commands, you can specify that more than one program share the same path to the data or the display station. Using the SHARE parameter allows more than one program to share the file status, positions, and storage area, and can improve performance by reducing the amount of main storage the job needs and by reducing the time it takes to open and close the file.

Using the SHARE(*YES) parameter lets an open data path (ODP) be shared between two or more programs running in the same job. An **open data path** is the path through which all input/output operations for the file are performed. It connects the program to a file. If not specified otherwise, every time a file is opened a new open data path is built. You can specify that if a file is opened more than once and an open data path is still active for it in the same job, the active ODP for the file can be used with the current open of the file, and a new open data path does not have to be created. This reduces the amount of time required to open the file after the first open, and the amount of main storage required by the job. SHARE(*YES) must be specified for the first open and other opens of the same file for the open data path to be shared. Specifying SHARE(*YES) for other files depends on the application.

Note: Most high-level language programs process an open or a close operation independent of whether or not the file is being shared. You do not specify that the file is being shared in the high-level language program. You indicate that the file is being shared in the same job through the SHARE parameter. The SHARE parameter is specified only on the create, change, and override file commands. Refer to your appropriate language manual for more information.

Understanding the Open Operation for Files Shared in a Job

The following items should be considered when opening a file that is shared in the same job by specifying SHARE(*YES).

- You must make sure that when the shared file is opened for the first time in a job, all the open options that are needed for subsequent opens of the file are specified. If the open options specified for subsequent opens of a shared file do not match those specified for the first open of a shared file, an error message is sent to the program. (You can correct this by making changes to your program to remove any incompatible options.)

  For example, PGMA is the first program to open FILE1 in the job and PGMA only needs to read the file. However, PGMA calls PGMB which will delete records from the same shared file. Because PGMB will delete records from the shared file, PGMA will have to open the file as if it, PGMA, is also going to delete records. You can accomplish this by using the correct specifications in the high-level language. (In order to accomplish this in some high-level languages, you may have to use file operation statements that are never run.) For more details, see your appropriate language manual.
- If you did not specify a library name in the program or the override command (*LIBL is used), the system assumes that the library list has not changed since the last open of the same shared file with *LIBL specified. If the library list has changed, you should specify the library name on the override command to ensure that the correct file is opened.
- Overrides and program specifications specified on the first open of the shared file are processed. Overrides and program specifications specified on subsequent opens, other than those that change the file name or the value specified on the SHARE or LVLCHK parameters on the override command, are ignored.

Understanding the Input/Output Operation for Files Shared in a Job

The system uses the same input/output area for all programs sharing the file, so the order of the operations is sequential regardless of which program does the operation. For example, if Program A is reading records sequentially from a file and it reads record 1 just before calling Program B, and Program B also reads the file sequentially, Program B reads record 2 with the first read operation. If Program B then ends and Program A reads the next record, it receives record 3. If the file was not being shared, Program A would read record 1 and record 2, and Program B would read record 1.

For display files, the display station remains in the same state as the last I/O operation.

For display and ICF files, programs other than the first program that opens the file may acquire more display or program display stations or release display or program display stations already acquired to the open data path. All programs sharing the file have access to the newly acquired display stations, and do not have access to any released display stations.

Understanding the Close Operation for Files Shared in a Job

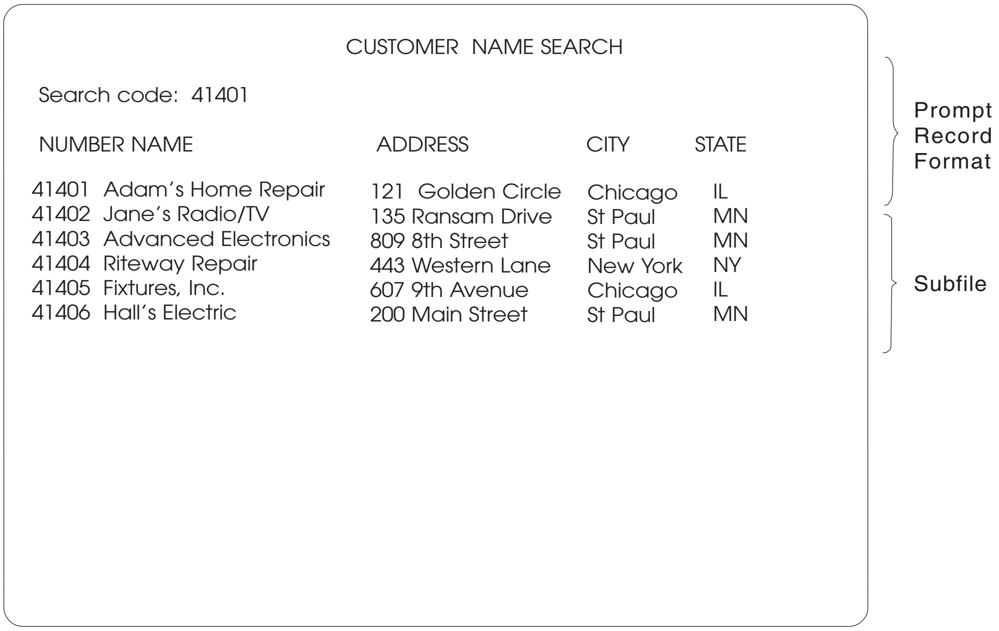
The processing done when a program closes a shared file depends on whether there are other programs currently sharing the open data path. If there are other programs, the main function that is performed is to detach the program requesting the close from the file. All other programs sharing the file are still attached to the ODP and can perform I/O operations.

If the program closing the file is the last program sharing the file, then the close operation performs all the functions it would if the file had not been opened with the share option. This includes releasing any allocated resources for the file and destroying the open data path.

The function provided by this last close operation is the function that is required for recovering from certain run-time errors. If your application is written to recover from such errors and it uses a shared file, this means that all programs that are attached to the file when the error occurs will have to close the file. This may require returning to previous programs in the program stack and closing the file in each one of those programs.

Chapter 4. Displaying Groups of Records Using Subfiles

A subfile is a group of records that have the same record format and are read from and written to a display station in one operation. The following sample display shows an example of a subfile:



```
                         CUSTOMER  NAME SEARCH

 Search code:  41401

 NUMBER NAME                  ADDRESS             CITY        STATE

41401  Adam's Home Repair     121  Golden Circle  Chicago      IL
41402  Jane's Radio/TV        135 Ransam Drive    St Paul      MN
41403  Advanced Electronics   809 8th Street      St Paul      MN
41404  Riteway Repair         443 Western Lane    New York     NY
41405  Fixtures, Inc.         607 9th Avenue      Chicago      IL
41406  Hall's Electric        200 Main Street     St Paul      MN
```

RV2W049-0

Information about DDS keywords

This chapter uses DDS keywords to describe subfiles. For more information about specific DDS keywords, see the **DDS** topic in the iSeries Information Center.

Recognizing Subfile Uses

Subfiles are useful when multiple records that are alike must be displayed. You can describe a subfile so that the number of records to be displayed fits on one display or exceeds the number of lines available on the display.

You can use subfiles for the following purposes:

- Display only, which allows the user to review the subfile records on the display (for example, all the line items for a particular order number, or a group of records containing customer names and addresses as shown on the previous sample display).
- Display with selection, which allows the user to request more information about one of the items on the display. On the first sample display of the following example, the user can request the records for a particular customer by entering the record number in the record number field. In the second sample display, the user can request the records for a particular customer by placing an X in the select number field.

```
Enter customer number:  41401
Enter record number:  ____
RECORD NUMBER NAME                 ADDRESS                CITY      STATE
  01   41401  Adam's Home Repair   121 Golden Circle      Chicago   IL
  02   41402  Jane's Radio/TV      135 Ransom Drive       St Paul   MN
  03   41403  Advanced Electronics 809 8th Street         St Paul   MN
  04   41404  Riteway Repair       443 Western Lane       New York  NY
  05   41405  Fixtures, Inc.       607 9th Avenue         Chicago   IL
  06   41406  Hall's Electric      200 Main Street        St Paul   MN
```

```
Enter customer number:  41401
SELECT
RECORD NUMBER NAME                 ADDRESS                CITY      STATE
  _    41401  Adam's Home Repair   121 Golden Circle      Chicago   IL
  _    41402  Jane's Radio/TV      135 Ransom Drive       St Paul   MN
  _    41403  Advanced Electronics 809 8th Street         St Paul   MN
  _    41404  Riteway Repair       443 Western Lane       New York  NY
  _    41405  Fixtures, Inc.       607 9th Avenue         Chicago   IL
  _    41406  Hall's Electronic    200 Main Street        St Paul   MN
```

- Changing information, which allows the user to change one or more of the records in the subfile. The following sample display allows the user to change the *QTY* and *SHIP* values:

```
                          UPDATE SHIP QUANTITY ON ORDERS
Order:  11589 Customer number:  11111   Customer name:  Al'Supply
ITEM     DESCRIPTION          QTY       SHIP            LOCATION
25764    Pliers                10         10            RST
33624    Hammer               500        250            RST
49821    Pliers               200        200            RST
26837    Wire Cutters          50         25            RST
```

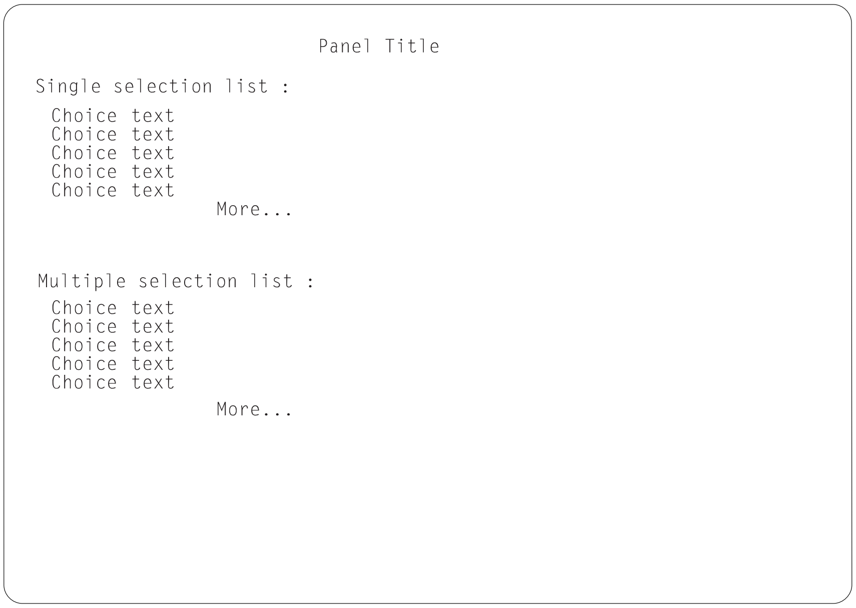
- Input only without validity checking, which allows the user to enter data as fast as possible; or input only with validity checking, which allows the user to enter data that is validity checked by the system or by the program for valid entries. The following sample display shows subfiles for input only:

```
Enter order number:  XXXXX
        ITEM NUMBER                   QUANTITY
        XXXXX                         XXX
        XXXXX                         XXX
        XXXXX                         XXX
        XXXXX                         XXX
        XXXXX                         XXX
        XXXXX                         XXX
        XXXXX                         XXX
        _____                         ___
```

- Combination of tasks, which can, for example, allow the user to change data as well as to enter new records. In the following example, the user can change existing names and addresses or enter new records.

```
                             CUSTOMER NAME SEARCH
 Search code:   41401
 NUMBER  NAME                    ADDRESS                 CITY       STATE
 41401   Adam's Home Repair____  121 Golden Circle_____  Chicago___  IL__
 41402   Jane's Radio/TV_______  135 Ransom Drive______  St Paul___  MN__
 41403   Advanced Electronics__  809 8th Street________  St Paul___  MN__
 _____   ______________________  ______________________  __________  ____
 _____   ______________________  ______________________  __________  ____
 _____   ______________________  ______________________  __________  ____
```

- Displaying single-choice and multiple-choice selection lists. A **single-choice selection list** is a potentially scrollable list from which the user can select one item. A **multiple-choice selection list** is a potentially scrollable list from which the user can select one or more items. For more information on selection lists, see “Selection Lists-Overview” on page 159.



```
                        Panel Title
 Single selection list :
   Choice text
   Choice text
   Choice text
   Choice text
   Choice text
                 More...


 Multiple selection list :
   Choice text
   Choice text
   Choice text
   Choice text
   Choice text
                 More...
```

RV3W077-0

Describing Subfiles in Your DDS Source

Each subfile you describe in your DDS source requires two types of record formats: a subfile record format and a subfile control record format.

- The **subfile record format** defines the fields in one row of the subfile.

  The high-level language program uses the subfile record format to read a subfile, write new records to a subfile, and update the subfile. Operations to the subfile record format are performed between the subfile and the high-level language program; the display is not changed on operations to a subfile record format.

  The subfile (SFL) keyword is required on the subfile record format.
- The **subfile control record format** contains heading information and controls subfile functions such as size, initialization, and clearing.

  The high-level language program performs operations on the subfile control record format to write the subfile to the display and to read the subfile from the display.

  The following DDS keywords are required on a subfile control record format:
  - Subfile control (SFLCTL) keyword, which identifies the subfile control record format for the subfile record format that immediately precedes it
  - Subfile size (SFLSIZ) keyword, which specifies the size of the subfile
  - Subfile page (SFLPAG) keyword, which specifies the size of the subfile page
  - Subfile display (SFLDSP) keyword, which specifies when to begin displaying records in a subfile

The DDS for the subfile record format must precede the DDS for the subfile control record format.

Each subfile has two types of records:

- An **active subfile record** is a record that has been:
  - Added to a subfile by a write operation.
  - Initialized as active by the subfile initialize (SFLINZ) keyword.
  - Changed when a write or update operation with the subfile next changed (SFLNXTCHG) keyword in effect was issued to the record.

- Changed by the user.

- An **inactive subfile record** is a record that was:
  - Not added to a subfile by the write operation.
  - Initialized as inactive by the SFLINZ keyword and the subfile records not active (SFLRNA) keyword.

You can also perform the following functions on subfiles:

Table 13. Optional Functions for Subfiles

Function	DDS keyword	Additional information
Allow a subfile to contain messages from a program message queue	Subfile message key (SFLMSGKEY), subfile message record (SFLMSGRCD), and subfile program message queue (SFLPGMQ)	See the **DDS** topic in the iSeries Information Center for more information about the subfile message keywords.
Clear the subfile of all records before new records are written	Subfile clear (SFLCLR)	The subfile is not erased from the display, however, until the SFLDSP keyword is in effect on the subfile control record. If the SFLCLR keyword is specified for a subfile with no records, it is ignored.
Control when to display a subfile control record	Subfile Display Control (SFLDSPCTL) or SFLDSP	The SFLDSP and SFLDSPCTL keywords are the only keywords that cause the contents of the display to change. The SFLDSPCTL keyword must be in effect if an input operation is done to retrieve the status of a CFnn or CAnn key even if no fields are displayed.
Delete the subfile to allow another subfile to be used or to continue processing the display file with no subfile used	Subfile delete (SFLDLT)	Normally, subfiles should not be deleted by the program. When the file containing the subfile is closed, the subfile is deleted automatically by the system. However, if the file is shared and is still open by another program, the subfile is not deleted, and you must delete it in your program. You should only delete a subfile if the maximum number of subfiles are already being used and you need to use another one. The SFLDLT keyword is ignored if the subfile does not exist.
Display a page of a subfile by a record number	Subfile record number (SFLRCDNBR)	If CURSOR is specified for the SFLRCDNBR keyword, the cursor is placed in the subfile record whose relative record number is identified by the contents of this field. The cursor is positioned at the first input-capable field in the subfile record. If there is no input-capable field, the cursor is positioned at the first output-only or constant field.

Table 13. Optional Functions for Subfiles (continued)

Function	DDS keyword	Additional information
Display a plus sign (+) in the lower corner at the extreme right of the subfile display area (page) when there are more records than fit on the display	Subfile end (SFLEND) or (SFLEND(*PLUS))	The plus sign is replaced by a blank when the last record is displayed. An option indicator must be specified with the SFLEND or SFLEND(*PLUS) keyword.
Display the word 'More...' on the line following the subfile display area (page) when there are more records than fit on the display	Subfile end SFLEND(*MORE)	The word 'More...' is replaced by the word 'Bottom' when the last record is displayed. An option indicator must be specified with the SFLEND(*MORE) keyword.
Display a scroll bar next to a subfile	Subfile end SFLEND(*SCRBAR)	For more information, see "Selection Lists-Overview" on page 159.
Return the relative record number of the record at the top of the current page of records	Subfile scroll (SFLSCROLL)	For more information, see "Selection Lists-Overview" on page 159.
Enable a command key to fold or truncate records in a subfile	Subfile fold (SFLFOLD) or subfile drop (SFLDROP)	If the SFLFOLD keyword is specified, the initial display of the records is folded. If the SFLDROP keyword is specified, the initial display of the records is automatically truncated. Then the user can press the command key to display the truncated or folded version, respectively, of the subfile record. If the page size equals the subfile size or the subfile fits on one display line, the specified keyword (SFLFOLD or SFLDROP) is ignored. Both SFLFOLD and SFLDROP can be used on the same subfile. Optional indicators can be used on these keywords. The optional indicators are used to determine which mode the subfiles are initially displayed in. If both keywords are optioned on or optioned off, then the subfile is initially displayed in folded mode. If the keyword is optioned off, the command key can still be used to display the truncated or folded version.
Enable the Enter key as the Roll Up key and enable a command key to return to the high-level language program	Subfile enter (SFLENTER)	If more than one subfile using SFLENTER is displayed at the same time, the only CAnn or CFnn key in effect as an Enter key is the CAnn or CFnn key specified for SFLENTER on the most recently displayed subfile. The cursor position at the time the Enter key is pressed determines which subfile is affected.

Table 13. Optional Functions for Subfiles (continued)

Function	DDS keyword	Additional information
Initialize a subfile with no active records even though the subfile is active	SFLINZ and subfile records not active (SFLRNA)	A record becomes active when one of the following happens: • An output operation is issued to the subfile for a specific record. The record is not considered changed unless the SFLNXTCHG keyword is used. • A user enters data into a displayed record. The record is considered active and changed. The records are displayed if the SFLDSP keyword is in effect. If default values were specified for fields in the records, they are included in the display.
Initialize all records by the field descriptions in the subfile record format in the display file	SFLINZ	When the SFLINZ keyword is in effect on an output operation to the subfile control record (SFLCTL), the system assumes that all option indicators on the subfile record are off; therefore, only those option indicators that are preceded by N are in effect. The subfile records are displayed if SFLDSP is in effect on an output operation. When the SFLINZ keyword is in effect on an output operation, the contents of input-capable fields without a default value are handled as follows: • Numeric fields are initialized to zeros. • Character fields are initialized to blanks. • Floating point fields are initialized to nulls.
Return a record to the program when a get-next-changed operation is performed	Subfile next changed (SFLNXTCHG)	The record is returned even if the record was not changed by the user
Roll by a specified number of records instead of by page	Subfile roll value (SFLROLVAL)	This field must have the keyboard shift attribute of signed numeric with zero decimal positions. It can be up to 4 digits long and must be defined as an output/input or input-only field.
Specify the number of spaces between each record on a line when more than one record is displayed on a line	Subfile line (SFLLIN)	This keyword is used for a horizontally displayed subfile. If the display file supports more than one screen size and the SFLLIN keyword is to apply to the secondary screen size in addition to the default (or primary) screen size, screen size condition names must be specified.

Table 13. Optional Functions for Subfiles (continued)

Function	DDS keyword	Additional information
Write a message to the message line on the display when your program does an output operation to the subfile control record	Subfile message (SFLMSG) and subfile message ID (SFLMSGID)	See the **DDS** topic in the iSeries Information Center for more information about the message keywords.
Determines where the cursor is located in a subfile	Subfile cursor relative record number (SFLCSRRRN)	The relative record number on which the cursor is positioned is returned in the hidden field specified as the parameter on the keyword.
Specify cursor progression for a subfile	Subfile cursor progression (SFLCSRPRG)	The SFLCSRPRG keyword causes the cursor to move from a field in a subfile record to the same field in the next displayed subfile record. For more information, see "Defining Cursor Progression for Entry Fields" on page 30.
Determines whether the subfile was in folded or truncated mode	Subfile mode (SFLMODE)	This is a required parameter and is only valid for subfile control records and the SFLCTL keyword must be specified.
Define a single-choice selection list	Subfile single-choice selection list (SFLSNGCHC)	For more information, see "Selection Lists-Overview" on page 159.
Define a multiple-choice selection list	Subfile multiple-choice selection list (SFLMLTCHC)	For more information, see "Selection Lists-Overview" on page 159.
Control the availability of choices in a selection list	Subfile choice control (SFLCHCCTL)	For more information, see "Selection Lists-Overview" on page 159.
Return all selected choices in a selection list using the get-next-changed operation	Subfile return selected choice (SFLRTNSEL)	For more information, see "Selection Lists-Overview" on page 159.

The DDS keywords can be specified in any order; however, the subfile record format (SFL) must precede the subfile control record format (SFLCTL).

You can use option indicators to condition many of the DDS subfile keywords.

You can specify a maximum of 512 subfiles in a display file, since the maximum number of record formats allowed in a display file is 1024. No more than 12 subfiles can be active at the same time to the same display station. One or more active subfiles can be displayed at the same time on the display station. A subfile must contain at least one field that can be displayed, and the subfile record format must not overlap the subfile control record format. If these records overlap, the display file cannot be created.

All named fields in a subfile record, including fields that are *not* input-capable, are returned to the program.

If any input data validity checking is specified for the subfile record, the validity checking is performed before any roll function is performed. If the data fails validity checking, the roll function is not performed.

When the relative record number of the record written to the subfile equals the subfile size, the system sends the program a CPF5003 message indicating that the subfile is full. (Not all records need to be active; that is, this message is sent even if the only record written to the subfile was the last record in the subfile.) If the subfile size does not equal the page size and the program then writes more records to the subfile, the system automatically extends the subfile as additional records are added. The program is not notified that the subfile has been extended. (A subfile cannot be extended past 9999 records.) Also, if the subfile size equals the page size, the program is not notified that the subfile is full unless the last record written to the subfile occupies the last line available on the subfile display area.

Processing of an extended subfile is less efficient because the extended space is not connected with the subfile. You can avoid extension by specifying a larger subfile size, but you will be wasting space if the extended space is used very seldom or never.

Figure 21 illustrates the order in which some of the DDS keywords used for subfile control are processed at run time:

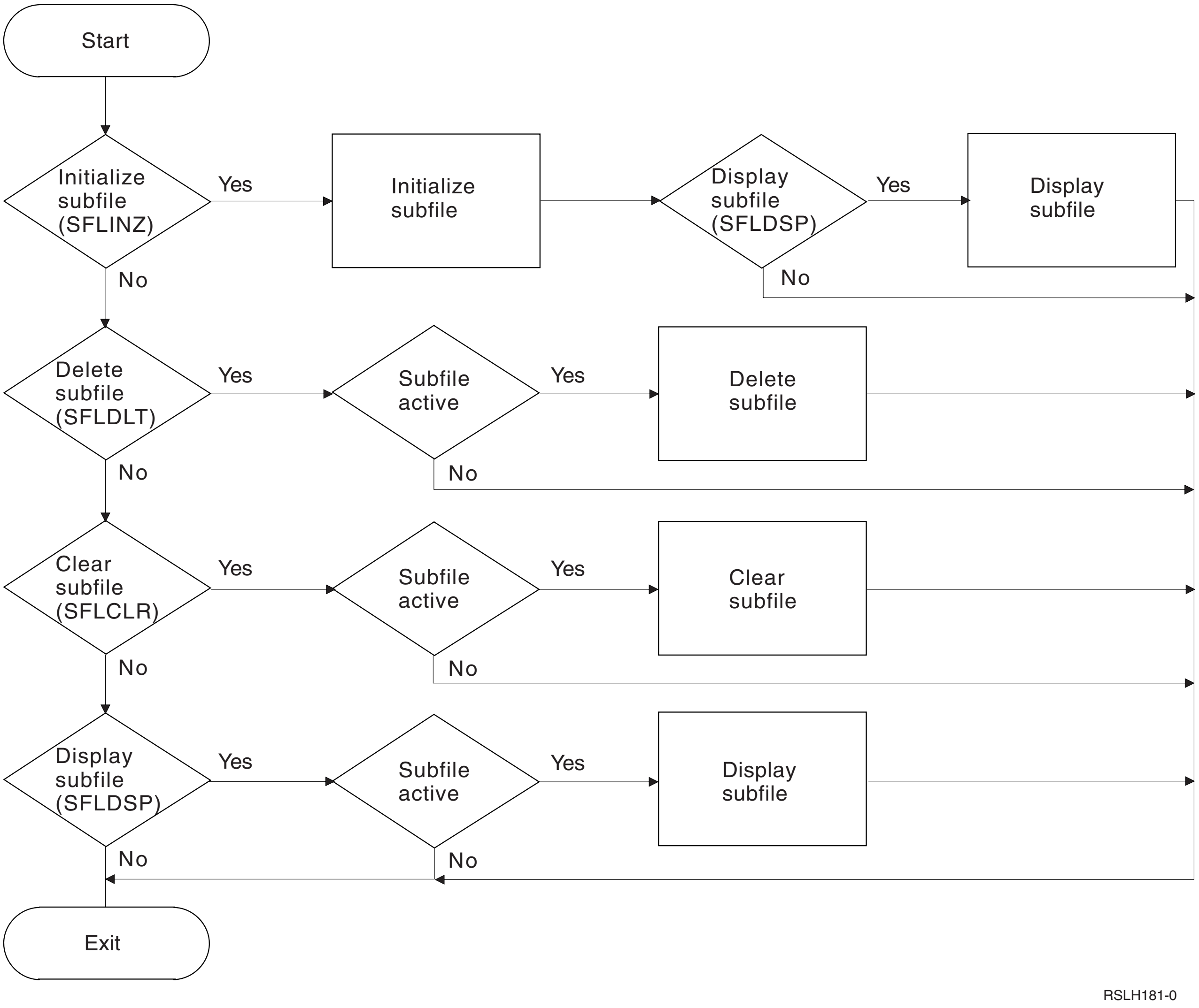
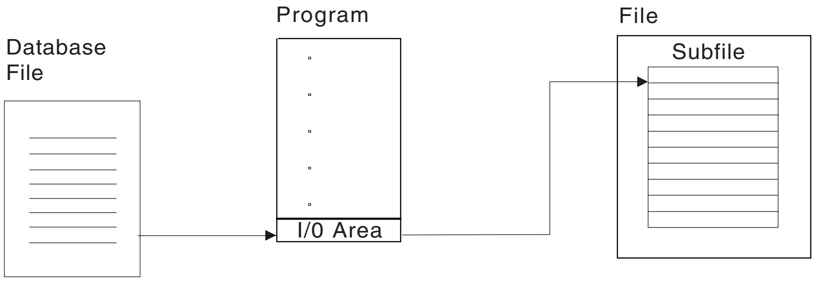


Figure 21. DDS Keyword Processing Order for Subfile Control

Using a Subfile in a Program

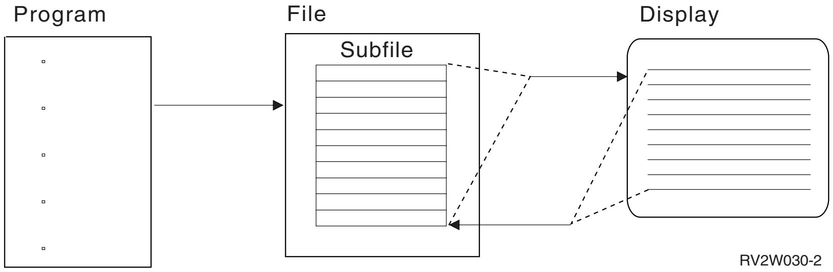
To use a subfile, you perform the following basic operations in your high-level language program:

1. Initialize the subfile. One way to initialize the subfile is to read records from a database file and write them to the subfile. Place the records in the subfile one at a time until the subfile is full or until there are no more records.



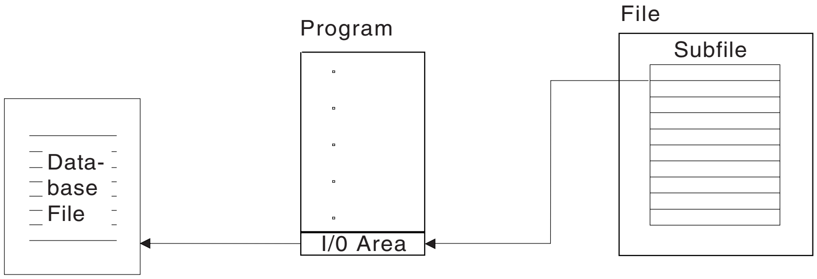
RV2W029-3

2. Send the subfile to the display in one output operation using the subfile control record format.
3. After the user reviews the records, changes them, or enters new records (depending on the function of the subfile), read the subfile control record format.



RV2W030-2

4. Process each record in the subfile individually, updating the database file or writing new records to the database file as required. If the function of the subfile is to update records, the program need only process the changed records by using the READC operation in ILE RPG or the Read Subfile Next Modified verb in ILE COBOL.



RV2W031-2

A display file that uses subfiles may display only a portion of a subfile at a time. The portion that is displayed is called a **subfile page**. The data entered into an input-only field on a subfile display goes to the subfile when a function key (such as a Roll key) is pressed. The field then displays a value in the subfile, and what happens when the Enter key is pressed depends on the application code.

Note: In a READC operation in ILE RPG, the data is moved from the subfile to the program. It does not remove it from the subfile, and it will continue to be displayed in the input-only field as if the fields were initialized to that value. Otherwise, it would appear that the subfile was empty when the data was actually there.

If the subfile is processed (for example, by an UPDAT operation in RPG), then the data is removed from the subfile, and the input-only fields are blanked out, reflecting the true condition of empty fields. This should be done after the READC operation moves the data to the program.

Requesting I/O Operations for a Subfile

An I/O request by a calling program to a subfile record format either writes a record to a subfile or reads a record from a subfile but never causes actual I/O to the display. To write subfile records to a display, the program must issue a request to the subfile control record format.

The valid requests that can be made to a subfile depend on whether the request is made to the subfile record format or the subfile control record format.

Requesting I/O Operations for a Subfile Record Format

By requesting the correct I/O operation for a *subfile record format*, you can do the following:

- Add a record (passed from a program) to a specified location in a subfile
- Update an active record that already exists in the subfile
- Read an active record at a specified location in the subfile
- Read the next changed record in the subfile that is greater than the relative record number previously read with a get-relative or get-next-changed operation

Adding a Record at a Specified Location in a Subfile

The **put-relative operation** adds a record (passed from a program) at a specified location in a subfile.

The location must be a valid relative record number in the subfile. The minimum relative record number is always 1. If the subfile size equals the subfile page, the maximum relative record number value is the subfile size value. If the subfile size is greater than the subfile page, the maximum relative record number value is 9999 because the system automatically extends the subfile as required. In addition, the relative record number cannot be the number of an active record already in the subfile. The relative record number is ignored when field selection is specified for the subfile record.

When a put-relative operation adds a record at the last record location (the subfile size value) in the subfile, a subfile-full condition occurs (message CPF5003). Both ILE RPG and ILE COBOL have special support for notifying the application program of this condition. See the appropriate high-level language manual.

The contents of input-capable fields without a default value specified are handled as follows:

- Numeric fields are initialized to zeros
- Character fields are initialized to blanks
- Floating point fields are initialized to nulls

Updating an Active Record in the Subfile

The **update operation** updates an active record that already exists in the subfile.

The active record must have been read before the update operation by a get request (either get relative or get-next-changed). No other I/O operations may be performed on the subfile to be updated between the read and the update. In addition, the subfile being updated may not be displayed again between the read and the update (for example, using subfile roll or SFLDROP processing).

Notes:

1. Some high-level languages do not allow I/O to any format in the display file between the read and the update of a single subfile record in the display file. Refer to the documentation for the high-level language you are using for more information.
2. If field selection is specified for the subfile record, only the fields that were selected when the record was placed in the subfile can be updated. Selecting different fields will cause results that cannot be predicted.

Reading an Active Record at a Specified Location in the Subfile

The **get-relative operation** reads an active record at a specified location in the subfile.

The location must be a valid relative record number in the subfile. The entire record, including response indicators (defined at the file level and on fields in a subfile record), input, output, output/input, and hidden fields, is passed to the program, the relative record number is placed in the input/output feedback area, and the record is no longer identified as a changed record. Response indicators defined at the file level are always returned as off. Response indicators defined on fields in a subfile record, such as the BLANKS or CHANGE keywords, are returned as on or off depending on the information in the field at the time the get operation was done.

If the record specified on the get-relative operation is not active, a not valid record number condition occurs (message CPF5020). This condition becomes a record-not-found condition in some high-level languages. See the appropriate high-level language manual.

Notes:

1. The get-relative operation and get-next-changed operation both update the relative record number in the input/output feedback area. Subsequent get-next-changed-record requests retrieve sequentially changed records greater than this relative record number.
2. The get-relative and get-next-changed operations do not process input data for overlapping fields in a subfile. The record returned to the program contains the data already existing in the buffer prior to the read operation for overlapped fields. If this is a problem, use the subfile initialize function to ensure all subfile fields are cleared.

Reading the Next Changed Record in a Subfile

The **get-next-changed operation** reads the next changed record in the subfile that is greater than the relative record number previously read with a get-relative or get-next-changed operation.

If the get-next-changed operation is used as the first read operation, the first changed record in the subfile is read. The entire record, including response indicators (defined at the file level and on fields in a subfile record), input, output, output/input, and hidden fields, is passed to the program, the relative record number is placed in the data management feedback area, and the record is reset to a *not changed* record. Response indicators defined at the file level are always returned as off. Response indicators defined on fields in a subfile record, such as the BLANKS or CHANGE keywords, are returned as on or off depending on the information in the fields at the time the get operation was done.

If there are no more changed records in the subfile, a message (CPF5037) indicating that the last changed record has already been retrieved, is sent to the program. See the appropriate high-level language manual for a description of how this condition is reported to your program.

If a record retrieved by a get-next-changed operation is updated and the SFLNXTCHG keyword is specified for an updated record, the updated record is set again as a changed record. This allows the program to ensure that the user has changed the record. For example, if the program detects an error in a record, it is advantageous to require the user to correct the error. The use of the SFLNXTCHG keyword allows the program to read that record again on a get-next-changed operation so it can continue to reject the record until the error has been corrected. The next get-next-changed operation does not retrieve this updated record. The record cannot be retrieved again with a get-next-changed operation until all the changed records following it in the subfile have been processed. This is because the changed records are accessed sequentially and the sequence does not start at the beginning until after the message indicating that there are no more changed records in the subfile has been sent to the program. A get-next-changed operation following this message gets the first changed record in the subfile. Because no I/O operation has been issued to the display, any changed record would be a record that was processed using the SFLNXTCHG keyword.

Notes:

1. The get-relative operation and get-next-changed operation both update the relative record number in the input/output feedback area. Subsequent get-next-changed-record requests retrieve sequentially changed records greater than this relative record number.
2. The get-relative and get-next-changed operations do not process input data for overlapping fields in a subfile. The record returned to the program contains the data already existing in the buffer prior to the read operation for overlapped fields. If this is a problem, use the subfile initialize function to ensure all subfile fields are cleared.

Requesting I/O Operations for a Subfile Control Record Format

By requesting the correct I/O operation for a *subfile control record format*, you can do the following:

- Display subfile records
- Place the subfile records on the display into the subfile for processing by the program
- Display and process subfile records at the same time

Displaying Subfile Records

You can display subfile records by issuing a **write operation** to the subfile control record format.

You can control the write operation using the following DDS keywords:

SFLDSP Display the subfile.

SFLDSPCTL Display the subfile control record.

SFLCLR Clear the subfile of active records.

SFLDLT Delete the subfile.

SFLINZ Initialize the subfile with active records, or if the SFLRNA keyword is specified, with inactive records. When the subfile is initialized, all option indicators in the subfile record are assumed to be off.

SFLEND Notify the user when the last available record is displayed.

SFLRCDNBR Display the specified page of the subfile.

Note: These keywords are described under "Describing Subfiles in Your DDS Source" on page 89.

Placing Subfile Records on the Display for Processing

A **read operation** must be issued to a displayed record format in order for the subfile records on the display to be placed into the subfile for processing by the program. The subfile records from the display are placed in their corresponding record positions in the subfile.

Displaying and Processing Subfile Records at the Same Time

The **write-read operation** is a single operation that combines the write and read operations and is more efficient than a single write operation followed by a single read operation.

Recognizing Subfile I/O Requests in High-Level Languages

Table 14 shows the I/O requests supported by the system and the equivalent high-level language operations:

Table 14. Subfile Operations Supported by the System and Equivalent HLL Commands

Operation	ILE C/C++ Function	ILE RPG Operation	ILE COBOL Statement	BASIC Statement
Put Relative	_Rwrited	WRITE, output specifications	WRITE SUBFILE	WRITE REC =

Table 14. Subfile Operations Supported by the System and Equivalent HLL Commands (continued)

Operation	ILE C/C++ Function	ILE RPG Operation	ILE COBOL Statement	BASIC Statement
Update	_Rupdate	UPDAT	REWRITE SUBFILE	REWRITE REC =
Get Relative	_Rreadd	CHAIN	READ SUBFILE	READ REC =
Get Next Changed	_Rreadnc	READC	READ SUBFILE NEXT MODIFIED	READ MODIFIED
Write	_Rwrite	WRITE	WRITE	WRITE
Read	_Rread	READ	READ	READ
Write-Read	_Rwriterd, _Rformat, _Rpgmdev	EXFMT		

Controlling the Appearance of Subfiles

Records in a subfile can be displayed either vertically or horizontally. In a vertically displayed subfile, a record is displayed on one or more lines, with each record beginning a new line (see Figure 22). In a horizontally displayed subfile, a record is complete on one line, and more than one record is displayed on a line (see Figure 23). You can specify that a subfile is to be displayed horizontally by using the SFLLIN keyword to define the number of spaces between each subfile record on a display line. Figure 24 on page 100 shows an example of a vertical subfile and a horizontal subfile being displayed at the same time.

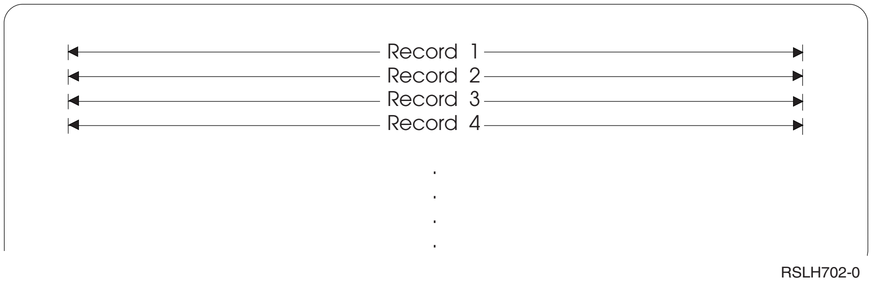


Figure 22. Vertically Displayed Subfile

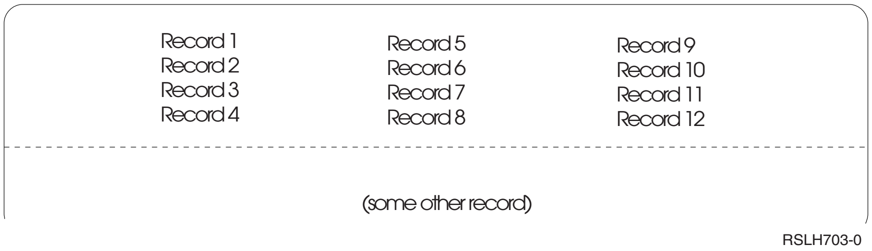


Figure 23. Horizontally Displayed Subfile

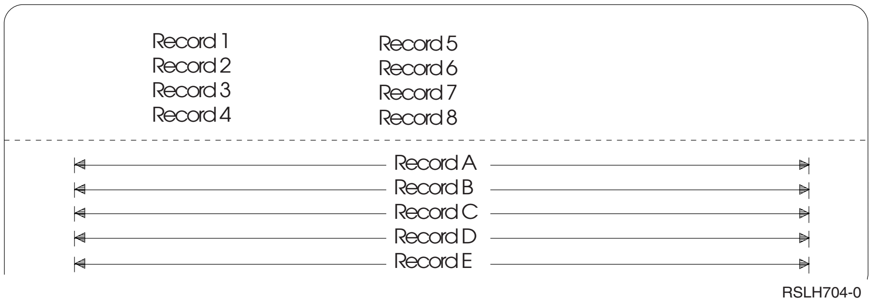


Figure 24. Horizontally and Vertically Displayed Subfiles Displayed at the Same Time

If a subfile is larger than the space allowed for the subfile on the screen, the user can roll the display from one group of records in the subfile to another. Each group of records displayed at the same time is called a page. When you create a display file with a subfile, you must specify the size of the page for a subfile by specifying the number of records in the page (SFLPAG keyword). Usually page size is based on the number of lines available on the display. You must also specify the size of the subfile by specifying the number of records in the subfile (SFLSIZ keyword).

Page size and subfile size can be the same; that is, all records in the subfile fit on one page. When page size equals subfile size, variable-length subfile records are supported. One record can take up only a single line while another record can take up more than one display line. Each record is placed in the first record position available in the subfile; this position is always a new line. In addition, the SFLDROP and SFLROLVAL keywords are ignored by display station support when page size equals subfile size.

For more information on page size and subfile size, see “Specifying Subfile Size Equal to Page Size” on page 102 and “Specifying Subfile Size Not Equal to Page Size” on page 103.

If records are to be displayed horizontally, the number of records to be displayed in a subfile (SFLPAG keyword) is adjusted so that the last line on the screen can be used to display a full line of records. For example, if the number of spaces between each record on a line (SFLLIN keyword) is specified such that six records fit on a line and 20 is specified for the page size (SFLPAG keyword), 20 is changed to 24, which is the nearest multiple of six. The number of records in the subfile (SFLSIZ keyword) is incremented by the same amount.

Note: For the initial display of a subfile, the more records placed in a subfile before it is displayed, the slower the response time.

Displaying Horizontal Subfiles with Display Modes

You can use the display mode (DSPMOD) keyword to specify which of the two **modes** (or display sizes), 24x80 or 27x132, you want to use for your display station.

When changing display modes, the display is cleared but the data is not cleared from the subfile. SFLDSP or SFLDSPCTL must be in effect for DSPMOD to be active in the control record.

The following example shows how to specify DSPMOD with subfiles:

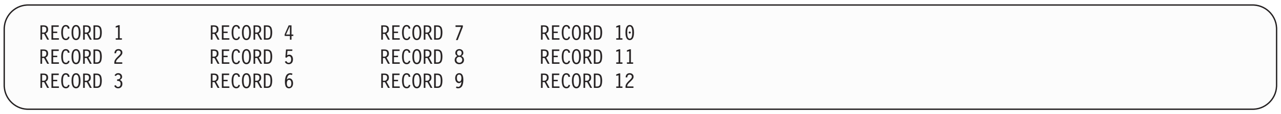
```
|...+....1....+....2....+....3....+....4....+....5....+....6....+....7...
     A                                     DSPSIZ(*DS4 *DS3)
     A          R SFLR                      SFL
     A            FLD1           8   O  1  5
     A            FLD2           7   I  1 16
     A            FLD3           7   B  1 24
     A          R SFLCTLR                   SFLCTL(SFLR)
     A                                      SFLDSP
     A                                      SFLDSPCTL
     A                                      SFLSIZ(60)
     A                                      SFLPAG(12)
     A                                      SFLLIN(4)
     A *DS3                                 SFLLIN(6)
     A  02                                  SFLEND
     A  10                                  DSPMOD(*DS3)
     A
     A
```

Figure 25. Sample DDS Using DSPMOD with Subfiles

In the previous example, if the user's program turns indicator 10 off and issues a write-read operation to the subfile control record format (SFLCTLR), the subfile is displayed as follows:

- *In 27 by 132 (*DS4) mode*, because indicator 10 for the DSPMOD keyword is off.
- *Horizontally*, because SFLLIN is specified. The SFLLIN value indicates the number of bytes between records. Because each record is 30 bytes long and the space between each record is 4 bytes long, four records can be displayed on one horizontal line, (4 x 30) + (3 x 4) = 132 bytes. The subfile is displayed on three lines because SFLPAG(12) is specified.

  The following example shows the subfile displayed in *DS4 mode.

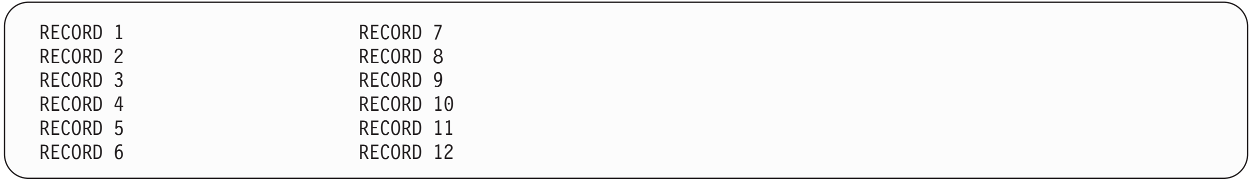


```
RECORD 1       RECORD 4       RECORD 7      RECORD 10
RECORD 2       RECORD 5       RECORD 8      RECORD 11
RECORD 3       RECORD 6       RECORD 9      RECORD 12
```

If the user presses the Enter key, control is returned to the user's program. If the user's program turns on indicator 10 and then issues another write-read operation to the subfile control record format (SFLCTLR), the subfile is displayed as follows:

- *In 24 by 80 (*DS3) mode*, because indicator 10 for the DSPMOD keyword is on.
- *Horizontally*, because SFLLIN is specified for the *DS3 mode. If SFLLIN was not specified for the *DS3 mode, the subfile would have been displayed vertically. If the SFLLIN keyword is to be used for more than one screen size, a screen size condition name for each secondary screen size is required. Because each record is 30 bytes long and the space between each record is 6 bytes long, two records can be displayed on one horizontal line, (2 x 30) + 6 = 66 bytes). The subfile is displayed on six lines because SFLPAG(12) is specified. To ensure other records are not erased, the SFLPAG may need to be specified for the secondary screen size.

  The following example shows the subfile displayed in *DS3 mode.



```
RECORD 1                    RECORD 7
RECORD 2                    RECORD 8
RECORD 3                    RECORD 9
RECORD 4                    RECORD 10
RECORD 5                    RECORD 11
RECORD 6                    RECORD 12
```

Specifying Subfile Size Equal to Page Size

You must specify the size of the subfile and the number of subfile records to be displayed at one time with the SFLSIZ and SFLPAG keywords. The use of subfile size equals page size is recommended when the number of subfile records to be displayed will fit on one page or when the number of records to be placed in the subfile is unknown and large. It is not an efficient use of resources to retrieve many database records to fill a large subfile if the user normally finds the needed information on the first page.

When subfile size equals page size, the system does not automatically support the use of the Roll Up and Roll Down keys. If you want the user to roll through the subfile using these keys, you must specify the ROLLUP or ROLLDOWN keyword in the subfile control record, and your program must handle the roll up or roll down function.

For example, if a subfile is used to allow the user to search through a long list, you can specify SFLSIZ equals SFLPAG and ROLLUP on the subfile control record:

```
|...+....1....+....2....+....3....+....4....+....5....+....6....+....7...
     A          R SFLCTLR                   SFLCTL(SFLRCD)
     A                                      SFLSIZ(16)
     A                                      SFLPAG(16)
     A                                      ROLLUP(20 'Roll Up')
```

Figure 26. Sample DDS Using SFLSIZ, SFLPAG, and ROLLUP Keywords

When the user presses the Roll Up key, indicator 20 is set on and control returns to the program. In your program, you would:

- Clear the subfile (a write operation to the subfile control record with the SFLCLR keyword in effect).
- Use the indicator to control a return to the logic that fills the subfile with another page of records.
- Display the new subfile page.

You could also allow the display station user to press a CFnn key to return to the start of the search. When the user presses the CFnn key, the associated indicator is set on and control returns to the program. In your program, you would:

- Clear the subfile.
- Use the indicator to control a return to the logic that built the first subfile page based on the search code entered. (The program needs to keep the original search code in order to do this.)

Using the ROLLDOWN keyword when subfile size equals page size requires more lines of code in the program because the program must keep track of the record position in the subfile and in the database file.

When the subfile size equals page size, you can use field selection and variable-length records in the subfile. If you use field selection, consider the following:

- If the fields are selected through the use of option indicators, the relative record number is ignored and each record is placed in the first available record position in the subfile.
- If a record is being updated, the field selection that does not match that on the original output is ignored. For example, assume that FIELD1 and FIELD2 are selected when the record is placed in the subfile. If the update selects FIELD2 and FIELD3, fields would overlay the original FIELD1 and FIELD2 fields, and the results could not be predicted.
- If field selection is specified on the subfile record, the number of records that can be displayed on the screen depends on the number of fields selected. When field selection is specified, the SFLPAG(value) keyword specifies the number of screen lines available to display the subfile record. In other cases, the SFLPAG(value) keyword specifies the number of subfile records that can be displayed at one time.
- The SFLFOLD, SFLDROP, and SFLROLVAL keywords are ignored.

When variable-length records are used, each record in the subfile is displayed beginning on the first available line on the page. If you use field selection for variable-length records, each record can take up a different number of lines on the display. Therefore, the number of records that actually fit in the subfile depends on the field selection of each record written to the subfile. The following shows an example of the DDS for a variable-length record:

```
|...+....1....+....2....+....3....+....4....+....5....+....6....+....7...
     A          R SFLRCD                    SFL
     A            ITMNBR         8Y 0   6  2
     A            ITMDSC        15      6 11
     A            QTYOH          4  0   6 28
     A            LSTPC          7  2   6 39
     A            ALLOH          8Y 0   6 49
     A            SLSMO         10  2   6 63
     A N80        SLSYR         12  2   7  7
     A N80        CSTYR         12  2     +3
```

Figure 27. Sample DDS for a Variable-Length Record

When indicator 80 is on, each record in the subfile fits on one line. However, when indicator 80 is off, each record uses two lines on the display.

Another typical use of variable-length records is where two or more entirely different formats are used to make up one format. In this case, each field would be separately conditioned by option indicators so that one record format might use multiple lines while another format uses only one line.

Specifying Subfile Size Not Equal to Page Size

Subfile size not equal to page size should be used when a finite number of records can be placed in the subfile and that number is small (for example, 50). The SFLSIZ keyword specifies the subfile size. The system allocates space to contain the subfile records based on the value specified for SFLSIZ. You should specify a value equal to the number of records that you normally have in the subfile. If your program places a record with a relative record number larger than the SFLSIZ value into the subfile, the system extends the subfile to contain it (up to a maximum of 9999 records).

When the subfile size is not equal to the page size, the use of the Roll Up and Roll Down keys is automatically supported.

To inform the user that there are more records in the subfile, use the SFLEND keyword on the subfile control record. When SFLEND is in effect (for example, the option indicator is on), a + (plus sign) is placed in the lower position to the extreme right of the screen on each page except the last page. On the last subfile page, the + is replaced with a blank.

When the subfile size is not equal to page size, you can use the SFLROLVAL keyword to allow the user to enter a value to specify how many records should be rolled up or down when the appropriate key is pressed. If the SFLROLVAL keyword is not used, the subfile is rolled by the SFLPAG value except for subfiles using SFLFOLD or SFLDROP. If the SFLFOLD or SFLDROP keyword is used, more records are displayed than the SFLPAG value when records are displayed in the truncated format. For truncated records, the display rolls by the number of records displayed in the truncated format. When the SFLROLVAL keyword is used and the Roll Up key is pressed, the uppermost record number in the displayed subfile is added to the roll value to determine the new uppermost record number. If this value is greater than the last record in the subfile, the last full page of records is displayed. If the Roll Up key is pressed when the last subfile page is displayed and the roll value is not less than the page size value, an error message is issued. If the roll value is less than the page size value, the roll function is performed.

Variable-length records and field selection cannot be used when the subfile size is not equal to the page size.

A technique to improve performance when you are using a multiple page subfile is to write only one page of subfile records at a time but use the operating system support to roll through the subfile. To do this, you need to define the ROLLUP keyword in DDS with a response indicator and also use the SFLRCDNBR keyword. In your program, you would write the records needed to fill one subfile page and then display that page. When the user wants to see more records, he or she presses the Roll Up key. The program then writes another page of records to the subfile, places the relative record number of a record from the second page into the SFLRCDNBR field, and displays the record.

The second page of subfile records is now displayed, and if the user presses the Roll Down key, the roll down is handled by the system. If the user presses the Roll Up key while the first page is displayed, the system will also handle the roll up. The program is notified only when the user attempts to roll up beyond the records currently in the subfile. The program would then handle any additional roll up requests in the same manner as for the second page. When you use this technique, the subfile appears to be more than one page because of the use of the roll keys. Yet, you can maintain good response time because the program only fills one subfile page before writing it to the display.

Checking Validity on Subfile Data

In addition to the DDS validity checking keywords (CHECK, COMP/CMP, RANGE, and VALUES), you can also do validity checking on subfile data in your program and require the user to correct the error.

For example, assume that you are using a subfile for an order entry program and you want to check the item number field to be sure it is a valid order number. You also want to check the quantity ordered field to ensure there are enough items on hand to fill the order. To do this, you can use the SFLNXTCHG keyword on the subfile record (SFL) to allow your program to diagnose the errors and require the user to correct them. The following DDS shows an example of using the SFLNXTCHG keyword:

```
|...+....1....+....2....+....3....+....4....+....5....+....6....+....7...
     A          R ORDENTD
     A                                  1 30'ORDER ENTRY DISPLAY'
     A                                  3  2'Enter customer number:'
     A            CUST          5      3 25
     A          R SFLRCD                    SFL
     A  61                                  SFLNXTCHG
     A            LINNBR        2      7  4
     A            ITMNBR        5   B  7  9
     A  40                                  DSPATR(RI PC)
     A            QTYORD        4   B  7 20
     A  35                                  DSPATR(RI PC)
     A          R SFLCTLR                   SFLCTL(SFLRCD)
     A                                      SFLSIZ(5)
     A                                      SFLPAG(5)
     A  55                                  SFLDSP
     A  50                                  SFLDSPCTL
     A  30                                  SFLCLR
     A  10                                  SFLINZ
     A  40                                  SFLMSG('Item number not valid' 40)
     A  35                                  SFLMSG('Qty not available' 35)
```

Figure 28. Sample DDS Using the SFLNXTCHG Keyword

When the program detects an error, it sets on the indicator that conditions the SFLNXTCHG keyword and issues a write operation to the subfile control record with the SFLDSP keyword in effect. The field in error is displayed in reverse image, and the cursor is positioned at that field. The associated error message is also displayed. The user then corrects the error.

A decision you must make when using the SFLNXTCHG keyword is whether to allow the user to change the subfile fields that were not in error. If you do not want the display station user to change those fields you can protect them with the DSPATR(PR) keyword. For those fields you do not want changed, the DSPATR(PR) keyword must be in effect only when the SFLNXTCHG keyword is in effect. If you allow the user to change the fields, you can:

- Define hidden fields for those fields that are to be checked.
- Move the data originally entered by the user into the hidden fields when an error occurs on a subfile.
- Compare the data in the hidden fields to the fields just read to identify which fields have been changed so you can update the records that have already been processed when the user makes the changes.

Displaying Error Messages from Subfiles

You can use a subfile to display messages for multiple errors. The messages to be placed in the subfile are on a program message queue. Each message written to the subfile is displayed on a separate line and is truncated, if necessary. Each message line contains an attribute character in position 1 that is displayed as a blank, followed by the message text. For the 24 by 80 display mode, 76 characters are displayed. For the 27 by 132 display mode, 128 characters are displayed. Because both the message identifier and the message data are available from the program message queue, message help and substitution text are supported for the messages placed in a message subfile. If the SFLMSGRCD keyword is specified, the SFLPGMQ and SFLMSGKEY keywords must also be specified.

The following shows an example of the DDS for a message subfile:

```
|...+....1....+....2....+....3....+....4....+....5....+....6....+....7...
     A          R MSGSFL                    SFL
     A                                      SFLMSGRCD(14)
     A            MSGKEY                    SFLMSGKEY
     A            PGMQ                      SFLPGMQ
     A          R MSGCTL                    SFLCTL(MSGSFL)
     A                                      SFLSIZ(8)
     A                                      SFLPAG(8)
     A  50                                  SFLDSP
     A  55                                  SFLDSPCTL
     A  60                                  SFLINZ
     A            PGMQ                      SFLPGMQ
     A            NBR            4  0H      SFLRCDNBR(CURSOR)
     A
```

Figure 29. Sample DDS for a Message Subfile

The SFLRCDNBR(CURSOR) keyword is used to position the cursor at the first displayed character in the message subfile that is specified in the SFLRCDNBR field so the Roll Up and Roll Down keys will apply to the message subfile.

For information on sending and receiving messages and on the program message queue, see the Messages section in the **CL** topic in the iSeries Information Center. Message subfiles are the only kind of subfiles supported for CL programs and for ILE C/C++ programs.

Positioning the Cursor on the Displayed Subfile

The DSPATR(PC) keyword lets you position the cursor for each page of the subfile record that is displayed. Write and update operations can be used to control DSPATR(PC) for:

- The initial display of subfile records. (A write or write-read operation to the subfile control record when the SFLDSP and SFLDSPCTL keywords are used.)
- Subfile records displayed using a roll key or a fold or truncate key (SFLDROP or SFLFOLD keywords).

Positioning the Cursor Initially

The cursor is positioned by the first of the following conditions that applies:

- The CSRLOC keyword on the subfile control record.
- The DSPATR(PC) keyword within the records being displayed.
- The DSPATR(PC) keyword within a field in the subfile control record.
- The SFLRCDNBR(CURSOR) keyword within the subfile control record.
- If nothing is specified, the cursor is positioned at the first input-capable field on the display.

Note: If the keyboard is unlocked prior to the output operation that displays the subfile, explicit cursor positioning is not performed.

Use the keywords in the following order:

- The CSRLOC keyword can be used to position the cursor anywhere on the screen.
- The DSPATR(PC) keyword can be used to position the cursor at any field in the first record displayed when the output operation specifies the SFLDSP keyword.
- The DSPATR(PC) keyword can be used to position the cursor at any field of the subfile control record.
- The SFLRCDNBR(CURSOR) keyword can be used to position the cursor at the first input-capable field of the record whose record number is used to select which page is to be displayed first. If no input fields exist, the cursor is positioned at the first selected output field in that record.
- If neither the DSPATR(PC) nor SFLRCDNBR(CURSOR) keyword is used, the cursor is positioned at the first input-capable field on the display.

Positioning the Cursor When a Roll Key Is Used

The positioning of the cursor when a roll key is used depends on whether the DSPATR(PC) keyword is used:

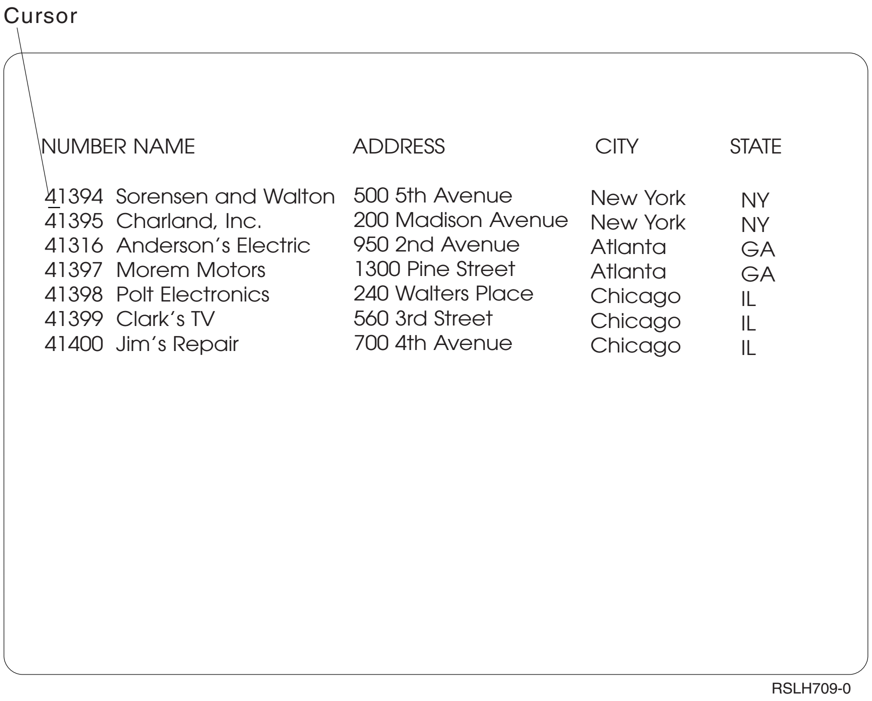
- If the DSPATR(PC) keyword is not used, the cursor is positioned at the same location as when the roll key was pressed.
- If the DSPATR(PC) keyword is used, the cursor is positioned at the first field in the displayed subfile records with the DSPATR(PC) keyword in effect.

The following example illustrates both and shows part of the DDS for a subfile in which records are displayed vertically. Customer number, name, address, city, and state are displayed. A user can change customer name, address, city, and state. Customer number cannot be changed; it is an output field only. The DSPATR(PC) keyword has been specified for the customer number field (CUST). Subfile size is 21 and page size is 7.

```
|...+....1....+....2....+....3....+....4....+....5....+....6....+....7...
     A          R SUBFIL1                   SFL
     A                                      TEXT('Subfile record')
     A            CUST           5      4  3DSPATR(PC)
     A            NAME          20   B  4 10CHECK(LC)
     A            ADDR          20   B  4 32CHECK(LC)
     A            CITY          20   B  4 54CHECK(LC)
     A            STATE          2   B  4 76
     A          R FILCTL1                   SFLCTL(SUBFIL1)
     A  50                                  SFLDSPCTL
     A  55                                  SFLDSP
     A                                      SFLSIZ(21)
     A                                      SFLPAG(7)
     A  60                                  SFLCLR
     A                                      TEXT('Subfile control record')
     A                                      OVERLAY
     A                                      PROTECT
     A                                      CA03(98 'End of program')
     A                                  2  2'NUMBER'
     A                                  2 10'NAME'
     A                                  2 32'ADDRESS'
     A                                  2 54'CITY'
     A                                  2 76'STATE'
```

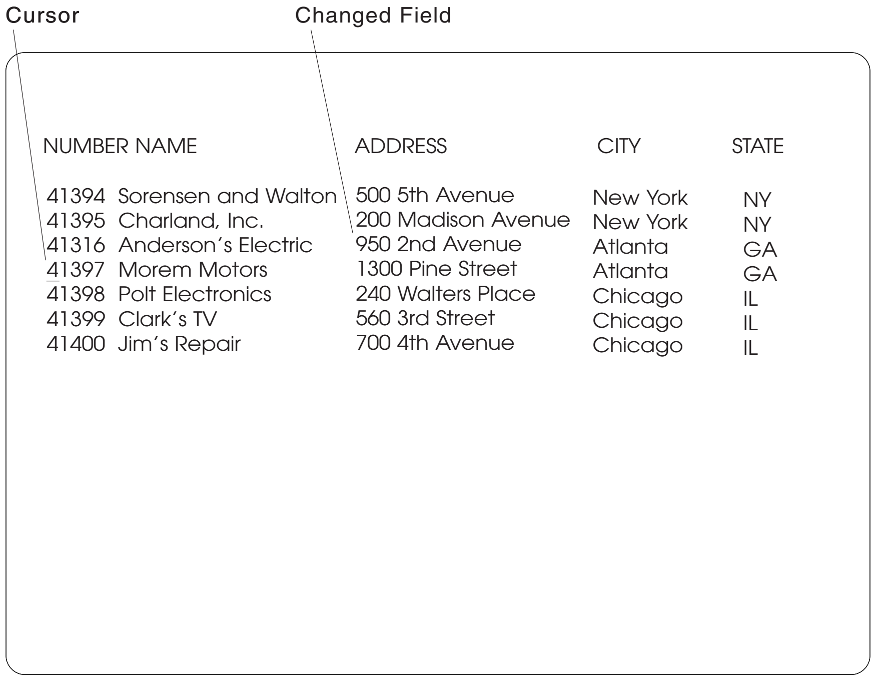
Figure 30. Sample DDS Using the DSPATR(PC) Keyword

The initial display looks like this:



```
NUMBER NAME                 ADDRESS               CITY        STATE

41394  Sorensen and Walton  500 5th Avenue        New York    NY
41395  Charland, Inc.       200 Madison Avenue    New York    NY
41316  Anderson's Electric  950 2nd Avenue        Atlanta     GA
41397  Morem Motors         1300 Pine Street      Atlanta     GA
41398  Polt Electronics     240 Walters Place     Chicago     IL
41399  Clark's TV           560 3rd Street        Chicago     IL
41400  Jim's Repair         700 4th Avenue        Chicago     IL
```

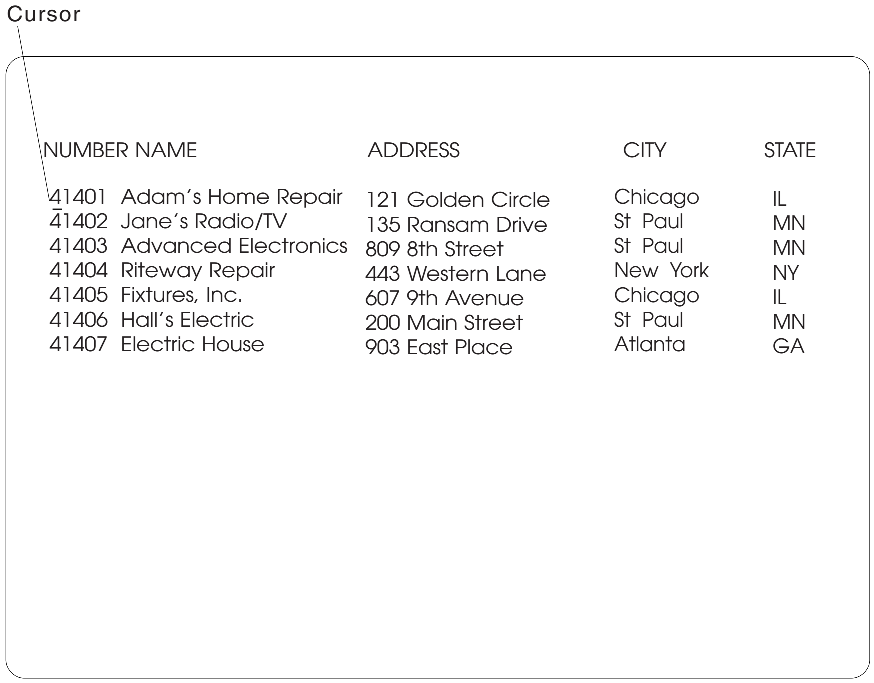
The first seven records in the subfile are displayed and the cursor is positioned under the customer number in the first record. The user moves the cursor to the third record, updates the address for that customer, and moves the cursor to the customer number of the fourth record:



```
NUMBER NAME                    ADDRESS              CITY        STATE

 41394  Sorensen and Walton  500 5th Avenue        New York     NY
 41395  Charland, Inc.       200 Madison Avenue    New York     NY
 41316  Anderson's Electric  950 2nd Avenue        Atlanta      GA
 41397  Morem Motors         1300 Pine Street      Atlanta      GA
 41398  Polt Electronics     240 Walters Place     Chicago      IL
 41399  Clark's TV           560 3rd Street        Chicago      IL
 41400  Jim's Repair         700 4th Avenue        Chicago      IL
```

RSLH183-0

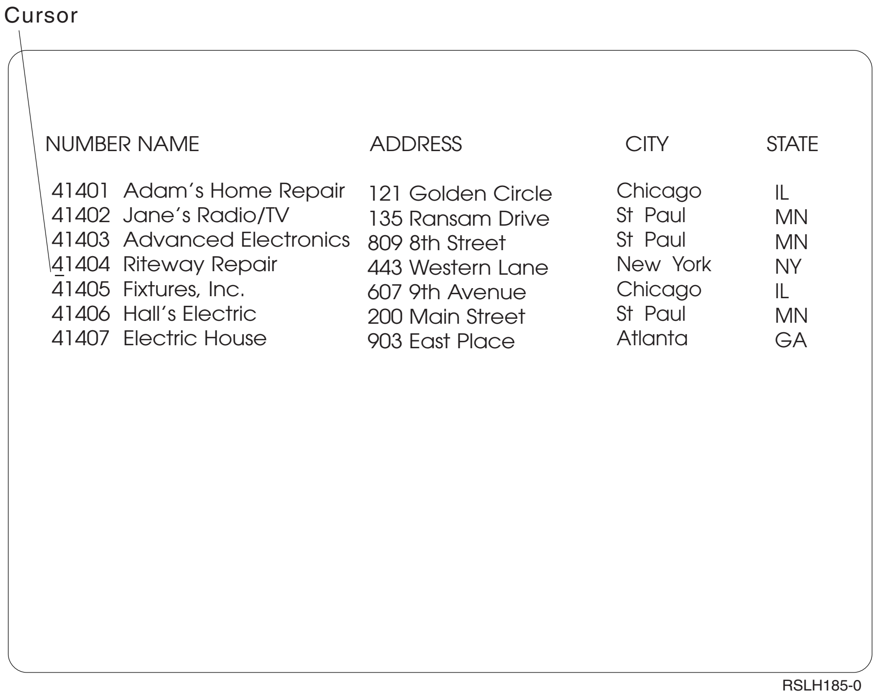
Now the user presses the Roll Up key to display the next seven records. The cursor is positioned under the customer number in the first record:



```
NUMBER NAME                       ADDRESS               CITY         STATE

 41401  Adam's Home Repair      121 Golden Circle     Chicago       IL
 41402  Jane's Radio/TV         135 Ransam Drive      St  Paul      MN
 41403  Advanced Electronics    809 8th Street        St  Paul      MN
 41404  Riteway Repair          443 Western Lane      New  York     NY
 41405  Fixtures, Inc.          607 9th Avenue        Chicago       IL
 41406  Hall's Electric         200 Main Street       St  Paul      MN
 41407  Electric House          903 East Place        Atlanta       GA
```

RSLH184-0

If the DSPATR(PC) keyword had not been specified and the user pressed the Roll Up key, the cursor would have been positioned at the fourth record under customer number:



```
NUMBER NAME              ADDRESS           CITY       STATE

 41401  Adam's Home Repair   121 Golden Circle  Chicago    IL
 41402  Jane's Radio/TV      135 Ransam Drive   St Paul    MN
 41403  Advanced Electronics 809 8th Street     St Paul    MN
 41404  Riteway Repair       443 Western Lane   New York   NY
 41405  Fixtures, Inc.       607 9th Avenue     Chicago    IL
 41406  Hall's Electric      200 Main Street    St Paul    MN
 41407  Electric House       903 East Place     Atlanta    GA
```

RSLH185-0

Positioning the Cursor When a Fold or Truncate Key Is Used

When you use the SFLFOLD or SFLDROP keyword to assign a CFnn or CAnn key, cursor positioning is handled the same way as described under "Positioning the Cursor When a Roll Key Is Used" on page 106. The cursor is positioned as specified for all displayed records, including those folded. When you use the SFLFOLD or SFLDROP keyword to assign a CFnn or CAnn key, the cursor is positioned only for those fields displayed. Cursor positioning specifications for fields in the folded portion of the record are ignored.

Positioning the Cursor and Rolling When Two or More Records Are Displayed

When you display two or more records at the same time, the position of the cursor determines the action taken when the user presses a roll key, regardless of which record was written to the display last.

The cursor can be positioned in the *roll-enabled area* of the display or in the area that is *not roll-enabled* of the display. A *roll-enabled area* is:

- A record without subfiles and with the ROLLUP/ROLLDOWN keyword in effect
- A subfile control record with the ROLLUP/ROLLDOWN keyword in effect
- A roll-enabled subfile, which is an active subfile with subfile size greater than page size
- An active subfile with subfile size equal to page size and the ROLLUP/ROLLDOWN keyword in effect for its subfile control record

Based on the location of the cursor, the action taken when the user presses a roll key is as follows:

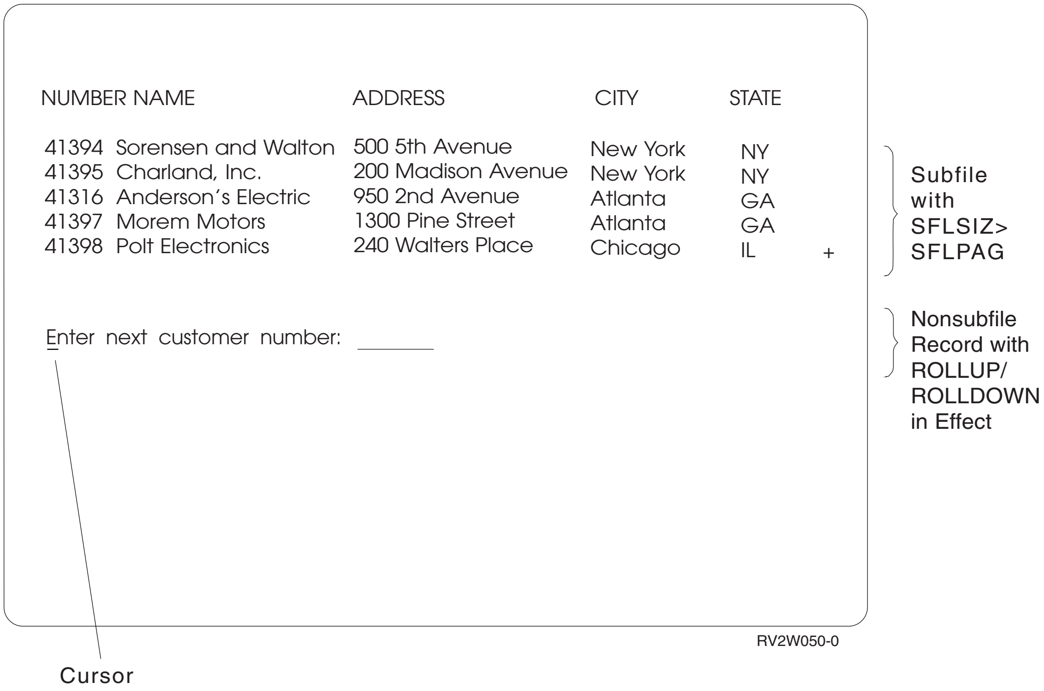
- If the cursor is positioned in the roll-enabled area at a roll-enabled subfile or at the subfile control record for a roll-enabled subfile, the subfile is rolled. If the subfile is at the end of the subfile and the corresponding ROLLUP/ROLLDOWN keyword is not in effect, the end-of-subfile message is sent to the user. If the ROLLUP/ROLLDOWN keyword is in effect at the end of the subfile, control returns to the program.

- If the cursor is positioned in the roll-enabled area at a record without subfiles with the ROLLUP/ROLLDOWN keyword in effect, at a subfile control record with the ROLLUP/ROLLDOWN keyword in effect, or at an active subfile with subfile size equal to page size and the ROLLUP/ROLLDOWN keyword in effect for the subfile control record, control returns to the program.
- If the cursor is not positioned in the roll-enabled area, the system attempts to find the uppermost roll-enabled area on the display and perform the action indicated, as listed previously. If there is no roll-enabled area on the display, the command-key-not-valid message is sent to the user.

  Note: Records that do not occupy display space (record formats with no fields, with hidden, program-to-system, or message fields only, or with the CLRL keyword specified and no input-capable fields; and message subfiles) are assumed to be at line 0. Therefore, these records are considered to be uppermost on the display, and the system attempts to roll them first.

The following examples illustrate what action is taken, based on the location of the cursor, when two records are displayed and the user presses a roll key.

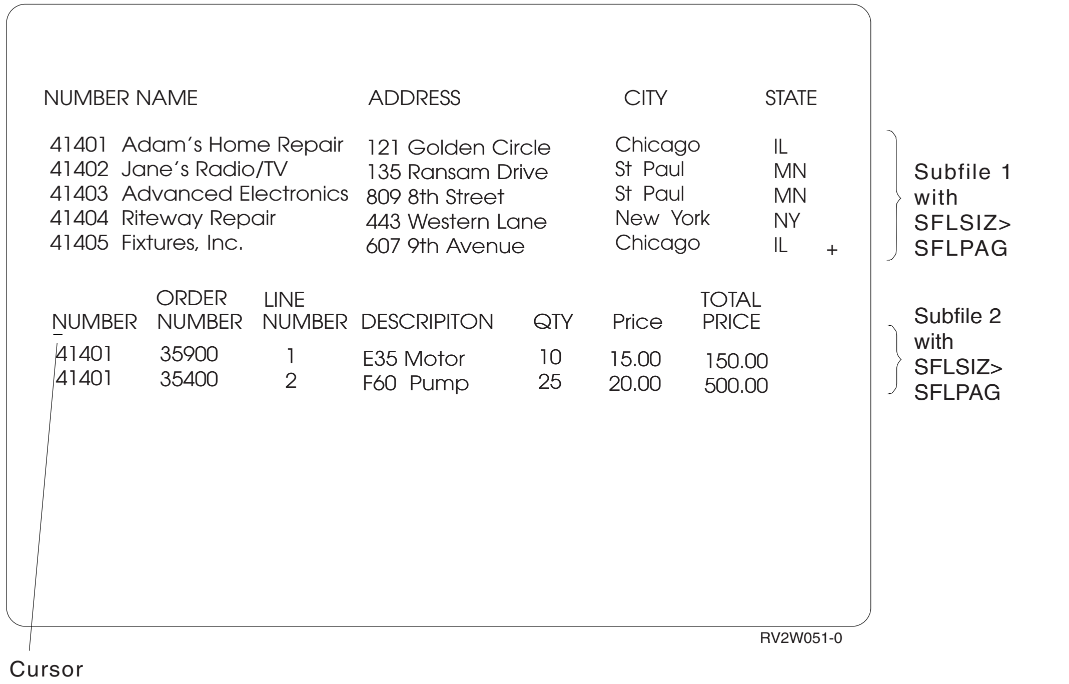
In the following example, control returns to the program if the corresponding ROLLUP/ROLLDOWN keyword is in effect. This occurs because the cursor is positioned at a record without subfiles and with the ROLLUP/ROLLDOWN keyword in effect.



```
NUMBER NAME                  ADDRESS              CITY        STATE

 41394  Sorensen and Walton  500 5th Avenue       New York     NY
 41395  Charland, Inc.       200 Madison Avenue   New York     NY
 41316  Anderson's Electric  950 2nd Avenue       Atlanta      GA
 41397  Morem Motors         1300 Pine Street     Atlanta      GA
 41398  Polt Electronics     240 Walters Place    Chicago      IL      +


 Enter  next  customer  number:  _______
```

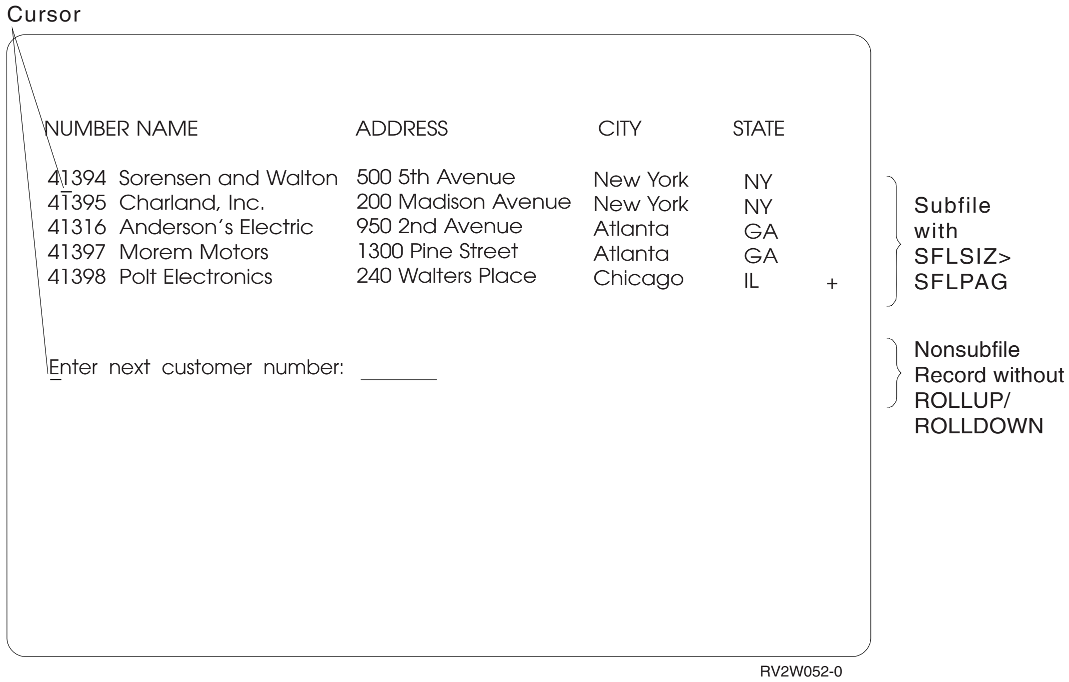
Subfile with SFLSIZ> SFLPAG
Nonsubfile Record with ROLLUP/ ROLLDOWN in Effect
RV2W050-0
Cursor

In the following example, two subfiles with the subfile size greater than the page size and their control records are displayed. The user positions the cursor in the bottom subfile control record. The bottom subfile is rolled because the cursor is positioned at a roll-enabled subfile in the roll-enabled area of the display.



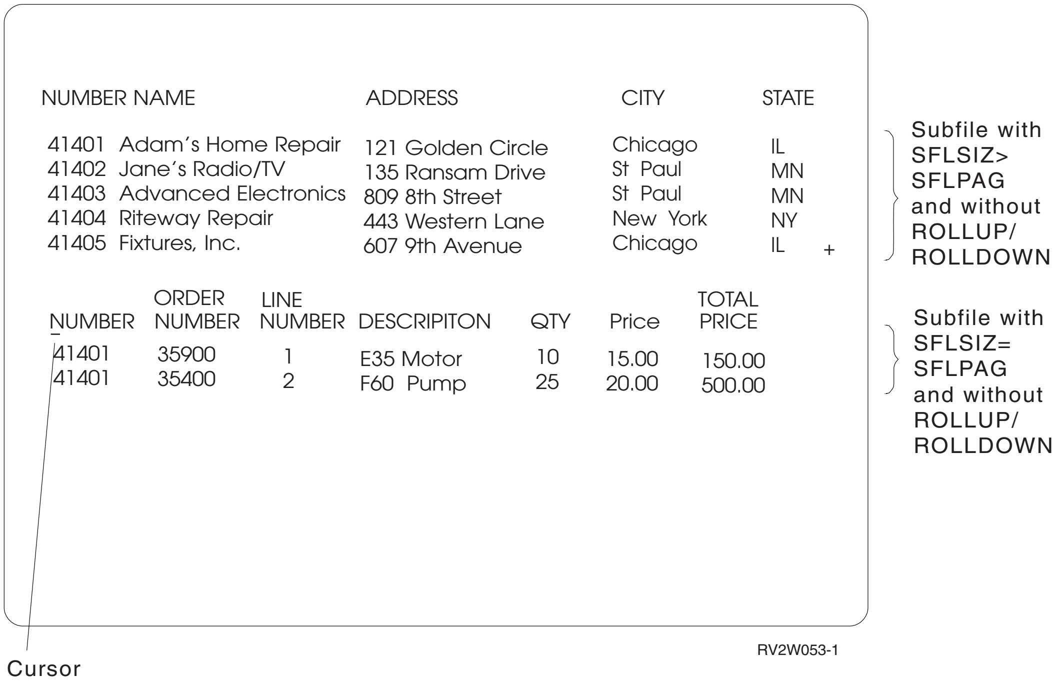
```
NUMBER NAME                  ADDRESS              CITY          STATE

 41401  Adam's Home Repair   121 Golden Circle    Chicago        IL
 41402  Jane's Radio/TV      135 Ransam Drive     St  Paul       MN
 41403  Advanced Electronics 809 8th Street       St  Paul       MN
 41404  Riteway Repair       443 Western Lane     New  York      NY
 41405  Fixtures, Inc.       607 9th Avenue       Chicago        IL     +

          ORDER     LINE                                    TOTAL
NUMBER   NUMBER    NUMBER   DESCRIPITON     QTY    Price    PRICE

 41401    35900      1      E35 Motor       10     15.00    150.00
 41401    35400      2      F60  Pump       25     20.00    500.00
```

In the following example, a subfile with the subfile size equal to the page size and the ROLLUP/ROLLDOWN keyword in effect is written to the display first. A record without subfiles and without the ROLLUP/ROLLDOWN keyword in effect is then written to the display. If the cursor is positioned at either the subfile record or the second record, control returns to the program.



```
NUMBER NAME                  ADDRESS               CITY        STATE

 41394  Sorensen and Walton  500 5th Avenue        New York     NY
 41395  Charland, Inc.       200 Madison Avenue    New York     NY
 41316  Anderson's Electric  950 2nd Avenue        Atlanta      GA
 41397  Morem Motors         1300 Pine Street      Atlanta      GA
 41398  Polt Electronics     240 Walters Place     Chicago      IL     +


 Enter  next  customer  number:  _______
```

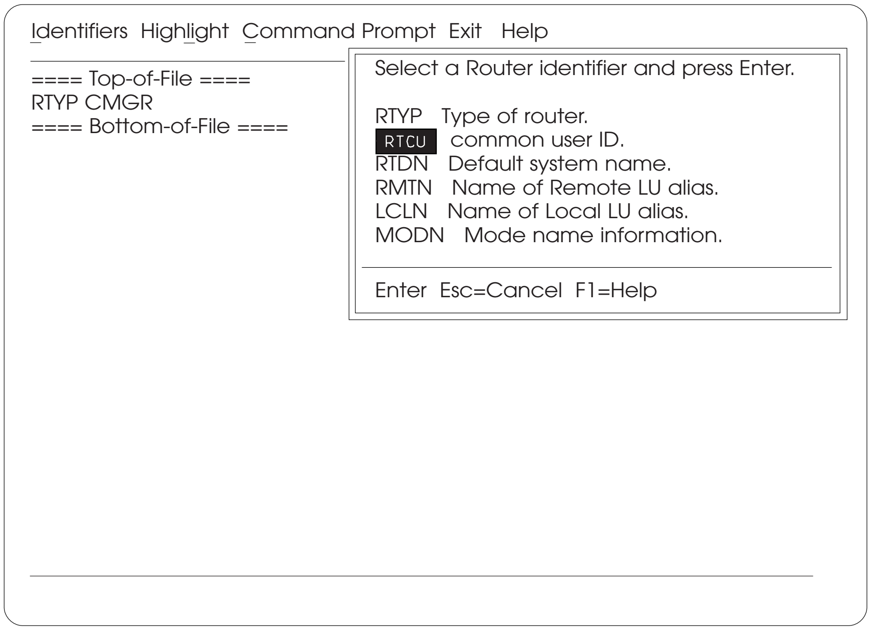
In the following example, the first subfile has a subfile size greater than the page size but the ROLLUP/ROLLDOWN keyword is not specified. The second subfile has a subfile size equal to the page size but the ROLLUP/ROLLDOWN keyword is not specified. If the cursor is positioned at the second subfile, the first subfile is rolled. In this position, the cursor is not in a roll-enabled area; therefore, the system finds the uppermost roll-enabled area in the display and performs the roll function.



```
NUMBER NAME                ADDRESS              CITY        STATE

 41401  Adam's Home Repair  121 Golden Circle   Chicago      IL
 41402  Jane's Radio/TV     135 Ransam Drive    St  Paul     MN
 41403  Advanced Electronics 809 8th Street     St  Paul     MN
 41404  Riteway Repair      443 Western Lane    New  York    NY
 41405  Fixtures, Inc.      607 9th Avenue      Chicago      IL    +

         ORDER    LINE                                    TOTAL
NUMBER   NUMBER   NUMBER  DESCRIPITON   QTY    Price      PRICE
 41401   35900     1      E35 Motor     10     15.00      150.00
 41401   35400     2      F60  Pump     25     20.00      500.00
```

Subfile with SFLSIZ> SFLPAG and without ROLLUP/ ROLLDOWN
Subfile with SFLSIZ= SFLPAG and without ROLLUP/ ROLLDOWN
Cursor
RV2W053-1

In the following example, the subfile with a subfile size greater than the page size and without ROLLUP/ROLLDOWN keyword is written to the display first. The record without subfiles and with ROLLUP/ROLLDOWN keyword in effect is then written to the display above the subfile. If the cursor is positioned within the subfile record, the subfile is rolled. If the cursor is not positioned within the subfile record, the subfile is not rolled and control returns to the program.



```
Identifiers  Highlight  Command Prompt  Exit   Help
==== Top-of-File ====      Select a Router identifier and press Enter.
RTYP CMGR
==== Bottom-of-File ====   RTYP   Type of router.
                           RTCU   common user ID.
                           RTDN   Default system name.
                           RMTN   Name of Remote LU alias.
                           LCLN   Name of Local LU alias.
                           MODN   Mode name information.

                           Enter  Esc=Cancel  F1=Help
```

Understanding Subfile DDS and Program Logic-Example

The following shows the DDS that describes the customer name search subfile shown earlier in this section. The DDS is followed by a description of the logic a program would use to process this subfile.

```
|...+....1....+....2....+....3....+....4....+....5....+....6....+....7...
     A** DISPLAY CUS220D  CUSTOMER NAME SEARCH
     A                                      CF03(99 'End of Program')
     A          R NAMESR                    OVERLAY
     A                                  1 29'CUSTOMER NAME SEARCH'
     A                                  3  2'Search code'
     A            SEARCH         5   I  3 15
     A          R SUBFIL1                   SFL
     A            CUST           5      7  2
     A            NAME          20   B  7  9
     A            ADDR          20   B  7 31
     A            CITY          20   B  7 53
     A            STATE          2   B  7 75
     A          R FILCTL                    SFLCTL(SUBFIL1)
     A  55                                  SFLDSPCTL
     A  50                                  SFLDSP
     A                                      SFLSIZ(18)
     A                                      SFLPAG(6)
     A  50                                  SFLEND
     A  60                                  SFLCLR
     A                                      OVERLAY PROTECT
     A            RCDNBR         2  0H      SFLRCDNBR(CURSOR)
     A  45                              5  2'NUMBER'
     A  45                              5  9'NAME'
     A  45                              5 31'ADDRESS'
     A  45                              5 53'CITY'
     A  45                              5 75'STATE'
```

Figure 31. Sample DDS Showing Customer Name Search Subfile

The following is an example of the logic a user program would use to process the subfile just shown. A write-read operation is a combined input and output operation. A read is an input operation. A write is an output operation. See the appropriate high-level language manual for the operations that can be performed in the high-level language program.

User Program

1. Opens a file and issues a write-read operation to the NAMESR record format to prompt for a search code.

User

2. Enters a zip code in the search code field. The program uses the search code field as a key field to find the first database record in the file with that key field. The program will build the subfile using that record as the first record in the subfile.

User Program

3. Obtains records (read operation) from the database file and places (write operation to SUBFIL1) them in the subfile one record at a time until the subfile is full or there are no more records to place in the subfile.
4. When all records are in the subfile, issues a write-read operation to the subfile control record format (FILCTL) with the following:
   a. A + (plus sign) is displayed in the lower right corner of the screen when there are more records than fit on one subfile page. Because the indicator for the SFLEND keyword is on, the system replaces the + with a blank when the last subfile page is displayed.

b. The indicator for the SFLDSP keyword is on, so the first subfile page is displayed.

c. The indicator for the SFLDSPCTL keyword is on, so the subfile control record is displayed.

d. The SFLRCDNBR(CURSOR) keyword is specified for a field. The program placed a value of 1 in this field, so the subfile page that contains relative record number 1 is displayed first and the cursor is positioned at the first input field in that record. (If no input field exists, the cursor is positioned at the first selected output field or constant in that record.)

e. The constant field (heading line) indicators are on, so the constants in the subfile control record are displayed.

f. The OVERLAY and PROTECT keywords are in effect in the subfile control record so that the prompt (NAMESR) can remain on the display without being changed.

User

5. Updates displayed records, using the roll keys to display different subfile records as needed. Presses the Enter key after completing all updates to the subfile.

User Program

6. Completes the input portion of the write-read operation to the subfile control record format. In this example, the subfile control record does not contain any input fields. The input portion of the write-read operation allows the display station user to enter data into the subfile.

7. Issues the get-next-changed operation to the subfile record to process the first subfile record changed by the user.

8. Uses each changed record to update the corresponding database file record.

9. Repeats steps 7 and 8 until a no-more-modified-records condition exists for step 7. When this condition is detected, step 10 is performed.

10. Issues a write-read operation to the prompt (NAMESR) to determine if the program should end or display another group of database records. The OVERLAY keyword is specified, so that the current display contents for the subfile are left unchanged. If the user presses the CF1 key, which turns on response indicator 99, the program will close the display file and end. If the user enters another search code for another group of records to be displayed, step 11 is performed.

11. Issues a write operation to the subfile control record (FILCTL) with the following:

    a. The indicator for the SFLCLR keyword is on, so the subfile is cleared of all records (the display is unchanged).

    b. The indicator for the SFLDSP keyword is off, so the contents of the display remains unchanged.

    c. The indicator for the SFLDSPCTL keyword is off, so the subfile control record is not displayed again.

12. Repeats steps 3 through 10. The subfile was cleared (SFLCLR keyword) in step 11 so that new records can be placed in the subfile. The write operation to the subfile control record in step 4 has the constant field indicator off so that heading information is not sent to the display again. As long as the subfile control record remains on the display (no intervening write operations without the OVERLAY keyword in effect have been performed), the fields do not have to be sent to the display again.

Chapter 5. Defining Windows with Display Files

This chapter explains how to use the specialized DDS window keywords to create windows in your applications. The DDS window keywords provide the simplest, most flexible method of creating windows for a variety of purposes. For example, they allow you to use subfiles to present data in windows, to have the system automatically save and restore the underlying display, and to position data in the window by referring to positions in the window itself instead of positions on the full screen.

Use the DDS window keywords if your windows must use other DDS functions, such as subfiles, display attributes, validity checking, and optioning. Also use the DDS window keywords when the window contains multiple input fields, or if the window location can be varied.

If you have help information defined with DDS and are using the HLPRCD keyword to display it, the WINDOW keyword can be used to easily display the information in a window.

The following sections describe the terminology used for windows, the functions of the window keywords, and how to use the keywords for the following tasks:

- Creating windows
- Defining window borders
- Reading data from windows
- Changing window borders and contents
- Moving and duplicating windows
- Making two windows seem active at one time
- Making one window in a series stand out
- Removing windows
- Improving application performance by bypassing system save and restore operations

For examples on using the window keywords, see “Programming Examples” on page 128.

For some applications, you might want to use a different method of creating windows:

- Chapter 20, “Defining Online Help Information,” on page 399 describes how to create help windows with the user interface manager (UIM). The UIM uses a different language from DDS. However, it automates many help window functions for you and provides a simple way of adding online help information to your existing applications.
- The *IBM WindowTool/400 PRPQ,* SC41-0050, describes how to create windows with the WindowTool/400 PRPQ. Consider using this program if you use windows primarily to construct application menus.
- Chapter 6, “Creating a Graphical Look for Displays” describes how to create menu bars, pull-down menus, selection fields, continued-entry fields, and how to use edit masks using DDS keywords.

Window Terminology

A **window** is information that overlays part of the display. The user can view information inside the window and the portion of the display that is not overlayed by the window. However, only the window is active; the user cannot work with the underlying display. When more than one window is displayed, only one window is active at a time.

The **active window** is the window subject to the most recent input or output operation. The active window appears to be the topmost window on the display. It is the only part of the display with which the work station user can interact.

A window remains on the display until your application or the system takes action to remove it. Removing a window and overlaying a window are different operations. When a window is **removed**, it no longer exists on the display, and you can no longer write to or read from it. When a window is **overlayed** by another window, it might not be visible to the work station user; however, it is still available for you to work with.

DDS Window Keywords

Four DDS keywords allow your applications to create and work with windows:

WINDOW (Window)
Creates a window on the display, changes the contents of an existing window, or makes an existing but inactive window active again.

WDWBORDER (Window Border)
Specifies the color, display attributes, and characters of a window border.

WDWTITLE (Window Title)
Specifies the text, color, and display attributes for a title of a window. The title is embedded in the top or bottom border of the window.

Note: Not all controllers support text in the bottom border of windows, nor the left and right alignment of text in the top or bottom border.

RMVWDW (Remove Window)
Removes other windows from the display when a new window is displayed or when an existing window is redisplayed as the active window.

USRRSTDSP (User Restore Display)
Prevents the i5/OS system from automatically saving and restoring the underlying display when windows are displayed and removed. In some situations, save and restore operations are unnecessary; bypassing them speeds up your application. You can also use the USRRSTDSP keyword to make an earlier window in a series pop up and overlay later windows, and to make two windows seem active at the same time.

For detailed reference information about each keyword, see the **DDS** topic in the iSeries Information Center .

Window Representation and Hardware Configuration

Window borders appear differently, depending on the type of display station and work station controller you are using.

For more information on how windows appear on different hardware configurations, see “Hardware Configuration” on page 139.

Creating Windows

The record-level keyword WINDOW allows a record format to be displayed inside a window. A maximum of 12 windows can be created on a display at one time.

To create a window, write a record that specifies a WINDOW keyword. The first window record you write must be a window definition record specifying the window size and its location on the display. The window definition record places the window borders on the display. Then you can write the same window definition record again or use one or more window reference records to complete the specifications for the window.

Window Definition Records

A window definition record is a record containing a WINDOW keyword that defines the window size and location. Size and location attributes include the position of the upper left corner of the window border and the number of rows and columns within the window.

A window definition record must be the first record written for each window you display. It is the record that actually creates the window and makes it visible on the display. The record can contain the same types of fields or data found in any typical record. It can also contain a WDWBORDER keyword defining the window border. (The WDWBORDER keyword can also be used at the file level.)

You can supply all the specifications for the window in the window definition record. You can supply additional specifications by writing the same window definition record again, as you would when displaying an error message in the window. You can also supply additional specifications by writing one or more window reference records after you write the window definition record.

Window Reference Records

Window reference records provide additional data to be placed in the window. They allow you to display more than one record format in a window.

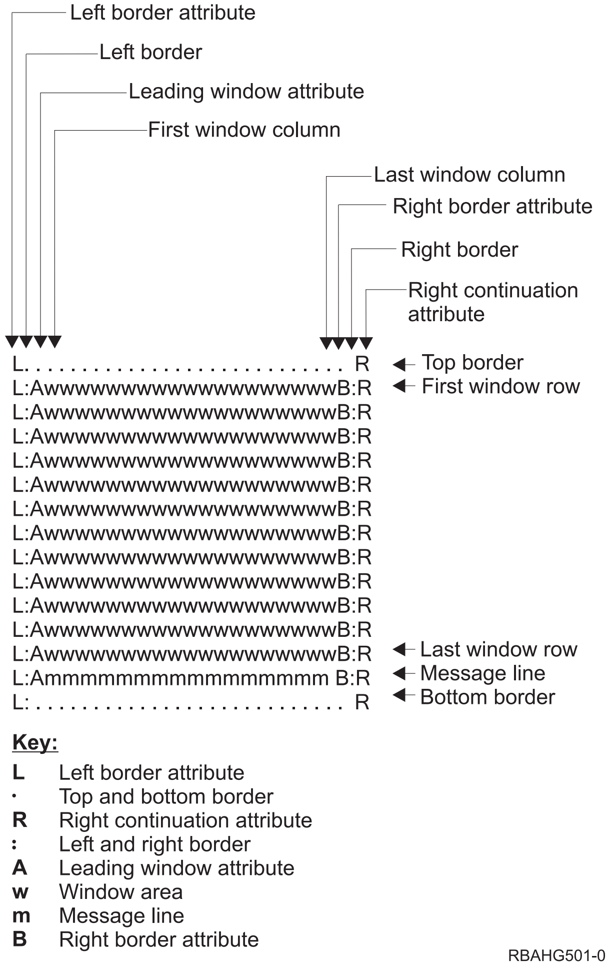
Each window reference record contains a WINDOW keyword specifying the name of the window definition record to which it applies. When the window reference record is written, the window definition record being referred to must be on the display. If the referenced record is not on the display, then a notify message stating that the window does not exist is returned to the application.

Window reference records do not contain size and position attributes, and any active WDWBORDER keywords are ignored.

You can use as many window reference records as needed to complete the window. However, you do not need to write any window reference records to display a window. You can display a window using only a window definition record.

Window Size and Location

The following diagram shows the parts of a window that is created when a window definition record is displayed:


Left border attribute
Left border
Leading window attribute
First window column
Last window column
Right border attribute
Right border
Right continuation attribute
L. . . . . R
L:AwwwwwwwwwwwwwwwwwwwwwB:R
L:AwwwwwwwwwwwwwwwwwwwwwB:R
L:AwwwwwwwwwwwwwwwwwwwwwB:R
L:AwwwwwwwwwwwwwwwwwwwwwB:R
L:AwwwwwwwwwwwwwwwwwwwwwB:R
L:AwwwwwwwwwwwwwwwwwwwwwB:R
L:AwwwwwwwwwwwwwwwwwwwwwB:R
L:AwwwwwwwwwwwwwwwwwwwwwB:R
L:AwwwwwwwwwwwwwwwwwwwwwB:R
L:AwwwwwwwwwwwwwwwwwwwwwB:R
L:AwwwwwwwwwwwwwwwwwwwwwB:R
L:AwwwwwwwwwwwwwwwwwwwwwB:R
L:Ammmmmmmmmmmmmmmmm B:R
L: . . . . R
Top border
First window row
Last window row
Message line
Bottom border
Key:
L Left border attribute
· Top and bottom border
R Right continuation attribute
: Left and right border
A Leading window attribute
w Window area
m Message line
B Right border attribute
RBAHG501-0

This window is specified as having a depth of 13 rows (also known as lines) and a width of 19 columns (also known as positions). The usable area inside the window borders is 12 rows deep and 19 columns wide. Row 13 is reserved for messages; it cannot contain fields.

With its border, the window actually takes up additional rows and columns on the display. All windows take up two more rows, one each for the top and bottom borders. Most windows also take up six more columns:

- Two for border attributes
- Two for border characters
- One for the leading window attribute
- One for the continuation attribute on the right.

If a window starts and ends in the first and last columns of the full display, it takes up only four more columns instead of six. If a window row overlays DBCS fields, that row requires the six columns mentioned previously plus up to four more columns for DBCS shift-out and shift-in characters on each side.

The window is positioned by the upper left corner of the border, which has a starting row position equal to the top border row and a starting column position equal to the left border column. Fields that do not fit within the window are diagnosed during file compilation. If the SLNO keyword is used and specifies the starting line number, the DDS compiler flags any fields that do not fit within the window. If *VAR is specified on the SLNO keyword and the run-time starting line number does not allow the entire record to be displayed, an exception occurs.

The DDS compiler diagnoses window location problems when either the upper left window line or upper left window position is specified as a constant. A run-time error occurs if a dynamically positioned window does not fit on the display.

The special value, *DFT, can be specified in place of the start-line and start-position parameters on the window keyword. *DFT indicates that the system will determine the start line and start position of the window. The window is positioned relative to the cursor location. The system uses the following sequence of rules to position the window when *DFT is used:

1. If the window will fit below the cursor position on the display, it is placed there. The top window border is positioned one row below the cursor. If possible, the left window border is positioned in the same column as the cursor; if not, the window is positioned as far to the left of the cursor as necessary for it to fit on the display.
2. If the window will fit above the cursor position, it is placed there. The bottom window border is positioned one row above the cursor. The window is positioned horizontally as described in step 1.
3. If the window will fit to the right of the cursor position, it is placed there. The right window border is positioned in the next-to-last column of the display. If possible, the top window border is positioned in the same row as the cursor; if not, the window is positioned as far above that as necessary for it to fit on the display.
4. If the window will fit to the left of the cursor position, it is placed there. The right window border is positioned two columns to the left of the cursor. The window is positioned vertically as described in step 3.
5. If the window cannot be positioned in any of the above areas, it is placed in the bottom right corner of the display.

Cursor Position

To position the cursor in a window, use the CSRLOC and DSPATR(PC) keywords in the same way as for a full-screen display. The cursor is positioned with reference to the upper left corner of the usable area of the window.

If *RSTCSR is specified on the WINDOW keyword, and the cursor is moved outside the usable area of the active window, only the Print and Home command function (CF) keys are active. If the work station user presses any other command function (CF) key, the alarm sounds and the cursor is moved back to its position for the previous write operation. *RSTCSR is the default.

Note: On display stations attached to a controller that supports an enhanced interface for nonprogrammable work stations, the cursor can be moved out of a window only with a mouse when *RSTCSR is specified.

If *NORSTCSR is specified on the WINDOW keyword, the user may move the cursor out of the active window and use any command function (CF) or command attention (CA) key.

Error Messages

When windows exist on a display and *MSGLIN is specified on the WINDOW keyword for the window, any error messages are displayed on the last usable line of the active window. The last usable line in the window is reserved for error messages; no records are displayed there. If the error message is longer than the line, it is truncated to fit. When windows exist on a display and *NOMSGLIN is specified on the WINDOW keyword for the window, any error messages are displayed at the bottom of the display or the location defined by the MSGLOC keyword.

Help is available through the Help key for error messages displayed in windows.

When messages reporting operational and keyboard errors, such as `Function key not allowed`, are displayed, the keyboard is locked and the user must press the Error Reset key to continue.

An informational message, stating that mismatching shift-out and shift-in characters were sent to the display, is placed in the job log under the following circumstances:

- The base display has a DBCS field that spans more than one line.
- A window is displayed and part of the DBCS field from the base display is on the window message line.
- A function key is pressed that results in an operational or keyboard error.

A few message-related keywords function differently when used for windows:

- The ERRSFL keyword is ignored. Its function is performed only when there are no windows on the display.
- The MSGLOC keyword is ignored if *MSGLIN is specified on the WINDOW keyword. Its function is performed only when there are no windows on the display or when *NOMSGLIN is specified on the WINDOW keyword.
- Messages resulting from the ERRMSGID, ERRMSG, SFLMSG, SFLMSGID, and DDS validity-checking keywords are displayed in the window but do not lock the keyboard. Such messages do lock the keyboard when displayed on full-screen displays.

Subfiles

A maximum of 24 subfiles can be active at any one time. A maximum of 12 subfiles can be displayed on the base display or in a single window at any one time.

If a subfile is displayed in a window and the window is removed from the display, the subfile is not deleted. The subfile remains active until the display file is closed or you explicitly delete the subfile.

DDS Help Records

If a window definition record is written to the display as a DDS help record, the current display is suspended as it normally is for application help, and the application help record is written to the display as a window. To avoid errors when using window records as help records, adhere to these requirements:

- Include the ASSUME keyword in the display file containing the help record. If the ASSUME keyword is not present, the rest of the display is blank while the help window is displayed.
- Use a value of *YES for the Restore Display (RSTDSP) parameter when creating, changing, or overriding the display file. When returning from help, RSTDSP(*YES) restores the suspended display and puts the cursor back where it was when the Help key was pressed.
- Use variable line and position values in your window definition record to allow the operating system to position the help window dynamically, according to cursor position. The system uses the same sequence of rules to position the window as if *DFT were specified on the window keyword. These rules are described in “Window Size and Location” on page 118. If the window definition record specifies that only the line value or only the position value is variable, the same rules are followed. However, the constant value is not changed.

Defining Window Borders

You can use the system defaults for your window borders or define them using the WDWBORDER keyword. This keyword specifies three border components: color, display attributes, and characters. More than one WDWBORDER keyword can be specified. You can use option indicators with the WDWBORDER keyword.

You can use the WDWBORDER keyword at the file level, where it applies to each window definition record in the file, or on individual window definition records. If you use it on window reference records, a warning message is issued when the file is created.

The following sections describe window border defaults, how the system handles multiple window border definitions, and how to use the WDWBORDER keyword to define UIM help window borders.

Border Defaults

When you do not use the WDWBORDER keyword, the system defaults are as follows:

Border Element	Default
Color	Blue on color displays. On noncolor display stations, this attribute is ignored.
Display attribute	Normal (that is, no attributes such as highlighting or reverse image)
Top and bottom border character	Period (.)
Left and right border character	Colon (:)
Top left and right corner character	Period (.)
Bottom left and right corner character	Colon (:)

Note: RUMBA/400 work stations and InfoWindow® II display stations attached to a controller that supports an enhanced interface have solid-line window borders. For more information, see Table 15 on page 139.

Multiple Border Definitions

When the WDWBORDER keyword is specified at the file level and in a window definition record, the parameter values of the keyword at the file and record level are combined. If the parameter values conflict, the record-level parameter value is used. For example, if the following is specified at the file level:

```
WDWBORDER((*COLOR RED) (*DSPATR RI))
```

and the following is specified at the record level:

```
WINDOW(2  5  10  20) +
WDWBORDER((*COLOR GRN)  +
      (*CHAR '........'))
```

then the window border consists of green periods in reverse image.

If more than one WDWBORDER keyword is specified at the same level, the parameters for the keywords that are in effect are combined. If different values are specified for the same parameter, the parameter value of the first keyword in effect is used. The values for individual components of the border are determined when a window definition record is written. The border values are determined by the following hierarchy:

1. Start with system defaults.
2. Override the defaults with any file-level border specifications.
3. Override any file-level border specifications with record-level border specifications.

The process is similar when more than one WDWBORDER keyword is in effect at the same level and the keywords specify the same WDWBORDER component (color, attribute, or character). The first component value is used. For example, assume that two WDWBORDER keywords are in effect at the file level:

```
WDWBORDER((*COLOR GRN
           *CHAR '........'))
WDWBORDER((*CHAR '---|||-|'))
```

and the following is specified at the record level:

```
WDWBORDER((*COLOR BLU))
```

then the border component values are determined as follows:

1. Start with the defaults for each component.
2. Override the character component and color component defaults with the file-level values. Because the character component is specified more than once, use the first character component value specified.
3. Override the file-level color value with the record-level value.

The window border is constructed using the record-level border color, the first file-level border characters, and the default border display attributes.

If a single WDWBORDER keyword does not specify all three border components, then those not specified use the values from any other WDWBORDER keywords in effect; they do not use the defaults. In the preceding example, this is demonstrated at the record level. Only the color component is specified. However, because the character component is specified at the file level, the file-level value is used instead of the default. Because the display attribute component is not specified at the record or file level, the default is used.

UIM Help Window Borders

You can use the WDWBORDER keyword to specify the border attributes of help windows created with UIM panel groups. Assume that you are using DDS for full-screen displays and the UIM for help windows. If a window is on the display when the Help key is pressed, the UIM help window has the same borders as the DDS window. If no windows are active but the WDWBORDER keyword is specified at the file level or at the record level of a nonwindow record currently on the display, the system determines the border attribute and character values for the UIM help window by combining the file-level, record-level, and default values. Assume that no windows are currently displayed. First, a nonwindow record with the WDWBORDER keyword specified is written to the display. A UIM help window is then written to the display. The UIM help window borders use the attributes specified in the WDWBORDER keyword on the nonwindow record.

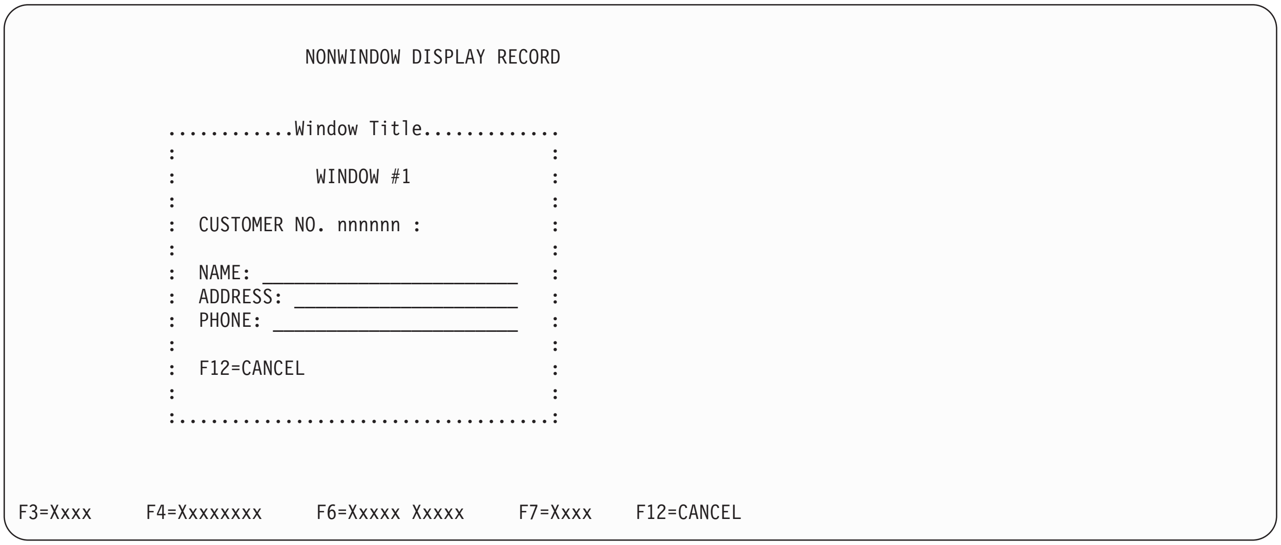
Defining a Window Title

Use the window title (WDWTITLE) keyword to specify the text, color, and display attributes for a title of a window. The title is embedded in the top or bottom border of the window. The length of the title can be up to the number of positions specified on the window-positions parameter specified on the associated window definition keyword.

Note: Some controllers do not support text in the bottom border of windows.

The WDWTITLE keyword must be specified on a record that contains a WINDOW keyword (in the definition format). If a WINDOW keyword that refers to another window is also specified, a warning message is issued.

Figure 32 on page 123 shows an example of a window title.



```
                        NONWINDOW DISPLAY RECORD


             ...........Window Title............
             :                                 :
             :            WINDOW #1            :
             :                                 :
             :  CUSTOMER NO. nnnnnn :          :
             :                                 :
             :  NAME: _____________________    :
             :  ADDRESS: __________________    :
             :  PHONE: ____________________    :
             :                                 :
             :  F12=CANCEL                     :
             :                                 :
             :.................................:


F3=Xxxx    F4=Xxxxxxxx    F6=Xxxxx Xxxxx    F7=Xxxx   F12=CANCEL
```

Figure 32. Window Title-Display Example

DDS for a Window Title-Example

```
|...+....1....+....2....+....3....+....4....+....5....+....6....+....7...
     A          R RECORD1                  WINDOW(6 15 18 51)
     A N01                                 WDWTITLE((*TEXT &TTL1) (*COLOR GRN))
     A  01                                 WDWTITLE((*TEXT &TTL1) (*COLOR RED))
     A            FIELD1         5A  B  2  2
     A            FIELD2        20A  B  8  5
     A            TTL1          10A  0P
     A*
     A          R RECORD2                  WINDOW(8 20 9 30)
     A                                     WDWTITLE((*TEXT &TTL2) +
     A                                              (*COLOR YLW) +
     A                                              (*DSPATR RI))
     A            FIELD3         5A  B  2  2
     A            FIELD4        20A  B  8  5
     A            TTL2          10A  0P
```

Figure 33. DDS for a Window Title

If the window defined by RECORD1 in Figure 33 is written to the display, the title will be whatever text is contained within the TTL1 field. The title will appear centered in the top border of the window. If indicator 01 is set off, the text will be green. If indicator 01 is set on, the text will be red.

If the window defined by RECORD2 is written to the display, the the title will be whatever text is contained within the TTL2 field. The title will appear centered in the top border of the window. The title will be in reverse image and yellow.

Reading Data from Windows

When windows are on the display, you can receive input only from the active window (that is, the last window written to the display). If your application reads a window record and the window exists on the display but is not the active window, then any windows subject to more recent input or output operations are removed. The window containing the record to be read is restored, becoming the active window. Then the record is read from the display.

If your application attempts to read a window record and the window does not exist on the display, a notify message stating that the window does not exist is returned. If the application attempts to read a window record and the window is on the display but the record is not on the display, a message stating that the record is not on the display is returned.

Changing Window Borders and Contents

To change the contents of a window already on the display, you must write that window's window definition record or a window reference record specifying that window to the display. To change the borders of a window already on the display, you must write that window definition record to the display.

If a window definition record is written to the display and a window with that name already exists on the display in the same position, a new window is not created. The new record is considered a normal write operation to the existing window. If the new record specifies different border attributes or characters, the new attributes or characters are displayed when the record is written.

Moving and Duplicating Windows

If a window definition record is written to the display and a window with that name already exists on the display in a different position, a new window is created. The new window appears in the specified position and has the same name as the existing window. To move the window, use the RMVWDW keyword in the window definition record being written. The existing window with that name and any other windows on the display are removed when the new window is written. In effect, the window is moved. If you do not use the RMVWDW keyword, the same window appears on the display in two different positions.

If a window reference record is written to the display and the specified window is on the display, the record is written to the most recently created window with the specified name. To write to an earlier window with that name, use a window definition record specifying the earlier window's location.

Making Two Windows Seem Active at Once

Although only one window can be truly active at a time, you can make two windows appear to be active at once. You might use this to display two windows side by side and allow the work station user to switch back and forth between them.

To make two windows appear active:

1. Set up function keys to perform the switching action for the work station user. For example, you might provide one key to page through data in the first window and another key to page through data in the second window.
2. Write the first window to the display.
3. Write the second window to the display using the USRRSTDSP keyword. The USRRSTDSP keyword keeps the first window from being saved when the second window is displayed. It keeps the second window from being removed when the user returns to the first window and then keeps the first window from being removed when the user returns to the second.

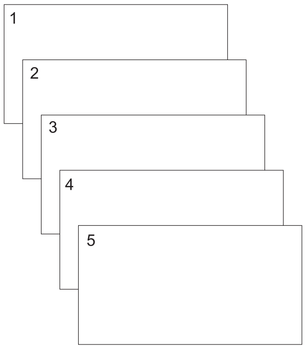
You can also use the USRRSTDSP keyword on the first window; it is not required on the first window because using it on the second window keeps the first window and all subsequent windows from being saved.

4. Be prepared to rebuild each window when the work station user presses the keys that perform the switching action. Once the two windows are displayed, you must rebuild each window whenever the user wants to move to it. The USRRSTDSP keyword keeps both windows from being saved and restored, so they must be rebuilt at every switch. In effect, the system does not know that the previous window existed on the display.

For more information about the USRRSTDSP keyword, see “Improving Application Performance” on page 126.

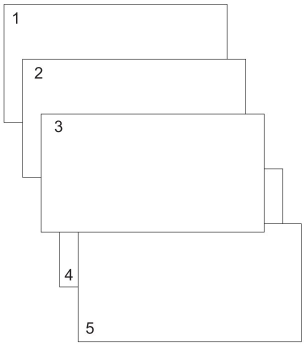
Making One Window in a Series Stand Out

Assume that you display a series of windows that looks like this:



RBAHG502-0

Now you want to return to Window 3 and make it appear to pop out of the series, so that the display looks like this:



RBAHG503-0

To make Window 3 stand out without removing Windows 4 and 5 from the display, take these steps:

1. Specify the USRRSTDSP keyword on Window 4 or any earlier window (that is, Windows 1 through 3). The USRRSTDSP keyword prevents the previous window and any subsequent windows from being saved.
2. Rebuild Window 3 so that it looks the way it did before Window 4 was first displayed. Because USRRSTDSP was used, the display was not saved when Window 4 was added to the display or at any later time. Thus, none of the windows are removed from the display, and Window 3 appears to pop out of the series.

For more information about the USRRSTDSP keyword, see “Improving Application Performance” on page 126.

Removing Windows

When a window is removed, it no longer exists on the display, and you can no longer write to or read from it. The window keywords provide you with several different ways to remove windows. Which method you use depends on which windows you want to remove and which operation you want to perform next. The different methods are described in the following sections.

Removing All Windows

Remove all the windows on a display in one of these ways:

1. Write to a nonwindow record. This allows you to remove all existing windows without displaying a new window.
2. Write to a window record that specifies the RMVWDW keyword. The RMVWDW keyword causes all other windows on the display to be removed when the specified window is displayed. If there are no other windows, the RMVWDW keyword is ignored, and no error is returned.

Removing More Recent Windows

To remove more recent windows that are overlaying the window you want to make active, read or write to the window you want to make active. Assume that Window 3 exists on the display but is not the active window. Windows 4, 5, and 6 were subject to more recent write operations than Window 3; therefore, they overlay Window 3. To remove Windows 4, 5, and 6 so that Window 3 is visible and is the active window, read or write to Window 3.

Improving Application Performance

In some cases, you can improve application performance by using the USRRSTDSP keyword to prevent the operating system from saving and restoring the display. The following sections describe how the system performs the operations, and how and when to use the USRRSTDSP keyword.

System Save and Restore Operations

If you do not use the USRRSTDSP keyword, the operating system automatically performs save and restore operations for your application. Before a window is displayed, the system saves the display, including any windows not being removed. When windows are removed, the system restores the display.

If a new window is being created on the display, the record that is active when the window is written is saved, and the entire display remains as background data. Then the new window becomes active. The saved record can be a window record or nonwindow record; the procedure is the same for both.

If a window record is written to the display to change or redisplay an existing window, any more recent windows are removed without being saved, the target window is restored, the new record is written, and the window becomes active.

If a nonwindow record is written to the display, any existing windows are removed without being saved, the new record is written to the initial display, and the display becomes active.

Response Time

The time needed for the system to perform save and restore operations depends on your communications setup and on the window being displayed.

The slowest response time occurs during the read and save operations performed when the first window is added to a display. Assume that the window is of average size and complexity. If the work station is attached to the system by a twinaxial, local area network (LAN), or other high-speed communications line, response time is quick. If the work station is attached by a 2400-baud dedicated line, it takes approximately 10 seconds to complete the read and save operations and then display the window. If the line speed is increased to 9600 baud, it usually takes about 2.5 seconds.

Other operations, such as saving the display before the second or third window is added, or restoring the display after windows are removed, take less time.

For more information, including details about other window sizes, terminal types, and line speeds, consult your marketing representative.

Bypassing System Save and Restore Operations

You can use the USRRSTDSP keyword to bypass system save and restore processing and instead have your application rebuild the display only when necessary. This technique can improve system performance and response time for the user of the application. Consider using it when you display only one window at a time and the windows are in a different display file, or when you display a series of windows in which the user will not return to earlier windows, or when you want more than one window to seem active at one time.

For example, under the following conditions, the system ordinarily performs two save operations:

- Your application displays only one window at a time.
- The display file is created with RSTDSP(*YES).
- The first window record to overlay the display is located in a separate file.

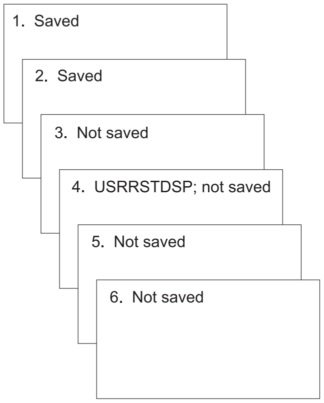
The first save operation is performed when the display file is suspended. The second save operation is performed because a window is being displayed. USRRSTDSP eliminates the second, unnecessary save operation.

To bypass system save and restore processing, take these steps:

1. Create your own procedure to rebuild the display after a window is removed. Be sure to include any data that the user enters and that must be redisplayed.
2. Specify the record-level USRRSTDSP keyword on the window following the first window you do not want the system to save. The USRRSTDSP keyword keeps the system from performing save and restore operations. The USRRSTDSP keyword is allowed only on records containing the WINDOW keyword; it is ignored on the window reference record.

   Once the USRRSTDSP keyword is specified, it remains in effect, even if the option indicator is set off, until you read or write to either the initial, windowless display or the window that is two windows before the window on which the USRRSTDSP keyword was specified.

   Assume that six windows are on the display and the USRRSTDSP keyword was specified on the fourth. To turn off USRRSTDSP and have the system resume saving the display, you must write to the second window. As shown in the diagram, the system has saved only the first two windows:



RBAHG504-0

USRRSTDSP Keyword Processing and Interactions

The USRRSTDSP keyword interacts with other keywords and window-related functions. Before using the keyword, you should understand the following (assume that the USRRSTDSP keyword is in effect):

- If a window record is written to a window that was saved (window 1 or 2 in the above example), the saved display is restored, the current record is written to the target window, and the target window becomes active. At this point, the USRRSTDSP keyword is no longer in effect.
- If a window definition record is written to a window that was not saved (window 3, 4, or 5 in the above example), it becomes a new window. It is merged with the previous display image and written to the display. No windows are removed.
- If a window record is read from a window that was not saved (window 3, 4, or 5 in the above example), an error message is returned to the application.
- If the initial display has been saved and the application writes to a window record specifying the RMVWDW keyword, any existing windows are removed. The new window is displayed on top of the initial display. The new window is active, and USRRSTDSP is no longer in effect.
- If the initial display is not saved and the application writes to a window record that specifies the RMVWDW keyword, all existing windows are removed. The new window is displayed on top of the initial display. The new window is active, and USRRSTDSP is still in effect.
- If a nonwindow record is written to the display and USRRSTDSP is specified on the first window, then the window is not removed, and the nonwindow record may overlay all or part of the window.

Programming Examples

The following sections illustrate the basic functions of the window keywords. The first example shows how to use a variety of window functions. It defines a full-screen display and several windows in one display file. The second example shows how to create windows for a full-screen display defined in a separate display file. The third example shows how to simulate menu bar support.

Using Basic Window Functions

The following scenario demonstrates the basic functions of the window keywords. The scenario is presented in three sections:

- The DDS used to define a full-screen display and windows
- The RPG program used to display the full-screen display and windows
- Discussion and illustration of the results

DDS Full-Screen Display and Window Definitions

The following DDS defines the initial display and two windows used in the scenario:

```
A*-------------------------------------------------------------------------*
A* DISPLAY FILE - DEMOFM
A*-------------------------------------------------------------------------*
A*-------------------------------------------------------------------------*
A* FILE LEVEL KEYWORDS
A*-------------------------------------------------------------------------*
A                                      DSPSIZ(24 80 *DS3)
A                                      HELP
A*
A*-------------------------------------------------------------------------*
A* RECORDS USED IN DEFINING INITIAL DISPLAY
A*-------------------------------------------------------------------------*
A          R INITIAL
A                                      CA03(03)
A                                      CA04(04)
A                                      CA06(06)
A                                      CA07(07)
A                                      CA12(12)
A                                      CLRL(*ALL)
A                                  3 28'NONWINDOW DISPLAY RECORD'
A                                  6  1'FLD #1:'
A            FLD48         30A  B  6  9
A                                  6 43'FLD #2:'
A            FLD49         15A  B  6 51
A                                 10  1'FLD #3:'
A            FLD50         30A  B 10  9
A                                 10 43'FLD #4:'
A            FLD51         15A  B 10 51
A                                 23  1'F3=Xxxx'
A                                 23 13'F4=Xxxxxxxx'
A                                 23 29'F6=Xxxxx Xxxxx'
A                                 23 48'F7=Xxxx'
A                                 23 59'F12=CANCEL'
A*
```

```
A*-------------------------------------------------------------------------*
A* RECORDS USED IN DEFINING WINDOW1
A*-------------------------------------------------------------------------*
A          R WINDOW1                   WINDOW(7 3 11 33)
A N01                                  WDWBORDER((*COLOR GRN))
A  01                                  WDWBORDER((*COLOR RED))
A                                  2 13'WINDOW #1'
A                                  4  2'CUSTOMER NO.'
A            FLD1           6A  O  4 15
A                                  4 22':'
A*
A          R REC2WIN1                  WINDOW(WINDOW1)
A                                      OVERLAY
A                                      CA12(12)
A                                 10  2'F12=CANCEL'
A*
A          R REC3WIN1                  WINDOW(WINDOW1)
A                                      OVERLAY
A                                  6  2'NAME:'
A            FLD2          24A  B  6  8
A                                  7  2'ADDRESS:'
A            FLD3          21A  B  7 11
A                                  8  2'PHONE:'
A            FLD4          23A  B  8  9
A*
A*-------------------------------------------------------------------------*
A* RECORDS USED IN DEFINING WINDOW2
A*-------------------------------------------------------------------------*
A          R WINDOW2                   WINDOW(9 25 11 32)
A*
A                                  2 12'WINDOW #2'
A            FLD5          22A  O  4  6
A            FLD6          25A  O  5  4
A            FLD7          25A  O  6  4
A            FLD8          25A  O  7  4
A*
A          R REC2WIN2                  WINDOW(WINDOW2)
A                                      OVERLAY
A                                      CA12(12)
A                                 10  8'Xxxxxxx :'
A            FLD9           6A  B 10 18
A*
A*-------------------------------------------------------------------------*
```

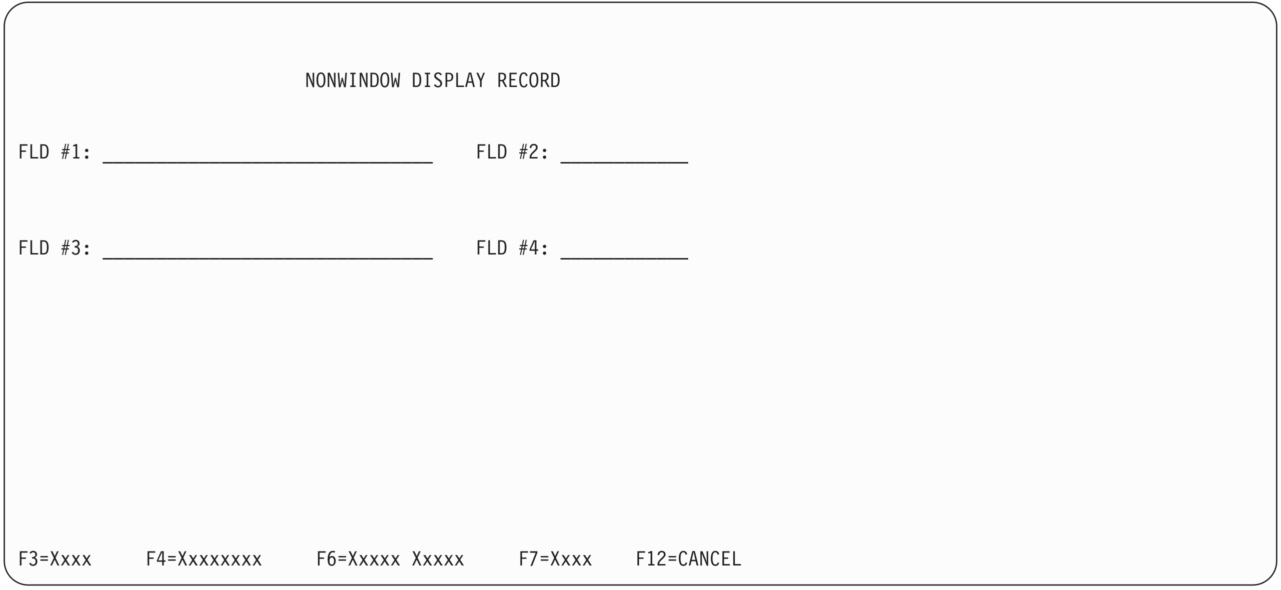
RPG Display Program

This RPG program displays the full-screen display and windows defined in the preceding section. Steps 1 through 5 are explained in the sections following the program.

```
F*****************************************************************
F*    RPG PROGRAM - WINDEMO
F*****************************************************************
FDEMOFM  CF  E                    WORKSTN
C*****************************************************************
C* Step 1:  Display Initial Display
C*****************************************************************
C                     EXFMTINITIAL
C*****************************************************************
C* Step 2:  Display Window #1
C*****************************************************************
C                     MOVE 'nnnnnn'  FLD1
C                     WRITEWINDOW1
C                     WRITEREC2WIN1
C                     EXFMTREC3WIN1
C*****************************************************************
C* Step 3:  Display Window #2
C*****************************************************************
C                     MOVEL'Xxxxxxx 'TEMP16 16
C                     MOVE 'xxxx x x'TEMP16
C                     MOVELTEMP16    FLD5
C                     MOVE 'xxxxxx'  FLD5
C*
C                     MOVEL'xxxxxx x'FLD6
C                     MOVEL'xx xxxx 'TEMP16
C                     MOVE 'xxx xxxx'TEMP16
C                     MOVELTEMP16    TEMP17 17
C                     MOVE ' '       TEMP17
C                     MOVE TEMP17    FLD6
C*
C                     MOVEL'xxxx xxx'FLD7
C                     MOVEL'xxxx x x'TEMP16
C                     MOVE 'x xxxx x'TEMP16
C                     MOVELTEMP16    TEMP17
C                     MOVE 'x'       TEMP17
C                     MOVE TEMP17    FLD7
C*
C                     MOVEL'xxxxxxx 'FLD8
C                     MOVEL'xxxx xxx'TEMP16
C                     MOVE 'xx xxxxx'TEMP16
C                     MOVELTEMP16    TEMP17
C                     MOVE '.'       TEMP17
C                     MOVE TEMP17    FLD8
C*
C                     WRITEWINDOW2
C                     EXFMTREC2WIN2


C*****************************************************************
C* Step 4:  Restore Window #1
C*****************************************************************
C                     EXFMTREC3WIN1
C*****************************************************************
C* Step 5:  Display Initial Display
C*****************************************************************
C                     READ INITIAL                  91
C*****************************************************************
C* End The RPG Program
C*****************************************************************
C                     SETON                     LR
C*****************************************************************
```

Step 1: Display Initial Display

The application creates the initial display without using a window keyword:

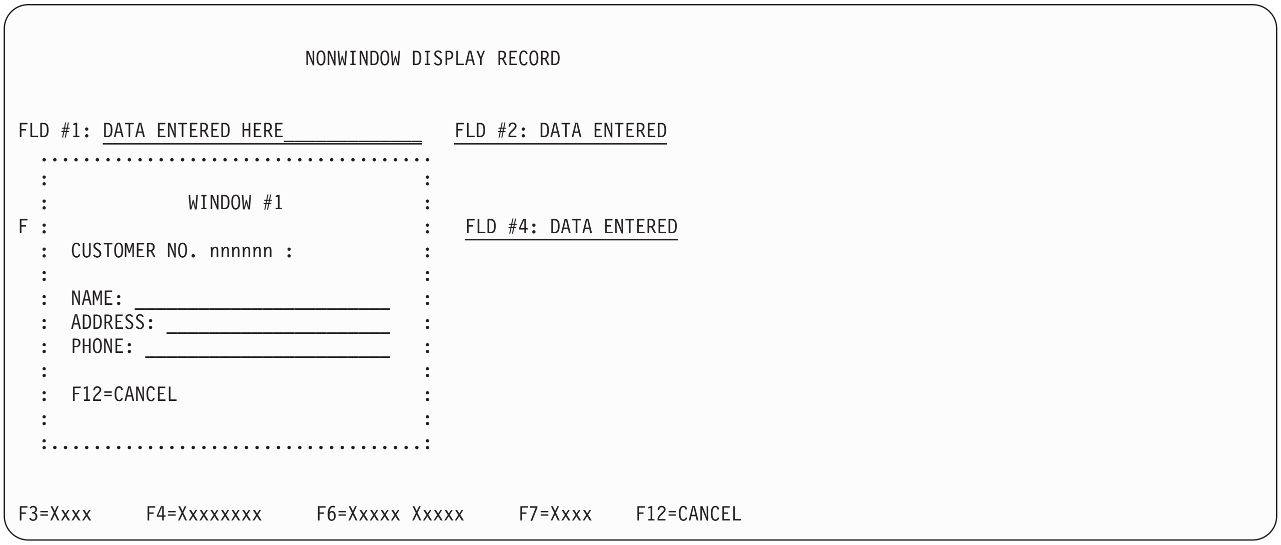


```
                          NONWINDOW DISPLAY RECORD


FLD #1: _______________________________    FLD #2: ____________



FLD #3: _______________________________    FLD #4: ____________











F3=Xxxx     F4=Xxxxxxxx     F6=Xxxxx Xxxxx     F7=Xxxx    F12=CANCEL
```

Step 2: Display Window 1

The user types some data on the display and presses the Enter key. The application writes to window definition record WINDOW1, which creates the window. Then the application adds information to the window by writing to window reference record REC2WIN1 and performing a write/read operation to window reference record REC3WIN1.

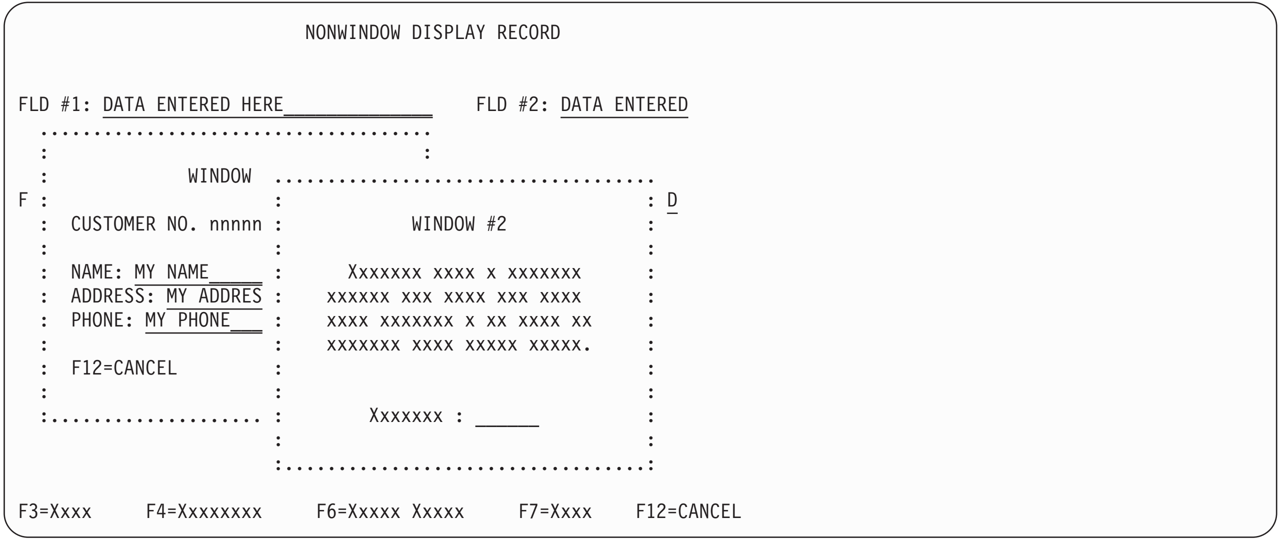
Before the window is displayed, the system performs a read screen immediate operation to obtain the display image and saves the underlying display. The system performs a read screen immediate operation only when the first window is added to the display. It performs a save operation each time a window is created.



```
                          NONWINDOW DISPLAY RECORD


FLD #1: DATA ENTERED HERE_____________   FLD #2: DATA ENTERED
  ....................................
  :                                  :
  :             WINDOW #1            :
F :                                  :   FLD #4: DATA ENTERED
  :  CUSTOMER NO. nnnnnn :           :
  :                                  :
  :  NAME: ________________________  :
  :  ADDRESS: _____________________  :
  :  PHONE: _______________________  :
  :                                  :
  :  F12=CANCEL                      :
  :                                  :
  :..................................:


F3=Xxxx     F4=Xxxxxxxx     F6=Xxxxx Xxxxx     F7=Xxxx    F12=CANCEL
```

Step 3: Display Window 2

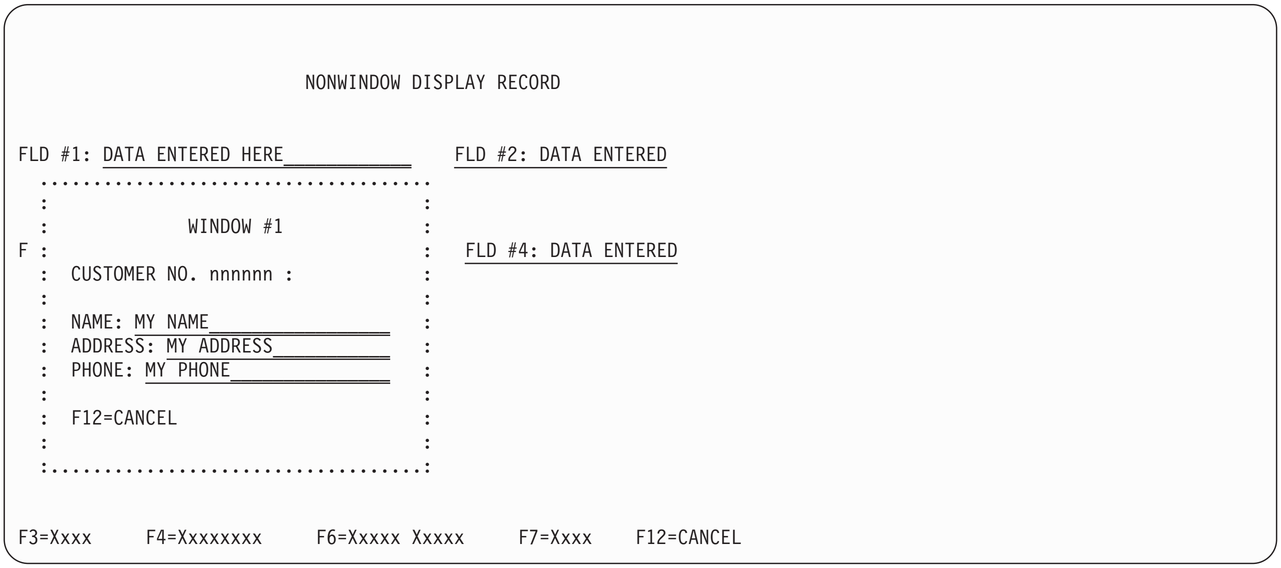
The user types some information and presses the Enter key. The application writes to window definition record WINDOW2. WINDOW2 is not the active window, and it is not currently on the display; therefore, the system saves the underlying display, associating the saved data with WINDOW1. Then a new window is created. The application adds information to the window by performing a write/read operation to window record REC2WIN2.



```
                            NONWINDOW DISPLAY RECORD


FLD #1: DATA ENTERED HERE______________    FLD #2: DATA ENTERED
  ....................................
  :                                  :
  :            WINDOW  ....................................
F :                    :                                  : D
  :  CUSTOMER NO. nnnnn :            WINDOW #2            :
  :                    :                                  :
  :  NAME: MY NAME_____ :      Xxxxxxx xxxx x xxxxxxx     :
  :  ADDRESS: MY ADDRES :    xxxxxx xxx xxxx xxx xxxx     :
  :  PHONE: MY PHONE___ :    xxxx xxxxxxx x xx xxxx xx    :
  :                    :    xxxxxxx xxxx xxxxx xxxxx.     :
  :  F12=CANCEL        :                                  :
  :                    :                                  :
  :................... :        Xxxxxxx : ______          :
                       :                                  :
                       :..................................:

F3=Xxxx     F4=Xxxxxxxx     F6=Xxxxx Xxxxx     F7=Xxxx    F12=CANCEL
```

Step 4: Restore Window 1

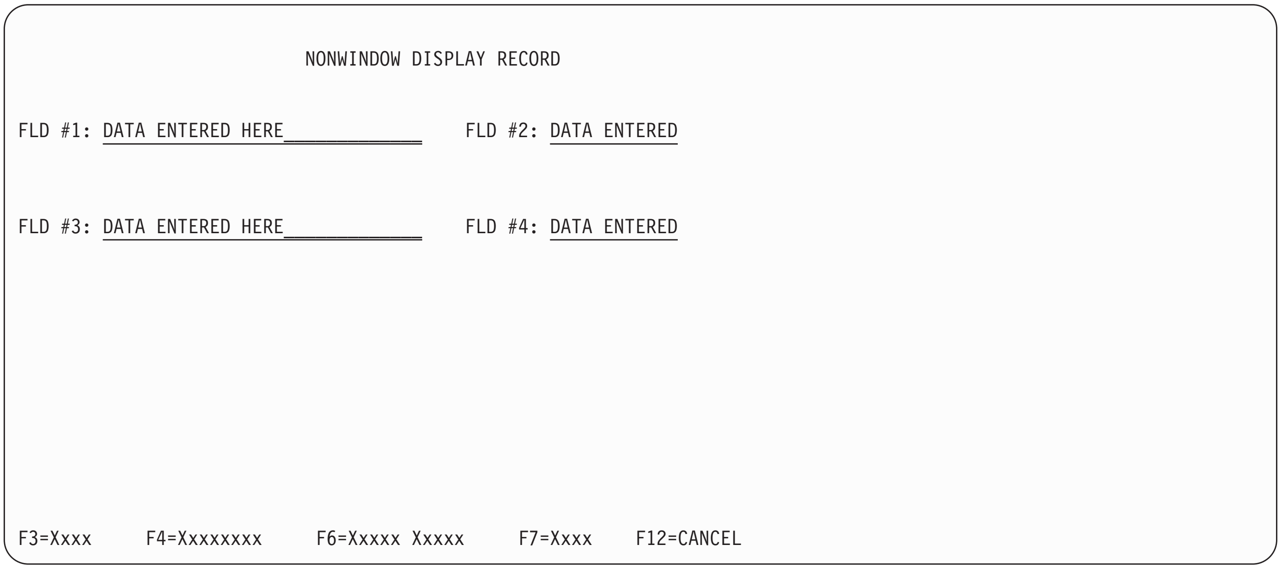
The user types some data and presses the Enter key. The application performs a write/read operation to window record REC3WIN1. The record format name specified on the WINDOW keyword, WINDOW1, is not the active window. However, the window is currently on the display; therefore, the system restores the saved display associated with WINDOW1. The restore operation removes WINDOW2, which was written after WINDOW1. Then REC3WIN1 is written to the restored window.



```


                            NONWINDOW DISPLAY RECORD


FLD #1: DATA ENTERED HERE____________    FLD #2: DATA ENTERED
  ....................................
  :                                  :
  :            WINDOW #1             :
F :                                  :  FLD #4: DATA ENTERED
  :  CUSTOMER NO. nnnnnn :           :
  :                                  :
  :  NAME: MY NAME_________________  :
  :  ADDRESS: MY ADDRESS___________  :
  :  PHONE: MY PHONE_______________  :
  :                                  :
  :  F12=CANCEL                      :
  :                                  :
  :..................................:


F3=Xxxx     F4=Xxxxxxxx     F6=Xxxxx Xxxxx     F7=Xxxx    F12=CANCEL
```

Step 5: Display Initial Display

The user presses the Enter key. The application performs a read operation to the initial display, which automatically removes the last window from the display.



```
                           NONWINDOW DISPLAY RECORD


FLD #1: DATA ENTERED HERE_____________    FLD #2: DATA ENTERED



FLD #3: DATA ENTERED HERE_____________    FLD #4: DATA ENTERED










F3=Xxxx     F4=Xxxxxxxx     F6=Xxxxx Xxxxx     F7=Xxxx    F12=CANCEL
```

Defining Windows in a Separate Display File

The following sections show the DDS code for a full-screen display and window and the procedural steps needed to use them. In contrast to the preceding example, this example keeps the window records in a separate display file from the file for the underlying display. This technique allows you to add windows for items, such as help, to existing applications, without rewriting the display-file code for the applications.

In the example, the application uses RSTDSP(*NO) to indicate that a save operation should not be done when a file is suspended. Because displaying a window also performs a save operation, using RSTDSP(*NO) prevents two save operations from being performed. Because removing the window restores the screen that was present prior to the window operation, the application is not required to rebuild the display after window processing. For more information on the USRRSTDSP keyword, see the notes at the end of the example.

DDS Full-Screen Display and Window Definitions

```
A*-----------------------------------------------------------------------*
A* DISPLAY FILE DISPLAY1 (RSTDSP=NO DFRWRT=*YES)
A*-----------------------------------------------------------------------*
A          R REC1
A                                   2 21'FIRST RECORD IN FILE'
A                                   4 17'Current Customer #:'
A            FIELD1         6A  B  4 38
A*
A          R REC2                      OVERLAY CA03(03)
A                                   6 21'SECOND RECORD IN FILE'
A                                   8 17'Current Customer #:'
A            FIELD2         6A  B  8 39DSPATR(HI)
A                                  24 02'CA03=EXIT'
A*-----------------------------------------------------------------------*
```

RPG Program Source

```
FDISPLAY1CF E
C                      WRITEREC1
C                      WRITEREC2
C          RETRY       TAG
C                      READ REC2                          90
C   03                 GOTO END
```

```
C                      CALL 'WINPGM'
C                      GOTO RETRY
C          END         TAG
C                      SETON                     LR
```

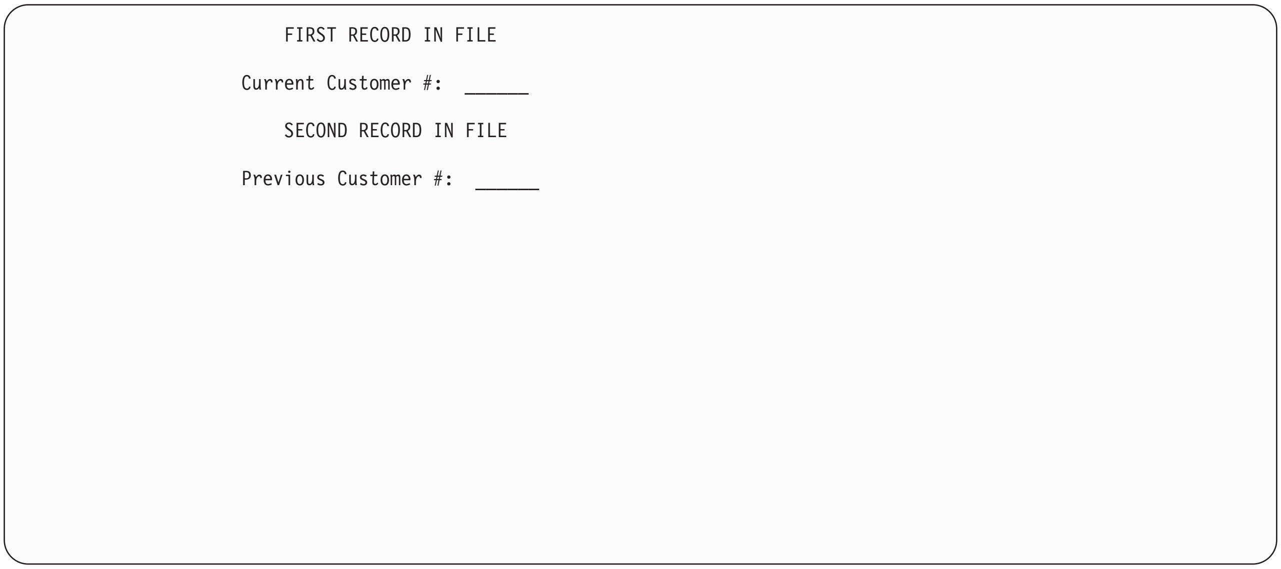
```
A*-----------------------------------------------------------------------*
A* DISPLAY FILE DISPLAY2 (RSTDSP=NO)
A*-----------------------------------------------------------------------*
A          R WINDOW1                   WINDOW(7 4 11 25)
A N01                                  WDWBORDER((*COLOR GRN))
A  01                                  WDWBORDER((*COLOR RED))
A                                  2  9'Window #1'
A *
A          R REC2WIN1                  WINDOW(WINDOW1)
A                                      CA12(12) OVERLAY
A                                  4  1'Customer No. nnnnnn:'
A                                  6  1'Name:'
A            FIELD3        19A  B  6  7
A                                  7  1'Address:'
A            FIELD4        16A  B  7 10
A                                  8  1'Phone:'
A            FIELD5        18A  B  8  8
A                                 10 1'F12=Cancel'
A*-----------------------------------------------------------------------*
A* Dummy record to remove window from display before returning
A*-----------------------------------------------------------------------*
A          R RMVWDW                    CLRL(*NO) OVERLAY FRCDTA
A*-----------------------------------------------------------------------*
A* No I/O will ever be done to this record.  This record prevents the
A* display from clearing.
A*-----------------------------------------------------------------------*
A          R DUMMY                     ASSUME
A                                 11  1' '
A*-----------------------------------------------------------------------*
```

RPG Program Source for WINPGM

```
FDISPLAY1CF E
C                      WRITEWINDOW1
C                      EXFMTREC3WIN1
C                      WRITERMVWDW
C                      RETRN
C                      SETON                    LR
```

Step 1: Display Initial Display

The application opens display file DISPLAY1, performs a write operation to record REC1, and performs a write/read operation to REC2.



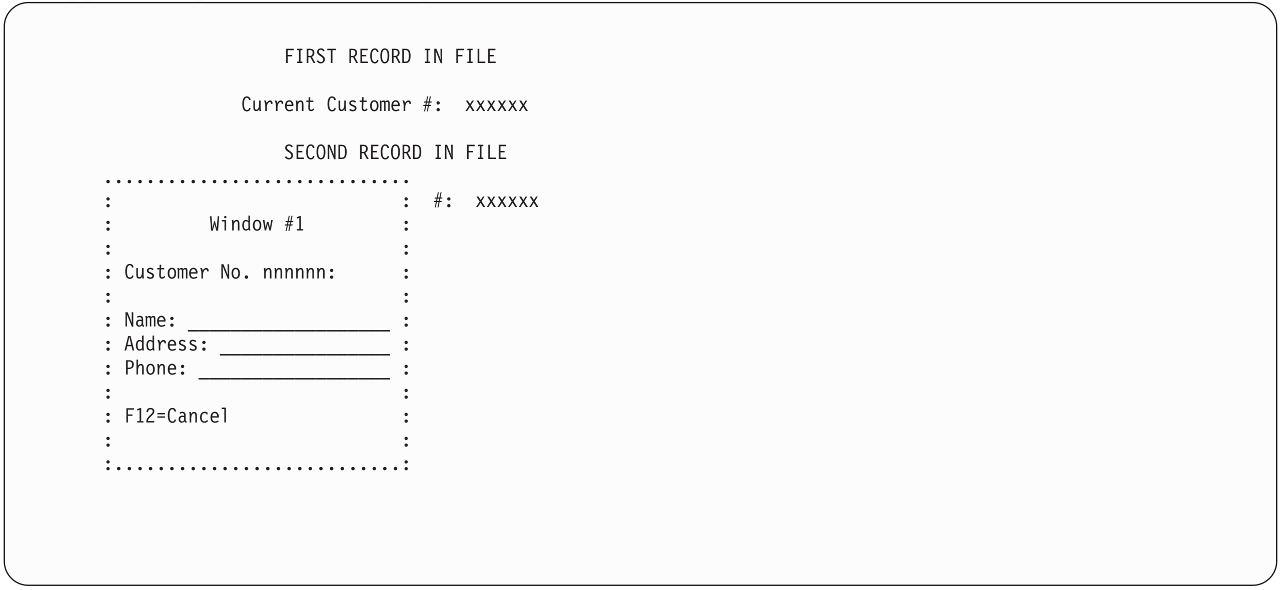
```
                         FIRST RECORD IN FILE

                     Current Customer #:  ______

                         SECOND RECORD IN FILE

                     Previous Customer #:  ______
```

Step 2: Display a Window

The user enters data indicating that a window should be displayed. The application opens window display file DISPLAY2. This can be done from a separate program such as the one which displayed the screen in Step 1. The ASSUME keyword on record DUMMY keeps the full-screen display from being cleared, and the system marks display file DISPLAY2 as suspended. No I/O operation ever needs to be performed to record DUMMY; it only needs to be present in the file.

The application performs a write operation to record WINDOW1. Display file DISPLAY1 is suspended; because of the RSTDSP(*NO) setting, no save is performed. Display file DISPLAY2 is restored; because of the RSTDSP(*NO) setting, no restore data is sent.

The application performs a write operation to record REC2WIN1.



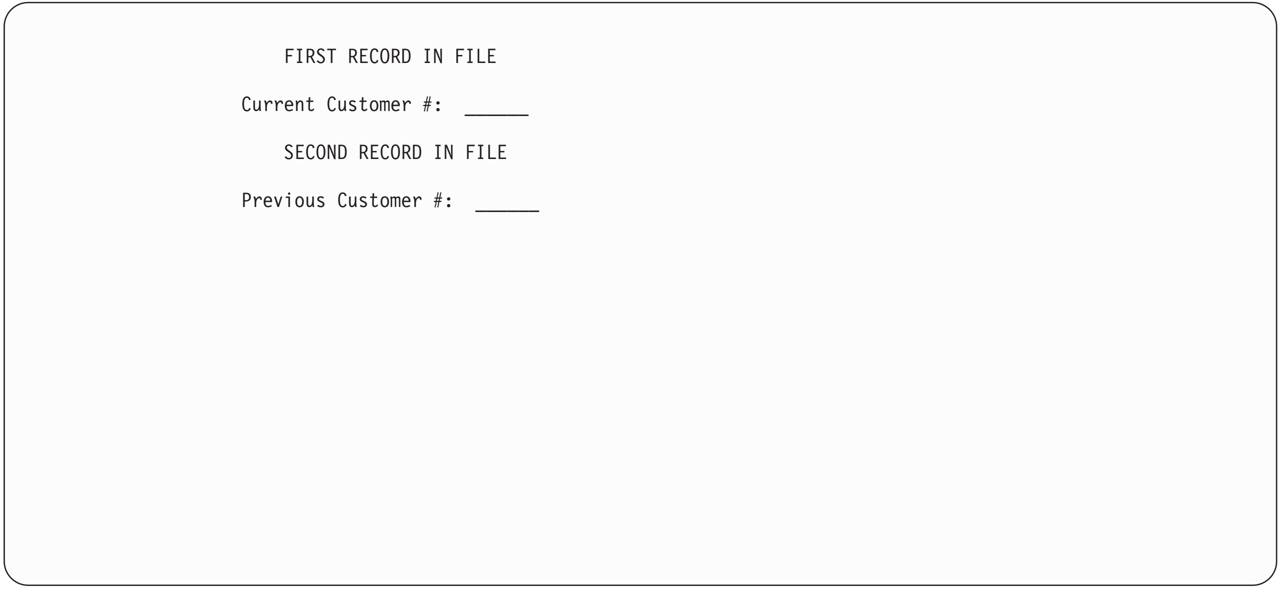
```
                         FIRST RECORD IN FILE

                     Current Customer #:  xxxxxx

                         SECOND RECORD IN FILE
           .............................
           :                           :  #:  xxxxxx
           :         Window #1         :
           :                           :
           : Customer No. nnnnnn:      :
           :                           :
           : Name: ___________________ :
           : Address: ________________ :
           : Phone: __________________ :
           :                           :
           : F12=Cancel                :
           :                           :
           :...........................:
```

Step 3: Return to the Initial Display

The user enters data indicating that the application should return to the initial display file, DISPLAY1. The application performs a write operation to record RMVWDW, which causes the system to remove all windows from the display. Because the application does not close the DISPLAY2 display file, and the USRRSTDSP keyword is not specified, then removing the window restores the initial display. The application is not required to rebuild the initial display. The RMVWDW record should contain the FRCDTA keyword or specify DFRWRT(*NO) when the display file is created.

The application performs a write operation to record REC1. Display file DISPLAY2 is suspended; because of the RSTDSP(*NO) setting, no save operation is performed. Display file DISPLAY1 is restored; because of the RSTDSP(*NO) setting, no data is sent.

The application performs a write/read operation to record REC2.



```
    FIRST RECORD IN FILE

Current Customer #:  ______

    SECOND RECORD IN FILE

Previous Customer #:  ______
```

Additional notes on this example:

1. If the application closes DISPLAY2 in Step 3, then the application must rebuild the initial display by performing a write operation to REC1 and then a write/read operation to REC2. This can be avoided by specifying RSTDSP(*YES) for DISPLAY1.
2. USRRSTDSP can be added to the window record in DISPLAY2. However, if this is done, the user must either specify RSTDSP(*YES) for DISPLAY1, or rebuild the initial display in Step 3. However, in Step 3, the application must still write RMVWDW, or the borders of the window are not displayed properly on the next write operation to the window.
3. The KEEP keyword should be added to the window format both of the following conditions are true:
   - The application closes DISPLAY2 in Step 3.
   - No I/O is done to file DISPLAY1 prior to opening DISPLAY2 and displaying the window again.

Chapter 6. Creating a Graphical Look for Displays

The graphical look is a change in what you see while you run DDS functions. Instead of dotted windows, you have crisp window borders. Instead of typing numbers in an option column to make a selection, you can use a mouse or mnemonics. A **mnemonic** is an underlined character within the text of a choice that you can type to select the choice. Instead of seeing option numbers, you can see radio buttons or check boxes. A **radio button** is a circle that precedes a choice in a single-choice selection field on a graphical display station. A **check box** is a square box that precedes a choice in a multiple-choice selection field on a graphical display station. You can click on radio buttons and check boxes to make choices.

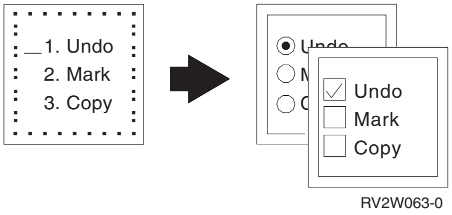


Figure 34. Radio Buttons and Check Boxes

In addition to a fresher look, the enhanced function includes menu bars and pull-down menus. Instead of having to simulate a menu bar by using lengthy DDS coding, you can use DDS keywords.

Factors Affecting the Graphical Look

The graphical functions described in this chapter appear differently, depending on the hardware configuration you have and the value you specify on the enhanced display (ENHDSP) parameter.

Hardware Configuration

Table 15 and Table 16 on page 140 show how each graphical function appears on different configurations. Letters A through F in the tables identify the configurations; these letters are referred to throughout this chapter.

Table 15. Functions Supported by Hardware Configurations A, B, and C

	Hardware Configuration		
	A	**B**	**C**
Function	**Graphical User Interface (GUI) Programmable Work Stations**[1]	**InfoWindow II Display Station**[2] **Attached to Controller Supporting Enhanced Interface**[3]	**3477 Display Station Attached to Controller Supporting Enhanced Interface**[3]
Windows[9]	GUI[4] windows and possible improvement in performance	Character-based GUI[5] and possible improvement in performance	Character windows and possible improvement in performance
Selection fields and menu bars	GUI[4]	Character-based GUI[5]	Mnemonics, bar selection cursor
Selection lists	Bar selection cursor. Possible check boxes for multiple-choice lists. Possible radio buttons for single-choice lists.	Bar selection cursor. Possible check boxes for multiple-choice lists. Possible radio buttons for single-choice lists.	Bar selection cursor. Input field to the left of list.
Continued-entry fields	One field	One field	One field

Table 15. Functions Supported by Hardware Configurations A, B, and C (continued)

	Hardware Configuration		
	A	B	C
Function	Graphical User Interface (GUI) Programmable Work Stations[1]	InfoWindow II Display Station[2] Attached to Controller Supporting Enhanced Interface[3]	3477 Display Station Attached to Controller Supporting Enhanced Interface[3]
Edit masks	Yes	Yes	Yes
Highlighting	Yes	Yes	Yes
Cursor progression	Yes	Yes	Yes
Word spill	Yes[8]	Yes	Yes
Simple hotspots	Yes	Yes	No
Scroll bars	GUI [4] scroll bars	Character-based GUI [5] scroll bars	*MORE, *PLUS, or character scroll bars [6]
Push buttons	Yes	Yes	Yes
Auto-Selection	Yes	Yes	Yes
Auto-Enter	Yes	Yes	Yes
Programmable Mouse Buttons	Yes[8]	Yes	No
Grid Lines[7]	No	No	No

Notes:

1. For example, RUMBA/400 (Microsoft® Windows® and OS/2®) and AIX® AS/400 Connection Program/6000 Release 2.
2. InfoWindow II display stations: 3486, 3487, 3488.
3. Twinaxial controllers: 5494 Release 1.1, and features 6050, 2661, 9146, and 9148.
4. GUI includes solid-line window borders, selectable background colors, use of a pointer device (for example, a mouse), mnemonic selection, bar selection cursor, radio buttons (for single-choice selection fields), and check boxes (for multiple-choice selection fields).
5. Character-based GUI is similar to GUI except that in character-based GUI, constructs are created using characters, and background colors are not selectable.
6. Scroll bars that appear on display stations without pointing devices are for display purposes only.
7. Grid lines are supported only on DBCS display stations. For the specific hardware required for grid lines, see "Hardware Requirements for Grid Line Structures" on page 207.
8. RUMBA/400 does not currently support this function.
9. RUMBA/400 does not currently support window footers.

Table 16. Functions Supported by Hardware Configurations D, E, and F

	Hardware Configuration		
	D	E	F
Function	5250 Display Station Attached to Controller Supporting Enhanced Interface[1]	ASCII Display Station Attached to ASCII Controller Supporting Enhanced Interface[2]	Any Display Station Attached to Controller Not Supporting Enhanced Interface[3]
Windows[7]	Character windows and possible improvement in performance	Character windows and possible improvement in performance	Character windows
Selection fields and menu bars	Bar selection cursor	Bar selection cursor	Entry field driven

Table 16. Functions Supported by Hardware Configurations D, E, and F (continued)

	Hardware Configuration		
	D	**E**	**F**
Function	**5250 Display Station Attached to Controller Supporting Enhanced Interface**[1]	**ASCII Display Station Attached to ASCII Controller Supporting Enhanced Interface**[2]	**Any Display Station Attached to Controller Not Supporting Enhanced Interface**[3]
Selection lists	Bar selection cursor. Input field to the left of list.	Bar selection cursor. Input field to the left of list.	Input field to the left of list.
Continued-entry fields	One field	One field	Multiple fields
Edit masks	Yes	Yes	Ignored
Highlighting	Yes	Yes	Ignored
Cursor progression	Yes	Yes	Ignored
Word spill[6]	Yes	Yes	Ignored
Simple hotspots	No	No	No
Scroll bars	*MORE, *PLUS, or character scroll bars [4]	*MORE, *PLUS, or character scroll bars [4]	*MORE, *PLUS, or character scroll bars [4]
Push buttons	Yes	Yes	Yes
Auto-Selection	Yes	Yes	No
Auto-Enter	Yes	Yes	Yes
Programmable Mouse Buttons	No	No	No
Grid Lines[5]	No	No	No

Notes:

1. Twinaxial controllers: 5494 Release 1.1, and features 6050, 2661, 9146, and 9148.
2. ASCII controllers that support an enhanced interface: features 6041, 6141, 2637, 9145, 9147.
3. For example: 5250 display stations attached to 5294 and 5394 controllers or features 2638, 6040, and 6140; some programmable work stations emulating a controller with an attached 5250 display station (for example, iSeries Access for Windows).
4. Scroll bars that appear on display stations without pointing devices are for display purposes only.
5. Grid lines are supported only on DBCS display stations. For the specific hardware required for grid lines, see "Hardware Requirements for Grid Line Structures" on page 207.
6. RUMBA/400 does not currently support this function.
7. RUMBA/400 does not currently support window footers.

Enhanced Display Parameter

The enhanced display (ENHDSP) parameter can be used with the CRTDSPF and CHGDSPF commands. Use this parameter to specify whether the data being shown at a display station uses the enhanced capabilities available on the display station.

Normally, DDS windows and CUA® graphical items are rendered using whatever enhanced capabilities are available on the display station. For example, window borders and menu-bar separators are presented graphically on a graphical display station.

You can use ENHDSP(*NO) to cause all records defined in the display file to be displayed in character-based mode, regardless of the capabilities of the display station. When ENHDSP(*NO) is specified, none of the enhanced capabilities that may be available on a particular display station are used. That is, records display just as they would on a display station in configuration F in Table 16 on page 140.

The default value for ENHDSP is *YES. Any enhanced capabilities of the display station are taken advantage of automatically. If you specify ENHDSP(*YES) and you use the default window border and menu-bar separator, the window border and the menu-bar separator appear as solid lines. If the display station is attached to a controller that does not support an enhanced interface for nonprogrammable work stations, ENHDSP(*YES) is ignored. The records in the display file are displayed on that display station in character-based mode (as if ENHDSP(*NO) were specified).

Writing records from files with ENHDSP(*YES) and files with ENHDSP(*NO) to the same display. If the record (or records) displayed is from a file with ENHDSP(*YES), the first write operation of a record (such as a window) from a file with ENHDSP(*NO) causes all menu bars, pull-down menus, and other windows on the display to change from graphical to character-based. The records from the file with ENHDSP(*YES) are switched to the ENHDSP(*NO) mode of display.

If the record displayed is from a file with ENHDSP(*NO), a write operation of a record (such as a window) from a file with ENHDSP(*YES) does not change the presentation of any menu bars, pull-down menus, or other windows on the display.

Notes:

1. The system file that is used for UIM help is shipped with ENHDSP(*NO). If you use UIM help with a file that has ENHDSP(*YES) specified, the display will changes from graphical to character-based.
2. Some programmable work stations that support an enhanced interface ignore the window border and menu-bar separator keywords.
3. If a window is written to the display station such that a border is in column 1, column 80 (for display size 24 by 80), or column 132 (for display size 27 by 132), the window is always displayed as though ENHDSP(*NO) were specified.

DDS Keywords

The tasks in this chapter refer to the DDS keywords, but may not provide all the details about them. For more information on each keyword, refer to the **DDS** topic in the iSeries Information Center.

CCSID (Coded Character Set Identifier)
Specifies that a “G” type field supports UCS-2 Level 1 data instead of DBCS-graphical data.

CHCACCEL (Choice Accelerator Text)
Specifies the text for the accelerator key on a single-choice selection field in a pull-down record.

CHCAVAIL (Choice Color/Display Attribute when Available)
Specifies the color or display attributes to be used when displaying the available choices in a menu bar or selection field.

CHCCTL (Choice Control)
Controls the availability of the choices for the field.

CHCSLT (Choice Color/Display Attribute when Selected)
Specifies the color or display attributes to be used when displaying a selected choice in a menu bar.

CHCUNAVAIL (Choice Color/Display Attribute when Unavailable)
Specifies the color or display attributes to be used when displaying the unavailable choices in a selection field.

CHOICE (Selection Field Choice)
Defines a choice for a selection field.

CNTFLD (Continued-Entry Field)
Defines a field as a continued-entry field. Continued-entry fields are sets of associated entry fields that are treated by the work station controller as a single-entry field during field data entry and editing.

EDTMSK (Edit Mask)
Defines an edit mask for fields with EDTCDE or EDTWRD keywords.

ENTFLDATR (Entry-Field Attribute)
Defines the leading attribute of the field that changes to a specified attribute whenever the cursor enters the field. When defined at both the field- and record-level, the field-level specification is used for the field.

FLDCSRPRG (Cursor Progression Field)
Defines the next field that the cursor moves to when exiting this field.

GRDATR (Grid Line Attribute)
Defines the color and line-type attributes for grid line structures in the file or record.

GRDBOX (Grid Box)
Defines the shape, positioning, and attributes of a box.

GRDCLR (Grid Clear)
Defines a rectangular area on a display within which all grid structures are erased.

GRDLIN (Grid Line)
Defines the shape, positioning, and attributes of a grid line.

GRDRCD (Grid Record)
Specifies that this record defines grid structures. No other display fields are allowed on records with this keyword.

HLPID (Help Identifier)
Specifies an identifier for the constant in the help for a field.

HTML (Hyper Text Markup Language)
Specifies if a data stream is sent to an i5/OS 5250 Workstation Gateway display, the HTML tags are sent along with the data stream. These HTML tags are processed on the HTML browser. This allows you to update applications to use on the Internet through the World Wide Web.

MLTCHCFLD (Multiple-Choice Selection Field)
Defines a field as a multiple-choice selection field. A multiple-choice selection field is a field that contains a fixed number of choices from which a user can select multiple choices.

MNUBAR (Menu Bar)
Defines a menu bar. A menu bar is a horizontal list of choices that is followed by a separator line.

MNUBARCHC (Menu-Bar Choice)
Defines a choice for a menu-bar field. A menu-bar choice represents a group of related actions that the application user can select.

MNUBARDSP (Menu-Bar Display)
Displays the menu bar.

MNUBARSEP (Menu-Bar Separator)
Specifies the color, display attributes, or character used to form the menu-bar separator line.

MNUBARSW (Menu-Bar Switch Key)
Assigns a CA key to the Switch-to-menu-bar key.

MNUCNL (Menu Cancel Key)
Assigns a CA key to be the cancel key for menu bars or pull-down menus.

MOUBTN (Programmable Mouse Button)
Allows an attention indicator (AID) to be associated with various pointer device events.

PSHBTNCHC (Push Button Choice)
Defines a push button within a push button field.

PSHBTNFLD (Push Button Field)

Defines a field as a push button field. A push button field is a field that contains a fixed number of push buttons. A **push button** is a button, labeled with text, graphics, or both that represents an action that starts when a user selects the push button.

PULLDOWN (Pull-Down Menu)

Defines a record as a pull-down menu for a menu bar.

SFLCSRPRG (Subfile Cursor Progression)

Causes the cursor to move to the same input field in the next subfile record when exiting this field.

SFLCHCCTL (Subfile Choice Control)

Controls the availability of the choices in a selection list.

SFLEND (Subfile End)

Displays a plus sign (+) or text (More... or Bottom) in the lower right location of the subfile. It can also display a scroll bar.

SFLMLTCHC (Subfile Multiple-Choice Selection List)

Defines a subfile as a multiple-choice selection list. A **multiple-choice selection list** is a potentially scrollable list from which the user can select one or more items.

SFLRCDNBR (Subfile Record Number)

Displays the page of the subfile containing the record whose relative record number is in this field.

SFLRTNSEL (Subfile Return Selected Choice)

Returns all selected choices in a selection list using the get-next-changed operation.

SFLSCROLL (Subfile Scroll)

Returns the relative record number of the subfile record that is at the top of the subfile when control is given to the application.

SFLSIZ (Subfile Size)

Specifies the number of records in the subfile.

SFLSNGCHC (Subfile Single-Choice Selection List)

Defines a subfile as a single-choice selection list. A **single-choice selection list** is a potentially scrollable list from which the user can select one item.

SNGCHCFLD (Single-Choice Selection Field)

Defines a field as a single-choice selection field. A single-choice selection field is a field that contains a fixed number of choices from which a user can select one choice.

Creating Menu Bars

Figure 35 shows an example of a **menu bar**, which is a horizontal list of choices that appears at the top of a display. An optional **menu-bar separator** appears below the list. When you select a choice from the menu bar, a pull-down menu appears. A **pull-down menu** is a group of actions associated with a menu-bar choice.

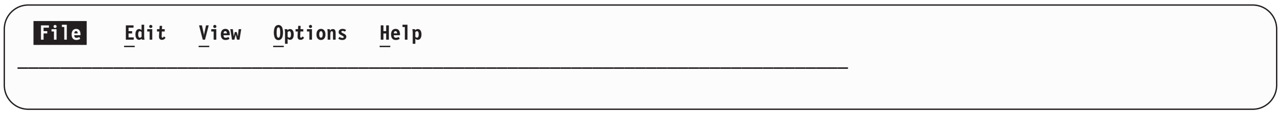


Figure 35. Example of a Menu Bar

Defining the Menu-Bar Choices

A menu bar is a special type of record containing a MNUBAR keyword and one menu-bar field. The **menu-bar field** is a numeric field containing one or more MNUBARCHC keywords. The MNUBARCHC keywords define the menu-bar choices and the pull-down menus associated with each choice. The menu bar always displays on the first line. The menu-bar record cannot contain any displayable fields other than the menu-bar field.

The number of rows occupied by the menu bar is determined by the number of choices, the maximum lengths of the choice text for all the menu-bar choices, and whether a menu-bar separator is specified. If you specify a menu-bar separator (the default), the number of rows the menu bar occupies (including the menu-bar separator) must be less than or equal to 12. If you do not specify a menu-bar separator, the number of rows the menu bar occupies must be less than or equal to 11. It is not possible to extend the range of menu-bar choices to line 12. However, you can use line 12 for another record.

If you define a pull-down record that is too large to fit beneath the maximum number of rows occupied by the menu bar, the file is not created.

Suppressing the Menu-Bar Separator

The default is for a menu-bar separator to display. To suppress the menu-bar separator, specify MNUBAR(*NOSEPARATOR). If you suppress the menu-bar separator, you cannot specify the menu-bar separator (MNUBARSEP) keyword.

Defining the Menu-Bar Separator

You can use the system defaults for the menu-bar separator or you can use the MNUBARSEP keyword. Using MNUBARSEP, you can specify the color and display attributes of the menu-bar separator and the character that makes up the separator. The default presentation of the menu-bar separator is a solid line on display stations in configurations A and B in Table 15 on page 139. The default character that makes up the menu-bar separator is an underline (_) on display stations in configuration C in Table 15 on page 139, and configurations D, E, and F in Table 16 on page 140. For an example of using the MNUBARSEP keyword, see Figure 36 on page 146.

Note: The menu-bar separator is used as the top border of the pull-down menus. Its color does not change when a pull-down menu is displayed. To ensure a consistent appearance for your displays, use the same color and display attributes for both the menu-bar separator and the pull-down menu borders.

Figure 36 on page 146 shows an example of the DDS for a menu bar.

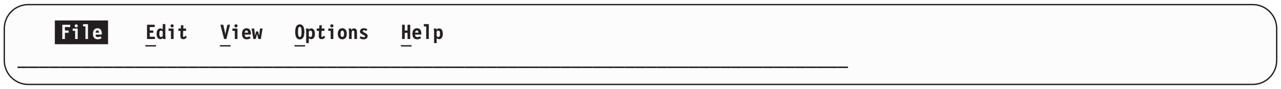
```
|...+....1....+....2....+....3....+....4....+....5....+....6....+....7...
     A          R MENUBAR                   MNUBAR
     A            MNUFLD         2Y 0B  1  2
     A                                      MNUBARCHC(1 PULLFILE +
     A                                                '>File         ')
     A  02                                  MNUBARCHC(2 PULLEDIT +
     A                                                &EDITTXT)
     A                                      MNUBARCHC(3 PULLVIEW +
     A                                                '>View         ')
     A  04                                  MNUBARCHC(4 PULLOPT +
     A                                                '>Options       ' +
     A                                                &RTNFLD)
     A                                      MNUBARCHC(5 PULLHELP +
     A                                                '>Help           ')
     A                                      MNUBARSEP((*COLOR WHT))
     A            EDITTXT       20A  P
     A            RTNFLD         2Y 0H
     A                      .
     A                      .
     A                      .
```

Figure 36. DDS for a Menu Bar. Assume that field EDITTXT contains the text `>Edit`.

You can control which menu-bar choices are displayed by specifying option indicators on the MNUBARCHC keywords. Option indicators are used by the application program to specify if a menu-bar choice should be displayed (optioned on) or should not be displayed (optioned off). The DDS for option indicators are shown in Figure 36 on page 146. The application specifies the option indicators as on or off and then writes the menu-bar record (without MNUBARDSP in effect) to send the option indicators to the system. If a menu-bar choice is optioned off, the list of choices is compressed. However, the number of rows occupied by the menu-bar record is not compressed (because records cannot be variable length). The number of rows occupied by the menu bar is the number of rows needed if all the choices were displayed plus one row for the menu bar separator. If, through optioning, the list of choices is compressed so that it is displayed using fewer rows, the separator line is displayed on the line following the last row of choices. There are blank lines between the menu-bar separator and the next record on the display.

The text which appears for each choice in a menu-bar comes from either the program-to-system fields named or the text specified for the choice text parameter of the MNUBARCHC keyword. The number of rows calculated by the system for the menu-bar record depends on the size of each program-to-system field or length of choice text. In addition, three spaces are assumed between each choice. Any trailing blanks in the choice text are removed; the remaining length is used for the calculation. The length of any program-to-system field is used as is, because trailing blanks can not be anticipated. However, when the menu-bar record is displayed, any trailing blanks are removed. Therefore, the number of rows actually occupied could be less than the number calculated. When that occurs, blank lines appear between the menu-bar separator and the next record on the display.

On display stations in configurations A and B from Table 15 on page 139, the menu bar looks like this:

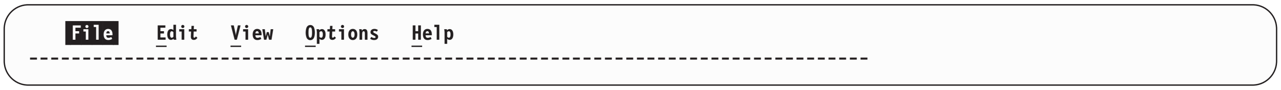


```
   File   Edit   View   Options   Help
```

Figure 37. Menu Bar on a Graphical Display Station with Enhanced Interface

On display stations in configuration C from Table 15 on page 139, the menu bar looks like this:

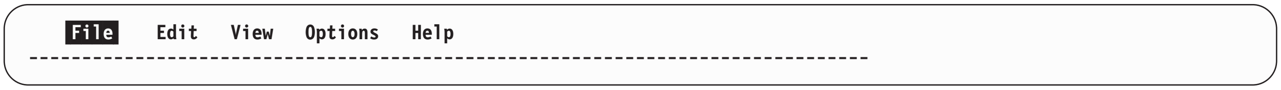
Note: In Figure 38 on page 147 the first character of each menu bar choice is underlined.



```
   File     Edit   View   Options   Help
-------------------------------------------------------------------------------
```

Figure 38. Menu Bar on a Nongraphical Display Station with Underline Capability

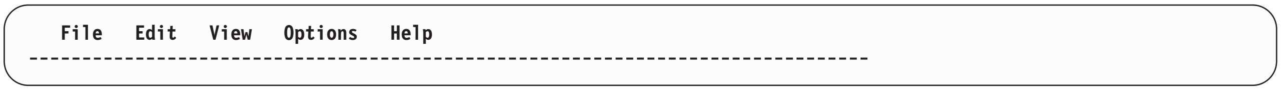
On display stations in configurations D and E from Table 16 on page 140, the menu bar looks like this:



```
   File     Edit   View   Options   Help
-------------------------------------------------------------------------------
```

Figure 39. Menu Bar on a Nongraphical Display Station without Underline Capability

On display stations in configuration F from Table 16 on page 140, the menu bar looks like this:



```
   File   Edit   View   Options   Help
-------------------------------------------------------------------------------
```

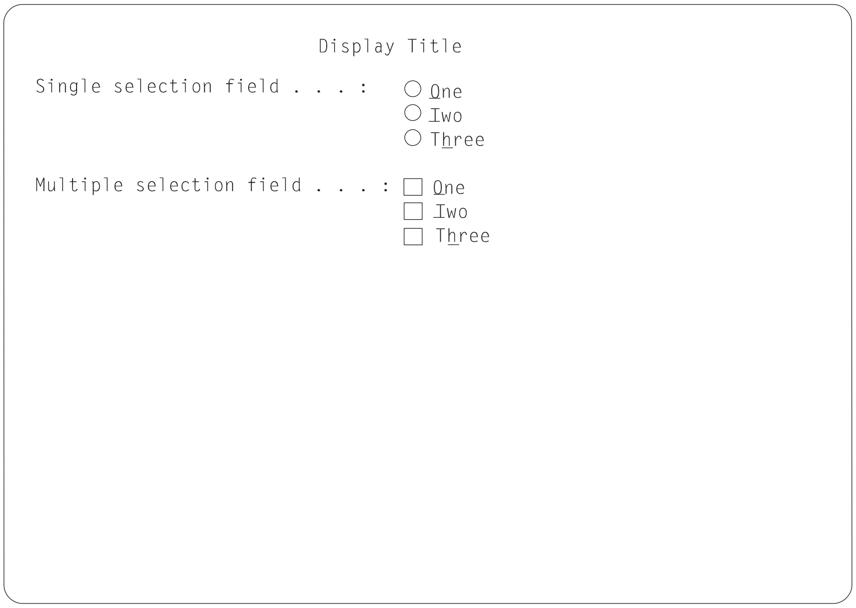
Figure 40. Menu Bar on a Display Station without Enhanced Interface

Selection Fields-Overview

There are two types of selection fields: single-choice and multiple-choice.

Single-choice selection fields and **multiple-choice selection fields** contain a fixed group of choices displayed in a vertical or horizontal list. You can select any number of choices from the multiple-choice selection field. You can only make one choice from the single-choice selection field.

On display stations in configurations A and B from Table 15 on page 139, the selection fields (vertical format) look like this:

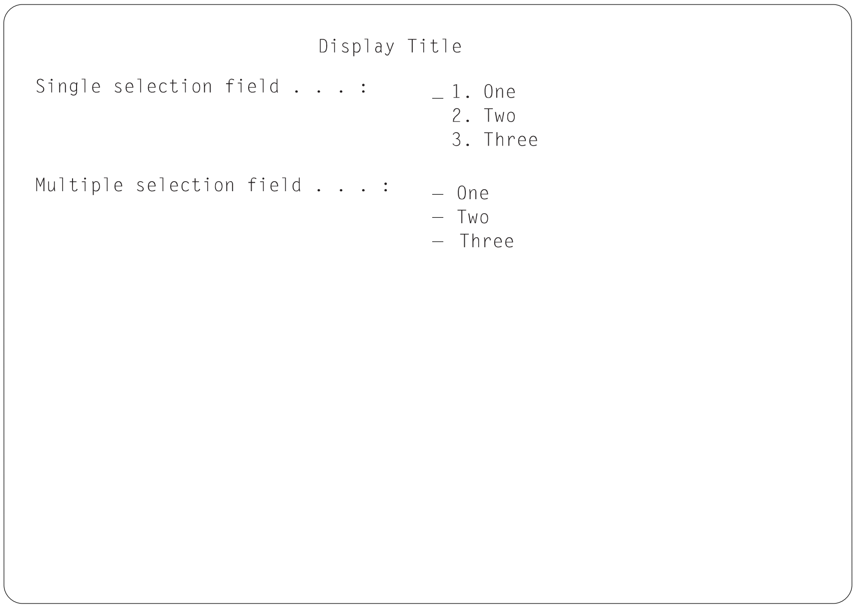


```
                         Display Title
Single selection field . . . :   ○ One
                                 ○ Two
                                 ○ Three

Multiple selection field . . . : □ One
                                 □ Two
                                 □ Three
```

RV2W860-1

Figure 41. Selection Fields on a Graphical Display Station with Enhanced Interface

On display stations in configuration C from Table 15 on page 139 and configurations D, E, and F from Table 16 on page 140, the selection fields (vertical format) look like this:



```
                         Display Title
Single selection field . . . :     _ 1. One
                                     2. Two
                                     3. Three

Multiple selection field . . . :   _ One
                                   _ Two
                                   _ Three
```

RV2W070-0

Figure 42. Selection Fields on a Nongraphical Display Station

DDS for Selection Fields-Example

Figure 43 on page 149 shows an example of the DDS for both single-choice and multiple-choice selection fields.

```
|...+....1....+....2....+....3....+....4....+....5....+....6....+....7...
     A          R RECORD
     A                                  2 30'Display Title'
     A                                  4  5'Single selection field . . . :'
     A            F1             2Y 0B  4 40SNGCHCFLD
     A                                      CHOICE(1 '>One')
     A                                      CHCCTL(1 &CTLONE1 MSG1111 QUSER/A)
     A    01                                CHOICE(2 '>Two')
     A                                      CHCCTL(2 &CTLTWO1 &MSG1 &LIB/&MSGF)
     A                                      CHOICE(3 'T>hree')
     A                                      CHCCTL(3 &CTLTHR1)
     A            CTLONE1        1Y 0H
     A            CTLTWO1        1Y 0H
     A            CTLTHR1        1Y 0H
     A            MSG1           7A  P
     A            LIB           10A  P
     A            MSGF          10A  P
     A                                  8  5'Multiple selection field . . . :'
     A            F2             2Y 0B  8 40MLTCHCFLD
     A                                      CHOICE(1 '>One')
     A                                      CHCCTL(1 &CTLONE2 MSG1112 QUSER/A)
     A    01                                CHOICE(2 '>Two')
     A                                      CHCCTL(2 &CTLTWO2 &MSG2 &LIB/&MSGF)
     A                                      CHOICE(3 'T>hree')
     A                                      CHCCTL(3 &CTLTHR2)
     A            CTLONE2        1Y 0H
     A            CTLTWO2        1Y 0H
     A            CTLTHR2        1Y 0H
     A            MSG2           7A  P
```

Figure 43. DDS for Single-Choice and Multiple-Choice Selection Fields

Creating a Vertical Single-Choice Selection Field

You can define the number of choices and the selection numbers for each choice. On display stations in configurations A and B from Table 15 on page 139, the choices are preceded by radio buttons. This is true unless *NOSLTIND is specified on the SNGCHCFLD keyword. A blank line appears between choices that are not sequential. The location that you specify for the single-choice selection field is the location of the input field (on a character-based nongraphical display). On display stations in configurations A and B from Table 15 on page 139, the location you specify for the field is the location of the first radio button.

Notes:

1. If you suppress the selection indicators, the location that you specify for the single-choice selection field is the location of the first character in the first choice.
2. If the single-choice selection field is within a pull-down menu, the location you specify is relative to the pull-down menu borders.

A single-choice selection field is a numeric field containing a SNGCHCFLD keyword and one or more CHOICE keywords. The CHOICE keywords define the choices within the single-choice selection field. Figure 43 shows an example of the keywords to use.

You can have choice numbers up to two digits long. The maximum number you can specify for a choice is 99. On output, if the field contains a choice number, that choice is the default selection.

The default is for vertical selection fields. You can create a horizontal selection field by using the *NUMCOL or *NUMROW values on the SNGCHCFLD keyword. See “Creating a Horizontal Selection Field” on page 150 for more information.

Creating a Vertical Multiple-Choice Selection Field

A multiple-choice selection field is a special numeric field containing:

- A MLTCHCFLD keyword to identify it as a multiple-choice selection field. When a user makes a selection, the field itself contains the number of choices that are selected.
- One or more CHOICE keywords that define the choices.
- A CHCCTL keyword for each choice to define a hidden field for each choice. The hidden field is used to indicate if the choice was selected. For more information on the CHCCTL keyword, see “Controlling the Availability of Choices” on page 168.

The default is for vertical selection fields. You can create a horizontal selection field by using the *NUMCOL or *NUMROW values on the MLTCHCFLD keyword. See “Creating a Horizontal Selection Field” for more information.

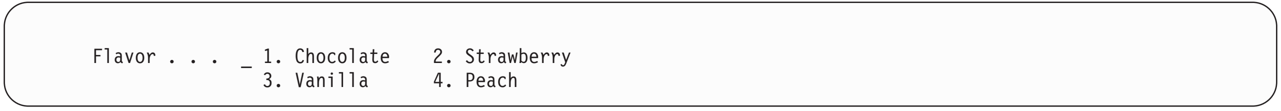
Creating a Horizontal Selection Field

The default orientation for single-choice and multiple-choice selection fields is vertical. To specify a horizontal field, use the *NUMCOL or *NUMROW values on the SNGCHCFLD and MLTCHCFLD keywords.

```
|...+....1....+....2....+....3....+....4....+....5....+....6....+....7...
     A          R RECORD
     A                                  2  2'Flavor . . . '
     A            F1            2Y 0B  2 16SNGCHCFLD((*NUMCOL 2))
     A                                      CHOICE(1 'Chocolate  ')
     A    01                                CHOICE(2 'Strawberry ')
     A                                      CHOICE(3 'Vanilla    ')
     A                                      CHOICE(4 'Peach      ')
```

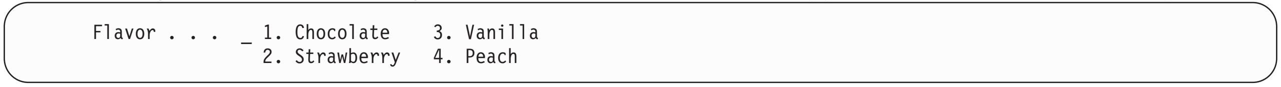
Figure 44. Example of DDS for Horizontal Selection Field. The `*NUMCOL 2` specifies that the field should display in two columns.

The following shows how this single-choice selection field would appear on a character-based display, assuming option indicator 01 is on:



```
Flavor . . .  _ 1. Chocolate    2. Strawberry
                3. Vanilla      4. Peach
```

The following shows how this single-choice selection field would appear if *NUMROW 2 were specified:



```
Flavor . . .  _ 1. Chocolate    3. Vanilla
                2. Strawberry   4. Peach
```

If the choices are nonsequential, no blank line or blank space is left for the omitted choice as would have happened with *NUMCOL.

You can optionally specify the number of spaces to appear between the choices by using the *GUTTER value on the SNGCHCFLD and MLTCHCFLD keywords. The gutter width must be at least 2. If *GUTTER is not specified, the default number of spaces between choices is 3.

Note: The gutter width includes the beginning and ending attribute of the choices on either side of the gutter.

The area occupied by the horizontal selection field is determined by the following:

- The number of choices specified

- The length of the longest choice
- The width of the gutter
- The number of columns specified
- The longest accelerator text specified (for a horizontal single-choice selection field in a pull-down menu)

A horizontal selection field must fit within the minimum display size specified for the file (24 x 80 or 27 x 132). If the record is a window or a pull-down menu, the horizontal selection field must fit within the minimum window size. Other fields may be specified to the right or to the left of a horizontal selection field. Option indicators can be specified on horizontal selection fields, as they can for other fields.

You can control which choices are displayed at one time by using option indicators on the CHOICE keywords. Unlike a vertical selection field, if a choice is optioned off, the remaining choices will be shifted to fill in the space.

Cursor Movement in a Vertical Selection Field

With *NORSTCSR on the SNGCHCFLD and MLTCHCFLD keywords, the cursor keys move the cursor to the next cursorable choice in the direction of the key that is pressed. The cursor skips null choices and choices defined as noncursorable. The up arrow key moves the cursor up one choice. The down arrow key moves the cursor down one choice. If the cursor is on the top choice and the up arrow key is pressed, the cursor leaves the field. Likewise, if the cursor is on the bottom choice and the down arrow is pressed, the cursor leaves the field. If the cursor input only (CSRINPONLY) keyword is in effect, the cursor moves to the next cursorable item on the display above or below the current cursor position. The cursor left and right keys move the cursor one space to the left or right.

Note: If the selection field is the only field defined within a pull-down menu, the cursor left and right keys close the present pull-down menu and open the next pull-down menu to the left or right.

To keep the cursor within a selection field, use the *RSTCSR value on the SNGCHCFLD and MLTCHCFLD keywords. If the cursor up key is pressed when the cursor is on the top choice in the list, the cursor moves to the last choice in the list. If the cursor down key is pressed when the cursor is on the bottom choice in the list, the cursor moves to the top choice in the list. If the cursor left key is pressed the cursor moves up one choice. If the cursor is on the top choice, the cursor moves to the bottom choice. If the cursor right key is pressed the cursor moves down one choice. If the cursor is on the bottom choice, the cursor moves to the top choice.

Note: If the selection field is the only field defined within a pull-down menu, the cursor left and right keys close the present pull-down menu and open the next pull-down menu to the left or right.

Cursor Movement in a Horizontal Selection Field

With *NORSTCSR on the SNGCHCFLD and MLTCHCFLD keywords, the cursor keys move the cursor to the next cursorable choice in the direction of the key that is pressed. The cursor skips null choices and choices defined as noncursorable. If the cursor is on the top choice of any column in the field and the up arrow key is pressed, the cursor leaves the field. Likewise, if the cursor is on the bottom choice of any column in the field and the down arrow key is pressed, the cursor leaves the field. If the cursor input only (CSRINPONLY) keyword is in effect, the cursor moves to the next cursorable item on the display above or below the current cursor position. If the left arrow key is pressed and there is a cursorable item to the left of the current choice, the cursor moves to the choice. If there is no cursorable item to the left, the cursor leaves the field. If the right arrow key is pressed and there is a cursorable item to the right of the current choice, the cursor moves to the choice. If there is no cursorable item to the right, the cursor leaves the field.

Note:

To keep the cursor within a selection field, use the *RSTCSR value on the SNGCHCFLD and MLTCHCFLD keywords. If the cursor up key is pressed when the cursor is on the top choice in any column in the field, the cursor moves to one of the following places:

- If there is a cursorable position in a column to the left, the cursor moves to the last choice in that column.
- If there is not a cursorable position in a column to the left or if there is no column to the left, the cursor moves to the last cursorable choice in right-most column in the field.

If the cursor down key is pressed when the cursor is on the last choice in any column in the field, the cursor moves to one of the following places:

- If there is a cursorable position in a column to the right, the cursor moves to the top choice in that column.
- If there is not a cursorable position in a column to the right or if there is no column to the right, the cursor moves to the first cursorable choice in left-most column in the field.

If the cursor left key is pressed and there is a cursorable choice to the left of the current choice, the cursor moves to that choice. If there is no cursorable choice to the left, the cursor moves to the first cursorable choice in the row above the present row (closest row, right-most choice). If the present row is the top row, the cursor moves to the right-most choice in the last row.

If the cursor right key is pressed and there is a cursorable choice to the right of the current choice, the cursor moves to that choice. If there is no cursorable choice to the right, the cursor moves to the first cursorable choice in the row below the present row (closest row, left-most choice). If the present row is the bottom row, the cursor moves to the left-most choice in the first row.

Note: The cursor left and right keys will close the present pull-down menu and open the next pull-down menu to the left or right when the following are true:

- The selection field is the only field defined within a pull-down menu
- There is no cursorable choice to the left or right of the current choice

Controlling the Selection Indicators in a Selection Field

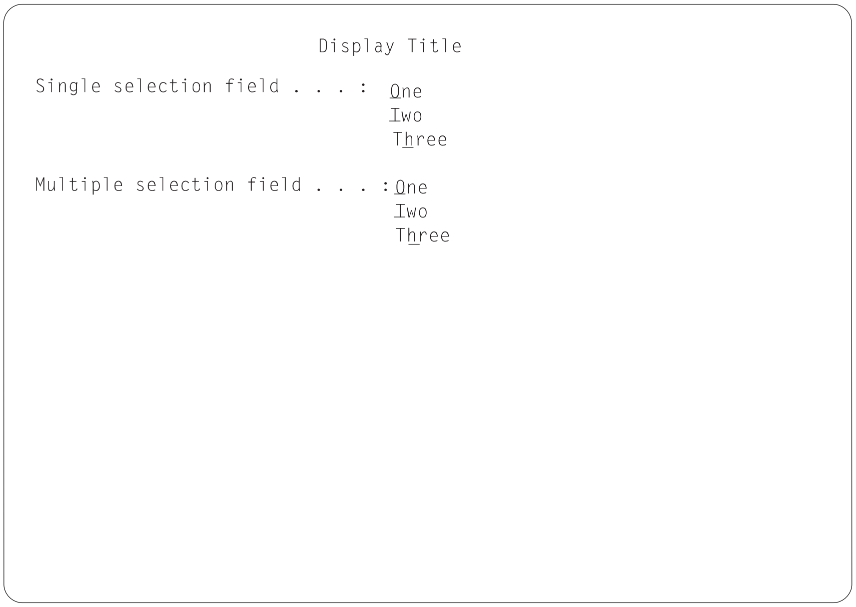
A **selection indicator** is an indicator that precedes a choice in a selection field or a selection list. It is used to select the choice or to show that a choice has been selected. An example of a selection indicator is a radio button. Radio buttons appear before choices in single-choice selection fields and single-choice selection lists. The default is for selection indicators to appear in selection fields. You can suppress the selection indicators in a selection field by specifying the *NOSLTIND parameter on the SNGCHCFLD and MLTCHCFLD keywords.

The *NOSLTIND value is ignored for display stations that are not attached to a controller that supports an enhanced interface for nonprogrammable work stations. Figure 45 on page 153 is an example of the DDS to suppress the selection indicators in a selection field.

```
|...+....1....+....2....+....3....+....4....+....5....+....6....+....7...
     A          R RECORD
     A                                  2 30'Display Title'
     A                                  4  5'Single selection field . . . :'
     A            F1             2Y 0B  4 40SNGCHCFLD(*NOSLTIND)
     A                                      CHOICE(1 '>One')
     A    01                                CHOICE(2 '>Two')
     A                                      CHOICE(3 'T>hree')
     A                                  8  5'Multiple selection field . . . :'
     A            F2             2Y 0B  8 40MLTCHCFLD(*NOSLTIND)
     A                                      CHOICE(1 '>One')
     A    01                                CHOICE(2 '>Two')
     A                                      CHOICE(3 'T>hree')
```

Figure 45. DDS for Suppressing Selection Indicators in a Selection Field

On display stations in configurations A and B from Table 15 on page 139, the selection fields look like this:



```
                          Display Title

  Single selection field . . . :  One
                                  Two
                                  Three

  Multiple selection field . . . :One
                                  Two
                                  Three
```

RV3W067-0

Figure 46. Suppressed Selection Indicators in Selection Field

Creating Pull-Down Menus Using Single-Choice Selection Fields

When you select a choice from the menu bar, a pull-down menu appears. A pull-down menu is a group of actions associated with a menu bar choice. Figure 47 is an example.

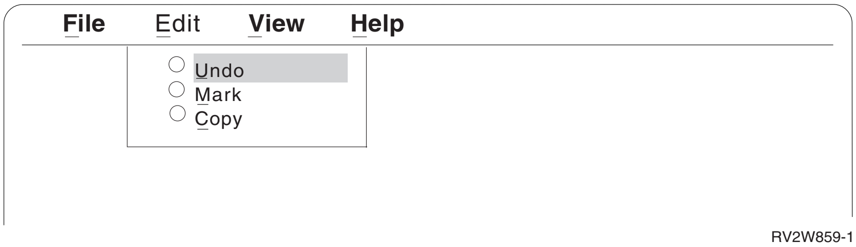


Figure 47. Example of a Pull-Down Menu

You must define the pull-down menus and the corresponding menu bar in the same file.

The last field in any pull-down menu always operates as though the CHECK(FE) (Field Exit) keyword were specified. This keeps the cursor in the pull-down menu after you enter the input that is in the last field in the pull-down menu. Then, if you press the Field Exit key with the cursor in the last field, the field is cleared and the cursor moves to the next pull-down menu. If the last field did not operate with CHECK(FE), the cursor automatically moves on to the next pull-down menu after you press the Enter key.

A pull-down record can contain anything that a window record can contain. However, use only single-choice selection fields or multiple-choice selection fields in a pull-down menu. If you use fields other than single-choice selection fields or multiple-choice selection fields, the cursor does not move consistently on all display stations. On display stations attached to a controller that supports an enhanced interface for nonprogrammable work stations, the cursor-right keys and cursor-left keys display the next pull-down menu when both of these conditions are true:

- There is only one selection field in the displayed pull-down menu, and
- The cursor is positioned on the selection field.

On display stations attached to a controller that does not support an enhanced interface for nonprogrammable work stations, the cursor movement keys move the cursor one character position within the pull-down menu.

Figure 48 is an example of the DDS for a pull-down menu. The figures that follow show how the pull-down menu appears on each type of display.

Note: Assume that the record PULLDOWN is specified on the MNUBARCHC keyword for the Edit choice, and that the MARKTXT field contains the text, `>Mark`.

```
|...+....1....+....2....+....3....+....4....+....5....+....6....+....7...
     A          R PULLDOWN                  PULLDOWN
     A            F1             2Y 0B  1  1SNGCHCFLD
     A    01                                CHOICE(1 '>Undo       ')
     A                                      CHOICE(2 &MARKTXT)
     A                                      CHOICE(3 '>Copy       ')
     A
     A                                      :
```

Figure 48. DDS for a Pull-Down Menu

On display stations in configurations A and B from Table 15 on page 139, the pull-down menu looks like this:

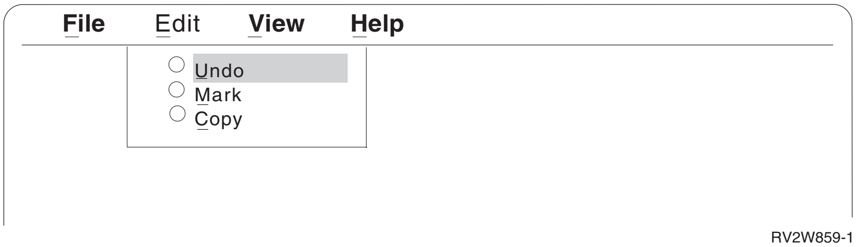


Figure 49. Pull-Down Menu on a Graphical Display Station with Enhanced Interface

On display stations in configuration C from Table 15 on page 139, the pull-down menu looks like this:

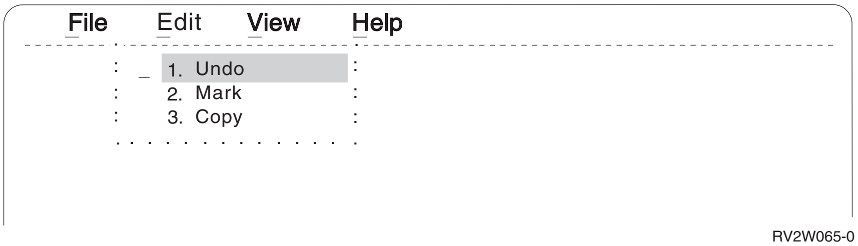


Figure 50. Pull-Down Menu on a Nongraphical Display Station with Underline Capability

On display stations in configurations D and E from Table 16 on page 140, the pull-down menu looks like this:

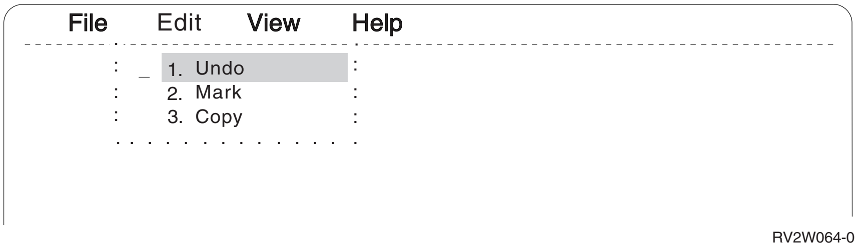


Figure 51. Pull-Down Menu on a Nongraphical Display Station without Underline Capability

On display stations in configuration F from Table 16 on page 140, the pull-down menu looks like this:

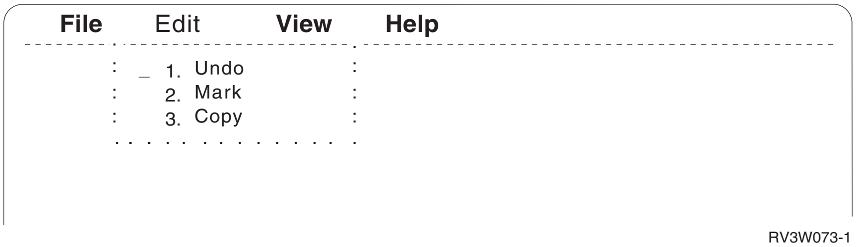


Figure 52. Pull-Down Menu on a Display without Enhanced Interface

Controlling the Selection Indicators in a Pull-Down Menu

A **selection indicator** is an indicator that precedes a choice in a selection field or a selection list. It is used to select the choice or to show that a choice has been selected. An example of a selection indicator is a radio button. Radio buttons appear before choices in single-choice selection fields and single-choice selection lists. The default is for selection indicators to appear in selection fields. You can suppress the selection indicators in a pull-down menu by specifying the *NOSLTIND parameter on the PULLDOWN keyword. Figure 53 is an example of the DDS to suppress the selection indicators in a pull-down menu.

```
|...+....1....+....2....+....3....+....4....+....5....+....6....+....7...
     A          R PULLDOWN                  PULLDOWN(*NOSLTIND)
     A            F1              2Y 0B  1  1SNGCHCFLD
     A    01                                CHOICE(1 '>Undo       ')
     A                                      CHOICE(2 &MARKTXT)
     A                                      CHOICE(3 '>Copy       ')
```

Figure 53. DDS for Suppressing Selection Indicators in a Pull-Down Menu

On display stations in configurations A and B from Table 15 on page 139, the pull-down menu looks like this:

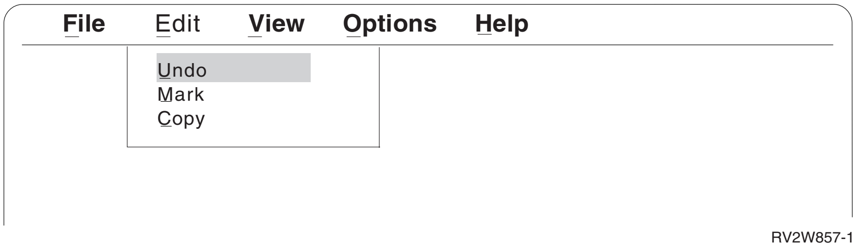


Figure 54. Suppressed Selection Indicators on Graphical Display Station

On display stations in configuration C from Table 15 on page 139, the pull-down menu looks like this:

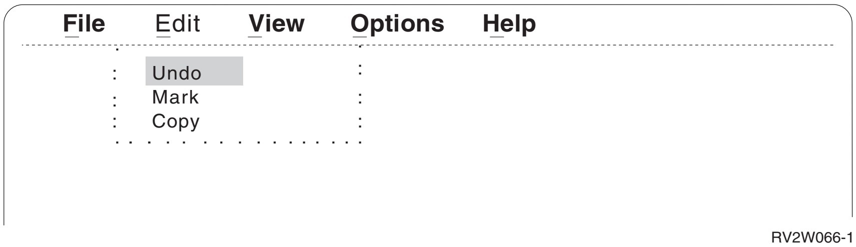


Figure 55. Suppressed Selection Indicators on Nongraphical Display Station

The *NOSLTIND value is ignored for display stations that are not attached to a controller that supports an enhanced interface for nonprogrammable work stations.

Defining Accelerator Keys

An **accelerator key** is a function key that starts the application-defined function and is displayed next to a pull-down menu choice.

You can specify accelerators for a single-choice selection field in a pull-down menu by doing the following:

1. Specify the necessary CFnn keys.
2. Use the CHCACCEL keyword.

Specify the accelerator text on the CHCACCEL keyword. You can use a P-field to specify the text. Note that the CHCACCEL keyword does not define the accelerator key itself. You must define the CFnn keyword for the key and design your application to recognize this key as an accelerator for this choice. You must also ensure that the text you specify on CHCACCEL correctly reflects the key you have defined. For example, if you want CF08 to be an accelerator key, specify something like F8 or CF08 on the CHCACCEL keyword for the appropriate choice.

The accelerator text appears three spaces after the length of the longest choice text in the field.

Because the accelerator key functions even if the pull-down menu is not displayed, you should define the necessary CFnn keys at the file level. If you define them at the record level, specify them for every record from which they should be available.

Single-choice selection fields may be defined in any record. However, an accelerator can be defined only for a single-choice selection field within a pull-down menu. Figure 56 is an example.

```
|...+....1....+....2....+....3....+....4....+....5....+....6....+....7...
     A          R PULLEDIT                CF04 CF06
     A                                      PULLDOWN
     A            F1             2Y 0B  1  1SNGCHCFLD
     A                                      CHECK(ER)
     A    01                                CHOICE(1 '>Undo       ')
     A                                      CHCACCEL(1 'F4')
     A                                      CHOICE(2 &MARKTXT)
     A                                      CHCACCEL(2 &F6)
     A                                      CHOICE(3 '>Copy       ')
     A            MARKTXT       20A  P
     A            F6             2   P
```

Figure 56. DDS for Accelerator Keys

On display stations in configurations A and B from Table 15 on page 139, the pull-down menu looks like this:

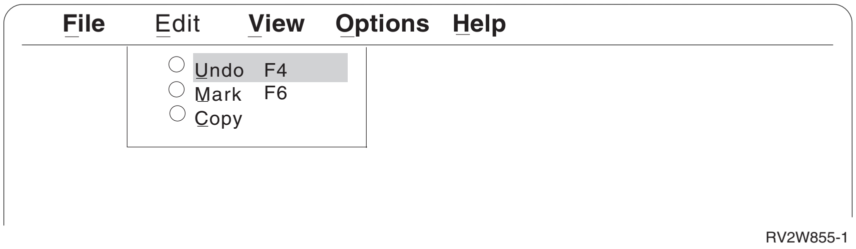


Figure 57. Accelerators in a Pull-Down Menu

Defining a Menu-Bar Switch Key

A **menu-bar switch key** alternates the cursor between the menu bar and the application display.

You can define a menu-bar switch key using the MNUBARSW keyword at either the file level or the record level. If the cursor is in the application record, pressing the menu-bar switch key moves the cursor to the first choice in the menu bar. Pressing the key again moves the cursor from the menu bar back to its previous location in the application record. If you move the cursor using the cursor keys from the application record to the menu bar and then press the menu-bar switch key, the cursor returns to its initial position on the application record. This is the first input field unless cursor positioning keywords are specified on the application record. If a pull-down menu is displayed, pressing the menu-bar switch key cancels the pull-down menu and moves the cursor to the application record.

The system always handles the menu-bar switch key regardless of whether the application or the system displayed the menu bar. (For more information, see the **DDS** topic in the iSeries Information Center.)

For the menu-bar switch key to be active, it must have been active on the last record written to the display. The easiest way to ensure that the menu-bar switch key will always be active is to specify MNUBARSW at the file level. If you specify MNUBARSW at the record level, you must specify it on all records on which it should be active.

Defining a Cancel Key

You can define a cancel key for the menu-bar record and pull-down menu records using the MNUCNL keyword. You can define them either at the file level or the record level.

A **cancel key** closes a pull-down menu and moves the cursor to the associated choice on the menu bar. This is true even if the cursor is not in the pull-down menu. If no pull-down menus are displayed and the cursor is located within the menu bar, the key cancels the menu bar and moves the cursor back to the application record. This location is the cursor's previous location in the application record if the menu-bar switch key was used to move the cursor to the menu bar. If the cursor had been moved to the menu bar with the cursor keys, this location is the initial location of the cursor within the application record. If no pull-down menus are displayed and the cursor is located on the application record, the key returns control to the application program. In this case, the MNUCNL keyword works just as any other key definition keyword, and includes the ability to return a response indicator.

Like the menu-bar switch key, the cancel key is active only if it was active for the last record written to the display. The easiest way to ensure that the cancel key will always be active is to specify the MNUCNL keyword at the file level. If you use the MNUCNL keyword at the record level, you must specify it on all records on which it should be active.

Figure 58 shows how to use the MNUBARSW keyword and the MNUCNL keyword. The example sets up command attention key 10 as the menu-bar switch key and command attention key 12 as the cancel key. (These settings are the defaults.)

```
|...+....1....+....2....+....3....+....4....+....5....+....6....+....7...
     A                                      MNUBARSW(CA10) MNUCNL(CA12)
     A          R MENUBAR                   MNUBAR
     A            MNUFLD         2Y 0B  1  2
     A                                      MNUBARCHC(1 PULLFILE +
     A                                                '>File        ')
     A  02                                  MNUBARCHC(2 PULLEDIT +
     A                                                &EDITTXT)
     A                                      MNUBARCHC(3 PULLVIEW +
     A                                                '>View        ')
     A  04                                  MNUBARCHC(4 PULLOPT +
     A                                                '>Options      ')
     A                                      MNUBARCHC(5 PULLHELP +
     A                                                '>Help         ')
     A            EDITTXT       20A  P
     A            .
     A            .
     A            .
```

Figure 58. DDS for Menu-Bar Switch Key and Cancel Key

Limiting Function When Cursor is Outside a Pull-Down Menu

If *NORSTCSR is specified on the PULLDOWN keyword, the user may move the cursor out of the active window and use any command function (CF) key. *NORSTCSR is the default.

If *RSTCSR is specified on the PULLDOWN keyword, and the cursor is moved outside the pull-down menu, only the Print and Home command function (CF) keys are active. If the work station user presses any other command function (CF) key, the alarm sounds and the cursor is moved back to its position for the previous write operation.

Selection Lists-Overview

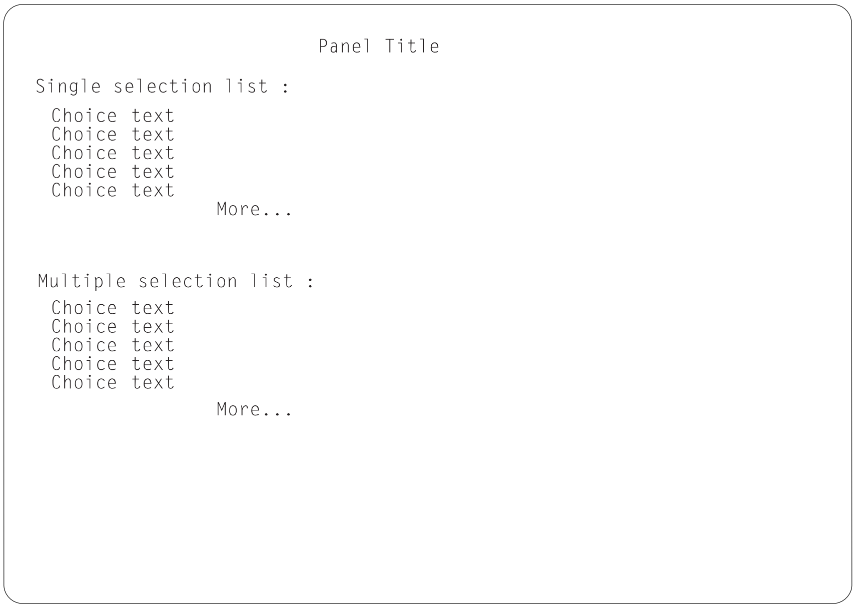
A **selection list** is a potentially scrollable list from which the user can select an item. There are two types of selection lists: single-choice and multiple-choice. A **single-choice selection list** is a potentially scrollable list from which the user can select one item. A **multiple-choice selection list** is a potentially scrollable list from which the user can select one or more items.

Single-choice selection lists and multiple-choice selection lists contain a group of choices displayed in a vertical list. These choices can be scrolled either by using the Page Up and Page Down keys or by using a scroll bar. A scroll bar is a part of a display that shows a user that more information is available in a particular direction and can be moved into view by using a pointing device or the page keys. For more information on scroll bars, see “Scroll Bars-Overview” on page 163.

You can select any number of choices from the multiple-choice selection list. You can only make one choice from the single-choice selection list. “Selection Lists-Overview” shows an example of a single-choice selection list, multiple-choice selection list, and scroll bars used with the selection lists.

The DDS in Figure 61 on page 160 produces the following displays:

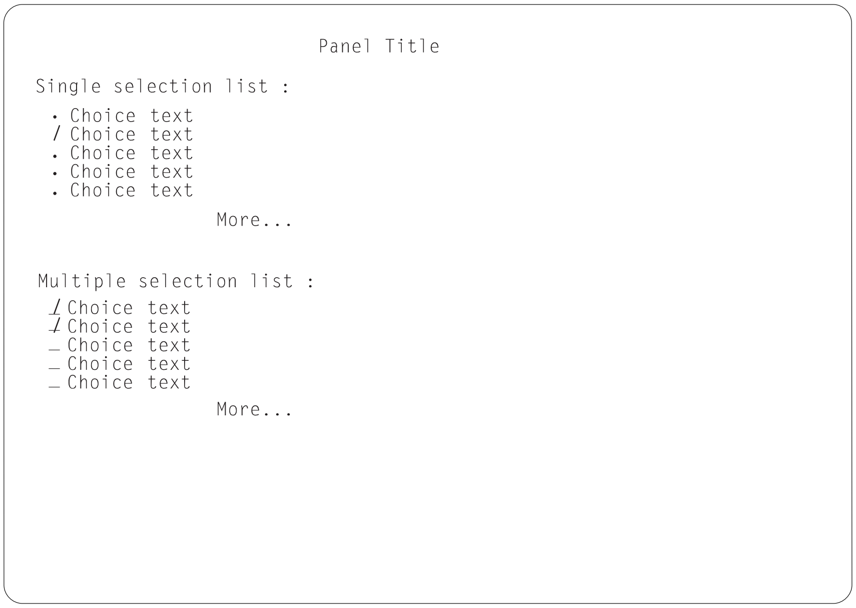
On display stations in configurations A and B from Table 15 on page 139, the selection lists look like this:



```
                              Panel Title

  Single selection list :

    Choice text
    Choice text
    Choice text
    Choice text
    Choice text
                   More...


  Multiple selection list :

    Choice text
    Choice text
    Choice text
    Choice text
    Choice text
                   More...
```

Figure 59. Selection Lists on a Graphical Display Station with Enhanced Interface

On display stations in configuration F from Table 16 on page 140, the selection lists (vertical format) look like this:



```
                        Panel Title

  Single selection list :

   • Choice text
   / Choice text
   • Choice text
   • Choice text
   • Choice text
                 More...


  Multiple selection list :

   / Choice text
   / Choice text
   _ Choice text
   _ Choice text
   _ Choice text
                 More...
```

Figure 60. Selection Lists on a Nongraphical Display Station with Underline Capability

DDS for Selection Lists-Example

The DDS in Figure 61 creates the displays shown in “Selection Lists-Overview” on page 159.

```
|...+....1....+....2....+....3....+....4....+....5....+....6....+....7...
     A          R SFLRCD                    SFL
     A            CTLFLD         1Y 0H      SFLCHCCTL
     A            F1            11A  O  6 10
     A          R SFLCTLRCD                 SFLCTL(SFLRCD)
     A                                      SFLSNGCHC
     A                                      SFLPAG(5) SFLSIZ(&SFLSIZ)
     A                                      SFLDSP SFLDSPCTL
     A                                      ROLLUP(10)
     A    10                                SFLEND(*MORE)
     A            F3             5S 0H      SFLSCROLL
     A            F2             4S 0H      SFLRCDNBR(CURSOR *TOP)
     A            SFLSIZ         5S 0P
     A                                  1 30'Panel Title'
     A                                  4  5'Single selection list:'
     A          R SFLRCD2                   SFL
     A            CTLFLD         1Y 0H      SFLCHCCTL
     A            F1            11A  O 13 10
     A          R SFLCTLRC2                 SFLCTL(SFLRCD2)
     A                                      SFLMLTCHC(&NUMSEL *RSTCSR)
     A                                      SFLPAG(5) SFLSIZ(&SFLSIZ)
     A                                      SFLDSP SFLDSPCTL
     A    10                                SFLEND(*MORE)
     A                                      ROLLUP(10)
     A            F2             4S 0H      SFLRCDNBR(CURSOR *TOP)
     A            F3             5S 0H      SFLSCROLL
     A            SFLSIZ         5S 0P
     A            NUMSEL         4Y 0H
     A                                 11  5'Multiple selection list:'
```

Figure 61. DDS for Selection Lists-Example

Creating Selection Lists

Selection lists are created using subfiles. For more information on subfiles, see Chapter 4, “Displaying Groups of Records Using Subfiles.” For each selection list you must specify a subfile record format and a subfile control record format. Within the subfile record format you must specify an output field for the text of the choice. You can specify only one output-only field in the record.

To specify a default choice in a selection list, use either the subfile next-changed (SFLNXTCHG) keyword or a control field. To use the SFLNXTCHG keyword, specify the keyword on the text field within the subfile record. To use a control field, specify a control field in the subfile record and specify the subfile choice control (SFLCHCCTL) keyword on the field. The control field can have the following values:

Control Value	Meaning on Output	Meaning on Input
0	Available	Not selected
1	Selected	Selected
2	Unavailable. Cannot place cursor on choice unless help for choice is available.[1],[2]	
3	Unavailable. Placing cursor on choice is allowed.	
4	Unavailable. Cannot place cursor on choice even if help for choice is available.[1],[2]	

Notes:

1. Applies only to displays attached to a controller that supports an enhanced interface for nonprogrammable work stations.
2. If the choice is the first choice displayed and there are no other cursorable choices in the list, the choice will be made unavailable and cursorable. Otherwise, an invalid data stream error would be issued.

The application uses the get-next-changed operation to determine which choices are selected. The get-next-changed operation returns all of the changed records. If the user deselects a default choice, the get-next-changed operation returns the deselected choice record because its control value changed. To have the get-next-changed operation return only the selected choices, specify the subfile return selected choices (SFLRTNSEL) keyword on the subfile control record. The next get-next-changed operation will return the selected choice. Then, perform an update operation to the default choice to change its control value to 0.

On display stations in configurations A and B from Table 15 on page 139, the choices are preceded by radio buttons (single-choice) and check boxes (multiple-choice). This is true unless *NOSLTIND is specified on the subfile single-choice selection list (SFLSNGCHC) and multiple-choice selection list (SFLMLTCHC) keywords. The location that you specify for the first field in the subfile record format is the location of the input fields (on a character-based nongraphical display). On display stations in configurations A and B from Table 15 on page 139, the location of the first field is the location of the first radio button or check box.

Notes:

1. If you suppress the selection indicators, the location that you specify for the first field in the subfile record format is the location of the first character in the first choice.
2. If the selection list is within a pull-down menu, the location you specify is relative to the pull-down menu borders.

The SFLSCROLL keyword is used to return the relative record number of the record at the top of the current page of records. If the user presses Enter, SFLSCROLL returns the relative record number of record that is currently displayed at the top of the page. If control is returned to the application because of the ROLLUP keyword, SFLSCROLL returns the relative record number of the last record in the subfile plus 1. If control is returned to the application because of the ROLLDOWN keyword, SFLSCROLL always returns 1.

To redisplay the subfile with the correct subfile record at the top of the list of choices, use the subfile record number (SFLRCDNBR) keyword. Specify SFLRCDNBR(*TOP) as a hidden field and use the relative record number returned by the SFLSCROLL keyword. Add more records to the subfile and redisplay the subfile.

Controlling the Selection Indicators in a Selection List

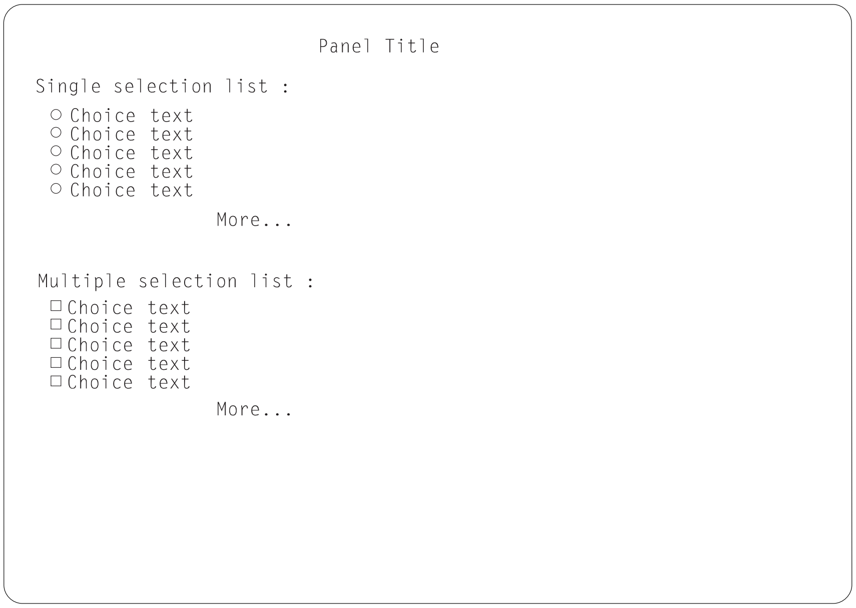
A **selection indicator** is an indicator that precedes a choice in a selection field or a selection list. It is used to select the choice or to show that a choice has been selected. An example of a selection indicator is a radio button. Radio buttons appear before choices in single-choice selection fields and single-choice selection lists. The default is for selection indicators to not appear in selection lists. You can have the selection indicators display in a selection list by specifying the *SLTIND parameter on the SFLSNGCHC and SFLMLTCHC keywords.

The *SLTIND value is ignored for display stations that are not attached to a controller that supports an enhanced interface for nonprogrammable work stations. Figure 62 is an example of the DDS to enable the selection indicators in a selection list.

```
|...+....1....+....2....+....3....+....4....+....5....+....6....+....7...
     A          R SFLRCD                    SFL
     A            CTLFLD         1Y 0H      SFLCHCCTL
     A            F1            11A  O  6 10
     A          R SFLCTLRCD                 SFLCTL(SFLRCD)
     A                                      SFLSNGCHC(*SLTIND)
     A                                      SFLPAG(5) SFLSIZ(&SFLSIZ)
     A                                      SFLDSP SFLDSPCTL
     A                                      ROLLUP(10)
     A    10                                SFLEND(*MORE)
     A            F3             5S 0H      SFLSCROLL
     A            F2             4S 0H      SFLRCDNBR(CURSOR *TOP)
     A            SFLSIZ         5S 0P
     A                                  1 30'Panel Title'
     A                                  4  5'Single selection list:'
     A          R SFLRCD2                   SFL
     A            CTLFLD         1Y 0H      SFLCHCCTL
     A            F1            11A  O 13 10
     A          R SFLCTLRC2                 SFLCTL(SFLRCD2)
     A                                      SFLMLTCHC(&NUMSEL *RSTCSR *SLTIND)
     A                                      SFLPAG(5) SFLSIZ(&SFLSIZ)
     A                                      SFLDSP SFLDSPCTL
     A    10                                SFLEND(*MORE)
     A                                      ROLLUP(10)
     A            F2             4S 0H      SFLRCDNBR(CURSOR *TOP)
     A            F3             5S 0H      SFLSCROLL
     A            SFLSIZ         5S 0P
     A            NUMSEL         4Y 0H
     A                                 11  5'Multiple selection list:'
```

Figure 62. DDS for Enabling Selection Indicators in a Selection List

On display stations in configurations A and B from Table 15 on page 139, the selection list looks like this:



```
                         Panel Title

  Single selection list :
   ○ Choice text
   ○ Choice text
   ○ Choice text
   ○ Choice text
   ○ Choice text
                  More...


  Multiple selection list :
   □ Choice text
   □ Choice text
   □ Choice text
   □ Choice text
   □ Choice text
                  More...
```

RV3W0079-0

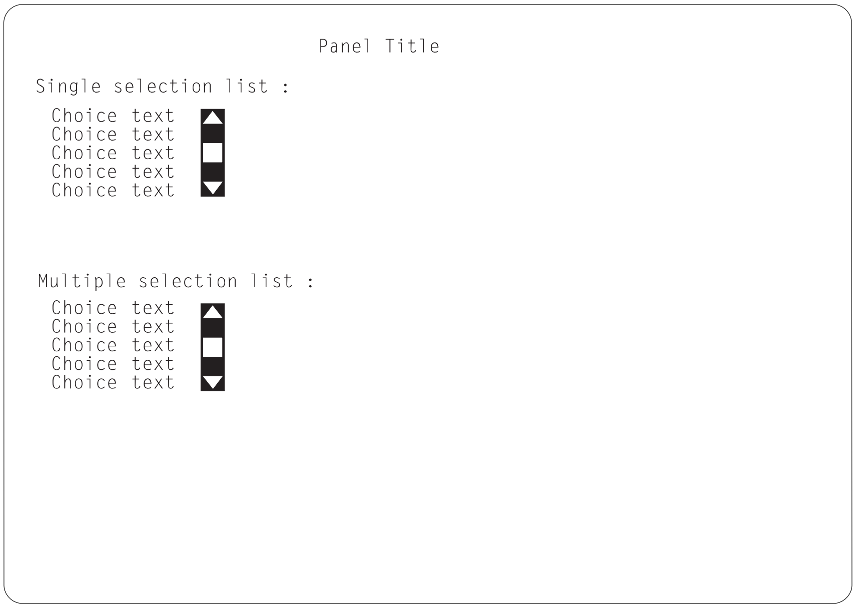
Figure 63. Selection Indicators on Graphical Display Station

Scroll Bars-Overview

A **scroll bar** is a part of a display that shows a user that more information is available in a particular direction and can be moved into view by using a pointing device or the page keys. A scroll bar can be defined for any subfile. The examples in this section show examples of scroll bars used with a single-choice selection list and a multiple-choice selection list.

The DDS in Figure 66 on page 166 produces the following displays:

On display stations in configurations A and B from Table 15 on page 139, the selection lists and scroll bars look like this:

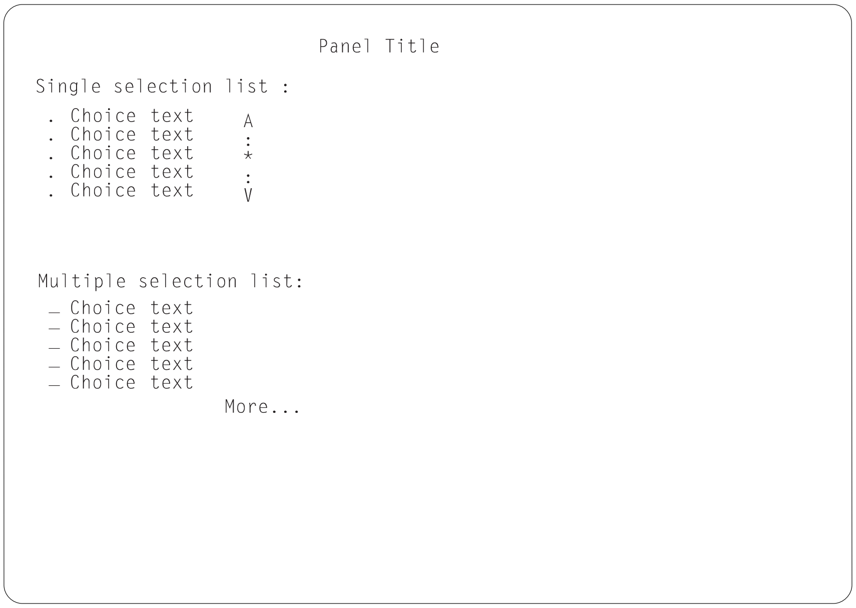


```
                        Panel Title
 Single selection list :
  Choice text
  Choice text
  Choice text
  Choice text
  Choice text



 Multiple selection list :
  Choice text
  Choice text
  Choice text
  Choice text
  Choice text
```

RV3W078-0

Figure 64. Scroll Bar on a Graphical Display Station with Enhanced Interface

On display stations in configuration F from Table 16 on page 140, the selection lists (vertical format) look like this:



```
                        Panel Title
 Single selection list :
  . Choice text    A
  . Choice text    :
  . Choice text    *
  . Choice text    :
  . Choice text    V



 Multiple selection list:
  _ Choice text
  _ Choice text
  _ Choice text
  _ Choice text
  _ Choice text
                 More...
```

RV3W080-0

Figure 65. Scroll Bar on a Nongraphical Display Station with Underline Capability

Creating a Scroll Bar

To create a scroll bar, use the scroll bar (*SCRBAR) value on the subfile end (SFLEND) keyword. The *SCRBAR value creates a graphical scroll bar on graphical display stations. It creates a character scroll bar on nongraphical display stations. The scroll bar appears to the right of the longest choice in the list. To have the scroll bar appear further to the right, add blanks to the text for the selection list choices.

In most cases, the number of subfile records represented by the scroll bar is the number of records that have been written to the subfile. The SFLEND keyword optioned on indicates that no more records will be written to the subfile. The SFLPAG value is not added to the number of records represented by the scroll bar. When the bottom of the subfile is reached and the PAGEDOWN keyword is not active, the scroll bar box is displayed exactly above the lower scroll bar button.

The SFLEND keyword optioned off indicates that more records will be written to the subfile. The SFLPAG value is added to the number of records represented by the scroll bar. Adding the SFLPAG value causes the scroll bar to appear as if more subfile records exist after the last subfile record. When the last subfile record is reached and the PAGEDOWN keyword is active, the scroll bar box is not displayed exactly above the lower scroll bar button. This indicates there are more records to display. If the PAGEDOWN keyword is active, control is given back to the application if the user tries to page down or scroll to the unseen records. The application can then write more records to the subfile.

Note: The SFLPAG value is not added to the number of records represented by the scroll bar when PAGEDOWN is active and when the number of records written is less than the SFLSIZ value. In this case, the number of subfile records represented by the scroll bar is the SFLSIZ value. The SFLSIZ value is used to show the total size of the subfile and allows the application to fill only the subfile records that the user wants to see.

If the PAGEDOWN keyword is active, no partial pages are displayed. If the user tries to roll to a partial page, control is given back to the application. The subfile roll value (SFLROLVAL) may override this. For more information on the SFLROLVAL keyword, see the **DDS** topic in the iSeries Information Center. If the PAGEDOWN keyword is not active, the SFLPAG value minus 1 is added to the number of records represented by the scroll bar. Adding the SFLPAG value minus 1 enables a partial page to be displayed.

Note: If a scroll bar is displayed with a horizontal subfile, you may not be able to use the top scroll button when the following are true:

- A partial page is reached
- Records are displayed in only the first column of the horizontal subfile

See Table 17 for a summary of how the scroll bar is sized under different conditions.

Table 17. How a Scroll Bar is Sized

			then the following values are added (yes or no) to the number of records represented by the scroll bar			
If SFLEND indicator is ...	**and PAGEDOWN is ...**	**and the number of records written is ...**	**SFLPAG**	**SFLPAG - 1**	**Relative record number of last record**	**SFLSIZ**
On	Active	≥SFLSIZ	No	No	Yes	No
		<SFLSIZ	No	No	No	Yes
	Not active	≥SFLSIZ	No	Yes	Yes	No
		<SFLSIZ	No	Yes	Yes	No
Off	Active	≥SFLSIZ	Yes	No	Yes	No
		<SFLSIZ	No	No	No	Yes
	Not active	≥SFLSIZ	Yes	Yes	Yes	No
		<SFLSIZ	Yes	Yes	Yes	No

DDS for Scroll Bars-Example

The DDS in Figure 66 on page 166 creates the displays shown in “Scroll Bars-Overview” on page 163.

```
|...+....1....+....2....+....3....+....4....+....5....+....6....+....7...
     A          R SFLRCD                    SFL
     A            CTLFLD         1Y 0H      SFLCHCCTL
     A            F1            11A  O  6 10
     A          R SFLCTLRCD                 SFLCTL(SFLRCD)
     A                                      SFLSNGCHC
     A                                      SFLPAG(5) SFLSIZ(&SFLSIZ)
     A                                      SFLDSP SFLDSPCTL
     A                                      ROLLUP(10)
     A    10                                SFLEND(*SCRBAR)
     A            F3             5S 0H      SFLSCROLL
     A            F2             4S 0H      SFLRCDNBR(CURSOR *TOP)
     A            SFLSIZ         5S 0P
     A                                  1 30'Panel Title'
     A                                  4  5'Single selection list:'
     A          R SFLRCD2                   SFL
     A            CTLFLD         1Y 0H      SFLCHCCTL
     A            F1            11A  O 13 10
     A          R SFLCTLRC2                 SFLCTL(SFLRCD2)
     A                                      SFLMLTCHC(&NUMSEL *RSTCSR)
     A                                      SFLPAG(5) SFLSIZ(&SFLSIZ)
     A                                      SFLDSP SFLDSPCTL
     A    10                                SFLEND(*SCRBAR *MORE)
     A                                      ROLLUP(10)
     A            F2             4S 0H      SFLRCDNBR(CURSOR *TOP)
     A            F3             5S 0H      SFLSCROLL
     A            SFLSIZ         5S 0P
     A            NUMSEL         4Y 0H
     A                                 11  5'Multiple selection list:'
```

Figure 66. DDS for Scroll Bars-Example

Scroll Bar Operation

See Table 18 for a summary of how the scroll bar operates.

Table 18. Scroll Bar Operation

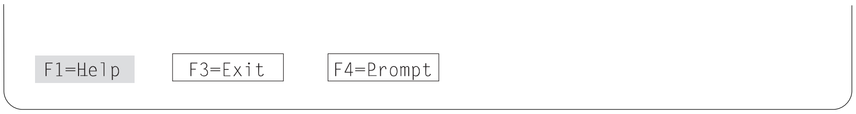
If the user ...	then ...
Clicks once on the top scroll button	The subfile is scrolled one record toward the bottom of the subfile
Clicks once on the bottom scroll button	The subfile is scrolled one record toward the top of the subfile
Clicks once on the shaft above the scroll box or presses the Page Down key	The subfile is scrolled one page toward the bottom of the subfile
Clicks once on the shaft below the scroll box or presses the Page Up key	The subfile is scrolled one page toward the top of the subfile
Drags the scroll box with the selection button and releases the selection button	The subfile page is scrolled to correspond to the position indicated by the scroll box

Push Buttons-Overview

A **push button** is a button, labeled with text, graphics, or both that represents an action that starts when a user selects the push button.

The DDS in Figure 71 on page 167 produces the following displays:

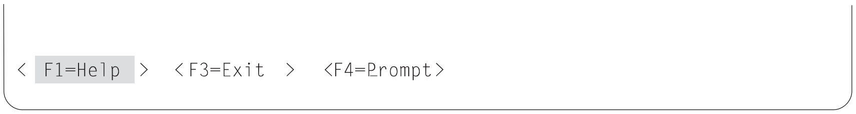
On display stations in configurations A and B from Table 15 on page 139, the push buttons look like this:


F1=Help   F3=Exit   F4=Prompt

RV3W070-0

Figure 67. Push Buttons on a Graphical Display Station with Enhanced Interface

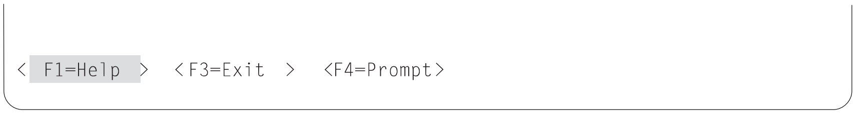
On display stations in configuration C from Table 15 on page 139, the push buttons look like this:


< F1=Help > <F3=Exit > <F4=Prompt>

RV3W071-0

Figure 68. Push Buttons on a Nongraphical Display Station with Underline Capability

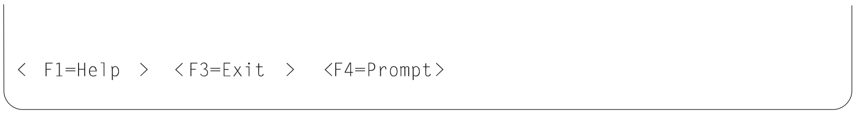
On display stations in configurations D and E from Table 15 on page 139, the push buttons look like this:


< F1=Help > <F3=Exit > <F4=Prompt>

RV3W072-0

Figure 69. Push Buttons on a Nongraphical Display Station without Underline Capability

On display stations in configuration F from Table 15 on page 139, the push buttons look like this:


< F1=Help > <F3=Exit > <F4=Prompt>

RV3W074-0

Figure 70. Push Buttons on a Display Station without Enhanced Interface

DDS for Push Buttons-Example

The DDS in Figure 71 creates the displays shown in “Push Buttons-Overview” on page 166.

```
|...+....1....+....2....+....3....+....4....+....5....+....6....+....7...
     A          R RECORD
     A
     A            PSHFLD1        2Y 0B 23  4PSHBTNFLD(*RSTCSR (*NUMCOL 3))
     A                                      PSHBTNCHC(1 'F1=>Help' HELP)
     A                                      PSHBTNCHC(2 &F3 CF03)
     A                                      CHCCTL(2 &CTL)
     A    02                                PSHBTNCHC(3 'F4=>Prompt' CF04)
     A                                      CHCAVAIL((*COLOR RED))
     A            F3            15A  P
     A            CTL            1Y 0H
```

Figure 71. DDS for Push Buttons-Example

Creating Push Buttons

A push button field is defined with the push button field (PSHBTNFLD) keyword. Each push button field must contain one or more push button choice (PSHBTNCHC) keywords.

Push buttons can be displayed vertically or horizontally (the default). The *NUMROW value on the PSHBTNFLD keyword specifies the number of rows to use when displaying the push buttons vertically. The *NUMCOL value on the PSHBTNFLD keyword specifies the number of columns to use when displaying the push buttons horizontally. When no parameters are specified on the PSHBTNFLD keyword, the push buttons are displayed in as many columns that fit on one line.

Use the choice control (CHCCTL) keyword to control the availability of the individual push buttons. Use the choice available (CHCAVAIL) and choice unavailable (CHCUNAVAIL) keywords to control the color or attribute of the individual push buttons.

Specify a key for each push button. In the following example, the CF04 key is returned when the F4=Prompt push button is selected:

```
PSHBTNCHC(3 'F4=>Prompt' CF04)
```

If you do not specify a key, Enter is returned when the push button is selected.

The key defined in the push button choice is automatically enabled for the record that contains the push button field.

The field that contains the push button contains the number of the push button choice that is selected. If the user presses the key associated with a push button (instead of selecting the push button itself), the number of the push button is not returned. Zero is returned if no choice is made.

Help can be specified for each push button.

Controlling the Availability of Choices

You can control the availability of the choices in a selection field and push button choices using the CHCCTL keyword. For information on controlling the availability of choices in selection lists, see "Creating Selection Lists" on page 161. Specify the name of a hidden field on the CHCCTL keyword. Your program can set a value in this field to specify whether the choice is available, unavailable, or selected (multiple-choice selection fields only). Figure 72 describes the control values and their meanings.

Control Value	Sets Choice to
0	Available (or not selected)
1	Selected
2	Unavailable. Cannot place cursor on choice unless help for choice is available.
3	Unavailable. Placing cursor on choice is allowed.
4	Unavailable. Cannot place cursor on choice even if help for choice is available.

Figure 72. Control Values for the CHCCTL Keyword

Note: The cursor restrictions in Figure 72 apply only to displays attached to a controller that supports an enhanced interface for nonprogrammable work stations.

Value 1 is ignored for single-choice selection fields, because default selection is done by setting the selection in the single-choice selection field itself. It is also ignored for push buttons. Value 1 is used for multiple-choice selection fields to let more than one choice default to the selected choice.

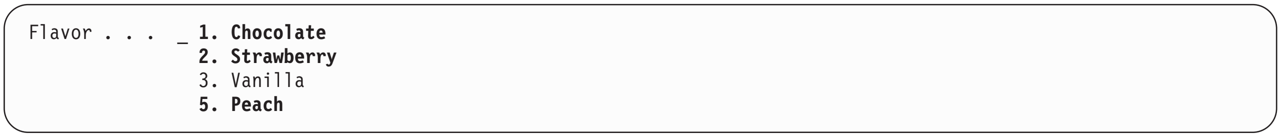
You can also specify a message to be displayed if the user selects an unavailable choice. If you do not specify a message, the system displays a default message.

Figure 73 is an example of the DDS to control the availability of choices showing a single-choice selection field in an ordinary record:

```
|...+....1....+....2....+....3....+....4....+....5....+....6....+....7...
     A          R RECORD
     A                                  2  2'Flavor . . . '
     A            F1             2Y 0B  2 16SNGCHCFLD
     A                                      CHOICE(1 '>Chocolate  ')
     A                                      CHCCTL(1 &CTLCHOC MSG1112 QUSER/A)
     A    01                                CHOICE(2 '>Strawberry ')
     A                                      CHCCTL(2 &CTLSTRA &MSG &LIB/&MSGF)
     A                                      CHOICE(3 '>Vanilla    ')
     A                                      CHCCTL(3 &CTLVANI);
     A                                      CHOICE(5 '>Peach      ')
     A            CTLCHOC        1Y 0H
     A            CTLSTRA        1Y 0H
     A            CTLVANI        1Y 0H
     A            MSG            7A  P
     A            LIB           10A  P
     A            MSGF          10A  P
```

Figure 73. DDS to Control the Availability of Choices

On display stations in configuration C from Table 15 on page 139 and configurations D, E, and F from Table 16 on page 140, the single-choice selection field looks like this:



```
Flavor . . .  _ 1. Chocolate
                2. Strawberry
                3. Vanilla
                5. Peach
```

The choice text, *Vanilla*, appears gray to indicate it is unavailable.

Figure 74. Single-Choice Selection Field with an Unavailable Choice. Assume that at run time, CTLCHOC and CTLSTRA are set to 0 (available) and CTLVANI is set to 2 (unavailable).

Auto-Selection in Single-Choice Selection Fields

The **auto-selection** function allows a user to select a choice in a single-choice selection field by placing the cursor on the choice and pressing Enter. It is not necessary to select the choice by 1) typing the choice number, 2) placing the cursor on the choice and pressing the spacebar, or 3) placing the pointer on the choice and pressing the left mouse button. This is the default for single-choice selection fields in a pull-down menu. The default for single-choice selection fields that are not in a pull-down menu is manual selection. The *AUTOSLT parameter on the SNGCHCFLD keyword indicates that the choice should be automatically selected when Enter is pressed. The *NOAUTOSLT parameter indicates that the choice must be manually selected. The *AUTOSLTENH parameter indicates that auto-selection is only in effect for displays attached to a controller that supports an enhanced interface for nonprogrammable workstations.

Auto-Enter in Single-Choice Selection Fields

The **auto-enter** function allows a selected choice to be returned to the program when the choice is selected. It is not necessary to press Enter to return the choice. The *AUTOENT parameter indicates that the Enter key will be returned as soon as the choice is selected. The choice is automatically returned on all display stations except where a double-digit selection number is required for any of the choices. The *NOAUTOENT parameter indicates that the choice will not be returned until the user presses Enter after selecting the choice. *NOAUTOENT is the default. The *AUTOENTNN parameter indicates that the choice will be returned as soon as the choice is selected only if numeric selection of the choice is not required.

Defining Mnemonics

You can define mnemonics for these items:

- Menu-bar choices
- Selection field choices (single and multiple)
- Selection list choices (single and multiple)
- Push buttons

Define mnemonics using a greater-than (>) character. To identify a mnemonic, place the > character just before the character that you want to be the mnemonic. The > character is not counted as part of the text length. If you want to use the > character as a character in the text rather than as a pointer to the mnemonic, you must use two > characters consecutively. If you specify the > character as the last character in the text, the > character appears as part of the text and no mnemonic appears. Figure 75 is an example.

Keyword Specification	Text Appears as
MNUBARCHC(1 PULLFILE ’>File’)	File
MNUBARCHC(2 PULLFIN ’F>inish’)	Finish
CHOICE(1 ’Save >As...’)	Save As...
CHOICE(2 ’X >= 1’)	X = 1
CHOICE(3 ’X >>>= 1’)	X >= 1

You cannot specify the > character as a mnemonic.

Figure 75. Examples of Valid DDS for Mnemonics

Note: The characters that appear highlighted in this example will be underlined on your display.

The mnemonic cannot be a blank. Only one mnemonic may be specified in the choice text. If more than one mnemonic is specified, only the first mnemonic is selectable. Figure 76 is an example of incorrect coding.

Keyword Specification	Error
MNUBARCHC(1 PULLFILE ’>File’)	Same mnemonic specified for
MNUBARCHC(2 PULLFIN ’>finish’)	more than one choice.
CHOICE(1 ’S>ave >As...’)	Two mnemonic characters
CHOICE(4 ’X >>>= >1’)	specified.

Figure 76. Examples of DDS Not Valid for Mnemonics

You can select a mnemonic by typing either the uppercase or lowercase mnemonic character. This is true for all languages. The system uses the monocase rules for the appropriate language to identify the mnemonic you typed. Double-byte characters cannot be mnemonics.

Because the system does not support both mnemonic and numeric selection for a field, mnemonics work only when choices are not displayed using numbers.

To determine which configurations support mnemonics, see Table 15 on page 139.

Defining Choice Colors and Attributes

Normally, the system uses the Common User Access® (CUA) default colors and display attributes when displaying menu-bar choices, selection-field choices, selection list choices, and push button choices. If you do not want to use these defaults, you can define the colors and attributes to be used for most choices using the CHCAVAIL, CHCUNAVAIL, and CHCSLT keywords. Table 19 shows which keywords can be used with each graphical item.

Table 19. Keywords Used to Define Colors and Display Attributes

	DDS Keyword		
Graphical Item	**CHCAVAIL**	**CHCUNAVAIL**	**CHCSLT**
Menu-Bar Choice	X		X
Selection Fields	X	X	X
Push Buttons	X	X	
Single Choice Selection List	X	X	X
Multiple Choice Selection List	X	X	X

The selection cursor on display stations attached to a controller that supports an enhanced interface for nonprogrammable work stations uses the reverse image form of the colors and attributes you specify for each of the choice states. For example, if you specify pink (normal display) for the available choices, the selection cursor is reverse image pink when it is located on an available choice. If you specify pink reverse image for the available choices, the selection cursor is pink (normal display). Likewise, if you do not specify CHCAVAIL, CHCUNAVAIL, or CHCSLT and use the CUA default colors and attributes, the selection cursor is the reverse image of those colors and attributes.

You can use only the CHCAVAIL keyword and the CHCSLT keyword for menu bars because menu-bar choices are either available or selected. For selection fields, use only the CHCAVAIL keyword and the CHCUNAVAIL keyword when you are using selection characters (for example, numbers or radio buttons). The CHCSLT keyword is ignored in these cases. However, you can use CHCSLT for selection fields in a pull-down menu on graphical display stations or character-based graphical display stations when you have specified that the pull-down menu should not contain selection indicators (PULLDOWN(*NOSLTIND) specified).

You can use only the CHCAVAIL keyword and the CHCUNAVAIL keyword for push buttons because push button choices are either available or unavailable.

The display-attribute (*DSPATR) parameter specifies the way in which the separator characters display. The parameter is expressed in this form:

```
(*DSPATR value1 <value2 <value3...>>)
```

The valid values for the display attributes are:

Value	Meaning
BL	Blink
CS	Column separator
HI	High intensity
ND	Nondisplay
RI	Reverse image
UL	Underline

The default display attribute for unavailable choices in a selection field on monochrome display stations is normal (or low) intensity. Also, the first character of an unavailable choice on a monochrome display station is overwritten with an asterisk (*).

Display attributes CS, HI, and BL can cause fields on 5292, 3179, and 3197 Model C1 and C2 display stations to appear as color fields. Display attributes HI, RI, and UL cause a separator line not to be displayed. For more information, see the CHCAVAIL keyword, the CHCUNAVAIL keyword, and the CHCSLT keyword in **DDS** topic in the iSeries Information Center.

In Figure 77, the choices in the menu bar are displayed in white on color display stations and in high intensity on monochrome display stations. When a menu-bar choice is selected, it is displayed in green on color display stations and reverts to normal (not high intensity) display on monochrome display stations.

```
|...+....1....+....2....+....3....+....4....+....5....+....6....+....7...
     A          R MENUBAR                   MNUBAR
     A            MNUFLD         2Y 0B  1  2
     A                                      MNUBARCHC(1 PULLFILE +
     A                                                '>File        ')
     A  02                                  MNUBARCHC(2 PULLEDIT +
     A                                                &EDITTXT)
     A                                      MNUBARCHC(3 PULLVIEW +
     A                                                '>View        ')
     A  04                                  MNUBARCHC(4 PULLOPT +
     A                                                '>Options       ')
     A                                      MNUBARCHC(5 PULLHELP +
     A                                                '>Help          ')
     A                                      MNUBARSEP((*COLOR WHT))
     A                                      CHCAVAIL((*COLOR WHT) (*DSPATR HI))
     A                                      CHCSLT((*COLOR GRN))
     A            EDITTXT       20A  P
                      .
                      .
                      .
```

Figure 77. DDS Using CHCAVAIL and CHCSLT for Menu-Bar Choices

In Figure 78 on page 173, the available selection-field choices are displayed in pink on color display stations and in high intensity on monochrome display stations. The unavailable selection-field choices are displayed in turquoise on color display stations. On monochrome display stations, the unavailable choices are displayed with normal (or low) intensity, and the first character in the choices is overwritten with an asterisk (*).

```
|...+....1....+....2....+....3....+....4....+....5....+....6....+....7...
     A            F1             2Y 0B  1  1SNGCHCFLD
     A  01                                  CHOICE(1 '>Chocolate  ')
     A                                      CHOICE(2 '>Strawberry ')
     A                                      CHOICE(3 '>Vanilla    ')
     A                                      CHCCTL(1 &CTLCHOC MSG1112 QUSER/A)
     A                                      CHCCTL(2 &CTLSTRA &MSG &LIB/&MSGF)
     A                                      CHCCTL(3 &CTLVANI)
     A                                      CHCAVAIL((*COLOR PNK) (*DSPATR HI))
     A                                      CHCUNAVAIL((*COLOR TRQ))
     A            CTLCHOC        1Y 0H
     A            CTLSTRA        1Y 0H
     A            CTLVANI        1Y 0H
     A            MSG            7A  P
     A            LIB           10A  P
     A            MSGF          10A  P
```

Figure 78. DDS Using CHCAVAIL and CHCUNAVAIL for Selection Fields

In Figure 79, the selection-field choice that is selected is displayed in yellow. If CHECK(ER) (automatic-enter) is specified on this field, control is returned immediately after selecting the choice.

```
|...+....1....+....2....+....3....+....4....+....5....+....6....+....7...
     A          R PULLDOWN                  PULLDOWN(*NOSLTIND)
     A            F1             2Y 0B  1  1SNGCHCFLD
     A  01                                  CHOICE(1 '>Undo       ')
     A                                      CHOICE(2 '>Mark       ')
     A                                      CHOICE(3 '>Copy       ')
     A                                      CHCCTL(1 &CTLUNDO MSG1112 QUSER/A)
     A                                      CHCCTL(2 &CTLMARK &MSG &LIB/&MSGF)
     A                                      CHCCTL(3 &CTLCOPY)
     A                                      CHCAVAIL((*COLOR PNK) (*DSPATR HI))
     A                                      CHCUNAVAIL((*COLOR TRQ))
     A                                      CHCSLT((*COLOR YLW))
     A            CTLUNDO        1Y 0H
     A            CTLMARK        1Y 0H
     A            CTLCOPY        1Y 0H
                   .
                   .
                   .
```

Figure 79. DDS Using CHCAVAIL, CHCUNAVAIL, and CHCSLT for Selection Fields

In Figure 80, the DDS source for a Single Choice Selection list is shown (the example does not show all the keywords necessary for a correct subfile definition). Available choices within the list will be displayed in yellow. Unavailable choices will be displayed in red. The selected choice will be displayed in green.

```
|...+....1....+....2....+....3....+....4....+....5....+....6....+....7...

     A          R SFLREC                    SFL
     A            CTLFLD         1Y 0H      SFLCHCCTL
     A          R SFLCTLRCD                 SFLCTL(SFLREC)
     A                                      SFLSNGCHC
     A                                         .
     A                                         .
     A                                         .
     A                                      CHCAVAIL((*COLOR YLW))
     A                                      CHCUNAVAIL((*COLOR RED))
     A                                      CHCSLT((*COLOR GRN))
```

Figure 80. DDS Using CHCAVAIL, CHCUNAVAIL, and CHCSLT for Single Choice Selection List Choices

In Figure 81, the DDS source for a Multiple Choice Selection list is shown (the example does not show all the keywords necessary for a correct subfile definition). Available choices within the list will be displayed in yellow. Unavailable choices will be displayed in red. Any selected choices will be displayed in green.

```
|...+....1....+....2....+....3....+....4....+....5....+....6....+....7...

     A          R SFLREC                    SFL
     A            CTLFLD         1Y 0H      SFLCHCCTL
     A          R SFLCTLRCD                 SFLCTL(SFLREC)
     A                                      SFLMLTCHC
     A                                         .
     A                                         .
     A                                         .
     A                                      CHCAVAIL((*COLOR YLW))
     A                                      CHCUNAVAIL((*COLOR RED))
     A                                      CHCSLT((*COLOR GRN))
```

Figure 81. DDS Using CHCAVAIL, CHCUNAVAIL, and CHCSLT for Multiple Choice Selection List Choices

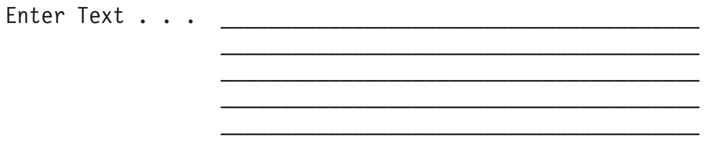
Continued-Entry Fields-Overview

A **continued-entry field** is a set of associated entry fields. Continued-entry fields are supported on displays attached to any controller. Controllers that support an enhanced interface for nonprogrammable work stations treat continued-entry fields as single-entry fields while data is being entered and edited in the fields.

Note: Controllers that do not support an enhanced interface for nonprogrammable work stations treat continued-entry fields as separate input fields. Insert and delete characters one segment at a time. When you reach the end of a segment, the cursor does not move automatically to the next segment.

Figure 82 illustrates the use of continued-entry fields to create a rectangular text entry field. Consider using this format to avoid using a single input field that wraps across multiple lines.

Note: The empty space at the end of the last line is still part of the continued-entry field. You cannot define another field in this space.



```
Enter Text . . .  ________________________________________
                  ________________________________________
                  ________________________________________
                  ________________________________________
                  ________________________________________
```

Figure 82. Continued-Entry Fields in Rectangular Arrangement

A continued-entry field allows a multiple-row entry field to be defined inside of a window or display.

The DDS for the field looks like this:

```
|...+....1....+....2....+....3....+....4....+....5....+....6....+....7...
     A            F1            90A  B  3  4CNTFLD(30)
```

The CNTFLD keyword tells the system that this is a continued-entry field and its parameter tells the system how wide the field should be. The system breaks the field into columns and uses as many lines as it needs to reach the specified width.

Specifying Word Wrap on Continued-Entry Fields

To specify the word wrap function for a continued-entry field, use the word wrap (WRDWRAP) keyword. This keyword can be used at the file, record, or field level.

Note: WRDWRAP cannot be used on DBCS continued-entry fields.

For more information about the WRDWRAP keyword, see “Specifying Word Wrap for Fields” on page 32.

DBCS Considerations with Continued-Entry Fields

DDS supports these DBCS data types:

J Only (only bracketed DBCS characters allowed)

E Either (either only SBCS or only bracketed DBCS characters allowed)

O Open (either SBCS or bracketed DBCS characters allowed - mixed)

G Graphic/Pure (only nonbracketed DBCS characters allowed)

These data types will have the following restrictions:

J The width of each continued-entry field segment must be an even number of at least 4 bytes.

E The width of each continued-entry field segment must be an even number of at least 4 bytes.

O The width of each continued-entry field segment must be at least 4 bytes wide.

G The width of each continued-entry field segment must be an even number of at least 4 bytes.

The length of the DBCS continued-entry field must account for the SO/SI character pairs that bracket the DBCS data on each segment of the continued-entry field. The following total field lengths are required to ensure the field data fits into DBCS continued-entry fields:

J or E (with DBCS data)
Data Length + (Number of segments - 1) * 2

O Data Length + (Number of segments - 1) * 3

G or E (with SBCS data)
Data Length

Note: The (Number of segments - 1) * 3 portion of the calculation in the second equation allows for the SO/SI sets that must bracket the DBCS data on the segments of the continued-entry field after the first segment. Additional consideration is made for the possibility that a NULL must be placed at the end of a segment wherever a DBCS character would be split.

How DBCS Data is Returned for Continued-Entry Fields

If the field is a DBCS-only or DBCS-either (with DBCS data) field, the following data is not returned to the application

- The extra SO characters at the start of the middle and last segments
- The extra SI characters at the end of the first and middle segments

If the field is a DBCS-open field, the following are removed before the field is returned to the application:

- All single byte subfields at the end of a segment which consist only of one null or are empty. This is to remove any SO/SI pairs that most likely have been automatically inserted as a result of double byte data falling on a segment boundary. Removing the extra SI/SO characters occurs for all hardware regardless of whether the controller supports an enhanced interface for nonprogrammable work stations.
- Single byte nulls that end a segment if the following is true:

- – The number of nulls is three or less
- – The previous character is not an SI character or null
- – The next segment begins with an SO character

This is to remove any nulls that most likely have automatically been inserted as a result of double byte data falling on a segment boundary.

Keyboard Functions with Continued-Entry Fields

The system processes local keyboard functions specified within the continued-entry fields on a display station attached to a controller that supports an enhanced interface for nonprogrammable work stations by the following definitions:

Character data

In replace mode, there is no unique character data processing. When character data is entered in the last character position of the first or one of the middle fields in the set, the cursor moves to the first character position of the next field in the set. When character data is entered in the last character position of the last field in the continued fields, forward field-exit processing is performed (see “Forward Field-Exit Processing” on page 179).

In insert mode and cursor direction matches field direction, the following actions occur when a character data key is pressed:

- The null in the last character position of the last field in the continued-entry fields is deleted. If there is no null in the last position of this field, operator error 0012 is posted.
- All field data within the continued-entry fields at (and logically following) the current cursor location are shifted one position. Each data character in the last character position of the current and remaining fields (except the last) in the set is shifted to the first character position of the following field.
- The data character entered is written at the cursor location.
- The cursor advances to the next cursor position.
- For DBCS-only, DBCS-either (with DBCS data), and DBCS-pure fields, the two nulls in the last two character positions (before the SI for DBCS-only or DBCS-either fields) of the last segment of the continued field are deleted. If there are not nulls in the last two positions of that segment, operator error 0012 is posted. Otherwise, all field data (not including the SO or SI characters) within the set of field segments at and logically following the current cursor location is shifted two positions in the cursor direction. Each double byte character in the last two character positions of the current and remaining segments (except the last) is moved to the first two character positions of the following segment. The DBCS character entered is written at the cursor location, and the cursor advances to the next double byte cursor position.
- For DBCS-open fields the data in the continued field segments, at and to the right of the cursor, is copied into one continuous buffer. The inserted character is placed at the start of the buffer. All single byte nulls are removed from the buffer, and the data is shifted toward the beginning of the buffer. All adjacent SI/SO character pairs (that is, empty single byte subfields) are also removed from the buffer. The data is again shifted toward the beginning of the buffer. The data is then placed back into the continued field one character at a time, according to the algorithm for writing data into a DBCS-open continued field. The remaining character positions are replaced with nulls. The cursor also advances to the next character position. If all the data in the buffer cannot fit into the continued field, operator error 0012 is posted, and the field data and cursor position are not changed.

In insert mode and cursor direction does not match field direction, the insert takes place within a subfield. The insert is the same as if the field and cursor direction matched, but the insert is performed within the subfield. The extent of a subfield is defined as follows:

- When the cursor direction is right-to-left, the subfield extends from the cursor to the first null logically following the cursor. If there is no such null, the subfield includes all positions logically following the cursor.

- When the cursor direction is left-to-right, the subfield extends from the cursor to the first null logically following the cursor. If there is no such null, the subfield includes all positions logically following the cursor.

Field Mark

Processed the same as character data.

Automatic Shape Determination (ASD) Processing

For Arabic, ASD occurs if the cursor direction is right-to-left.

Delete

If the delete key is pressed within a continued-entry field and the cursor direction matches the field direction, the following actions occur:

- All field data within the continued-entry fields logically following the current cursor location is shifted toward the cursor one position. Each data character in the first character position of the remaining fields in the set is shifted to the last character position of the preceding field.
- A null is written in the last character position in the continued-entry fields.
- For DBCS-only, DBCS-either (with DBCS data), and DBCS-pure continued fields, all field data (not including the SO and SI characters) within the set of field segments logically following the current cursor location is shifted toward the cursor two positions. Each DBCS character in the first double byte character position of the remaining segments is moved to the last double byte character position of the preceding segment. A double byte null is written in the last double byte character position of the last segment.
- For DBCS-open fields, the data in the set of field segments, at and to the right of the cursor, is copied into one continuous buffer. The deleted character or subfield is removed from the start of the buffer. In addition, all single byte nulls are removed from the buffer, and the data is shifted toward the beginning of the buffer. All adjacent SI/SO character pairs (that is, empty single byte subfields) are also removed from the buffer. The data is again shifted toward the beginning of the buffer. The remaining data, then, is placed back into the continued field one character at a time, according to the algorithm for writing data into a DBCS-open continued field. The remaining character positions are replaced with nulls.

If the delete key is pressed within a continued field when the cursor direction and field direction do not match, the delete is performed within a subfield. The definition for subfields is the same as for the insert key.

Erase EOF

All field positions at (and logically following) the current cursor location within the continued-entry fields are nulled. In DBCS-only, DBCS-either (with DBCS data), and DBCS-pure fields, the SO and SI characters are not nulled out. In DBCS-open fields, an SI character may additionally be written at the current cursor location if the erase began in a double byte subfield.

Erase Input

All field positions of all changed fields are nulled. This includes all continued-entry field segments if any continued-entry field segment has been changed. In DBCS-only, DBCS-either (with DBCS data), and DBCS-pure fields, the SO and SI characters are not nulled out.

Reverse

The cursor direction is reversed. If the preceding keystroke was not a cursor movement key, the cursor is repositioned to the new first character position of the current segment.

Close

The close key operates on a single continued field segment. All embedded nulls are removed. The cursor direction is set to the field direction. The remaining characters are shifted to begin at the first character position of the continued field segment. The remainder of the segment is padded with nulls and the cursor is placed logically following the last non-null character.

Field Exit

Pressing the Field Exit key within a continued-entry field causes the following actions to occur:

- All field data within the continued-entry fields at (and logically following) the current cursor location are nulled. In DBCS-only, DBCS-either (with DBCS data), and DBCS-pure fields, the SO and SI characters are not nulled out. In DBCS-open fields, an SI character may additionally be written at the current cursor location if the nulling began in a double byte subfield.
- Forward field-exit processing is performed (see "Forward Field-Exit Processing" on page 179).

Field Plus

Processed the same as Field Exit.

Field Minus

Not allowed. Operator error 0016 is posted.

Dup

Pressing the Dup key within a continued-entry field causes the following actions to occur:

- All field data within the continued-entry fields at (and logically following) the current cursor location are set to the Dup character (1C). In DBCS-only, DBCS-either (with DBCS data), and DBCS-pure fields, the SO and SI characters are not replaced with the Dup character. In DBCS-open fields, the cursor must be on the very first character (whether it is a single byte or double byte character) when the Dup key is pressed. Every character of every segment is replaced with the Dup character ('1C') including all SO and SI characters. If the cursor is not on the first character, operator error 0019 is posted.
- Forward field-exit processing is performed (see "Forward Field-Exit Processing" on page 179).

Kanji

The Kanji key causes the following actions to occur when pressed within a DBCS-either continued field:

- If the cursor is not at the first field position (when in single byte mode) of the first segment, or at the second field position (when in double byte mode) of the first segment, operator error 0062 is posted.
- Otherwise, if the field is currently in double byte mode, it is placed into single byte mode by replacing every character position of every segment with nulls. The cursor is also placed at the first field position of the first segment.
- Otherwise, if the field is currently in single byte mode, it is placed into double byte mode by replacing every character position of every segment with nulls, and writing SO and SI characters at the start and end of each field segment respectively. The cursor is also placed in the first segment immediately following the shift out character.

In DBCS-open continued fields, the Kanji key inserts either a SO/SI character pair, or an SI/SO character pair as is currently done for non-continued open fields. For continued fields, this insert is performed using the same algorithm that characters are inserted into an DBCS-open continued field. However, when a SI/SO pair is inserted, the empty single byte subfield that is created is not immediately removed. The cursor is also placed under the second shift character and the keyboard goes into insert mode.

Character Backspace

Pressing the Character Backspace key in the first position (or first DBCS character in any DBCS-pure, DBCS-only, or DBCS-either field) of the first segment, moves the cursor to the last position of the previous field. (The previous field could be a continued-entry field. The cursor moves to the last position of the last segment which could be further down on the display.) If the resulting cursor position is in a DBCS-open, DBCS-only, or DBCS-either field, any SI character at the last position is skipped. Pressing the Character Backspace key in the first position in a segment (or first DBCS character in DBCS-pure, DBCS-only, or DBCS-either fields in double byte mode) other than the first segment, moves the cursor to the last position of the previous segment. If that resulting position is on a DBCS-only or DBCS-either field SI character, the SI character is skipped. In addition, within DBCS-open fields, single byte subfields at the end of a non-last segment are skipped by character backspace if they consist only of 1 null character. This is to skip nulls that most likely have been automatically inserted when splitting DBCS data on segment boundaries.

Character Advance

Pressing the Character Advance key in the last position (or last double byte character in DBCS-pure, DBCS-only, or DBCS-either fields in double byte mode) of the last segment, moves the cursor to the first position of the next field. Pressing the Character Advance key in the last position (or last double byte character in DBCS-pure, DBCS-only, or DBCS-either fields in double byte mode) of a segment other than the last segment, moves the cursor to the first position of the next segment. If the resulting position is on a DBCS-only or DBCS-either field SO character, the SO character is skipped. In DBCS-open fields, single byte subfields at the end of a segment are skipped if they consist of only one null. This is to skip nulls that most likely have been automatically inserted when splitting DBCS data on segment boundaries.

New Line

Pressing the New Line key generally moves the cursor to the next position on the display that allows a cursor. If the cursor is in a continued-entry field and an additional continued-entry field segment is on the next row or a subsequent row, the cursor moves to the first position of that segment. If the continued-entry field is also a highlighted field with an invisible text cursor, then pressing the New Line key exits the continued-entry field. If the resulting position is in a new DBCS-open, DBCS-only, or DBCS-either field segment of the same field, any S0 character at the resulting cursor location is skipped. When the cursor moves into a continued-entry field because the New Line key was pressed, the cursor is always positioned in the first position of the first segment. Pressing the New Line key never moves the cursor into a middle or last continued-entry field segment.

Field Advance

Field Advance performs forward field-exit processing (see “Forward Field-Exit Processing”).

Field Backspace

When pressed while the cursor is not in the first position (or first DBCS character in a DBCS-pure, DBCS-only, or DBCS-either field) of the first segment, the cursor moves to the first position (or first DBCS character in a DBCS-pure, DBCS-only, or DBCS-either field) of the first segment. Otherwise, Field Backspace performs backward field-exit processing (see “Backward Field-Exit Processing” on page 180).

Forward Field-Exit Processing

The system does not validate field data when the cursor exits continued-entry fields because mandatory fill and self-check functions are not supported.

If the continued-entry fields are specified as automatic-enter or forward-edge trigger, the system performs automatic-enter or forward-edge trigger processing for the last position of the last segment.

If cursor progression is specified on the first continued-entry field segment, the cursor moves to the cursor-progression target field when it exits any of the continued-entry field segments.

If cursor progression is not specified on the first continued-entry field segment, and the cursor exits the continued-entry field in the forward direction, the cursor skips any subsequent segments of the continued-entry field and moves to the next nonprotected field. The next nonprotected field is determined by exiting the first continued field segment, independent of which segment contained the cursor. If the resulting position is in a DBCS-open, DBCS-only or DBCS-either field, any SO character at the first position is skipped. The cursor is placed at the second position.

Note: If a continued-entry field is also defined as a highlighted field, the system restores the leading-field attribute of each segment when the cursor exits the field.

Backward Field-Exit Processing

When exiting the field backward, the cursor skips any previous segments of the continued-entry field and moves to the previous nonprotected field. Cursor progression may cause the cursor to move to a different field. The previous nonprotected field is determined by exiting the first continued field segment, independent of which segment contained the cursor. If the resulting position is in a DBCS-open, DBCS-only or DBCS-either field, any SO character at the first position is skipped. The cursor is placed at the second position.

Note: DBCS support within continued-entry fields are available for displays attached to any controller. However, the keyboard functions are only available for displays connected to a controller that supports an enhanced interface for nonprogrammable work stations.

How the Menu Bar Interacts with the Application

The MNUBARDSP keyword is used to display a menu bar. The MNUBARDSP keyword can be used on an application record (the record that defines the application display) or on a menu-bar record. Option indicators can be used on the MNUBARDSP keyword to control when the menu bar is displayed. If MNUBARDSP is used on the application record, several optioned MNUBARDSP keywords can be specified so that the application can display different menu bars for the same record. If more than one MNUBARDSP keyword is in effect, the system uses the first one.

Defining the MNUBARDSP Keyword on the Application Record

When the MNUBARDSP keyword is used on the record that defines the application display, the system handles all menu-bar operations for the application. The system returns the number of the menu-bar choice selected in a hidden field that is specified on the MNUBARDSP keyword and defined in the application record. If no menu-bar choice is selected, 0 is returned in the hidden field.

```
|...+....1....+....2....+....3....+....4....+....5....+....6....+....7...
     A                                      HELP ALTHELP CF03
     A                                      MNUBARSW(CA10) MNUCNL(CA12)
     A          R MENUBAR                   MNUBAR
     A            MNUFLD         2Y 0B  1  2
     A                                      MNUBARCHC(1 PULLFILE +
     A                                                '>File        ')
     A  02                                  MNUBARCHC(2 PULLEDIT +
     A                                                &EDITTXT)
     A                                      MNUBARCHC(3 PULLVIEW +
     A                                                '>View        ')
     A  04                                  MNUBARCHC(4 PULLOPT +
     A                                                '>Options      ')
     A                                      MNUBARCHC(5 PULLHELP +
     A                                                '>Help          ')
                  :
                  :
     A          R APPSCR                    MNUBARDSP(MENUBAR &MNUCHOICE)
     A            FIELD1        10A  B 10 12
     A            FIELD2         5S 0B 14 12
     A                                 24  1'F1=Help   F3=Exit   +
     A                                       F10=Actions   F12=Cancel   '
     A            MNUCHOICE      2Y 0H
```

Figure 83. DDS Using MNUBARDSP on the Application Record

The DDS in Figure 83 causes the following to happen:

1. The application writes any pull-down menu records or menu-bar records that have output to be supplied by the application.
2. The application does a write-read operation to the record with the MNUBARDSP keyword.

3. The system displays the menu bar identified by the MNUBARDSP keyword and handles all interaction between the menu bar and the pull-down menu.
4. If the user selects a menu-bar choice and enters input in the pull-down menu, the system returns the number of the menu-bar choice selected in the choice-hidden field specified on MNUBARDSP.

The system displays the menu-bar record first and then performs the write-read operation to the application record. The active function keys and command keys are those defined on the application record and not those defined on the menu-bar record. If you want to use the MNUCNL keyword or the MNUBARSW keyword, define them at the file level or on the application record.

Ordinarily, writing a record to the display without the OVERLAY keyword causes the entire display to be erased before the record is displayed. The system displays the menu-bar record followed by the application record as if they were logically one record. The system automatically prevents the application record from erasing the menu-bar record. In Figure 83 on page 180, when the system writes the application record, it clears the entire display except for the menu-bar record. The menu-bar record is always processed as though it contains the OVERLAY keyword regardless of any other specifications. For example, if the CLRL keyword is specified on the menu-bar record, it is not used when the menu-bar record is processed. The OVERLAY keyword and the CLRL keyword are processed normally for the application record; however, the menu-bar record is not cleared.

Defining the MNUBARDSP Keyword on the Menu-Bar Record

If you use the MNUBARDSP keyword on the record containing the menu bar, the application controls when the menu bar is displayed. The system continues to handle any pull-down menu interaction that takes place, and returns the number of the menu-bar choice selected in the menu-bar field itself. If no menu-bar choice was selected, 0 is returned in the menu-bar field. The application must both write the menu-bar record to display the menu bar and read the menu-bar record to determine what choice, if any, was selected.

```
|...+....1....+....2....+....3....+....4....+....5....+....6....+....7...
     A                                      HELP ALTHELP CF03
     A                                      MNUBARSW(CA10) MNUCNL(CA12)
     A          R MENUBAR                   MNUBAR OVERLAY
     A  01                                  MNUBARDSP
     A            MNUFLD         2Y 0B  1  2
     A                                      MNUBARCHC(1 PULLFILE +
     A                                                '>File        ')
     A  02                                  MNUBARCHC(2 PULLEDIT +
     A                                                &EDITTXT)
     A                                      MNUBARCHC(3 PULLVIEW +
     A                                                '>View        ')
     A  04                                  MNUBARCHC(4 PULLOPT +
     A                                                '>Options      ')
     A                                      MNUBARCHC(5 PULLHELP +
     A                                                '>Help           ')
     A            EDITTXT       20A  P
                  :
                  :
     A          R APPSCR                    OVERLAY
     A            FIELD1        10A  B 10 12
     A            FIELD2         5S 0B 14 12
     A                                 24  1'F1=Help   F3=Exit   +
     A                                       F10=Actions   F12=Cancel   '
```

Figure 84. DDS Using MNUBARDSP on the Menu-Bar Record

Following is one scenario using the DDS in Figure 84.

1. The application writes the menu-bar record, with the MNUBARDSP keyword active, to display the menu bar.

2. The application does a write-read operation to the application record.
3. The system handles any interaction between the menu bar and the pull-down menu.
4. On input, the application receives the application record.
5. The application reads the menu-bar record to determine which menu-bar choice, if any, was selected.

The only active command keys and function keys are those defined on the application record and not those defined on the menu-bar record. If you want to use the MNUCNL keyword or the MNUBARSW keyword, define them at the file level or on the application record.

Following is another scenario using the DDS in Figure 84 on page 181.

1. The application writes the application record.
2. The application performs a write-read operation to the menu-bar record, with the MNUBARDSP keyword active, to display the menu bar.
3. The system handles any interaction between the menu bar and the pull-down menu.
4. On input, the application receives the menu-bar record and determines which menu-bar choice, if any, was selected.
5. The application can also read the application record to receive any input entered on the display.

The only command keys and function keys that are valid are those defined on the menu-bar record. If you want to use the MNUCNL keyword or the MNUBARSW keyword, define them at the file level or on the menu-bar record.

Receiving Input from the Pull-Down Menus

If a pull-down menu record contains output data, it must be written before writing the menu-bar record. For example, option indicators may be set or output fields may be filled in. These pull-down menu records are not displayed when written; the system processes and saves the record output until the menu bar is displayed.

A menu-bar record may also be written without being displayed. If the system attempts to write a pull-down menu record or a menu-bar record (without the MNUBARDSP keyword optioned on) while the corresponding menu bar is displayed, an error occurs and the record is not written.

When a valid attention identifier (AID) key (other than the cancel key and the menu-bar switch key) is pressed when a pull-down menu is displayed, control is returned to the application and input may be received from the pull-down menu. The valid AID keys that return input are the Enter key and any CFxx keys that are defined on the pull-down menu record. (A CAxx key returns control to the application, but does not return input.) Keys that are defined for the background display are not valid unless they are also defined on the pull-down menu record. Therefore, define the background keys and the pull-down menu keys once at the file level.

Once a valid AID key (one that returns input) is pressed for a pull-down menu, the application receives input for the record being read (either the application record or the menu-bar record). By looking at the menu-bar choice number that is returned in this record, the application can determine which pull-down menu record has input. The application then must read that pull-down menu record. No I/O operation is done to the display; the input from the pull-down menu is returned to the application and the pull-down menu remains displayed.

Receiving Input from Pull-Down Menus Using the Pull-Down Input Parameter

When one or more of the pull-down menu records contain only one single-choice selection field, you can use the pull-down input (PULLINPUT) parameter on the MNUBARDSP keyword. This lets you receive the single-selection field choice along with the menu-bar choice, instead of reading the pull-down menu record to receive the single-selection field choice.

Figure 85 illustrates the use of the PULLINPUT parameter on the MNUBARDSP keyword.

```
|...+....1....+....2....+....3....+....4....+....5....+....6....+....7...
     A                                      HELP ALTHELP CF03
     A                                      MNUBARSW(CA10) MNUCNL(CA12)
     A          R MENUBAR                   MNUBAR
     A            MNUFLD         2Y 0B  1  2
     A                                      MNUBARCHC(1 PULLFILE +
     A                                                '>File        ')
     A                                      MNUBARCHC(2 PULLEDIT +
     A                                                '>Edit        ')
     A                                      MNUBARCHC(3 PULLVIEW +
     A                                                '>View        ')
     A          R PULLFILE                  PULLDOWN
     A                                  2  1'File name . . . '
     A            FNAME         10A  I  2 18
     A          R PULLEDIT                  PULLDOWN
     A            F1             2Y 0B  1  1SNGCHCFLD
     A                                      CHOICE(1 '>Copy       ')
     A                                      CHOICE(2 '>Delete     ')
     A          R PULLVIEW                  PULLDOWN
     A            F1             2Y 0B  1  1SNGCHCFLD
     A                                      CHOICE(1 '>All        ')
     A                                      CHOICE(2 '>Some...    ')
     A            F2             2Y 0B  4  1SNGCHCFLD
     A                                      CHOICE(1 'By >date    ')
     A                                      CHOICE(2 'By >subject ')
     A          R APPSCR                    MNUBARDSP(MENUBAR &MNUCHOICE +
     A                                                &PULLINPUT)
     A            FIELD1        10A  B 10 12
     A            FIELD2         5S 0B 14 12
     A                                 24  1'F1=Help   F3=Exit   +
     A                                       F10=Actions   F12=Cancel   '
     A            MNUCHOICE      2Y 0H
     A            PULLINPUT      2S 0H
```

Figure 85. DDS for Pull-Down Input (PULLINPUT) Parameter

When the PULLINPUT parameter is specified on the MNUBARDSP keyword, one of the following values will be returned to the application:

Note: The value is returned in the hidden field (PULLINPUT in Figure 85) you have defined in the record with the MNUBARDSP keyword.

PULLINPUT Contents	Meaning
0	No selection made.
n	Choice n in the pull-down menu was selected.
-1	Pull-down menu record contains something other than one single-choice selection field. You must read the pull-down menu record to receive its contents.

Table 20 shows the values that are returned in the MNUCHOICE field and the PULLINPUT field using the DDS in Figure 85.

Table 20. Values Returned in MNUCHOICE and PULLINPUT

	Value Returned in Appropriate Field	
Menu-Bar Choice Selected	**MNUCHOICE**	**PULLINPUT**
File	1	-1

Table 20. Values Returned in MNUCHOICE and PULLINPUT (continued)

	Value Returned in Appropriate Field	
Menu-Bar Choice Selected	**MNUCHOICE**	**PULLINPUT**
Edit (no selection made in pull-down menu)	2	0
Edit (Copy choice in pull-down menu selected)	2	1
View	3	-1

Removing a Pull-Down Menu after Receiving Input

After the application receives input from a pull-down menu, it may remove the pull-down menu under certain conditions. For example, the application removes the pull-down menu when the application writes or reads (1) a non-window record (in the same display file) or (2) a window record (in the same display file) that contains the RMVWDW keyword. The pull-down menu remains on the display if the application writes or reads a window record (without RMVWDW). This allows the application, UIM help, or application help to write a help window to the display without removing the pull-down menu.

To remove a pull-down menu and present another non-window application display, the application writes the appropriate application record, and the pull-down menu is removed.

To remove the pull-down menu and keep the current application display (perhaps updated), the application performs another write-read (or read) operation to the current (or changed) application record.

To remove the pull-down menu and display a window, the application writes a window with the RMVWDW keyword specified.

To remove the pull-down menu and call another program, the application writes to a dummy record and then calls the other program.

To leave the pull-down menu on the display and present a window (perhaps a help window), the application writes the window record (without RMVWDW).

A pull-down menu remains on the display while UIM help is displayed in a window. A pull-down menu also remains on the display while application help is displayed if the application help record is a window and does not have the RMVWDW keyword specified.

Updating a Pull-Down Menu before Displaying

You can enable your application to update a pull-down menu using the return-field parameter on the MNUBARCHC keyword. The pull-down menu is updated after a menu-bar choice is selected and before the pull-down menu is displayed. The return-field parameter is a hidden field that returns the number of the choice selected for the application to (1) determine that control was returned before the pull-down menu was displayed rather than because input was entered in the pull-down menu, and (2) determine which pull-down menu record to update and write.

```
|...+....1....+....2....+....3....+....4....+....5....+....6....+....7...
     A          R MENUBAR                  MNUBAR
     A            MNUFLD         2Y 0B  1  2
     A                                      MNUBARCHC(1 PULLFILE +
     A                                                '>File        ' &RTNFLD)
                  :
                  :
     A                                      MNUBARCHC(4 PULLOPT +
     A                                                '>Options       ' +
     A                                                &RTNFLD)
                  :
     A            RTNFLD         2Y 0H
                  :
     A          R PULLFILE
     A                                      PULLDOWN
     A            F1             2Y 0B  1  1SNGCHCFLD
                  :
                  :
     A          R PULLOPT
     A                                      PULLDOWN
     A            F1             2Y 0B  1  1SNGCHCFLD
                  :
                  :
```

Figure 86. DDS for Return-Field Parameter

For the DDS in Figure 86, if the user selects menu-bar choice 1 or 4, control is returned to the application with the choice number set in the RTNFLD field. The menu-bar field or the choice field in the application record contains 0, indicating no pull-down menu input was received. The application must read the menu-bar record to get the contents of the RTNFLD field. The application then updates the pull-down menu record for that choice and writes it. The application must read the menu-bar record or the application record to request the display. After control has been returned for updating the pull-down menu, the next record written must be the pull-down menu specified on the MNUBARCHC keyword. In this example, if choice 1 was selected, record PULLFILE must be written; if choice 4 was selected, record PULLOPT must be written. The system then displays the pull-down menu for the choice and resumes control of the menu bar and pull-down menu interaction. A read operation to a pull-down menu is not allowed until input has been received for the pull-down menu. A write-read operation is never allowed.

Defining Application Help

You can define application help for menu-bar choices, selection-field choices, named fields, constant fields, menu bars, and pull-down menus.

Defining Choice-Level Help

You can define help for menu-bar choices, single-choice selection fields, and multiple-choice selection fields using the HLPARA keyword. Figure 87 on page 186 is an example of the DDS coding for menu-bar choice help.

```
|...+....1....+....2....+....3....+....4....+....5....+....6....+....7...
     A                                      HELP
     A          R MENUBAR                   MNUBAR
     A          H                           HLPARA(*FLD MNUFLD 1)
     A                                      HLPRCD(FILEHLP LIB/FILE)
     A          H                           HLPARA(*FLD MNUFLD 2)
     A                                      HLPRCD(EDITHLP LIB/FILE)
     A          H                           HLPARA(*FLD MNUFLD 3)
     A                                      HLPRCD(VIEWHLP LIB/FILE)
     A          H                           HLPARA(*FLD MNUFLD 4)
     A                                      HLPRCD(OPTHLP LIB/FILE)
     A          H                           HLPARA(*FLD MNUFLD 5)
     A                                      HLPRCD(HLPHLP HLPLIB/HLPFILE)
     A            MNUFLD         2Y 0B  1  2
     A                                      MNUBARCHC(1 PULLFILE +
     A                                                '>File         ')
     A   02                                 MNUBARCHC(2 PULLEDIT +
     A                                                &EDITTXT)
     A                                      MNUBARCHC(3 PULLVIEW +
     A                                                '>View         ')
     A   04                                 MNUBARCHC(4 PULLOPT +
     A                                                '>Options        ')
     A                                      MNUBARCHC(5 PULLHELP +
     A                                                '>Help           ')
     A           EDITTXT        10   P
```

Figure 87. DDS for Menu-Bar Choice Help

Figure 88 is an example of the DDS coding for single-selection field choice help.

```
|...+....1....+....2....+....3....+....4....+....5....+....6....+....7...
     A                                      HELP
     A          R PULLEDIT                  CF04 CF06
     A                                      WDWBORDER((*CHAR '+-+||+-+'))
     A                                      PULLDOWN
     A          H                           HLPARA(*FLD F1 1)
     A                                      HLPRCD(UNDOHLP LIB/FILE)
     A          H                           HLPARA(*FLD F1 2)
     A                                      HLPRCD(MARKHLP LIB/FILE)
     A          H                           HLPARA(*FLD F1 3)
     A                                      HLPRCD(COPYHLP LIB/FILE)
     A            F1             2Y 0B  1  1SNGCHCFLD
     A                                      CHECK(ER)
     A     01                               CHOICE(1 '>Undo       ')
     A                                      CHOICE(2 '>Mark       ')
     A                                      CHOICE(3 '>Copy       ')
```

Figure 88. DDS for Single-Selection Field Choice Help

Use the *FLD special value on the HLPARA keyword to indicate that the help area is for a field. Following *FLD, specify the name of the field for which you are defining help. Following the name of the field, specify the number of the choice for which you are defining help.

The help area that you specify for a choice is the area that the choice text occupies (plus the attribute byte positions on either side). If compression occurs, the help area moves with the choice. When a choice is optioned off, the help specification for that choice is also optioned off because the help area for that choice does not exist.

As with any help specification, the text for choice-level help can be defined using DDS records (HLPRCD keyword) or UIM panel groups (HLPPNLGRP).

Help specifications must define help areas within the area encompassed by the menu-bar record or the pull-down record.

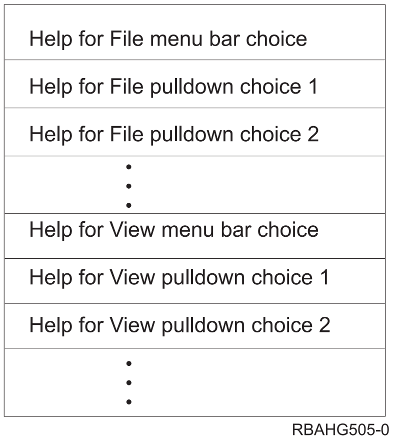
If you use UIM help, item-specific help for a menu-bar choice includes the help modules for the menu-bar choice and its pull-down menu. If you use DDS, item-specific help for the menu-bar choice is the help for the menu-bar choice. The help area for the menu-bar choice includes the text for the choice plus one space on each side of the text. The middle space between the choices is part of the extended help area.

Item-specific help for menu bars is displayed when the cursor is located in one of the following areas:

- A menu-bar choice (if there is help for that choice).
- Anywhere within an active pull-down record that does not include an active help area. When the cursor is on the pull-down menu border, item-specific help is displayed for the menu-bar choice item.

Item-specific help for a pull-down menu choice is displayed when the cursor is in an active help area in the pull-down menu. The help area for a pull-down menu choice starts on the first digit of the number and ends at the border of the selection field. Help for pull-down menus and menu bars are part of the extended help for the display. When the pull-down menu is displayed, all help areas are active except the ones that are overlapped by the pull-down menu. Help for overlapped areas can be viewed only in extended help. If a base display help area is partially overlapped by a pull-down menu, the part of the help area that is not overlapped is still active.

There are two help lists; one for the base display and one for the menu bar and its pull-down menus. The help list for the menu bar is created when the menu bar is displayed. The help list is destroyed when the menu bar is removed from the display. For an example of a help list for a menu bar, see Figure 89. The help list is updated when the menu bar or pull-down menu is written again. When the cursor is in a menu bar or pull-down menu, the menu-bar help list is searched for item-specific help. The menu-bar help list is considered above the base-display help list. When extended help is displayed, help for menu bars and pull-down menus is presented immediately after general help. Base display help is presented after menu-bar help and pull-down menu help. If DDS help is used and the user initially selects pull-down menu help, the user then can page up to see menu-bar help and page down to see base-display help.



Help for File menu bar choice

Help for File pulldown choice 1

Help for File pulldown choice 2

•
•
•

Help for View menu bar choice

Help for View pulldown choice 1

Help for View pulldown choice 2

•
•
•

Figure 89. A help list for a menu bar

Defining Help for a Field

Help can be defined for a field or constant using the HLPARA keyword. For a named field, specify the help area using the *FLD special value and the name of the field. For a constant field, specify the help area using the *CNST special value and an identifier for the constant field. The identifier you specify on the HLPARA keyword should be the same as the value specified on the HLPID keyword for the constant field.

```
|...+....1....+....2....+....3....+....4....+....5....+....6....+....7...
     A                                      HELP
     A          R RECORD
     A          H                           HLPARA(*FLD FIELD)
     A                                      HLPRCD(FIELDHLP LIB/FILE)
     A            FIELD        10A   B  5  5
```

Figure 90. Help for a Named Field

```
|...+....1....+....2....+....3....+....4....+....5....+....6....+....7...
     A                                      HELP
     A          R RECORD
     A          H                           HLPARA(*CNST 1)
     A                                      HLPRCD(HLPCNST1 LIB/FILE)
     A                                 2  2'Constant field' HLPID(1)
```

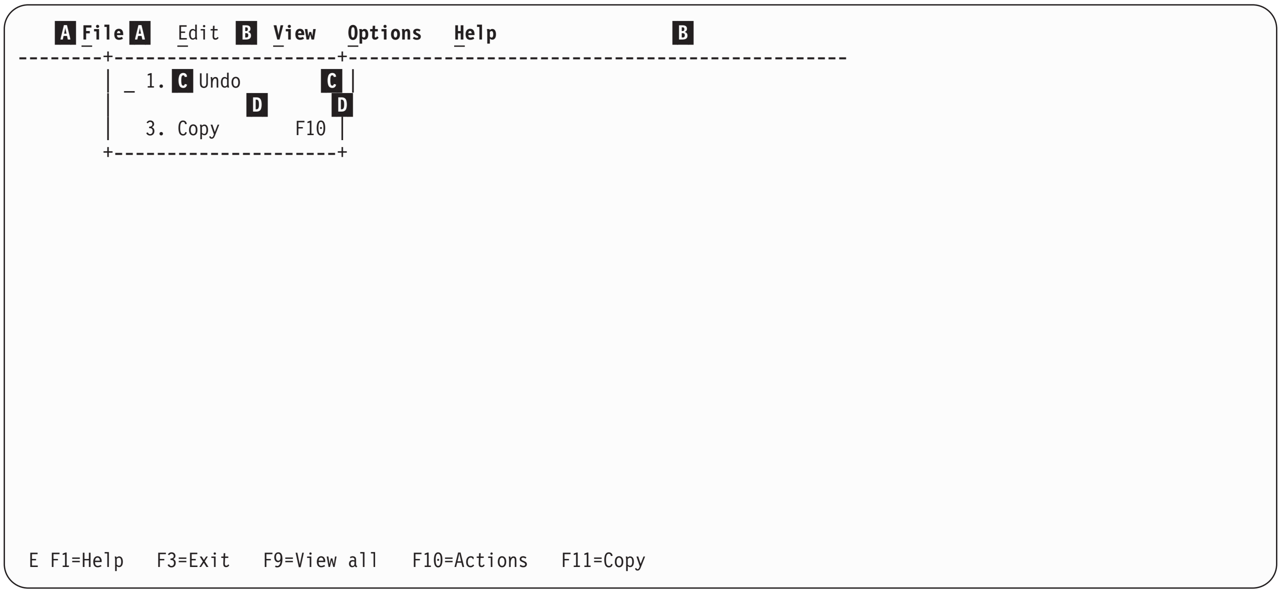
Figure 91. Help for a Constant Field

When the display file is created, the actual help area coordinates are determined by the DDS compiler. These coordinates are shown in the expanded source section of the DDS listing (unless the field is a choice field).

If the field or constant field location or length is changed and the file is re-created, the help area for the field is updated to reflect the new location or length.

Key Interaction for Menu Bars and Pull-Down Menus

Figure 92 shows an example of a display you might create with a menu bar and a pull-down menu. Capital letters A, B, C, D, and E indicate possible cursor locations.



```
   A File A  Edit B View   Options   Help             B
--------+---------------------+---------------------------------------------
        | _ 1. C Undo       C |
        |            D       D|
        |   3. Copy       F10 |
        +---------------------+


 E F1=Help   F3=Exit   F9=View all   F10=Actions   F11=Copy
```

Figure 92. Cursor Locations

Table 21 describes the actions that are performed when certain keys are pressed at the cursor locations.

Table 21. Actions Performed at Different Cursor Locations

	Action Performed When Appropriate Key Is Pressed			
Cursor Location	**Help**	**Cancel**	**Menu-Bar Switch**	**Enter**
A	Help for File menu-bar choice is displayed. (Includes help for the pull-down menu.)	Pull-down menu is removed. Cursor moves to Edit menu-bar choice.	Pull-down menu is removed. Cursor returns to display work area.	File pull-down menu is displayed; Edit pull-down menu is removed.
B	Extended help is displayed.	Pull-down menu is removed. Cursor moves to Edit menu-bar choice.	Pull-down menu is removed. Cursor returns to display work area.	*Cursor not on menu bar choice* message is displayed. Cursor does not move.
C	Help for Undo choice is displayed.	Pull-down menu is removed. Cursor moves to Edit menu-bar choice.	Pull-down menu is removed. Cursor returns to display work area.	If user made a selection, control returns to application. If not, cursor moves to input field and message to make a selection is sent.
D	Help for Edit menu-bar choice is displayed. (Includes help for the pull-down menu.)	Pull-down menu is removed. Cursor moves to Edit menu-bar choice.	Pull-down menu is removed. Cursor returns to display work area.	If user made a selection, control returns to application. If not, cursor moves to input field and message to make a selection is sent.
E	Help for function keys is displayed.	Pull-down menu is removed. Cursor moves to Edit menu-bar choice.	Pull-down menu is removed. Cursor returns to display work area.	If user made a selection, control returns to application. If not, sound beep and cursor moves to input field.

Cursor Movement

You can move the cursor on the application displays that you create using the Tab key or the Cursor keys.

Pressing the Tab Key

Pressing the tab key moves the cursor from field to field on the display, progressing from left to right and top to bottom.

On display stations in configuration F from Table 16 on page 140, the following is true:

- The Menu-bar choices are individual fields.
- The single-choice selection field is one field.
- The multiple-choice selection fields are individual fields.

Pressing the Tab key for one of these displays when no pull-down menu is displayed moves the cursor from choice to choice in the menu bar. Then, the cursor moves from input field to input field on the rest of the display. When a pull-down menu is displayed on one of these displays, pressing the Tab key does the following:

1. Moves the cursor from choice to choice in the menu bar.

2. Skips over the choice selected.
3. Moves the cursor from input field to input field within the pull-down menu.
4. Moves the cursor back to the first choice in the menu bar (the pull-down menu for that choice is not automatically displayed).

While the pull-down menu is displayed, you cannot enter data in the input fields on the base display nor can you tab to the input fields.

On display stations in configurations A, B, and C from Table 15 on page 139, and configurations D and E from Table 16 on page 140, the menu bar is a single field. If no pull-down menu is displayed, pressing the Tab key moves the cursor from choice to choice within the menu bar. The cursor performs a wraparound. When a menu-bar choice is selected, the cursor is placed in the first input field within the pull-down menu. Pressing the Tab key moves the cursor from input field to input field within the pull-down menu and then moves the cursor to the next menu-bar choice. While the pull-down menu is displayed, you cannot enter data in the input fields on the base display nor can you tab to the input fields on the base display.

Pressing the Cursor Keys

On display stations in configuration F from Table 16 on page 140, pressing a cursor key moves the cursor one position in the appropriate direction.

On display stations in configurations A, B, and C from Table 15 on page 139, and configurations D and E from Table 16 on page 140, the cursor moves differently depending on the type of field it is in.

- In the menu bar, pressing the cursor-right or cursor-left keys moves the cursor from choice to choice. The cursor does not perform a wraparound.
- In the base display, pressing any cursor key moves the cursor one position in the appropriate direction.
- When the cursor is positioned on a selection field, see:
  - “Cursor Movement in a Vertical Selection Field” on page 151.
  - “Cursor Movement in a Horizontal Selection Field” on page 151.

Programming Examples

Following are examples of the DDS required to display a menu bar and a pull-down menu with a description of how the DDS coding works.

Using the MNUBARDSP Keyword on the Application Record

```
|...+....1....+....2....+....3....+....4....+....5....+....6....+....7...
     A                                      HELP ALTHELP CF03
     A                                      MNUBARSW(CA10) MNUCNL(CA12)
     A                                      CF04(04) CF06(06)
     A          R MENUBAR                   MNUBAR
     A          H                           HLPARA(*FLD MNUFLD 1)
     A                                      HLPRCD(FILEHLP HLPLIB/HLPFILE)
     A          H                           HLPARA(*FLD MNUFLD 2)
```

```
A                                      HLPRCD(EDITHLP HLPLIB/HLPFILE)
A          H                           HLPARA(*FLD MNUFLD 3)
A                                      HLPRCD(VIEWHLP HLPLIB/HLPFILE)
A          H                           HLPARA(*FLD MNUFLD 4)
A                                      HLPRCD(OPTHLP HLPLIB/HLPFILE)
A          H                           HLPARA(*FLD MNUFLD 5)
A                                      HLPRCD(HELPHLP HLPLIB/HLPFILE)
A          H                           HLPARA(*FLD MNUFLD)
A                                      HLPRCD(MNUBARHLP HLPLIB/HLPFILE)
A            MNUFLD         2Y 0B  1  2
A                                      MNUBARCHC(1 PULLFILE +
A                                                '>File        ')
A  02                                  MNUBARCHC(2 PULLEDIT +
A                                                &EDITTXT)
A                                      MNUBARCHC(3 PULLVIEW +
A                                                '>View        ')
A  04                                  MNUBARCHC(4 PULLOPT +
A                                                '>Options       ' &RTNFLD)
A                                      MNUBARCHC(5 PULLHELP +
A                                                '>Help          ')
A                                      MNUBARSEP((*COLOR WHT))
A                                      CHCAVAIL((*COLOR YLW) (*DSPATR HI))
A                                      CHCSLT((*COLOR GRN))
A            EDITTXT       20A  P
A            RTNFLD         2Y 0H
A          R PULLEDIT
A                                      PULLDOWN
A          H                           HLPARA(*FLD F1 1)
A                                      HLPRCD(UNDOHLP HLPLIB/HLPFILE)
A          H                           HLPARA(*FLD F1 2)
A                                      HLPRCD(MARKHLP HLPLIB/HLPFILE)
A          H                           HLPARA(*FLD F1 3)
A                                      HLPRCD(COPYHLP HLPLIB/HLPFILE)
A            F1             2Y 0B  1  1SNGCHCFLD
A                                      CHECK(ER)
A    01                                CHOICE(1 '>Undo       ')
A                                      CHOICE(2 &MARKTXT)
A                                      CHOICE(3 '>Copy       ')
A                                      CHCCTL(1 &CTLUNDO MSG1112 QUSER/A)
A                                      CHCCTL(2 &CTLMARK &MSG &LIB/&MSGF)
A                                      CHCCTL(3 &CTLCOPY)
A                                      CHCACCEL(1 'F4')
A                                      CHCACCEL(2 'F6')
A                                      CHCAVAIL((*COLOR WHT))
A                                      CHCUNAVAIL((*COLOR BLU))
A            MARKTXT       20A  P
A            CTLUNDO        1Y 0H
A            CTLMARK        1Y 0H
A            CTLCOPY        1Y 0H
A            MSG            7A  P
A            LIB           10A  P
A            MSGF          10A  P
A          R PULLOPT
A                                      PULLDOWN
A          H                           HLPARA(*FLD F1 1)
                                       :
                                       :
```

```
     A            F1             2Y 0B  1  1SNGCHCFLD
                                            :
                                            :
     A          R APPSCR                    MNUBARDSP(MENUBAR &MNUCHOICE +
     A                                                &PULLINPUT)
     A            FIELD1        10A  B 10 12
     A            FIELD2         5S 0B 14 12
     A                                  24  1'F1=Help   F3=Exit   +
     A                                        F10=Actions   F12=Cancel   '
     A            MNUCHOICE      2Y 0H
     A            PULLINPUT      2S 0H
```

Description

In the example using the MNUBARDSP keyword on the application record, the application does a write-read operation to the APPSCR record. This causes the MENUBAR record and the APPSCR record to be displayed. Because the field EDITTXT in the MENUBAR record contains the text >Edit, Edit is displayed as the text for the second menu-bar choice.

Note: The E in Edit will be underlined on your display.

Pressing the F10 key provides quick access to the menu bar (which is always active). Pressing F10 moves the cursor to the first choice in the menu bar. Pressing F10 again (or F12) moves the cursor back to where it was on the application display. On display stations in configuration F from Table 16 on page 140, pressing the Tab key moves the cursor from choice to choice in the menu bar. The cursor skips over any menu-bar choice that is selected. It then moves from input field to input field on the entire display. On display stations in configurations A, B, and C from Table 15 on page 139, and configurations D and E from Table 16 on page 140, when the cursor is located within a menu bar, the cursor movement keys or the Tab key moves the cursor from choice to choice within the menu bar.

If the user selects the Edit action, the system displays the pull-down menu record (PULLEDIT). CHECK(ER) specifies automatic-enter. When the user types a value in F1, control is returned to the application without the user having to press the Enter key. If the user presses F4 or F6, control is also returned to the application.

Because the application performed a write-read operation to the APPSCR record, the APPSCR record is returned to the application. Field MNUCHOICE contains a 2 to identify that menu-bar choice 2 was selected. The PULLINPUT field contains the single-selection field choice (the contents of F1) of the PULLEDIT record.

If the user selects the Options action, control is returned to the application, with 4 set in field RTNFLD and 0 set in field MNUCHOICE. The application determines from the 0 in field MNUCHOICE that control has been returned for pull-down menu update. The application reads record MENUBAR to obtain the choice number set in field RTNFLD. The application updates the record PULLOPT and then writes record PULLOPT. The system then displays PULLOPT as the pull-down menu for the options choice. The system resumes control of the menu bar interaction when the application performs a read operation to the APPSCR record.

On display stations in configurations A and B from Table 15 on page 139, the menu bar separator and the pull-down menus display as solid lines.

On display stations in configuration C from Table 15 on page 139, and configurations D, E, and F from Table 16 on page 140), the menu-bar separator is made up of dashes. The side and bottom borders of the pull-down menu are made up of colons and periods, respectively.

On display stations in configuration C from Table 15 on page 139, each menu-bar choice has a mnemonic. On display stations in configurations A and B from Table 15 on page 139, each single-selection field choice in the PULLEDIT pull-down menu has a mnemonic. This is true unless ENHDSP(*NO) is specified on the CRTDSPF command or CHGDSPF command). In this example, the mnemonic has been set up to be the first character in each of the choices.

Using the MNUBARDSP Keyword on the Menu-Bar Record

```
|...+....1....+....2....+....3....+....4....+....5....+....6....+....7...
     A                                      HELP ALTHELP CF03(03)
     A                                      MNUBARSW(CA10) MNUCNL(CA12 12)
     A                                      CF04(04) CF06(06)
     A          R MENUBAR                   MNUBAR
     A  01                                  MNUBARDSP
     A          H                           HLPARA(*FLD MNUFLD 1)
     A                                      HLPRCD(FILEHLP HLPLIB/HLPFILE)
     A          H                           HLPARA(*FLD MNUFLD 2)
     A                                      HLPRCD(EDITHLP HLPLIB/HLPFILE)
     A          H                           HLPARA(*FLD MNUFLD 3)
     A                                      HLPRCD(VIEWHLP HLPLIB/HLPFILE)
     A          H                           HLPARA(*FLD MNUFLD 4)
     A                                      HLPRCD(OPTHLP HLPLIB/HLPFILE)
     A          H                           HLPARA(*FLD MNUFLD 5)
     A                                      HLPRCD(HELPHLP HLPLIB/HLPFILE)
     A            MNUFLD         2Y 0B  1  2
     A                                      MNUBARCHC(1 PULLFILE +
     A                                                '>File        ')
     A  02                                  MNUBARCHC(2 PULLEDIT +
     A                                                &EDITTXT)
     A                                      MNUBARCHC(3 PULLVIEW +
     A                                                '>View        ')
     A  04                                  MNUBARCHC(4 PULLOPT +
     A                                                '>Options      ' &RTNFLD)
     A                                      MNUBARCHC(5 PULLHELP +
     A                                                '>Help          ')
     A                                      MNUBARSEP((*COLOR WHT) +
     A                                             (*CHAR '-'))
     A            EDITTXT       20A  P
     A            RTNFLD         2Y 0H
     A          R PULLEDIT
     A                                      PULLDOWN
     A          H                           HLPARA(*FLD F1 1)
     A                                      HLPRCD(UNDOHLP HLPLIB/HLPFILE)
     A          H                           HLPARA(*FLD F1 2)
     A                                      HLPRCD(MARKHLP HLPLIB/HLPFILE)
     A          H                           HLPARA(*FLD F1 3)
     A                                      HLPRCD(COPYHLP HLPLIB/HLPFILE)
     A            F1             2Y 0B  1  1SNGCHCFLD
     A    01                                CHOICE(1 '>Undo        ')
     A                                      CHOICE(2 &MARKTXT)
     A                                      CHOICE(3 '>Copy        ')
     A                                      CHCCTL(1 &CTLUNDO MSG1112 QUSER/A)
     A                                      CHCCTL(2 &CTLMARK &MSG &LIB/&MSGF)
     A                                      CHCCTL(3 &CTLCOPY)
     A                                      CHCACCEL(1 'F4')
     A                                      CHCACCEL(2 'F6')
```

```
A            MARKTXT       20A  P
A            CTLUNDO        1Y 0H
A            CTLMARK        1Y 0H
A            CTLCOPY        1Y 0H
A            MSG            7A  P
A            LIB           10A  P
A            MSGF          10A  P
A
A          R PULLOPT
A                                      PULLDOWN
A          H                           HLPARA(*FLD F1 1)
                                       :
                                       :
A            F1             2Y 0B  1  1SNGCHCFLD
             :
             :
A
A          R APPSCR
A            FIELD1        10A  B 10 12
A            FIELD2         5S 0B 14 12
A                                 24  1'F1=Help   F3=Exit   +
A                                       F10=Actions   F12=Cancel   '
```

Description

In the example using the MNUBARDSP keyword on the menu-bar record, the application writes the MENUBAR record to display the menu bar. The application then performs a write-read operation to the APPSCR record to display the application display.

As in the previous example, F10 provides quick access to the menu bar (which is always active). Pressing F10 moves the cursor to the first choice in the menu bar. Pressing F10 again (or F12) moves the cursor back to where it was on the application display.

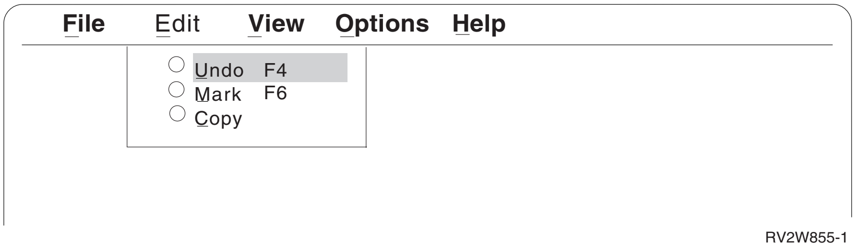
On display stations in configuration F from Table 16 on page 140, pressing the Tab key moves the cursor from choice to choice in the menu bar and from input field to input field on the entire display. The cursor skips over any menu-bar choice that is selected.

When the cursor is located within a menu bar on display stations in configurations A, B, and C from Table 15 on page 139, and configurations D and E from Table 16 on page 140, the cursor movement keys or the Tab key moves the cursor from choice to choice within the menu bar.

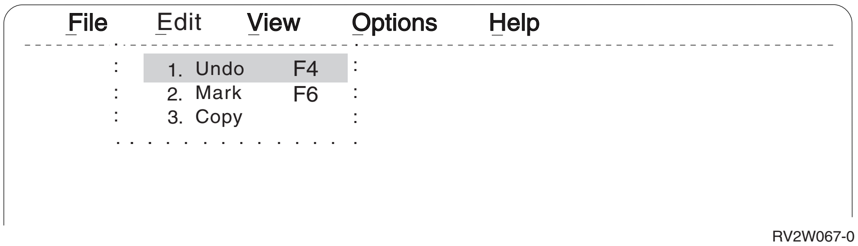
If the user selects the Edit action, the system displays the pull-down menu record (PULLEDIT). Because CHECK(ER) is not specified on field F1, the user enters a value by typing the choice number and pressing the Enter key, or by pressing F4 or F6. When a value is entered or an accelerator key is pressed, control is returned to the application. Because the application was doing a write-read operation to the APPSCR record, the APPSCR record is returned to the application. The application must then read the MENUBAR record to determine the choice selected (2 is returned in field MNUFLD). Because the input-field parameter was not specified on the MNUBARDSP keyword, the application reads record PULLEDIT to receive the pull-down menu input.

How the Displays Look

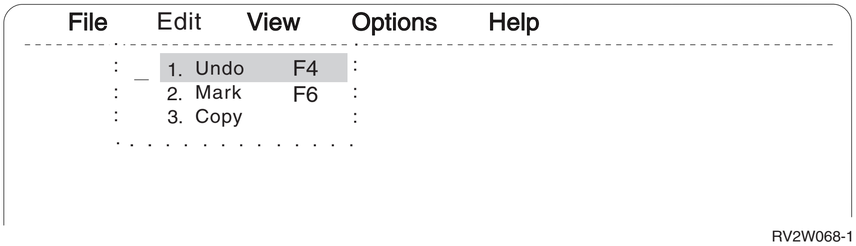
On display stations in configurations A and B from Table 15 on page 139, the display looks like this:



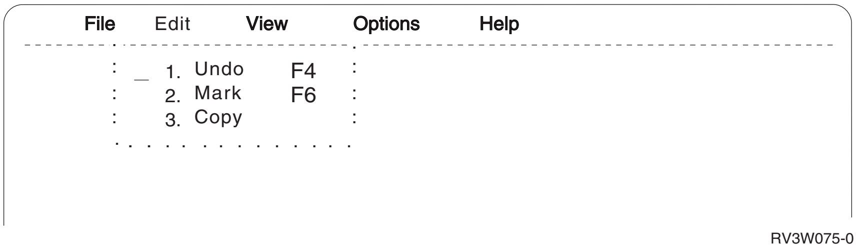
On display stations in configuration C from Table 15 on page 139, the display looks like this:



On display stations in configurations D and E from Table 16 on page 140, the display looks like this:



On display stations in configuration F in Table 16 on page 140, the display looks like this:



Simple Hotspots

Controllers that support an enhanced interface for nonprogrammable work stations provide simple hotspots. A **hotspot** is an area of a display that, when clicked on, performs a function. You must have a display station with a mouse. Hotspots are available on configurations A and B from Table 15 on page 139. On InfoWindow II display stations, the hotspot must be selected with the left mouse button.

The following hotspots are provided by controllers that support an enhanced interface for nonprogrammable work stations:

Command key emulation

Page Up and Page Down key emulation

Enter key emulation

The user can perform the Enter function on InfoWindow II display stations by double-clicking the left mouse button.

Command Key Emulation

When the user clicks on a command key, the pointer device cursor must be between the first and last characters (inclusive) of the command key. When the user selects the command key, the keyboard locks and the command key is processed. The function performed is defined by DDS. The system performs the function as if the actual command key had been pressed. This includes setting response indicators.

When the user clicks on a command key, the system scans the command key text to the left until it finds one of the following:

An attribute
Two blanks
Two nulls
A blank and a null
Column one of the row

After finding the beginning of the command key text string, the system scans the text to the right until it finds a match with one of the following:

Fx= x can be 1 to 9.

Fyx= y can be 0, 1, or 2. If y is 0, x can be 1 to 9. If y is 1, x can be 0 to 9. If y is 2, x can be 0 to 4.

PFx= x can be 1 to 9.

PFyx= y can be 0, 1, or 2. If y is 0, x can be 1 to 9. If y is 1, x can be 0 to 9. If y is 2, x can be 0 to 4.

PFx x can be 1 to 9.

PFyx y can be 0, 1, or 2. If y is 0, x can be 1 to 9. If y is 1, x can be 0 to 9. If y is 2, x can be 0 to 4.

After the system finds a match, it performs the command key function.

Page Up and Page Down Key Emulation

The user can click on the plus (+) and minus (–) characters to page down (roll up) and page up (roll down), respectively. When the user clicks on the + or – characters, the keyboard locks and a Roll Up AID key and Roll Down AID key is generated. The function performed is defined by DDS. The system performs the function as if the actual key had been pressed.

One way to implement this function is to do the following:

1. Specify *MORE on the Subfile End (SFLEND) keyword.

   This causes the display file to use the text defined on the CPX6AB2 and CPX6AB1 messages. The default text for these messages is `More ...` and `Bottom`, respectively.
2. Change the text for the CPX6AB2 message to `More: +/-`.
3. Change the text for the CPX6AB1 message to `Bottom: -`.

Programmable Mouse Buttons-Overview

The programmable mouse buttons function allows attention indicators (AIDs) to be associated with various pointer device events. AID codes are normally associated with various command keys on the keyboard. These keys are used to communicate action requests from the user to the system or application. Some command keys that generate AIDs are Enter, Help, Rollup, Rolldown, and the 24 command attention or function keys. Single event AIDs and two event AIDs can be programmed. Up to 18 pairs of pointer device events and associated AIDs may be defined. These events consist of 3 buttons with 3 events each (up, down, and double click) in two keyboard states (shifted and unshifted).

A single event AID definition would also associate an AID code with a single pointer device event whereas the two event AID definition would also associate an AID with two consecutive pointer device events.

Use the programmable mouse button (MOUBTN) keyword to associate a command key or Event-ID with one or two pointer device events. This keyword can be specified at the file or record level.

Notes:

1. This function is available only for displays attached to a controller that supports an enhanced interface for nonprogrammable work stations.
2. The only pointer device supported is a mouse or a device that emulates a mouse.

Pointer Device Events

With a three button mouse, there are 18 pointer device events possible: 3 buttons with 3 events each (up, down, double click) in two keyboard states (shifted and unshifted). The pointer device events are:

- Left button pressed
- Left button released
- Left button double click
- Right button pressed
- Right button released
- Right button double click
- Middle button pressed
- Middle button released
- Middle button double click
- Shifted left button pressed
- Shifted left button released
- Shifted left button double click
- Shifted right button pressed
- Shifted right button released
- Shifted right button double click
- Shifted middle button pressed
- Shifted middle button released
- Shifted middle button double click

Notes:

1. The Shift key or Shift Lock key must be held down for a shifted pointer device event. Caps Lock state and Shift Lock state are not considered shifted. Releasing the Shift key does not reset Caps Lock state or Shift Lock state if used for a shifted pointer device event.
2. The nonprogrammable work station (NWS) has a setup option to switch the functions of the left and right buttons within the NWS. The system has no knowledge of this. This provides the concept of left- and right-handed mice. For this document, all references to the mouse buttons assumes a right-handed mouse where the left and right buttons follow the usual definitions for left and right.

AID Codes to be Returned

The AID associated with a pointer device event may be any currently supported AID or a host-defined AID value between X'70' and X'7F'. The following AIDs are supported:

X'31' - X'3C' CA/CF01-CA/CF12 (Cmd 1 - 12)

X'70' - X'7F' E00-E15 (EVENTS)

X'B1' - X'BC' CA/CF13-CA/CF24 (Cmd 13 - 24)

X'BD'	CLEAR (Clear)
X'F1'	Enter or Record Advance
X'F3'	HELP (Help - not in error state)
X'F4'	ROLLDOWN (Page Down)
X'F5'	ROLLUP (Page Up)
X'F6'	PRINT (Print)
X'F8'	HOME (Record Backspace)

Programmable Mouse Buttons-Benefits

Mouse buttons can be used for such things as reordering the layering of windows and selecting objects. For example, an application programmer can program middle button down as a single event AID to enable window reordering when multiple windows are on a display. When the user presses the middle button, the cursor moves to the pointer device cursor location and the host-defined AID is returned. The application uses the host-defined AID to recognize a reorder request. If the row and column cursor address indicates that the pointer device event occurred within an overlaid window, the windows can be reordered. If the row and column cursor address indicates that the pointer device event did not occur within an overlaid window, the application can ignore the AID or post a message.

Note: If the EVENT IDs (E00-E15) are used, you may consider defining equivalent functions to be performed using the keyboard. For example, you may allow the user to position the text cursor and press a function key to perform a function equivalent to pressing a mouse button. This would enable functions on displays attached to controllers that do not support an enhanced interface for nonprogrammable work stations and on nonprogrammable displays without pointer devices.

Programmable Mouse Buttons Operation

When a pointer device event is performed that has been programmed as a single event and no other function has higher priority, the following occurs:

1. The keyboard is locked (as it is for function keys).
2. The cursor is moved to the pointer device cursor location.
3. The specified AID is returned to the host.
4. If the AID or moving the cursor normally results in validation of entry field data, the data is validated.
5. If the specified AID normally returns inbound entry field data, inbound entry field data is included. The format of the inbound data is like typical inbound data.
6. Control is returned to the application.

   There will be no way for the application to differentiate between the pointer device event and the corresponding command key. However, the pointer device event may be associated with an EVENT ID (E00-E15) that is not also associated with any command key. This provides a way to detect a pointer device event.

When a two event pointer device event is performed, the system looks for the leading edge event. When the leading edge event is received, the following occurs:

Note: Inbound data is not returned until the trailing edge event occurs.

1. A programmable-two-event state is entered.
2. A marker box is drawn around the location of the pointer device cursor on nonprogrammable work stations capable of displaying a marker box. The marker box appears as 4 blue lines around the character.

3. The pointer device color is changed to white on nonprogrammable work stations capable of displaying white.
4. The system looks for the trailing edge event.

Keystrokes and host data streams will cancel the programmable-two-event state. Some pointer device events are ignored while waiting for the trailing edge event. For more information, see "Programmable Mouse Buttons-NWS Considerations." When the trailing edge event is received, the following occurs:

1. The marker box is erased
2. The pointer device cursor color is changed to input inhibited
3. The keyboard is locked
4. The text cursor is moved to the location of the pointer device cursor

   If the specified host-defined AID normally returns inbound entry field data, inbound data is included. The ending row and column location is returned.

   Note: The RTNCSRLOC keyword can also be used to retrieve the starting location of the cursor. This may be different from the ending cursor location when using a two event definition.

Using programmable mouse buttons can prohibit other pointer device functions on the display. For example, the copy and paste function has a lower priority than the programmable mouse buttons for the shifted left button press and release. For more information on the priority of pointer device events, see "Programmable Mouse Buttons-Event Processing Priority" on page 200.

Programmable Mouse Buttons-NWS Considerations

Many of the pointer device events result in either the text cursor being moved or an AID being sent to the host. If the text cursor was in an entry field, entry field requirements (for example, mandatory fill) are checked before the text cursor is allowed to move or an AID is sent. This could result in an error code being posted and the pointer device event would not be processed. For example, the 0014 error code may be posted indicating that a mandatory fill field contains a null.

The NWS passes an event to the system any time a button is pressed or released. It passes a double-click event to the system if a button is pressed, released, and pressed again within a user-specified double click time. The system sees a button pressed event, button released event, and a double click event, and eventually a button released event.

A pointer device event will be ignored by the system if any of the following are true:

- The keyboard is locked. An exception is single event programmable mouse buttons which can be defined to be queued if the keyboard is locked.
- The keyboard is in system request state or ss message state.
- The keyboard is in operator error state. An exception is the left button down and shifted left button down which can reset an operator error.
- The display is in WP mode.
- The system has any stored type ahead keystrokes.
- The system does not have a pending read. An exception is single event programmable mouse buttons. When the keyboard is unlocked, the system normally has a pending read. However, 3270DE sometimes unlocks the keyboard without a pending read. Processing a pointer device event could be confusing in this case.

Programmable Mouse Buttons-Event Processing States

Pointer device event processing can be in various states. The following events, when received in an unexpected state, cause the state of the pointer device event processing to be reset:

- Most mouse button events (except as noted)

- Most keyboard events (not shift key make/break, for example)
- All host display updates

The state is reset as follows:

- If the scroll bar drag and drop state is active, the drag and drop state is reset and the scroll bar is re-written to the original state.
- If the copy and paste state is active, the copy and paste state is reset and the ending point indication is removed from the display.
- If the programmable mouse buttons two-event state is active, the state is reset and no AID is posted. The follow events are ignored in the two-event state so as not to reset the state when that would not be the desired effect:
  - A mouse button release event associated with the leading edge event.
  - A mouse button press event which must be generated to get to the trailing edge event. The trailing edge must be the button release or double click on the same button and shift state.
  - A mouse button release event which must be generated to get to the trailing edge event. The trailing edge must be the double click on the same button and shift state.

Mouse button events which cause states to be reset are not processed any further. Keyboard events and host screen updates are processed as usual after resetting any mouse event processing state.

Programmable Mouse Buttons-Event Processing Priority

This section describes the event processing priorities for the following events:

- Unshifted left button pressed
- Unshifted left button released
- Unshifted left button double clicked
- Shifted left button pressed
- Shifted left button released
- Shifted right button pressed
- All other events

Unshifted Left Button Pressed Event Processing

If the shift key is not down and the system receives a left button pressed event, the system determines the position of the pointer device cursor and performs exactly one of the following functions, checking in the order listed:

1. If an operator error is on the display, the pointer device cursor can be used to reset the operator error. Depending on the position of the pointer device cursor, one of the following functions is performed:
   a. If the pointer device cursor is on the operator error line, then the same function is performed as if the user pressed the Reset key. The reset function is also performed if the pointer device cursor is on the last line of the display and the separately displayable operator error line line is being used (line 25 or line 28).
   b. Otherwise, the pointer device event is ignored and the following checks are not done.
2. If the pointer device cursor is on a selection field choice:
   a. If the choice is a cursorable choice, the cursor is moved to the location of the pointer device cursor. A selection cursor is created. The function of a Spacebar key is performed (this includes posting operator error 0084 if the choice is unavailable). The cursor must be somewhere between the first text attribute and the last text attribute in the choice. If selection indicators are used, the cursor must be somewhere between the selection indicator attribute and the last text attribute in the choice.
   b. If on a noncursorable or null choice, the cursor is not moved and operator error 0084 is posted.

3. If the pointer device cursor is on a scroll bar character (arrow character or shaft character), a roll AID may be sent to the host. If a roll AID is sent to the host the field MDT is set on. The text cursor may be moved to the position of the pointer device cursor. This depends on one of the following:
   - The setting of the ″move the cursor to the scroll bar on a pointer device interaction″ flag in the Define Scroll Bar command
   - The position of the cursor if this scroll bar is associated with a selection field

   The keyboard is locked (as if the user pressed a normal AID key).
   a. If the pointer device cursor is in the shaft above the slider, a Roll Down AID with a scroll increment of X'00000000' is sent to the host. Similarly, if the pointer device cursor is in the shaft below the slider, a Roll Up AID is sent.
   b. If the pointer device cursor is on the top arrow character and the slider is not already at the top of the scroll bar, a Roll Down AID with a scroll increment of X'00000001' is sent to the host. Similarly, if the pointer device cursor is on the bottom arrow character and the slider is not already at the bottom of the scroll bar, a Roll Up AID with a scroll increment of X'00000001' is sent to the host. If the pointer device cursor is on the top arrow character and the slider is already at the top of the scroll bar, the left button pressed is ignored. The bottom arrow character functions similarly.
   c. If the pointer device cursor is on the slider, a drag and drop function should be started (scroll bar slider drag and drop state). The NWS is told to pass pointer device cursor movement to the system. In scroll bar drag and drop state, for each movement event, the system calculates the row of the pointer device cursor and compares this value with the last row which was processed. If the row values are different, the scroll bar characters are re-written. The slider is re-positioned within the shaft. If the pointer device cursor moved up one row, the slider is moved up one row. If the pointer device cursor is moved up more rows than exist in the shaft above the slider, the slider is moved to the top of the scroll bar shaft. When the left button is released, the drop function is performed; see Left Button Released for a description of the roll AID request. If any pointer event other than left button released occurs, or if any keyboard key is pressed, or if any screen update is done by the host, the following occurs:
      - The scroll bar characters are re-written to their original state
      - The scroll bar slider drag and drop state is reset
      - No roll AID is sent
      - The MDT is not set
4. If the pointer device cursor is in a light pen field (first field position through the last field position), the system treats the event as if a light pen tip switch were activated at the position of the pointer device cursor.
5. If this mouse button event has been programmed with the Programmable Mouse Buttons structured field, the event is handled as described above.
6. If the pointer device cursor is on a simple hot spot, a hot spot function is performed. Hot spots enable a pointer device to partially drive older applications. In order to be considered a hot spot, the pointer device cursor must not be in an entry field. The hot spot functions for unshifted left button pressed events are:
   - Command Key emulation. For more information, see “Command Key Emulation” on page 196.
   - Page Up and Page Down Key Emulation. For more information, see “Page Up and Page Down Key Emulation” on page 196.
   - Enter key emulation. For more information, see “Unshifted Left Button Double Click Event Processing” on page 202.
7. If pull-down cancel mode is active, the cursor is moved to the position of the pointer device cursor, the keyboard is locked (treated like a normal AID), and the specified AID is returned to the host. Pull-down cancel mode is active if a selection field was written to the display and a Pull-Down Cancel AID was specified in the Define Selection Field major structure.

   Note: Pull-down cancel mode is lower priority than hot spots because pull-down menus could have command keys or More -/+ inside them.

8. Otherwise, the cursor is moved to the location of the pointer device cursor. The cursor could be a text cursor or a selection cursor (for a highlighted entry field). The system allows the text cursor to move to a noncursorable text location even if cursor movement to input-capable positions only is set on.

Unshifted Left Button Released Event Processing

If the shift key is not down and the system receives a left button released event, the system determines the position of the pointer device cursor and performs exactly one of the following functions, checking in the order listed:

1. If scroll bar slider drag and drop state is active, the scroll bar characters may be updated. For more information on scroll bars, see “Unshifted Left Button Pressed Event Processing” on page 200. If the row position of the pointer device cursor is different than the row position when the slider was last written, the scroll bar characters should be written. If the final slider position is different than the original slider position (when the drag and drop was started), the following occurs:
   a. The MDT is set on
   b. The text cursor may be moved to the scroll bar slider
   c. A Roll AID is sent to the host with a scroll increment indicating the number of rows or columns to be scrolled
   d. The keyboard is locked (treated like a normal AID)

   The specific AID depends on the direction the slider moved. If the slider did not move, no AID is sent. In all cases, the scroll bar slider drag and drop state is reset.
2. If this mouse button event has been programmed with the Programmable Mouse Buttons structured field, the event is handled as described above.
3. Otherwise, the pointer device event is ignored.

Unshifted Left Button Double Click Event Processing

If the shift key is not down and the system receives a left button double click event, then the system determines the position of the pointer device cursor and performs exactly one of the following functions, checking in the order listed:

1. If this mouse button event has been programmed with the Programmable Mouse Buttons structured field, the event is handled as described above.
2. Otherwise, if the previous left button pressed event simply positioned the cursor, the keyboard is locked and the Enter AID is sent to the host. This is a hot spot function.
3. Otherwise, the unshifted left button double click event is ignored.

   Note: The user must have done one of the following:
   - Selected a selection field choice
   - Operated against a scroll bar
   - Caused a pointer device selectable AID
   - Selected a hot spot (for example, a command key)
   - Caused some other left button pressed event function, other than the default action of simply positioning the cursor.

Shifted Left Button Pressed Event Processing

If the shift key is down and the system receives a left button pressed event, the system determines the position of the pointer device cursor and performs exactly one of the following functions, checking in the order listed:

1. If an operator error is on the display, the pointer device cursor can be used to reset the operator error. Depending on the position of the pointer device cursor, one of the following functions is done:
   a. If the pointer device cursor is on the operator error line, then the same function is performed as if the user pressed the Reset key. The reset function is also performed if the pointer device cursor is on the last line of the display and the separately displayable operator error line line is being used (line 25 or line 28).

    b. Otherwise, the pointer device event is ignored and the following checks are not performed.

2. If this mouse button event has been programmed with the Programmable Mouse Buttons structured field, the event is handled as described above.
3. Otherwise, a copy and paste function (frequently called cut and paste) is started. The copy and paste state is set. The location of the pointer device cursor is marked to indicate an end point of the select. A line is drawn above and below the character location of the pointer device cursor. If any other pointer device event other than shifted left button released occurs, or the user presses any key, or any data is received from the host, the copy and paste state is reset and the two lines are removed. When the shifted left button is released, copy and paste processing continues (see "Shifted Left Button Released Event Processing") for later use in a paste operation (see "Shifted Right Button Pressed Event Processing" on page 204).

Shifted Left Button Released Event Processing

If the shift key is held down and the system receives a left button released event, the system determines the position of the pointer device cursor and performs exactly one of the following functions, checking in the order listed:

1. If copy and paste state is active, the user has now marked the second end point of the copy. The second end point is marked by drawing three lines around the second end point and adding one line around the first end point. This looks like square brackets enclosing the copy data. The user can mark either the starting or ending point first. Then, the marked copy data is stored in the display (for example, 348X NWS) for later use (see "Shifted Right Button Pressed Event Processing" on page 204). For performance reasons, the copy and paste data will be copied from the display to the system-managed buffer in the display (without crossing the twinaxial cable). Some formatting may be done by the system to the copy and paste data after the data has been copied within display memory:
    - If the display is a SBCS display (not capable of DBCS) and the data was copied from a non-display area, the system writes a non-display attribute over the first position of the copy and paste data. A non-display area is an area in which a non-display attribute was in effect for the copy and paste data.
    - If the display is a DBCS display, but not capable of DBCS-pure fields, the following formatting takes place:

      Note: Support for DBCS-pure fields includes supports extended SO/SI attributes which do not require a screen position.

        – If the first byte of copy and paste data is DBCS data (preceded by a SO character) and the starting point of the DBCS data is the second byte of a DBCS character, the starting point of the copy is decreased by one position (to include the entire character).

        – If the last byte of copy and paste data is DBCS data and the ending point of the DBCS data is the first byte of a DBCS character, the ending point of the copy is increased by one position (to include the entire character).

        – If the first byte of copy and paste data is DBCS data (preceded by a SO character), the system writes an SO ahead of the first byte of copy and paste data.

        – If the last byte of copy and paste data is DBCS data, the system writes an SI after the last byte of copy and paste data.

        – If the data was copied from a non-display area, the system writes a non-display attribute over the first non-SO/SI character in the copy and paste data. If the character in the copy and paste data is a DBCS character, the system writes two non-display attributes over the first two non-SO/SI characters.

    The end point indications on the display should be reset when one of the following occurs:

    - Any key is pressed other than the four cursor movement keys. The system allows the user to move the text cursor without resetting the end points.
    - Another pointer device event occurs.
    - The host updates the display.

The copy and paste data is lost only when the NWS powers down or the user completes the copy portion of another copy and paste operation.

2. If this mouse button event has been programmed with the Programmable Mouse Buttons structured field, the event is handled as described above.

Shifted Right Button Pressed Event Processing

The system performs exactly one of the following functions, checking in the order listed:

1. If this mouse button event has been programmed with the Programmable Mouse Buttons structured field, the event is handled as described above.
2. If the system receives a right button pressed event and the user has previously selected data for a copy and paste operation, the system auto-keys the selected data at the location of the text cursor. This is done independently of the location of the pointer device cursor. The system performs the following steps:
   - If the text cursor is not in an entry field, operator error X'0005' is posted.
   - The copy and paste data is read from the display into main storage. The length of the data read is the smaller of the length of the copy and paste data, and the length of data which will fit into the entry field.

     Note: Continued fields are considered a single entry field.
   - Data following a non-display attribute is converted to blanks, until another non-display attribute is found.
   - Attributes are converted to blanks.
   - GUI-like characters are converted to blanks.
   - Nulls are converted to blanks.
   - The system will perform all normal field checks (for example, digits only, alphanumeric only, and so on). This could result in an operator error. If an operator error is posted, no data following the point of the error is pasted.
   - If the field is a monocase field, the data is monocased.
   - If the display is a DBCS display and the field is a DBCS field, the following processing takes place:
     - If the field is a DBCS-only field, the copy and paste data must begin with an SO (otherwise, operator error ″0092″ is posted). The DBCS characters are placed into the field until the field ending SI is found, or an SI is found in the copy and paste data. The SO and SI characters are not pasted.
     - If the field is a DBCS-either field, the copy and paste data must be the proper format or operator error 0092 is posted. Operator error 0092 indicates that either no SO character is allowed (if the field is SBCS), or that the data must start with an SO character (if the field is DBCS). DBCS paste is just like describe above for a DBCS-only field.
     - If the field is a DBCS-pure field, the copy and paste data must begin with an SO (otherwise, operator error ″0092″ is posted). The DBCS characters are placed into the field until the end of the field is found, or an SI is found in the copy and paste data. The SO and SI characters are not pasted.
     - If the field is a DBCS-open field, the copy and paste data is auto-keyed into the field. If the copy and paste data starts with an SO character, the SO is pasted if the cursor is under a DBCS character. If the cursor is under an SBCS character, the SO is not pasted. If the copy and paste data does not start with an SO character, an SI will be pasted if the cursor is under a DBCS character. If the last character in the field is a DBCS character, the system will reserve room for an SI after the last DBCS character.
   - If the user is in replace mode and the paste data is too long to fit in the entry field, as much data is placed in the field as possible, the cursor is placed in the last position of the entry field, and operator error 0012 is posted.

- If the user is in insert mode and the paste data requires more positions than the number of nulls at the end of the entry field, as much data is placed in the field as possible, the cursor is placed on the last export character, and operator error 0012 is posted.

Note: The copy and paste data is never reset. The user is allowed to paste the most recent copy and paste data multiple times.

3. If the user has not previously selected data for a copy and paste operation, the system will ignore the event.

Any Other Pointer Device Event Processing

1. If the mouse button event has been programmed using the Programmable Mouse Buttons structured field, the event is handled as described above.
2. Otherwise, the event is ignored.

Grid Line Structures-Overview

Grid line structures include horizontal lines, vertical lines, and boxes. They can only be displayed on DBCS display stations. For details on the required hardware, see "Hardware Requirements for Grid Line Structures" on page 207. An example of grid line structures appears in Figure 93.

The grid line keywords also allow you to do the following:

- Clear grid lines within a specified rectangle
- Erase a specified grid line structure
- Control the attributes of grid line structures such as color and line type

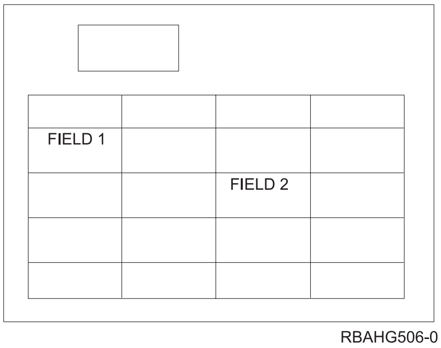


Figure 93. Grid line structures

DDS for Grid Line Structures-Example

The DDS in Figure 94 on page 206 creates the grid line structure in Figure 93.

```
|...+....1....+....2....+....3....+....4....+....5....+....6....+....7...
     A                                      GRDATR((*COLOR BLU) (*LINTYP SLD))
     A
     A          R GRDREC                    GRDRCD
     A  96                                  GRDCLR
     A                                      GRDBOX((*POS (3 10 4 20)) +
     A                                      (*TYPE PLAIN) +
     A                                      (*COLOR RED) (*LINTYP THK) +
     A                                      (*CONTROL &CNTL1))
     A
     A                                      GRDBOX((*POS (10 2 10 60)) +
     A                                      (*TYPE HRZVRT 2 15) +
     A                                      (*CONTROL &CNTL2))
     A            CNTL1          1S 0P
     A            CNTL2          1S 0P
     A
     A          R REC1
     A
     A            FIELD1         7A  B 15  5CHGINPDFT
     A
     A            FIELD2         7A  B 18 35CHGINPDFT
```

Figure 94. DDS for Grid Line Structures-Example

Notes:

1. It is recommended to add a CHGINPDFT keyword to an input/output field when a grid line structure is defined directly under the field.
2. In general, grid line structures are written in the order that they are coded in the display file.
3. If a display file is created or changed to nonenhanced display (ENHDSP *NO), no grid line records are written to the display.

When record GRDREC is written with option indicator 96 on, the GRDCLR keyword causes all grid line structures on the screen to be cleared. The remainder of the keywords are then processed in the order that they are coded. If the program-to-system field CNTL1 contains a 1, the first GRDBOX keyword is ignored. If CNTL1 contains 0, a PLAIN box grid is displayed. This box begins at row 3, column 10. It has a depth of 4 rows and a width of 20 columns. The box has a thick line type and is red. If CNTL1 contains -1, the grid line structure at the defined position is erased and all other grid line structures on the display are left intact.

If the program-to-system field CNTL2 contains 1, the second GRDBOX keyword is ignored. If CNTL2 contains 0, a horizontally and vertically ruled box is displayed. This box begins at row 10, column 2. It has a depth of 10 rows and a width of 60 columns. The box has a horizontal rule every 2 rows and a vertical rule every 15 columns. Because there are no attributes defined on the GRDBOX keyword, the attributes default to those defined on the GRDATR keyword at the file level. These attributes are a solid line type and color blue. If CNTL1 contains -1, the grid line structure at the defined position is erased and all other grid line structures on the display are left intact.

Grid line structures are defined in records separate from other data fields. Grid line records are displayed independent of all other data records. Grid line records are also cleared independent of other data records. If the device file has DFRWRT(*YES), all records written are buffered until a GET operation is done. This includes all grid line records. A READ or PUTGET operation cannot be done to a grid line record because there is no possible input for the grid line record. The FRCDTA keyword is allowed on grid line records. In this case, the grid line record written is immediately displayed regardless of the DFRWRT keyword.

Grid Line Structures and Windows

If a grid line structure is on the display when a GUI window is displayed, the controller removes all grid lines under the window prior to displaying the window. If grid lines are on the display when a non-GUI window (for example, a UIM help window) is displayed, all grid lines are removed from the display. When the window is removed from the display, the display is restored including all grid line structures.

A grid line record can define a window or specify a window reference record. When a grid line structure is defined with a window, all start- row and start-column parameters are relative to the start of the window. This includes both the start-row and start-column parameters defined with DDS and the start-row and start-column values set at run time with program-to-system fields. A grid line structure can be displayed outside of the window. The depth, width, and length parameters will truncate to the end of the display if they are too large for the display size.

If a window is displayed, writing a grid line record to the base display causes the window to be removed unless the USRRSTDSP keyword is defined on the window record. The base screen is restored prior to displaying the grid line structures.

Hardware Requirements for Grid Line Structures

Grid line support requires DBCS hardware. This hardware should have the capability of running Japanese DOS. Japanese DOS is supported by the following PS/55 systems:

- 5523-S, 5523-V, 8551-S
- 5535-S (laptop)
- 5530-T/V/W, 5541-T, 5551-S/TV/W, 5561-W (desktop)
- 5571-T/V, 5580-W/Y (floorstanding model)

Note: PS/1 and PS/2® systems do not support Japanese DOS.

For the desktop and floorstanding PS/55 systems, the Japanese keyboard (5576-001/002/003/A01) and DBCS-capable display (5574) are required. The 5530 system has an integrated display. As a twinaxial communication adapter, 5250 adapter/A (ID#65X1092) is required.

The i5/OS system can sense if the attached display is capable of displaying grid line structures. If a grid line record is written to a display that does not support grid line structures, the record is ignored. However, window keywords on the grid line record are processed.

Grid line structures (non-field level file) can be printed using the Print Screen key if the printer supports DBCS.

Inserting HTML Tags

The World Wide Web (or Web for short) is a graphical interface that provides access to an enormous amount of information available on the Internet. The Web allows Internet users to access documents that contain text and non-text objects (such as video, sound, graphics, and so on). The documents can contain “links” (hyperlinks) to other documents that may also contain links to other documents. The text within a Web document that contains hyperlinks is called hypertext. The chain of reference from document to document is virtually endless since all documents can link to other documents.

You can tailor documents on the Web to present multimedia information from a variety of sources allowing end users to optionally access the information identified by the links.

DDS support of the i5/OS 5250 Workstation Gateway allows you to change existing applications to enable them for the Internet through the World Wide Web. The i5/OS 5250 Workstation Gateway translates all 5250 data streams to an HTML (Hypertext Markup Language) document and exports the document from i5/OS. You can insert HTML tags into a display file that allow graphic capabilities of the client web browser to be utilized with only minor changes to the display file source.

The HTML keyword is given a row/col position in the DDS file. However, row and column positioning has no meaning in an HTML document. The row and column in the display file determines the order of HTML tags in the HTML document that is created. For example, an HTML keyword at row 2/column 4 appears before an HTML keyword with a row 2/column 6 position. HTML keywords with the exact same row and column position will be placed into the HTML document in the same order in which they are defined in the DDS file.

Resolving HTML Field Overlap

The following examples show where the HTML will appear when you use the following coding:

Example 1

The HTML field has a starting column 2 before an output field. The HTML will appear before the field.

```
        A  01        FLD1A         20   O 15  7DFTVAL('Output Field')
        A  01                             15  5HTML('<p>HTML code')
will result in:
                      HTML code
                      Output Field
```

Example 2

The HTML field has a starting column 1 before an output field, meaning that the HTML starts at the attribute byte of the output field. The HTML will appear before the field.

```
        A  01        FLD1A         20   O 15  6DFTVAL('Output Field')
        A  01                             15  5HTML('<p>HTML code')

will result in:
                      HTML code
                      Output Field
```

Example 3

The HTML field has a starting column equal to an output field. The HTML will appear before the first character of the field.

```
        A  01        FLD1A         20   O 15  6DFTVAL('Output Field')
        A  01                             15  6HTML('<p>HTML code')

will result in:
                      HTML code Output Field
```

Example 4

The HTML field has a starting column 1 past the starting column of an output field. The HTML will appear after the 1st character of the output field.

```
        A  01        FLD1A         20   O 15  6DFTVAL('Output Field')
        A  01                             15  7HTML('<p>HTML code')

will result in:
                     OHTML codeutput Field
```

Example 5

The HTML field has a starting column 1 past the ending column of an output field, meaning that it overlaps the ending attribute. The HTML will appear after the last character of the output field.

```
        A  01        FLD1A         20   O 15  6DFTVAL('Output Field')
        A  01                             15 27HTML('<p>HTML code')

will result in:
                      Output FieldHTML code
```

Example 6

The HTML field has a starting column 2 past the ending column of an output field. The HTML will appear after the output field.

```
        A  01        FLD1A         20   O 15  6DFTVAL('Output Field')
        A  01                             15 28HTML('<p>HTML code')

   will result in:
                      Output Field
                      HTML code
```

Notes:

1. Merging of HTML and DDS fields does not occur for input fields. Merging occurs only for output fields.
2. HTML tags are inserted into the data stream if the device query indicates that the device is an i5/OS 5250 Workstation Gateway virtual terminal. Otherwise for normal displays, the HTML tags are ignored.

Programming Examples

You can add the IMG HTML keyword to an existing display file and show a graphic image along with the display as the following examples show.

Figure 95 shows an example of the DDS before adding HTML keyword

```
|...+....1....+....2....+....3....+....4....+....5....+....6....+....7...
     A
     A  FILE LEVEL KEYWORDS
     A
     A                                      DSPSIZ(24 80 *DS3)
     A                                      CA03(03)
     A
     A
     A          R REC1
     A                                      CA01(01)
     A                                      CA02(02)
     A                                  5 30'Description'
     A                                      DSPATR(HI)
     A                                      DSPATR(UL)
     A                                  5 13'Item'
     A                                      DSPATR(HI)
     A                                      DSPATR(UL)
     A                                  5 65'Price'
     A                                      DSPATR(HI)
     A                                      DSPATR(UL)
     A            FLD001        10A  O  6 13
     A            FLD002        25A  O  6 30
     A            FLD003         6A  O  6 65
     A                                  1 36'Catalog'
```

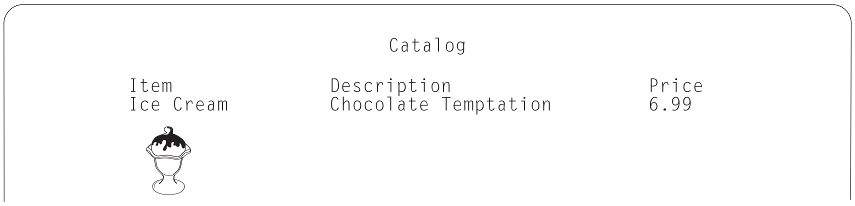
Figure 95. DDS Coding Before Adding HTML Keyword

Figure 96 on page 210 shows an example of the DDS after adding HTML Keyword

```
|...+....1....+....2....+....3....+....4....+....5....+....6....+....7...
     A
     A  FILE LEVEL KEYWORDS
     A
     A                                      DSPSIZ(24 80 *DS3)
     A                                      CA03(03)
     A
     A
     A          R REC1
     A                                      CA01(01)
     A                                      CA02(02)
     A                                  5 30'Description'
     A                                      DSPATR(HI)
     A                                      DSPATR(UL)
     A                                  5 13'Item'
     A                                      DSPATR(HI)
     A                                      DSPATR(UL)
     A                                  5 65'Price'
     A                                      DSPATR(HI)
     A                                      DSPATR(UL)
     A            FLD001        10A  O  6 13
     A            FLD002        25A  O  6 30
     A            FLD003         6A  O  6 65
     A                                  7  2HTML('<img src="http://www.ice.com +
     A                                      /bin/sundae.gif')
     A                                  1 36'Catalog'
```

Figure 96. DDS Coding After Adding HTML Keyword

On i5/OS 5250 Gateway display stations, you would see the following graphic image if you used the DDS source in Figure 96:



```
                              Catalog

          Item              Description                 Price
          Ice Cream         Chocolate Temptation        6.99
```

RV3W133-0

Figure 97. Graphic Image on an i5/OS 5250 Gateway Display

Restrictions

The 5250 Gateway reserves the F1 function key as General Help. F1 cannot be redefined for other uses.

Due to limitations of the web browser, the 5250 Gateway does not support many of the graphic structures that DDS allows.

The following keywords are allowed, but are not supported. Attempting to display them will have unpredictable results.

Windows	(WINDOW keyword)
Menubars	(MNUBAR keyword)
Pulldown	(PULLDN Keyword)
Selection Lists	(SFLMLTCHC/SFLSNGCHC keywords)
Selection Fields	(SNGCHCFLD/MLTCHCFLD keywords)
PushButtons	(PSHBUTTON keyword)
Scrollbars	(SCRBAR keyword)
Continued Entry Fields	(CNTENTFLD keyword)

Chapter 7. Overriding Display Files and Display File Attributes

You can use overrides to temporarily change a file name, a device name associated with the file, or some of the other attributes of a file. Overrides allow you to make minor changes to the way a program functions or to select the data on which it operates without having to recompile the program.

Determining Whether or Not to Use Overrides

The following properties of overrides will help you decide if overrides are appropriate for the task you want to perform:

- Overrides remain in effect only for the job, program, or display station session in which they are issued. They do not permanently change the attributes of a file.
- Overrides have no effect on other jobs that may be running at the same time.
- Overrides that are to be applied must be specified either before the file is opened by a program or before a program that opens the file is compiled.
- Override commands may be entered interactively from a display station or as part of a batch job.
- Override commands may be included in a control language (CL) program, or they may be issued from other programs via a call to the program QCMDEXC.
- Overrides allow you to make minor changes to the way a program functions or for selecting the data on which it operates, without having to recompile the program.

Overriding File Attributes in HLL Programs

File attributes are built as a result of the following:

Create file commands
These commands build file attributes when the file is first created.

Program using the files
At compile time, the user program can specify some of the file attributes. (The attributes that can be specified depend on the high-level language in which the program is written.)

Override commands
At program run time, these commands can override the file attributes previously built by the merging of the file description and the file parameters specified in the user program.

The simplest form of overriding a file in a high-level language (HLL) program is to override some attributes of the file using the Override with Display File (OVRDSPF) command.

Example

You create a display file named DISPLAY33 using the Create Display File command:

```
CRTDSPF FILE(QGPL/DISPLAY33) SRCFILE(DDSFILE1) +
  DEV(STATION1) IGCDTA(*YES) WAITFILE(30) LVLCHK(*NO)
```

Your application program specifies display file DISPLAY33 with STATION50 for the display station and *NO for the IGCDTA parameter, which determines double-byte character processing.

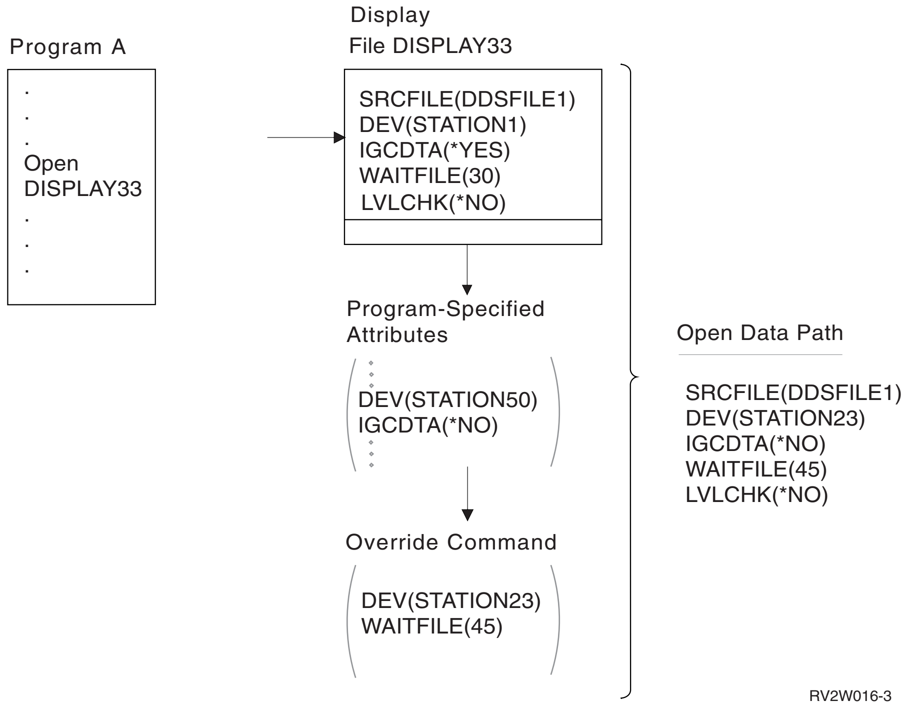
Before you run the application program, you want to change the display station to STATION23 and wait file time to 45 seconds. The override command looks like this:

```
OVRDSPF  FILE(DISPLAY33)  DEV(STATION23) WAITFILE(45)
```

When the application program opens the file, the file overrides, program-specified attributes, and file attributes are merged to form the open data path (ODP), which is used during the running of the program to manage access to the file and to manage its attributes. File overrides have precedence over program-specified attributes. Program-specified attributes have precedence over file-specified attributes. In this example, when the file is opened, the following is used:

- Display station STATION23
- A double-byte character processing value of *NO
- A wait file time of 45 seconds
- The level identifiers of the records formats are not checked when the file is opened.

The following illustration explains the previous example:



Overriding File Names in HLL Programs

Another simple form of overriding a file in a high-level language program is to change the file that is used by the program. This may be useful for files that have been moved or renamed after the program has been compiled. Use the OVRDSPF command for file name overrides also.

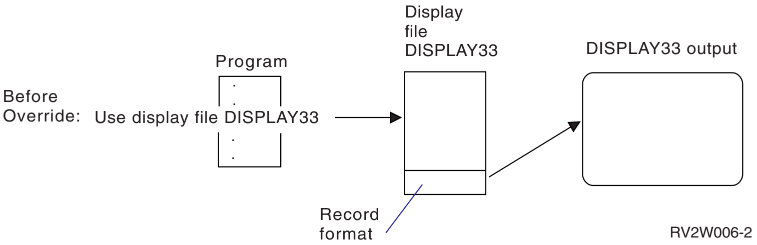
Example

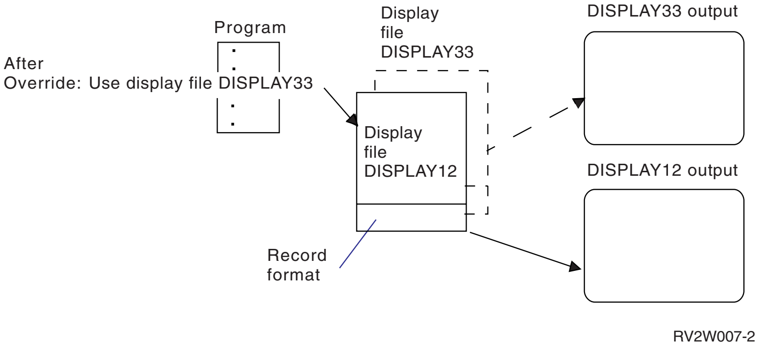
You want the output from your application program to be displayed using the display file DISPLAY12 instead of the display file DISPLAY33 (DISPLAY33 is specified in the application program). Before you run the program, enter the following:

```
OVRDSPF FILE(DISPLAY33) TOFILE(DISPLAY12)
```

The file DISPLAY12 must have been created by a CRTDSPF command before it can be used.

The following illustration explains the previous example:





You may want to override a file with a file that has a different file type; for example, you may want to override a display file with a diskette file using the Override Diskette File (OVRDKTF) command. To determine if your file can be overridden with a file that has a different file type, see the information under "Using File Redirection to Override File Names and Libraries or File Types" on page 217. More information about overriding with different file types is available in the **Files and File Systems** collection in the iSeries Information Center.

Overriding Both File Names and Attributes in HLL Programs

This form of overriding a file in a high-level language program is simply a combination of overriding file attributes and overriding file names or types. With this form of override, you can override the file that is to be used in a program and you can also override the attributes of the overriding file. Use the OVRDSPF command to override both the display file name and attributes.

Example

You want the output from your application program to be displayed using the display file REPORTS instead of the display file OUTPUT. (OUTPUT is specified in the application program.) In addition to having the application program use the display file REPORTS, you wish to override the wait file time to 50 seconds.

Assume the file REPORTS was created with the following command:

```
CRTDSPF FILE(REPORTS) SRCFILE(DDSFILE1) SRCMBR(MEMBER1) WAITFILE(25)
```

Before you run the program, type the following command:

```
OVRDSPF FILE(OUTPUT) TOFILE(REPORTS) WAITFILE(50)
```

Then call the application program. The display file REPORTS is used with a wait file time of 50 seconds.

Note that the single override command used in the previous example is *not* equal to the following two override commands:

Override 1	`OVRDSPF`	`FILE(OUTPUT)`	`TOFILE(REPORTS)`
Override 2	`OVRDSPF`	`FILE(REPORTS)`	`WAITFILE(50)`

Only one override is applied for each call level for an open operation of a particular file. If you want to override the file that is used by the program and also override the attributes of the overriding file from one call level, you must use a single command. If two overrides are used, override 1 causes the output to be displayed using the display file REPORTS, but override 2 is ignored.

Applying Overrides When Compiling a Program

Overrides may be applied at the time a program is being compiled for either of two purposes:

- To select the display file
- To provide external data definitions for the compiler to use in defining the record formats to be used on I/O operations

Overrides to the source file are handled just like any other override. They may select another file, another member of a database file, or change other file attributes.

Overrides may also be applied to files that are used within the program being compiled, if they are being used as externally described files in the program. These files are not opened at compile time, and thus the overrides are not applied in the normal manner. These overrides are used at compile time only to determine the file name and library that will be used to define the record formats and fields for the program to use I/O operations. Any other file attributes specified on the override are ignored at compile time. It is necessary that these file overrides be active at compile time only if the file name specified in the source for the program is not the file name that contains the record formats that the application needs.

The file name that is opened when the compiled program is run is determined by the file name that the program source refers to, changed by whatever overrides are in effect at the time the program runs. The file name used at compile time is not kept. The record formats in the file that is actually opened must be compatible with those used when the program was compiled. Obviously, the easiest way to assure record compatibility is to have the same overrides active at run time that were active at compile time. If your program uses externally described data and a different field level file is used at run time, it is usually necessary to specify LVLCHK(*NO) on the override. See “Using File Redirection to Override File Names and Libraries or File Types” on page 217 for details.

Example

Assume that the source for the program INVENTORY, which has a wait file time of 15 seconds, contains an open to the display file LISTOUT:

```
Override 1    OVRDBF FILE(RPGSRC) TOFILE(SRCPGMS) MAXRCDLEN(77)
Override 2    OVRDSPF FILE(OUTPUT) TOFILE(REPORTS)
                  CALL PGM(A)

                      Program  A
Override 3       OVRDSPF FILE(LISTOUT) TOFILE(OUTPUT)
Override 4       OVRDBF FILE(RPGSRC) WAITFILE(30)
                      CRTRPGPGM PGM(INVENTORY) SRCFILE(RPGSRC)
                      RETURN

Override 5    OVRDSPF FILE(LISTOUT) TOFILE(REPORTS) IGCDTA(*YES)
                   CALL PGM(INVENTORY)
```

The program INVENTORY opens the display file REPORTS in place of display file LISTOUT and allows DBCS data.

The program INVENTORY is created (compiled) from database file SRCPGMS and allows a maximum record length of 77 characters. Override 4 (applied first) overrides an optional file attribute. Override 1 (applied last) causes the file RPGSRC to be overridden with the database file SRCPGMS and a maximum record length of 77 characters.

The program INVENTORY is created with the display formats from the file REPORTS. Override 3 (applied first) causes the file LISTOUT to be overridden with OUTPUT. Override 2 (applied last) overrides OUTPUT with REPORTS. Other attributes may be specified here, but it is not necessary because only the record formats are used at compile time.

At run time, override 3 is no longer active, because program A has ended. Therefore override 2 has no effect on LISTOUT. However, override 5, which is active at run time, replaces LISTOUT with REPORTS and allows DBCS data. Because the same file is used for both compilation and run-time, level checking may be left on.

Deleting Overrides

If you want to delete an override, you can use the Delete Override (DLTOVR) command.

If you use the DLTOVR command in an application that either calls or transfers control to other programs, the override may or may not be deleted. More information about deleting overrides in application programs is available in the **Files and File Systems** collection in the iSeries Information Center.

Displaying Overrides

You can display all file overrides or file overrides for a specific file using the Display Override (DSPOVR) command.

If you use the DSPOVR command to display the overrides used by an application that either calls or transfers control to other programs, you can control which overrides are displayed. More information about displaying overrides used in application programs is available in the **Files and File Systems** collection in the iSeries Information Center.

Using File Redirection to Override File Names and Libraries or File Types

File redirection refers to using overrides to change the file name and library or the type of the file to be processed. For example, you can substitute one display file for another or change from using an ICF file to using a display file. The system may or may not support file redirection. Refer to “Recognizing Commands That Ignore or Restrict Overrides” on page 220 for rules on how the system processes overrides.

Overriding Files with the Same File Types

When you replace the file that is used in a program with another file of the same type, the new file is processed in the same manner as the original file. If a field level file, or any other file containing externally described data is redirected, it usually is necessary to either specify LVLCHK(*NO) or recompile the program. With level checking turned off, it is still necessary that the record formats in the file be compatible with the records in the program. If the formats are not compatible, the results cannot be predicted.

Overriding Files with Different File Types

If you change to a different type of file, the device-dependent characteristics are ignored and records are read or written sequentially. Some device parameters must be specified in the new device file or the override. Defaults are taken for others. The effect of specific redirection combinations is described later in this section.

Any attributes specified on overrides of a different file type than the final file type are ignored. The parameters SPOOL, SHARE, and SECURE are exceptions to this rule. They will be accepted from any override applied to the file, regardless of device type.

Some redirection combinations present special problems due to the specific characteristics of the device. In particular:

- File redirection is not recommended for save files.
- Display files and ICF files that use multiple devices (MAXDEV or MAXPGMDEV > 1) can be redirected only to a display file or ICF file.
- Redirecting a display file to any other file type, or another file type to a display file, requires that the program be recompiled with the override active if there are any input-only or output-only fields. This is necessary because the display file omits these fields from the record buffer in which they are not used, but other file types do not.

Table 22 summarizes valid file redirections:

Table 22. File Redirections

	From-File					
To-File	**Printer**	**ICF**	**Diskette**	**Display**	**Database**	**Tape**
Printer	O*	O	O	O	O	O
ICF	O	I/O O I	O I	I/O O I	O I	O I
Diskette	O	O I	O I	O I	O I	O I
Display	O	I/O O I	O I	I/O O I	O I	O I
Database	O	O I	O I	O I	O I	O I
Tape	O	O I	O I	O I	O I	O I

Legend:

I=input file

O=output file

I/O=input/output file

*=redirection to a different type of printer

To use this chart, identify the file type to be overridden in the FROM-FILE columns and the file type overriding in the TO-FILE column. The intersection specifies an I or O or both, meaning that the substitution is valid for these two file types when used as input files or as output files.

For instance, you can override a diskette output file with a tape output file, and a diskette input file with a tape input file. The chart refers to file type substitutions only. That is, you cannot change the program function by overriding an input file with an output file.

The following chart describes the specific defaults taken and what is ignored for each redirection combination involving display files:

Table 23. File Redirection Combinations

From	To	Specific Defaults Taken and What Is Ignored
Printer	Display	Records are written to the display with each record overlaying the previous record. For program-described files, you can request each record using the Enter key. Printer control information is ignored.
ICF input	Display	Records are retrieved from the display one at a time. Type in the data for each record and press the Enter key when the record is complete.
ICF output	Display	Records are written to the display with each record overlaying the previous record.
ICF input/output	Display	Input records are retrieved from the display one at a time. Type in the data for each record and press the Enter key when the record is complete. Output records are written to the display with each record overlaying the previous input or output record. Input and output records are essentially independent of each other and may be combined in any manner.
Diskette input	Display	Records are retrieved from the display one at a time. Type in the data for each record and press the Enter key when the record is complete. A non-field-level display file must be specified. Diskette label information is ignored.
Diskette output	Display	Records are written to the display with each record overlaying the previous record. You can request each output record using the Enter key.
Display input	ICF	Records are retrieved from the ICF file one at a time.
	Diskette	Records are retrieved in sequential order. Diskette label information must be provided in the diskette file or on an override command.
	Database	Input records are retrieved.
	Tape	Records are retrieved in sequential order. Tape label information must be specified in the tape file or on an override command.
Display output	ICF	Records are written to the ICF file one at a time.
	Database	Records are written to the database in sequential order.
	Diskette	The amount of data written on diskette is dependent on the exchange type of the diskette. Diskette label information must be provided in the diskette file or on an override command.
	Tape	Records are written on tape in sequential order. Tape label information must be specified in the tape file or on an override command.
	Printer	Records are printed and folding or truncating is performed as specified in the printer file.
Display input/output	ICF	Input records are retrieved from the
Database input (sequentially processed)	Display	Records are retrieved from the display one at a time. Type in the data for each record and press the Enter key when the record is complete. A non-field-level display file must be specified.
Database output (sequentially processed)	Display	Records are written to the display with each record overlaying the previous record. You can request each output record using the Enter key.
Tape input	Display	Records are retrieved from the display one at a time. Type in the data for each record and press the Enter key when the record is complete. A non-field-level display file must be specified. Tape label information is ignored.
Tape output	Display	Records are written to the display with each record overlaying the previous record. You can request each output record using the Enter key.

Recognizing Commands That Ignore or Restrict Overrides

The following commonly used commands ignore overrides entirely:

- ADDLFM
- ADDPFM
- ALCOBJ
- APYJRNCHG
- CHGOBJOWN
- CHGPTR
- CHGSBSD
- CHGXXXF (all change file commands)
- CLRPFM
- CLRSAVF
- CPYIGCTBL
- CRTDKTF
- CRTDUPOBJ
- CRTAUTHLR
- CRTSBSD
- CRTTAPF
- DLCOBJ
- DLTF
- DLTAUTHLR
- DSPDBR
- DSPFD
- DSPFFD
- DSPJRN
- EDTOBJAUT
- EDTDLOAUT
- ENDJRNPF
- GRTOBJAUT
- INZPFM
- MOVOBJ
- RGZPFM
- RMVJRNCHG
- RMVM
- RNMOBJ
- RSTUSRPRF
- RVKOBJAUT
- SAVCHGOBJ
- SAVLIB
- SAVOBJ
- SAVPGMPRD
- SAVSAVFDTA
- SAVSYS
- SBMDBJOB

- SIGNOFF
- STRDBRDR
- STRJRNPF

Overrides are not applied to any system files that are opened as part of an end-of-routing step or end-of-job processing. For example, overrides cannot be specified for the job log file. In some cases, when you need to override something in a system file, you may be able to change it through a command other than an override command. For example, to change the output queue for a job log, the output queue could be changed before sign-off using the OUTQ parameter on the Change Job (CHGJOB) command to specify the name of the output queue for the job. If the printer file for the job log contains the value *JOB for the output queue, the output queue is the one specified for the job.

The SRCFILE and SRCMBR parameters on the following commands are affected by overrides: overrides for the SRCFILE

- CRTCMD
- CRTICFF
- CRTDSPF
- CRTLF
- CRTPF
- CRTPRTF
- CRTSRCPF
- CRTTBL
- CRTXXXPGM (All create program commands. These commands also use overrides to determine which file will be opened by a compiled program. See “Applying Overrides When Compiling a Program” on page 216 for more information.)

The OPNQRYF command is affected by the following override parameters: TOFILE, MBR, SEQONLY, LVLCHK, and INHWRT.

The following commands allow overrides, but do not allow changing the MBR to *ALL:

- CPYFRMPCD
- CPYTOPCD

The following commands do not allow overrides to be applied to the display files they use. Overrides to the printer files they use should not change the file type or the file name. Various restrictions are placed on changes that may be made to printer files used by these commands, but the system cannot guarantee that all combinations of possible specifications will produce an acceptable report.

DMPOBJ and DMPSYSOBJ
In addition to the preceding limitations, these commands do not allow overrides to the file they dump.

DSPXXXXXX
All display commands. The display commands that display information about a file do not allow overrides to that file.

DSPIGCDCT and EDTIGCDCT
Message file can be overridden.

GO

PRTXXXXXX
All print commands.

QRYDTA

TRCXXX
All trace commands.

WRKXXXXXX
All work-with commands.

Chapter 8. Handling Messages and Errors for Display Files

This chapter covers the following:

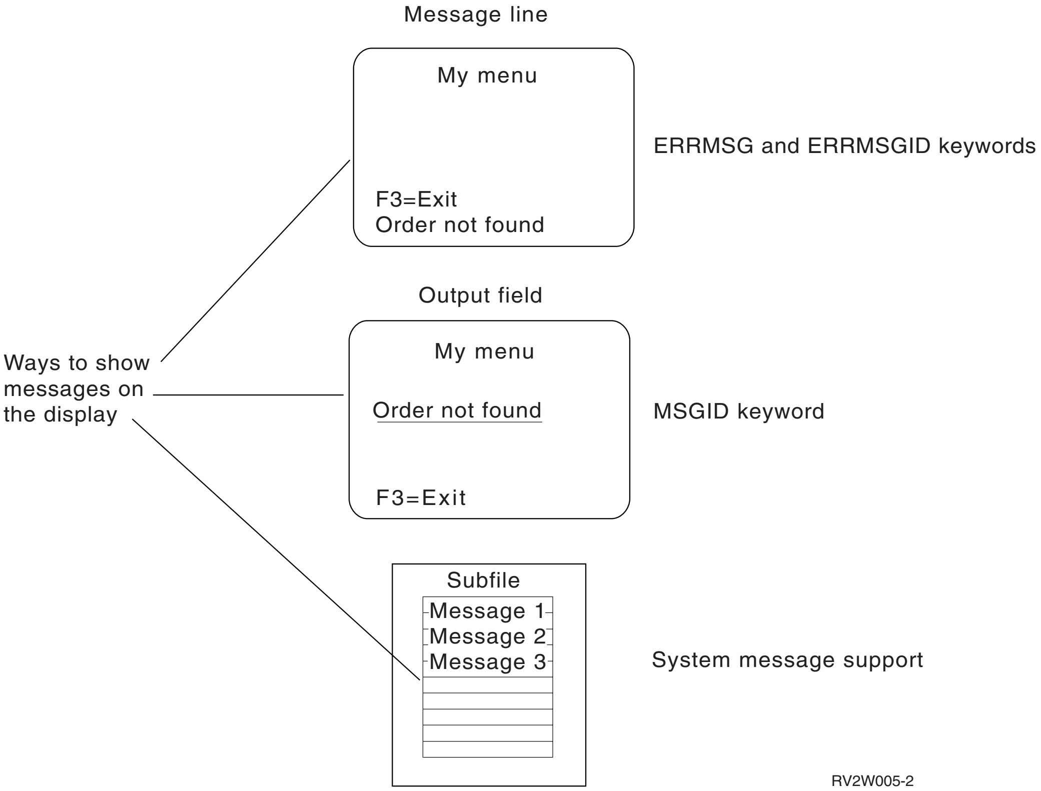
- Creating and displaying your own messages
- Analyzing error messages sent from the system

Creating and Displaying Your Own Messages

You can create and display your own messages on i5/OS. These messages may indicate that a processing error has occurred, that incorrect input has been entered, or simply that the keyboard is temporarily locked while the system processes a lengthy request.

You can specify the following message handling functions for display files:

- Display a message on the message line
- Display a message on the message line when a subfile control record is written
- Define a message line
- Display messages in a field on the display
- Display messages in a program message queue



You can also do the following:

- Display error messages on the message line using a system-supplied error subfile
- Have the system automatically handle jobs that are about to receive a permanent I/O error

Information about DDS keywords

This section uses DDS keywords to define and display messages. For more information about specific DDS keywords, see the **DDS** topic in the iSeries Information Center.

Displaying a Message on the Message Line

You can specify that a message is to be displayed on the message line with the ERRMSG or ERRMSGID keyword. For a message defined as a constant, use the ERRMSG keyword; message help is not supported for these kinds of messages. For a message defined in a message file, use the ERRMSGID keyword. When you use these keywords, the keyboard locks and the user must press the Reset key to clear the message from the display and continue.

When you use ERRMSG, the record that you want to present the message for must already be on the display. If it is not, the ERRMSG function is not performed.

When you use the ERRMSG keyword to present a message, that message is written to the message line of the display, which is usually the line at the bottom of the screen, depending on the software you use to connect your server to your clients. The user would then press the Reset key to clear the message from the display and unlock the keyboard to continue typing. You provide the text of the message right on the ERRMSG keyword. When you write a record that has the ERRMSG keyword in effect, it causes that message to appear. Typically, you would use an option indicator to cause the ERRMSG keyword to take effect. When the application program turns on an option indicator, the keywords that have that option indicator specified in the DDS then take effect. In this case, an application program would leave an indicator that optioned an ERRMSG keyword off until the message needed to be displayed.

The ERRMSGID and SFLMSGID keywords have an optional parameter for message data (msg-data). This allows you to define a program-to-system field that contains the message data (substitution text). For more information on how the message data parameter works, refer to the Send Program Message CL command in the **CL** topic in the iSeries Information Center.

Note: When you use the ERRMSG or ERRMSGID keyword, you should specify RSTDSP(*YES) on the CRTDSPF or CHGDSPF command; otherwise, data may be lost if the display file is suspended.

Displaying a Message on the Message Line When a Subfile Control Record is Written

You can specify that messages are to be displayed on the message line when a subfile control record is written using the SFLMSG or SFLMSGID keyword. If the message is a constant, use the SFLMSG keyword; if it is defined in a message file, use the SFLMSGID keyword. The restrictions on these keywords are the same as ERRMSG and ERRMSGID.

Displaying a Message on the Message Line Using a Message Field

You can specify a message field (M in position 38). The value from this output field will appear on the message line. The value in the field is specified by the application program in the output buffer. The length of this field should not exceed 78 positions if the message is to be displayed on a 24 by 80 screen, or 130 positions if the message is to be displayed on a 27 by 132 screen. Message help and substitution variables are not supported for a message line.

Priorities for Displaying Messages on a Message Line

The message displayed on the message line is determined by the following order of priority (1 is the highest):

1. ERRMSG

2. ERRMSGID
3. SFLMSG
4. SFLMSGID
5. Position 38 in DDS is M

In addition, you can change the line on which messages are displayed by using the MSGLOC keyword. If not specified otherwise, the message line is the last line on the display. If you use the MSGLOC keyword, the new message line applies to messages that are displayed for validity-checking errors and keys that are not valid as well as to user-defined messages.

Notes:

1. If the MSGLOC keyword is not specified, and a display station capable of displaying 27 lines is attached to a local display station controller or attached to a remote 5294 or 5394 controller, the default values are:
   - Line 28 for the 27 by 132 screen size
   - Line 25 for the 24 by 80 screen size
2. If line 25 is specified for the 24 by 80 mode, either because the default was used or line 25 was specified in the MSGLOC keyword, the message actually appears on line 24 unless the display is capable of displaying the message on line 25.
3. The normal system display for message help gives the user access to extended help (by pressing the Help key again), and may allow the user to use F10 to display all messages in the job log. The message help display used for a display station other than the job requester display station or for a display station associated with a multi-display file does not provide these functions.

Displaying Messages in a Field on the Display

You can specify that messages are to be displayed in a field on the display using the MSGID keyword. The message is truncated if it is longer than the MSGID field. The message is padded with blanks if it is shorter than the MSGID field.

When the MSGID keyword is used, the keyboard is not locked because the field on which it is specified is a normal output-capable field. Message help and substitution variables are not supported for the MSGID keyword.

Displaying Messages on a Program Message Queue

You can specify that messages that are contained on a program message queue are to be displayed by using the SFLMSGRCD, SFLMSGKEY, and SFLPGMQ keywords. When you use subfile support for messages, you can display more than one message at a time and the keyboard does not lock while the message is being displayed. Because the messages specified here are from the program message queue, both message help and substitution text are supported. See “Displaying Error Messages from Subfiles” on page 105 for more information. See also the Messages section in the **CL** topic in the iSeries Information Center for information on message queues and sending messages.

Displaying Error Messages through a Subfile

The ERRSFL (error subfile) keyword can be used to indicate that the error messages associated with ERRMSG, ERRMSGID, SFLMSG, and SFLMSGID keywords will be displayed on the message line using a system-supplied error subfile.

The ERRSFL keyword can be used in addition to the ERRMSG and ERRMSGID or SFLMSG and SFLMSGID keywords to allow a user to roll through a subfile of error messages. One error message is displayed at a time. The user’s program handles the validity checking of the fields, setting on the option indicators for the appropriate message to be sent for the fields in error. The system handles putting the message associated with the field in error in the error subfile, and displaying the error messages.

The ERRSFL keyword can also be used to put error messages into an error subfile when the system handles validity checking. Error messages are put into the error subfile when input fails the validity check for the following reasons:

- DDS keywords CHECK(M10 M11 VN VNE), COMP, RANGE, and VALUES
- Floating point operations
- More than one decimal point is entered in a field that has one or more decimal positions
- Either too many or too few decimal positions are entered in a field that has one or more decimal positions

The messages can be viewed by paging through the error subfile. Only one error message is displayed at a time. When the error subfile is displayed, the keyboard is not locked. It is not necessary to press the Reset key prior to correcting the fields in error.

If there is a record format currently displayed that covers the line defined to be the message line, the ERRSFL keyword is ignored.

When both validity checking and ERRMSG/ERRMSGID or SFLMSG/SFLMSGID are used on the screen at the same time, the resulting errors are not all present in the error subfile at the same time. When the write operation is done, the messages from ERRMSG, ERRMSGID, SFLMSG, and SFLMSGID are present in the error subfile. After data is typed and validity checking errors occur, the error subfile is cleared and only the validity checking errors are present.

Following is an example of how the ERRSFL keyword can be used:

```
|...+....1....+....2....+....3....+....4....+....5....+....6....+....7...
     A                                      MSGLOC(24)
     A                                      ERRSFL
     A          R RCD1
     A            FIELD1         5A  B  2  3
     A  10                                  ERRMSGID(MSG0001 MSGF1 10 &MDTA1)
     A  11                                  ERRMSG('Error Msg 1' 11)
     A            FIELD2        10A  B  3  3
     A  20                                  ERRMSG('Error Msg 2' 20)
     A  21                                  ERRMSGID(MSG0002 MSGF1 21)
     A  22                                  ERRMSGID(MSG0003 MSGF1 22 &MDTA3)
     A            FIELD3         2A  B  4  3
     A  30                                  ERRMSGID(MSG0004 MSGF1 30)
     A            MDTA1         78A  P
     A            MDTA3         78A  P
     A
     A
     A
     A
     A
     A
```

Figure 98. Sample DDS Source for ERRSFL Keyword

In this example, assume that RCD1 is currently on the display. When validating the input data, your program detects several errors and sets on the option indicators 11, 21, and 30. On the subsequent output operation:

- FIELD1, FIELD2, and FIELD3 are displayed in reverse image.
- The cursor is located in position 2, 3 (start of FIELD1).
- The keyboard is not locked.
- An error subfile is displayed on line 24. The subfile contains three records: Error Msg 1, the MSG2 message, and the MSG4 message. The user can page through the messages, and by placing the cursor on the message line and pressing the Help key, view the message help for MSG2 or MSG4. Message

CPF9897, which indicates that no help information is available, is displayed if the cursor is not located on the message line when the user is attempting to view the message help of MSG2 or MSG4.

- If the Help key is pressed on the message line for Error Msg 1, the message help for message CPF9897 appears. The message help explains that no message help is available.
- On the subsequent display of RCD1, the error subfile is cleared of the previous error messages.

The following is a DDS example of defining the SFLMSGID keyword in a display file:

```
|...+....1....+....2....+....3....+....4....+....5....+....6....+....7...
     A                                      MSGLOC(24)
     A                                      ERRSFL
     A          R SFLRCD1                   SFL
     A            SFL1           5A  B  2  3
     A          R CTLRCD1                   SFLCTL(SFLRCD1)
     A                                      SFLPAG(1)
     A                                      SFLSIZ(15)
     A                                      SFLDSP SFLDSPCTL
     A  10                                  SFLMSGID(SFL0001 USERMSGF 10)
     A  11                                  SFLMSG('Error Msg 1' 11)
     A          R SFLRCD2                   SFL
     A            SFL2           7A  O  4  3DFT('FIELD 2')
     A          R CTLRCD2                   SFLCTL(SFLRCD2)
     A                                      SFLPAG(1)
     A                                      SFLSIZ(15)
     A                                      SFLDSP SFLDSPCTL
     A  20                                  SFLMSG('Error Msg 2' 20)
     A  21                                  SFLMSGID(SFL0002 USERMSGF 21)
     A
     A
```

Figure 99. Sample DDS Source for SFLMSGID Keyword

Assume that CTLRCD1 and CTLRCD2 are on the display. When validating the input data, your program detects several errors and sets on the option indicators 11 and 21. On the subsequent output operation:

- The cursor is located in position 2, 3 (start of SFLRCD1).
- The keyboard is not locked.
- An error subfile is displayed on line 24. The subfile contains two records: Error Msg 1 and the message of SFL0002. The user can page through the messages, and by placing the cursor on the message line and pressing the Help key, view the message help for SFL0002. If the Help key is pressed on the message line for Error Msg 1, the message help for message CPF9897 appears, which explains that no help information is available.
- On the subsequent display of CTLRCD1 and CTLRCD2, the error subfile is cleared of the previous error messages.

The SFLEND keyword is specified on the ERRSFL subfile control record. An error is issued and the keyboard is locked if the user attempts to roll beyond the top or bottom of the file.

The following considerations apply to the ERRSFL keyword, except when used with the ERRMSG and ERRMSGID keywords and the SFLMSG and SFLMSGID keywords:

- When the error subfile is displayed (that is, there are one or more fields in error), pressing a Roll key results in the error subfile being rolled, regardless of the cursor position at the time or whether other subfiles are currently displayed.
- When processing errors from validity check, the error subfile is built every time a valid command key, Roll key, or Enter key is pressed and errors occur.
- When a valid command key or Enter key is pressed, the message for the first field in error is always displayed.

- If the error subfile is rolled and the field that corresponds to the message that was displayed when the Roll key was pressed is valid, the error subfile is displayed with the message of the first field in error.
- If the error subfile is rolled and the field that corresponds to the message that was displayed when the Roll key was pressed is not valid, the error subfile is rolled to the next field in error.
- When rolling through the error subfile, the cursor is positioned in the field corresponding to the error message being displayed.
- If no fields are in error and a Roll key is pressed, a roll is attempted on the subfile and a roll error message is displayed (assuming no other area that can be rolled is encountered first). See Chapter 4, “Displaying Groups of Records Using Subfiles,” on page 87 for more information about rolling subfiles.
- If the program writes a record and the user presses a Roll key after typing data that is not valid, the error subfile is built and the error message for the first field in error is displayed.

Sounding an Alarm for Messages

To sound an audible alarm when an active ERRMSG, ERRMSGID, SFLMSG, or SFLMSGID keyword is detected or when a validity check message is displayed, specify the MSGALARM keyword. This keyword is allowed at the file or record level. The alarm sounds for only a short time.

The MSGALARM keyword can be used on any record in a display file, including subfile control records. The MSGALARM keyword can also be used with the ERRSFL keyword.

Note: If the MSGALARM and ALARM keywords are specified and active on the same record format, the alarm sounds only once.

If the SFLMSG or SFLMSGID keyword is active and both the MSGALARM and ALARM keywords are specified, an alarm is sounded. If no message is displayed, the ALARM keyword and not the MSGALARM keyword is responsible for sounding the alarm.

Automatically Handling Permanent I/O Errors on Display Stations

Permanent I/O errors on display stations can be handled automatically when you use the device-recovery-action job attribute. The Device Recovery Action (DEVRCYACN) parameter of the Change Job (CHGJOB) command indicates a system action for jobs that are about to receive a permanent I/O error (such as a display station being powered off). You can use the following parameter values:

***DSCMSG**
When the I/O error occurs, the system runs a Disconnect Job (DSCJOB) command to suspend the job. The user remains disconnected until signing on again to the same display. The job is resumed at the point just after the I/O operation. Display data management sets the major/minor return code to 83E1 and sends a CPF509F message to the program message queue. These indicate that the display has been cleared and is again available for use.

***DSCENDRQS**
This option also runs the DSCJOB command when an I/O error occurs, but when the job is resumed, the system runs the End Request (ENDRQS) command to give control to the most recent request-processing program.

***ENDJOB**
When an I/O error occurs, the job ends automatically and its priorities are lowered to conserve system resources.

***ENDJOBNOLIST**
Again, the job ends automatically and its priorities are lowered to conserve system resources. However, no job log is produced for the job.

When an I/O error occurs (especially in the case of a communications line failing for several active jobs at once), using the device-recovery-action job attribute can save system resources. Also, the device-recovery-action job attribute makes the job of coding for I/O errors easier. Once the display station is recovered, the user application can continue at the point the error occurred by using the *DSCMSG and

*DSCENDRQS values. For *DSCMSG, if the major/minor return code is 83E1 after an I/O operation, the program needs to assume that the display is blank. The program must branch back to the point at which the first I/O operation to the display is done. For *DSCENDRQS, the request-processing program receives control when connecting to the job again. This is similar to selecting option 2 on the System Request menu. No error recovery is performed for *MSG or *DSCMSG until an I/O operation is performed to the device in error. For all other values, the recovery occurs immediately when the error occurs.

The device-recovery-action job attribute works only for *REQUESTER display stations and does not apply to batch jobs, pass-through jobs, or work-station-function jobs.

Analyzing Error Messages Sent from the System

This section describes error conditions that an application program may encounter during its operation and the provisions that can be made within the program itself to attempt to deal with these conditions. The Debugging CL programs and procedures section in the **CL** topic in the iSeries Information Center discusses how to use the debug functions to resolve unexpected errors encountered in the application programs. Also, the **Troubleshooting** topic describes the programs that are available for analyzing and reporting system errors and hardware failures.

Errors can be detected when a file is opened, when a program device is acquired or released, during I/O operations to a file, and when the file is closed. When appropriate, the system will automatically try to run a failing operation again, up to a retry limit. When a retry is successful, neither operator nor program action is required. Errors that can affect the processing of the program may be reported in any or all of the following ways:

- A notify, status, diagnostic, or escape message may be sent to the program message queue of the program using the file. These messages may also appear in the job log, depending on the message logging level set for the job.
- A notify, status, diagnostic, or escape message may be sent to the operator message queue (QSYSOPR) or the history message queue (QHST).
- A file status code may be returned by the high-level language.
- A major/minor return code is returned in the I/O feedback area for display files.
- Information regarding the error may be saved in the system error log for use by the problem analysis and resolution programs.
- An alert message may be sent to an operator at another system in the network.
- The normal program flow may be interrupted and control may be transferred to an error-handling subroutine, or other language operations may occur. For additional information about how to handle run-time errors, see the appropriate high-level language manual.

Only some of these are significant to a program that is attempting error recovery.

Not all file errors allow programmed error recovery. Some errors are considered permanent; that is, the file, device, or program cannot work until some corrective action is taken. This might involve resetting the device by varying it off and on again, or correcting an error in the device configuration or the application program. Some messages and return codes are used to inform the user or the application program of conditions that are information rather than errors, such as change in the status of a communications line, or system action taken for an unexpected condition. In many cases, it is possible for the application program to test for an error condition and take some preplanned recovery action which allows the program to continue without intervention from the user.

Understanding Messages and Message Monitors

Displayed messages are the primary source of information for anyone who is testing a new application. A message usually contains more specific information than the file status code, the indicators, or the major/minor return code. The control language allows messages to be monitored so that the CL program can intercept a message and take corrective action. See the Messages section in the **CL** topic in the iSeries

Information Center for more information about message types and message monitors. In most high-level languages, either the file status code or major/minor return code (described in the following section) is a more convenient source of information.

Message numbers are assigned in categories to make it easier for a program to monitor for any of a group of related messages. The following message number ranges are assigned for file error messages:

Table 24. System Message Number Ranges

Message IDs	Operation	Message Type
CPF4001-40FF	Open	Diagnostic and status.
CPF4101-43FF	Open	Escapes that make the file unusable.
CPF4401-44FF	Close	Diagnostic and status.
CPF4501-46FF	Close	Escapes that make the file unusable.
CPF4701-48FF	I/O, Acquire, and Release	Notify with a default reply of cancel, status and escapes that do not make the file or device unusable.
CPF4901-49FF	I/O, Acquire, and Release	Notify with a default reply of ignore.
CPF5001-50FF	I/O, Acquire, and Release	Notify with a default reply of cancel.
CPF5101-53FF	I/O, Acquire, and Release	Escapes that make the file or device unusable.
CPF5501-56FF	I/O, Acquire, and Release	Escapes that make the file or device unusable.

Some status messages, CPF4018 for example, are preceded by a diagnostic message that provides additional information. Diagnostic messages may be kept in the job log, depending on the message logging level of the job. If a CL program monitors for CPF4018, CPF5041, or similar messages, it can retrieve the accompanying diagnostic message from the program message queue.

If an error occurs for which an escape message is issued and the message is not monitored, your program will be ended and the message displayed in the operator message queue. Status messages may also be monitored, but if they are not monitored, the program continues. Most high-level languages except CL monitor for all the file errors that are likely to be encountered, and provide some standard recovery. Depending on the severity of the error, the high-level language may simply end the program and issue a message of its own. Alternatively, you may code an error recovery routine to handle errors that are anticipated in that particular application.

Within these error-handling routines, it is usually necessary to examine the file status or major/minor return codes to determine the cause of the error. The manuals for the language you are using explain how to access file status and major/minor return codes. The language manuals also explain the file status codes as they are defined for each language.

Understanding Major/Minor Return Codes

Major/minor return codes are used to report errors and certain status information for display files. They are usually stated as four characters: the first two referring to the major code and the second two referring to the minor code. The major code indicates the general type of error, and the minor provides further detail. Minor codes, except zero, have the same or a similar meaning, regardless of the major code with which they are combined.

The application program can test the return code after each I/O operation. If the major return code is 00, the operation completed successfully and the minor return code contains status information that indicates whether a read or a write operation should be performed next. A major return code of 04 or above indicates that an error occurred. The program may test for any specific errors for which programmed recovery is attempted. The application program may test for a specific condition by comparing the major and minor codes as a unit, or may identify a class of conditions by testing the major code alone.

Most major/minor return codes are accompanied by any one of several message numbers, for which the typical recovery action is similar. File status codes are defined by the individual languages and may be set based on the major/minor return codes.

The following table defines the major return codes. Appendix D, "Display File Return Codes" contains specific definitions of the major and minor return codes as they are used for display files and the message numbers associated with each.

Table 25. Major Return Code Definitions

Code	Definition
00	The operation requested by your program completed successfully. The minor includes state information, such as change direction.
02	Input operation completed successfully, but job is being ended (controlled). The minor includes state information.
03	Successful input operation, but no data was received. The minor includes state information.
04	Error occurred because an output operation was attempted while data was waiting to be read.
08	An acquire operation failed because the device has already been acquired or the session has already been established.
11	A read-from-invited-program-devices operation failed because no device or session was invited.
34	An input exception occurred. The data length or record format was not acceptable for the program.
80	A permanent system or file error, which cannot be recovered from, occurred. Programmer action is required to correct the problem.
81	A permanent device or session error, which cannot be recovered from, occurred during an I/O operation.
82	A device or session error occurred during an open or acquire operation. Recovery may be possible.
83	A device or session error occurred during an I/O operation. Recovery may be possible.

Recovering from Errors

The following sections describe the error recovery action that is appropriate for each group of major return codes.

Normal Completion

A major/minor return code of 0000 indicates that the operation requested by your program was completed successfully. Most of the time, no message is issued. In some cases, a diagnostic message might be used to inform the user of some unusual condition that the system was able to handle, but which might be considered an error under some conditions. For example, a parameter that is not valid might be ignored, or some default action taken.

Completion with Exceptions

Several rather specific major return codes have been assigned to conditions for which a specific response from the application program is appropriate.

A major return code of 02 indicates that the requested input operation completed successfully, but the job is being ended (controlled). The application program should complete its processing as quickly as possible. The controlled cancel is intended to allow programs time to end in an orderly manner. If your program does not end within the time specified on the ENDJOB command, the job will be ended by the system without further notice.

A major return code of 03 indicates that an input operation completed successfully without transferring any data. For some applications, this might be an error condition, or it might be expected when the user presses a function key instead of entering data. It might also indicate that all the data has been processed, and the application program should proceed with its completion processing. In any case, the contents of the input buffer in the program should be ignored.

A major/minor code of 0309 is used to indicate that no data was received *and* the job is being ended (controlled). A major/minor code of 0310 indicates that there is no data because the specified wait time has ended. Other minor return codes accompanying the 02 or 03 major code are the same as for a 00 major code, indicating communications status and the operation to be performed next.

A major return code of 04 indicates that an output exception occurred. Specifically, your program attempted to send data when there was data waiting to be received. This is probably the result of not handling the minor return code properly on the previous successful completion. Your program can recover by simply receiving the incoming data and then repeating the write operation.

A major return code of 34 indicates that an input exception occurred. The received data was either too long or incompatible with the record format. The minor return code indicates what was wrong with the received data, and whether the data was truncated or rejected. Your program can probably handle the exception and continue. If the data was rejected, you may be able to read it by specifying a different record format.

Two other return codes in this group, 0800 and 1100, are both usually the result of application programming errors, but are still recoverable. 0800 indicates that an acquire operation failed because the device has already been acquired or the session has already been established. 1100 indicates that the program attempted to read from invited devices with no devices invited. In both cases, the request that is not valid is ignored, and the program may continue.

No message is issued with a 02 major code or most minor codes with the 03 major code, but the other exceptions in this group are usually accompanied by a message in the CPF4701-CPF47FF or CPF5001-CPF50FF range.

Permanent System or File Error

A major return code of 80 indicates a serious error affecting the file. The application program must close the file and reopen it before attempting to use it again, but recovery is unlikely until the problem causing the error is found and corrected. To reset an error condition in a shared file by closing it and opening it again, all programs sharing the open data path must close the file. This may require returning to previous programs in the program stack and closing the shared file in each of those programs. You should refer to the text of the accompanying message to determine what action is appropriate for the particular error.

Within this group, several minor return codes are of particular interest. A major/minor code of 8081 indicates a serious system error for which an APAR probably will be required. The message sent with the major/minor return code may direct you to run the Analyze Problem (ANZPRB) command to obtain more information.

A major/minor code of 80EB indicates that incompatible options or options that are not valid were specified in the display file or as parameters on the open operation. In most cases you can close the file, end the program, correct the parameter that is not valid with an override command, and run the program again. The override command affects only the job in which it is issued. It allows you to test the change easily, but you may eventually want to change or re-create the display file as appropriate to make the change permanent.

Permanent Device or Session Error on I/O Operation

A major return code of 81 indicates a serious error affecting the device or session. This includes hardware failures affecting the device, communications line, or communications controller. It also includes errors due to a device being disconnected or powered off unexpectedly and abnormal conditions that were discovered by the device and reported back to the system. Both the minor return code and the accompanying message provide more specific information regarding the cause of the problem.

Depending on the file type, the program must either close the file and open it again, release the device and acquire it again, or acquire the session again. To reset an error condition in a shared file by closing it and opening it again, all programs sharing the open data path must close the file. In some cases, the message may instruct you to reset the device by varying it off and on again. It is unlikely that the program will be able to use the failing device until the problem causing the error is found and corrected, but recovery within the program may be possible if an alternate device is available.

Some of the minor return codes in this group are the same as those for the 82 major return code. Device or line failures may occur at any time, but an 81 major code occurs on an I/O operation. This means that your program had already established a link with the device or session. Therefore, some data may have been transferred, but when the program is started again, it starts from the beginning. A possible duplication of data could result.

Message numbers accompanying an 81 major code may be in the range indicating either an I/O or a close operation. A device failure on a close operation simply may be the result of a failure in sending the final block of data, rather than action specific to closing the file. An error on a close operation may result in the file being left partially closed. Your error recovery program should respond to close failures with a second close operation. The second close will always complete, regardless of errors.

Device or Session Error on Open or Acquire Operation

A major return code of 82 indicates that a device or session error occurred during an open or acquire operation. Both the minor return code and the accompanying message will provide more specific information regarding the cause of the problem.

Some of the minor return codes in this group are the same as those for the 81 major return code. Device or line failures may occur at any time, but an 82 major code indicates that the device or session was unusable when your program first attempted to use it. Thus no data was transferred. The problem may be the result of a configuration or installation error.

Depending on the minor return code, it may be appropriate for your program to recover from the error and try the failing operation again after some waiting period. The number of times you try should be specified in your program. It may also be possible to use an alternate or backup device or session instead.

Message numbers accompanying an 82 major code may be in the range indicating either an open or an acquire operation. If the operation was an open, it is necessary to close the partially opened file and reopen it to recover from the error. If the operation was an acquire, it may be necessary to do a release operation before trying the acquire again. In either case, the file wait time should be specified long enough to allow the system to recover from the error.

Recoverable Device or Session Errors on I/O Operation

A major return code of 83 indicates that an error occurred in sending data to a device or receiving data from the device. Recovery by the application program is possible. Both the minor return code and the accompanying message provide more specific information regarding the cause of the problem.

Most of the errors in this group are the result of sending commands or data that are not valid to the device, or sending valid data at the wrong time or to a device that is not able to handle it. The application program may recover by skipping the failing operation or data item and going on to the next one, or by substituting an appropriate default. There may be a logic error in the application.

Chapter 9. Creating and Accessing Menus Using Display Files

This chapter describes user-defined menus and tells you how to create them and use them with system menus.

The first two types of user-defined menus can be created using the information in this chapter:

Display file menus
Menus that use a display defined by data description specifications (DDS) to present a menu format. The menu functions are controlled by a message file containing the commands used to run each of the menu options.

Program menus
Menus that use a high-level language program to present the menu format and provide the functions necessary to run the menu options.

UIM menus
Menus that a menu object defined by the user interface manager (UIM) panel group definition language. For more information on defining your own menus using the UIM, see “Defining a Menu Object Using UIM” on page 280.

> **Information about CL commands**
> Some of the examples in this chapter use CL commands. For more information about specific CL commands, see the **CL** topic in the iSeries Information Center.

Running System and User-Defined Menus

Menus that consist of *MENU object types are run using the Go to Menu (GO) command. The GO command allows you to specify either a particular menu or a generic menu name. If you specify a generic name, you are shown the Work with Menus display, which shows all the menus available for your use. From this list, you can choose a menu to run.

Returning to a Menu after Running the GO command

Using the Return Point (RTNPNT) parameter of the GO command, you can specify whether or not you want to return to the menu from which the command was entered after running the menu specified by the GO command.

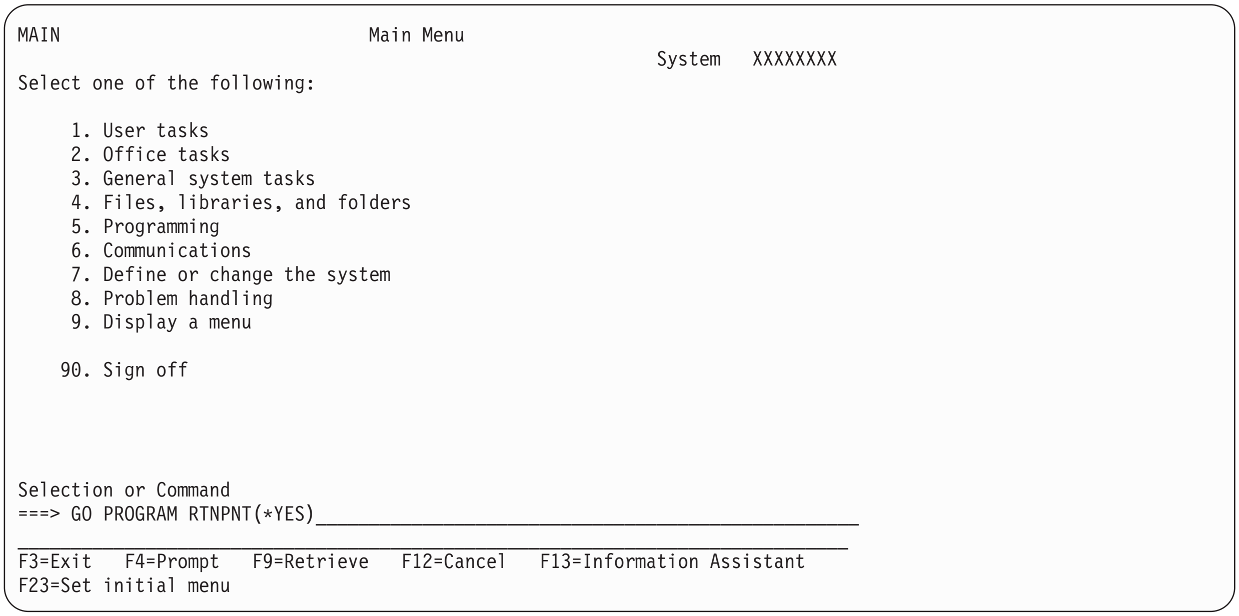
Determining the Previous Menu

A menu is placed on an internal menu stack before it is run. If a stack is not available for the menu, one is created. When the Cancel key is pressed for a menu, the previous menu in the stack is shown. Each menu stack is 10 elements (menus) deep. When the eleventh menu is placed on the menu stack, the first, or oldest, menu is removed from the stack. This menu cannot be returned to by using the Cancel key.

Using the Cancel and Exit Keys on Menus

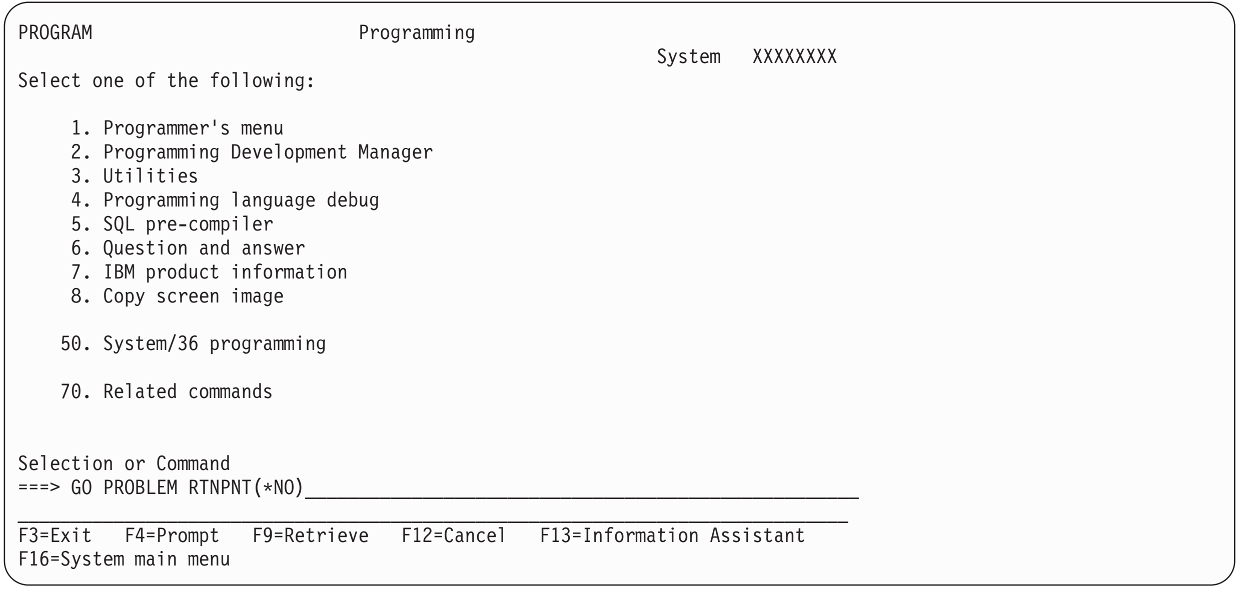
In the following example, a series of menus is presented using both the menu options and the GO command.

1. From the i5/OS Main Menu, `GO PROGRAM RTNPNT(*YES)` is typed on the command line. Specifying RTNPNT(*YES) here means that the Main Menu will be used as a return point.

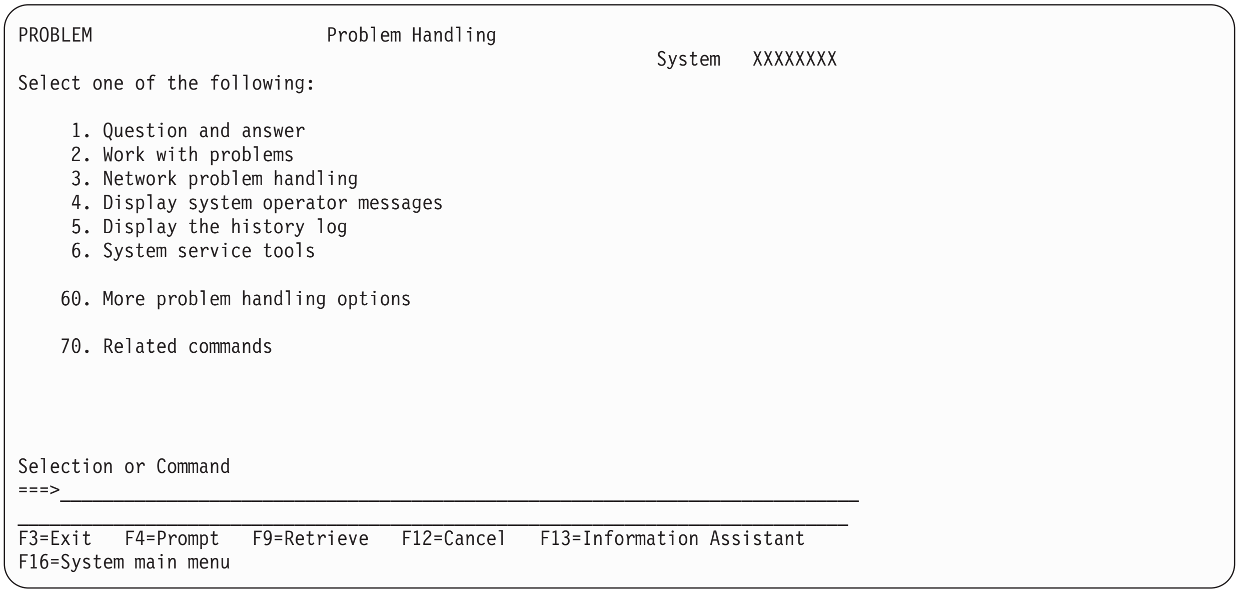


```
MAIN                            Main Menu
                                                           System   XXXXXXXX
Select one of the following:

     1. User tasks
     2. Office tasks
     3. General system tasks
     4. Files, libraries, and folders
     5. Programming
     6. Communications
     7. Define or change the system
     8. Problem handling
     9. Display a menu

    90. Sign off




Selection or Command
===> GO PROGRAM RTNPNT(*YES)___________________________________________________
_______________________________________________________________________________
F3=Exit   F4=Prompt   F9=Retrieve   F12=Cancel   F13=Information Assistant
F23=Set initial menu
```

2. Next, by typing `GO PROBLEM RTNPNT(*NO)` on the command line of the Programming menu and pressing the Enter key, the Problem Handling menu is shown. The Programming menu is *not* set as a return point.



```
PROGRAM                        Programming
                                                           System   XXXXXXXX
Select one of the following:

     1. Programmer's menu
     2. Programming Development Manager
     3. Utilities
     4. Programming language debug
     5. SQL pre-compiler
     6. Question and answer
     7. IBM product information
     8. Copy screen image

    50. System/36 programming

    70. Related commands


Selection or Command
===> GO PROBLEM RTNPNT(*NO)____________________________________________________
_______________________________________________________________________________
F3=Exit   F4=Prompt   F9=Retrieve   F12=Cancel   F13=Information Assistant
F16=System main menu
```



```
PROBLEM                     Problem Handling
                                                           System   XXXXXXXX
Select one of the following:

     1. Question and answer
     2. Work with problems
     3. Network problem handling
     4. Display system operator messages
     5. Display the history log
     6. System service tools

    60. More problem handling options

    70. Related commands




Selection or Command
===>___________________________________________________________________________
_______________________________________________________________________________
F3=Exit   F4=Prompt   F9=Retrieve   F12=Cancel   F13=Information Assistant
F16=System main menu
```

3. In this example, if the Cancel key is pressed while viewing the Problem Handling menu, the user is returned to the Programming menu because the Programming menu is the preceding display on the menu stack. If F3 (Exit) is pressed while viewing the Problem Handling menu, the user is returned to the Main Menu, because the Main Menu is the most recent display from which a GO command was entered with RTNPNT(*YES).

Note that since the default value for the RTNPNT parameter is *YES, `GO PROBLEM` will have the same effect as `GO PROBLEM RTNPNT(*YES)`.

Pressing either F3 (Exit) or F12 (Cancel) while viewing the help for a menu returns the user to the menu itself.

Choosing the Menu That Is Shown at Sign-On Time

To specify a menu as your **initial menu**, or the first menu you see when you sign on, press F23 when the menu is shown. F23 sets the initial menu attribute in your user profile. Unless your initial program requests some other menu, the menu selected by pressing F23 is shown as the initial menu the next time you sign on.

Note: You can only use this key if LMTCPB(*NO), which allows you to change the initial menu, is specified in your user profile.

See the *iSeries Security Reference* for information about initial menus and limited users.

Defining Your Own Display File Menus

Display file menus consist of three parts:

- A DDS-defined display file to define the way the menu looks at the display station
- A message file to define what action is taken when any of the menu options are selected
- A menu object (*MENU) that contains the menu

Help is available on display file menus for the following:

Commands Command help is shown by typing the command on the command line, then pressing the Help key.

Function keys Help for function keys is shown by moving the cursor to the function area and pressing the Help key.

Messages Help for messages is shown by moving the cursor to a message and pressing the Help key.

Help for the menu options may also be provided using DDS. See "Naming Help Formats for Menus" on page 239 for more about help information.

Information about DDS keywords

More information about the specific DDS keywords used in this chapter is found in the **DDS** topic in the iSeries Information Center.

Understanding DDS and Display File Considerations for Menus

Several restrictions exist for the DDS used in defining a menu:

Table 26. Restrictions for Display File Menus

Affected Area	Restriction
Display size	The menu format and the associated help record formats for the menu in the display file must have a display size of 24 rows by 80 columns, DSPSIZ(*DS3) in DDS.
Display file and record format names	The menu record format name must be the same as the display file name.
Help format names	Help format names must follow the convention #Hxxyy, described in "Naming Help Formats for Menus" on page 239. The help record format is limited to 150 items. **Note:** The symbol preceding Hxxyy is a number sign.
Indicator area	A separate indicator area (INDARA keyword) must be declared in the DDS for the display.
Subfile use	Subfiles cannot be used.
Allowed lines	Only the first 21 lines of the display should be described. Only the first 21 lines of the help display should be described. Lines 22 and 23 are used with long command lines; system messages are shown on line 24. (Line 23 is used with short command lines and line 24 is used for messages.) The following lines can be used when a function key legend is requested: • For a long command line with no function key legend, lines 1 through 21 • For a long command line with a function key legend, lines 1 through 19 • For a short command line with no function key legend, lines 1 through 22 • For a short command line with a function key legend, lines 1 through 20 • For no command line with no function key legend, lines 1 through 21 • For no command line with a function key legend, lines 1 through 19
Keyboard locking	The LOCK record level keyword should be used to prevent the keyboard from unlocking before the display is shown.
Paging	The Allow Roll (ALWROL) record level keyword cannot be specified.
Deferring the write operation	DFRWRT(*NO) is *required* on the CRTDSPF command when creating a display file menu. DFRWRT(*NO) ensures the menu format is displayed when the menu is run.

The following suggestions can also be used to help you define your menu using DDS:

Table 27. Suggestions for Display File Menus

Affected Area	Option
Restoring the display	The RSTDSP(*NO) is optional on the CRTDSPF command, but may significantly improve system performance if specified.
Menu and display station alias	An alias can be used for the name of the menu and for the name of the display station (for the System/36™ environment). When issuing the CRTMNU command, you can specify a different name than the name used in the DDS. This way, the name of the display station identifier will be displayed with the menu. By giving the menu a different name, you can use the GO command to find that menu.

Describing Menu Actions in a Message File

Message files are used to describe what action is taken when a menu option is selected. Commands controlling the action to be taken are usually placed in the message. In some cases, the command string may be too long to fit in the message. In these cases, the message contains the message identifier for the message; the command is then placed in the message help.

Messages used in display file menus must be named using the message prefix USR. The remaining four digits of the message identifier correspond to the menu option number.

For example, if option 3 on a given display file menu is:

```
3.  Personnel Menu
```

then the message describing the action taken when 3 is entered on the command line uses identifier USR0003. The message could contain any command, such as `'GO PERSMENU RTNPNT(*NO)'`.

See "Creating and Displaying Your Own Messages" on page 223 for more information about messages.

Naming Help Formats for Menus

Help formats for display file menus are named using the form:

```
#Hxxyy
```

where:

The symbol preceding Hxxyy is a number sign.

`xx` is the number of the first menu option to which the help information applies

`yy` is the number of the last menu option to which the help information applies

For example, `#H0103` could be used to name help that applies to menu options 1 through 3. `#H0202` names the help that applies to menu option 2.

`#H0000` names the extended help for the menu.

The help formats may be described in any order in the DDS. The system sorts the help formats in ascending order when the menu is run.

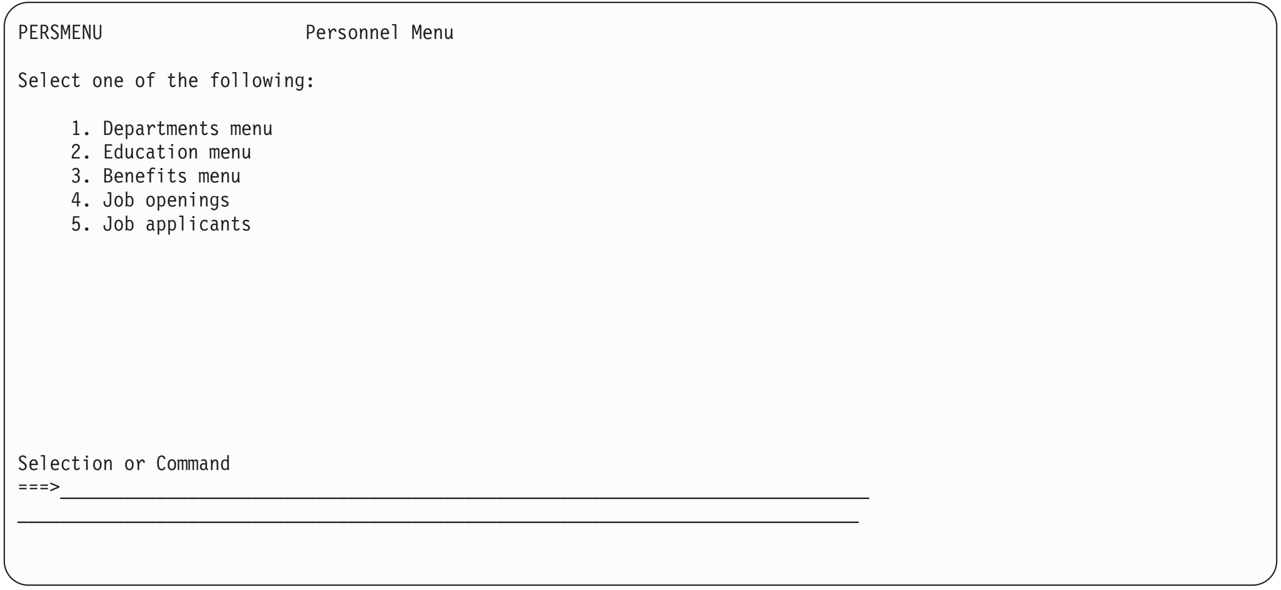
If two or more help format names apply to the same option, only the first help format (as sorted by the system) will be shown when Help is requested for that option. For example, if the following help formats are given:

```
#H0000
#H0101
#H0205
#H0306
#H0707
```

Typing 3 on the command line (to select option 3) and pressing the Help key will show the help designated #H0205. In this example, #H0306 can be viewed using the Page Down key, but will be shown *directly* only by requesting help for option 6.

Similarly, by moving the cursor to the command line and pressing the Help key with no option typed on the command line, the user is shown extended help for the menu (#H0000). The cursor movement keys can be used to look through the other help formats for the menu options.

The following example shows a menu with five options. The names of the help formats and the menu options to which they apply are shown following the example.



```
PERSMENU                   Personnel Menu

Select one of the following:

     1. Departments menu
     2. Education menu
     3. Benefits menu
     4. Job openings
     5. Job applicants











Selection or Command
===>____________________________________________________________________________
________________________________________________________________________________
```

Help Name	Contents
#H0000	General help for the Personnel Menu (PERSMENU)
#H0101	Help for option 1, Departments menu
#H0202	Help for option 2, Education menu
#H0303	Help for option 3, Benefits menu
#H0405	Help for options 4 and 5, Job openings and Job applicants

The help information for the menu options can be shown by typing the option number on the command line and pressing the Help key, or by pressing Help (extended help for the menu), and then using the Page Down key to page through the help for the options.

Building a Display File Menu

The following steps allow you to create a display file (*DSPF) menu:

1. Describing the menu and menu help information
2. Creating the display file
3. Creating the message file
4. Adding messages to the message file
5. Creating the menu
6. Running the menu

The following sections explain each step in detail.

Describing the Menu and Menu Help Information

DDS is used to describe both the appearance and the help for the menu.

The following DDS is for the sample display called PERSMENU. The first record format (PERSMENU) is used to describe the appearance of the menu when displayed. Five menu options are listed, starting in position 7 of lines 5 through 9.

Five more record formats are shown: #H0000, #H0101, #H0202, #H0303, and #H0405. Each of these is used to provide help information for the menu and its options. Following the system convention described above (see "Naming Help Formats for Menus" on page 239), #H0000 is used to provide extended help for the menu. The other record formats provide help for the menu options.

```
|...+....1....+....2....+....3....+....4....+....5....+....6....+....7...
     A* PERSONNEL MENU (PERSMENU) SPECIFICATION
     A*
     A                                      PRINT DSPSIZ(*DS3)
     A                                      INDARA
     A*
     A          R PERSMENU                  LOCK
     A                                  1 02'PERSMENU'
     A                                  1 29'PERSONNEL MENU'
     A                                  3 02'SELECT ONE OF THE FOLLOWING:'
     A                                  5 07'1. DEPARTMENTS MENU'
     A                                  6 07'2. EDUCATION MENU'
     A                                  7 07'3. BENEFITS MENU'
     A                                  8 07'4. JOB OPENINGS'
     A                                  9 07'5. JOB APPLICANTS'
     A                                 21 02'SELECTION OR COMMAND'
     A*
     A          R #H0000                    LOCK
     A                                  1 02'GENERAL'
     A                                  2 26'HELP FOR THE PERSONNEL MENU
     A                                  3 02'THIS IS THE GENERAL HELP'
     A
     A
     A                                  3 27'FOR THE PERSONNEL MENU.'
     A*
     A          R #H0101                    LOCK
     A                                  1 02'HELP1'
     A                                  1 26'HELP FOR OPTION 1'
     A                                  3 02'THIS IS THE HELP'
     A                                  3 19'FOR OPTION 1.'
     A*
     A          R #H0202                    LOCK
     A                                  1 02'HELP2'
     A                                  1 26'HELP FOR OPTION 2'
     A                                  3 02'THIS IS THE HELP'
     A                                  3 19'FOR OPTION 2.'
     A*
     A          R #H0303                    LOCK
     A                                  1 02'HELP3'
     A                                  1 26'HELP FOR OPTION 3'
     A                                  3 02'THIS IS THE HELP'
     A                                  3 19'FOR OPTION 3.'
     A*
     A          R #H0405                    LOCK
     A                                  1 02'HELP45'
     A                                  1 26'HELP FOR OPTIONS 4 AND 5'
     A                                  3 02'THIS IS THE HELP'
     A                                  3 19'FOR OPTIONS 4 AND 5.'
     A*
```

Figure 100. DDS Source for Sample Menu Called PERSMENU

Creating the Display File

To create the display file for the menu, enter the following command:

```
CRTDSPF FILE(PERLIB/PERSMENU) +
        SRCFILE(USERLIB/SFPERS) +
        DFRWRT(*NO) +
        RSTDSP(*NO)
```

The DDS source member, PERSMENU, in the source file SFPERS in library PERLIB, is used to create the display file for the PERSMENU menu.

Creating the Message File

Enter the following Create Message File (CRTMSGF) command to create a message file called PERSMENU (same name as record format):

```
CRTMSGF MSGF(PERLIB/PERSMENU) +
       TEXT('Message file of commands for menu PERSMENU.')
```

The message file will be used to contain messages describing the actions taken when the various menu options are selected.

Adding Messages to the Message File

The Add Message Description (ADDMSGD) command is used to add messages to the message file. The messages describe the different actions taken when various menu options are selected.

The MSGID parameter of each ADDMSGD command is in the form USRxxxx, where xxxx is the menu option number. The MSG parameter contains the command to run the menu option.

In the following example, the MSG parameters of the first three messages contain commands to run menus (using the GO command); the last two messages contain commands which will call programs (using the CALL command) when either of those menu options are selected from the menu.

```
ADDMSGD MSGID(USR0001) MSGF(PERLIB/PERSMENU) +
       MSG('GO DEPTMENU RTNPNT(*NO)')
ADDMSGD MSGID(USR0002) MSGF(PERLIB/PERSMENU) +
       MSG('GO EDUCMENU RTNPNT(*NO)')
ADDMSGD MSGID(USR0003) MSGF(PERLIB/PERSMENU) +
       MSG('GO BENEMENU RTNPNT(*NO)')
ADDMSGD MSGID(USR0004) MSGF(PERLIB/PERSMENU) +
       MSG('CALL JOBOPEN')
ADDMSGD MSGID(USR0005) MSGF(PERLIB/PERSMENU) +
       MSG('CALL JOBAPPS')
```

Creating the Menu Object

Enter the following Create Menu (CRTMNU) command to create the menu object:

```
CRTMNU  MENU(PERLIB/PERSMENU) TYPE(*DSPF) DSPF(*MENU) +
       MSGF(*MENU) DSPKEY(*NO) CMDLIN(*LONG) +
       TEXT('Personnel menu')
```

The DSPKEY parameter specifies whether the function key legend is shown at the bottom of the menu when the menu is displayed. *NO specifies that the function key legend is not shown at the bottom of the menu. *YES specifies that the function key legend is shown at the bottom of the menu. You do not have the option to display only certain function keys.

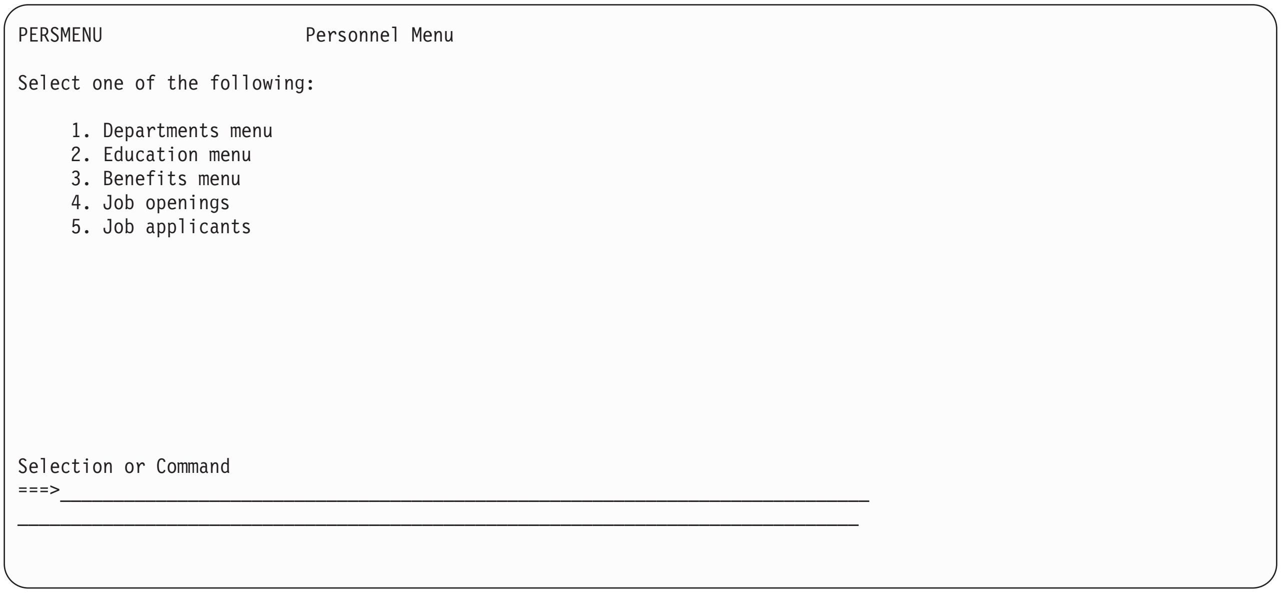
The CMDLIN parameter specifies the length of the command line to be displayed. *LONG specifies a 153-byte long command line. *SHORT specifies a 73-byte long command line. *NONE specifies a 4-byte option line in place of a command line.

Note: Note that for both the DSPF and MSGF parameters, the value *MENU was specified. *MENU specifies that the name of the display or message file is the same as the name of the menu. Display and message file names *do not* have to have the same name as the menu.

Running the Menu

The new display file menu can now be run using the GO command:

```
GO PERSMENU
```



```
PERSMENU                  Personnel Menu

Select one of the following:

     1. Departments menu
     2. Education menu
     3. Benefits menu
     4. Job openings
     5. Job applicants









Selection or Command
===>____________________________________________________________________________
_______________________________________________________________________________
```

Defining Your Own Program Menus

Program menus provide an alternative to display file and system menus by using the GO command to call a program. The user has complete control of the display and function of the program menu through the program.

Passing Parameters for Program Menus

Three parameters are passed to the program when the menu is run:

- The 10-character name of the menu object
- The 10-character name of the library containing the menu object
- A 2-character binary return code

All three parameters must be declared as variables in the program. The return code determines how the program menu is exited, simulating the function keys found on IBM-supplied menus. The four possible values are:

Binary Return Code	Hex Equivalent	Description
0	0000	Call the program (display the menu) again
-1	FFFF	Exit function requested
-2	FFFE	Previous function requested
-4	FFFC	Home function requested (display the initial menu)

Note: The hexadecimal values shown above are used for those high-level languages (such as CL) that do not support binary numbers. CL programs should use a 2-byte character variable.

Building a Program Menu

This example shows how a program (*PGM) menu is created. The following steps will be shown and described in detail:

1. Describing the menu
2. Creating the display file
3. Entering source and creating a CL program to control the menu function

4. Creating the menu
5. Running the menu

Describing the Menu

The program menu is designed using data description specifications (DDS) to describe the menu's appearance. The following DDS is for the sample display called PGMMENU.

```
|...+....1....+....2....+....3....+....4....+....5....+....6....+....7...
     A                                      DSPSIZ(*DS3)
     A                                      PRINT
     A                                      CA03(01)
     A                                      CA12(02)
     A                                      HOME(03)
     A                                      INDARA
     A          R MENUFMT                   BLINK OVERLAY
     A            DSPMNUN       10A  O  1  2
     A                                  1 72 TIME
     A                                  1 31'EXAMPLE PROGRAM MENU'
     A                                      DSPATR(HI)
     A                                  2  2'SELECT ONE OF THE FOLLOWING:'
     A                                  3  5'1.' DSPATR(HI)
     A                                  3 10'DISPLAY LIBRARY LIST  (DSPLIBL)'
     A                                  4  5'2.' DSPATR(HI)
     A                                  4 10'WORK WITH ACTIVE JOBS (WRKACTJOB)'
     A                                  5  5'3.' DSPATR(HI)
     A                                  5 10'WORK WITH YOUR JOB    (WRKJOB)'
     A                                  6  5'90.' DSPATR(HI)
     A                                  6 10'RETURN'
     A                                 23  2'OPTION:'
     A            OPTION         3   I 23 12DSPATR(PC)
```

Figure 101. DDS Source for Program Menu Example

Creating the Display File

Create the display file using the Create Display File (CRTDSPF) command:

```
CRTDSPF FILE(QGPL/PGMDDS) SRCFILE(USERLIB/SOURCE1)
```

The DDS source file called SOURCE1 is used to create the display file for the PGMMENU menu.

Entering the Source and Creating a CL Program

The following CL program is used to control the menu function. (See the CL programming section of the **CL** topic in the iSeries Information Center for more information about entering CL source.) Note that any high-level language supported by the system could be used to control a menu.

```
 PGM        PARM(&MENUNAME &MENULIB &ACTION) /* Begin the program and +
              indicate that 3 parameters will be passed to it when +
              called.  The parameters include 1) The menu name, 2) +
              The menu library name 3) The action desired by this +
              program on return. */
 DCL        VAR(&MENUNAME) TYPE(*CHAR) LEN(10) /* Menu name */
 DCL        VAR(&MENULIB) TYPE(*CHAR) LEN(10) /* Menu library name */
 DCL        VAR(&ACTION) TYPE(*CHAR) LEN(2) /* Action variable */
 DCLF       FILE(QGPL/PGMDDS) RCDFMT(MENUFMT) /* Display file */
 CHGVAR     VAR(&DSPMNUN) VALUE(&MENUNAME) /* Set the menu name on the +
              menu                  */
 SNDRCVF    DEV(*FILE) RCDFMT(MENUFMT) /* Display the menu at the +
              display station */
 CHGVAR     VAR(&ACTION)  VALUE(X'0000') /* Indicate the menu should +
              be displayed again */
 /*****************************************************************/
 /* Handle function keys                                          */
 /*****************************************************************/
 IF         COND(&IN01 *EQ '1') THEN(CHGVAR VAR(&ACTION) +
             VALUE(X'FFFF')) /* If F3 was pressed, set action +
             to EXIT */
IF          COND(&IN02 *EQ '1') THEN(CHGVAR VAR(&ACTION) +
             VALUE(X'FFFE')) /* If F12 was pressed, set action +
             to PREVIOUS */
IF          COND(&IN03 *EQ '1') THEN(CHGVAR VAR(&ACTION) +
             VALUE(X'FFFC')) /* If HOME was pressed, set action +
             to HOME */
/*******************************************************************/
/* Handle menu options                                             */
/*******************************************************************/
IF          COND(&OPTION *EQ '1') THEN(DSPLIBL) /* If the menu user +
              has selected option 1, then display the library list */
IF          COND(&OPTION *EQ '2') THEN(WRKACTJOB) /* If the menu user +
              has selected option 2, then display the active jobs */
IF          COND(&OPTION *EQ '3') THEN(WRKJOB) /* If the menu user +
              has selected option 3, then display the current job. */
IF          COND(&OPTION *EQ '90') THEN(CHGVAR VAR(&ACTION) +
              VALUE(X'FFFE')) /* If the menu user has selected option +
              90, then set the action to previous. */
ENDPGM      /* End the program */
```

Create the CL program using the Create CL Program (CRTCLPGM) command:

```
CRTCLPGM PGM(QGPL/PGMCL) SRCFILE(USERLIB/SOURCE1)
```

Creating the Menu

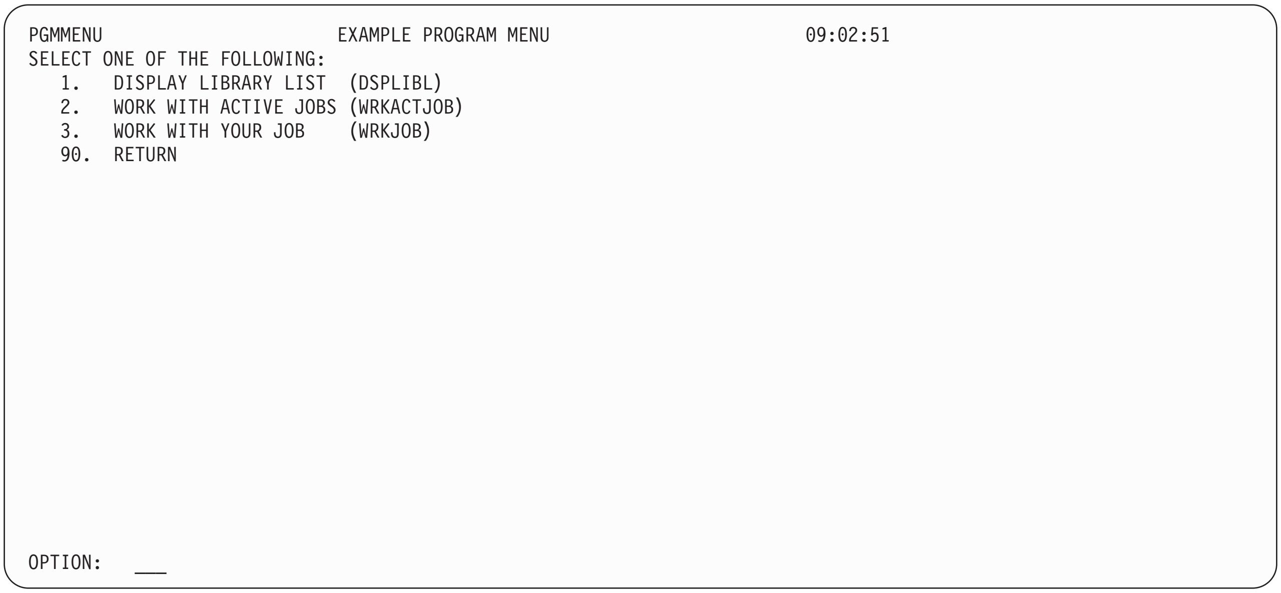
Create the program menu using the Create Menu (CRTMNU) command:

```
CRTMNU MENU(QGPL/PGMMENU) TYPE(*PGM) PGM(QGPL/PGMCL)
```

Running the Menu

The new display file menu can now be run using the GO command:

```
GO PGMMENU
```



```
PGMMENU                    EXAMPLE PROGRAM MENU                    09:02:51
SELECT ONE OF THE FOLLOWING:
   1.   DISPLAY LIBRARY LIST  (DSPLIBL)
   2.   WORK WITH ACTIVE JOBS (WRKACTJOB)
   3.   WORK WITH YOUR JOB    (WRKJOB)
   90.  RETURN









OPTION:   ___
```

Exiting from a Program Menu without Returning to the Previous Menu

When you code a TYPE(*PGM) menu to go from one program menu to another and use the RTNPNT(*NO) parameter, you need to tell the menu driver when to exit the original menu.

The program menu needs to communicate with the menu driver that F3 was pressed. If the program menu is coded to go to program MENU2, with a RTNPNT(*NO), it has to be able to convey information between programs and to convey information to the menu driver to tell it what should be done. You can use data queues to communicate between the programs. For more information about data queues, see the CL programming section in the **CL** topic in the iSeries Information Center. The following example shows you how to communicate between program menus if you want to use the RTNPNT(*NO) parameter to go from one menu to another:

Program 1

```
      PGM
      DCL  VAR(&FLDLEN) TYPE(*DEC) LEN(5 0) VALUE (10)
      DCL  VAR(&FIELD) TYPE(*CHAR) LEN(10)
      DCL  VAR(&WAIT) TYPE(*DEC) LEN(5 0) VALUE(0)  /* Don't wait */
      .
      .
      /*    Handle the function keys on the menu                  */
      IF    COND(&IN03 *EQ '1')     +    /* F3=Exit pressed       */
              THEN(DO)
                   /*   Delete the Data Queue                     */
                   DLTDTAQ  DTAQ(QGPL/DTAQ1)
                   /*   Create the Data Queue                     */
                   CRTDTAQ  DTAQ(QGPL/DTAQ1) MAXLEN(10)           +
                            TEXT('Test Data Queue')
                   GOTO SNDDQ           /* Send info to data queue*/
                   ENDDO
      .
      /*    Handle the options on the menu                        */
      IF    COND(&OPTION *EQ 1)                                    +
             THEN(GO MENU(*LIBL/MENU2) RTNPNT(*NO)
      .
      .
      /*    Returned from menu 2 after F3 or F12 pressed          */
      CALL  PGM(QRCVDTAQ)  PARM(QGPL DTAQ1 &FLDLEN &FIELD &WAIT)
      IF    COND((&FLDLEN *NE 0) *AND (&FIELD *EQ 'EXIT      '))
              THEN(GOTO SNDDQ)
      ELSE  GOTO END
      .
      .
SNDDQ:DO
        /*  Change the variable so the menu driver will know to   */
        /*  exit and send information to the data queue to        */
        /*  communicate to next program what happened             */
        CHGVAR VAR(&ACTION) VALUE(X'FFFF')
        CHGVAR VAR(&FLDLEN) VALUE(4)
        CHGVAR VAR(&FIELD) VALUE('EXIT      ')
        CALL PGM(QSNDDTAQ) PARM(QGPL DTAQ1 *FLDLEN &FIELD)
        GOTO END
      ENDDO
END:  ENDPGM
```

Program 2

```
      PGM
      DCL  VAR(&FLDLEN) TYPE(*DEC) LEN(5 0) VALUE (10)
      DCL  VAR(&FIELD) TYPE(*CHAR) LEN(10)
      DCL  VAR(&WAIT) TYPE(*DEC) LEN(5 0) VALUE(0)  /* Don't wait */
      .
      .
      /*    Handle the function keys on the menu                  */
      IF    COND(&IN03 *EQ '1')     +    /* F3=Exit pressed       */
              THEN(DO)
                   /*   Delete the Data Queue                     */
                   DLTDTAQ  DTAQ(QGPL/DTAQ1)
                   /*   Create the Data Queue                     */
                   CRTDTAQ  DTAQ(QGPL/DTAQ1) MAXLEN(10)           +
                            TEXT('Test Data Queue')
                   GOTO SNDDQ           /* Send info to data queue*/
                   ENDDO
      .
      .
      /*    Handle the options on the menu                       */
      IF    COND(&OPTION *EQ 1)                                   +
             THEN(GO MENU(*LIBL/MENU2) RTNPNT(*NO)
      .
      .
      /*    Returned from menu 2 after F3 or F12 pressed         */
      CALL  PGM(QRCVDTAQ)  PARM(QGPL DTAQ1 &FLDLEN &FIELD &WAIT)
      IF    COND((&FLDLEN *NE 0) *AND (&FIELD *EQ 'EXIT      '))
              THEN(GOTO SNDDQ)
      ELSE  GOTO END
      .
      .
SNDDQ:DO
        /*  Change the variable so the menu driver will know to   */
        /*  exit and send information to the data queue to       */
        /*  communicate to next program what happened            */
        CHGVAR VAR(&ACTION) VALUE(X'FFFF')
        CHGVAR VAR(&FLDLEN) VALUE(4)
        CHGVAR VAR(&FIELD) VALUE('EXIT      ')
        CALL PGM(QSNDDTAQ) PARM(QGPL DTAQ1 *FLDLEN &FIELD)
        GOTO END
      ENDDO
END:  ENDPGM
```

Avoiding Menu Name Conflict

Because high-level system menus are named by full words, there is some potential for conflict with user-defined menu names. For example, if you use the CRTMNU command to create a menu called MAIN, there may be a conflict with the system-supplied menu, MAIN. If you use the GO command to call that menu (GO MAIN), the system menu MAIN will probably be shown, because it resides in the QSYS library. A conflict can also occur if a future release contains a new system menu that has the same name as a menu you created during the previous release.

Naming Your Menus

To avoid naming conflicts with system command menus, you should avoid menu names that start with CMD. The system command menu names use the format CMDxxxxxxx, where xxxxxxx is any subject or verb used in CL command names.

Placing Your Menu in a Higher Library in the Library List

All branches to system menus from other system menus use the library list search order. Use of the library list search order allows multilingual support for the system menus. It also allows you to override a system menu by placing your own version of it in a library higher on the search order than library QSYS.

Specifying the Library That Contains the Menu

You can avoid menu name conflict by specifying the library that contains the menu you want to run. For example, you can use GO MENU(*USRLIBL/menu-name) to call user-defined menus, and use GO MENU(*LIBL/menu-name) to call system menus.

Using the Generic Menu Specification

When working interactively, you can use a generic menu specification on the GO command to avoid menu name conflict. In an example given earlier (see “Avoiding Menu Name Conflict”), a potential conflict existed between system and user-defined menus called MAIN. By specifying GO MAIN*, the system will display a list of all menu names that start with MAIN in your library list and allow you to select which menu you want.

Changing the Command Default after Duplicating a Command

You can use the Create Duplicate Object (CRTDUPOBJ) command to create a duplicate of the GO command for calling user-defined menus. The following example creates a duplicate of the GO command called GOUSR:

```
CRTDUPOBJ OBJ(GO) FROMLIB(QSYS) OBJTYPE(*CMD) NEWOBJ(GOUSR) TOLIB(QGPL)
```

Next, use the Change Command Default (CHGCMDDFT) command to change the command default for the MENU parameter to use library *USRLIBL rather than *LIBL:

```
CHGCMDDFT CMD(QGPL/GOUSR) NEWDFT('MENU(*USRLIBL/*N)')
```

With the MENU parameter default changed to *USRLIBL, the GOUSR command will bypass system menus and find only those menus in the user portion of your library list.

Displaying Menu Attributes

The Display Menu Attributes (DSPMNUA) command can be used to show the attributes of a menu object. These include:

- The menu type (*UIM, display file, or program menu)
- The display and message files used by display file (*DSPF) menus
- The programs used by program (*PGM) menus
- The current (CURLIB) and product (PRDLIB) libraries associated with the menu
- The text describing the menu object

Changing Menu Attributes

The Change Menu (CHGMNU) command can be used to change the attributes of a menu object without having to re-create the object. The attributes which can be changed are:

- The current (CURLIB) and product (PRDLIB) libraries for all menu types
- The descriptive text for all menu types
- The display and message files used by display file (*DSPF) menus
- The programs used by program (*PGM) menus

Deleting Menus

The Delete Menu (DLTMNU) command can be used to delete menu objects from a library. The DLTMNU command can be used to delete only the menu object. Referenced display and message files (used by display file menus) and programs (used by program menus) can also be deleted.

DLTMNU is a *generic* command. For example, all menus in a library called OELIB could be deleted using the command, `DLTMNU OELIB/*ALL`.

As another example, if only those menus in OELIB whose names started with ACC were to be deleted, the command, `DLTMNU OELIB/ACC*` could be used.

Chapter 10. Using User-Defined Data Streams

Instead of having the system control and process the 5250 display data stream, you can control and process it. To do so, you must use the USRDFN keyword in the DDS for a display file. The data is sent to the display station using a normal output operation that uses the name of the record format containing the USRDFN keyword. When input data is received after the output operation, your program must determine through the input record area what was received from the display station.

Note: You should be careful when using this support because error conditions can cause an apparent i5/OS malfunction.

When you use the USRDFN keyword at the record level, the format does not contain any fields. Therefore, the buffer length of the file defaults to the length of the longest normal record or 100 (whichever is greater). If the user-defined data stream is longer than this default buffer length, you should perform the following steps to obtain a larger buffer length:

- Define an externally described file, and create the program using this file and a record format in that file. The format should not have response indicators defined for it. This includes file-level indicators that are spread to all record formats. Any fields in the format should be defined with a field use of *both* (input and output).
- Create a second file and specify `LVLCHK(*NO)`. The second file should have two record formats:
  - A format with the same name as the format in the first file and which contains the USRDFN keyword.
  - A format with one field in it. The length of this field must be as long as the longest user-defined data stream that is to be sent to the screen.
- When you run the program, override the first file with the second display file.

Information about 5250 Display Data Streams

For information about coding 5250 display data streams, see the following manuals:

- *IBM 5250 Information Display System Functions Reference*
- *IBM 5494 Remote Control Unit Functions Reference*

Understanding Display Station Differences

If you use user-defined data streams, you should be aware of the following differences between local display stations (those attached to the display station controller) and remote display stations (those attached to remote display station controllers).

- For local display stations, a start-of-header order length greater than 11 causes the negative response hex 1005012B to be sent. No negative response is sent for remote display stations.
- For local display stations, do not accept more than four field control words for each input-field definition. More than four causes the negative response hex 10050130 to be sent. No negative response is sent for remote display stations.
- For local display stations, self-check fields can be as long as 33 bytes for signed numeric fields and 32 bytes for all other fields. If the length is exceeded, the status response hex 00000287 is sent to the user when he or she tries to exit the self-check field. For remote display stations, all fields can be 33 bytes.
- Two forms of the Request Maintenance Statistics command are supported for remote display stations: (1) requests that the error log area be reset, and (2) requests that the error log area not be reset. For local display stations, the error log area is always reset.

Understanding User-Defined Data Stream Limitations

The output record area for a user-defined record format must contain the following in the order specified:

1. Display-station-specific information (required by the system), which causes the system to send the display-station-specific data stream to the display station.
2. Display-station-specific data stream, which is sent to the display station.

The USRDFN keyword is specified at the record format level and excludes for that record format most other functions such as the use of indicators and all field-related functions. However, the display file can contain other record formats not containing the USRDFN keyword, and the record formats can be used in any order by a program. When you use such a display file, you should be aware of the following:

- When the system write routine recognizes a user-defined request, it disregards all previous requests to the display. At the completion of the user-defined request, the system assumes that a single record format is on the screen and that this is the record format containing the USRDFN keyword. All erasing, all resetting, and unlocking of the keyboard is your responsibility. The next I/O request can be another user-defined request or a normal field-level request.
- A normal field-level request after a user-defined request is handled as follows:
  - If the OVERLAY keyword is not specified, the screen is erased before the request is run.
  - If the OVERLAY keyword is specified, only the portion of the display needed (entire lines) is erased.
  - The 5250 format buffer is reset, which means that all input fields are changed to output-only fields.

  The system assumes that only this record format exists on the screen. All previous requests are disregarded.
- Help specifications are allowed in user-defined record formats.
- When the USRDFN keyword is specified in a record format, no fields can be defined for that record format. The only valid keywords are:

File Level	Record Format Level
KEEP	ALTNAME
OPENPRT	HELP
PASSRCD	HLPCLR
PRINT	HLPRTN
	INVITE
	PRINT
	TEXT

- The user-defined data stream can alter the CFnn and CAnn keys and the location of the message line in the display. However, display station support assumes they are the same as when they were last set.
- All display files that contain user-defined data streams should be opened as both input and output files. This is because read and write commands in the data stream are not dependent on write and read requests.

The output data stream should start with an escape character hex 04 and be followed by a clear unit hex 40 or write to display command hex 11.

The output buffer must include information needed to send and receive the appropriate line controls. The buffer format for a 5250 display station is:

Byte	Contents
0-1	Send data length (in hexadecimal), defines the output data stream length
2-3	Receive data length (in hexadecimal), defines the maximum input data length

Byte	Contents
4	Requested function Hex 51 Send (WP mode) Hex 53 Send/Receive (WP mode) Hex 61 Send (3270 data stream) Hex 63 Send/Receive (3270 data stream) Hex 71 Send Hex 73 Send/Receive
5-n	Output data stream

- The 3270 Model 4 display station supports a display size of 43 lines by 80 columns. If you have a 3270 Model 4 display station, you can use function codes 61 and 63 to write to all 43 lines.
- For 3270 Model 4 display stations that are using function code 61 or 63, the system displays status messages on line 43. Any data that is on line 43 when a status message is displayed is replaced by the status message.
- If you are using function code 61 or 63 and press the key that is defined as the System Request key, the key is returned to the application program instead of displaying the System Request menu. If you need access to the system request function when using function code 61 or 63, you can define the Attention key to display the System Request menu. Before starting the application that uses function codes 61 and 63, specify:

  ```
  SETATNPGM SET(*ON) PGM(QSYS/QWSSYSRQ)
  ```

  When the Attention key is pressed, the System Request menu is displayed.

This support prohibits the use of a read operation except after a write-read(nowait) operation. A write-read operation that sends a read command and specifies the receive data length performs the operation normally performed with a read request. The write-read function can be performed by doing one of the following:

Operation	ILE RPG	ILE COBOL	Control Language
Write-read(wait)	EXFMT		SNDRCVF WAIT(*YES)
Write-read(nowait)	WRITE with INVITE, READ	WRITE with INVITE, READ	SNDRCVF WAIT(*NO)

Both the write-read(wait) and the write-read(nowait) operations are a combination of a write operation and a read operation to the same record format. When using the write-read(wait) operation, control does not return to the program until input is received from the display device. The write-read(nowait) operation is used to send a request for input to a display device and return to the program without waiting for the input to arrive. This allows a program to request input from one or more devices, and continue processing without waiting for any of the devices to respond. For each I/O request, you must make sure that the function byte and the send depth actually reflect the data stream sent. The receive length must be long enough to accommodate all data returned by the display station. If any of these is wrong, unexpected or unacceptable functions may happen. For example, if the function byte indicates a send to the display station and the data stream specifies a read modified command, the system sends the data stream to the display station and no read is performed.

On input operations, the input buffer of the program contains the data received from the display station. For example, when a read modified completes, the input buffer contains an aid identification (AID) byte and the cursor address followed by each changed field. Each changed field is preceded by the buffer address order and field location on the screen.

All AID bytes are accepted by display station support for user-defined data streams. If the Print key is pressed, the system attempts to perform the print function.

All write and write-read operations must specify the record format name. A read operation that specifies the name of a record format that contains the USRDFN keyword and is not one of the appropriate read operations for the display station causes an exception to be issued.

Each request to a 5250 display station can contain more than one command. Each command must be requested using the appropriate system operation identified as follows:

Operation	Output Buffer Send Length (Bytes 0-1)	Output Buffer Receive Length (Bytes 2-3)	Output Buffer Request Function (Byte 4)	Output Buffer Command (Bytes 5-n)
Write	nn	0	71 (Send)	Clear unit Clear format table Write to display Write error line Restore Roll Copy
Write-Read	nn	nn	73 (Send/Receive)	Clear unit Clear format table Write to display Write error line Restore Roll Read input fields Read MDT fields Read immediate Save

Chapter 11. Passing Data between Programs

You can use the system I/O operations to pass data between programs both in the same routing step of a job or across routing steps.

Passing Data in the Same Routing Step in a Job

To pass data between programs in the same routing step, you can either share the file between the programs, or you can use the KEEP and ASSUME DDS keywords to share a file between programs. The programs must open the same file and the file must be designated as a shared file. (See "Sharing Display Files in the Same Job" on page 85.) Because the programs share the file, the read operations are performed as described under "Reading Input from the Display" on page 70 and "Understanding How the System Reads Input from the Display" on page 76.

You can also use the KEEP and ASSUME keywords to pass data between programs. The data is written to the display by the first program, and used from the display by the second.

You can use the KEEP keyword to keep data on the display for review after the program has ended, or you can use it to pass data between programs.

Normally, when a file is closed, the current display is cleared. However, you can control this by using the KEEP keyword. If a record that is on the display when the file is closed has the KEEP keyword specified, the system saves the name of the first such record to support passing data. Using this keyword alone, you cannot support processing of passed data, you must also use the ASSUME keyword in the display file opened by the next program.

The ASSUME keyword causes a read operation to a specific record format name to be valid when no preceding write operation to that record format name (or any other record format) has occurred since the display file was opened. The following shows a typical example of what happens when the ASSUME keyword is specified:

1. Program PGM1 issues a write operation to the record format PGM1ANY in the display file DSPFIL1 and calls program PGM2. PGM1ANY specifies the LOCK keyword.
2. PGM2 opens DSPFIL2 and issues a read operation to record format PGM2ANY. The ASSUME keyword is specified. Input data is read from the display and processed by record format PGM2ANY.

Records from record formats having the ASSUME keyword specified cannot overlay one another on the display. In addition, all records must have at least one field that is displayed.

When the system reads assumed records from the display, only fields whose modified data tags (MDTs) are on are returned. (This assumes that either the DSPATR(MDT) keyword was specified for the field on the last output operation or the user has typed in the field.) When the ASSUME keyword is in effect, the data returned to the program is as follows:

- For input-only and output/input fields with their MDTs on, the changed data is returned.
- For other input-capable fields, blanks are returned for character fields and zeros are returned for numeric fields.

In addition, only those fields received whose line and position on the display match the field of an assumed record are processed. The data is processed using the field descriptions in the assumed record, independent of how the fields were written to the display.

For the 5250 display station, the first write operation or write-read operation after the file is opened sends a CA and CFnn key specification to the display station. Because there is no write operation after an open when using the ASSUME keyword, the CFnnkey specification remains as it was left by the last application. The CA and CFnnkey specification in this file will not be used until after the first write operation to the file.

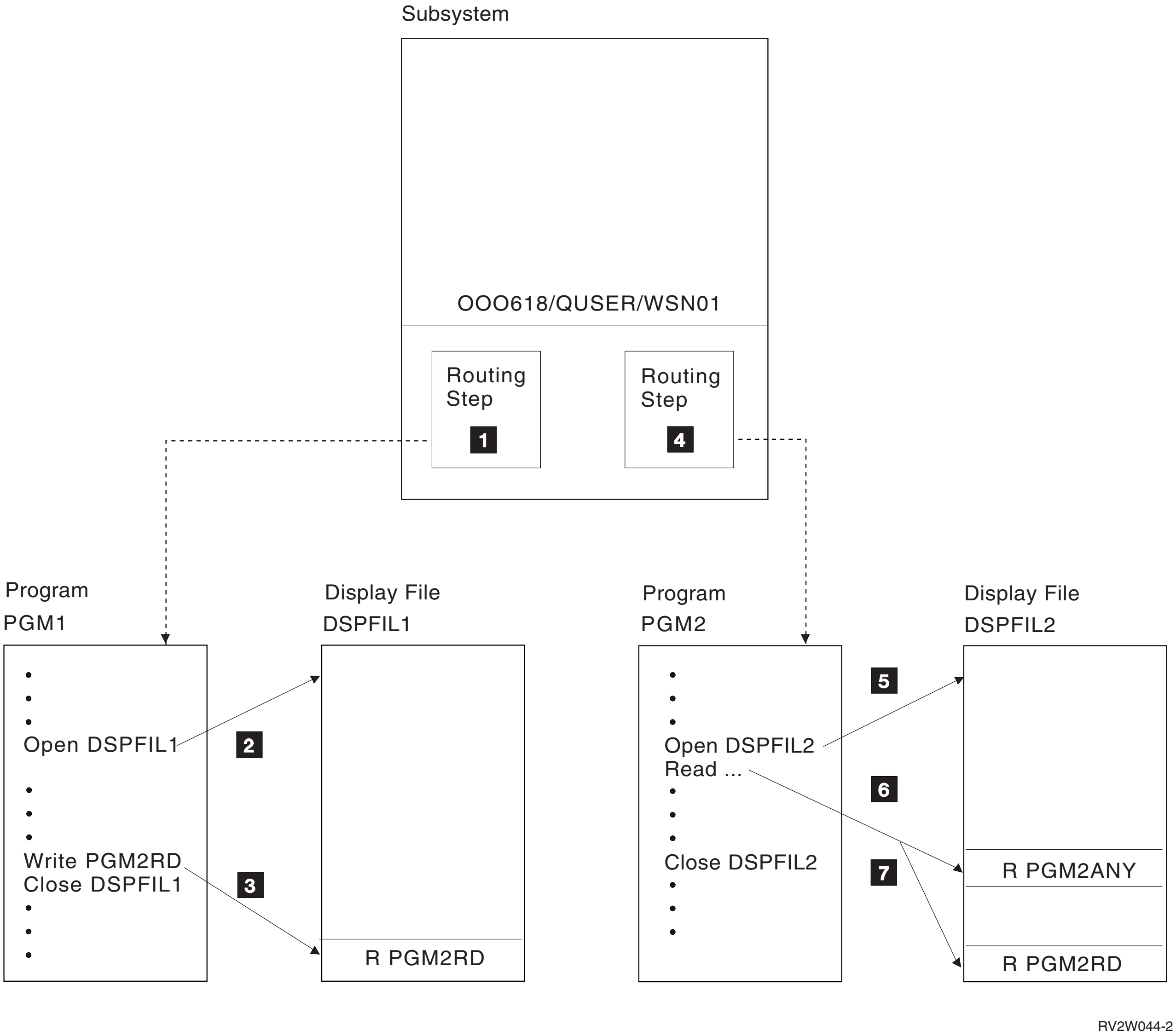
After the file is opened and the first write operation is issued, the display will be erased if the OVERLAY keyword is not specified. When the OVERLAY keyword is specified, all input-capable fields on the display become output-only fields. After the first write operation, assumed records cannot be read by a program.

Fields not received from the assumed records on the display are returned to the user program as follows:

Field	Initialized To
Field with DFT keyword used	Value specified in DFT keyword
Character field (no DFT keyword)	Blanks
Numeric field (no DFT keyword)	Zero
Hidden field	Zero

Passing Data between Routing Steps in a Job

The following example shows the steps used in passing display data between programs in different routing steps. Note the use of the KEEP, ASSUME, and PASSRCD keywords. Programs PGM1 and PGM2 are user programs started by the subsystem SBS using different routing entries. PGM1 and PGM2 run in different routing steps but are both contained in the job 000618/QUSER/WSN01.



1 The user signs on and SBS starts a routing step based on the routing data. The first program in the routing step is PGM1.

2 PGM1 opens the display file DSPFIL1.

3 PGM1 interacts with the user and issues the following before ending:

- A write operation to DSPFIL1 with the record format name PGM2RD. The KEEP keyword is specified in the record format PGM2RD.
- A close operation to DSPFIL1. The information displayed on WSN01 is not cleared because the KEEP keyword is specified in record format PGM2RD.

PGM1 then ends by issuing a RRTJOB command specifying routing data that will cause SBS to call PGM2.

4 SBS starts a new routing step based on the data supplied by PGM1 in the RRTJOB command.

5 PGM2 opens the display file DSPFIL2.

6 PGM2 performs a read operation to DSPFIL2 with or without the record format name PGM2ANY. If the record format name is not specified, the system tries to use the record format PGM2RD to process the data because the KEEP keyword was specified (in step 5). To do so, record format PGM2RD must exist in DSPFIL2, but it does not have to be identical to record format PGM2RD in DSPFIL1; only the fields that are required by PGM2 must be identical. (If a field is returned from the display station for which no field description exists in PGM2ANY, the field is ignored. If the field description requires validation and the data received fails the validity check, PGM2 reissues a read operation to allow the current user to correct the data.) If a record format is not specified or if record format PGM2RD does not exist in DPSFIL2 or does not have the ASSUME keyword specified, the data passed to the display station via the KEEP keyword cannot be processed. (The ASSUME keyword must be specified for the record format used to process passed data.)

7 PGM2 processes the data and issues a close operation to DSPFIL2, which clears the display.

When PGM2 ends, its routing step ends and the display returns to the control of SBS which shows the sign-on display.

Chapter 12. Waiting for Input from a Display File, an ICF File, and a Data Queue

You can use data queues for a program that waits for data on a display file, an ICF file, and a data queue at the same time (in any combination). When you specify the DTAQ parameter for the following commands:

- Create Display File (CRTDSPF) command
- Change Display File (CHGDSPF) command
- Override with Display File (OVRDSPF) command
- Create ICF File (CRTICFF) command
- Change ICF File (CHGICFF) command
- Override ICF File (OVRICFF) command

You can indicate a data queue that will have entries placed on it when any of the following occurs:

- An enabled command key or Enter key is pressed from an invited display station.
- Data becomes available from an invited ICF session.

By using the IBM-supplied QSNDDTAQ program, jobs running on the system can also place entries on the same data queue as the one specified in the DTAQ parameter.

An application program uses the IBM-supplied QRCVDTAQ program to receive each entry placed on the data queue and then processes the entry based on whether it was placed there by a display file, by an ICF file, or by the QSNDDTAQ program. For a display file, the application then issues a read or read-from-invited devices operation to receive the data. For more information on the QRCVDTAQ function and syntax, and examples of waiting on one or more files and a data queue, refer to the CL programming section in the **CL** topic in the iSeries Information Center.

The display file and ICF file entry that is put on the data queue is 80 characters long and contains the field attributes described in Table 28. Therefore, the data queue that is specified using the commands listed above must have a length of at least 80 characters.

Entries placed on the data queue by jobs using QSNDDTAQ are defined by the user.

Table 28. Display File and ICF File Entry Field Attributes

Position	Data Type	Meaning
1 through 10	Character	The type of file that placed the entry on the data queue. This field will have one of two values: *ICFF (ICF file) *DSPF (display file) If the job receiving the data from the data queue has only one display file or one ICF file open, then this is the only field that needs to be used to determine what type of entry has been received from the data queue.
11 through 12	Binary	Unique identifier for the file. The value of the identifier is the same as the value in the open feedback area for the file. This field should be used by the program receiving the entry from the data queue only if more than one file with the same name is placing entries on the data queue.

Table 28. Display File and ICF File Entry Field Attributes (continued)

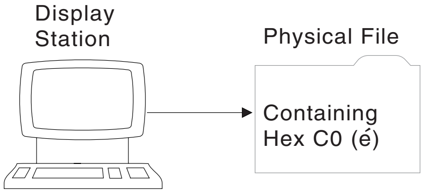
Position	Data Type	Meaning
13 through 22	Character	The name of the display or ICF file. This is the name of the file actually opened, after all overrides have been processed, and is the same as the file name found in the open feedback area for the file. This field should be used by the program receiving the entry from the data queue only if more than one display file or ICF file is placing entries on the data queue.
23 through 32	Character	The library where the file is located. This is the name of the library, after all overrides have been processed, and is the same as the library name found in the open feedback area for the file. This field should be used by the program receiving the entry from the data queue only if more than one display file or ICF file is placing entries on the data queue.
33 through 42	Character	The program device name, after all overrides have been processed. This name is the same as that found in the program device definition list of the open feedback area. For file type *DSPF, this is the name of the display station where the command or Enter key was pressed. For file type *ICFF, this is the name of the program device where data is available. This field should be used by the program receiving the entry from the data queue only if the file that placed the entry on the data queue has more than one device or session invited prior to receiving the data queue entry.
43 through 80	Character	Reserved

Chapter 13. Using Alternative Character Sets and Code Pages

In multinational environments, data in one national character set may need to be displayed or entered on display stations that support another national character set. This is particularly true of characters with accents and other characters with diacritical marks (such as c with a cedilla, n with a tilde, and u with an umlaut). In this section, these characters are called **extended alphabetics**.

For example, assume that a physical file on the system contains data in the Basic French code page, and includes an é (e with an acute accent). In the Basic French character set, this character is hex C0. The data could have been entered on a display station that can handle the character or could have been sent to this system from another system over a communications line.

Note: On some display stations, extended alphabetics can be entered on the keyboard (by pressing Cmd, then the key to its right with two diacritical marks on it, then the 2 hexadecimal digits that represent that character).



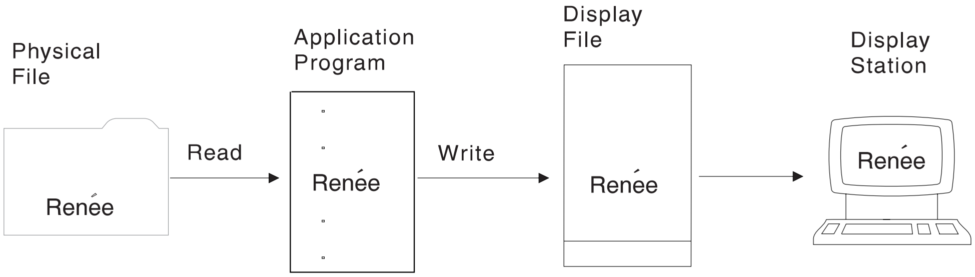
RV2W036-3

When the hex C0 is sent to a display station that does not recognize hex C0 as é, the hex C0 is not displayed as é. For example, a display station that recognizes the U.S. code page displays hex C0 as { (a left brace). The é is at code point 51 in the U.S. code page. To properly display é, the hex C0 must be converted to hex 51 for display on a U.S. basic display station.

With the IBM-supplied conversion tables, the system can convert most extended alphabetics, but not all. If an extended alphabetic character does not have a clearly preferable equivalent, the system converts the character to a (-) hyphen.

System Has Characters Not Normally Displayed on the Device

The case of an output/input field shows how character conversion occurs at a display station.



RV2W037-3

Assume that a record in a physical file contains a field with the value Renée. An application program reads the record from the physical file, and writes a record containing the data to the display file. One way to achieve character conversion is in the DDS for the display file, the output/input field in which

Renée appears has the Character Identifier (CHRID) keyword specified. [1]This keyword asks the system to perform character conversion if needed. Character conversion is needed if the CHRID parameter value for the display file differs from the CHRID value of the display station. In this case, character translation is needed because the display file has CHRID(288 297), which is the basic French code page and character set, and the device description has CHRID(101 37), which is the basic U.S. code page and character set.

When displaying the data, the system converts the hex C0 to hex 51, and Renée appears on the display. If no conversion occurred, Ren{e would appear on the screen.

On input, one of the following occurs:

- If the modified data tag (MDT) is turned on for the field (this happens when the user types into the field or if the DSPATR(MDT) keyword is in effect for the field) the contents of the field are converted from the device CHRID to the CHRID parameter value specified on the Display File (DSPF) command and is returned to the program.
- If the modified data tag (MDT) is not turned on for the field, the saved contents of the field (with original, untranslated, data) is returned to the program.

Another way to achieve character conversion on a file basis is to specify the *JOBCCSID value on the CHRID parameter for the Create Display File (CRTDSPF), the Change Display File (CHGDSPF), or the Override Display File (OVRDPSF) commands. More information about *JOBCCSID is in "Specifying Character Translation for Fields"

Device Passes Characters Not Displayed on the System

If the user can enter data not normally displayed on the system (for instance, a user on a French-language display station in Montreal dials up a remote line to a system in Toronto with display stations that cannot display extended alphabetics), no change occurs when the data is sent to the system. The program can read data from the French-language display station and write it to a physical file. However, when another user on the Toronto system displays data from the physical file, the system attempts to translate the data for the display station that cannot display extended alphabetics.

Specifying Character Translation for Fields

If the CHRID keyword is specified on a field, character conversion occurs only if the CHRID parameter value in the display file is different from the CHRID value in the device description.

1. The CHRID parameter on the display file is set by the CHRID parameter on the CRTDSPF, CHGDSPF, or OVRDSPF command. The CHRID parameter can have one of the following values:

   *DEVD (the default): Use the CHRID parameter specified on the device description on which the application is currently running. No character conversion occurs. Any CHRID keywords specified in the DDS are ignored.

   *SYSVAL: Use the character set and code page specified in the system value QCHRID on which the application is currently running. Specify the CHRID keyword in the DDS for the display file fields that will need conversion. Constants are not converted.

   *JOBCCSID: Use the character set and code page specified by the job. This allows named fields and unnamed (constant) fields to be automatically translated when the job CCSID or the display file CCSID does not match the device description CHRID. If the job CCSID does not match the device description CHRID, named fields are translated from the job CCSID to the device description CHRID on output and conversely on input. If the display file CCSID does not match the device description CHRID, data in the display file is translated from the display file CCSID to

[1] If *JOBCCSID is specified by the CHRID parameter on the CRTDSPF, CHGDSPF, or OVRDSPF commands, the DDS keyword CHRID is ignored. For more information, see the description for *JOBCCSID in "Specifying Character Translation for Fields."

the device description CHRID on output. When *JOBCCSID is specified, the DDS CHRID keyword is ignored. Use the No Coded Character Set Identifier (NOCCSID) keyword on named and unnamed fields that you do not want translated.

Character set and code page: Use the character set and code page specified. See Table 29 on page 264 for the list of valid values.

2. The CHRID value on the device description is set by the CHRID parameter on the Create Device Description Display (CRTDEVDSP) or Change Device Description Display (CHGDEVDSP) command. This parameter can have one of the following values:

   *SYSVAL: Use the character set and code page specified in the system value QCHRID.

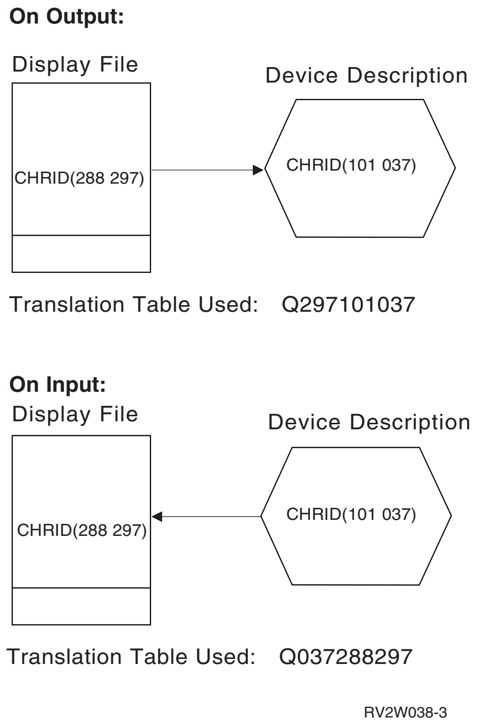
   *KBDTYPE: Use the character set and code page specified in the keyboard language prompt (KBDTYPE). This value corresponds to the keyboard language identifier.

   Character set and code page: Use the character set and code page specified. See Table 29 on page 264 for the list of valid values.

When dealing with UCS-2 Level 1 data from a physical file, it will be necessary to convert this data to EBCDIC before displaying the data on the screen. To accomplish this conversion, place the DDS CCSID keyword at either the file-, record-, or field-level to enable conversion of UCS-2 data from the UCS-2 CCSID value specified with the keyword to the device CHRID on output. On input, the data is converted from the device CHRID to the UCS-2 value. For more information, see “UCS-2 Level 1 Considerations for DDS” in the **DDS** topic in the iSeries Information Center book.

Another conversion that occurs every time data is displayed is converting the hex “3F” character to the hex “1F” character on output and conversely on input. The Character Data Representation Architecture (CDRA) specifies the hex “3F” character as a replacement character. This character is also a field attribute definition for the 5250 data stream specification. Translation to convert hex “3F” character to hex “1F” character for output is done for all fields whether *JOBCCSID translation is active or inactive. Use the NOCCSID keyword to prevent translation at the field level.

When character conversions are necessary, the system uses a conversion table in library QUSRSYS to convert the data. The name of the conversion table used is derived from parts of the source and target CCSID for which the table is needed as shown in the following example:



IBM supplies a number of tables in library QUSRSYS to handle conversion among the most frequently used combinations of character sets and code pages. If you need to tailor the way characters are converted, you can create your own table in library QUSRSYS, using the Create Table (CRTTBL) command to do the conversion. To determine what conversion tables are available to use, look in the QUSRSYS library for object type *TBL.

Determining the Character Identifier (CHRID) Value for Your Display

The CHRID value specified should be based on the attributes of the display station. The table below shows CHRID values that are appropriate for each display station keyboard type. For some display stations, you do not need to specify the KBDTYPE parameter, but the KBDTYPE value for the equivalent keyboard can be used to determine the CHRID value for the display station:

Table 29. CHRID Values

Language/Country or Region	Keyboard Type (KBDTYPE)	Limited CHRID	Full CHRID
International and US ASCII	INB	103 038	697 500
Multinational	AGI BLI CAI DMI FAI FNI FQI ICI INI ITI JEI NEI NWI PRI SFI SGI SPI SSI SWI UKI USI		697 500
Arabic	CLB		235 420
Austria/Germany	AGB	265 273	697 273
Belgium Multinational	BLI		697 500
Canada/French	CAB	277 260	341 260
Cyrillic	CYB		960 880
Denmark/Norway	DMB NWB	281 277	697 277
Finland/Sweden	FNB SWB	285 278	697 278
France	FAB FQB	288 297	697 297
Greece	GKB		218 423

Table 29. CHRID Values (continued)

Language/Country or Region	Keyboard Type (KBDTYPE)	Limited CHRID	Full CHRID
Hebrew	NCB		941 424
Iceland	ICB		697 871
Italy	ITB	293 280	697 280
Japan/English	JEB	297 281	697 281
Japan/Kanji	JKB (For Personal System/55, 5295 and 3477-J display stations)		332 290
Japan/Katakana	KAB (For 5251, 5291, 5292, and 3180 Katakana display stations)		332 290
Korean	KOB		933 833
Latin 2	ROB		959 870
Netherlands	NEB		697 037
Portugal	PRB	301 037	697 037
Simplified Chinese	RCB		936 836
Spain	SPB	305 284	697 284
Spanish Speaking	SSB	309 284	697 284
Switzerland/French Multinational	SFI		697 500
Switzerland/German Multinational	SGI		697 500
Thai	THB		938 838
Traditional Chinese	TAB		101 037
Turkey	TKB		965 905
United Kingdom/English	UKB	313 285	697 285
United States/English	USB	101 037	697 037
Countries of the former Yugoslavia	YGI		959 870

Chapter 14. Improving System Performance with Displays

System performance is improved by doing the following:

- Minimizing the number of bytes sent and received when designing a display. For example, to reduce the number of fields on an application display:
  - Split a display with many fields into more than one display.
  - Consider removing fields which were added to help with development, testing, or problem reporting.
- Taking advantage of the system functions that reduce the number of bytes sent to and received from the display station. This chapter describes some of these system functions.

Deferring the Write Operation for a Display File

The DFRWRT parameter on the Create Display File (CRTDSPF) or Change Display File (CHGDSPF) command allows you to delay the writing of data to the display until needed. Deferring the write operation is useful when the final display shown at the display station is formed by several record formats being written to the display.

Using DFRWRT(*YES) on a display file improves systems performance in some cases.

More information on the Defer Write parameter can be found in "Deferring the Write Operation Until a Read Request is Made" on page 64.

Designating the Primary Screen Size for a Display File

Normally, the display files are set up for a 24 by 80 screen (default size). The DSPSIZ keyword specifies which display sizes are valid for a file and indicates which sizes are the primary and secondary screen sizes. (The primary screen size is the first or only DSPSIZ value.) On the DSPSIZ keyword, the screen size can be specified as *DS3, *DS4, 24 80, or 27 132. For example, DSPSIZ (24 80) specifies a screen size of 24 by 80.

The screen size designated as the primary screen size should be the one with which the display file will most often be used. A performance benefit will be realized by coding the DSPSIZ keyword in this manner. Additional processing is performed when the actual screen size is the secondary screen size.

Writing Only One Page of Subfile Records at a Time

A technique to improve performance when you are using a multiple page subfile is to write only one page of subfile records at a time but use the i5/OS support to roll through the subfile. To do this, you need to define the ROLLUP keyword in DDS with a response indicator and also use the SFLRCDNBR keyword. In your program, you would write the records needed to fill one subfile page and then display that page. When the user wants to see more records, he or she presses the Roll Up key. The program then writes another page of records to the subfile, places the relative record number of a record from the second page into the SFLRCDNBR field, and displays the record.

The second page of subfile records is now displayed, and if the user presses the Roll Down key, the roll down is handled by the system. If the user presses the Roll Up key while the first page is displayed, the system will also handle the roll up. The program is notified only when the user attempts to roll up beyond the records currently in the subfile. The program would then handle any additional roll up requests in the same manner as for the second page. When you use this technique, the subfile appears to be more than one page because of the use of the roll keys. Yet, you can maintain good response time because the program only fills one subfile page before writing it to the display.

Sharing an Open Data Path (ODP) for the Same Job

i5/OS data management support offers a close level of sharing within a job that allows more than one program to share the same path to the data or display sation. By specifying *YES for the SHARE parameter on the Create Display File (CRTDSPF) command, the Change Display File (CHGDSPF) command, and the override file commands, more than one program can share the file status, positions, and storage area. Performance is improved by reducing the amount of main storage the job needs and the time it takes to open and close the file.

Note: To use SHARE(*YES) to improve performance, you need to define many record formats within the display file. However, the program may not use all the record formats in the display file, even though the program has work storage allocated for all the record formats defined in the display file. This causes the PAG storage associated with the job to be larger, which can adversely affect performance.

More information about sharing open data paths is found in "Sharing Display Files in the Same Job" on page 85.

Sending Records with Input Fields to the Display in Order

Records containing input fields should be sent to the display station in the order in which they appear on the display. This technique provides better performance than if record formats with input fields are sent randomly or in some other order.

Overlapping and Not Deleting Repeatedly Sent Records

You can use the CLRL(*NO) keyword to prevent an overlapped record from being deleted when the overlapping record is written to the display. If you use this keyword, any records being displayed that are to be overlapped are not deleted from the screen; the new record overlays them entirely or partially. There is a performance advantage to using CLRL(*NO) if you have a display that contains constants and data that is repeatedly sent to the screen. By sending the constants as a separate format and using CLRL(*NO) for the format containing only the data, you can reduce the time required to send the record format to the display.

The use of the PUTOVR keyword causes only those fields for which the OVRDTA or OVRATR keyword have been specified to be sent to the display when a subsequent write or write-read operation is issued to the same record format.

You can also use the OVERLAY keyword to clear only that portion of the display affected by the record format being written.

You can use ERASEINP to improve response time by causing the display to clear fields instead of requiring blanks to be sent to the display. If the fields erased at the display do not have their modified data tags set on for the next read operation, data is returned for those fields from the input save area. This is data saved by the system from the previous return of the field from the display station.

You can use the INZINP keyword at the record level with ERASEINP(*ALL) and PUTOVR to initialize the input save area without sending data for the cleared fields to the display.

Restoring the Display

When *YES is specified for the Restore Display (RSTDSP) parameter, an image of the current display is saved when the display is suspended. When the display file is activated again, the saved image is used to restore the display to its appearance before being suspended.

The RSTDSP(*YES) parameter *must* be specified for the following keywords. If the parameter is not specified, data on the display can be lost if the file is suspended.

- CLRL
- OVERLAY
- PUTOVR
- PUTRETAIN
- ERRMSG
- ERRMSGID

If none of the previous keywords are used, you can improve performance by specifying *NO for the RSTDSP parameter.

Defining Command Attention Keys Rather Than Command Function Keys

Command attention (CAnn) keys return only the indication of the key pressed and not data. Command function (CFnn) keys cause all input data to be returned. If you only need to detect a function key and do not need to return data from the display, define keys as command attention keys.

Using the Invite Operation

The invite operation, specified by the INVITE keyword, allows the program to be responsive to various display station and system events. The invite operation also allows the program to process between the last WRITE and READ operations.

Using Windows

DDS-described windows may improve performance because they only affect the portion of the display where they are placed.

Part 3. Programming Application Displays Using Panel Groups

Chapter 15. Improving Productivity with User Interface Manager 273
Increasing User Productivity 273
Increasing Application Programmer Productivity 273
What to Consider before Using UIM Instead of Data Description Specifications (DDS) 273

Chapter 16. Introduction to the User Interface Manager. 275
Overview of UIM 275
What the UIM Supports 276
What Is a Panel Group 276
What Is a Menu 276
Creating Objects 277
Elements Within a Panel Group 277
Using the UIM Language Tags 277
Using Dialog Commands 278
Using Control Language (CL) Commands 279
Using an Application Programming Interface (API) 279
Defining a Menu Object Using UIM 280
Creating a Menu Panel 280
Required Tags for a Menu Panel 282
Source for Example Menu 282
Defining a Panel Group Object Using UIM . . . 287
Creating a List Panel 287
Required Tags for a List Panel. 289
Source for Example List Panel. 290
Application Programming for a List Panel. . . 298
Creating a Confirmation List Panel 299
Required Tags for a Confirmation List Panel . . 300
Source for Example Confirmation Panel . . . 300
Automatic Confirmation Processing 303
Application Programming for Confirmation Processing 303
Creating a Data Presentation Panel 303
Required Tags for a Data Presentation Panel . . 306
Source for Example Data Presentation Panel . . 306
Application Programming for a Data Presentation Panel. 317
Data Entry Panel 320
Creating a Panel with a Menu Bar 320
Required Tags for a Panel with a Menu Bar . . 322
Source for Example Panel with a Menu Bar . . 323
Application Programming for a Menu Bar Panel 332

Chapter 17. Details of Using User Interface Manager. 335
Opening a UIM Application 335
Defining Dialog Variables 335
Restrictions on Using Dialog Variables 336
Dialog Variable Error Messages 337
Providing Field Values for a Display Panel Using Dialog Variables 337
Using Variable Pool Services 338
| Dialog Variables and Special Values 338
Character Set and Code Page Considerations 338
Displaying 339
Printing 339
Managing a List 340
Defining a List 341
Initializing a List 341
Displaying a List 341
Updating a List. 342
Incomplete List Processing 342
Removing and Inserting an Entry from a List 342
Controlling List Entries on a List Display . . . 343
Improving Interactive Response Time for a List Display 343
Using Action Lists and Selection Lists 344
Using Action Lists. 344
Using Selection Lists 345
Using Selection Characters 345
Managing Panel Functions 345
Enabling Conversion to a GUI. 346
Scrolling Support 346
Defining Scrollable Areas 346
Defining Function Key Scrolling 347
Scrolling and Error Conditions 347
Scrolling a List Area 347
Scrolling a Menu Area 347
Scrolling an Information Area 348
Scrolling Data Item Groups. 348
Scrolling a Text Area 348
Defining Contextual Help 348
Command Line Restrictions 350
Command Line Interpretation 350
Entering Commands That Are Too Long . . . 351
Defining Function Keys 351
Formatting Function Keys 351
Handling Function Keys and VARUPD Value 351
Panel Formatting Concepts 352
When Panel Formatting Is Performed 353
Application Control of Panel Formatting . . . 353
Limits of the Panel Formatter 354
Folding Up Multiple Panels When EXIT Is Requested 354
Folding Up a List Panel 355
Adding a Pop-Up Window over Another Panel 356
Using Menu Bars 357
Differences Between Pull-Down Menus and Pop-Up Windows 358
Using Pop-Up Windows. 359
Defining Application Windows 359
Adding and Removing Windows. 360
Using the Command Line in a Window . . . 361
UIM as a Request Processor Program When Displaying a Panel 361
Printing Concepts 362

Printing a Print Head Panel 364
Printing a Print Panel 364
Using Blank Lines for Separating 364
Fonts and Highlighting 364
Printing the Trailer 364
Defining Prolog Areas 364
Defining Header Areas 365
Using the Page-Eject Function During Printing 365
Sharing and Overriding Printer Files 365
Printing Double-Byte Character Set (DBCS)
Considerations 365
Commonly Asked UIM Questions 365

Chapter 15. Improving Productivity with User Interface Manager

Using the user interface manager (UIM) can improve user and application programmer productivity.

Increasing User Productivity

The UIM controls the panel's appearance and assures consistency with panels developed by IBM. This gives an application a consistent user interface which increases user productivity.

Increasing Application Programmer Productivity

Application programmer productivity is increased by using the UIM to create panels. The application programmer can describe the contents of a panel without specifying all the details. The format of the panel's elements is handled automatically by the UIM. For more information on the panel elements that UIM handles, see "What the UIM Supports" on page 276.

Using the UIM language tags allows the application programmer to ″link″ an option number or function key to a specific command or program. Then, when a user selects an option or presses a function key, the UIM automatically handles running the command or program. The UIM also handles scrolling through a multiple-page panel. The Major Command Groups menu is an example of a multiple-page panel.

What to Consider before Using UIM Instead of Data Description Specifications (DDS)

The UIM and DDS each use a different source language to create interactive displays. Display functions in interactive applications can be built using DDS, a combination of DDS and UIM, or only UIM. With DDS, the application programmer can customize display screens, and with UIM, the display screens are automatically formatted resulting in a consistent appearance. In addition, the UIM performs more dialog management functions than DDS, so using UIM results in less application programming.

The UIM supports the following types of display screens (or panels). Each one can be scrollable or nonscrollable:

- List
- Menu
- Information
- Data entry

For more information on creating these display screens, see Chapter 16, "Introduction to the User Interface Manager," on page 275.

Consider the following before using UIM:

- If you are creating a new application, the UIM simplifies making display screens standard.
- If you are rewriting an old application, evaluate the effort involved to rewrite the display screens versus the productivity benefits already described. In some cases, conversion to UIM may not be warranted unless a major redesign or rewrite of an application is required for other reasons.
- Because UIM was designed for 525x nonprogrammable workstations, using UIM does not appreciably simplify converting an application oriented to a programmable workstation except by ensuring standard display screens.

- If an application program makes extensive use of database files, UIM does not take advantage of file descriptions the way DDS does.
- Using UIM to create data panels should be for low frequency and low volume output/input. Nonscrollable data entry applications with high frequency and volume of interactions should consider using display files created from DDS for the best performance. This is because the UIM does not accept high frequency and high volume keyboard input as quickly as DDS does. That is, the UIM may not be able to accept keyboard input as quickly as the user can type it. However, a keyboard with type-ahead function may compensate for this.

The following lists some of the advantages and differences between using UIM or DDS:

- UIM Advantages
  - Uses same standards as system so no need to redefine standards. User applications work the same way as the system panels. UIM formats panels based on what you want displayed.
  - List processing offers ability to process commands, to prompt or call programs easily from a list panel, to specify programs for UIM to call after option is selected, and allows more efficient list entry access or update processing.
  - Operates well with languages that efficiently process structures.
  - Provides for more modular programming techniques (one program can process all incomplete list exit calls, open all applications, and so on).
  - Offers ability to condition menu options
  - Formats and handles scrolling of large areas without user program intervention (data, list, info, menu, and function key areas, for example).
- DDS Advantages
  - Provides more flexibility in screen design
  - SDA helps in initial formatting
  - Ability to use UIM help or help in folders
  - Ability to take advantage of GUI windows
  - Subfile processing
  - Use of EDTCDE and EDTWRD and user-defined editing
  - Faster for smaller applications since set-up time is less.

For additional factors to consider when using UIM, see “Choosing between Panel Groups and Records for Help” on page 370.

Chapter 16. Introduction to the User Interface Manager

This chapter gives an introduction to the user interface manager (UIM) that IBM uses to develop the i5/OS panels. It explains what the UIM provides to make creating and managing panels easy and also gives the panels the look of an i5/OS panel.

For detailed information on the UIM, see Chapter 17, “Details of Using User Interface Manager,” on page 335.

For information on using the examples mentioned in this chapter, see source member T0011INF in source file QATTINFO in library QUSRTOOL.

Overview of UIM

The i5/OS UIM is a part of the system that allows you to define panels and dialogs for your application and provides the following support:

- A tag-based language for describing data and panels.
- A compiler to create panel group objects and menu objects using the tag-based language.
- A set of application programming interfaces (APIs) to use as panel group objects to display and print panels.
- The UIM also provides the following functions:
  - Dialog commands for screen management
  - Contextual online help
  - Index search
  - Pop-up windows
  - Menu bars
  - Command line for entering CL commands
  - Tailoring of the contents of a panel for different users or environments
  - Fast paths through menu networks
  - Double-byte character set (DBCS) languages
  - Bidirectional (BIDI) language support
  - Graphical user interface (GUI) support
  - UCS-2 support

UIM supports common panel types, such as menus, information displays, list displays, and entry displays. When all display types and interfaces are consistent, users adapt more quickly to new applications.

UIM applications can coexist with and share the requester display device with other open display files that are not under UIM control. However, a UIM panel and a DDS-defined record format cannot appear on the display at the same time. When a UIM panel replaces a DDS panel or vice versa, the system suspends operations of one file or panel group and restores the display as needed.

What the UIM Supports

When you design a panel, the UIM provides the correct placement and format of many panel elements, such as:

- Panel name
- Panel title
- Separator lines
- Column headings
- Entry fields
- Command line
- Message line
- Function keys
- Pop-up windows
- Menu bars
- Pull-down menus

In addition, the UIM provides the following support:

- Correct placement of the cursor

  The user interface style has several rules for positioning the cursor which UIM supports.
- Cursor-sensitive help information

  Depending on where the cursor is located when the Help key is pressed, the user interface style has rules for what type of help information is displayed.
- Scrolling
- Fast paths through menu networks
- National language considerations to make translating easier
- Windows for help information and application panels
- 132-column panels
- Left-to-right and right-to-left (bidirectional) national language formatting
- Hypertext links
- Control over left-to-right and right-to-left orientation of text
- Ability to create online help information
- Ability to enable a panel for conversion to a graphical user interface (GUI)

What Is a Panel Group

A **panel** is a visual presentation of data on the screen. A **panel group** is an object that contains a collection of display formats, print formats, or help information. The system-recognized identifier for the object type is *PNLGRP.

For detailed information on the panel group (PNLGRP) language tag, see “PNLGRP (Panel Group)” on page 609.

What Is a Menu

A **menu** is an object that contains the definition of a panel which contains one or more options. The user can select an option from the panel in order to start using a program or command, or to go to another menu. The system-recognized identifier for the object type is *MENU.

i5/OS provides the GO command to display a menu. Therefore, no application program is necessary to display and handle the user interaction for a menu.

You can define the following types of menus using the Create Menu (CRTMNU) command.

***DSPF**
An existing display file (*DSPF) and message file (*MSGF) are used to display the menu.

***PGM** i5/OS calls an application-defined program to display the menu. The program is responsible for displaying the menu to the user and processing options requested by the user.

***UIM** The menu object is created using a member in a source file that contains a description of the menu. The source describes the menu using the UIM tags.

This chapter discusses how to create a *UIM type of menu. For information on how to create a *DSPF or *PGM type of menu, see Chapter 9, "Creating and Accessing Menus Using Display Files," on page 235.

Creating Objects

The UIM creates and changes the following objects:

- Panel group objects
- Menu objects
- Search index objects

The panel group and menu objects contain panel definitions and online help information. A menu object (*MENU), which contains a panel group definition, is created using the TYPE(*UIM) parameter on the Create Menu (CRTMNU) command. The panel group and menu objects are created using a tag-based language that specifies definitions for UIM elements.

The UIM creates search index objects that contain search terms extracted from online help information. A search index object makes it more efficient for a user to locate specific online information using the index search function.

Elements Within a Panel Group

The tag-based language used to define the panels, menus, and online help also supports definitions of symbolic elements that include the following:

- Variable classes
- Data elements that can be accessed through the UIM application programming interface (API), such as dialog variables and lists
- Variable records that define buffers passed by application programs
- Conditional expressions that must be true if certain processing is to take place
- Key lists containing the definition of function keys
- Menu bars containing the definition of one or more pull-down menus
- Panel definitions containing one or more areas to present data, information, lists, and menus
- Online help text modules

Using the UIM Language Tags

The language tags are an easy-to-use function for defining panels. Using the language tags helps application programmers create consistent appearing panels. For more information on the language tags, see Appendix A, "UIM Panel Group Definition Language," on page 465.

Using Dialog Commands

Dialog commands are special functions, recognized only by the UIM, that allow the user to navigate through the panels.

Following is a list of the dialog commands and the functions they perform:

ACTIONS Alternates the cursor position between the panel and the menu bar; removes, if shown, a pull-down menu from the panel.

CALL Calls an application program to perform a function.

CANCEL Backs up one panel (returns to the previous panel).

CHGVIEW Changes the displayed view of a list by switching defined sets of columns to be shown.

CMD Submits an i5/OS CL command (or System/36 environment OCL command) to the system for processing.

CMDLINE Displays a pop-up window with a command line.

DSPHELP Displays a module of help text.

ENTER Initiates processing of an action; submits panel input for processing.

EXIT Returns the user from a group of displays or menus.

EXTHELP Displays the *extended* help text for the panel.

HELP Displays help information for the panel, based on the position of the cursor.

HELPHELP Displays information about how to use the help facilities.

HELPIDX Initiates the index search function for the application to allow the user to make a search request.

HOME Displays the initial (home) menu of the job.

KEYSHELP Displays the help for the function keys shown on the displayed panel.

MENU Displays a subsequent menu as a result of selecting a menu item or pressing a function key.

MOREKEYS Displays an additional set of active function keys and their descriptions; used when all keys cannot be shown at once.

MOVETOP Moves a cursor-selected line to the top of the scrollable information area.

MSG Displays a message on the message line.

PAGEDOWN Pages (scrolls) *forward* by one panel.

PAGEUP Pages (scrolls) *backward* by one panel.

PRINT Prints all the information shown on the current display.

PROMPT Prompts (seeks input) for commands, action list options, or entry fields.

PULLDOWN Displays the pull-down menu for the first choice shown on the menu bar.

RETRIEVE Retrieves and displays the previously entered command.

RETURN Returns control (and a return value) to an application for processing.

Note: To provide the Refresh/Redisplay function for an application, use the RETURN dialog command.

For detailed information on the dialog commands, see Appendix B, “UIM Dialog Commands.”

Using Control Language (CL) Commands

The following CL commands are used to work with panel groups, menus, and index search:

- ADDSCHIDXE (Add Search Index Entry)
- CHGMNU (Change Menu)
- CHGSCHIDX (Change Search Index)
- CRTMNU (Create Menu)
- CRTPNLGRP (Create Panel Group)
- CRTSCHIDX (Create Search Index)
- DLTMNU (Delete Menu)
- DLTPNLGRP (Delete Panel Group)
- DLTSCHIDX (Delete Search Index)
- DSPMNUA (Display Menu Attributes)
- GO (Go to Menu)
- RMVSCHIDXE (Remove Search Index Entry)
- STRSCHIDX (Start Search Index)
- WRKSCHIDXE (Work Search Index Entry)

For detailed information on the CL commands, see the **CL** topic in the iSeries Information Center.

Using an Application Programming Interface (API)

The UIM provides several API services. The service calls to the UIM consist of the following:

- Application services
- Variable pool services
- List services
- Display services
- Print services

The application services consist of the following:

- Open Display Application (QUIOPNDA)
- Open Print Application (QUIOPNPA)
- Close Application (QUICLOA)

The variable pool services consist of the following:

- Get Dialog Variable (QUIGETV)
- Put Dialog Variable (QUIPUTV)

The list services consist of the following:

- Add List Entry (QUIADDLE)
- Add List Multiple Entries (QUIADDLM)
- Get List Entry (QUIGETLE)
- Get List Multiple Entries (QUIGETLM)
- Update List Entry (QUIUPDLE)
- Remove List Entry (QUIRMVLE)
- Delete List (QUIDLTL)
- Set List Attributes (QUISETLA)
- Retrieve List Attributes (QUIRTVLA)

The display services consist of the following:

- Display Panel (QUIDSPP)
- Display Help (QUHDSPH)
- Add Pop-up Window (QUIADDPW)
- Remove Pop-up Window (QUIRMVPW)
- Set Screen Image (QUISETSC)
- Display Long Text (QUILNGTX)

The print services consist of the following:

- Print Panel (QUIPRTP)
- Print Help (QUHPRTH)
- Add Print Application (QUIADDPA)
- Remove Print Application (QUIRMVPA)

For details on the APIs, see the **APIs** topic in the iSeries Information Center:

Defining a Menu Object Using UIM

Using the UIM, you can create a *UIM type of menu object. The menu object contains the definition of a menu panel with one or more options. To create a *UIM type of menu, do the following.

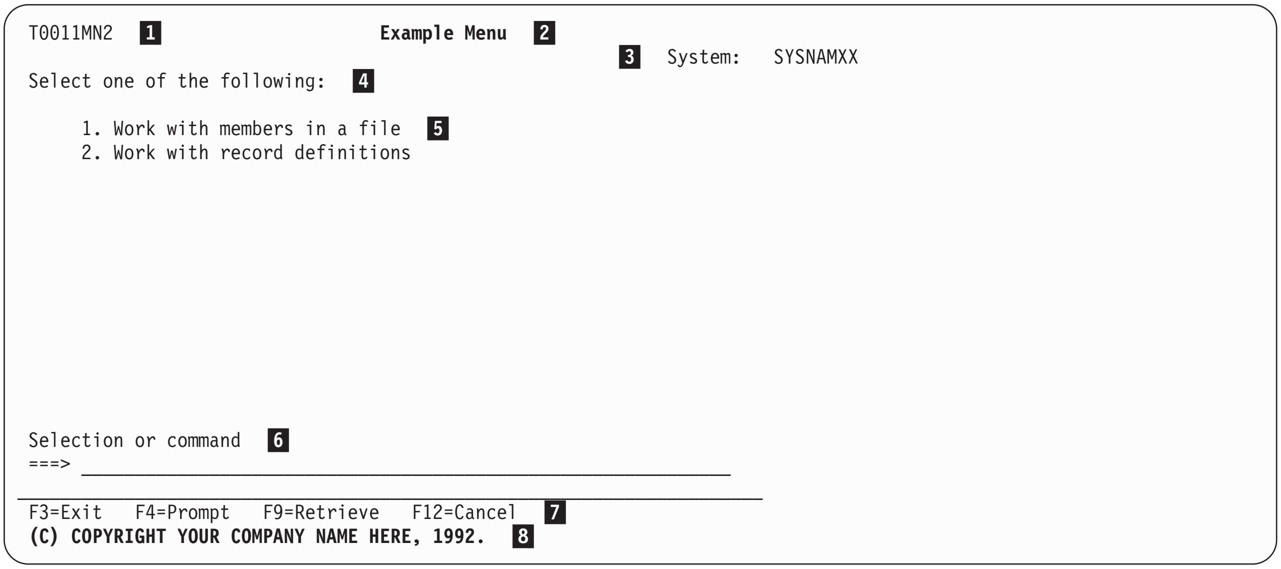
1. Create a member in a source physical file.
2. Enter the UIM tag source to describe the menu using a source editor, such as the Source Entry Utility (SEU). An example menu exists in member T0011MN2 in file QATTUIM in library QUSRTOOL. You can copy this example to use as a template to create your own menu.
3. Use the Create Menu (CRTMNU) command to create the menu object using the tag source as input. Here is an example of using the CRTMNU command.

   ```
   CRTMNU MENU(MYLIB/MYMENU) TYPE(*UIM) SRCFILE(MYLIB/QMNUSRC)
   ```

   The source member defaults to the name of the menu being created.

Creating a Menu Panel

The panel shown in Figure 102 on page 281 is an example menu. If you need more information on creating help for a menu area, see “Help in a Menu Area” on page 389.



```
T0011MN2  1                      Example Menu  2
                                                      3  System:   SYSNAMXX
Select one of the following:  4

     1. Work with members in a file  5
     2. Work with record definitions















Selection or command  6
===> ____________________________________________________________
________________________________________________________________________
F3=Exit   F4=Prompt   F9=Retrieve   F12=Cancel  7
(C) COPYRIGHT YOUR COMPANY NAME HERE, 1992.  8
```

Figure 102. An Example Menu

The reference numbers in the example menu do not appear on the display. They are shown for illustration purposes and also appear in the UIM tag source shown in “Source for Example Menu” on page 282. These reference numbers show the portions of the source which define text and information that appear on the menu.

1 The panel identifier is defined by declaring the UIM-defined Z-variable, ZMENU, using the VAR tag and specifying the Z-variable to be used as the panel identifier on the `PANELID` attribute of the PANEL tag.

2 The panel title is defined as text following the period of the PANEL tag.

3 The system name is shown because the TOPSEP=SYSNAM attribute was specified on the PANEL tag.

4 The top instruction line is defined as text following the period of the top instruction (TOPINST) tag.

5 Each option on the menu is defined using the menu item (MENUI) tag. The option number is defined using the `OPTION` attribute and the text following the option number is defined as text following the MENUI tag.

6 The command line is defined using the command line (CMDLINE) tag. The command line prompt text is defined as text following the CMDLINE tag. The UIM automatically provides the arrow (`===>`) for every command line.

An option line can be defined by using the option line (OPTLINE) tag instead of the CMDLINE tag. In this case, the user enters option numbers, but is not allowed to enter system commands.

7 Each function key is defined using a key list item (KEYI) tag. The text, including the function key name, is defined as text following the period of the KEYI tag.

All function keys for a panel are defined by placing the KEYI tags between a key list (KEYL) and EKEYL tag. The name specified on the `NAME` attribute of the KEYL tag is also specified on the `KEYL` attribute of the PANEL tag.

Some function keys, such as the Enter key, do not have text specified for the KEYI tag. In this case, no text appears on the display. However, all function keys must be defined using the KEYI tag so the UIM knows what action is assigned for each key and what help module should be displayed when help is displayed for the function keys.

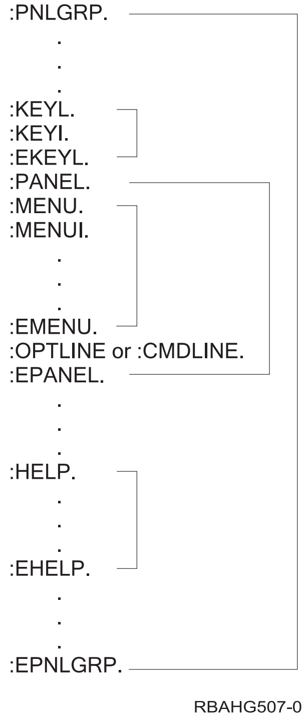
8 The copyright message is defined as text following the period of the copyright (COPYR) tag. The copyright message is displayed once when the user initially sees the menu. The message is not shown when the menu is redisplayed after processing an option or command.

This example menu is not defined as scrollable because it contains only two options. If you need to define a scrollable menu, change SCROLL=NO to SCROLL=YES on the MENU tag and add one menu item (MENUI) tag for each additional menu option. You also need to define a new help module for each menu option.

Required Tags for a Menu Panel

Figure 103 shows the required tags for creating a UIM menu panel. For an example of the required and optional tags, see “Source for Example Menu.”

Note: The tags in Figure 103 require attributes. These attributes are not shown. Without these attributes, the example in Figure 103 will not compile. For a description of the required attributes for these required tags, see Appendix A, “UIM Panel Group Definition Language.”



```
:PNLGRP.
      .
      .
      .
:KEYL.
:KEYI.
:EKEYL.
:PANEL.
:MENU.
:MENUI.
      .
      .
      .
:EMENU.
:OPTLINE or :CMDLINE.
:EPANEL.
      .
      .
      .
:HELP.
      .
      .
      .
:EHELP.
      .
      .
      .
:EPNLGRP.
```

Figure 103. Required UIM tags for a menu panel

Source for Example Menu

This is a listing of the tag source for the menu shown in Figure 102 on page 281. The entire source can be found in member T0011MN2 in source file QATTUIM in library QUSRTOOL.

```
.* -----------------------------------------------------------------
.*
.* Beginning of menu source.
.*
.* -----------------------------------------------------------------
:PNLGRP   DFTMSGF=t0011msgf2
          SUBMSGF=t0011msgf2.
.*
.* -----------------------------------------------------------------
.* Copyright statement appears when the menu is initially displayed.
.* ------------------------------------------------------------------
:COPYR.   8
(C) COPYRIGHT YOUR COMPANY NAME HERE, 1992.
.*
.* ------------------------------------------------------------------
.* UIM Z-variable to be used as the panel identifier
.* ------------------------------------------------------------------
:VAR      NAME=ZMENU.   1
.*
.* ------------------------------------------------------------------
.* Define keys for the menu
.* ------------------------------------------------------------------
:KEYL     NAME=menukeys   7
          HELP=keyl.
:KEYI     KEY=F1
          HELP=helpf1
          ACTION=HELP.
:KEYI     KEY=F3
          HELP=exit
          ACTION='EXIT SET'
          VARUPD=NO.
F3=Exit   7
:KEYI     KEY=F4
          HELP=prompt
          ACTION=PROMPT.
F4=Prompt
:KEYI     KEY=F9
          HELP=retrieve
          ACTION=RETRIEVE.
F9=Retrieve
:KEYI     KEY=F12
          HELP=cancel
          ACTION='CANCEL SET'
          VARUPD=NO.
F12=Cancel
:KEYI     KEY=F24
          HELP=morekeys
          ACTION=MOREKEYS.
F24=More keys
:KEYI     KEY=ENTER
          HELP=enter
          ACTION=ENTER.
:KEYI     KEY=HELP
          HELP=help
          ACTION=HELP.
:KEYI     KEY=HOME
          HELP=home
          ACTION=HOME.
:KEYI     KEY=PAGEDOWN
          HELP=pagedown
          ACTION=PAGEDOWN.
:KEYI     KEY=PAGEUP
          HELP=pageup
         ACTION=PAGEUP.
:KEYI     KEY=PRINT
          HELP=print
          ACTION=PRINT.
:EKEYL.
.*
.* ------------------------------------------------------------
.* Define Example Menu panel
.* -----------------------------------------------------------------
:PANEL    NAME=xmpmenu
          HELP='menu/help'
          KEYL=menukeys    7
          ENTER='MSG CPD9817 QCPFMSG'
          PANELID=ZMENU    1
          TOPSEP=SYSNAM.   3
Example Menu   2
.*
.* -------------------------------------
.* Define the menu area
.* -------------------------------------
:MENU     DEPTH='*'
          SCROLL=NO
          BOTSEP=SPACE.
:TOPINST.Select one of the following:   4
.*
.* -------------------------------------
.* Specify the action to be taken for each option
.* -------------------------------------
:MENUI    OPTION=1   5
          ACTION='CMD ?t0011cm2'
          HELP='menu/option1'.
Work with members in a file
.*
:MENUI    OPTION=2
          ACTION='CMD ?t0011cm3'
          HELP='menu/option2'.
Work with record definitions
.*
:EMENU.
.*
.* -------------------------------------
.* Use a command line and allow commands and option numbers
.* -------------------------------------
:CMDLINE  SIZE=LONG.   6
Selection or command
.*
:EPANEL.
.*
.*
.* -----------------------------------------------------------------
.* Define help modules for the menu panel
.* -----------------------------------------------------------------
:HELP     NAME=keyl. Function Keys - Help
:XH3.Function keys
:EHELP.
.*
:HELP     NAME=helpf1.
:PARML.
:PT.F1=Help
:PD.
Provides additional information about using the display or a
specific field on the display.
:EPARML.
:EHELP.
.*
:HELP     NAME=exit.
:PARML.
:PT.F3=Exit
:PD.
Ends the current task and returns to the display from which the
task was started.
:EPARML.
:EHELP.
.*
```

```
:HELP     NAME=prompt.
:PARML.
:PT.F4=Prompt
:PD.
Provides assistance in entering or selecting a command.
:EPARML.
:EHELP.
.*
:HELP     NAME=retrieve.
:PARML.
:PT.F9=Retrieve
:PD.
Displays the last command you entered on the command line and
any parameters you included.  Pressing this key once, shows the
last command you ran.  Pressing this key twice, shows the
command you ran before that and so on.
:EPARML.
:EHELP.
.*
:HELP     NAME=cancel.
:PARML.
:PT.F12=Cancel
:PD.
Returns to the previous menu or display.
:EPARML.
:EHELP.
.*
:HELP     NAME=morekeys.
:PARML.
:PT.F24=More keys
:PD.
Shows additional function keys.
:EPARML.
:EHELP.
.*
:HELP     NAME=enter.
:PARML.
:PT.Enter
:PD.
Submits information on the display for processing.
:EPARML.
:EHELP.
.*
:HELP     NAME=help.
:PARML.
:PT.Help
:PD.
Provides additional information about using the display.
:EPARML.
:EHELP.
.*
:HELP     NAME=home.
:PARML.
:PT.Home
:PD.
Goes to the menu that was shown after you signed on the system.
This menu is either the initial menu defined in your user
profile or the menu you requested from the Sign-On display.
:EPARML.
:EHELP.
.*
:HELP     NAME=pagedown.
:PARML.
:PT.Page Down (Roll Up)
:PD.
Moves forward to show additional information for this display.
:EPARML.
```

```
:EHELP.
.*
:HELP     NAME=pageup.
:PARML.
:PT.Page Up (Roll Down)
:PD.
Moves backward to show additional information for this display.
:EPARML.
:EHELP.
.*
:HELP     NAME=print.
:PARML.
:PT.Print
:PD.
Prints information currently shown on the display.
:EPARML.
:EHELP.
.*
:HELP     NAME='menu/help'.
Example Menu - Help
:P.
The Example Menu shows an example of a menu created using the UIM.
:XH3.
How to Use a Menu
:P.
To select a menu option, type the option number and press Enter.
:P.
To run a command, type the command and press Enter.  For assistance
in selecting a command, press F4 (Prompt) without typing anything.
For assistance in entering a command, type the command and press F4
(Prompt).  To see a previous command you entered, press F9
(Retrieve).
:P.
To go to another menu, use the Go to Menu (GO) command. Type GO
followed by the menu ID, then press the Enter key.  For example, to
go to the User Tasks (USER) menu, type GO USER and press the Enter
key.  The menu ID is shown in the upper left corner of the menu.
For assistance in entering the GO command, type GO and press F4
(Prompt).  If you do not know the entire menu name you can use a
generic name.  For example, GO US* will show a list of all menus
that start with US.
:EHELP.
.*
:HELP     NAME='menu/option1'.
Option 1 - Help
:XH3.Option 1. Work with members in a file
:P.
Select this option to work with the members in a file.  You will be
prompted for the name of the file.
:EHELP.
.*
:HELP     NAME='menu/option2'.
Option 2 - Help
:XH3.Option 2. Work with record definitions
:P.
Select this option to work with record definitions for a file.
You will be prompted for the name of the file.
:EHELP.
.*
.*
.*
.* ------------------------------------------------------------------
.* End of menu source
.* ------------------------------------------------------------------
:EPNLGRP.
```

Defining a Panel Group Object Using UIM

Using the UIM, you can create a panel group object. The panel group object can contain the definitions for dialog variables, lists, panels and help modules. To create a panel group object, do the following.

1. Create a member in a source physical file.
2. Enter the UIM tag source to describe the panel group using a source editor such as the Source Entry Utility (SEU). An example panel group exists in member T0011PN2 in file QATTUIM in library QUSRTOOL. You can copy this example to use as a template to create your own panel group.
3. Use the Create Panel Group (CRTPNLGRP) command to create the panel group object using the tag source as input. Here is an example of using the CRTPNLGRP command.

   ```
   CRTPNLRP PNLGRP(MYLIB/MYPNLGRP) SRCFILE(MYLIB/QPNLSRC)
   ```

   The source member defaults to the name of the panel group being created.

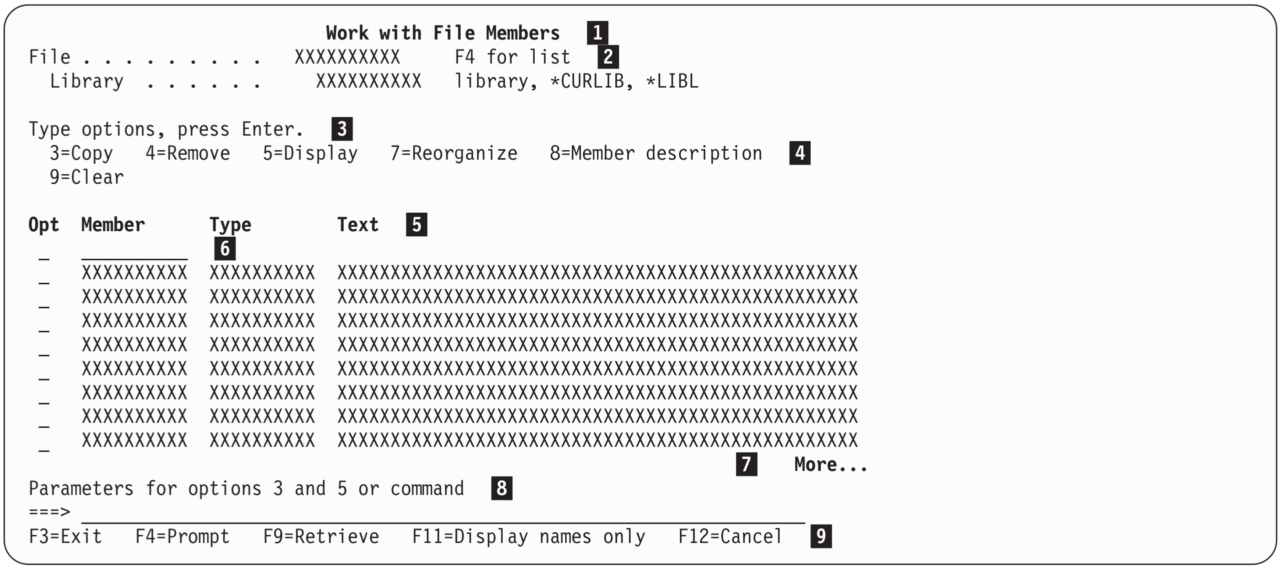
Creating a List Panel

Any of the ″Work with...″ panels are list panels. In this example, the Work with File Members panel is used. Defining the list panel is similar to defining a subfile in DDS, but the UIM determines the appearance of the panel.

The panel shown in Figure 104 shows an example of a mixed panel with a list area. The panel is mixed because it contains two types of areas: a data presentation area at the top and a list area at the bottom.

The data presentation area at the top consists of the fields which identify the file and library name. The list area begins with the instruction line, *Type options, press Enter* and ends with the scroll information (*More...*).

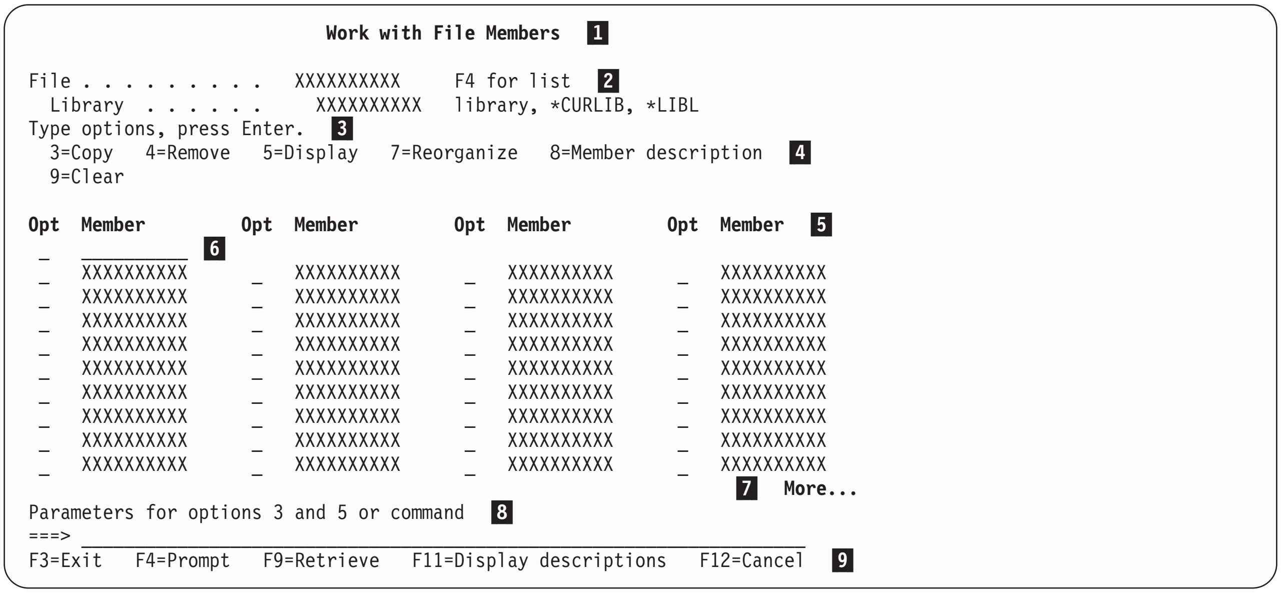
In this example, the panel contains a special type of list area called an action list. An action list is a list which contains an option column. The user types in allowed option numbers to perform actions against the object represented by the entry in the list. If you need more information on creating list panel help, see “Help in a List Area” on page 388.



```
                            Work with File Members  1
File . . . . . . . . .   XXXXXXXXXX     F4 for list  2
  Library  . . . . . .     XXXXXXXXXX   library, *CURLIB, *LIBL

Type options, press Enter.  3
  3=Copy   4=Remove   5=Display   7=Reorganize   8=Member description  4
  9=Clear

Opt  Member      Type        Text  5
 _   __________  6
 _   XXXXXXXXXX  XXXXXXXXXX  XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
 _   XXXXXXXXXX  XXXXXXXXXX  XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
 _   XXXXXXXXXX  XXXXXXXXXX  XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
 _   XXXXXXXXXX  XXXXXXXXXX  XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
 _   XXXXXXXXXX  XXXXXXXXXX  XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
 _   XXXXXXXXXX  XXXXXXXXXX  XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
 _   XXXXXXXXXX  XXXXXXXXXX  XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
 _   XXXXXXXXXX  XXXXXXXXXX  XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
                                                                   7   More...
Parameters for options 3 and 5 or command  8
===> ____________________________________________________________________
F3=Exit   F4=Prompt   F9=Retrieve   F11=Display names only   F12=Cancel  9
```

Figure 104. Example List Panel

The panel shown in Figure 105 on page 288 is an alternate view shown when the user presses F11 (Display names only) on the panel shown in Figure 104. The alternate view shown here uses a multiple-column layout. The panel is divided into four layout columns of equal size. The *Opt* and *Member* columns appear in each layout column. The alternate view shows four times as many list entries as the original view, but does not show the *Type* and *Text* columns for each entry.



```
                            Work with File Members  1

File . . . . . . . . .   XXXXXXXXXX     F4 for list  2
  Library  . . . . . .     XXXXXXXXXX   library, *CURLIB, *LIBL
Type options, press Enter.  3
  3=Copy   4=Remove   5=Display   7=Reorganize   8=Member description  4
  9=Clear

Opt  Member          Opt  Member          Opt  Member          Opt  Member  5
 _   __________  6
 _   XXXXXXXXXX       _   XXXXXXXXXX       _   XXXXXXXXXX       _   XXXXXXXXXX
 _   XXXXXXXXXX       _   XXXXXXXXXX       _   XXXXXXXXXX       _   XXXXXXXXXX
 _   XXXXXXXXXX       _   XXXXXXXXXX       _   XXXXXXXXXX       _   XXXXXXXXXX
 _   XXXXXXXXXX       _   XXXXXXXXXX       _   XXXXXXXXXX       _   XXXXXXXXXX
 _   XXXXXXXXXX       _   XXXXXXXXXX       _   XXXXXXXXXX       _   XXXXXXXXXX
 _   XXXXXXXXXX       _   XXXXXXXXXX       _   XXXXXXXXXX       _   XXXXXXXXXX
 _   XXXXXXXXXX       _   XXXXXXXXXX       _   XXXXXXXXXX       _   XXXXXXXXXX
 _   XXXXXXXXXX       _   XXXXXXXXXX       _   XXXXXXXXXX       _   XXXXXXXXXX
 _   XXXXXXXXXX       _   XXXXXXXXXX       _   XXXXXXXXXX       _   XXXXXXXXXX
                                                                 7  More...
Parameters for options 3 and 5 or command  8
===> ____________________________________________________________________
F3=Exit   F4=Prompt   F9=Retrieve   F11=Display descriptions   F12=Cancel  9
```

Figure 105. Example of Alternate View of List

The reference numbers in the example panel do not appear on the display. They are shown for illustration purposes and also appear in the UIM tag source shown in "Source for Example List Panel" on page 290. These reference numbers show which portions of the source define text and information which appears on the panel.

1 The panel title is defined as text following the period of the display panel (PANEL) tag.

2 The text for the file and library name is defined as text following the period of the data item (DATAI) tag. The file name and library name are dialog variable values. Each dialog variable is defined using the variable definition (VAR) tag. The name of the dialog variable is specified on the VAR attribute of the DATAI tag. The possible choices text for each input field is defined as text following the period of the data item choices (DATAC) tag.

The file name and library are formatted as a qualified object name because GRPSEP=QINDENT is specified on the data group (DATAGRP) tag which surrounds the two DATAI tags.

The dialog variables for the file name and library name are defined using the VAR tags at A and B, respectively.

3 The instruction line is defined as text following the period of the top instruction (TOPINST) tag.

4 Each list option is defined using the list action (LISTACT) tag. The text, including the option number with the equal sign, is defined as text following the period of the LISTACT tag.

5 The column heading for each column is defined as text following the period of the list column (LISTCOL) tag. The VAR attribute of the LISTCOL tag identifies the name of the dialog variable whose value is displayed under the column heading.

The dialog variable name specified, must be defined using the VAR tag. The dialog variable name must also be specified on the VARS attribute of the list definition (LISTDEF) tag for the list appearing in the panel.

The dialog variables for the option, member name, member type, and description text are defined using the VAR tags at C, D, E, and F, respectively.

The names of these variables also appear in the VARS attribute of the LISTDEF tag at **G**. The name of the list definition must be specified on the LISTDEF attribute of the list area (LIST) tag at **H**.

6 The extended action entry that appears immediately below the column headings is defined by specifying EXTACT=YES on the LIST tag. Each column which has an input field in the extended action entry must specify EXTACT=YES on the LISTCOL tag.

The extended action entry allows the user to perform actions without having to scroll to a specific entry in the list.

Each list option defined by the LISTACT tag, specifies whether the action is allowed to be performed against the extended action entry, the list entries appearing below the extended action entry, or both. This specification is made using the ACTFOR attribute of the LISTACT tag.

7 The scroll information, More... and Bottom, are shown because the list area is defined as scrollable because the SCROLL attribute defaults to YES on the LIST tag.

8 The command line is defined using the command line (CMDLINE) tag. The command line prompt text is defined as text following the period of the CMDLINE tag. The UIM automatically provides the arrow (===>) for every command line.

The command line can be used to enter parameters for list options. Before processing list options, the UIM places the contents of the command line into a dialog variable. The dialog variable can be:

- Substituted into a CL command specified as the action to perform on a LISTACT tag
- Used by an exit program specified as the action to perform on the LISTACT tag
- Used by the application program when it is performing the actions for the list

The dialog variable used to contain parameters from the command line is defined using the VAR tag at **I**. It is also specified on the PARMS attribute of the list area (LIST) tag at **J**. The UIM places the contents of the command line in the dialog variable specified by the PARMS attribute before beginning list option processing.

Then the dialog variable is used as a substitution variable in a command string on a LISTACT tag at **K**.

9 Each function key is defined using the key list (KEYL) tag. The text, including the function key name, is defined as text following the period of the KEYI tag.

All function keys are defined by placing the KEYI tags between the key list (KEYL) and EKEYL tags. The name specified on the NAME attribute of the KEYL tag is also specified on the KEYL attribute of the PANEL tag.

Some function keys, such as Enter, do not have text specified for the KEYI tag. In this case, no text appears on the display. However, all function keys must be defined using the KEYI tag so the UIM knows what action is assigned to each key and what help module to use when help is displayed for the function keys.

Required Tags for a List Panel

Figure 106 on page 290 shows the required tags for creating a UIM list panel. For an example of the required and optional tags, see “Source for Example List Panel” on page 290.

Note: The tags in Figure 106 on page 290 require attributes. These attributes are not shown. Without these attributes, the example in Figure 106 on page 290 will not compile. For a description of the required attributes for these required tags, see Appendix A, “UIM Panel Group Definition Language.”

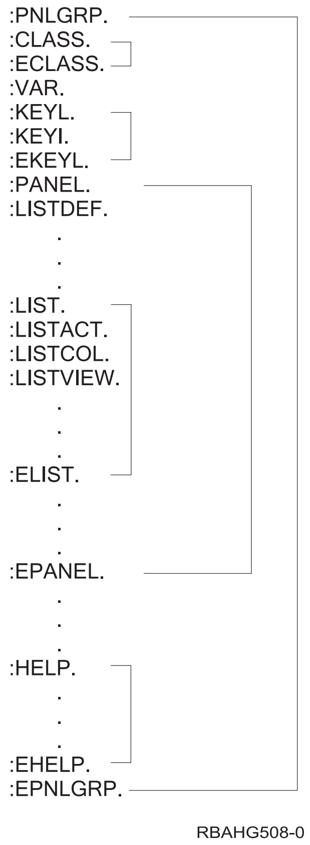


Figure 106. Required tags for a list panel

Source for Example List Panel

This is a partial listing of member T0011PN2 in source file QATTUIM in library QUSRTOOL.

```
.* ------------------------------------------------------------------
.*
.* Beginning of panel group source.
.*
.* ------------------------------------------------------------------
:PNLGRP   DFTMSGF=t0011msgf2
          SUBMSGF=t0011msgf2.
.*
.* The import tag specifies that all help is to be found
.* in panel group T0011HL2 searching the library list.
:IMPORT   NAME='*'
          PNLGRP=t0011hl2.
.*
.* ------------------------------------------------------------------
.* Define all variable classes
.* ------------------------------------------------------------------
.* -----------------
.* Option
.* Note: Need WIDTH=1 to preserve column alignment on confirmation panel.
:CLASS    NAME=optcl
          BASETYPE='ACTION'
          WIDTH=1.
:ECLASS.
```

```
.* -----------------
.* Object name
:CLASS    NAME=namecl
          BASETYPE='OBJNAME 10'.
:ECLASS.
.* -----------------
.* Library name
:CLASS    NAME=libcl
          BASETYPE='OBJNAME 10'.
:TL.
:TI       VALUE='"*LIBL"'.*LIBL
:TI       VALUE='"*CURLIB"'.*CURLIB
:ETL.
:ECLASS.
.* -----------------
.* File attribute
:CLASS    NAME=attrcl
          BASETYPE='CHAR 10'
          CASE=UPPER.
:ECLASS.
.* -----------------
.* Descriptive text
:CLASS    NAME=textcl
          BASETYPE='IGC 50'
          SUBST=QUOTED.
:ECLASS.
.* -----------------
.* Source type
:CLASS    NAME=srctypcl
          BASETYPE='CHAR 10'.
:ECLASS.
  ⋮
  Additional CLASS tags in member T0011PN2 are not shown here
  ⋮
.* -----------------
.* Command line parameters
:CLASS    NAME=parmcl
          BASETYPE='CHAR 255'.
:ECLASS.
.* -----------------
.* Exit program specification for CALL dialog command
:CLASS    NAME=exitcl
          BASETYPE='CHAR 20'.
:ECLASS.
.* -----------------
.* View number
:CLASS    NAME=vwnumcl
          BASETYPE='BIN 15'.
:ECLASS.
.* -----------------
.* Classes for pad variables in variable record definitions.
:CLASS    NAME=pad1cl
          BASETYPE='CHAR 1'.
:ECLASS.
:CLASS    NAME=pad2cl
          BASETYPE='CHAR 2'.
:ECLASS.
:CLASS    NAME=pad10cl
          BASETYPE='CHAR 10'.
:ECLASS.
:CLASS    NAME=pad13cl
          BASETYPE='CHAR 13'.
```

```
:ECLASS.
:CLASS    NAME=pad48cl
          BASETYPE='CHAR 48'.
:ECLASS.
:CLASS    NAME=pad50cl
          BASETYPE='CHAR 50'.
:ECLASS.
.*
.* -----------------------------------------------------------------
.* Define all dialog variables
.* -----------------------------------------------------------------
.*
.* -------------------------------------
.* Variables for file and library
.* -------------------------------------
.* -----------------
.* File name
:VAR      NAME=file  A
           CLASS=namecl.
.* -----------------
.* Library name
:VAR      NAME=lib   B
          CLASS=libcl.
.* -----------------
.* attributes
:VAR      NAME=fattr
          CLASS=attrcl.
.* -----------------
.*
.* -------------------------------------
.* Variables for list of members
.* -------------------------------------
.* -----------------
.* option for list of members
:VAR      NAME=mopt  C
          CLASS=optcl.
.* -----------------
.* Object name
:VAR      NAME=mbr   D
          CLASS=namecl.
.* -----------------
.* member type
:VAR      NAME=mtype   E
          CLASS=attrcl.
.* -----------------
.* Descriptive text
:VAR      NAME=mtext   F
          CLASS=textcl.
   ⋮
  Additional VAR tags in member T0011PN2 are not shown here
   ⋮
.*
.* -------------------------------------
.* Variable for command line parameters
.* -------------------------------------
.* -----------------
.* Command line parameters
:VAR      NAME=parms  I
           CLASS=parmcl.
.*
.* -------------------------------------
.* Variables for specifying CALL/exit programs
.* -------------------------------------
.* -----------------
.* Program to call for all UIM exits
:VAR      NAME=exitpgm
```

```
          CLASS=exitcl.
.*
.* -------------------------------------
.* Variables for controlling list views
.* -------------------------------------
.* -----------------
.* View number for list of members
:VAR      NAME=mbrview
          CLASS=vwnumcl.
   ⋮
  Additional VAR tags in member T0011PN2 are not shown here
   ⋮
.*
.* -------------------------------------
.* Variables for padding in variable record definitions.
.* Padding is needed in variable records so the layout
.* of the record matches a list format returned from
.* an API.  The pad variables are used as placeholders for
.* variables not used in the API format or for reserved space
.* in the API format.
.* -------------------------------------
:VAR      NAME=pad1
          CLASS=pad1cl.
:VAR      NAME=pad2
          CLASS=pad2cl.
:VAR      NAME=pad10
          CLASS=pad10cl.
:VAR      NAME=pad13
          CLASS=pad13cl.
:VAR      NAME=pad48
          CLASS=pad48cl.
:VAR      NAME=pad50
          CLASS=pad50cl.
.*
.* -----------------------------------------------------------------
.* Define a variable record for file, library and file attribute
.* -----------------------------------------------------------------
:VARRCD   NAME=filelib
          VARS='file lib fattr'
          NOGET='fattr'
          .
.*
.* -----------------------------------------------------------------
.* Define a variable record for exit program
.* -----------------------------------------------------------------
:VARRCD   NAME=exitprog
          VARS='exitpgm'
          .
.*
.* -----------------------------------------------------------------
.* Define a variable record for list of members.
.* The layout of this record is designed to match the
.* List Database File Members API (QUSLMBR) format name MBRL0200.
.* -----------------------------------------------------------------
:VARRCD   NAME=mbrl0200
          VARS='mbr mtype pad13 pad13 mtext'
          NOPUT='pad13'
          NOGET='mtype pad13 mtext'
          .
   ⋮
  Additional VARRCD tags in member T0011PN2 are not shown here
   ⋮
.*
.*
.* -----------------------------------------------------------------
.* Define a list of members
.* -----------------------------------------------------------------
G
:LISTDEF  NAME=mbrlist
          VARS='mopt mbr mtype mtext'
          MSGID=USR0101.
   ⋮
  Additional LISTDEF tags in member T0011PN2 are not shown here
   ⋮
.*
.* -----------------------------------------------------------------
.* Define all conditions
.* -----------------------------------------------------------------
.* -----------------
.* Condition for physical files
:COND     NAME=pf
          EXPR='fattr="PF        "'.
.* -----------------
.* Conditions for views of members list
:COND     NAME=mbrview1
          EXPR='mbrview=0'.
:COND     NAME=mbrview2
          EXPR='mbrview=1'.
   ⋮
  Additional COND tags in member T0011PN2 are not shown here
   ⋮
.*
.* -----------------------------------------------------------------
.* Define truth table to specify that mbrview1 and mbrview2
.* are mutually exclusive conditions
.* This will cause UIM to reserve only one line of function
.* keys on the work with members panel
.* -----------------------------------------------------------------
:TT       NAME=mbrtt
          CONDS='mbrview1 mbrview2'.
:TTROW    VALUES='   1        0   '.
:TTROW    VALUES='   0        1   '.
:ETT.
   ⋮
  Additional TT tags in member T0011PN2 are not shown here
   ⋮
   ⋮
  All MBAR tags in member T0011PN2 are not shown here
   ⋮
.*
.*
.* -----------------------------------------------------------------
.* Define keys for work with members panel
.* -----------------------------------------------------------------
:KEYL     NAME=mbrkeys   9
          HELP=keyl.
:KEYI     KEY=F1
          HELP=helpf1
          ACTION=HELP.
:KEYI     KEY=F3
          HELP=exit
          ACTION='EXIT SET'
          VARUPD=NO.
F3=Exit
:KEYI     KEY=F4
          HELP=prompt
          ACTION=PROMPT
          PRIORITY=30.
F4=Prompt
:KEYI     KEY=F9
          HELP=retrieve
          ACTION=RETRIEVE
          PRIORITY=35.
F9=Retrieve
:KEYI     KEY=F11
          HELP=mbrviewname
          ACTION=CHGVIEW
          PRIORITY=25
          COND=mbrview1.
F11=Display names only
:KEYI     KEY=F11
          HELP=mbrviewdesc
          ACTION=CHGVIEW
          PRIORITY=25
          COND=mbrview2.
F11=Display descriptions
:KEYI     KEY=F12
          HELP=cancel
          ACTION='CANCEL SET'
          VARUPD=NO.
F12=Cancel
:KEYI     KEY=F24
          HELP=morekeys
          ACTION=MOREKEYS.
F24=More keys
:KEYI     KEY=ENTER
          HELP=enter
          ACTION=ENTER.
:KEYI     KEY=HELP
          HELP=help
          ACTION=HELP.
:KEYI     KEY=PAGEDOWN
          HELP=pagedown
          ACTION=PAGEDOWN.
:KEYI     KEY=PAGEUP
        HELP=pageup
          ACTION=PAGEUP.
:KEYI     KEY=PRINT
          HELP=print
          ACTION=PRINT.
:EKEYL.
   ⋮
  Additional KEYL tags in member T0011PN2 are not shown here
   ⋮
.*
.*
.* ----------------------------------------------------------------
.* Define Work with Members panel
.* ----------------------------------------------------------------
:PANEL    NAME=wrkmbr
          HELP='wrkmbr/'
          KEYL=mbrkeys   9
          TT=mbrtt
          ENTER='RETURN 500'
          TOPSEP=SPACE.
Work with File Members   1
.*
.* ------------------------------------
.* Define a data presentation area to display the
.* library/file name whose members are listed.
.* ------------------------------------
:DATA     DEPTH=3
          SCROLL=NO
          LAYOUT=1
          BOTSEP=SPACE
          COMPACT
          .
```

```
.* ------------------------------------
.* Divide the layout width into two columns.
.* The first column is for the prompt text with leader dots.
.* The second column is for the variable values.
:DATACOL  WIDTH=22.
:DATACOL  WIDTH=12.
:DATACOL  WIDTH='*'.
.* ------------------------------------
.* Display qualified file name
:DATAGRP  GRPSEP=QINDENT
          HELP='wrkmbr/filelib'
          COMPACT
          .
:DATAI    VAR=file  2
          USAGE=INOUT
          PROMPT='CALL exitpgm'
          .
File
:DATAC.F4 for list
:DATAI    VAR=lib
          USAGE=INOUT
          .
Library
:DATAC.library, *CURLIB, *LIBL
:EDATAGRP.
.*
:EDATA.
.*
.* ------------------------------------
.* Define the list area
.* ------------------------------------
:LIST     DEPTH='*'  7
 LISTDEF=mbrlist H
          ACTOR=UIM
          EXTACT=YES  6
          MAXHEAD=4
          MAXACTL=3
          VIEW=mbrview
          PARMS=parms J
          BOTSEP=SPACE.
:TOPINST.Type options, press Enter.  3
.*
.* ------------------------------------
.* Specify the action to be taken for each option
.* ------------------------------------
:LISTACT  OPTION=3
          ACTFOR=BOTH  6
          NOCMD=PROMPT
          NOEXT=PROMPT
          HELP='wrkmbr/cpyf'
          ENTER='CMD CPYF ?*FROMFILE(&lib/&file)'
          ENTER=' ?*FROMMBR(&mbr) &parms' K
          PROMPT='CMD ?CPYF ?*FROMFILE(&lib/&file)'
          PROMPT=' ?*FROMMBR(&mbr) &parms'  K
          EXTENTER='CMD ?CPYF ?*FROMFILE(&lib/&file)'
          EXTENTER=' ??FROMMBR(&mbr) &parms'  K
          EXTPROMPT='CMD ?CPYF ?*FROMFILE(&lib/&file)'
          EXTPROMPT=' ??FROMMBR(&mbr) &parms'.  K
3=Copy  4
.*
:LISTACT  OPTION=4
          ACTFOR=LISTE
          HELP='wrkmbr/rmvm'
          ENTER='CMD RMVM FILE(&lib/&file) MBR(&mbr)'
          PROMPT='CMD ?RMVM ?*FILE(&lib/&file) ?*MBR(&mbr)'
          USREXIT='CALL exitpgm'.
4=Remove
```

```
.*
:LISTACT  OPTION=5
          COND=pf
          ACTFOR=BOTH
          NOEXT=PROMPT
          HELP='wrkmbr/dsppfm'
          ENTER='CMD DSPPFM FILE(&lib/&file) MBR(&mbr) &parms'
          PROMPT='CMD DSPPFM ?*FILE(&lib/&file) ?*MBR(&mbr) &parms'
          EXTPROMPT='CMD DSPPFM ?*FILE(&lib/&file) ??MBR(&mbr) &parms'.
5=Display
.*
:LISTACT  OPTION=7
          COND=pf
          ACTFOR=BOTH
          HELP='wrkmbr/rgzm'
          ENTER='CMD RGZPFM FILE(&lib/&file) MBR(&mbr)'
          PROMPT='CMD ?RGZPFM ?*FILE(&lib/&file) ?*MBR(&mbr)'.
7=Reorganize
.*
:LISTACT  OPTION=8
          ACTFOR=BOTH
          HELP='wrkmbr/dspfd'
          ENTER='CALL exitpgm'
          PROMPT='CALL exitpgm'.
8=Member description
.*
:LISTACT  OPTION=9
          COND=pf
          ACTFOR=LISTE
 HELP='wrkmbr/clrm'
          ENTER='CMD CLRPFM FILE(&lib/&file) MBR(&mbr)'
          PROMPT='CMD ?CLRPFM ?*FILE(&lib/&file) ?*MBR(&mbr)'.
9=Clear
.*
.*
.* -------------------------------------
.* Define the columns and headings to display
.* -------------------------------------
:LISTCOL  VAR=mopt
          USAGE=INOUT
          EXTACT=YES   6
          HELP='wrkmbr/option'
          MAXWIDTH=6.
Opt   5
:LISTCOL  VAR=mbr
          USAGE=OUT
          EXTACT=YES
          HELP='wrkmbr/mbr'
          MAXWIDTH=10.
Member
:LISTCOL  VAR=mtype
          USAGE=OUT
          HELP='wrkmbr/type'
          MAXWIDTH=10.
Type
:LISTCOL  VAR=mtext
          USAGE=OUT
          HELP='wrkmbr/text'
          MAXWIDTH='*'.
Text
.*
.* -------------------------------------
.* Define multiple views for F11 to toggle between
.* -------------------------------------
:LISTVIEW COLS='mopt mbr mtype mtext'.
:LISTVIEW COLS='mopt mbr' layout=4.
.*
```

```
:ELIST.
.*
.* ------------------------------------
.* Use a command line and allow parameters to be given
.* ------------------------------------
:CMDLINE  SIZE=SHORT.   8
Parameters for options 3 and 5 or command
.*
:EPANEL.
   ⋮
  Additional PANEL tags in member T0011PN2 are not shown here
   ⋮
.*
.* ---------------------------------------------------------------
.* End of panel group source
.* ---------------------------------------------------------------
:EPNLGRP.
```

Application Programming for a List Panel

An example of an application program to display the list panel shown in Figure 104 on page 287 can be found in member T0011CP2 in source file QATTSYSC in library QUSRTOOL. This is an ILE C/C++ program which calls the appropriate UIM application programming interfaces (APIs) to display the panel.

A general example of an RPG application using the UIM APIs can be found in QUSRTOOL. Refer to member T0011INF in source file QATTINFO in library QUSRTOOL. See Chapter 21, “Designing IBM i5/OS-Style Displays,” on page 415 for more information on using the examples.

To write a program in any language to display the example list panel, the program should do the following:

1. Call the Open Display Application (QUIOPNDA) API to open the panel group. The panel group must already be created using the Create Panel Group (CRTPNLGRP) command.
2. Set up a buffer containing the values for the following dialog variables:

   FILE A CHAR 10 variable which is the name of the file.

   LIB A CHAR 10 variable which is the name of the library where the file resides.

   FATTR
   A CHAR 10 variable which is the file attribute of the file. This variable is used to condition on list options which are only allowed for physical files.
3. Call the Put Dialog Variable (QUIPUTV) API to change the contents of the dialog variables using variable record FILELIB and the buffer initialized in the previous step.
4. Set up a buffer containing a value for the following dialog variable:

   EXITPGM
   A CHAR 20 variable which identifies the program to be called as a UIM exit program. For extended program model (EPM) languages, this BASETYPE attribute of the CLASS tag used to define the EXITPGM dialog variable must be changed to a CHAR 130 dialog variable.

   For information on how this dialog variable must be set, see the description of the CALL dialog command in Appendix B, “UIM Dialog Commands,” on page 641.
5. Call the Put Dialog Variable (QUIPUTV) API to change the contents of the dialog variable using variable record EXITPROG and the buffer initialized in the previous step.
6. Create a user space that will receive a list of members in a file using the Create User Space (QUSCRTUS) API. The members in a file are retrieved using the List Database File Members (QUSLMBR) API. If the application programming language supports pointers, use the Retrieve Pointer to User Space (QUSPTRUS) API to obtain a pointer to the contents of the user space. This lets the

application program directly manipulate the data. Otherwise, use the Retrieve User Space (QUSRTVUS) API to obtain the contents of the user space. For a description of these APIs, see The **APIs** topic in the iSeries Information Center:

7. For every member in the file to be displayed, do the following. A list of the members in a file can be retrieved using the List Database File Members (QUSLMBR) API. For a description of this API, see "List Database File Members (QUSLMBR) API" in the APIs topic.
   a. Set up a buffer containing values for the following dialog variables:

      MBR A CHAR 10 variable which is the member name.

      MTYPE
      A CHAR 10 variable which is the member type.

      PAD13
      A CHAR 13 reserved space in the buffer.

      PAD13
      A CHAR 13 reserved space in the buffer.

      MTEXT
      A CHAR 50 variable which is the descriptive text for the member.

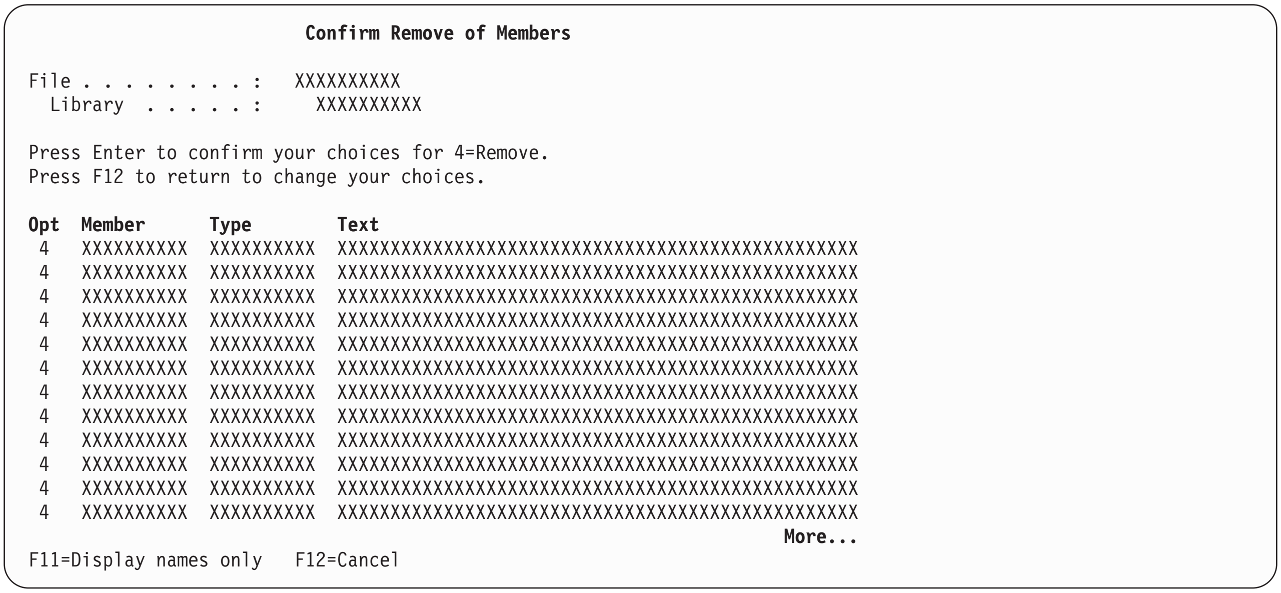
      Note: The layout of this buffer is designed to match the layout of the entries in the user space returned by the QUSLMBR API using the MBRL0200 format. Therefore, instead of setting up a buffer, the application program can pass the buffer as it exists in the user space.

      The two PAD13 variables are used to tell the UIM to ignore two variables in the MBRL0200 format returned by the QUSLMBR API. The variables are the creation date and time and the last source change data and time.
   b. Call the Add List Entry (QUIADDLE) API to add a list entry to the list named MBRLIST using variable record MBRL0200 and the buffer initialized in the previous step.
8. Call the Display Panel (QUIDSPP) API to display panel WRKMBR. The UIM returns control to the application when one of the following occurs:
   - The user presses the Enter key without typing any list options or a command on the command line. The program variable passed as the function requested parameter to the QUIDSPP API is set to 500. This is done because ENTER='RETURN 500' is specified on the PANEL tag which defines the WRKMBR panel.
   - The user presses the F12 (Cancel) key. The program variable passed as the function requested parameter to the QUIDSPP API is set to -8. This is the value defined for the CANCEL dialog command.
   - The user presses the F3 (Exit) key. The program variable passed as the function requested parameter to the QUIDSPP API is set to -4. This is the value defined for the CANCEL dialog command.
9. Call the Close Application (QUICLOA) API to close the UIM application. This frees up the system resources used by the UIM application.

Creating a Confirmation List Panel

Options on an action list panel that perform destructive operations should use a confirmation panel to allow the user to confirm or cancel the request before it is performed. A confirmation panel should be provided for actions such as *4=Remove* and *9=Clear* shown in the example panel in Figure 104 on page 287.

An example of a confirmation panel is shown in Figure 107 on page 300. This confirmation panel is used when the user types option 4 on the panel shown in Figure 104 on page 287.



```
                         Confirm Remove of Members

File . . . . . . . . :   XXXXXXXXXX
  Library  . . . . . :     XXXXXXXXXX

Press Enter to confirm your choices for 4=Remove.
Press F12 to return to change your choices.

Opt  Member      Type        Text
 4   XXXXXXXXXX  XXXXXXXXXX  XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
 4   XXXXXXXXXX  XXXXXXXXXX  XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
 4   XXXXXXXXXX  XXXXXXXXXX  XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
 4   XXXXXXXXXX  XXXXXXXXXX  XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
 4   XXXXXXXXXX  XXXXXXXXXX  XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
 4   XXXXXXXXXX  XXXXXXXXXX  XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
 4   XXXXXXXXXX  XXXXXXXXXX  XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
 4   XXXXXXXXXX  XXXXXXXXXX  XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
 4   XXXXXXXXXX  XXXXXXXXXX  XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
 4   XXXXXXXXXX  XXXXXXXXXX  XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
 4   XXXXXXXXXX  XXXXXXXXXX  XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
 4   XXXXXXXXXX  XXXXXXXXXX  XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
                                                                        More...
F11=Display names only   F12=Cancel
```

Figure 107. Example Confirmation List Panel

The confirmation panel appears similar to the action list panel with the following exceptions:

- The instructions tell the user to confirm or cancel the option.
- The displayed list is for output only. The user cannot change any of the option numbers.
- The F3 (Exit) key is not allowed from a confirmation panel.
- There is no command line on a confirmation panel.

When the user presses the F11 key to change to a different view, the view of the action list panel is changed when it is redisplayed. This is done because the same dialog variable is specified on the VIEW attribute of the LIST tag for both the action list panel and the confirmation panel.

Required Tags for a Confirmation List Panel

The required tags for a confirmation list panel are basically the same as those required for a list panel which are described in Figure 106 on page 290. However, the LISTACT tag is not required.

Source for Example Confirmation Panel

To define this confirmation panel, additional source needs to be added to the source shown in “Source for Example List Panel” on page 290. The following are excepts from member T0011PN2 in source file QATTUIM in library QUSRTOOL. First, a new list definition needs to be made. This list definition contains a copy of the list entries being confirmed. The confirmation list contains the same dialog variables as are contained in the list of members. The following list definition (LISTDEF) tag should be placed after the LISTDEF tag at G .

```
.*
.* ---------------------------------------------------------------
.* Define a list of members for confirmation panels
.* ---------------------------------------------------------------
:LISTDEF  NAME=mbrconf
          VARS='mopt mbr mtype mtext'.
```

Next, function keys need to be defined for the confirmation panel. Confirmation panels should not allow F3 (Exit). The following key list (KEYL) tag should be placed after the KEYL tag, shown in “Source for Example List Panel” on page 290.

```
.*
.* ------------------------------------------------------------------
.* Define keys for confirm remove of members
.* ------------------------------------------------------------------
:KEYL     NAME=mconfkeys
          HELP=keyl.
:KEYI     KEY=F1
          HELP=helpf1
          ACTION=HELP.
:KEYI     KEY=F11
          HELP=altview
          ACTION=CHGVIEW
          COND=mbrview1.
F11=Display names only
:KEYI     KEY=F11
          HELP=altview
          ACTION=CHGVIEW
          COND=mbrview2.
F11=Display descriptions
:KEYI     KEY=F12
          HELP=cancel
          ACTION='CANCEL SET'
          VARUPD=NO.
F12=Cancel
:KEYI     KEY=F24
          HELP=morekeys
          ACTION=MOREKEYS.
F24=More keys
:KEYI     KEY=ENTER
          HELP=enter
          ACTION=ENTER.
:KEYI     KEY=HELP
          HELP=help
          ACTION=HELP.
:KEYI     KEY=PAGEDOWN
          HELP=pagedown
          ACTION=PAGEDOWN.
:KEYI     KEY=PAGEUP
          HELP=pageup
          ACTION=PAGEUP.
:KEYI     KEY=PRINT
          HELP=print
          ACTION=PRINT.
:EKEYL.
```

The confirmation panel needs to be defined. The following source should be placed after the EPANEL tag, shown in "Source for Example List Panel" on page 290.

```
.*
.* ------------------------------------------------------------------
.* Define panel for confirm remove of members from WRKMBR panel
.* ------------------------------------------------------------------
:PANEL    NAME=confrmvm
          HELP='confrmvm/'
          KEYL=mconfkeys
          TT=mbrtt
          ENTER='RETURN 100'
          TOPSEP=space.
Confirm Remove of Members
.*
.* ------------------------------------
.* Define a data presentation area to display the
.* library/file name whose members are listed.
.* ------------------------------------
:DATA     DEPTH=3
          SCROLL=NO
          LAYOUT=1
          BOTSEP=SPACE
          COMPACT
          .
.* -------------------------------------
.* Divide the layout width into two columns.
.* The first column is for the prompt text with leader dots.
.* The second column is for the variable values.
:DATACOL  WIDTH=22.
:DATACOL  WIDTH='*'.
.* -------------------------------------
.* Display qualified file name
:DATAGRP  GRPSEP=QINDENT
          HELP='wrkmbr/filelib'
          COMPACT
          .
:DATAI    VAR=file
          USAGE=OUT
          .
File
:DATAI    VAR=lib
          USAGE=OUT
          .
Library
:EDATAGRP.
.*
:EDATA.
.*
.* -------------------------------------
.* Define the list area
.* -------------------------------------
:LIST     DEPTH='*'
          MAXHEAD=4
          LISTDEF=mbrconf
          VIEW=mbrview.
:TOPINST.Press Enter to confirm your choices for 4=Remove.
:TOPINST.Press F12 to return to change your choices.
.*
.* -------------------------------------
.* Define the columns of the list
.* -------------------------------------
:LISTCOL  VAR=mopt
          USAGE=OUT
          HELP='confrmvm/option'
          MAXWIDTH=6.
Opt
:LISTCOL  VAR=mbr
          USAGE=OUT
          HELP='wrkmbr/mbr'
          MAXWIDTH=10.
Member
:LISTCOL  VAR=mtype
          USAGE=OUT
          HELP='wrkmbr/type'
          MAXWIDTH=10.
Type
:LISTCOL  VAR=mtext
          USAGE=OUT
          HELP='wrkmbr/text'
          MAXWIDTH='*'.
Text
.*
.* -------------------------------------
.* Define multiple views for F11 to toggle between
.* -------------------------------------
:LISTVIEW COLS='mopt mbr mtype mtext'.
```

```
:LISTVIEW COLS='mopt mbr' layout=4.
.*
:ELIST.
:EPANEL.
```

Automatic Confirmation Processing

The UIM provides support to automatically perform confirmation processing. This support is available when the UIM is in control of processing list options by specifying ACTOR=UIM on the LIST tag that defines the action list.

To have the UIM perform confirmation processing for a list option, specify the name of the confirmation panel on the CONFIRM attribute of the list action (LISTACT) tag that defines the option to be confirmed. The following UIM source shows the LISTACT tag for option 4 with the CONFIRM attribute specifying the name of the confirmation panel.

```
.*
:LISTACT  OPTION=4
          ACTFOR=LISTE
          HELP='wrkmbr/rmvm'
          CONFIRM=confrmvm
          ENTER='CMD RMVM FILE(&lib/&file) MBR(&mbr)'
          PROMPT='CMD ?RMVM ?*FILE(&lib/&file) ?*MBR(&mbr)'
          USREXIT='CALL exitpgm'.
4=Remove
.*
```

For more information about defining a confirmation panel using the CONFIRM attribute, see “LISTACT (List Action)” on page 562.

When the UIM processes list options and finds an option with the CONFIRM attribute specified, the UIM does the following:

1. Deletes the list specified on the LISTDEF attribute of the LIST tag in the confirmation panel.
2. Finds all entries in the action list with an option number the same as the option number being confirmed, and copies these entries to the confirmation list.
3. Displays the confirmation panel.
4. Remembers that the option number has been confirmed by the user if the user presses the Enter key. The confirmed options are processed in the order they appear in the action list. No further confirmation processing is done for this option number until the user types the option number for additional entries in the action list.
5. Stops processing list options, if the user presses the F12 (Cancel) key, and redisplays the action list panel showing the first entry with the option number that was not confirmed.

Application Programming for Confirmation Processing

When ACTOR=UIM is specified on the LIST tag in the action list panel, there is no application programming needed because the UIM performs all confirmation processing.

When ACTOR=CALLER is specified on the LIST tag in the action list panel, the application program that processes the list options should perform confirmation processing similar to the UIM processing as described in “Automatic Confirmation Processing.”

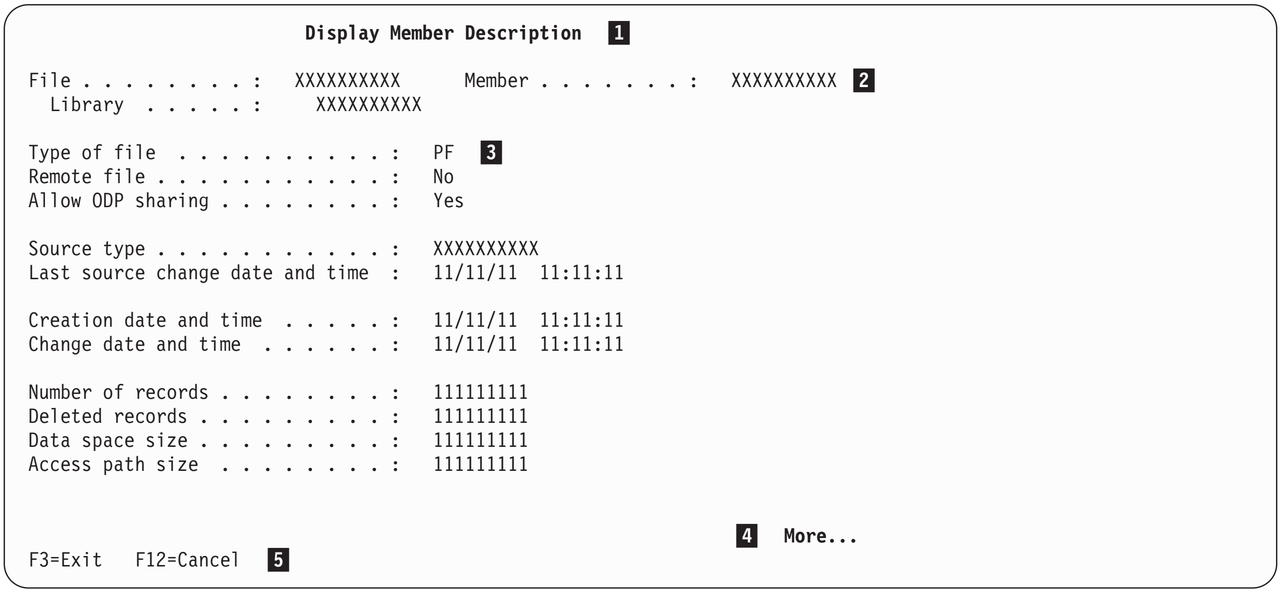
Creating a Data Presentation Panel

Data presentation panels are used to display user data or allow input of user data in connection with an option on a list panel. When data input/output is scrollable and low in frequency and volume, UIM simplifies creating consistent data presentation panels.

The following shows the coding to create the Display Member Description display from the Work with File Members display.

The panel shown in Figure 108 shows an example of a data presentation panel. This panel is shown as a result of using option 5 on the panel shown in Figure 104 on page 287.

This panel contains two data presentation areas. The first area is using a vertical layout with two layout columns. This area shows the file, library, and member name. The second area also uses a vertical layout, but only one layout column. This area shows the detail information for the member identified in the first area. If you need more information on creating help for a data presentation panel, see "Help in a Data Area" on page 391.

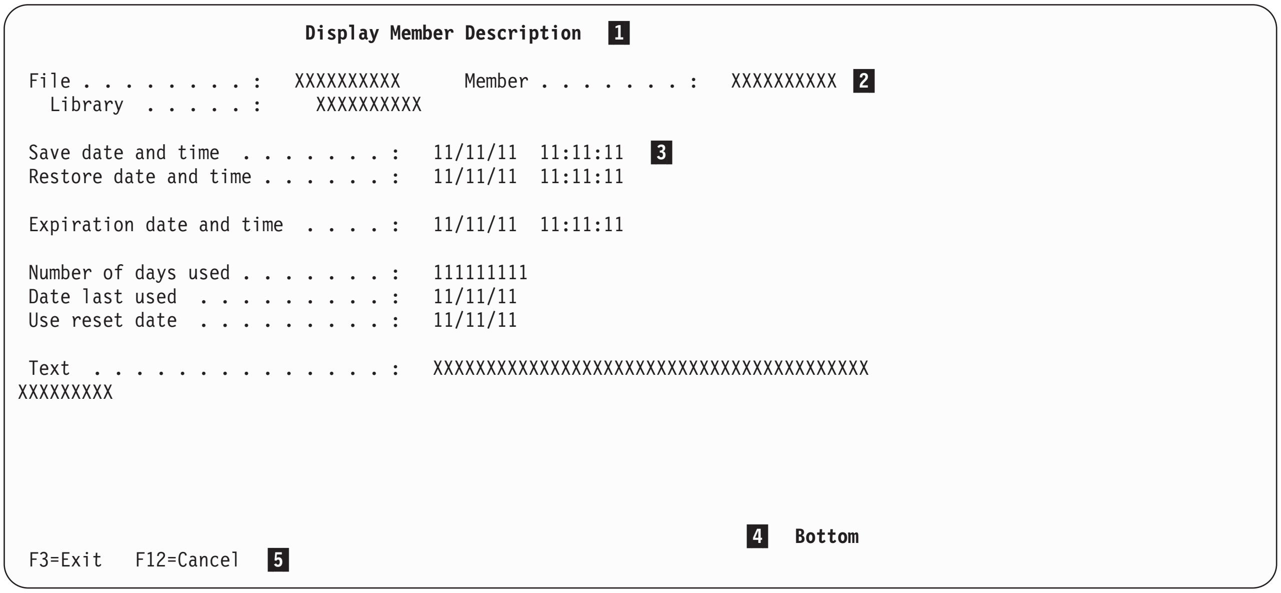


```
                           Display Member Description  1

File . . . . . . . . :   XXXXXXXXXX      Member . . . . . . . :   XXXXXXXXXX  2
  Library  . . . . . :     XXXXXXXXXX

Type of file  . . . . . . . . . . . :   PF  3
Remote file . . . . . . . . . . . . :   No
Allow ODP sharing . . . . . . . . . :   Yes

Source type . . . . . . . . . . . . :   XXXXXXXXXX
Last source change date and time  :   11/11/11  11:11:11

Creation date and time  . . . . . . :   11/11/11  11:11:11
Change date and time  . . . . . . . :   11/11/11  11:11:11

Number of records . . . . . . . . . :   111111111
Deleted records . . . . . . . . . . :   111111111
Data space size . . . . . . . . . . :   111111111
Access path size  . . . . . . . . . :   111111111


                                                                     4   More...
F3=Exit   F12=Cancel  5
```

Figure 108. Example Data Presentation Panel

The second area is defined as a scrollable area because there are more items to show than will fit on a display at one time. The UIM automatically handles scrolling when the user presses the Page Up (Rolldown) or Page Down (Rollup) keys. When the Page Down operation is performed, the contents of the second data presentation area is replaced with the next set of items to show. This results in the panel shown in Figure 109 on page 305.

Notice that the contents of the first area remain the same when the second area is scrolled. The scroll operation only applies to one area of the panel based on the location of the cursor. Also, because the first area is not defined as scrollable, the scroll operation does not apply to that area even when the cursor position is within the area.



```
                          Display Member Description  1

 File . . . . . . . . :   XXXXXXXXXX      Member . . . . . . . :   XXXXXXXXXX 2
   Library  . . . . . :     XXXXXXXXXX

 Save date and time  . . . . . . . :   11/11/11  11:11:11  3
 Restore date and time . . . . . . :   11/11/11  11:11:11

 Expiration date and time  . . . . :   11/11/11  11:11:11

 Number of days used . . . . . . . :   111111111
 Date last used  . . . . . . . . . :   11/11/11
 Use reset date  . . . . . . . . . :   11/11/11

 Text  . . . . . . . . . . . . . . :   XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXX





                                                                    4   Bottom
 F3=Exit   F12=Cancel   5
```

Figure 109. Example Data Presentation Panel after Scrolling

The reference numbers in the example panel do not appear on the display. They are shown for illustration purposes and also appear in the UIM tag source shown in “Source for Example Data Presentation Panel” on page 306. These reference numbers show which portions of the source define text and information which appears on the panel.

1 The panel title is defined as text following the period of the display panel (PANEL) tag.

2 In the first data area, the text for the file, library, and member name is defined as text following the period of the data item (DATAI) tag. The file, library, and member names are dialog variable values. Each dialog variable is defined using the variable definition (VAR) tag. The name of the dialog variable is specified on the `VAR` attribute of the DATAI tag.

The file name and library are formatted as a qualified object name because `GRPSEP`=QINDENT is specified on the data group (DATAGRP) tag which surrounds the two DATAI tags.

The dialog variables for the file name and library name and member name are defined using the VAR tags at **A**, **B**, and **C**, respectively.

This data presentation area is defined by specifying `LAYOUT=2` on the DATA tag that defines the area. This is shown at **D**.

3 In the second data area, the text for each item is defined as text following the period of the DATAI tag. The value for each item is a dialog variable value. Each dialog variable is defined using the VAR tag. The name of the dialog variable is specified on the `VAR` attribute of the DATAI tag.

Several items in the area show date and time values. This is done by specifying the dialog variable for the date value on the `VAR` attribute of the DATAI tag and specifying the dialog variable for the time value on the `VAR` attribute of the data item extender (DATAIX) tag.

The date and time dialog variables are defined by specifying `BASETYPE`=DATE and `BASETYPE`=TIME on the class definition (CLASS) tag used to define the variables. This is shown at **E**. The UIM formats the date and time variables according to the date format and separator attributes and the time separator attribute of the job.

4 The scroll information, More... and Bottom, is shown because the list area is defined as scrollable. `SCROLL`=YES is specified on the DATA tag.

5 Each function key is defined using the key list item (KEYI) tag. The text, including the function key name, is defined as text following the period of the KEYI tag.

All function keys are defined by placing the KEYI tags between the key list (KEYL) and EKEYL tags. The name specified on the `NAME` attribute of the KEYL tag is also specified on the KEYL attribute of the PANEL tag.

Some function keys, such as Enter, do not have text specified for the KEYI tag. In this case, no text appears on the display. However, all function keys must be defined using the KEYI tag so the UIM knows what action is assigned to each key and what help module to use when help is displayed for the function keys.

Required Tags for a Data Presentation Panel

Figure 110 shows the required tags for creating a UIM data presentation panel. For an example of the required and optional tags, see “Source for Example Data Presentation Panel.”

Note: The tags in Figure 110 require attributes. These attributes are not shown. Without these attributes, the example in Figure 110 will not compile. For a description of the required attributes for these required tags, see Appendix A, “UIM Panel Group Definition Language.”

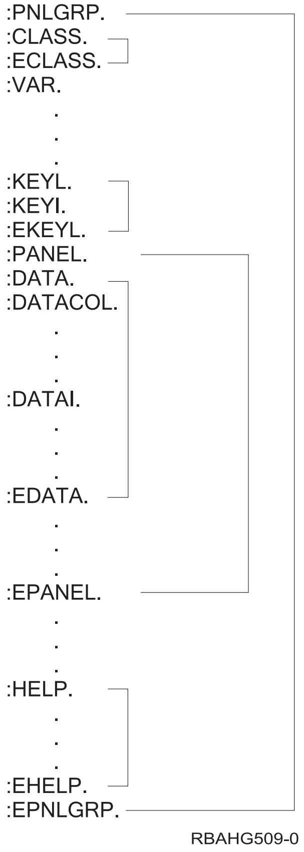


Figure 110. Required UIM tags for a data presentation panel

Source for Example Data Presentation Panel

This is a partial listing of the member T0011PN2 in source file QATTUIM in library QUSRTOOL.

```
.* ---------------------------------------------------------------
.*
.* Beginning of panel group source.
```

```
.*
.* -----------------------------------------------------------------
:PNLGRP   DFTMSGF=t0011msgf2
          SUBMSGF=t0011msgf2.
.*
.* The import tag specifies that all help is to be found
.* in panel group T0011HL2 searching the library list.
:IMPORT   NAME='*'
          PNLGRP=t0011hl2.
.*
.* -----------------------------------------------------------------
.* Define all variable classes
.* -----------------------------------------------------------------
.* -----------------
.* Option
.* Note: Need WIDTH=1 to preserve column alignment on confirmation panel.
:CLASS    NAME=optcl
          BASETYPE='ACTION'
          WIDTH=1.
:ECLASS.
.* -----------------
.* Object name
:CLASS    NAME=namecl
          BASETYPE='OBJNAME 10'.
:ECLASS.
.* -----------------
.* Library name
:CLASS    NAME=libcl
          BASETYPE='OBJNAME 10'.
:TL.
:TI       VALUE='"*LIBL"'.*LIBL
:TI       VALUE='"*CURLIB"'.*CURLIB
:ETL.
:ECLASS.
.* -----------------
.* File attribute
:CLASS    NAME=attrcl
          BASETYPE='CHAR 10'
          CASE=UPPER.
:ECLASS.
.* -----------------
.* Descriptive text
:CLASS    NAME=textcl
          BASETYPE='IGC 50'
          SUBST=QUOTED.
:ECLASS.
.* -----------------
.* Source type
:CLASS    NAME=srctypcl
          BASETYPE='CHAR 10'.
:ECLASS.
.* -----------------
.* Date
:CLASS    NAME=datecl  E
          BASETYPE='DATE'.
:TL.
:TI VALUE='"       "'.
:TI VALUE='"0000000"'.
:ETL.
:ECLASS.
```

```
.* -----------------
.* Time
:CLASS    NAME=timecl   E
          BASETYPE='TIME'.
:TL.
:TI VALUE='"      "'.
:TI VALUE='"000000"'.
:ETL.
:ECLASS.
.* -----------------
.* Yes or No flag class
:CLASS    NAME=yesnocl
          BASETYPE='CHAR 1'
          WIDTH=3.
:TL.
:TI VALUE='"0"'.No
:TI VALUE='"1"'.Yes
:ETL.
:ECLASS.
.* -----------------
.* Type of file
:CLASS    NAME=typfcl
          BASETYPE='CHAR 1'
          WIDTH=2.
:TL.
:TI VALUE='"0"'.PF
:TI VALUE='"1"'.LF
:ETL.
:ECLASS.
.* -----------------
.* Binary 31
:CLASS    NAME=bin31
          BASETYPE='BIN 31'.
:ECLASS.
   ⋮
  Additional CLASS tags in member T0011PN2 are not shown here
   ⋮
.* -----------------
.* Classes for pad variables in variable record definitions.
:CLASS    NAME=pad1cl
          BASETYPE='CHAR 1'.
:ECLASS.
:CLASS    NAME=pad2cl
          BASETYPE='CHAR 2'.
:ECLASS.
:CLASS    NAME=pad10cl
          BASETYPE='CHAR 10'.
:ECLASS.
:CLASS    NAME=pad13cl
          BASETYPE='CHAR 13'.
:ECLASS.
:CLASS    NAME=pad48cl
          BASETYPE='CHAR 48'.
:ECLASS.
:CLASS    NAME=pad50cl
          BASETYPE='CHAR 50'.
:ECLASS.
.*
.* -----------------------------------------------------------------
.* Define all dialog variables
.* -----------------------------------------------------------------
.*
.* -------------------------------------
.* Variables for file and library
.* -------------------------------------
.* -----------------
.* File name
:VAR      NAME=file A
          CLASS=namecl.
.* -----------------
.* Library name
:VAR      NAME=lib  B
          CLASS=libcl.
.* -----------------
.* attributes
:VAR      NAME=fattr
          CLASS=attrcl.
.* -----------------
.*
.* -------------------------------------
.* Variables for list of members
.* -------------------------------------
.* -----------------
.* option for list of members
:VAR      NAME=mopt
          CLASS=optcl.
.* -----------------
.* Object name
:VAR      NAME=mbr  C
          CLASS=namecl.
.* -----------------
.* member type
:VAR      NAME=mtype
          CLASS=attrcl.
.* -----------------
.* Descriptive text
:VAR      NAME=mtext
          CLASS=textcl.
.*
.* -------------------------------------
.* Variables for member description
.* -------------------------------------
.* -----------------
.* Source type
:VAR      NAME=mbrsrc
          CLASS=srctypcl.
.* -----------------
.* Creation date
:VAR      NAME=mbrcrtdat
          CLASS=datecl.
.* -----------------
.* Creation time
:VAR      NAME=mbrcrttim
          CLASS=timecl.
.* -----------------
.* Member last source change date
:VAR      NAME=mbrschgdat
          CLASS=datecl.
.* -----------------
.* Member last source change time
:VAR      NAME=mbrschgtim
          CLASS=timecl.
.* -----------------
.* Member remote source file indicator
:VAR      NAME=mbrsrcfil
          CLASS=yesnocl.
.* -----------------
.* Member remote file indicator
:VAR      NAME=mbrremote
          CLASS=yesnocl.
.* -----------------
.* Member type of file
:VAR      NAME=mbrtypf
          CLASS=typfcl.
.* -----------------
.* Member ODP sharing
:VAR      NAME=mbrodpshr
          CLASS=yesnocl.
.* -----------------
.* Member current number of records
:VAR      NAME=mbrcurrec
          CLASS=bin31.
.* -----------------
.* Member number of deleted records
:VAR      NAME=mbrdltrec
          CLASS=bin31.
.* -----------------
.* Member data space size
:VAR      NAME=mbrspcsiz
          CLASS=bin31.
.* -----------------
.* Member access path size
:VAR      NAME=mbracpsiz
          CLASS=bin31.
.* -----------------
.* Number of database file members
:VAR      NAME=mbrdbfmbrs
          CLASS=bin31.
.* -----------------
.* Member change date.
:VAR      NAME=mbrchgdat
          CLASS=datecl.
.* -----------------
.* Member change time
:VAR      NAME=mbrchgtim
          CLASS=timecl.
.* -----------------
.* Member save date
:VAR      NAME=mbrsavdat
          CLASS=datecl.
.* -----------------
.* Member save time
:VAR      NAME=mbrsavtim
          CLASS=timecl.
.* -----------------
.* Member restore date
:VAR      NAME=mbrrstdat
          CLASS=datecl.
.* -----------------
.* Member restore time
:VAR      NAME=mbrrsttim
          CLASS=timecl.
.* -----------------
.* Member expiration date
:VAR      NAME=mbrexpdat
          CLASS=datecl.
.* -----------------
.* Member expiration time
:VAR      NAME=mbrexptim
          CLASS=timecl.
.* -----------------
.* Member number of days used
:VAR      NAME=mbrdysuse
          CLASS=bin31.
.* -----------------
.* Member date last used
:VAR      NAME=mbrlastuse
          CLASS=datecl.
```

```
.* -----------------
.* Member use reset date
:VAR      NAME=mbrrsetdat
          CLASS=datecl.
.* -----------------
   ⋮
  Additional VAR tags in member T0011PN2 are not shown here
   ⋮
.*
.* -------------------------------------
.* Variables for padding in variable record definitions.
.* Padding is needed in variable records so the layout
.* of the record matches a list format returned from
.* an API.  The pad variables are used as placeholders for
.* variables not used in the API format or for reserved space
.* in the API format.
.* -------------------------------------
:VAR      NAME=pad1
          CLASS=pad1cl.
:VAR      NAME=pad2
          CLASS=pad2cl.
:VAR      NAME=pad10
          CLASS=pad10cl.
:VAR      NAME=pad13
          CLASS=pad13cl.
:VAR      NAME=pad48
          CLASS=pad48cl.
:VAR      NAME=pad50
          CLASS=pad50cl.
   ⋮
  The first VARRCD tags in member T0011PN2 are not shown here
   ⋮
.*
.* -----------------------------------------------------------------
.* Define a variable record for list of members.
.* The layout of this record is designed to match the
.* List Database File Members API (QUSLMBR) format name MBRL0200.
.* -----------------------------------------------------------------
:VARRCD   NAME=mbrl0200
          VARS='mbr mtype pad13 pad13 mtext'
          NOPUT='pad13'
          NOGET='mtype pad13 mtext'
          .
.*
.* -----------------------------------------------------------------
.* Define a variable record for member description.
.* The layout of this record is designed to match the
.* Retrieve Member Description API (QUSRMBRD) format name MBRD0200.
.* -----------------------------------------------------------------
:VARRCD   NAME=mbrd0200
          VARS='pad48 mbrsrc mbrcrtdat mbrcrttim mbrschgdat'
          VARS='mbrschgtim pad50 mbrsrcfil mbrremote mbrtypf'
          VARS='mbrodpshr pad2 mbrcurrec mbrdltrec mbrspcsiz'
          VARS='mbracpsiz mbrdbfmbrs mbrchgdat mbrchgtim mbrsavdat'
          VARS='mbrsavtim mbrrstdat mbrrsttim mbrexpdat mbrexptim'
          VARS='mbrdysuse mbrlastuse mbrrsetdat'
          NOPUT='pad48 pad50 pad2'
         NOGET='pad48 pad50 pad2'
          .
   ⋮
  Additional VARRCD tags in member T0011PN2 are not shown here
   ⋮
   ⋮
  All LISTDEF tags in member T0011PN2 are not shown here
```

```
   ⋮
.*
.* -----------------------------------------------------------------
.* Define all conditions
.* -----------------------------------------------------------------
.* -----------------
.* Condition for physical files
:COND     NAME=pf
          EXPR='fattr="PF        "'.
   ⋮
  Additional COND tags in member T0011PN2 are not shown here
   ⋮
.* -----------------
.* Condition for source files
:COND     NAME=srcpf
          EXPR='mbrsrcfil="1"'.
.* -----------------
.* Condition for member in a physical file
:COND     NAME=mbrpf
          EXPR='mbrtypf="0"'.
.* -----------------
.* Condition for member in a logical file
:COND     NAME=mbrlf
          EXPR='mbrtypf="1"'.
   ⋮
  Additional COND tags in member T0011PN2 are not shown here
   ⋮
   ⋮
  All TT tags in member T0011PN2 are not shown here
   ⋮
   ⋮
  All MBAR tags in member T0011PN2 are not shown here
   ⋮
   ⋮
  The first KEYL tags in member T0011PN2 are not shown here
   ⋮
.*
.* -----------------------------------------------------------------
.* Define basic key for panels without a command line
.* or multiple views.
.* -----------------------------------------------------------------
:KEYL     NAME=basickeys   5
          HELP=keyl.
:KEYI     KEY=F1
          HELP=helpf1
          ACTION=HELP.
:KEYI     KEY=F3
          HELP=exit
          ACTION='EXIT SET'
          VARUPD=NO.
F3=Exit
:KEYI     KEY=F12
          HELP=cancel
          ACTION='CANCEL SET'
          VARUPD=NO.
F12=Cancel
:KEYI     KEY=F24
          HELP=morekeys
          ACTION=MOREKEYS.
F24=More keys
:KEYI     KEY=ENTER
          HELP=enter
          ACTION=ENTER.
:KEYI     KEY=HELP
          HELP=help
          ACTION=HELP.
:KEYI     KEY=PAGEDOWN
          HELP=pagedown
          ACTION=PAGEDOWN.
:KEYI     KEY=PAGEUP
          HELP=pageup
          ACTION=PAGEUP.
:KEYI     KEY=PRINT
          HELP=print
          ACTION=PRINT.
:EKEYL.
   ⋮
  The first PANEL tags in member T0011PN2 are not shown here
   ⋮
.*
.* ------------------------------------------------------------------
.* Define Display Member Description panel
.* ------------------------------------------------------------------
:PANEL    NAME=dspmbr
          HELP='dspmbr/'
          KEYL=basickeys  5
          ENTER='RETURN 500'
          TOPSEP=SPACE.
Display Member Description  1
.*
.* -------------------------------------
.* Define a data presentation area to display the
.* library/file and member name whose description is displayed.
.* -------------------------------------
:DATA     DEPTH=3
          SCROLL=NO
          LAYOUT=2  D
          BOTSEP=SPACE
          COMPACT
          .
.* -------------------------------------
.* Divide the layout width into two columns.
.* The first column is for the prompt text with leader dots.
.* The second column is for the variable values.
:DATACOL  WIDTH=22.
:DATACOL  WIDTH='*'.
.* -------------------------------------
.* Display qualified file name
:DATAGRP  GRPSEP=QINDENT
          HELP='dspmbr/filelib'
          COMPACT
          .
:DATAI    VAR=file  2
          USAGE=OUT
          .
File
:DATAI    VAR=lib  2
          USAGE=OUT

          .
Library
:EDATAGRP.
.* -------------------------------------
.* Display member name
:DATAI    VAR=mbr  2
          HELP='dspmbr/mbr'
          USAGE=OUT
          .
Member
.*
:EDATA.
.*
```

```
.* ----------------------------------------
.* Define a data presentation area to display the
.* member definition.
.* ----------------------------------------
:DATA     DEPTH='*'
          SCROLL=YES   4
          LAYOUT=1
          BOTSEP=SPACE
          .
.* ----------------------------------------
.* Divide the layout width into two columns.
.* The first column is for the prompt text with leader dots.
.* The second column is for the variable values.
:DATACOL  WIDTH=35.
:DATACOL  WIDTH='*'.
.* ----------------------------------------
.* Display information about the file
:DATAGRP  GRPSEP=NONE
          COMPACT
          .
.* ----------------------------------------
.* Display type of file
:DATAI    VAR=mbrtypf   3
          HELP='dspmbr/mbrtypf'
          USAGE=OUT
          .
Type of file
.* ----------------------------------------
.* Display remote file
:DATAI    VAR=mbrremote
          HELP='dspmbr/mbrremote'
          USAGE=OUT
          .
Remote file
.* ----------------------------------------
.* Display ODP sharing
:DATAI    VAR=mbrodpshr
          HELP='dspmbr/mbrodpshr'
          USAGE=OUT
          .
Allow ODP sharing
:EDATAGRP.
.* ----------------------------------------
.* Display information only if file is a source file
:DATAGRP  GRPSEP=NONE
          COMPACT
          COND=srcpf
          .
.* ----------------------------------------
.* Display source type
:DATAI    VAR=mbrsrc
     HELP='dspmbr/mbrsrc'
          USAGE=OUT
          .
Source type
.* ----------------------------------------
.* Display last source change date and time
:DATAI    VAR=mbrschgdat
          HELP='dspmbr/mbrschgdt'
          USAGE=OUT
          .
Last source change date and time
:DATAIX   VAR=mbrschgtim
          USAGE=OUT
          .
:EDATAGRP.
.* ----------------------------------------
.* Display create and change information
:DATAGRP  GRPSEP=NONE
          COMPACT
          .
.* ------------------------------------
.* Display creation date and time
:DATAI    VAR=mbrcrtdat
          HELP='dspmbr/mbrcrtdt'
          USAGE=OUT
          .
Creation date and time
:DATAIX   VAR=mbrcrttim
          USAGE=OUT
          .
.* ------------------------------------
.* Display change date and time
:DATAI    VAR=mbrchgdat
          HELP='dspmbr/mbrchgdt'
          USAGE=OUT
          .
Change date and time
:DATAIX   VAR=mbrchgtim
          USAGE=OUT
          .
:EDATAGRP.
.* ------------------------------------
.* Display information about the size of the member
:DATAGRP  GRPSEP=NONE
          COMPACT
          .
.* ------------------------------------
.* Display current records for physical file member
:DATAI    VAR=mbrcurrec
          COND=mbrpf
          HELP='dspmbr/mbrcurrec'
          USAGE=OUT
          .
Number of records
.* ------------------------------------
.* Display current index entries for logical file member
:DATAI    VAR=mbrcurrec
          COND=mbrlf
          HELP='dspmbr/mbrcurrec'
          USAGE=OUT
          .
Number of index entries
.* ------------------------------------
.* Display deleted records
:DATAI    VAR=mbrdltrec
          HELP='dspmbr/mbrdltrec'
          USAGE=OUT
          .
Deleted records
.* ------------------------------------
.* Display data space size for physical file member
:DATAI    VAR=mbrspcsiz
          COND=mbrpf
          HELP='dspmbr/mbrspcsiz'
          USAGE=OUT
          .
Data space size
.* ------------------------------------
.* Display access path size
:DATAI    VAR=mbracpsiz
          HELP='dspmbr/mbracpsiz'
          USAGE=OUT
          .
```

```
Access path size
.* ----------------------------------------
.* Display database file members for logical file member
:DATAI    VAR=mbrdbfmbrs
          COND=mbrlf
          HELP='dspmbr/mbrdbfmbrs'
          USAGE=OUT
          .
Number of database file members
:EDATAGRP.
.* ----------------------------------------
.* Display save restore information
:DATAGRP  GRPSEP=NONE
          COMPACT
          .
.* ----------------------------------------
.* Display save date and time
:DATAI    VAR=mbrsavdat   3
          HELP='dspmbr/mbrsavdt'
          USAGE=OUT
          .
Save date and time
:DATAIX   VAR=mbrsavtim   3
          USAGE=OUT
          .
.* ----------------------------------------
.* Display restore date and time
:DATAI    VAR=mbrrstdat
          HELP='dspmbr/mbrrstdt'
          USAGE=OUT
          .
Restore date and time
:DATAIX   VAR=mbrrsttim
          USAGE=OUT
          .
:EDATAGRP.
.* ----------------------------------------
.* Display expiration date
:DATAI    VAR=mbrexpdat
          HELP='dspmbr/mbrexpdt'
          USAGE=OUT
          .
Expiration date and time
:DATAIX   VAR=mbrexptim
         USAGE=OUT
          .
.* ----------------------------------------
.* Display usage information
:DATAGRP  GRPSEP=NONE
          COMPACT
          .
.* ----------------------------------------
.* Display number of days used
:DATAI    VAR=mbrdysuse
          HELP='dspmbr/mbrdysuse'
          USAGE=OUT
          .
Number of days used
.* ----------------------------------------
.* Display date last used
:DATAI    VAR=mbrlastuse
          HELP='dspmbr/mbrlastuse'
          USAGE=OUT
          .
Date last used
.* ----------------------------------------
.* Display use reset date
```

```
:DATAI    VAR=mbrrsetdat
          HELP='dspmbr/mbrrsetdat'
          USAGE=OUT
          .
Use reset date
:EDATAGRP.
.* ------------------------------------
.* Display text description
:DATAI    VAR=mtext
          HELP='dspmbr/text'
          USAGE=OUT
          .
Text
.*
:EDATA.
.*
:EPANEL.
   ⋮
  Additional PANEL tags in member T0011PN2 are not shown here
   ⋮
.*
.* ---------------------------------------------------------------
.* End of panel group source
.* ---------------------------------------------------------------
:EPNLGRP.
```

Application Programming for a Data Presentation Panel

An example of an application program to display the data presentation panel shown in Figure 108 on page 304 can be found in member T0011CP3 in file QATTSYSC in library QUSRTOOL. This is an ILE C/C++ program which calls the appropriate UIM application programming interfaces (APIs) to display the panel. This program is called by the UIM to process option 5 (Display) from the example list panel shown in Figure 104 on page 287.

A general example of an RPG application written using the UIM APIs can be found in QUSRTOOL. Refer to member T0011RP5 in source file QATTRPG in library QUSRTOOL.

To write a program in any language to display the example data presentation panel, the program should do the following:

1. If the program is not called by the UIM to process option 5 from the example list panel shown in Figure 104 on page 287, the program should first do the following. For example, this would be the case if the user could display the member description directly by using a CL command.
   a. Call the Open Display Application (QUIOPNDA) API to open the panel group. The panel group must already be created using the Create Panel Group (CRTPNLGRP) command.
   b. Set up a buffer containing the values for the following dialog variables:

      FILE A CHAR 10 variable which is the name of the file.

      LIB A CHAR 10 variable which is the name of the library where the file resides.

      FATTR
      A CHAR 10 variable which is the file attribute of the file. This variable is used to condition on list options which are only allowed for physical files.
   c. Call the Put Dialog Variable (QUIPUTV) API to change the contents of the dialog variables using variable record FILELIB and the buffer initialized in the previous step.
   d. Set up a buffer containing values for the following dialog variables:

      MBR A CHAR 10 variable which is the member name.

      MTYPE
      A CHAR 10 variable which is the member type.

PAD13
A CHAR 13 reserved space in the buffer.

PAD13
A CHAR 13 reserved space in the buffer.

MTEXT
A CHAR 50 variable which is the descriptive text for the member.

Note: The layout of this buffer is designed to match with the layout of the entries in the user space returned by the QUSLMBR API. Therefore, instead of setting up a buffer, the application program can pass the buffer as it exists in the user space.

e. Call the Put Dialog Variable (QUIPUTV) API to change the contents of the dialog variables using variable record MBRL0200 and the buffer initialized in the previous step.

Note: If the program is called by the UIM to process option 5 from the example list panel, the file name and library name were already set by the application program before displaying the list panel. Also, when the UIM performs list option processing, the MBR, MTYPE, and MTEXT dialog variables will already be set to the values from the list entry which calls the application program to process option 5.

2. Set up a buffer containing values for the following dialog variables:

PAD48 A CHAR 48 reserved space in the buffer.

MBRSRC A CHAR 10 variable which is the source type for the member.

MBRCRTDAT A CHAR 7 variable which is the creation date for the member in the form required for BASETYPE=DATE dialog variables.

MBRCRTTIM A CHAR 6 variable which is the creation time for the member in the form required for BASETYPE=TIME dialog variables.

MBRSCHGDAT
A CHAR 7 variable which is the source changed date for the member in the form required for BASETYPE=DATE dialog variables.

MBRSCHGTIM
A CHAR 6 variable which is the source changed time for the member in the form required for BASETYPE=TIME dialog variables.

PAD50 A CHAR 50 reserved space in the buffer.

MBRSRCFIL A CHAR 1 variable indicating whether or not the member is in a source file (″0″=no, ″1″=yes).

MBRREMOTE
A CHAR 1 variable indicating whether or not the member is in a remote file (″0″=no, ″1″=yes).

MBRTYPF A CHAR 1 variable indicating whether the member is in a physical or logical file (″0″=physical ″1″=logical).

MBRODPSHR
A CHAR 1 variable indicating whether or not ODP sharing is allowed for the member (″0″=no, ″1″=yes).

PAD2 A CHAR 2 reserved space in the buffer.

MBRCURREC
A BINARY 4 variable containing the number of records in the member.

MBRDLTREC A BINARY 4 variable containing the number of deleted records in the member.

MBRSPCSIZ A BINARY 4 variable containing the size of the space for the member.

MBRACPSIZ A BINARY 4 variable containing the access path size for the member.

MBRDBFMBRS
A BINARY 4 variable containing the number of database file members for this logical file member.

MBRCHGDAT
A CHAR 7 variable which is the last changed date for the member in the form required for BASETYPE=DATE dialog variables.

MBRCHGTIM
A CHAR 6 variable which is the last changed time for the member in the form required for BASETYPE=TIME dialog variables.

MBRSAVDAT A CHAR 7 variable which is the date the member was last saved in the form required for BASETYPE=DATE dialog variables.

MBRSAVTIM A CHAR 6 variable which is the time the member was last saved in the form required for BASETYPE=TIME dialog variables.

MBRRSTDAT A CHAR 7 variable which is the date the member was last restored in the form required for BASETYPE=DATE dialog variables.

MBRRSTTIM A CHAR 6 variable which is the time the member was last restored in the form required for BASETYPE=TIME dialog variables.

MBREXPDAT A CHAR 7 variable which is the date of expiration for the member in the form required for BASETYPE=DATE dialog variables.

MBREXPTIM A CHAR 6 variable which is the time of expiration for the member in the form required for BASETYPE=TIME dialog variables.

MBRDYSUSE A BINARY 4 variable containing the number of days the member has been used.

MBRLASTUSE
A CHAR 7 variable which is the date the member was last used in the form required for BASETYPE=DATE dialog variables.

MBRRSETDAT
A CHAR 7 variable which is the date the use information was reset for the member in the form required for BASETYPE=DATE dialog variables.

Note: The layout of this buffer is designed to match with the layout of the data returned by the Retrieve Member Description (QUSRMBRD) API using format name MBRD0200. Therefore, instead of setting up a buffer, the application program can pass the buffer as it is returned by the API.

3. Call the Put Dialog Variable (QUIPUTV) API to change the contents of the dialog variables using variable record MBRD0200 and the buffer initialized in the previous step.
4. Call the Display Panel (QUIDSPP) API to display panel WRKMBR. The UIM returns control to the application when one of the following occurs:
   - The user presses the Enter key without typing any list options or a command on the command line. The program variable passed as the function requested parameter to the QUIDSPP API is set to 500. This is done because ENTER='RETURN 500' is specified on the PANEL tag which defines the WRKMBR panel.
   - The user presses the F12 (Cancel) key. The program variable passed as the function requested parameter to the QUIDSPP API is set to -8. This is the value defined for the CANCEL dialog command.
   - The user presses the F3 (Exit) key. The program variable passed as the function requested parameter to the QUIDSPP API is set to -4. This is the value defined for the CANCEL dialog command.

5. If the program is not called by the UIM to process option 5 from the example list panel shown in Figure 104 on page 287, the program should do the following.
   a. Call the Close Application (QUICLOA) API to close the UIM application. This frees up the system resources being used by the UIM application.

Data Entry Panel

A data entry panel is a form of data presentation panel where the user is allowed to enter new values for one or more of the dialog variables displayed. The USAGE attribute of the data item (DATAI) and data item extender (DATAIX) tags defines whether or not the user is allowed to enter a new value for the dialog variable displayed by the tag.

When USAGE=INOUT is specified, the user is allowed to enter a new value for the dialog variable. For data areas using a vertical layout, the leader dots following the descriptive text for the item end with a period instead of a colon. A colon is used for data items which do not allow input.

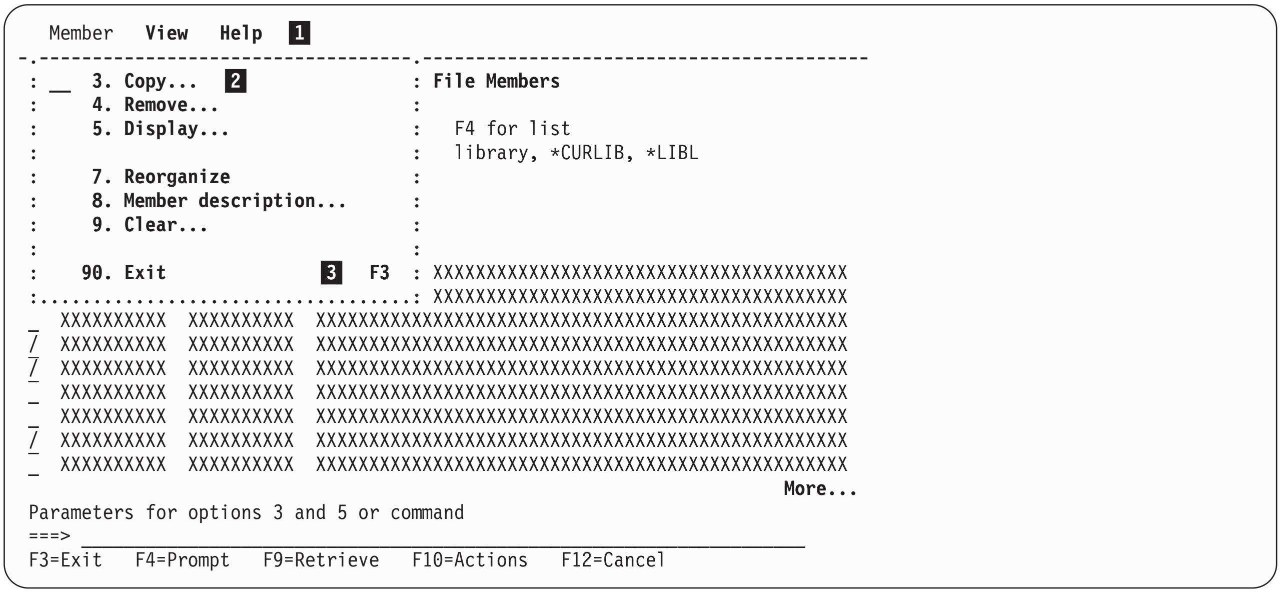
When the UIM returns control from the Display Panel (QUIDSPP) API and the function requested parameter contains the value specified for the RETURN dialog command on the ENTER attribute of the display panel (PANEL) tag, the application program uses the Get Dialog Variable (QUIGETV) API to retrieve the values for the dialog variables changed by the user.

Creating a Panel with a Menu Bar

A menu bar can be defined for any full-screen panel. The menu bar acts as an extension of the panel and contains definitions of choices for the menu bar. Each choice in the menu bar has a pull-down menu defined. When the user selects the menu bar choice, the UIM displays the pull-down menu in a window below the menu bar. Each choice within the pull-down menu represents an action which can be performed. In this example, a Work with File Members panel is defined using a menu bar. This panel is functionally similar to the panel shown in “Creating a List Panel” on page 287. Although the panel appears slightly different from the example list panel, this example panel using a menu bar performs the same function.

The panel shown in Figure 111 on page 321 shows an example of an action list panel which also has a menu bar. This example explains how the menu bar is defined and how it interacts with the rest of the panel definition. This example does not describe how to define the action list portion of the panel. For more information on creating an action list, see “Creating a List Panel” on page 287. If you need more information on creating help for a menu bar area, see “Help in a Menu Bar Area” on page 394.

The example panel is shown with a pull-down menu displayed. The pull-down is active because the user selected entries in the list and pressed the Enter key.



```
   Member   View   Help  1
-.----------------------------------.-----------------------------------------
 : __  3. Copy...  2                : File Members
 :     4. Remove...                 :
 :     5. Display...                :   F4 for list
 :                                  :   library, *CURLIB, *LIBL
 :     7. Reorganize                :
 :     8. Member description...     :
 :     9. Clear...                  :
 :                                  :
 :    90. Exit            3  F3     : XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
 :..................................: XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
 _  XXXXXXXXXX  XXXXXXXXXX  XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
 /  XXXXXXXXXX  XXXXXXXXXX  XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
 /  XXXXXXXXXX  XXXXXXXXXX  XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
 _  XXXXXXXXXX  XXXXXXXXXX  XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
 _  XXXXXXXXXX  XXXXXXXXXX  XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
 /  XXXXXXXXXX  XXXXXXXXXX  XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
 _  XXXXXXXXXX  XXXXXXXXXX  XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
                                                                         More...
 Parameters for options 3 and 5 or command
 ===> ____________________________________________________________________
 F3=Exit   F4=Prompt   F9=Retrieve   F10=Actions   F12=Cancel
```

Figure 111. Example Panel with a Menu Bar

The reference numbers in the example panel do not appear on the display. They are shown for illustration purposes and also appear in the UIM tag source shown in "Source for Example Panel with a Menu Bar" on page 323. These reference numbers show which portions of the source define text and information which appears on the panel.

1 Each choice in the menu bar is defined using a menu bar choice (MBARC) tag. The choice shown on the panel is defined as text following the period of the MBARC tag.

All choices for a menu bar are defined by placing the MBARC tags between the menu bar (MBAR) and EMBAR tags. The name specified on the NAME attribute of the MBAR tag is also specified on the MBAR attribute of the PANEL tag.

2 Each option in the pull-down menu is defined using the pull-down field choice (PDFLDC) tag. The option number is defined using the OPTION attribute and the text following the option number is defined as text following the period of the PDFLDC tag.

All choices for a pull-down menu are defined by placing the PDFLDC tags between the pull-down field (PDFLD) and EPDFLD tags. The PDFLDC and EPDFLDC tags are placed between a MBARC and EMBARC tag.

3 An accelerator key description shown along the right side of the pull-down menu is defined as text following the period of the PDACCEL tag. An accelerator is a key which performs the same function as an option in a pull-down menu. The PDACCEL tag is placed immediately after the PDFLDC tag for the option to which the accelerator applies.

From this panel, the application user can perform operations against entries in the list using either of the following methods:

- When the user selects entries in the list and presses the Enter key, the UIM displays the pull-down menu for the member choice in the menu bar. This is done because SELECT=PULLDOWN is specified on the PANEL tag shown at **A**.

  When the user selects option 3, 4, 5, 7, 8 or 9 from the pull-down menu, the UIM performs the action defined for that option once for each list entry selected by the user. The UIM does this because ACTFOR=LIST is specified on the PDFLDC tag at **B**. The UIM makes these options unavailable when the users displays the pull-down menu without selecting any entries in the list.

When the user selects option 90 from the pull-down menu, the UIM returns control to the application program that displayed the panel. The UIM does this because ACTFOR=PANEL is specified on the PDFLDC tag at **C**. Because EXIT is specified for the ACTION attribute on the PDFLDC tag, the UIM returns control to the calling program with an indication that the user requested Exit.

- When the user types option numbers next to the desired list entries and presses the Enter key, the UIM performs the action for each option. The UIM does this because ACTOR=UIM is specified on the list area (LIST) tag at **D** and a list action (LISTACT) tag at **E** specifies the action for the UIM to perform for each option number.

  Note that in this case, no text is shown on the panel describing the option numbers for the list. This is because no text is specified following the period of the LISTACT tags. The primary interface defined for this panel is to select entries in the list and then choose an action from a pull-down menu. The list option numbers are available as a faster alternative for more experienced users. Although the list option numbers do not appear on the panel, they are described in the help shown when the Help key is pressed with the cursor positioned in the input column of the list.

  For this example, the actions for the option numbers in the pull-down menu are the same as the actions for the option numbers that can be entered in the list. This design is recommended for a consistent user interface, but the UIM does not prevent the application developer from defining an action in a pull-down menu that is different from the action for a list option with the same option number.

Required Tags for a Panel with a Menu Bar

Figure 112 on page 323 shows the required tags for creating a UIM panel with a menu bar. For an example of the required and optional tags, see “Source for Example Panel with a Menu Bar” on page 323.

Note: The tags in Figure 112 on page 323 require attributes. These attributes are not shown. Without these attributes, the example in Figure 112 on page 323 will not compile. For a description of the required attributes for these required tags, see Appendix A, “UIM Panel Group Definition Language.”

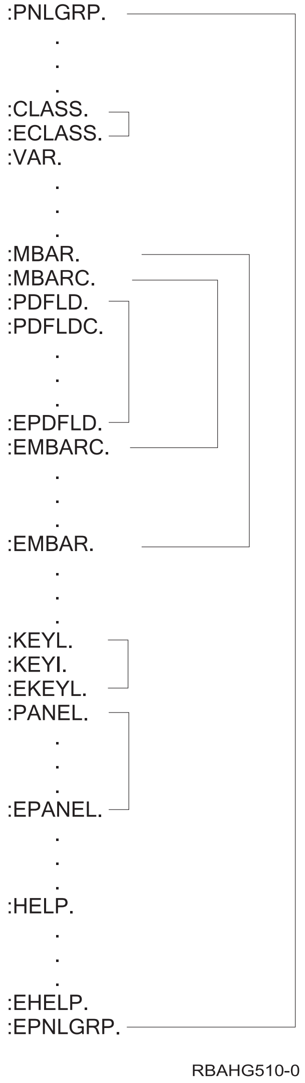


Figure 112. Required UIM tags for a panel with a menu bar

Source for Example Panel with a Menu Bar

This is a partial listing of member T0011PN2 in source file QATTUIM in library QUSRTOOL.

```
.* ---------------------------------------------------------------
.*
.* Beginning of panel group source.
.*
.* ---------------------------------------------------------------
:PNLGRP   DFTMSGF=t0011msgf2
          SUBMSGF=t0011msgf2.
.*
.* The import tag specifies that all help is to be found
.* in panel group T0011HL2 found by searching the library list.
:IMPORT   NAME='*'
          PNLGRP=t0011hl2.
.*
.* ---------------------------------------------------------------
.* Define all variable classes
.* ------------------------------------------------------------------
.* -----------------
.* Option
.* Note: Need WIDTH=1 to preserve column alignment on confirmation panel.
:CLASS    NAME=optcl
          BASETYPE='ACTION'
          WIDTH=1.
:ECLASS.
.* -----------------
.* Object name
:CLASS    NAME=namecl
          BASETYPE='OBJNAME 10'.
:ECLASS.
.* -----------------
.* Library name
:CLASS    NAME=libcl
          BASETYPE='OBJNAME 10'.
:TL.
:TI       VALUE='"*LIBL"'.*LIBL
:TI       VALUE='"*CURLIB"'.*CURLIB
:ETL.
:ECLASS.
.* -----------------
.* File attribute
:CLASS    NAME=attrcl
          BASETYPE='CHAR 10'
          CASE=UPPER.
:ECLASS.
.* -----------------
.* Descriptive text
:CLASS    NAME=textcl
          BASETYPE='IGC 50'
          SUBST=QUOTED.
:ECLASS.
.* -----------------
.* Source type
:CLASS    NAME=srctypcl
          BASETYPE='CHAR 10'.
:ECLASS.
   ⋮
  Additional CLASS tags in member T0011PN2 are not shown here
   ⋮
.* -----------------
.* Command line parameters
:CLASS    NAME=parmcl
          BASETYPE='CHAR 255'.
:ECLASS.
.* -----------------
.* Exit program specification for CALL dialog command
:CLASS    NAME=exitcl
          BASETYPE='CHAR 20'.
:ECLASS.
.* -----------------
.* View number
:CLASS    NAME=vwnumcl
          BASETYPE='BIN 15'.
:ECLASS.
.* -----------------
.* Classes for pad variables in variable record definitions.
:CLASS    NAME=pad1cl
          BASETYPE='CHAR 1'.
```

```
:ECLASS.
:CLASS    NAME=pad2cl
          BASETYPE='CHAR 2'.
:ECLASS.
:CLASS    NAME=pad10cl
          BASETYPE='CHAR 10'.
:ECLASS.
:CLASS    NAME=pad13cl
          BASETYPE='CHAR 13'.
:ECLASS.
:CLASS    NAME=pad48cl
          BASETYPE='CHAR 48'.
:ECLASS.
:CLASS    NAME=pad50cl
          BASETYPE='CHAR 50'.
:ECLASS.
.*
.* -----------------------------------------------------------------
.* Define all dialog variables
.* -----------------------------------------------------------------
.*
.* -------------------------------------
.* Variables for file and library
.* -------------------------------------
.* -----------------
.* File name
:VAR      NAME=file
          CLASS=namecl.
.* -----------------
.* Library name
:VAR      NAME=lib
          CLASS=libcl.
.* -----------------
.* attributes
:VAR      NAME=fattr
          CLASS=attrcl.
.* -----------------
.*
.* -------------------------------------
.* Variables for list of members
.* -------------------------------------
.* -----------------
.* option for list of members
:VAR      NAME=mopt
          CLASS=optcl.
.* -----------------
.* Object name
:VAR      NAME=mbr
          CLASS=namecl.
.* -----------------
.* member type
:VAR      NAME=mtype
          CLASS=attrcl.
.* -----------------
.* Descriptive text
:VAR      NAME=mtext
          CLASS=textcl.
   ⋮
  Additional VAR tags in member T0011PN2 are not shown here
   ⋮
.*
.* -------------------------------------
.* Variable for command line parameters
.* -------------------------------------
.* -----------------
.* Command line parameters
```

```
:VAR      NAME=parms
          CLASS=parmcl.
.*
.* -------------------------------------
.* Variables for specifying CALL/exit programs
.* -------------------------------------
.* -----------------
.* Program to call for all UIM exits
:VAR      NAME=exitpgm
          CLASS=exitcl.
.*
.* -------------------------------------
.* Variables for controlling list views
.* -------------------------------------
.* -----------------
.* View number for list of members
:VAR      NAME=mbrview
          CLASS=vwnumcl.
   ⋮
  Additional VAR tags in member T0011PN2 are not shown here
   ⋮
.*
.* -------------------------------------
.* Variables for padding in variable record definitions.
.* Padding is needed in variable records so the layout
.* of the record matches a list format returned from
.* an API.  The pad variables are used as placeholders for
.* variables not used in the API format or for reserved space
.* in the API format.
.* -------------------------------------
:VAR      NAME=pad1
          CLASS=pad1cl.
:VAR      NAME=pad2
          CLASS=pad2cl.
:VAR      NAME=pad10
          CLASS=pad10cl.
:VAR      NAME=pad13
          CLASS=pad13cl.
:VAR      NAME=pad48
          CLASS=pad48cl.
:VAR      NAME=pad50
          CLASS=pad50cl.
.*
.* -----------------------------------------------------------------
.* Define a variable record for file, library and file attribute
.* -----------------------------------------------------------------
:VARRCD   NAME=filelib
          VARS='file lib fattr'
        NOGET='fattr'
          .
.*
.* -----------------------------------------------------------------
.* Define a variable record for exit program
.* -----------------------------------------------------------------
:VARRCD   NAME=exitprog
          VARS='exitpgm'
          .
.*
.* -----------------------------------------------------------------
.* Define a variable record for list of members.
.* The layout of this record is designed to match the
.* List Database File Members API (QUSLMBR) format name MBRL0200.
.* -----------------------------------------------------------------
:VARRCD   NAME=mbrl0200
          VARS='mbr mtype pad13 pad13 mtext'
          NOPUT='pad13'
          NOGET='mtype pad13 mtext'
          .
  ⋮
  Additional VARRCD tags in member T0011PN2 are not shown here
  ⋮
.*
.*
.* -----------------------------------------------------------------
.* Define a list of members
.* -----------------------------------------------------------------
:LISTDEF  NAME=mbrlist
          VARS='mopt mbr mtype mtext'
          MSGID=USR0101.
  ⋮
  Additional LISTDEF tags in member T0011PN2 are not shown here
  ⋮
.*
.* -----------------------------------------------------------------
.* Define all conditions
.* -----------------------------------------------------------------
.* -----------------
.* Condition for physical files
:COND     NAME=pf
          EXPR='fattr="PF        "'.
.* -----------------------------------------------------------------
.* Conditions for views of members list
:COND     NAME=mbrview1
          EXPR='mbrview=0'.
:COND     NAME=mbrview2
          EXPR='mbrview=1'.
  ⋮
  Additional COND tags in member T0011PN2 are not shown here
  ⋮
  ⋮
  All TT tags in member T0011PN2 are not shown here
  ⋮
.*
.* -----------------------------------------------------------------
.* Define menu bar for work with members panel
.* -----------------------------------------------------------------
:MBAR     NAME=mbarmbr   1
          HELP='mbarmbr/'
          .
:MBARC    HELP='mbarmbr/member'
         .
Member   1
:PDFLD.
:PDFLDC   OPTION=3
          ACTFOR=LIST  B
          HELP='mbarmbr/cpyf'
          ACTION='CMD CPYF ?*FROMFILE(&lib./&file.)'
          ACTION=' ?*FROMMBR(&mbr.) &parms.'
          .
Copy...   2
.*
:PDFLDC   OPTION=4
          ACTFOR=LIST
          HELP='mbarmbr/rmvm'
          ACTION='CMD RMVM FILE(&lib./&file.) MBR(&mbr.)'
.*        CONFIRM=confrmvmmb
          USREXIT='CALL exitpgm'
          .
Remove...
.*
:PDFLDC   OPTION=5
```

```
          COND=pf
          ACTFOR=LIST
          HELP='mbarmbr/dsppfm'
          ACTION='CMD DSPPFM FILE(&lib./&file.) MBR(&mbr.) &parms.'
          .
Display...
.*
:PDFLDC   OPTION=7
          COND=pf
          ACTFOR=LIST
          HELP='mbarmbr/rgzm'
          ACTION='CMD RGZPFM FILE(&lib./&file.) MBR(&mbr.)'
          .
Reorganize
.*
:PDFLDC   OPTION=8
          ACTFOR=LIST
          HELP='mbarmbr/dspfd'
          ACTION='CALL exitpgm'
          .
Member description...
.*
:PDFLDC   OPTION=9
          COND=pf
          ACTFOR=LIST
          HELP='mbarmbr/clrm'
.*        CONFIRM=confclrmmb
          ACTION='CMD CLRPFM FILE(&lib./&file.) MBR(&mbr.)'
          .
Clear...
.*
:PDFLDC   OPTION=90
          ACTFOR=PANEL  C
          HELP='mbarmbr/exit'
          ACTION='EXIT SET'
          VARUPD=NO
          .
Exit
:PDACCEL.F3   3
:EPDFLD.
:EMBARC.
.*
:MBARC    HELP='mbarmbr/view'

         .
View
:PDFLD.
:PDFLDC   OPTION=1
          ACTFOR=PANEL
          HELP='mbarmbr/view2'
          AVAIL=mbrview1
          ACTION=CHGVIEW
          .
Names only
:PDACCEL.F11
:PDFLDC   OPTION=2
          ACTFOR=PANEL
          HELP='mbarmbr/view1'
          AVAIL=mbrview2
          ACTION=CHGVIEW
          .
Names and descriptions
:PDACCEL.F11
:EPDFLD.
:EMBARC.
.*
:MBARC    HELP='mbarmbr/help'
          .
```

```
Help
:PDFLD.
:PDFLDC   OPTION=1
          ACTFOR=PANEL
          HELP='mbarmbr/helphelp'
          ACTION=helphelp
          .
Help for help...
:PDFLDC   OPTION=2
          ACTFOR=PANEL
          HELP='mbarmbr/exthelp'
          ACTION=exthelp
          .
Extended help...
:PDFLDC   OPTION=3
          ACTFOR=PANEL
          HELP='mbarmbr/keyshelp'
          ACTION=keyshelp
          .
Keys help...
:PDFLDC   OPTION=4
          ACTFOR=PANEL
          HELP='mbarmbr/about'
          ACTION='DSPHELP about'
          .
About...
:EPDFLD.
:EMBARC.
:EMBAR.
   ⋮
  The first KEYL tags in member T0011PN2 are not shown here
   ⋮
.*
.*
.* ------------------------------------------------------------------
.* Define keys for work with members panel with a menu bar
.* The keys are the same but some do not have descriptions
.* ------------------------------------------------------------------
:KEYL     NAME=mbrkeysmb
         HELP=keyl.
:KEYI     KEY=F1
          HELP=helpf1
          ACTION=HELP.
:KEYI     KEY=F3
          HELP=exit
          ACTION='EXIT SET'
          VARUPD=NO.
F3=Exit
:KEYI     KEY=F4
          HELP=prompt
          ACTION=PROMPT
          PRIORITY=30.
F4=Prompt
:KEYI     KEY=F9
          HELP=retrieve
          ACTION=RETRIEVE
          PRIORITY=35.
F9=Retrieve
:KEYI     KEY=F10
          HELP=actions
          ACTION=ACTIONS
          PRIORITY=40.
F10=Actions
:KEYI     KEY=F11
          HELP=mbrviewname
          ACTION=CHGVIEW
          COND=mbrview1.
:KEYI     KEY=F11
          HELP=mbrviewdesc
          ACTION=CHGVIEW
          COND=mbrview2.
:KEYI     KEY=F12
          HELP=cancel
          ACTION='CANCEL SET'
          VARUPD=NO.
F12=Cancel
:KEYI     KEY=F24
          HELP=morekeys
          ACTION=MOREKEYS.
F24=More keys
:KEYI     KEY=ENTER
          HELP=enter
          ACTION=ENTER.
:KEYI     KEY=HELP
          HELP=help
          ACTION=HELP.
:KEYI     KEY=PAGEDOWN
          HELP=pagedown
          ACTION=PAGEDOWN.
:KEYI     KEY=PAGEUP
          HELP=pageup
          ACTION=PAGEUP.
:KEYI     KEY=PRINT
          HELP=print
          ACTION=PRINT.
:EKEYL.
   ⋮
  Additional KEYL tags in member T0011PN2 are not shown here
   ⋮
   ⋮
  The first PANEL tags in member T0011PN2 are not shown here
   ⋮

.*
.*
.* ------------------------------------------------------------------
.* Define Work with Members panel with a menu bar to process options
.* ------------------------------------------------------------------
:PANEL    NAME=wrkmbrmbar
          HELP='wrkmbr/'
          MBAR=mbarmbr   1
          KEYL=mbrkeysmb
          ENTER='RETURN 500'
          SELECT=PULLDOWN  A
          TOPSEP=SPACE.
Work with File Members
.*
.* ------------------------------------
.* Define a data presentation area to display the
.* library/file name whose members are listed.
.* ------------------------------------
:DATA     DEPTH=3
          SCROLL=NO
          LAYOUT=1
          BOTSEP=SPACE
          COMPACT
          .
.* ------------------------------------
.* Divide the layout width into two columns.
.* The first column is for the prompt text with leader dots.
.* The second column is for the variable values.
:DATACOL  WIDTH=22.
:DATACOL  WIDTH=12.
```

```
:DATACOL  WIDTH='*'.
.* -------------------------------------
.* Display qualified file name
:DATAGRP  GRPSEP=QINDENT
          HELP='wrkmbr/filelib'
          COMPACT
          .
:DATAI    VAR=file
          USAGE=INOUT
          CSRLOC=NO
          PROMPT='CALL exitpgm'
          .
File
:DATAC.F4 for list
:DATAI    VAR=lib
          USAGE=INOUT
          CSRLOC=NO
          .
Library
:DATAC.library, *CURLIB, *LIBL
:EDATAGRP.
.*
:EDATA.
.*
.* -------------------------------------
.* Define the list area
.* -------------------------------------
:LIST     DEPTH='*'
          LISTDEF=mbrlist
          ACTOR=UIM  D
          MAXHEAD=4
          MAXACTL=3
          VIEW=mbrview
          PARMS=parms
          BOTSEP=SPACE.
:TOPINST.Select members using /, press Enter.
.*
.* -------------------------------------
.* Specify the action to be taken for each option
.* -------------------------------------
:LISTACT  OPTION=3  E
          NOCMD=PROMPT
          HELP='wrkmbr/cpyf'
          ENTER='CMD CPYF ?*FROMFILE(&lib./&file.)'
          ENTER=' ?*FROMMBR(&mbr.) &parms.'
          PROMPT='CMD ?CPYF ?*FROMFILE(&lib./&file.)'
          PROMPT=' ?*FROMMBR(&mbr.) &parms.'.
.*
:LISTACT  OPTION=4
          HELP='wrkmbr/rmvm'
.*        CONFIRM=confrmvmmb
          ENTER='CMD RMVM FILE(&lib./&file.) MBR(&mbr.)'
          PROMPT='CMD ?RMVM ?*FILE(&lib./&file.) ?*MBR(&mbr.)'
          USREXIT='CALL exitpgm'.
.*
:LISTACT  OPTION=5
          COND=pf
          HELP='wrkmbr/dsppfm'
          ENTER='CMD DSPPFM FILE(&lib./&file.) MBR(&mbr.) &parms.'
          PROMPT='CMD DSPPFM ?*FILE(&lib./&file.) ?*MBR(&mb.r) &parms.'.
.*
:LISTACT  OPTION=7
          COND=pf
          HELP='wrkmbr/rgzm'
          ENTER='CMD RGZPFM FILE(&lib./&file.) MBR(&mbr.)'
          PROMPT='CMD ?RGZPFM ?*FILE(&lib./&file.) ?*MBR(&mbr.)'.
.*
```

```
:LISTACT  OPTION=8
          HELP='wrkmbr/dspfd'
          ENTER='CALL exitpgm'
          PROMPT='CALL exitpgm'.
.*
:LISTACT  OPTION=9
          COND=pf
          HELP='wrkmbr/clrm'
.*        CONFIRM=confclrmmb
          ENTER='CMD CLRPFM FILE(&lib./&file.) MBR(&mbr.)'
          PROMPT='CMD ?CLRPFM ?*FILE(&lib./&file.) ?*MBR(&mbr.)'.
.*
.*
.* ------------------------------------
.* Define the columns and headings to display
.* ------------------------------------
:LISTCOL  VAR=mopt
          USAGE=INOUT
          HELP='wrkmbr/option'
          MAXWIDTH=6.
:LISTCOL  VAR=mbr
          USAGE=OUT
          HELP='wrkmbr/mbr'
          MAXWIDTH=10.
Member
:LISTCOL  VAR=mtype
          USAGE=OUT
          HELP='wrkmbr/type'
          MAXWIDTH=10.
Type
:LISTCOL  VAR=mtext
  USAGE=OUT
          HELP='wrkmbr/text'
          MAXWIDTH='*'.
Text
.*
.* ------------------------------------
.* Define multiple views for F11 to toggle between
.* ------------------------------------
:LISTVIEW COLS='mopt mbr mtype mtext'.
:LISTVIEW COLS='mopt mbr' layout=4.
.*
:ELIST.
.*
.* ------------------------------------
.* Use a command line and allow parameters to be given
.* ------------------------------------
:CMDLINE  SIZE=SHORT.
Parameters for options 3 and 5 or command
.*
:EPANEL.
   ⋮
  Additional PANEL tags in member T0011PN2 are not shown here
   ⋮
.*
.* ------------------------------------------------------------------
.* End of panel group source
.* ------------------------------------------------------------------
:EPNLGRP.
```

Application Programming for a Menu Bar Panel

An example of an application program to display the panel shown in Figure 111 on page 321 can be found in member T0011CP2 in source file QATTSYSC in library QUSRTOOL. This is an ILE C/C++ program which calls the appropriate UIM application programming interfaces (APIs) to display the panel.

The application program should do the processing described in “Application Programming for a List Panel” on page 298.

A general example of an RPG program can be found in member T0011INF in source file QATTINFO in library QUSRTOOL.

Chapter 17. Details of Using User Interface Manager

This chapter explains the structure of the i5/OS user interface manager (UIM) and provides an overview of its objects and functions.

Opening a UIM Application

An application program must first open a UIM application that uses the panel group to access dialog variables, lists, and panel definitions. An application is opened by the Open Display Application (QUIOPNDA) or Open Print Application (QUIOPNPA) APIs and closed by the Close Application (QUICLOA) API. A UIM application is managed much like an open file. It is meaningful only for a particular call for reclaim resources processing, and it is automatically closed as necessary by the reclaim resources function or by cleanup processing at the end of a job.

When an application is opened, UIM assigns each application an application handle. A **handle** is a variable that represents an object, an instance of an application using some function, or a processing session. This handle must be specified as an input parameter to all APIs that operate with the application. The same panel group object can be associated with more than one open application in the same job, but each application contains a completely independent set of dialog variables, active lists, and internal control information that define the state of the UIM application.

Defining Dialog Variables

A **dialog variable** is a UIM element that contains a value. This value can be referred to and updated by programs that use the open application and by the UIM when it performs functions such as displaying panels to the user.

The dialog variables contained in an open application are determined by the panel group object associated with the application. Dialog variables are defined using the variable definition (VAR) language tag in the source for a panel group. Special dialog variables defined by UIM have names that begin with the letter Z and are referred to as **Z-variables**. For more information on the VAR language tag, see “VAR (Variable Definition)” on page 634. For more information on Z-variables, see “Dialog Variables Defined by UIM” on page 635.

Every open application associated with a particular panel group object contains a complete set of all the dialog variables defined in the panel group, including whatever Z-variables are defined in the tag language for that panel group. The set of all dialog variables in an application is called the **variable pool** for the application. UIM is not able to communicate directly with a program’s storage to get the values of its variables. The program can use the variable pool to communicate to UIM what it would like to display on the panel. When a panel is to be displayed, UIM retrieves the values from the variable pool and displays them on the panel. When the user updates the fields on the panel, UIM reads the values from the panel and places them into the variable pool for the user program to retrieve and act upon. Because each open application has its own variable pool, each time a user program wishes to gain access to the dialog variables in an application, it must provide the variable pool APIs with the application handle that was assigned by the QUIOPNDA API.

For more information on variable pools, see “Using Variable Pool Services” on page 338. The dialog variables are used to display panel field values that are not constant text. The dialog variables are also used to tailor the format of the panel using the condition definition (COND) tag. For more information on the COND tag, see “COND (Condition Definition)” on page 492.

The definition of each dialog variable specifies a base data type value that controls both the form of the internal storage of the variable and its editing characteristics on the display. The following base type values are supported:

- Character
- IGC
- Graphic
- Binary (numeric)
- Packed and zoned decimal
- Date
- Time
- Action (list option or selection)
- Name
- Object name
- Pointer

The class definition (CLASS) language tag defines a class of dialog variables to be associated with a base data type, specific validity checking, and display value translation functions. Validity checking is done only for values entered by the user in an input field on a display. The validity checking is not done for values provided by an application program through an API. For more information on the CLASS tag, see "CLASS (Class Definition)" on page 478.

A translation function allows the application program to operate with internal values. The values are automatically mapped by the UIM to and from specified character string values, when displayed or printed. An example of this is the months in a year. The application program reference the values 1 through 12, but the user sees only the names of the months on the display. For more information on the translation function, see "TL (Translation List)" on page 629.

When the application is initially opened, every dialog variable in the application has an initial value. For dialog variables defined by the application programmer, the initial value is determined by the base type as follows:

Table 30. Initial Values of Dialog Variable

Dialog Variable	Initial Value
Numeric	0
Date and Time	0
Character	Single-byte blanks
IGC	Single-byte blanks
Graphic	Double-byte blanks
Pointer	Null
Z-variable	Defined by the definition of the variable (see "Defining Dialog Variables" on page 335)

Any dialog variable that is referenced without a set value is assigned the initial value, whether the reference is made by an application program or by the UIM.

Restrictions on Using Dialog Variables

A dialog variable *should not* be used for the value of more than one display item per panel as multiple input fields, because the results might be undesirable. This means that when used as input fields, no dialog variable should be named on multiple data item (DATAI), data item extender (DATAIX), or list column (LISTCOL) tags that are part of the same panel definition without conditioning. This ensures that the same dialog variable does not appear on the display in more than one place.

Because the variable pool exists until the application is closed, the UIM, and all programs that use the application, can determine the last value assigned to any dialog variable in the pool. Once the application program assigns the value of a dialog variable, it never needs to be assigned again as long as it does not need to be changed. Similarly, once an input value is accepted from the user and stored in a dialog variable, that value continues to be available until the variable is updated by either an application program or the user.

Dialog Variable Error Messages

By defining a variable of basetype char 1 on the ERRVAR attribute of the VAR tag, the user may indicate to UIM that a variable is in error by using the Put Dialog Variable (QUIPUTV) API to set this variable to (X"F1"). When this happens, the field in error is highlighted and the cursor is positioned to the first input field that is in error. The application program reports the cause of the error and requests that the UIM display the messages to the user using the message reference key parameter of the Display Panel (QUIDSPP) API.

All variables in error in the open application are reset when the next dialog command is processed by the UIM. Exceptions to this are the Menu Bar Cursor Action (ACTIONS), Command Line (CMDLINE), Change View (CHGVIEW), HELP, Display More Function Keys (MOREKEYS), Move to Top (MOVETOP), PAGEUP, PAGEDOWN, Print Screen (PRINT), and Retrieve Command String (RETRIEVE) dialog commands. The error status for each dialog variable is reset when control returns from the Display Panel (QUIDSPP) API or when control is passed from the UIM to a program or command identified by a menu item, action list, pull-down choice, or function key item.

When a function key is pressed or a pull-down choice is selected that performs an action where VARUPD=YES was specified, ((see "KEYI (Key List Item)" on page 542), and "PDFLDC (Pull-Down Field Choice)" on page 605), the UIM validates the values of all entered (changed) input fields. If an error is found, the panel is displayed again with the fields in error highlighted and one or more messages is issued. The values of the dialog variable associated with the input fields are not changed until the user enters a correct value.

The application developer can specify that some function keys and pull-down choices operate without updating the variable pool by using VARUPD=NO. If control returns to the program that ran the QUIDSPP API for such a function, all values entered by the user are lost. If the requested function causes the UIM to call a program or system command, all input field values are stored internally by the UIM in such a way that they can be shown when control returns to the UIM and the panel is displayed again. These input values are stored locally for the panel being displayed. Any panel that is presented using the same open application by a program or command with VARUPD=NO specified, displays the values of all dialog variables as they exist in the variable pool. However, if VARUPD=YES is specified, the updated value is used instead of the stored value when control returns and the original panel is displayed again.

Providing Field Values for a Display Panel Using Dialog Variables

The tag language allows you to use dialog variables to provide field values for a display panel and substitution values for a CL command that is called by the UIM for a pull-down choice, menu choice, function key, or action list selection. Whenever a dialog variable is referred to in one of these ways, the internal value stored in the variable pool is converted to an external form determined by the editing rules for the variable base type and any translation list defined for the variable class.

When the external form of a dialog variable contains characters that are incorrect for a display device, the UIM converts the incorrect characters to hexadecimal 1F characters (■). When the external form of a dialog variable contains characters that are incorrect for a printer, the UIM converts the incorrect characters to the replacement characters in effect for the printer file, or to blanks when RPLUNPRT(*NO) is in effect for the printer file.

Incorrect display characters are characters with hexadecimal values equal to hexadecimal FF or less than 40, not including hexadecimal 00, 0E, and 0F. Incorrect printer characters are the same as the incorrect display characters, except that hexadecimal 00 is also incorrect. This conversion does not take place on variable values that have been translated with a translation list.

If the dialog variable that contained the incorrect characters is used as an input field and retrieved with the Get Dialog Variable (QUIGETV) API by the program, the incorrect characters might be changed to hexadecimal 3F. This occurs if the user modifies the field or presses the Help key.

When the UIM displays a panel, it retrieves the current value for all variable fields from the variable pool or from list entries. If the user types a value in an input field, the UIM validates and translates the value according to the CLASS attribute of the associated dialog variable, and then stores the value in the dialog variable or associated list entry. The application program can determine the value entered by the user by using the Get Dialog Variable (QUIGETV) or the Get List Entry (QUIGETLE) API to retrieve the dialog variable value.

Using Variable Pool Services

The Get Dialog Variable (QUIGETV) and Put Dialog Variable (QUIPUTV) APIs are the primary interfaces for application programs to retrieve and update dialog variables. When a dialog variable is updated, the attributes of the new value must match the base type of the dialog variable. When a dialog variable is retrieved, the value is always returned to the application program in the internal form specified by the base type of the dialog variable. On the call to the variable pool services APIs, a variable record is required. A variable record, defined using the VARRCD tag in the UIM panel group source, defines a group of dialog variables that may be updated or accessed together in one call to the APIs.

No data conversion or validity checking is done when variable values are retrieved or updated using the QUIGETV and QUIPUTV APIs. For more information on these APIs, see the APIs topic

Some of the dialog variables defined by UIM, referred to as Z-variables, can be retrieved but not modified. The tag language does not allow a variable with only read access to be used as an input field on a panel definition. The UIM returns an exception to an application program if it attempts to update such a Z-variable using the QUIPUTV API.

Dialog Variables and Special Values

Variables with a BASETYPE of TIME defined with the ZONE option can have special values for the time zone. See the “CLASS (Class Definition)” on page 478 tag for more information. Either a user at a display station or the application can specify the special values. The UIM resolves these special values when the value is placed into the variable pool. Therefore, the variable will contain the resolved value, and not the special value as entered, on the next operation that retrieves the value of this variable.

So a user at a display station may enter the value ″01:30:00 *LCL″ into a data item. When another data item on the panel displays the same variable or the application retrieves the value of the dialog variable, the value will appear as ″01:30:00 CST″. Similarly, if the application updates the dialog variable with the special value, and retrieves it again, the value will have changed. Be aware of this behavior if your application uses the special time zone values.

Character Set and Code Page Considerations

Data that is always stored in a specific character set and code page can be converted to another character set and code page when it is displayed or printed by the UIM. If your application has data that must be converted, do one the following:

- Code CHRID=PNLGRP on the CLASS tag of the variables that need to be converted. See “CLASS (Class Definition)” on page 478. Then, specify the number of the graphic character set and code page of your data on the CHRID parameter of the Create Panel Group (CRTPNLGRP) command that is used to create your panel. For more information on the CRTPNLGRP command, see the **CL** topic in the iSeries Information Center.

- On the CHRID parameter of the Create Panel Group (CRTPNLGRP) command that is used to create your panel, specify *JOBCCSID. For more information on the CRTPNLGRP command, see the **CL** topic in the iSeries Information Center.

Displaying

The UIM compares the character set and code page of the device to the character set and code page of the panel group. If they are different, then outbound and inbound conversion tables are used to convert the appropriate dialog variables. If a conversion table is not available, then the Open Display Application (QUIOPNDA) API sends a diagnostic message and continues.

Display operations for Arabic and Hebrew bidirectional panel groups, which have BIDI=LTR or BIDI=RTL specified on the panel group (PNLGRP) tag, are only allowed when the device is configured to use code page 420 or 424.

See Table 31

Table 31. UIM CCSID/CHRID Conversions for Display

	Panel Group CHRID		
Type of Data	**XXX**	***JOBCCSID**	***DEVD**
Panel group constant text	No conversion.	Convert from panel group primary source file CCSID to device CCSID.	No conversion.
Variable with CHRID=PNLGRP on CLASS tag	Convert from XXX to device CHRID.	Convert from job CCSID to device CCSID.	No conversion.
Variable without CHRID=PNLGRP on CLASS tag	No conversion.	Convert from job CCSID to device CCSID.	No conversion.

Printing

The UIM compares the character set and code page of the printer file to the character set and code page of the panel group. If they are different, an outbound conversion table is used to translate the appropriate dialog variables. If CHRID(*DEVD) is specified for either the panel group or the printer file, no conversion of the dialog variables is done. If a translation table is not available, the Open Display Application (QUIOPNDA) or the Add Print Application (QUIADDPA) API sends a diagnostic message and continues.

When a printer file is printed on a printer device, the character set and code page of the printer file is compared to the character set and code page that is loaded in the printer. If they are different, all the printer file data, including the constant text from the UIM tags, is translated to the character set and code page of the printer.

To minimize the number of times that character set and code page translations takes place, specify the same CHRID value on both your printer file and your panel group.

Print operations for Arabic and Hebrew bidirectional panel groups, which have BIDI=LTR or BIDI=RTL specified on the panel group (PNLGRP) tag, must have code page 420 or 424 specified for the printer file. Also, any call to the QUIADDPA API for a bidirectional panel group must have the same code page specified for the printer file as is used by the display device.

See Table 32 on page 340

Table 32. UIM CCSID/CHRID Conversions for Print

	Panel Group CHRID or Menu CHRID		
Print File CHRID	**XXX**	***JOBCCSID**	***DEVD**
YYY	Panel group constant text: no conversion.	Panel group constant text: convert from panel group primary source file CCSID to YYY.	Panel group constant text: no conversion.
	Variables with CHRID=PNLGRP on CLASS tag: convert from XXX to YYY.	Variables with CHRID=PNLGRP on CLASS tag: convert from job CCSID to YYY.	Variables with CHRID=PNLGRP on CLASS tag: no conversion.
	Variables without CHRID=PNLGRP on CLASS tag: no conversion.	Variables without CHRID=PNLGRP on CLASS tag: convert from job CCSID to YYY.	Variables without CHRID=PNLGRP on CLASS tag: no conversion.
*JOBCCSID	Panel group constant text: no conversion.	Panel group constant text: convert from panel group primary source file CCSID to job CCSID.	Panel group constant text: convert from panel group primary source file CCSID to job CCSID.
	Variables with CHRID=PNLGRP on CLASS tag: convert from XXX to job CCSID.	Variables with CHRID=PNLGRP on CLASS tag: no conversion.	Variables with CHRID=PNLGRP on CLASS tag: no conversion.
	Variables without CHRID=PNLGRP on CLASS tag: no conversion.	Variables without CHRID=PNLGRP on CLASS tag: no conversion.	Variables without CHRID=PNLGRP on CLASS tag: no conversion.
*DEVD	Panel group constant text: no conversion.	Panel group constant text: convert from panel group primary source file CCSID to job CCSID since variables are in job CCSID and device CCSID is unknown.	Panel group constant text: no conversion.
	Variables with CHRID=PNLGRP on CLASS tag: no conversion.	Variables with CHRID=PNLGRP on CLASS tag: no conversion.	Variables with CHRID=PNLGRP on CLASS tag: no conversion.
	Variables without CHRID=PNLGRP on CLASS tag: no conversion.	Variables without CHRID=PNLGRP on CLASS tag: no conversion.	Variables without CHRID=PNLGRP on CLASS tag: no conversion.

Managing a List

A UIM list is a sequential set of list entries. Each entry contains a copy of the value for one or more dialog variables. The entries in a list can be presented to the user in a scrollable area of a display. The UIM provides application programming interfaces (API) that allow an application program to perform the following operations on lists:

- Add a new entry between any two entries in the list.
- Add an entry at the beginning or end of the list.
- Update the values in a list entry.
- Remove a list entry.
- Position the current entry pointer for the list to a specific entry; several position options are available.
- Set and retrieve list attributes that control UIM processing when the list is displayed.
- Delete an active list. (Remove all entries from a list and make the list inactive for the application.)

Defining a List

The definition for each list using the list definition (LISTDEF) language tag is specified in the source for the panel group object. Attributes of the list definition (LISTDEF) tag specify the name of the list and what dialog variables should be associated with the list (that is, what variable values are to be stored in each list entry). For more information on the LISTDEF tag, see “LISTDEF (List Definition)” on page 573.

Each entry in a list contains a copy of the values of all dialog variables associated with the list. The values of the dialog variable are copied into a list entry when the entry is first added to the list using the Add List Entry (QUIADDLE) or Add List Multiple Entries (QUIADDLM) API, and whenever the entry is updated using the Update List Entry (QUIUPDLE) API. The values in a list entry can be copied into the corresponding dialog variables when the entry is retrieved using the Get List Entry (QUIGETLE) or Get List Multiple Entries (QUIGETLM) API.

Initializing a List

All UIM APIs for processing lists require the program to specify the list name on the list object definition (LISTDEF) tag in order to identify the target list for the operation. Each list defined in a panel group is either active or inactive for each open application using the panel group. Each list is initially inactive when an application is opened using the Open Display Application (QUIOPNDA) or Open Print Application (QUIOPNPA) API. A list becomes inactive when it is deleted using the Delete List (QUIDLTL) API. A list may be active in multiple different open applications using the same panel group. A list becomes active when the first entry is added to the list using the Add List Entry (QUIADDLE), or Add List Multiple Entries (QUIADDLM) API, or when the list attributes are set using the Set List Attribute (QUISETLA) API.

The maximum size of a list is approximately 16MB. The maximum number of entries that can be added to a list is based on the size of each entry. You can estimate the size of your list using the following calculations:

1. Add the following values:
   - The size of the entry
   - An overhead value of 19 bytes for each entry
   - A variable overhead of between 5 and 15 bytes for each entry
2. Divide the result value into 16MB (16 777 216 bytes) minus 4096 bytes (the object’s header on i5/OS)

For example, if you have defined a value of 94 bytes for each list entry, you might make the following calculation:

```
(16 777 216 - 4096)/(94 bytes + 19 bytes + 5 bytes) =
16 773 120 / 118 =
142,145 entries
```

Displaying a List

When a panel containing a list area is displayed, entries from the list are used to build a display area that appears to the user as a table. Each list area is associated with one list definition (see “LIST (List Area)” on page 552) and contains an independent selection of columns to display. For more information on the LISTCOL language tag, see “LISTCOL (List Column)” on page 568. A list can be referred to by more than one list area. The list area allows one list to be presented on different panels within the same open application. If more than one list area exists in a panel definition, a different list must be used for each list area.

Use the LISTCOL tag to specify the specific values presented in a list area. A list area can present all the values in the associated list entries, or it can present only a subset of the values available in each list entry. The list area the user sees contains a row for each list entry and a column for each field specified in a LISTCOL tag in the definition of each view of the list area.

You can define a list area to present the values associated with only one list entry per display line or with multiple list entries per display line. When multiple entries are shown on each display line, the display is formatted in multiple layouts with entries presented in order from top to bottom within layout columns, and from left to right between layout columns. See “Example 2: List Area with Three Layout Columns” on page 558.

Updating a List

Values of list entries are used to fill the display for all fields in the list area; see “LISTCOL (List Column)” on page 568. The user can update any input fields on the display, including those in the list area. If the value specified by the user satisfies all validity checks for the associated dialog variable, the corresponding value in the list entry is updated with the value specified by the user.

The UIM APIs provided to retrieve, add, and update list entries all operate by using the dialog variables associated with the list. No support is provided to directly establish or change values of list entries without referencing dialog variables. Every time a list entry is retrieved, added, or updated, the entire set of values for dialog variables is copied to or from the list entry. When the UIM refers to or updates list entries, such as processing user options in an action list, it can also update values of dialog variables that correspond with the list columns.

Note: When working with an action list you should take care during incomplete list processing. To avoid undesirable results, the action dialog variable should be included in the VARRCD for the list entry and updated in the same way as the other variables in the variable record. If you don’t do this, you run the risk of updating each new list entry (during incomplete list processing) with the value of the last option that was entered on the panel. For example, you have a list panel with 12 entries. You enter option “4” on one entry and scroll down. The incomplete list exit is called to add more entries to the list. The option dialog variable has a value of “4”, so each entry that is added may now have a “4” in the option field.

Each entry inserted in a list is assigned an identifier, called a **list entry handle**, that uniquely identifies the entry within the active list until the entry is removed from the list. The identifier is meaningful only for a particular combination of open application, active list, and list entry instances. An identifier has no meaning in any other open UIM application, or even in the same application if the list or the entry is deleted and then created again. Undesirable results are possible if an identifier is used outside of this definition.

Incomplete List Processing

When adding entries to an action list during incomplete list processing, take care to ensure the option field contains the appropriate value (usually blanks) at the time the QUIADDLE or QUIADDLM API is called. If you are using a VARRCD, the option variable should be part of the VARRCD.

Removing and Inserting an Entry from a List

After an entry is removed from a list, it can no longer be accessed by the application program, and it does not appear even as a blank line in a list area on a panel. When a new entry is added to a list, the UIM can assign the same list entry handle for that entry that was used for an entry that was previously removed from the list. An identifier value has no relationship to the logical position of the entry within the sequence defined for the entries in the list.

The UIM maintains a current entry pointer for each active list to use as a reference point for list-entry operations. The list entry manipulation APIs support a list entry handle parameter. This parameter returns to the application program the identifier where the current entry pointer for the list was positioned at the end of the operation. The UIM sets the current entry point to:

- The entry just added by the Add List Entry (QUIADDLE) or Add list Multiple Entries (QUIADDLM) API

- The list entry requested by the Get List Entry (QUIGETLE) or Get List Multiple Entries (QUIGETLM) API, or left unchanged if the requested entry is not found or not available
- The entry that preceded the entry removed by the Remove List Entry (QUIRMVLE) API

The current entry point is unchanged by the Update List Entry (QUIUPDLE) API.

The Get List Entry (QUIGETLE), Get List Multiple Entries (QUIGETLM), and Remove List Entry (QUIRMVLE) APIs may set the current entry pointer position to either the top or bottom of the list. The top is the position that is always logically before the first entry in the list, and the bottom is the position that is always logically after the last entry in the list. Each of these positions has a special identifier value for the list entry handle, but because the top and bottom are not ″real″ entries, they cannot be updated or removed from the list. The application program receives an error if it attempts to insert an entry before the top of the list or after the bottom of the list.

Controlling List Entries on a List Display

The application program can control what list entry appears as the topmost entry in a list area on a panel by setting the display position attribute to a valid list entry handle before the Display Panel (QUIDSPP) API is run. For example, this function can be used to position to a particular list entry specified by the user. When the display position attribute is not set by the Set List Attributes (QUISETLA) API between the time the list is initialized and the first time the list is displayed, the UIM presents the first entry at the top of the list area in the panel. For more information on the Display Panel (QUIDSPP) API, see the APIs topic.

The display position attribute is updated whenever the user moves to a new page of the list. Any time the application program removes the entry identified by the display position attribute, the display position attribute is automatically updated to refer to the entry preceding the one removed. If there is no entry before the one removed, the display position attribute is set to the top of the list, and the next display of the panel presents the first entry at the top of the list area in the panel.

When an entry is added or updated in a list, the error state of every dialog variable associated with the list is saved with the list entry. The processing described in “Defining Dialog Variables” on page 335 for dialog variables in error state is also used for list entry values that are in error state. This processing concerns how validation errors are detected by the UIM and how the VARUPD attribute of the key item (KEYI) and pull-down field choice (PDFLDC) tags affects updating list values.

Improving Interactive Response Time for a List Display

To improve interactive response for a list display, the UIM list processing allows an application program (the incomplete list exit program) to build only part of a list before it is shown to the user. Because the UIM handles scrolling lists as part of displaying a panel, without returning to the program that ran the Display Panel (QUIDSPP) API, the application program must tell the UIM whether the entries in the list represent only the top, middle, or bottom part of the complete list. If the program calls the Set List Attribute (QUISETLA) API to set the list contents attribute to a value other than ALL, and if the user attempts to page to list entries that are available to the application but not yet added to the list, the UIM calls an application program to add more list entries. The application program is also called by the UIM as part of Get List Entry (QUIGETLE) or Get List Multiple Entries (QUIGETLM) API processing if the specified entry is not found but might exist in the part of the list that has not yet been added.

When a panel is displayed that contains a list area and the associated list is either not active or contains no entries, the user sees a blank list area and a message indicates that there are no entries in the list. However, if the list attributes are set to indicate that the list is not complete, an application program is called to add entries to the list. The panel is not displayed to the user until either enough entries are added to the list to present a full display page or the application program marks the list as complete at either the top or the bottom. The application program is called not only when the list is empty but any time there are not enough entries in an incomplete list to fill a list area on the panel.

Using Action Lists and Selection Lists

The UIM supports the following types of lists:

- Action list
- Selection list
- Output only list
- Input/output list

Using Action Lists

An **action list** is a list area where the user can type option numbers to perform actions against one or more entries in the list. The definition of the UIM list displayed as an action list must include a variable defined with BASETYPE=ACTION specified on the CLASS tag. This variable is referred to as the **action variable** of the list.

To define an action list area, the application developer must specify ACTOR=UIM or ACTOR=CALLER on the LIST tag. In addition, one LISTACT tag must be defined for each option number that can be entered by the user.

For an ACTOR=UIM action list, each LISTACT tag specifies the action the UIM performs when the user types the option number and presses the Enter key. The application developer can also define an action the UIM performs when the user presses a key assigned to the PROMPT dialog command. Two types of actions can be performed: run a CL command, or call a program. If the action to be performed is to run a CL command, an action-list exit program might be needed to update the list after the CL command has performed the function for the option. No interface between UIM and a CL command allows the program called by the CL command to obtain the application handle. Therefore, if the program called by the CL command performs a change or delete operation, an action-list exit program is required to update the list.

For an ACTOR=CALLER action list, there is no specification of an action. When the user types the option number and presses the Enter key or a function key assigned to the PROMPT dialog command, the UIM returns control to the application program. The function requested parameter contains a value which indicates that the application program should perform the actions associated with the options selected by the user.

Having the UIM as the actor for the action list is the preferred method because of the following benefits:

- The UIM performs all actions entered by the user.
- The UIM automatically displays a confirmation panel for destructive actions, such as option 4 to delete an object. No application program code needs to be written to provide confirmation support.
- The UIM redisplays the action list panel in cases where an action does not complete successfully. When the UIM receives an escape message, the UIM redisplays the list with the cursor located on the option number. The option number is shown in error. When the Exit (F3) or Cancel (F12) function is requested from a panel displayed as a result of the list option, the UIM redisplays the panel as appropriate. For more information about the UIM processing a request for Exit and Cancel when displaying an action list, see “Folding Up a List Panel” on page 355.

When the application program is the actor for an action list, the above processing must be done by the application program.

When a panel contains an action list and a menu bar, pull-down choices can be defined within the menu bar which operate against each selected entry in the action list. When at least one pull-down choice is defined which operates against selected list entries, the UIM allows the user to select entries by typing a valid selection character. The user selects one or more entries in the list, selects a pull-down menu from the menu bar, and then selects a choice from the pull-down menu. The UIM performs the selected pull-down choice for each entry selected by the user.

Using Selection Lists

A **selection list** is a list area in which the user can type a selection character for one or more entries in the list. A selection list is defined to allow the user to select a single entry or multiple entries in the list.

A selection list can be used for the following purposes:

- Allow the user to select an object or value from a previously displayed entry field. This type of selection list is displayed when the user positions the cursor to an entry field and requests the PROMPT dialog command. The F4 key is recommended to be assigned to the PROMPT dialog command.
- Allow the user to select entries and then perform an action defined in a pull-down menu from the menu bar.

The definition of the UIM list displayed as a selection list must include a variable defined with `BASETYPE`=ACTION specified on the CLASS tag. This variable is referred to as the **action variable** of the list. When the user enters a valid selection character for a list entry, the UIM sets the action variable for that entry to 1000. A value of 1000 for an action variable always indicates that the list entry is selected.

When a panel contains a selection list and a menu bar, pull-down choices can be defined within the menu bar which operate against each selected entry in the list. The user selects one or more entries in the list, selects a pull-down menu from the menu bar, and then selects a choice from the pull-down menu. The UIM performs the selected pull-down choice for each entry selected by the user.

Using Selection Characters

The user selects entries in an action list panel that contains a menu bar or a selection list panel by typing a slash (/), or country-designated selection character, in the option or selection field for the desired list entries. The country-designated selection character is determined for each panel group when the panel group is created. The UIM retrieves the first level text of message CPX5A0C in the CCSID of the panel group source file. The first two characters of this message are stored in the panel group object. These characters are the allowed uppercase and lowercase country-designated selection characters for selection lists and multiple-choice selection fields defined in the panel group.

If *JOBCCSID is specified for the CHRID attribute when the panel group is created, the country-designated selection characters are converted at run time from the panel group source file CCSID to the job CCSID. This enables the comparison to be done in an equivalent CCSID.

Managing Panel Functions

The UIM supports the following panel management functions:

- Enabling panel conversion to a graphical user interface (GUI)
- Scrolling
- Limiting user capabilities
- Defining contextual help
- Command line restrictions
- Defining function keys
- Formatting panels
- Folding up panels when exit is requested
- Folding up list panels
- UIM as a request processor when displaying a panel

All operations performed by the UIM are done in a single process called a routing step. It is the application developer’s responsibility to consider the effects of things such as recursion, locks, and static storage, for each of the following functions:

- Function key action
- Menu or action list option
- Pull-down choice
- Exit program
- Command entered by the user on a command line

The UIM does not guarantee that the call-sensitive effects of programs and CL commands, such as overrides and the Set Attention Program (SETATNPGM) command, are preserved when commands are run from a panel in any of the ways listed above. The effect of any command that scopes its function to the program call that used the command might be lost by the time the next command or program is called. This is because the UIM call to the function that was scoped was destroyed in the interim.

Enabling Conversion to a GUI

The UIM supports encoding information in the panel which allows a client program to convert your panels to a graphical user interface (GUI).

You can use the ENBGUI attribute on the PANEL or PNLGRP tag to specify whether you want the encoded information included in your panels. More information on how to set this attribute can be found in “PANEL (Display Panel)” on page 595 and “PNLGRP (Panel Group)” on page 609.

Scrolling Support

The UIM supports page up and page down scrolling of all panel information, data presentation, list, and menu areas. Multiple scrollable areas can be defined on a single panel; the UIM imposes no order on them. However, using scrollable menus is discouraged because of usability.

No left-to-right scrolling of text is provided, but the Change View (CHGVIEW) dialog command for lists can be used to show more fields.

Defining Scrollable Areas

To use the UIM scroll function, the panel must have at least one scrollable area and have function keys defined to perform the PAGEUP and PAGEDOWN dialog commands. For more information on the PAGEUP and PAGEDOWN dialog commands, see Appendix B, “UIM Dialog Commands,” on page 641.

Panels with scrollable areas must have function keys assigned to the PAGEUP and PAGEDOWN dialog commands because the UIM does not automatically enable any function keys. Normally these are the Page Up and Page Down keys,[2] but if you use a function key, F7 and F8 are recommended. The UIM does not prevent assigning different function keys to the PAGEUP and PAGEDOWN dialog commands, but it does not allow assigning the engraved page keys to anything other than their implied functions which is page up and page down.

Scrolling status information is also managed by the UIM and is displayed at the bottom of each scrollable area.

The message line is under UIM control. A plus sign (+) on the message line indicates that more messages can be viewed by pressing the Page Down key. Because you can scroll a message area, the function keys for scrolling should always be defined.

[2] On a 5250 keyboard, these are the Rolldown and Rollup keys respectively.

Defining Function Key Scrolling

The UIM handles all scroll- and page-key functions for UIM-defined panels; these functions cannot be overridden by the application. However, you can define other function keys that work much like scroll keys.

When a function key assigned to the PAGEUP or PAGEDOWN dialog command is pressed, the UIM scrolls the appropriate amount for the panel type. This is always a ″page,″ but the UIM tries to avoid having data item groups, menu item groups, and selection fields split across page boundaries.

The position of the cursor determines which area to scroll. If the cursor is not in a scrollable area (except for a pull-down menu), the scroll request applies to the topmost scrollable area on the display. If the cursor is in a scrollable area, the scroll request applies to that area. The bounds of the scrollable area are the top and bottom line of that area as defined by the menu area (MENU), data presentation area (DATA), list area (LIST), and information area (INFO) language tags. For more information on these language tags, see Appendix A, "UIM Panel Group Definition Language," on page 465. Titles and instruction lines are part of the area but not part of the scrollable portion of the area.

The command line is not considered part of a scrollable area, but it may be associated with a particular area on the screen. For example, a panel with a scrollable menu area has a command line and it is associated with the menu. However, for scrolling purposes, the command line is not part of the area.

Scrolling and Error Conditions

The message line always exists and is always the last scrollable area on the screen so the scroll function keys must always be active.

When scrolling the message line, the UIM does not perform any validity checking or updating of the dialog variables. When scrolling any other area, validity checking or updating of the dialog variables is performed on all input fields on the screen. The scroll function is not performed if an error is found. All input and output fields on the screen must be correct before the scroll operation can proceed.

If a page key is pressed and the operation cannot be performed, a message is displayed. For example, a message is shown if a display is already at the top when the Page Up key is pressed. The alarm sounds when trying to scroll the message line past its top or bottom.

Scrolling a List Area

List processing allows an incomplete list to be displayed. If the Page Down key is pressed and the remainder of the incomplete list does not fill the list area, the UIM calls the specified application program that handles incomplete lists. This program adds additional entries to the list or marks the list as complete at the bottom, then the program returns to the UIM. The UIM knows an incomplete list by the setting of the contents attribute of lists on the Set List Attributes (QUISETLA) API. Once the list is complete, the UIM handles scrolling without application program intervention.

A list might be incomplete at both the top and bottom. If a list has missing information above the current location, pressing the Page Up key causes the same type of processing.

Scrolling a Menu Area

The UIM attempts to keep menu item groups together but splits the group if it does not fit in the defined area. Any individual menu item or text for a menu group must fit within the available scrollable space or the panel group is not created successfully.

Scrolling an Information Area

Using the Move to Top (MOVETOP) dialog command moves a cursor-selected line to the top of the scrollable information area. This allows the user to position the information in the most readable manner. For more information on the MOVETOP dialog command, see "MOVETOP (Move to Top)" on page 654.

Scrolling Data Item Groups

The UIM keeps data item groups together if possible. A group is split only if it does not fit within a column of the data area. When a data item group is nested inside another data item group and the outer group must be split, the UIM still attempts to keep the inner groups together. If a selection field does not fit in a column of the data area, the panel group is not created.

Data presentation areas can also be presented in multiple columns and the columns are filled from left to right. Generally, scrollable data presentation areas should not be defined with a two-column layout because the usability is poor.

Scrolling a Text Area

Scrolling for a text area must be handled by the text area exit program. UIM does not handle scrolling for text areas. When a key assigned to a scrolling dialog command is pressed and the text area should be scrolled, UIM calls the text area exit program. On return, the UIM redisplays the panel. The text area exit program should change the value of the dialog variable for the text area in order to scroll the text area.

A general panel exit program should be used to diagnose if the user has scrolled too far. If the user has scrolled too far, the general panel exit should send an appropriate message followed by the special message to cancel the determined action. For the message to cancel the determined action, see the section on the general panel exit program in the APIs topic.

Defining Contextual Help

Contextual help is provided jointly by the UIM and the system help function. Using the tag language, a panel is defined to associate help modules with different areas on the screen. When the Help key is pressed, the UIM determines which help modules to display.

When defining a panel, the user can associate help text with the following areas:

- Entire panel
- Menu bar
- Menu bar choice
- Pull-down field choice
- Specific menu item
- Specific data item
- Specific data selection field
- Choice for data selection field
- Specific list column
- List column group
- Specific function key
- Specific option in an action list
- Data group
- Data area

When the Help key is pressed, the UIM displays the help text depending primarily on where the cursor is. The rules are as follows:

Cursor Position	Help Selection Rules
On a menu item	Help for that menu item is displayed.
In a list column	Help for that column is displayed. The left and right boundaries of a column are defined by the left and right edge of the field or heading, whichever is greater, plus one blank on either side. The top and bottom boundaries of a column are defined by the first line of the heading or column group heading and the last line where a list entry may be displayed. Help can be associated with a list column group rather than the individual entries. In this case, positioning within the group gives help for the entire group regardless of the specific column the cursor is in. Boundaries of column groups are defined by the sum of the individual column boundaries plus any separators between columns of the group. If the column for which help is being displayed is the option column of an action list, the help displayed includes the help specified for the list column and the help specified for each active list action.
On any line in the function-key-definition area	Help for all active function keys is displayed.
On a message line with a message displayed	Help for that message is displayed. Help for a message is provided by the Additional Message Information display. Help for a message does not use a help module in a panel group.
On a menu bar choice	Help for that menu bar choice and all currently active pull-down choices for that menu bar choice is displayed.
On a pull-down choice	Help for that pull-down choice is displayed.
In a pull-down input field	Help for the menu bar choice (see ″On a menu bar choice″ above) that the pull-down is associated with is displayed. If a valid pull-down choice is entered in the field, help for that pull-down choice is displayed.
On a command line	The displayed help depends on the contents of the command line. If the command line is associated with a menu area and the command line contains a number, help for that menu item is displayed as if the cursor were placed on that menu item. Help is available for any active menu item regardless of whether it is displayed. If the command line is assumed to contain a command, help for that command is displayed. If the command line is being treated as a parameter line, extended panel help is displayed. If the command line is blank, extended panel help is displayed.
On a data item	If help is specified for the data item, that specific help is displayed. If no help was specified on the data item, the data item is part of a group, and if help was specified at the group level, then data group help is displayed. If no help was specified on the item, the data item is not part of a group, or if no group help was specified, and help was specified on the area, then area-level help is presented.
On a data group heading	The same rules apply as when the cursor is on a data item. Help is presented for the lowest level (group, area, or extended panel).
In a data area	If help is available for the area, that level of help is displayed. Otherwise, extended panel help is displayed.

Cursor Position	Help Selection Rules
Any other position	If none of the above conditions are met, extended panel help is displayed. This is true whenever the cursor is on the title line, any instruction line, and most blank spaces. Whenever there is no specific help to display, extended help is displayed.

Although HELP for a specific action on a list option or function key is never displayed, help text is individually defined for them. This allows conditioning of active options and function keys. The displayed help describes only the active function keys or options.

Command Line Restrictions

The UIM allows two sizes for a command line:

- A short command line is one line
- A long command line is two lines

Depending on whether the command line is short or long, the UIM formats and displays the command line to occupy the one or two lines preceding the function key area. This position remains unchanged regardless of scrolling, item conditioning, or the number of list entries.

Command lines are defined by the command line (CMDLINE) language tag. For more information on the CMDLINE tag, see "CMDLINE (Command Line)" on page 491. The UIM allows specifying this tag on any panel, but does not require it.

When a command line is defined for a panel, it is recommended that the F4 key be assigned to the PROMPT dialog command and the F9 key be assigned to the RETRIEVE dialog command. This allows the user to prompt for commands entered on the command line and to retrieve previously entered commands. For more information on dialog commands, see Appendix B, "UIM Dialog Commands," on page 641, and for more information on defining function keys, see "KEYL (Key List)" on page 545.

The UIM sometimes associates the command line with one area on the panel. If a menu area is present, the command line is associated with the menu area. If an action list is present, the command line is associated with the action list. If neither is present, the command line is not associated with an area on the panel. The UIM does not allow two menu areas, two action lists, or a combination of these within a single panel. Also, command lines and menu option lines are mutually exclusive. For more information on the option line (OPTLINE) tag, see "OPTLINE (Option Line)" on page 593.

Command Line Interpretation

Command lines are used to select a menu option, to run i5/OS or System/36 Environment commands, and to specify parameters used in conjunction with options on an action list. The UIM selectively interprets the character strings entered on the command line.

Many dialog commands do not require any input on the command line. The following conditions assume the action is one requiring the UIM to know the command line contents. These actions are for the ENTER, HELP, and PROMPT dialog commands. For more information on the Enter, Help, and Prompt functions, see Appendix B, "UIM Dialog Commands," on page 641.

- For a command line associated with a menu, the UIM examines the first nonblank set of characters. If it is composed of only digits (hexadecimal F0 through F9), the command line is assumed to select a menu item. Otherwise, the command line is treated as containing a command.
- For a command line associated with an action list, the command line is assumed to contain parameters if any option is selected on any action list entry. Otherwise, the command line is treated as if it contains a command.
- For a command line associated with a panel that contains neither a menu nor an action list, the command line is always assumed to contain a command.

Exactly what function is performed next depends on what function key is pressed. For details, see the descriptions of the ENTER, HELP, and PROMPT dialog commands.

Entering Commands That Are Too Long

Commands that are too long for the command line can still be submitted by prompting for them. If the command causes an exception, the command line is displayed with the current command contents. If the full command string is too long for the command line, the display shows as much as possible and replaces the last three characters on the displayed command line with ellipses (...) to indicate continuation. The UIM still maintains the full command string internally.

Any changes to the command line causes the UIM to discard the internal version and treat the changed command line as a new request. If no change is made and the command is prompted or submitted, the internal version is not reset and is submitted as the new command string. Modification is based on a character comparison of the command line contents.

Defining Function Keys

The UIM does not automatically enable function keys; the application developer is responsible for enabling the correct set of function keys and ensuring that the correct functions are assigned to each. Function keys are defined with the key list (KEYL) and key list item (KEYI) language tags. For more information on these language tags, see Appendix A, "UIM Panel Group Definition Language," on page 465. The UIM requires that engraved keys be assigned to their dialog command equivalent. The Help, Enter, Page Up, Page Down, Print, and Home keys must be assigned their respective dialog commands. With the exception of the PRINT dialog command, this does not prevent assigning these dialog commands to other keys as well. For example, it is recommended that the HELP dialog command be assigned to the F1 key as well as the Help key. The PRINT dialog command can be assigned only to the Print key and cannot be conditioned.

F1 through F24 can be assigned to any of the dialog commands except PRINT and MOREKEYS. The MOREKEYS dialog command can be assigned only to F24.

Formatting Function Keys

When a panel is created, the UIM determines the worst-case display of function keys and their descriptions and then allocates one or two lines accordingly. Two lines is the maximum amount of room available to display function keys. Then this space is fixed. For example, if the UIM determines that two lines are required for a specific set of conditions, then two lines are always used, even if some conditions cause only one line of description for the function keys.

Handling Function Keys and VARUPD Value

The dialog command assigned to a function key determines the function of that particular key. For more information on handling function keys, see Appendix B, "UIM Dialog Commands," on page 641.

Function keys that are inactive due to conditioning are treated as if they are not defined. If the user presses a function key that is not defined, a message is displayed to indicate that the key is not allowed. When help is displayed for the function key area, no help information is displayed for inactive keys.

When defining a function key, the `VARUPD` attribute of the key list item (KEYI) or pull-down field choice (PDFLDC) language tag defines whether dialog variables and list entry values should be updated with user entered values when the function key is pressed or when the pull-down choice is selected. If any fields on the panel fail validity checking, the action associated with the function key or pull-down choice is not performed.

Although most dialog commands have a predefined, unchangeable value for VARUPD, some dialog commands do not. When a function key is assigned to one of these dialog commands, processing of that request depends on the value of VARUPD.

When VARUPD=YES is specified, all values keyed in by the user must pass validity checks. If any values fail the validity checks, the following occurs:

- Any dialog variable that does not pass validity checking is not updated.
- The specified function is not performed.
- The general exit program for the panel, if specified, is not called.
- The UIM displays the same panel again with the appropriate error messages.

When VARUPD=NO is specified, the following occurs:

- No validity checking is performed and no dialog variables are updated.
- If specified, the general exit is called. This assumes that the dialog command to be performed is one for which the exit program is normally called.
- The specified function is performed unless the general exit program indicates that the UIM should not perform the function. For more information on exit programs, see the APIs topic.

When VARUPD=NO is specified for a dialog command, which causes the UIM to return control to the application program, variable values entered by a user are not available to the application program. The values are stored for the panel and can be shown again by using the redisplay parameter of the Display Panel (QUIDSPP) API. For more information on this, see theAPIs topic. When VARUPD=NO is specified on the CMD and CALL dialog commands, the UIM saves the screen copies of dialog variables and uses them when the panel is displayed again after the specified action completes.

Note that when VARUPD=NO is specified on the CMD and CALL dialog commands, any modification of dialog variables or list entries causes the saved version to be lost. The saved version for any field is used if the underlying dialog variable has not changed since the field was saved. For lists, the saved version of a variable in a list entry is used unless one of the following occurs:

- Underlying list entry has changed
- List entry changed position on the screen
- List view has changed

If a change was made, the saved version is lost and a new displayed value is derived from the dialog variable or list entry.

For example, assume dialog variable VARX is present on panel PANELX and is modified by the user just before pressing a function key using the CALL dialog command where VARUPD=NO. If the target of that CALL dialog command modified VARX, the saved version is lost. When the UIM displays the panel again after the call returns, the displayed version of VARX is based on the modification. This allows actions, which are unrelated to the current panel, to be performed without requiring the panel contents to pass all validity checks. VARUPD=NO should not be used with CMD or CALL dialog commands if the intended action is to use the same panel or any of its associated dialog variables, lists, or conditioning dialog variables.

Panel Formatting Concepts

When formatting UIM panels, keep the following in mind:

- The application developer does not define the format of a panel in terms of rows and columns. Instead, the UIM interprets the descriptions of the tag language and then determines where to place each element in a panel. For example, the display panel (PANEL) tag contains a description of the panel title, but no explicit definition of where the title should be displayed. The UIM determines that the title should be centered within the panel width, intensified, and treated as mixed case. The same concept applies to all other areas of the panel, though most other examples are much more complex.

- The UIM performs as much formatting as possible when the panel group is created, but the actual placement for many panel elements is determined at run time. This allows the UIM to consider screen depth and other run-time circumstances. For example, the UIM allows menu items to be conditionally displayed. These conditions are evaluated when a panel is displayed, and the format of the menu area is based on the results.

When Panel Formatting Is Performed

There is a difference between panel formatting and panel contents. **Panel formatting** refers to the organization of the various areas on the panel. **Panel contents** refers to the values of fields as dictated by dialog variable contents.

Panel formatting takes place when the panel group is compiled and again at run time.

Most decisions are made when the panel group is compiled and the information is kept in the panel group object. Those formatting decisions that cannot be made when the panel group is compiled are made during Display Panel (QUIDSPP) API processing.

An example of this distinction is a data item that is conditioned. All details about the prompt text layout, number of leader dots, field position, and formatting of possible choices are performed when the panel is created.

At run time, the QUIDSPP API processing determines whether or not the item should be displayed by evaluating the conditions and builds an internal format. Within an application defined by the Open Display Application (QUIOPNDA) and Close Application (QUICLOA) APIs, each panel has only one internal format at any time. The internal format is built the first time the panel is displayed, and the format is kept for subsequent use until the application is closed.

This does not mean that the internal format cannot change within an application. Most of the internal format does not change, but some panel elements such as menu items, list actions, and function keys, are conditioned by the application. The effect of changes to these areas are evaluated each time the panel is displayed, and then the format is updated.

Because the UIM performs many functions, such as submitting commands and calling programs, before returning control to the application that called the QUIDSPP API, the specified panel can be presented to the user several times. Each time the panel is displayed, the UIM reevaluates the panel formatting. For example, pressing a function key causes an application program to be called. When the application program returns control to the UIM, the UIM displays the panel again, but only after checking to see if the application program changed the conditions that affected the internal format. For example, if the QUIDSPP API is called to display a panel with a menu area on it, and the application program changed a dialog variable affecting the conditioning of some of the menu items, when the UIM displays the menu area again, it reflects the changes.

Application Control of Panel Formatting

The language tags allows the application developer to control panel formatting in the following ways:

- By specifying part of the formatting, such as column widths, vertical versus horizontal presentation of data areas, and depths of the areas. These specifications are fixed when the panel is compiled and cannot be changed at run time by the application. This allows you to select alternate formatting techniques and it gives the UIM information that helps reduce run-time processing.
- By specifying run-time conditioning. Dialog variables can be used to condition menu items, menu item groups, action list options, function keys, data items, pull-down choices, data selection fields, selection field choices, and data item groups. In most cases, these conditions are evaluated every time a panel is displayed to determine what should be included in the display.

Conditioning can be based on any of the following:

- Application defined dialog variable.
- UIM defined dialog variable.
- Existence of an object on the system or the user's authority to an object on the system.
- User class of the application user.
- Conditioning can be based on the setting of a user profile, a user class, or the Limit User Capability (ZLMTCPB) dialog variable. For more information on dialog variables, see "Dialog Variables Defined by UIM" on page 635.

For more information on defining conditions, see "COND (Condition Definition)" on page 492.

The UIM allows additional control over conditioning. The panel definition can specify which conditions are evaluated only once, when first needed, or at all times. The purpose of this is primarily related to performance. Most items are conditioned on things that are or should be fixed when the panel is first displayed, such as a user profile and authorization. Specifying that these conditions be evaluated only once avoids unnecessarily evaluating the conditions.

Limits of the Panel Formatter

The UIM provides a formatter designed to handle most normal formatting requirements. However, the formatter is not always able to match the results of manual formatting because the formatter is constrained by the run-time environment and also must handle generalized input. An example of this is formatting descriptions of function keys. Manual formatting might include changing the wording to achieve the best alignment and balance. The UIM does not have the option of changing the text length or assuming a balanced set of lengths.

Generally, the closer information is packed on a panel, the less likely the UIM formatter can match what manual formatting might achieve.

Folding Up Multiple Panels When EXIT Is Requested

To support the ability of an application to fold up multiple panels and programs as part of processing a single exit request, the operating system maintains a flag for each job indicating whether or not exit is requested. This exit flag is used in combination with the user task parameter on the Display Panel (QUIDSPP) API to cause the exit request to fold up multiple panels.

There are two ways the application program can turn on the exit flag:

1. Specify the SET parameter on the EXIT dialog command. When the UIM performs the EXIT dialog command with the SET parameter, it turns on the job's exit flag. For more information on the EXIT dialog command, see Appendix B, "UIM Dialog Commands," on page 641.
2. Set the UIM-defined variable ZEXIT to ″1″ using the Put Dialog Variable (QUIPUTV) API. By setting this Z-variable, the UIM turns on the job's exit flag. The UIM automatically turns off the exit flag when it begins processing actions.

Whenever the UIM regains control after performing a panel-defined action, such as submitting a command assigned to a function key or calling a program because a menu item is selected, the job's exit flag is checked. If the flag is on and the panel is displayed as an old user task (user task parameter on the QUIDSPP API), the UIM returns control to the calling application at that point. Also, the function-requested parameter of the QUIDSPP API contains an indication that the panel display ended due to the EXIT dialog command.

If the job's exit flag is off or the panel is displayed as a new user task, nothing special happens. The UIM turns off the exit flag and displays the panel again.

If the action performed is not panel-defined, for example, when a command is entered on the command line, the exit flag is always turned off, and the processing is not affected by whether the panel is displayed as a new or old user task.

An example of how the exit flag works follows; Figure 113 illustrates the example. Assume that program PGMX, displays PANELX that has a function key assigned to the CALL dialog command. Assume also that the program that is called, PGMY, displays another panel called PANELY. Now assume that PANELX is displayed as an old user task, the function key is pressed, and PANELY is displayed. From PANELY, the user presses the function key assigned to the EXIT dialog command using the SET parameter and control returns to PGMY. When the EXIT dialog command is processed, the UIM turns on the exit flag. If PGMY now returns control to the UIM, the UIM returns control to PGMX with an indication that the EXIT dialog command was requested. In effect, two panels are bypassed via a single exit request.

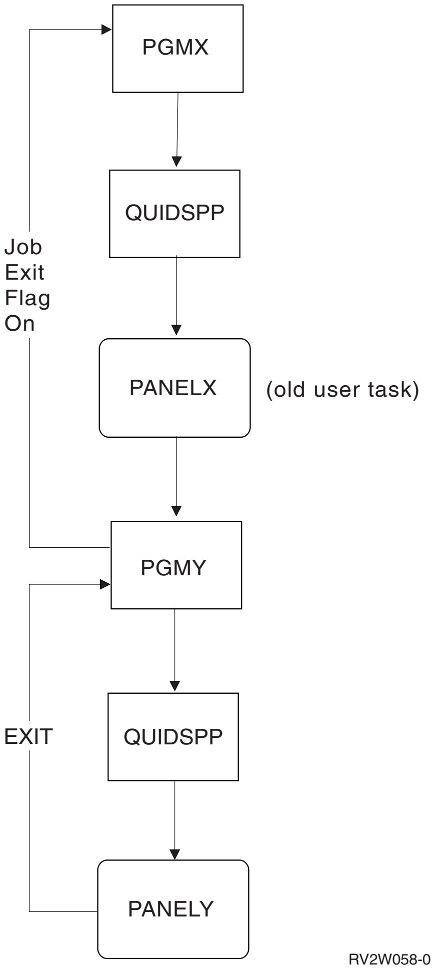


Figure 113. Example of Job Exit Flag

If the function key on PANELY assigned to the EXIT dialog command does not use the SET parameter, the UIM displays PANELX again because the job’s exit flag is not turned on.

For more information about APIs, see the **APIs** topic in the iSeries Information Center.

Folding Up a List Panel

For panels displayed as a result of processing a list option on an action list panel, the CANCEL, EXIT and ENTER dialog commands should work as follows.

The operating system maintains a flag for each job indicating whether or not cancel is requested. This flag is similar to the flag described for exit processing; see "Folding Up Multiple Panels When EXIT Is Requested" on page 354. However, the cancel flag is not used to bypass more than one panel, and it is not used in conjunction with the user task parameter on the Display Panel (QUIDSPP) API.

There are two ways the application program can turn on the cancel flag:

1. Specify the SET parameter on the CANCEL dialog command. When the UIM performs the CANCEL dialog command with the SET parameter, it turns on the job's cancel flag. (For more information on the CANCEL dialog command with the set parameter, see Appendix B, "UIM Dialog Commands," on page 641.)
2. Set the UIM defined variable, ZCANCEL, to ″1″ using the QUIPUTV API. By setting this Z-variable, the UIM turns on the job's cancel flag.

It is the application developer's responsibility to ensure that pressing the Cancel key stops processing of list actions and displays the action list panel again at the point where processing was stopped. If another panel can be displayed as a result of processing a list option, the job's cancel flag should be turned on when the Cancel key is pressed by specifying the SET parameter on the CANCEL dialog command. When the UIM performs action-list processing, it checks the cancel flag after each list action completes. When the UIM finds that the flag is on, it stops processing the remaining list options and displays the list again. The option field for the list entry just processed is cleared and the options of list entries whose action was not attempted are left in the list. The UIM only checks the cancel flag after processing a list action.

It is the application developer's responsibility to ensure that pressing the Exit key stops the processing of list actions and either redisplays the action list panel at the point where processing was stopped or exits the action list panel. If another panel can be displayed as a result of processing a list option, the job's exit flag should be turned on when the Exit key is pressed by specifying the SET parameter on the EXIT dialog command. When the UIM performs action list processing, it checks the exit flag after each list action completes. When the UIM finds that the flag is on, it stops processing the remaining list options and the following occurs:

- If the action list panel is displayed as a new user task, the panel is displayed again.
- If the action list panel is displayed as an old user task, the panel is not displayed again, and control is returned to the program that displayed the action list panel with a return function indicating EXIT was used.

In either case, the option field for the list entry just processed is cleared and the options of list entries whose action was not attempted are left in the list. The UIM also checks the job's exit flag when it gets control back after processing menu options, function keys and pull-down menu choices. For more information, see "Folding Up Multiple Panels When EXIT Is Requested" on page 354

It is the application developer's responsibility to ensure that pressing the Enter key continues the processing of list actions. When neither the cancel flag nor the exit flag is on, the next list action is processed.

Adding a Pop-Up Window over Another Panel

You can add pop-up windows over another panel in one of the following ways:

- Use the CALL dialog command to have the UIM call an application exit program so it can add and display the window. To achieve consistent results from the End Request (ENDRQS) command, this is the recommended design.
- Use the RETURN dialog command to have the UIM return to the program that displayed the underlying panel so it can add and display the window.

The UIM does not become a request processor program when displaying a panel in a pop-up window that has a menu or action list unless the panel also has a command line. This is true even when the window is displayed over a primary panel that has a command line, action list, or menu area.

When the UIM becomes a request processor program while displaying a full-screen panel, and the application is designed to use the CALL dialog command to add and display a pop-up window, the program stack contains the UIM request processor program for the full-screen panel. However, when the application is designed to use the RETURN dialog command to add and display a window, the program stack does not contain the UIM request processor program. Therefore, the ENDRQS command causes different results depending on the internal design of the application.

As a general guideline, you should avoid designing long-running functions that run as a result of a list option or menu option from a pop-up window, unless your program is a request processor at appropriate times.

If your program becomes a request processor program and displays one or more pop-up windows by using a design that relies on the RETURN dialog command, then the code needs to clean up the window stack if the function that displayed a window is canceled by the ENDRQS command. When this happens, the Remove Pop-up Window (QUIRMVPW) API must be called once for each window to be removed from the application window stack.

If your program becomes a request processor program and displays one or more pop-up windows by using a design that relies on the CALL dialog command, then the UIM cleans up the window stack automatically if the function that displayed a window is canceled by the ENDRQS command.

For more information on a request processor program, see the **CL** topic in the iSeries Information Center.

Using Menu Bars

A menu bar is located at the top of a panel and extends the entire width of the panel. Menu bars are allowed only on panels that are 24 by 80 bytes or 27 by 132 bytes and are not allowed within a pop-up window.

The menu bar contains a list of choices that users may request. The choices are listed horizontally on one to three lines with three blanks between all choices.

When a user selects a choice by pressing the Enter key, a pull-down menu is displayed directly below the menu bar. From this menu, the user may select one choice.

Each choice in a pull-down menu does one of the following:

- Performs an action against the panel or an area of the panel as a whole. In this case, the UIM performs the action once each time the choice is selected by the user.
- Performs an action against each entry selected in an action list or selection list on the panel. In this case, the UIM performs the action once for each selected list entry. The user selects an entry or an action on a selection list by entering a slash (/) or country-designated selection character next to the desired entry.

For more information on defining menu bars, see the following language tags in Appendix A, “UIM Panel Group Definition Language,” on page 465.

- PANEL (Display Panel)
- MBAR (Menu Bar)
- MBARC (Menu Bar Choice)
- PDFLD (Pull-down Field)
- PDFLDC (Pull-Down Field Choice)

Differences Between Pull-Down Menus and Pop-Up Windows

A pull-down menu visually appears similar to a pop-up window. While pull-down menus and pop-up windows overlay a portion of the underlying panel image, a pop-up window is a separate application panel and a pull-down menu is an extension of the panel which contains the menu bar.

Visually, a pull-down menu appears different from a pop-up window in the following ways:

- The separator line appearing below the menu bar is always used as the top border of the pull-down menu. The border for a pop-up window is never part of the underlying panel.
- A pull-down menu does not have a title. A pop-up window can have a title.
- A pull-down menu does not have function key descriptions and a pull-down menu never overlays the function key descriptions of the underlying panel. Pop-up windows usually have function key descriptions and a window can overlay the function key descriptions of the underlying panel.
- A pull-down menu does not have a message line and any messages displayed as a result of the user interacting with the pull-down menu appear in the message line of the underlying panel. A pop-up window always contains a message line and any messages displayed as a result of interaction with the pop-up window appear in the message line of the pop-up window.

Pull-down menus and pop-up windows also differ in how the user is allowed to interact with each. The differences are as follows:

- When a pull-down menu is displayed, pressing the cursor tab key moves the cursor to the first unselected choice within the menu bar. Press the tab key again to move the cursor to the next unselected choice, until the cursor is moved back to the pull-down menu. When a pop-up window is displayed, pressing the cursor tab key moves the cursor to the next input field defined within the pop-up window. The tab key never moves the cursor outside the border of the pop-up window.
- When a pull-down menu is displayed, all the function keys defined for the underlying panel are active. Each function key performs the same function as if the pull-down menu was not displayed with the following exceptions:
  - The Enter key, or any other key assigned to the ENTER dialog command, is used to process the action assigned to the pull-down choice number selected by the user.
  - Any key assigned to the CANCEL dialog command causes the pull-down menu to be removed and the underlying panel to be redisplayed. Normally, F12 is the key assigned to the CANCEL dialog command.
  - The Page Down and Page Up keys, or any other key assigned to the PAGEDOWN or PAGEUP dialog commands, are not allowed when the cursor is within the pull-down menu.
  - Any key assigned to the PROMPT dialog command is not allowed when the cursor is within the pull-down menu. Normally, F4 is the key assigned to the PROMPT dialog command.

  For cursor-sensitive function keys, the user is allowed to move the cursor outside the border of the pull-down menu and press the function key. In this case, the UIM removes the pull-down menu and processes the action assigned to the function key as if the pull-down menu was not displayed at the time the user pressed the function key. Cursor-sensitive function keys are any keys assigned to the dialog commands in the following table. The dialog commands in the table are usually assigned to the keys as shown.

Table 33. Cursor-Sensitive Function Keys Assigned to Dialog Commands

Dialog Command That May Have a Cursor-Sensitive Function Key Assigned	Dialog Command Is Usually Assigned to This Key
HELP	Help
PAGEDOWN	Page Down
PAGEUP	Page Up
MOVETOP (move to top)	F10
CHGVIEW (change view)	F11

Table 33. Cursor-Sensitive Function Keys Assigned to Dialog Commands (continued)

Dialog Command That May Have a Cursor-Sensitive Function Key Assigned	Dialog Command Is Usually Assigned to This Key
PROMPT	F4

As an example, assume that a pull-down menu is displayed. If the user moves the cursor to a field in the underlying panel, outside the border of the pull-down menu, and presses the Help key, the UIM removes the pull-down menu and displays the contextual help defined for the field.

When a pop-up window is displayed, only the function keys defined for the panel displayed within the window are active. When any function key is pressed and the cursor is located outside the border of the pop-up window, the window is redisplayed and the alarm sounds to indicate that the cursor is not allowed outside the window.

Further differences between pull-down menus and pop-up windows are in how the application developer can control them. Following are the differences:

- A pull-down menu can only contain a list of numbered choices. The application developer defines each choice, which appears in the pull-down menu, and also assigns an action for each choice. When the user selects one of the choices, the UIM performs the action defined for that choice. A pop-up window can contain any panel element, which can appear in a full-screen panel, with the following exceptions.
  - A pop-up window cannot contain a field (dialog variable) which spans multiple lines of the display. Every field must fit within the width of the window.
  - A pop-up window cannot contain a menu bar.
- Based on user interaction, the UIM decides when a pull-down menu is displayed and removed. The application program cannot control when a pull-down menu is displayed or removed; nor can the application program control the location or size of the pull-down menu when it is displayed. The application program has control over when and where a pop-up window is displayed and removed. For more information about displaying pop-up windows, see "Adding and Removing Windows" on page 360. The size of a pop-up window is specified by the application developer by specifying the WIDTH and DEPTH attributes on the PANEL tag for the panel to be displayed in the window.

Using Pop-Up Windows

A pop-up window is information that overlays part of the display. The user can view information inside the window and the portion of the screen that is not overlayed by the window. However, only the window is active; the user cannot work with the underlying panel. When more than one window is displayed, only one window is active at a time.

Following are some principles of pop-up windows:

- Windows contain a message line, and if the message exceeds the boundary of the window, the message is truncated. To indicate the message is truncated, an ellipsis (...) is added to the end of the message. To see the rest of the message, place the cursor on the message line of the window and press the Help key.
- When a window is displayed, its size and location cannot be changed.
- The cursor is positioned in the active window. If the user moves the cursor outside the active window and presses any function key except Print, the display alarm sounds and the cursor moves back to its previous window location.
- When you leave a window, you are returned to the underlying display at the location where you were before you requested the window.
- The Print key applies to the entire display, not just to the current window.

Defining Application Windows

Application-defined windows can be displayed over UIM and DDS panels.

Application-defined means that the application developer defines the exact size of the window on the display panel (PANEL) tag on the panel group source. The location of the window is specified when the application program calls the Add Pop-Up Window (QUIADDPW) API.

Adding and Removing Windows

The UIM maintains a stack to keep track of windows. Windows can be added to the stack in one of the following ways:

- The UIM adds windows to the stack in order to display help information and command line windows.
- The application developer can request that a window be added to the stack to display a window.

Windows can be removed from the stack internally by the UIM or by the application developer.

Adding and removing windows is controlled by the application program, but the UIM provides a set of APIs to manipulate these windows.

To display a window, an application program must first use the Add Pop-Up Window (QUIADDPW) API to inform the UIM that the next display is for a window. A window displaying a selection list should use field-adjacent positioning, while a window displaying a prompt for positioning a list should use offset positioning.

The QUIADDPW API call does not specify the panel to be displayed in the window. It specifies only the location information necessary to position the window.

A window can be added to a display only after a full-screen panel is determined. A full-screen panel is determined by calling the Display Panel (QUIDSPP) or Set Screen Image (QUISETSC) API. Once a full-screen panel is displayed within an application, a window can be added to the application.

To position a selection list window using field-adjacent positioning, the programmer must specify the name of the field for which the selection window is being displayed. When a panel is displayed within the window, the UIM uses a set of general window positioning rules to position the window so that, if possible, the window does not overlay the specified field.

For a window displaying the prompt for positioning a list, offset positioning should be specified on the Add Pop-Up Window (QUIADDPW) API call. The upper left corner of the window is positioned below and indented to the right of the upper left corner of the underlying panel.

Once a window is added to the display, the Display Panel (QUIDSPP) API is used to display a panel within the window. When the QUIDSPP API is called, the UIM performs the following operations:

1. Determines the actual location for the window. The location is determined according to the location information provided by the call to the QUIADDPW API.
2. Formats the panel.
3. Merges the newly formatted panel and window border with the underlying panel image.
4. Displays the merged panel output.

The end user must interact with the most recently displayed window before interacting with any other underlying panel or window.

Use the Remove Pop-Up Window (QUIRMVPW) API to remove windows from the display. A window cannot be removed if the UIM is currently processing an action or running an exit program for the panel displayed in the window. Multiple windows can be removed at one time.

The QUIRMVPW API does not cause the UIM to automatically display the underlying panel again. The panel is displayed again without the window on the next call to the QUIDSPP API, or when the UIM automatically displays a panel again after processing a function or running an exit program.

Using the Command Line in a Window

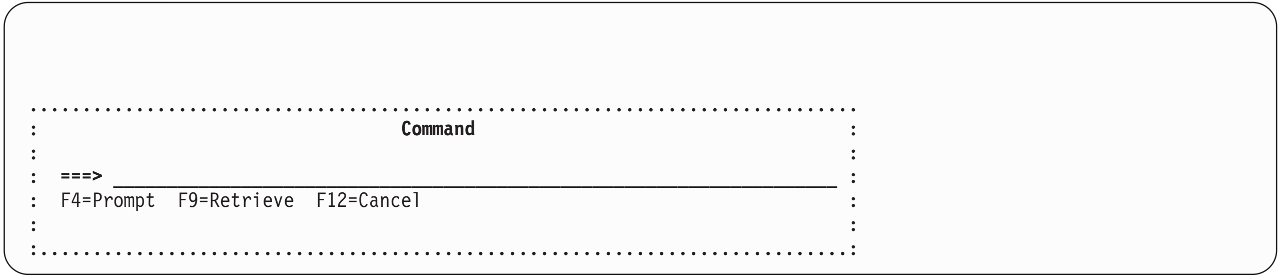
A command line in a window allows the user to enter commands without requiring any reserved space on the panel for a permanent command line. The UIM provides support to display a command line in a window. For UIM application panels, this is done by assigning a function key to the CMDLINE dialog command. The F9 key is recommended. The CMDLINE dialog command is also allowed as the action for a choice in a pull-down menu. When a function key or pull-down field choice is assigned to the CMDLINE dialog command, the key or choice can be conditioned based on the ZLMTCPB dialog variable. Doing this only allows the user to get to the command line window if the user does not have limited capabilities.

For DDS panels, a command line window can be provided by calling the QUSCMDLN API.

Keep the following in mind when using a command line in a window:

- A pop-up window is displayed when the user presses a function key causing the Command Line (CMDLINE) dialog command to process. The cursor is placed on the first position of the command line and the user can use the command line to enter any system commands.
- The UIM provides a panel definition for the window and puts it at the bottom of the display.
- Neither the size nor the location of the window for the command line can be changed. The user must remove the window by pressing the Enter key or the F12 (Cancel) function key. Running a command does not remove the window.

The following example shows a command line pop-up window used on a display.



```
..........................................................................
:                                Command                                 :
:                                                                        :
:  ===> ________________________________________________________________ :
:  F4=Prompt  F9=Retrieve  F12=Cancel                                    :
:                                                                        :
:........................................................................:
```

UIM as a Request Processor Program When Displaying a Panel

Sometimes the UIM becomes a request processor program when displaying a panel. When the UIM becomes a request processor program, your application is isolated from commands entered on the command line. For example, if the End Request (ENDRQS) command is used to cancel a command entered on the command line, the UIM handles the end request and the application program is not ended. The following types of panels cause the UIM to automatically become a request processor:

- Any panel with a command line. The UIM is required to be a request processor when submitting a command from a command line. When the panel has a command line, the UIM becomes a request processor program so that all functions it performs for the panel work consistently for the End Request (ENDRQS) command.
- A full-screen panel with an action list. Most action list panels have a command line to allow parameters for list options. So for consistency, the UIM becomes a request processor program for all full-screen panels with an action list. This allows ACTOR=UIM action lists without a command line to work in the same way as those that have one when the ENDRQS command is requested while a list option is being processed. This allows a user to perform the ENDRQS command to cancel a long-running option without also canceling the action list panel and the program that displayed it. Any unprocessed list options are left pending in the action list when the ENDRQS command is used to cancel a UIM-processed list option.

  If you code ACTOR=CALLER for an action list, you should consider having your code become a request processor program for consistency with the UIM.

- A full-screen panel with a menu area. Most menu panels have a command line. For consistency, the UIM becomes a request processor program for all panels with a menu area. This allows menu panels without a command line to work in the same way as those that have one when the ENDRQS command is requested while a menu option is being processed. This allows a user to perform the ENDRQS command to cancel a long-running option without also canceling the menu panel and the program that displayed it.

  If you code the RETURN dialog command for a menu item, you should consider having your code become a request processor program for consistency with the UIM.

  For more information on a request processor program, see the **CL** topic in the iSeries Information Center.

Printing Concepts

The UIM allows applications to define and generate printed output to the level required for the OUTPUT(*PRINT) parameter on a CL command. This support provides a hardcopy form of information in a format similar to that displayed by the UIM, but it does not allow for generalized printing via the UIM.

The application developer does not need to provide a print panel and a display panel. However, it is recommended that both print and display panels be placed in the same panel group. For printing panels, use the Print Panel (QUIPRTP) API.

A panel group can be opened for either display or print. Once a panel group has been opened for display, it can also be used to produce printed output. The Add Print Application (QUIADDPA) API is provided to open a printer file for an already open display application, and the Remove Print Application (QUIRMVPA) API closes the printer file that was opened with the QUIADDPA API.

For more information about APIs, see the **APIs** topic in the iSeries Information Center.

The UIM tag language is used to define print head panels and print panels. The print head panel, created with the print head (PRTHEAD) tag, defines the header information that is printed at the top of each page. The print panel, created with the print panel (PRTPNL) tag, defines the different panel areas to be printed. Both the print head panels and print panels must be printed for the same open application.

Consider the following points when you create help text:

- Some tags truncate a line of text if the text does not fit into the print area.
- Other tags (for example, XMP) wrap the text to the next line.
- The UIM allows only 72 columns of data.

When you nest tags, you must also factor in the width of the nested lines. For example, if you define an unordered list and include an example in one of the list items, the text for the example is indented two bytes to the right of the text in the list item.

The following areas make up a printed listing and the example that follows is how it would look:

Title The title consists of two lines printed at the top of every page. The title lines contain information such as the output title, time and date, system name, product information, and page number. All of the information in the title lines is defined by the print head panel.

Prolog *(Optional.)* The prolog section is printed only on the first page and comes right after the title lines. The prolog section is used for such things as indicating what parameters were specified to print the output. The prolog section is defined as the portion of the print head panel that has TYPE=PROLOG specified on the data presentation area (DATA) or information area (INFO) tag.

Header
(Optional.) The header section is used to print the same data on every page. This data usually

includes things such as the file name, library, and member name for which information is printed. This section comes after the prolog on the first page and after the title lines on all remaining pages. The header section is defined as the portion of the print head panel that has TYPE=NORMAL specified or defaulted.

Page body
The page body is made up of one or more panels of information. These panels of information appear very similar to panels that are used to display information on a display. These panels can contain data, information, and list areas. Menu areas and selection fields cannot be defined for printing. Because there is no such thing as a scrollable print panel, all information and data associated with the panel is printed via the Print Panel (QUIPRTP) API.

Trailer The trailer is a single line of text that is printed after the last panel area on the last page. This trailer usually says something like

```
E N D   O F   L I S T I N G
```

or

```
E N D   O F   S O U R C E
```

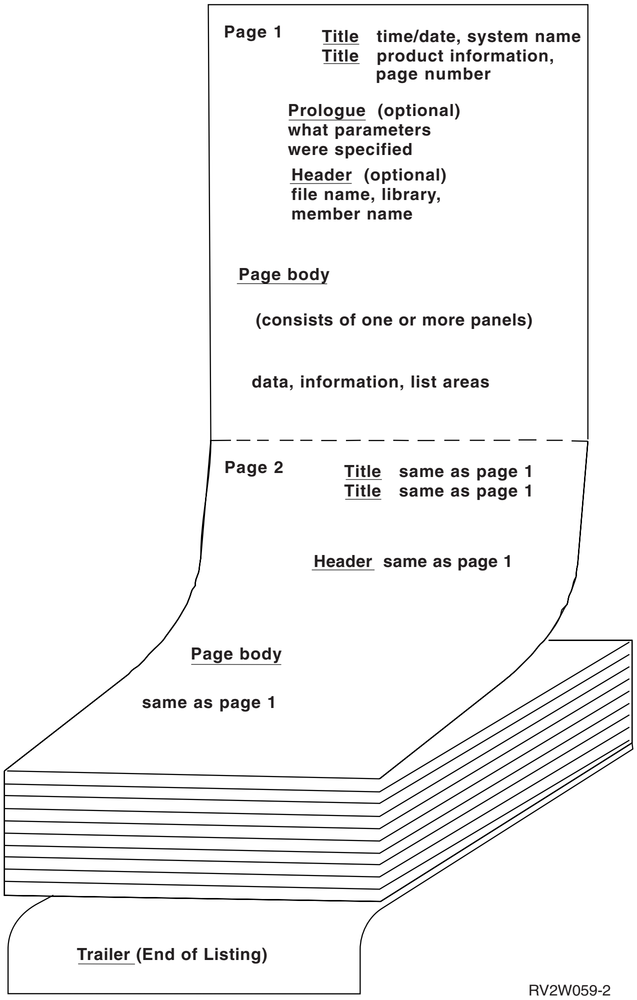

Page 1
Title time/date, system name
Title product information, page number
Prologue (optional) what parameters were specified
Header (optional) file name, library, member name
Page body
(consists of one or more panels)
data, information, list areas
Page 2
Title same as page 1
Title same as page 1
Header same as page 1
Page body
same as page 1
Trailer (End of Listing)
RV2W059-2

Figure 114. Example of Printout

Printing a Print Head Panel

When a print head (PRTHEAD) panel is printed by calling the Print Panel (QUIPRTP) API, the heading is formatted and saved until a print panel is printed.

Printing a Print Panel

When a print panel (PRTPNL) is printed by calling the Print Panel (QUIPRTP) API, the UIM:

1. Formats one page of data and prints it.
2. Formats the next page of data and prints it.
3. Formats page by page until the entire panel is printed.
4. Formats and prints the last part of the panel on a partial page. The next call to the QUIPRTP API causes the rest of the partial page to be formatted and printed, assuming that no page eject is requested. If the first area of this next panel does not fit on one page, this area is printed on the next page. This causes the panel title, top separator, and area title, if one exists, to be printed on the next page.

Using Blank Lines for Separating

A single blank line is inserted in the following places by the UIM:

- After the second title line of every page
- After list column and group headings
- After data column headings
- Before the trailer line

Fonts and Highlighting

Only printer files created without using a DDS source may be used by the UIM. No special fonts are supported. The H1 through H4 heading tags are printed as normal text with no highlighting or underlining. H1 headings are centered.

The HP0 through HP9 highlighting phrase tags are not allowed in print panel definitions. Highlighting, underline, and double strike are not supported. (A **double strike** is when the printer prints a letter and then prints the same character or another character in the same position.)

Printing the Trailer

The one-line trailer is specified using the print trailer message (PRTTRAIL) tag in a print head panel. This is a one-line statement such as: `'E N D   O F   L I S T I N G'`. It is printed at the end of an application's printout when the Remove Print Application (QUIRMVPA) or Close Application (QUICLOA) API is called.

Defining Prolog Areas

A print head panel (PRTHEAD) is used to define the prolog and header sections. The print head panel can contain only information and data areas. List areas are not allowed.

With TYPE=PROLOG specified, the prolog section consists of one or more data areas, one or more information areas, or a combination of both. The prolog section is printed only from the first print head panel that is printed for each open printer file. An attempt to print a print head panel containing a prolog section causes an escape message if the print head panel is not the first one printed for an open printer file.

Defining Header Areas

With TYPE=NORMAL specified, the header section is made up of one or more data areas, one or more information areas, or a combination of both. The total depth of the information and data areas in the header section must be less than or equal to six lines.

The minimum page length is 18 lines. To ensure that there is room for data on each page, with the possible exception of the first, the UIM limits the combined total depth of the prolog and header sections to 14 lines. If these maximums are not followed, a compile-time error message occurs.

A title line must be printed at the top of every page, so the UIM requires that a print head panel be printed before any other print panels. The Print Panel (QUIPRTP) API needs to be called only once for the print head panel for each printer file. After that, the same header information is printed on each page of the printer file. To change the header information, call the QUIPRTP API again for the same or different print head panel.

Using the Page-Eject Function During Printing

An automatic page eject occurs when a print head panel is printed after printing a print panel. However, even if eject is specified on the print panel (PRTPNL) tag, when the next print panel is printed, it does not cause a second page eject.

When a print head panel is processed by the Print Panel (QUIPRTP) API, it is not actually printed until another QUIPRTP API is called for a print panel. Therefore, if the QUIPRTP API is called for five different print head panels before the QUIPRTP API is called for a print panel, only the last print head panel affects the listing.

Sharing and Overriding Printer Files

The UIM allows multiple UIM applications to print to the same printer file by opening the printer file for sharing. However, the UIM does not share information between multiple UIM applications.

If multiple UIM applications are printing to the same printer file, it is possible to have more than one prolog section per printer file, because a prolog section, if there is one, is printed once per open application.

Care must be taken when working with a shared file to avoid inadvertently sharing with a higher or previous program in the program stack. Sharing a printer file between a UIM application and an application using a DDS record format file does not work. The shared file is closed when the last application sharing the shared file uses a Close Application (QUICLOA) or Remove Print Application (QUIRMVPA) API.

The printer file is opened to allow most overrides.

Printing Double-Byte Character Set (DBCS) Considerations

The UIM checks the `TXTMODE` attribute of the panel group (PNLGRP) tag to determine if double-byte data might be printed. If it might be printed, the UIM opens the printer file to contain DBCS data. For more information about DBCS, see Double-byte character set support in the iSeries™ Information Center.

Commonly Asked UIM Questions

Here are some commonly asked questions which relate to exit programs and call programs:

- What are exit programs and call programs?
  - Exit programs—UIM provides the ability for the user to define programs that are called after normal processing has completed in those cases where further processing on the data is required. For example, a general panel exit is allowed to do more specific validity checking on the panel than what UIM functions will handle.

- Call programs—Call programs are programs that are written by the user to be called by UIM for such items as processing a function key, a pull-down option, or list option. This processing may include refreshing a list panel, calling another program to display a more detailed panel or to update a file with the information entered, and so on.

- What is the difference between an exit program and a call program?
  - An exit program is called after normal processing is complete and a call program is called as part of the normal processing. Normal processing refers to processing items such as ACTION, ENTER, and PROMPT actions on a LISTACT tag.

Part 4. Programming Help Displays

Chapter 18. Making Online Help Information Accessible for Your Display File 369
Enabling the Help Key 370
Choosing between Panel Groups and Records for Help 370
Defining Which Areas of Your Display Need Online Help Information 371
Specifying Panel Groups for Help in Your Display File 373
Defining Panel Groups with Option Indicators 375
Copying QUSRTOOL Examples That Specify Help Using Panel Groups 376
Specifying Records in Your Display File 376
Defining Records with Option Indicators . . . 377
Entering the Records That Contain the Help Information 378
Using Records and Documents for Help in the Same Display File 378
Understanding the Restrictions on Records . . 379
Paging between Help Displays That Use Records 379
Understanding How the System Pages Help Displays 379
Displaying Secondary Online Help Information 381
Returning Control to Your Program after Pressing the Help Key 383
Returning Control to Your Program after Showing the Help Display 383
Returning Control to Your Program without Showing the Help Display 384

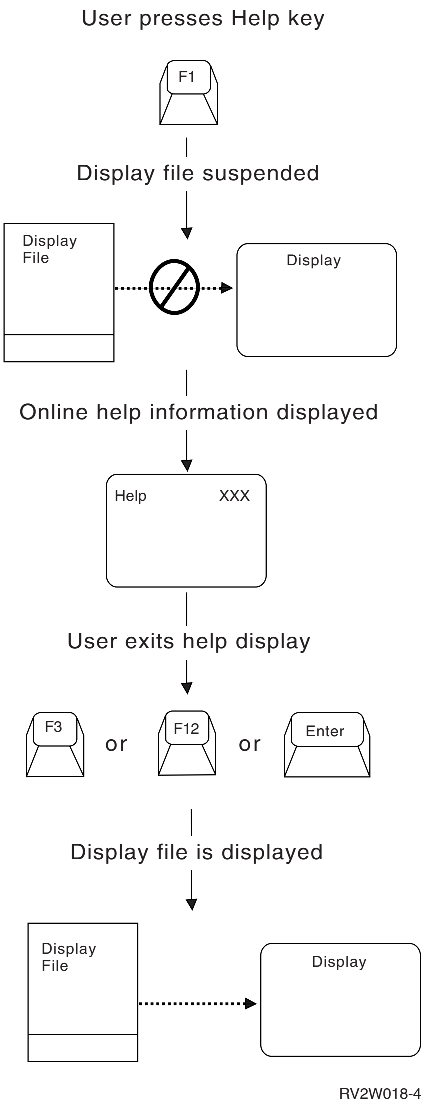
Chapter 19. Making Online Help Accessible for Your Panel Group 385
Definitions and Explanations 385
Giving Help Panel Groups Access to Index Search 387
Giving Help Panel Groups Access to A User-Defined Panel Group 387
Removing Access to F18=More Indexes. . . . 388
Help in a List Area 388
Coding Help 389
Help in a Menu Area. 389
Coding Help 390
Help in a Data Area 391
Coding Help 392
Help in a Menu Bar Area 394
Coding Help 395
Help in a Function Key Area 396
Coding Help 397

Chapter 20. Defining Online Help Information 399
Defining Online Help Information in a Panel Group 399
Entering the UIM Source for a Panel Group for Help 399
Organizing a Panel Group with Help Modules 399
Using the Information in a Help Module More Than Once 400
Using a Help Module Contained in a Different Help Panel Group 400
Emphasizing and Formatting the Text within a Help Module 401
Adding Index Search Tags to a Help Panel Group 403
Linking Help Modules 408
Creating and Deleting Panel Groups. 410
Assigning Panel Groups as Help for Commands 411
Using Panel Groups in a Search Index 411
Creating a Search Index 411
Adding Entries to a Search Index. 411
Removing Entries from a Search Index . . . 412
Deleting a Search Index 412
Copying QUSRTOOL Examples That Define Help in a Panel Group 412
Defining Online Help Information in a DDS Record 412

Chapter 18. Making Online Help Information Accessible for Your Display File

Application help is a function provided by the system that lets you define online help information for the records in a display file and then presents that help to the user when the Help key is pressed.

The following illustration shows what happens when the Help key is pressed and online help information is available and accessible:



Note: Caution is advised when using application help with multiple display stations acquired to the file because all display stations wait for the completion of the application help display.

To provide online help information for your display, you need to do the following:

- Add to your display file the necessary DDS keywords to make online help information accessible for your display. This chapter provides instructions for adding these keywords.

- Define the help information that the user sees using either a panel group, document, or record. Chapter 20, “Defining Online Help Information,” on page 399 provides instructions for defining the help information.

Consider the following points when you create help text:

- Some tags truncate a line of text if the text does not fit into the display area.
- Other tags (for example, XMP) wrap the text to the next line.
- The size of the display (24 x 80 or 27 x 132) determines how many columns are allowed.

When you nest tags, you must also factor in the width of the nested lines. For example, if you define an unordered list and include an example in one of the list items, the text for the example is indented two bytes to the right of the text in the list item.

Information about DDS keywords

For more information about the DDS keywords described in this chapter, see the **DDS** topic in the iSeries Information Center.

Enabling the Help Key

If online help information is available for your display, you must enable the Help key on the keyboard. To enable the Help key, specify the HELP keyword at the file or record level of your DDS source.

Choosing between Panel Groups and Records for Help

You can define online help information by using one of the following:

Table 34. Different Ways to Define Online Help Information

Method	Description
Panel group	Objects into which user interface manager (UIM) source is entered
Records	A set of DDS keywords contained in a source file member

Each method has characteristics that can help you choose the method that will work best for you. The following table lists these characteristics and the method or methods that they apply to:

Table 35. Characteristics of Different Methods of Online Help Information

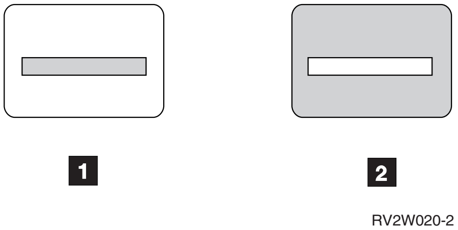
Characteristic	Panel Groups	Records
Accessible through H specifications for DDS-described displays	X	X
Allows cursor-sensitive help	X	X
Allows extended help	X	X
Index search function available	X	
Windows used	X	
Hypertext linking available	X	
Used by system displays	X	
Can be used for command help	X	
Word processing functions, such as spell checking, available	X (See note.)	
Bookmark and exit function available (allows you to return to the same location in the help the next time you press the Help key)		
Allows printing of help information	X	X

Table 35. Characteristics of Different Methods of Online Help Information (continued)

Characteristic	Panel Groups	Records
Index for displayed information available		
Table of contents for displayed information available		
Can be placed in same member as display source		X
Same source type as display source		X
QUSRTOOL example available	X	
Option indicators allowed	X	X
Supports DBCS online help information	X	X
Right-to-left and left-to-right orientation of text	X	
Supported for acquired devices		X

Defining Which Areas of Your Display Need Online Help Information

You can define the display file so that you supply the online help information for all or part of a display, as represented by the shaded areas in the following figure:



1 **Contextual help** Information for rectangular areas that are defined by a record or part of a record format. Contextual information describes one or more fields on a display.

2 **Extended help** All information for the entire display, beginning with the information about the purpose and use of the display and followed by contextual help for each field. Extended help is displayed when you press the Help key on an area of the display that does not have contextual help defined for it. If your online help information is defined using panel groups, you can also reach extended help from contextual help by pressing F2.

The cursor location is determined when the Help key is pressed. If the Help key is enabled and the cursor is in any active help area, the online help information associated with that help area is displayed.

Each help area on a display is defined in the DDS source using the following:

H specification
Defines help for the containing record. An H in column 17 of the DDS source indicates a help specification level. The help specifications are defined before the first field in the record.

Help Area (HLPARA) keyword
Defines a help area by giving the upper-left row column and lower-right row column of the rectangular area. These coordinates must be located in the screen area, but are not required to be in the record area.

If *RCD is specified for the HLPARA keyword, the help is associated with the entire record area, which includes all columns in the lines occupied by the record. When the cursor is located in the record area and the Help key is pressed, the online help information from the record on the HLPRCD keyword is displayed.

Help (HLPxxx) keyword

Defines whether a panel group (HLPPNLGRP), document (HLPDOC), or record format (HLPRCD) contains the actual online help information that you see when you press the Help key.

The following illustration shows how H specifications are entered in your DDS source:

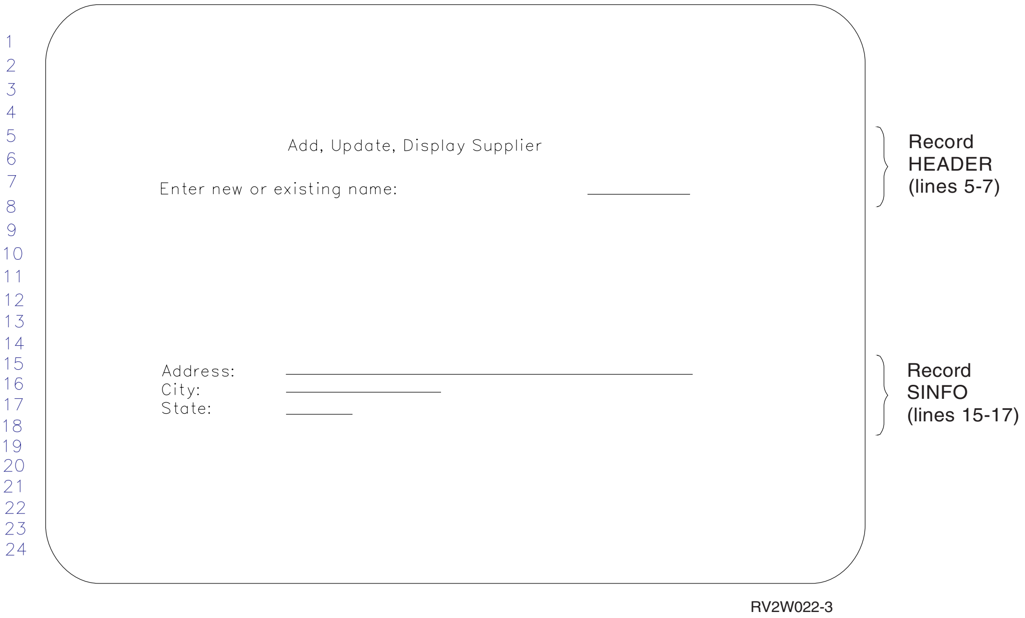
```
|...+....1....+....2....+....3....+....4....+....5....+....6....+....7...
                H                            HLPARA(*RCD)
                                             HLPxxx(required-variables)
```

Figure 115. Sample H Specification in DDS Source

This sample DDS defines help for a record. The required variables for the help keyword depend on whether HLPxx is defined as a HLPPNLGRP, HLPDOC, or HLPRCD keyword. Only one help keyword can be defined for the file, but you can specify one or more H specifications for a record.

The order of H specifications is important because the first match found is selected. Therefore, the more specific HLPARA locations should be listed first.

The following sample display is defined with two record formats, HEADER and SINFO:


1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
Add, Update, Display Supplier
Enter new or existing name:
Address:
City:
State:
Record HEADER (lines 5-7)
Record SINFO (lines 15-17)
RV2W022-3

The following table shows one way to define the online help information for the sample display:

Table 36. Help for Sample Display

Location on Display	Location in DDS Source	Description of Help Provided
Lines 5-7	Entire record format HEADER	Help is specified for the whole record
Line 15	First line in record format SINFO	Help is specified for the line only
Line 16	Second line in record format SINFO	Help is specified for the line only
Line 17	Third line in record format SINFO	Help is specified for the line only
Remaining lines	Areas in display file that are not defined with record formats	Help is specified for the whole file

Depending on the help method you plan to use, continue with any of the following to define H specifications in your DDS source:

Panel groups (UIM)
"Specifying Panel Groups for Help in Your Display File"

Records (DDS)
"Specifying Records in Your Display File" on page 376

To compare and contrast the different ways to create online help information, see "Choosing between Panel Groups and Records for Help" on page 370.

Specifying Panel Groups for Help in Your Display File

The HLPPNLGRP keyword in DDS identifies the panel group that contains online help information for a display. The HLPPNLGRP keyword may be specified in an H specification or at the file level. To use the HLPPNLGRP keyword, you need to know the name of the help module and the name of the panel group and library that contains that help module.

Several other DDS functions are available for use with panel groups. The functions and their associated DDS keywords are described in the following table:

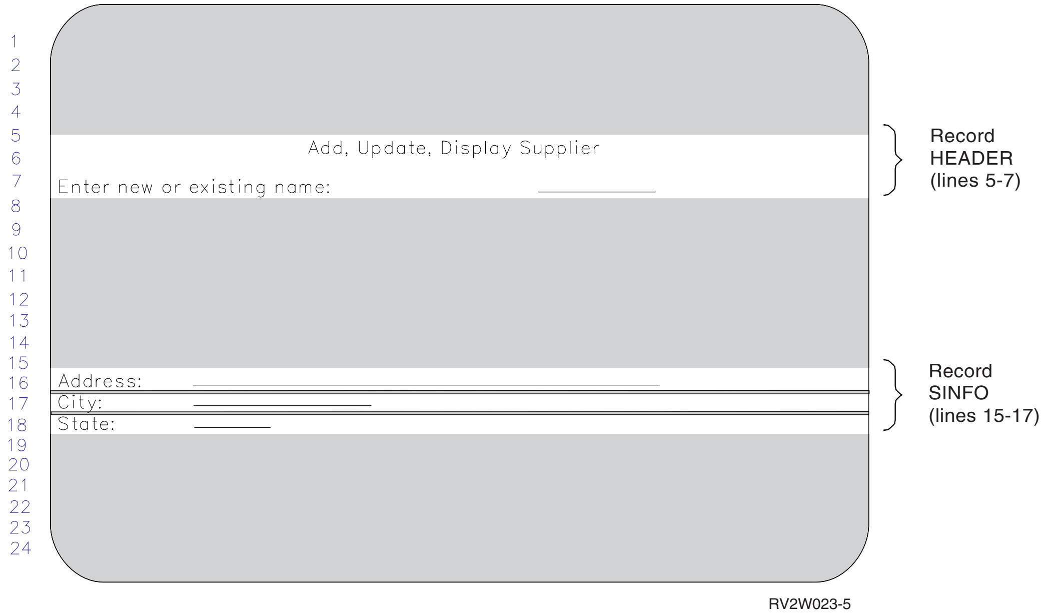
Table 37. Other DDS Keywords for UIM Help

Function	DDS keyword	Description
Defining the name of the help screen	HLPTITLE	The text to be displayed on the first line of the help display is defined with the file- or record-level DDS keyword HLPTITLE (Help Title). This text should be the name of the display that is displayed when the Help key is pressed. The HLPTITLE keyword must be used in the display file. This keyword is used only on full-screen displays of help when no help title is specified in the help source.
Indicating full-screen online help information	HLPFULL	Using the DDS file-level HLPFULL (Help Full) keyword, the UIM-defined online help information for the application is displayed in a full-screen replacement display rather than in a window. When the HLPFULL keyword is not specified, the help is displayed in a window unless the user's profile specifies otherwise.

Table 37. Other DDS Keywords for UIM Help (continued)

Function	DDS keyword	Description
Excluding panel group help as secondary help information	HLPEXCLD	The HLPEXCLD keyword excludes a help panel group with a duplicate name from being displayed within extended help. To do this, place HLPEXCLD on all but the first H specification that names an identical help panel group. HLPEXCLD indicates that the information associated with the H specification is not displayed as part of the extended help. When the HLPEXCLD keyword is not specified, extended help consists of the information associated with both the file-level HLPPNLGRP keyword (if any) and the HLPPNLGRP keywords on all active H specifications. At least one help panel group name must not specify the HLPEXCLD keyword.
Enabling the Index Search function	HLPSCHIDX	The Index Search function is enabled, and the search index object used for the Index Search function is specified by the file-level DDS keyword HLPSCHIDX (Help Search Index).

Using the sample display from "Defining Which Areas of Your Display Need Online Help Information" on page 371, the following online help information is defined for each help area:


1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
Add, Update, Display Supplier
Enter new or existing name:
Address:
City:
State:
Record HEADER (lines 5-7)
Record SINFO (lines 15-17)

RV2W023-5

Table 38. Help for Sample Display Using Panel Groups

Location on Display	Location in DDS Source	Name of Help Module in Panel Group SMPPNL in Library SMPLIB
Lines 5-7	Entire record format HEADER	HLPCMPY
Line 15	First line in record format SINFO	HLPADDR

Table 38. Help for Sample Display Using Panel Groups (continued)

Location on Display	Location in DDS Source	Name of Help Module in Panel Group SMPPNL in Library SMPLIB
Line 16	Second line in record format SINFO	HLPCITY
Line 17	Third line in record format SINFO	HLPST
All shaded areas	Areas in display file that are not defined with records	SUPHELP

The following DDS source shows how the help areas are defined for the sample display using panel groups:

```
|...+....1....+....2....+....3....+....4....+....5....+....6....+....7...
                                            HELP
                                            HLPPNLGRP(SUPHELP SMPLIB/SMPPNL)
                                            HLPTITLE('ADD,UPDATE,DISPLAY SUPPLIER')
                R HEADER                    OVERLAY
                H                           HLPARA(*RCD)
                                            HLPPNLGRP(HLPCMPY SMPLIB/SMPPNL)
                                        5 25'ADD, UPDATE, DISPLAY SUPPLIER'
                                        7 10'ENTER NEW OR EXISTING NAME:'
                  CONAME        10A  I  7 47
                R SINFO                     OVERLAY PROTECT
                H                           HLPARA(15 1 15 79)
                                            HLPPNLGRP(HLPADDR SMPLIB/SMPPNL)
                H                           HLPARA(16 1 16 79)
                                            HLPPNLGRP(HLPCITY SMPLIB/SMPPNL)
                H                           HLPARA(17 1 17 79)
                                            HLPPNLGRP(HLPST SMPLIB/SMPPNL)
                                       15 10'ADDRESS:'
                  ADDR          30A  B 15 32
                                       16 10'CITY:'
                  CITY          10A  B 16 32
                                       17 10'STATE:'
                  STATE          2A  B 17 32
```

Figure 116. Sample DDS Source Showing HLPPNLGRP

The file-level help in SUPHELP provides extended help for the display (when the cursor is not located in the record area for either HEADER or SINFO) because HLPARA(*RCD) is the location specified on the H specification.

The panel groups that contain the online help information must be created by using the UIM source from a source file member. More information about creating panel groups is found in “Defining Online Help Information in a Panel Group” on page 399.

Note: HLPPNLGRP and HLPRCD are not allowed in the same display file; HLPPNLGRP and HLPDOC are also not allowed in the same display file.

Defining Panel Groups with Option Indicators

The HLPPNLGRP keyword can be specified with option indicators. In the following example, assume the SINFO record has indicator 90 on and the cursor is in the help area defined for the H specification. When the Help key is pressed, the panel group HELP1 is displayed. If indicator 90 is off, panel group HELP2 is displayed.

The DDS for this file is:

```
|...+....1....+....2....+....3....+....4....+....5....+....6....+....7...
                                            HELP
                R HEADER                    OVERLAY
                                        5 25'ADD, UPDATE, DISPLAY SUPPLIER'
                                        7 10'ENTER NEW OR EXISTING NAME'
                  CONAME        10A  I  7 47
                R SINFO                     OVERLAY PROTECT
                H                           HLPARA(15 9 15 61)
        90                                  HLPPNLGRP(HELP1 SMPLIB/SMPPNL)
                H                           HLPARA(15 9 15 61)
       N90                                  HLPPNLGRP(HELP2 SMPLIB/SMPPNL)
                                       15 10'ADDRESS'
                  ADDR          30A  B 15 32
```

Figure 117. Sample DDS Source Showing HLPPNLGRP and Option Indicators

Copying QUSRTOOL Examples That Specify Help Using Panel Groups

The QUSRTOOL library provides four sample DDS-described displays that access online help information using panel groups. You can copy the source for these sample displays into a library of your choosing and then tailor them for your own use. For more information about these sample displays, see “Using the Displays Example in the QUSRTOOL Library” on page 415.

Specifying Records in Your Display File

The source of the online help information that is identified by a help record in DDS source is specified with the help-specification or file-level DDS keyword HLPRCD (Help Record).

HLPRCD and HLPPNLGRP are not allowed in the same display file.

Using the sample display from “Defining Which Areas of Your Display Need Online Help Information” on page 371, the following online help information is defined for each help area:

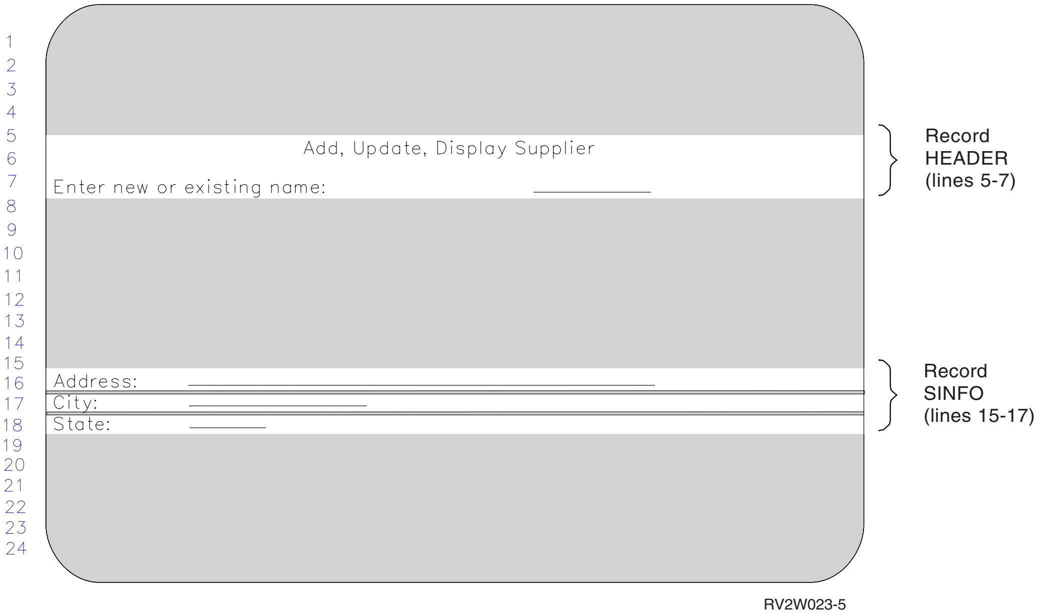

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
Add, Update, Display Supplier
Enter new or existing name:
Address:
City:
State:
Record HEADER (lines 5-7)
Record SINFO (lines 15-17)
RV2W023-5

Table 39. Help for Sample Display Using HLPRCD

Location on Display	Location in DDS Source	Name of Record Format that Contains Help
Lines 5-7	Entire record format HEADER	Record format HLPCMPY
Line 15	First line in record format SINFO	Record format HLPADDR
Line 16	Second line in record format SINFO	Record format HLPCITY
Line 17	Third line in record format SINFO	Record format HLPST
All shaded areas	Areas in display file that are not defined with help record	Record format SUPHELP

The following DDS source shows how the help areas are defined for the sample display using help records:

```
|...+....1....+....2....+....3....+....4....+....5....+....6....+....7...
                                            HELP
                                            HLPRCD(SUPHELP)
                R HEADER                    OVERLAY
                H                           HLPARA(*RCD)
                                            HLPRCD(HLPCMPY)
                                        5 25'ADD, UPDATE, DISPLAY SUPPLIER'
                                        7 10'ENTER NEW OR EXISTING NAME:'
                  CONAME        10A  I  7 47
                R SINFO                     OVERLAY PROTECT
                H                           HLPARA(15 1 15 61)
                                            HLPRCD(HLPADDR)
                H                           HLPARA(16 1 16 41)
                                            HLPRCD(HLPCITY)
                H                           HLPARA(17 1 17 33)
                                            HLPRCD(HLPST)
                                       15 10'ADDRESS:'
                  ADDR          30A  B 15 32
                                       16 10'CITY:'
                  CITY          10A  B 16 32
                                       17 10'STATE:'
                  STATE          2A  B 17 32
```

Figure 118. Sample DDS Source Showing HLPRCD

The file level help is used to provide general help for the display when the cursor is not located in any of the defined help areas for either HEADER or SINFO.

You are not required to define each help area with a different record. This means that the same record may be used to define one or more help areas on a display, including the help area that defines the entire display.

Defining Records with Option Indicators

The HLPRCD keyword can be specified with option indicators. In the following example, assume the SINFO record is put with indicator 90 on and the cursor is in the help area defined for the H specification. When the Help key is pressed, the record HELP#1 is displayed. If indicator 90 is off, the record HELP#2 is displayed.

The DDS for this file is:

```
|...+....1....+....2....+....3....+....4....+....5....+....6....+....7...
                                            HELP
                R HEADER                    OVERLAY
                                        5 25'ADD, UPDATE, DISPLAY SUPPLIER'
                                        7 10'ENTER NEW OR EXISTING NAME'
                  CONAME        10A  I  7 47
                R SINFO                     OVERLAY PROTECT
                H                           HLPARA(15 9 15 61)
       90                                   HLPRCD(HELP#1)
                H                           HLPARA(15 9 15 61)
      N90                                   HLPRCD(HELP#2)
                                       15 10'ADDRESS'
                  ADDR          30A  B 15 32
```

Figure 119. Sample DDS Source Showing HLPRCD and Option Indicators

Entering the Records That Contain the Help Information

The records that give the actual online help information may be included in the same file as the DDS source for the application display. The records may also be contained in a different file. If the records are contained in a different file, the file and the library that contains the file must be identified on the HLPRCD keyword.

Regardless of where the information is contained, the DDS source for the previous example looks like the following:

```
|...+....1....+....2....+....3....+....4....+....5....+....6....+....7...
                R SUPHELP
                                        2  5'TO ADD, UPDATE, OR DISPLAY -
                                            THE SUPPLIER NAME AND -
                                            ADDRESS, ENTER THE SUPPLIER -
                                            NAME.  ITS CURRENT ADDRESS, -
                                            IF ANY, WILL BE SHOWN AND MAY -
                                            BE UPDATED.'
                R HLPCMPY
                                        2  5'ENTER THE FIRST 10 -
                                            CHARACTERS OF THE COMPANY'
                R HLPADDR
                                        2  5'ENTER THE NEW OR CHANGED -
                                            ADDRESS.'
                R HLPCITY
                                        2  5'ENTER THE FIRST 10 -
                                            CHARACTERS OF THE CITY'
                R HLPST
                                        2  5'ENTER THE 2 CHARACTER ABBR -
                                            FOR THE STATE.'
```

Figure 120. Sample DDS Source with HLPRCD

Using Records and Documents for Help in the Same Display File

HLPRCD and HLPDOC are allowed in the same display file; however, the following considerations apply:

- When the Help key is pressed with the cursor location in a help area described by a document, that document is the only help displayed when the roll keys are used. No other document or help records are displayed.
- When the Help key is pressed with the cursor location in a help area described by a record, the normal sequencing used for records is followed with documents being ignored.

- The HLPDOC and HLPRCD keywords cannot both be specified at the file level or on the same H specification.
- The HLPBDY keyword cannot be specified with the HLPDOC keyword. For more information about the HLPBDY keyword, see “Displaying Secondary Online Help Information” on page 381.

Understanding the Restrictions on Records

The following restrictions apply when using the record form of application help:

- The application program does not control the displaying of the records. When the Help key is pressed, the system controls the displaying of help. Because the system does not control the buffers or hidden message line areas of the application program, the following takes place:
  - Output fields in the record formats are displayed as blanks.
  - The ALIAS keyword is allowed but ignored.
  - CSRLOC and MSGID are processed by the system, but the hidden and program-to-system fields are not passed from the application program.
- No input is returned to the user program:
  - Input-capable fields on a record are displayed with underlines; however, no input is returned to the application program. Any input typed in an input field is lost when application help is finished.
  - Response indicators are not returned.
- Help records can contain H specifications, but they are ignored.
- Option indicators are assumed to be off.
- Screen size conditioning can be used.
- Only the following keywords are active when specified for records:

  COLOR
  DATE
  DFT
  DSPATR
  DSPSIZ
  MSGCON
  MSGLOC
  TEXT
  SLNO (constant value). If SLNO(*VAR) is specified, 1 is used for the starting line number.
  SYSNAME
  TIME
  USER

  All other display file keywords, though allowed, do not make sense for records and may or may not be processed while displaying the help.

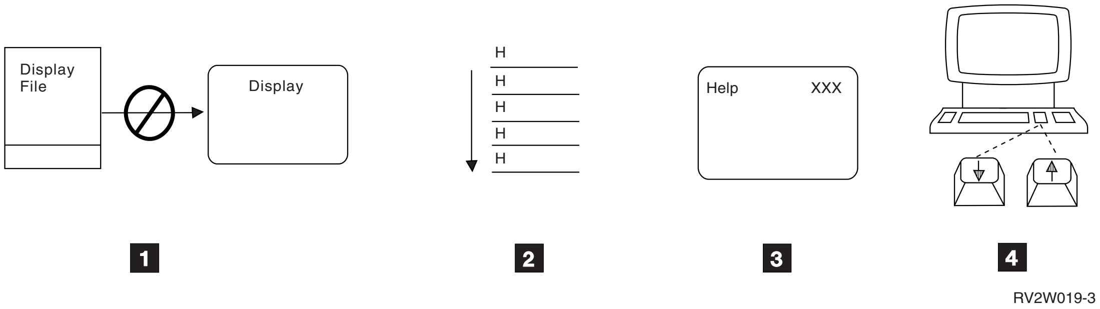
Paging between Help Displays That Use Records

If you use records to define help information, the first display that appears when the Help key is pressed is called **primary help**. When a Roll Up or Roll Down key is pressed on a primary help display, **secondary help** is shown.

Understanding How the System Pages Help Displays

The system maintains a list of all active H specifications from record formats that are on the display. This list of H specifications is called the **help list**. The H specifications on the front of the help list are the first to be searched when the user presses the Help key.

Pressing the Help key in a help area on the display does the following:



1 Suspends the display file.

2 Searches the list of active help areas to find the first one in the list that contains the cursor location when the Help key is pressed.

3 Opens the display file that contains the help record and displays the information.

4 Allows the user to roll forward or backward through the online help information.

The order and content of the H specifications in the list are determined by the following:

- When a record format is added to the display, the H specifications for that record format are placed on the front of the list.
- If the record format contains more than one H specification, they are added to the help list in the order in which they are defined in the display file.
- The help list is cleared either when a record format is written that clears the display, or when one is written that has the HLPCLR keyword enabled.
- If a record has the OVERLAY keyword in effect, has H specifications, or completely or partially overlaps another record already on the display, the help list is updated in different ways. The following table describes the type of help-list updating for records for each combination of the three factors:

Overlay	H Specifications	Overlaps	Help List Update
No	No	N/A	Help list is cleared.
No	Yes	N/A	Help list is cleared and H specifications for the record are added to the help list.
Yes	No	No	Help list is not changed.
Yes	No	Yes	H specifications are removed from the help list if they are within the boundaries of the record being written.
Yes	Yes	No	H specifications for the record are added to the help list. H specifications are removed from the help list if they are within the boundaries of any H specification in the record being added.
Yes	Yes	Yes	H specifications are removed from the help list if they are within the boundaries of the record being written. Then the H specifications for the record are added to the list. Finally, H specifications are removed from the help list if they are within the boundaries of any H specification in the record being added.

Note: An H specification with *NONE specified for the help area is removed from the help list when the first H specification with a help area defined above it is removed. If a help specification with *NONE specified for its help area is the first help specification, then it is only removed when the help list is cleared or when a help specification with a help area is placed above this help specification. This H specification is removed when the one above it is removed.

Displaying Secondary Online Help Information

The record formats that are displayed as secondary help come from the same help group, or if none is available there, from the same help sublist as the record format that is currently displayed.

A **help group** is defined with the HLPSEQ keyword and consists of those record formats that have the same group name specified. The HLPSEQ keyword allows you to specify the help group name and the help sequencing number. The help sequencing number specifies the order in which the help will be displayed. If two record formats that are in different display files happen to have the same group name, they are still considered to be in separate help groups. Record formats that do not have a HLPSEQ keyword specified are considered to be groups of one.

A **help sublist** contains all of those H specifications defined between help boundaries. The HLPBDY keyword partitions the help list into sublists by defining **help boundaries**. (The H specification that has the HLPBDY keyword coded is considered to be before the boundary.) Sublists are important when using the roll keys to look at more online help information.

Determining the Sequence of Secondary Help: Depending on whether you are using the Roll Up or Roll Down key, the system selects secondary help as follows:

1. A help record format that is in the same help group as the current help record format and has the next *highest* help sequencing number (if the Roll Up key is pressed) or next *lowest* help sequencing number (if the Roll Down key is pressed) is selected.
2. If the current help record format already has the *highest* help sequencing number (if the Roll Up key is pressed) or the *lowest* help sequencing number (if the Roll Down key is pressed) in the group, the help sublist is searched for the *next* H specification that does not refer to the same help group as the currently displayed record format.

   Notes:

   a. To prevent including unexpected help, it is recommended that the HLPBDY keyword be specified on the last H specification in each record of the application display file. This defines one sublist for each record that has help. However, if multiple records are on the screen, this may not be desirable. In this case, the HLPBDY keyword should be in effect only on the last record put to the screen.

   b. Because this second method works only when a sublist can be identified, it is not used if the primary help was the default record format for the file (for example, if it was selected from the file-level HLPRCD keyword).
3. Searching in the sublist continues until the boundary of the sublist is reached. The search then wraps to the other end of the sublist and continues until the current H specification being displayed is reached.
4. If no H specification that has a satisfactory HLPRCD is found, the record format in the help display file that has the same help group name and the *lowest* help sequence number (if the Roll Up key is pressed) or the *highest* help sequence number (if the Roll Down key is pressed) is selected. This method always finds a match because the current help format always meets this criteria if no other format does.

In the following example, the fields and HLPARA keywords are not specified because neither has an effect on the order of secondary information:

```
|...+....1....+....2....+....3....+....4....+....5....+....6....+....7...
                                            HELP
                                            HLPRCD(HELP11)
                A RECORD1
                H                           HLPRCD(HELP1)
                H                           HLPRCD(HELPSCR1)
                                            HLPBDY
                H                           HLPRCD(HELP2SRC)
                                            HLPBDY
```

Figure 121. Sample DDS Source to Show Secondary Help

The sample file HELPFILE contains the following record formats with HLPSEQ keywords coded as shown:

Record	Keyword Specification
HELP	HLPSEQ(GROUPA 1)
HELP1	HLPSEQ(GROUPA 2)
HELP11	HLPSEQ(GROUPA 3)
HELPSCR1	HLPSEQ(GROUPB 1)
HELPSCR2	HLPSEQ(GROUPB 2)
HELP2SRC	

Note that the help record formats HELP and HELPSCR2 are not referred to in the application display file. Because they are not referred to in this way, they are not primary help, but they are displayed as secondary help as follows:

```
Sequence 1:
  Primary help format       HELP11
  Press Roll Down           HELP1
  Press Roll Down           HELP
  Press Roll Down           HELP11
```

In this first sequence, HELP1 is shown when the Roll Down key is pressed because this is the previous help record format in GROUPA. Similarly, HELP is shown when Roll Down is pressed the second time. When Roll Down is pressed the third time, the end of the help group is reached. HELP11 is then selected again because it is the last help record format in the original help group.

```
Sequence 2:
    Primary help format     HELPSCR1
    Press Roll Up           HELPSCR2
    Press Roll Up           HELP1
    Press Roll Down         HELP
```

The second sequence starts with HELPSCR1 being displayed as the primary help record format. Pressing Roll Up causes HELPSCR2 to be displayed because it is the next help record format in GROUPB. Rolling up again runs off the end of the group and HELP1 is found because it is the next help record format found, after wrapping, in the sublist that contains HELPSCR1. Pressing Roll Down now causes HELP to be displayed because it is the previous entry in GROUPA.

```
Sequence 3:
    Primary help format        HELP2SRC
    Press Roll Up or Down      HELP2SRC
```

Because HELP2SRC is not in a help group and is the only one in its sublist, HELP2SRC is to be shown when rolling in either direction.

Understanding the Restrictions of Records for Secondary Help: The following restrictions apply when using the record form of application help:

- Application help controls all function keys when a help record is displayed:

- – The Roll Up key is enabled and causes the next record to be displayed according to the rules for displaying secondary help.
  - – The Roll Down key is enabled and causes the previous record to be displayed according to the rules for secondary help.
  - – The Enter key is enabled and causes a return to the application program. Any CAnn or CFnn key enabled for the record will also cause a return to the application program. All other function keys, including the Help key, are ignored.
- Records with the USRDFN, SFL, and SFLCTL keywords may not be used as records. When a display file is created, a diagnostic is issued if the HLPSEQ keyword is found on a record with one of these keywords. When the application is running, error reset message CPD4050 is issued if a record with one of these keywords is used as help. The help record is not displayed.

  The KEEP and ASSUME keywords should be avoided on records because they cause results that cannot be predicted.

Returning Control to Your Program after Pressing the Help Key

Depending on how you code your DDS, you can return control to your program in one of two ways after you press the Help key:

- You can display online help information and then return control to the program.
- You can return control to the program immediately without displaying the online help information.

Returning Control to Your Program after Showing the Help Display

Use the help command key (HLPCMDKEY) keyword to return control to your application program from the application help record format after a command attention (CAnn) or command function (CFnn) key has been pressed. A CFnn key returns data to the application program, while a CAnn key does not return data to the application program.

The command key must apply to both the application record format and the application-help record format. If a CAnn or CFnn key does not apply to the application-help record format, the HLPCMDKEY keyword is ignored.

The following DDS source, which uses help records, shows how the HLPCMDKEY keyword is specified:

```
|...+....1....+....2....+....3....+....4....+....5....+....6....+....7...
                R APPRCD                    CA01
                                            CA03
                                            CF04
                                            HELP
                H                           HLPRCD(HELPRCD)
                                            HLPARA(1 1 24 80)
                                       8  2'THIS IS THE APPLICATION'
                                       9  2'RECORD FORMAT'
                  INPUT1        10  B  12 10
                  INPUT2        10  B  13 10
                  INPUT3        10  B  14 20
      *
                R HELPRCD                   HLPCMDKEY
                                            CA01
                                            CF03
                                       8  2'SPECIFY COMPANY NAME'
                                       9  2'SPECIFY STREET'
                                      10  2'SPECIFY CITY, STATE, ZIP'
```

Figure 122. Sample DDS Source to Show HLPCMDKEY

If the user is on the application help display that uses the previous data description specifications, the following happens when the various keys are pressed:

- The CF04 key, which is specified only on the application record format, acts the same as the Enter key.
- The CMD3 key acts the same as the Enter key. Corresponding CAnn or CFnn keys must be specified on both the application-help record format and application record format for control to return to the program.
- The CA01 key returns control to the application program.

In the next example, response indicators are used. When a response indicator is specified on a CAnn or CFnn key on the application record format (for example, CF12(12)), the response indicator is returned after the application-help record format is displayed. When a response indicator is specified on a CAnn or CFnn key on the application-help record format (for example, CF01(11)), the response indicator is ignored.

```
|...+....1....+....2....+....3....+....4....+....5....+....6....+....7...
                R APPRCD                    CA01
                                            CA03
                                            CF12(12)
                                            HELP
                H                           HLPRCD(HELPRCD)
                                            HLPARA(1 1 24 80)
                                        8  2'THIS IS THE APPLICATION'
                                        9  2'RECORD FORMAT'
                  INPUT1        10  B  12 10
                  INPUT2        10  B  13 10
                  INPUT3        10  B  14 20
      *
                R HELPRCD                   HLPCMDKEY
                                            CF01(11)
                                            CF12
                                        8  2'SPECIFY COMPANY NAME'
                                        9  2'SPECIFY STREET'
                                       10  2'SPECIFY CITY, STATE, ZIP'
```

Figure 123. Sample DDS Source to Show HLPCMDKEY and Response Indicators

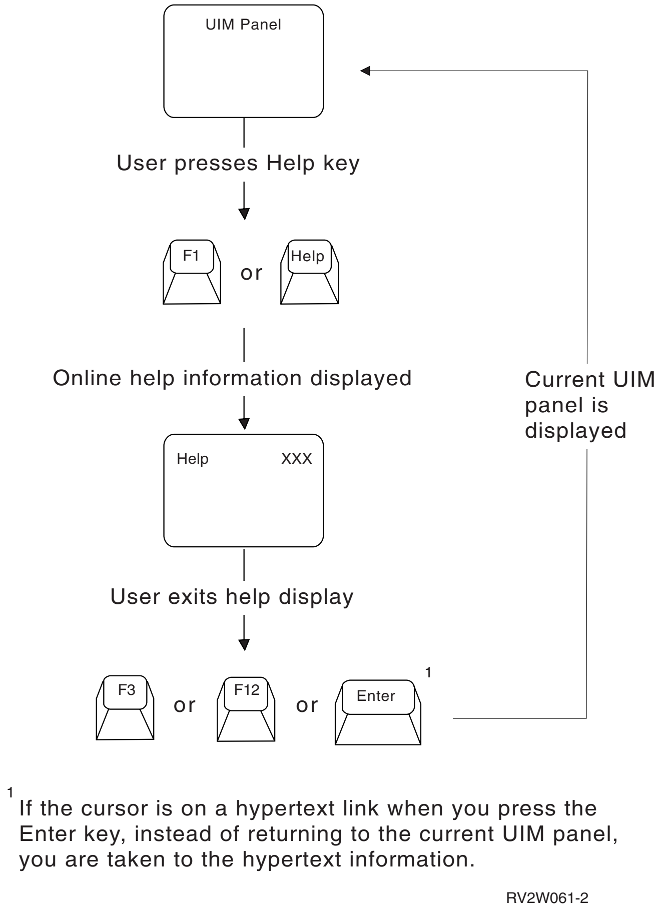
Returning Control to Your Program without Showing the Help Display

Use the HLPRTN keyword to immediately return control to your program (instead of displaying help) when the Help key is pressed.

Chapter 19. Making Online Help Accessible for Your Panel Group

The Help key on most keyboards is either a key specifically labeled Help, or it is a key defined to work as a Help key and labeled with another name. This varies among keyboards, but the F1 key is commonly defined as an alternate for the Help key.

When the user presses the Help key, a sequence of events occur, as illustrated by the following diagram:


UIM Panel
User presses Help key
F1
or
Help
Online help information displayed
Current UIM panel is displayed
Help
XXX
User exits help display
F3
or
F12
or
Enter
1
1 If the cursor is on a hypertext link when you press the Enter key, instead of returning to the current UIM panel, you are taken to the hypertext information.
RV2W061-2

When the user presses the Help key, the current panel is suspended by the UIM and a help panel is displayed with information about a specific item, group of items, area on a panel, or the entire panel. When the user has read the displayed help, the user presses the F3, F12, or Enter key to return to the current panel before the Help key was pressed.

Definitions and Explanations

Help for a panel is constructed from help modules referred to in the panel definition. The UIM requires you to define online help information for panels in a panel group, and presents the help to the user when the HELP dialog command is requested.

The user requests the HELP dialog command by pressing a key assigned to it. Typically, the engraved Help key and the F1 key are assigned to the HELP dialog command. In this section, the term **Help key** refers to any key which is assigned to the HELP dialog command.

Online help information is the information displayed to the user when the Help key is pressed on the keyboard. The level of help displayed depends on the location of the cursor when the user presses the Help key.

Help is always available for the entire panel. This help is known as **extended help**, which is the general information for a panel. Extended help begins with information about the purpose and use of the panel and is followed by contextual help for each item on the panel. Extended help includes contextual help for items that do not currently show on the screen, but which can be shown by using the Page Up and Page Down keys. (**Contextual help** provides information about a single item or group of related items where the cursor is positioned when the user requests help.) An item is excluded from extended help when the item is not currently active for the panel by using the COND attribute for a specific item on the panel.

Extended help is displayed when the Help key is pressed for an area of the screen that does not have contextual help defined. It is also displayed when the user presses F2 (Extended help) while viewing contextual help.

The extended help is defined on the HELP attribute of the display panel (PANEL) tag. This attribute identifies the help module containing the beginning of the extended help for the panel.

When coding online help for a panel, the tag which actually contains the information that is displayed when the user presses the Help key is the help module (HELP) tag. For more information on the attributes of this tag, see "HELP (Help Module)" on page 529.

To code for help in a panel, the HELP attribute must be specified on the tag for which the help is provided. The name of the help module containing the help information is specified on the HELP attribute of a tag.

The help module identified on a HELP attribute exists in the same panel group or menu where that HELP attribute appears, or in another panel group. When the panel group or menu is created, the UIM compiler determines the panel group in which the UIM will find the help module when the Help key is pressed by finding a help module name that matches the name specified on the HELP attribute in one of the following places:

1. If the panel group or menu being created contains a matching name on the NAME attribute of a HELP tag, the help module is found in the same panel group or menu when the Help key is pressed.
2. If the panel group or menu being created contains a matching name on the NEWNAME attribute of an import (IMPORT) tag, the help module is found in the panel group specified on the PNLGRP attribute of the IMPORT tag.
3. If the panel group or menu being created contains an IMPORT tag with NAME='*' specified, the help module is found in the panel group specified on the PNLGRP attribute of the IMPORT tag.

If all of the above are not true, the UIM compiler reports an error message and the panel group or menu is not created.

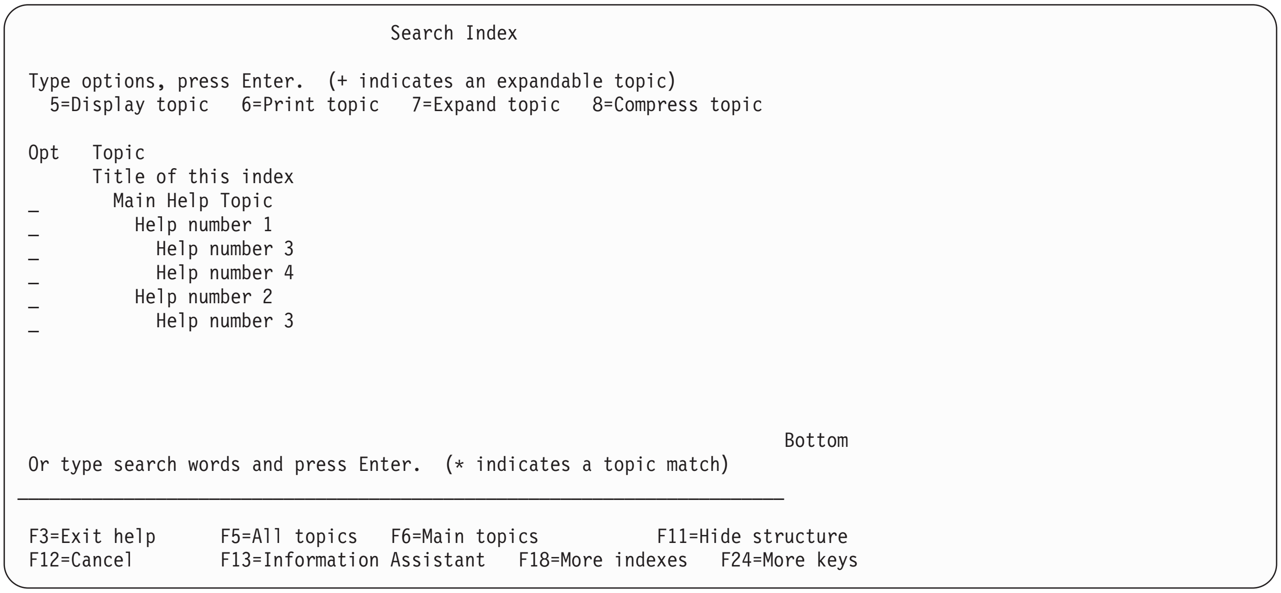
When specifying the name of a help module, apostrophes are not necessary unless characters other than A through Z, a through z, and 0 through 9 are used. If any other keyboard characters are used, apostrophes are required. For example, in a name specifying HELP='key/enter', the apostrophes are required because the slash (/) is not an alphabetic character or a numeral. For more information on the rules for names, see "Name Syntax" on page 469.

Giving Help Panel Groups Access to Index Search

Index search provides access to user created search indexes. You can give a UIM help panel group access to these search indexes using the :SCHIDX parameter on the :PNLGRP tag. Here is an example of the UIM coding to give a help panel group access to index search:

```
:PNLGRP SCHIDX=search index
   ⋮
:PANEL.
   ⋮
:EPANEL.
   ⋮
:EPNLGRP.
```

The :SCHIDX parameter causes the F11=Search Index key to display on the help panel for the UIM panel. Pressing F11=Search Index displays the index search main display as shown in Figure 124. The search index specified on the :SCHIDX parameter will be the search index displayed on the index search screen.



```
                                  Search Index

Type options, press Enter.  (+ indicates an expandable topic)
  5=Display topic   6=Print topic   7=Expand topic   8=Compress topic

Opt   Topic
      Title of this index
_       Main Help Topic
_         Help number 1
_           Help number 3
_           Help number 4
_         Help number 2
_           Help number 3




                                                                       Bottom
Or type search words and press Enter.  (* indicates a topic match)
________________________________________________________________________

F3=Exit help      F5=All topics   F6=Main topics           F11=Hide structure
F12=Cancel        F13=Information Assistant   F18=More indexes   F24=More keys
```

Figure 124. Index Search Display

Giving Help Panel Groups Access to A User-Defined Panel Group

The help panels provides access to a user-created panel group. This user-defined panel group object (*PNLGRP) must be called QGUHISF9, contain a panel called USERDEF, be located in the user's library list, and the user must have *USE authority to this object. If the above criteria is met, a function key, F9=User defined menu will appear on all UIM help and search index panels. This panel group is intended to be set up in a menu format (similar to the Information Assistant menu that appears when the user presses F13 from a help panel). An example panel group would be:

```
:PNLGRP.
:KEYL  NAME=mainkeys help=helpname.
:KEYI  KEY=f1                HELP=helpname ACTION='help'.
F1=Help
:KEYI  KEY=f3                HELP=helpname ACTION='exit' VARUPD=no.
F3=Exit
:KEYI  KEY=f12        HELP=helpname   ACTION='cancel' VARUPD=no.
F12=Cancel
:KEYI  KEY=f24        HELP=helpname       ACTION='morekeys'.
F24=More keys
```

```
:KEYI  KEY=enter      HELP=helpname       ACTION='enter'.
:KEYI  KEY=help       HELP=helpname       ACTION='help'.
:KEYI  KEY=print      HELP=helpname       ACTION='print'.
:EKEYL.
:PANEL  NAME=userdef HELP=helpname KEYL=mainkeys TOPSEP=sysnam
                     ENTER='MSG CPD9817'.
User's Info Assist Menu
:MENU  DEPTH='*' BOTSEP=space SCROLL=no.
:TOPINST.To select one of the following, type its number below and
press Enter:
:MENUI  HELP=helpname  OPTION=1 ACTION='cmd dsplibl'.
Display user's library list
:EMENU.
:OPTLINE.Type a menu option below
:EPANEL.
:HELP name=helpname.Option 1 - Help
```

help text for given help name

```
:EHELP.
:EPNLGRP.
```

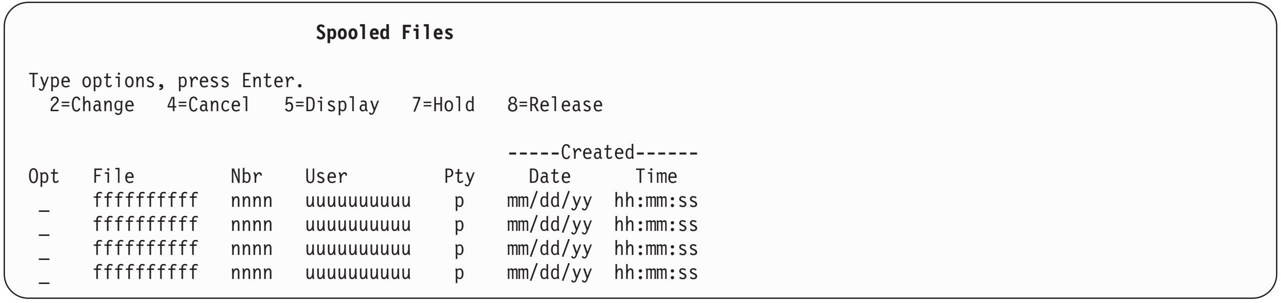
Removing Access to F18=More Indexes

To not allow users the ability to change the search indexes they are using, create a data area called QUHISF18 in the user’s library list. The user must have at least *USE authority to this data area. Creating this data area will condition off the F18=More indexes function key on the Search Index panel.

Help in a List Area

A list area displays the contents of a UIM list. The list consists of rows and columns of variable information, which appear in a table format. The following screen is an example of a panel using a list area to display a list of eight rows.

For this panel, the user may enter numbers 2, 4, 5, 7, or 8 in the *Opt* column to specify an action to be performed against one of the rows in the list.



```
                           Spooled Files

Type options, press Enter.
  2=Change   4=Cancel   5=Display   7=Hold   8=Release

                                             -----Created------
Opt   File         Nbr    User         Pty     Date      Time
 _    ffffffffff   nnnn   uuuuuuuuuu    p    mm/dd/yy  hh:mm:ss
 _    ffffffffff   nnnn   uuuuuuuuuu    p    mm/dd/yy  hh:mm:ss
 _    ffffffffff   nnnn   uuuuuuuuuu    p    mm/dd/yy  hh:mm:ss
 _    ffffffffff   nnnn   uuuuuuuuuu    p    mm/dd/yy  hh:mm:ss
```

Help in a list area must be defined at the group level and at the column level for columns which are not part of a group.

Group Level Help
To define help at the group level, specify the HELP attribute on the list group (LISTGRP) tag. This attribute identifies the help module which explains the group of columns in the list.

Column Level Help
To define help at the column level in a list area, specify the HELP attribute on the list column (LISTCOL) tag. This attribute identifies online help information which explains the purpose of the column in the list.

The HELP attribute is not allowed if the column is part of a list column group defined by the LISTGRP tag, but is required if the column is not part of a group.

The HELP attribute of the list action (LISTACT) tag identifies the help module which explains the list action. The online help information for all active list actions is displayed as part of the contextual help for the action column.

Coding Help

The following source shows how help is defined for the sample panel with a list area shown above. All help modules for this sample list area are imported from panel group *LIBL/EXAMPL2. When the cursor is positioned on the *Opt* column, the ’splf/option’ help module is displayed, followed by these help modules:

- ’splf/change’
- ’splf/cancel’
- ’splf/display’
- ’splf/hold’
- ’splf/release’

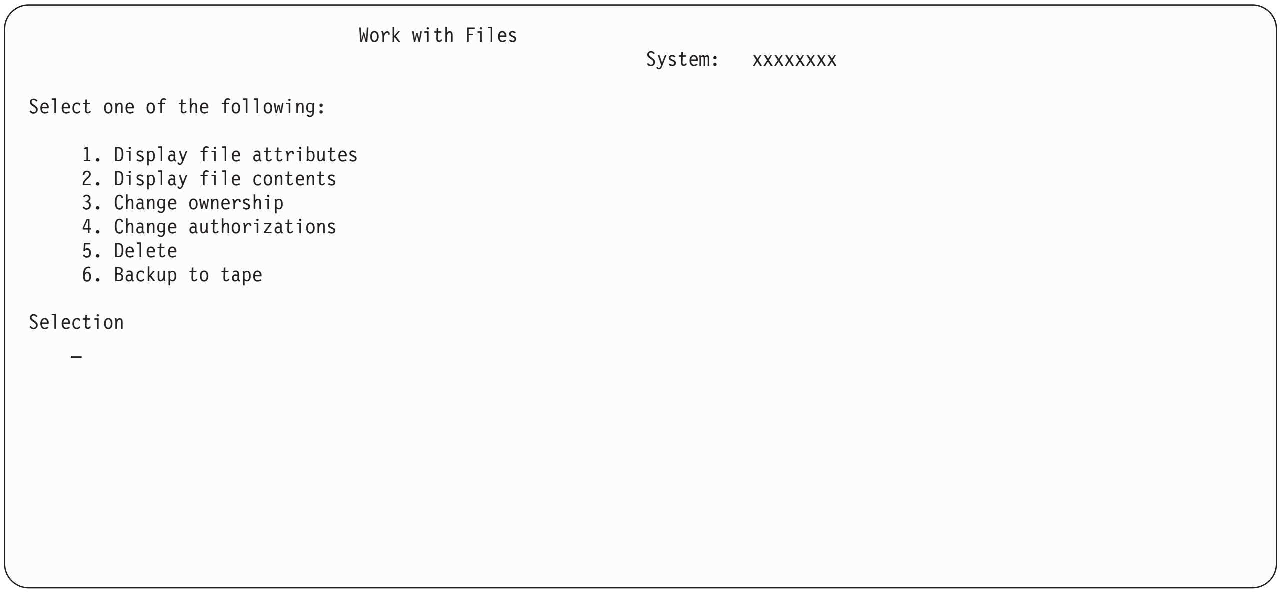
When the cursor is positioned on either the *Date* or *Time* column, the ’splf/created date time’ help module is displayed. With the cursor on any other column, the help module identified on the LISTCOL tag is displayed.

```
:import name='*' pnlgrp='*libl/example2'.
   ⋮
:listact option=2 help='splf/change' enter='call listactpgm'.2=Change
:listact option=4 help='splf/cancel' enter='call listactpgm'.4=Cancel
:listact option=5 help='splf/display' enter='call listactpgm'.5=Display
:listact option=7 help='splf/hold' enter='call listactpgm'.7=Hold
:listact option=8 help='splf/release'enter='call listactpgm'.8=Release
   ⋮
:listcol var=option help='splf/option' usage=inout maxwidth=6.Opt
:listcol var=filenam help='splf/file_name' usage=out maxwidth=12.File
:listcol var=filenbr help='splf/file_number' usage=out maxwidth=6.Nbr
:listcol var=usernam help='splf/user_name' usage=out maxwidth=12.User
:listcol var=filepty help='splf/file_priority' usage=out maxwidth=6.Pty
:listgrp col=filecrt help='splf/created_date_time'.Created
:listcol var=filedat usage=out maxwidth=8.Date
:listcol var=filetim1 usage=out maxwidth=8.Time
:elistgrp.
:listview cols='option filenam filenbr usernam filepty filecrt'.
```

Help in a Menu Area

A menu area contains one or more items, each of which contains an option number and a description of an action which can be performed. To select an option, the user enters the number of the option on the command or option line and presses the Enter key. The following screen is an example of a panel with a menu area.

This panel has six options the user can choose, and each option has a brief description of the action it performs. The option number chosen by the user is entered on the *Selection* line.



```
                              Work with Files
                                                       System:   xxxxxxxx

Select one of the following:

     1. Display file attributes
     2. Display file contents
     3. Change ownership
     4. Change authorizations
     5. Delete
     6. Backup to tape

Selection
    _
```

Help in a menu area must be defined at the item level.

Item Level Help

To define help for a menu area, specify the HELP attribute on the menu item (MENUI) tag. This attribute identifies the help module which explains the purpose of the menu item.

The online information identified by the HELP attribute is displayed when help is requested while the cursor is positioned on the text for the menu item, or while the cursor is on the command or option line and a valid menu item number has been entered.

Coding Help

The following source shows how help is defined for the sample panel shown on page 390.

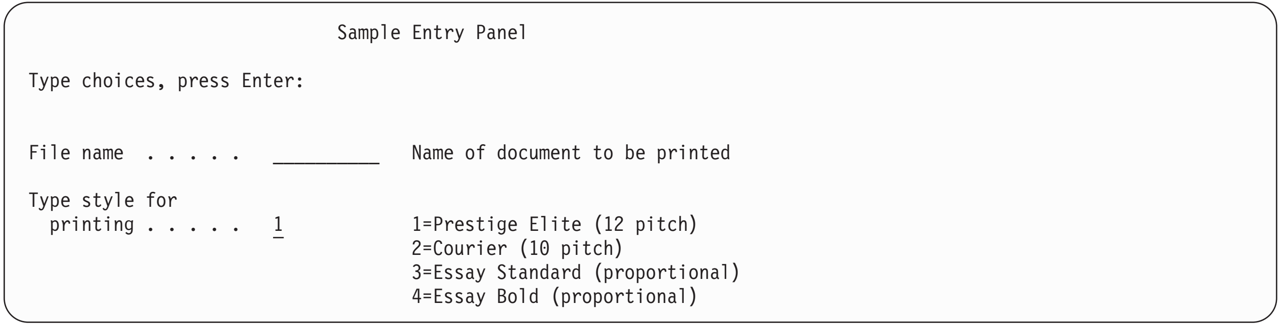
Reference numbers (n) are used in this example to show the relationship between referring to a help module using the HELP attribute of a tag, and the definition of the help module using the HELP tag.

```
:menui option=1 help='option/display_attr' action='call menuipgm'   1
       .Display file attributes
:menui option=2 help='option/display_cont' action='call menuipgm'   2
      .Display file contents
:menui option=3 help='option/change_owner' action='call menuipgm'   3
      .Change ownership
:menui option=4 help='option/change_auth' action='call menuipgm'    4
      .Change authorizations
:menui option=5 help='option/delete' action='call menuipgm'   5
      .Delete
:menui option=6 help='option/backup_tape' action='call menuipgm'   6
       .Backup to tape
   ⋮
:help name='option/display_attr'   1
      .Display File Attributes - Help
:xh3.1. Display file attributes
:p.Choose this option to display the attributes associated with this file.
The attributes include all the information about the definition of the file.
:ehelp.
:help name='option/display_contents'   2
      .Display File Contents - Help
:xh3.2. Display file contents
:p.Choose this option to display the data contained in this file.
:ehelp.
:help name='option/change_owner'   3
```

```
      .Change Ownership - Help
:xh3.3. Change ownership
:p.Choose this option to change the owner of this file.
You are prompted to enter the name of the new owner.
:ehelp.
:help name='option/change_auth'    4
      .Change Authorizations - Help
:xh3.4. Change authorizations
:p.Choose this option to change the list of users
who have authority to access this file.
You are prompted for the user names and authorizations for the file.
:ehelp.
:help name='option/delete'   5
      .Delete - Help
:xh3.5. Delete
:p.Choose this option to delete this file.
The file and all the data in the file is erased from the system
and the storage used by the file is made available for other use.
:ehelp.
:help name='option/backup_tape'   6
      .Backup to Tape - Help
:xh3.6. Backup to tape
:p.Choose this option to
save a copy of this file on a magnetic tape.
You are prompted for more information about how to back up the
file on the tape.
:ehelp.
```

Help in a Data Area

A data area contains one or more data entry items whose variable information may be changed by the user. A data area can also contain output items whose variable information can be viewed by the user but cannot be changed. The following screen is an example of a panel with a data area.



```
                              Sample Entry Panel

 Type choices, press Enter:


 File name  . . . . .   __________   Name of document to be printed

 Type style for
   printing . . . . .   1            1=Prestige Elite (12 pitch)
                                     2=Courier (10 pitch)
                                     3=Essay Standard (proportional)
                                     4=Essay Bold (proportional)
```

Help in a data area must be defined for every item at one of three levels.

Area Level Help

To define help at the area level, specify the HELP attribute on the data presentation area (DATA) tag. This attribute identifies the help module which explains all items in the area.

If no HELP attribute is specified on the DATA tag, a help module name must be associated with each item in the area by specifying the HELP attribute on the data group (DATAGRP), the data item (DATAI), and data selection field (DATASLT) tags.

To provide the user with easy access to the online information for a data area with the LAYOUT=HORIZ attribute specified on the DATA tag, specify the HELP attribute on the DATA tag instead of the HELP attribute on the individual DATAI tags.

Group Level Help

To define help at the group level, specify the HELP attribute on the DATAGRP tag. This attribute identifies the help module that explains all items in the group.

If no HELP attribute is specified on the DATA tag or on any outer or nested DATAGRP tags, the HELP attribute is required on all DATAI and DATASLT tags within the group.

Item Level Help

To define help at the item level, specify the HELP attribute on the DATAI or DATASLT tag. This attribute identifies the help module that explains the data item or selection field.

If no HELP attribute is specified on the DATA tag or on DATAGRP tags containing a data item, the HELP attribute is required on the DATAI and DATASLT tags.

To provide the user with easy access to the online information for a LAYOUT=HORIZ data area, use the HELP attribute on the DATA tag instead of the HELP attribute on the individual DATAI tags in the horizontal area.

The HELP attribute applies to all data item extender (DATAIX) tags associated with a data item.

If the HELP attribute is specified on the DATASLT tag, the HELP attribute can also be specified for each choice on the DATASLTC tag. If the HELP attribute is specified on a DATASLTC tag within a selection field, all DATASLTC tags within that selection field must have the HELP attribute specified.

For multiple-choice selection fields, the online information identified for each choice is included as part of the contextual help displayed when the cursor is positioned anywhere within the selection field.

For single-choice selection fields, the online information identified for each choice is displayed when help is requested while the cursor is positioned on the text of the choice. This online information is also included as part of the contextual help displayed when the cursor is positioned within the selection field but not on the text for one of the choices within the field. This includes occurrences when the cursor is positioned on the prompt text for the selection field or in the entry field for the selection field. If the cursor is in the entry field and a valid choice is entered, when help is requested, the help for that choice is displayed.

Coding Help

The following source shows how help is defined for the sample data area panel shown previously. All the help modules for this sample data area are imported from panel group *LIBL/DATAXMP.

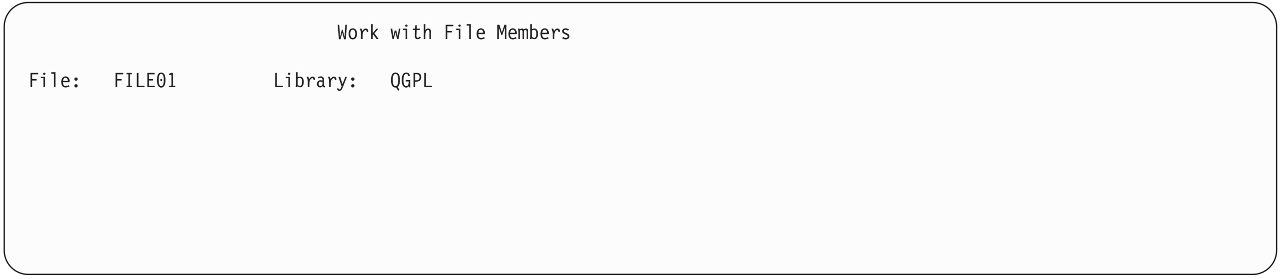
```
   ⋮
:import name='*' pnlgrp='*libl/dataxmp'.
   ⋮
:datai var=filename help='print/filename' usage=inout.File name
:datac.Name of document to be printed
:datai var=typestyle help='print/style' usage=inout.Type style for printing
:datac.1=Prestige Elite (12 pitch)
:datac.2=Courier (10 pitch)
:datac.3=Essay Standard (proportional)
:datac.4=Essay Bold (proportional)
:datai var=leftmarg help='print/left_margin' usage=inout.Left margin
:datac.Number of spaces from the left edge of the paper (1-20)
:datai var=copies help='print/copies' usage=inout.Copies
:datac.Number of copies (1-99)
:datai var=duplex help='print/duplex' usage=inout.Duplex
:datac.1=Yes (Print both sides of paper)
:datac.2=No (Print one side only)
   ⋮
```

The following source defines the help modules in panel group *LIBL/DATAXMP. These help modules are imported when the Help key is pressed for the previous sample data area.

```
   ⋮
:help name='print/filename'.File Name - Help
:xh3.File name
:p.Enter the name of document to be printed.
:ehelp.
:help name='print/style'.Type Style for Printing - Help
```

```
:xh3.Type style for printing
:p.Enter one of the following to select the type style.
:parml.
:pt.1
:pd.A 12 pitch elite style is used.
:pt.2
:pd.A 10 pitch courier style is used.
:pt.3
:pd.A proportional essay standard style is used.
:pt.4
:pd.A proportional essay bold style is used.
:eparml.
:ehelp.
:help    name='print/left_margin'.Left Margin - Help
:xh3.Left margin
:p.Enter a number from 1 to 20 for the number of spaces from
the left edge of the paper to the beginning of the printed text.
:ehelp.
:help name='print/copies'.Copies - Help
:xh3.Copies
:p.Enter a number from 1 to 99 for the number of copies of the
document to be printed.
:ehelp.
:help name='print/duplex'.Duplex - Help
:xh3.Duplex
:p.Enter one of the following to select whether or not to print on
both sides of the paper.
:parml.
:pt.1
:pd.Print text on both sides of the paper.
:pt.2
:pd.Print text only on one side of the paper.
:eparml.
:ehelp.
   ⋮
```

The following example panel shows the file name and library name in a data area with a horizontal layout.



```
                            Work with File Members

 File:   FILE01        Library:   QGPL
```

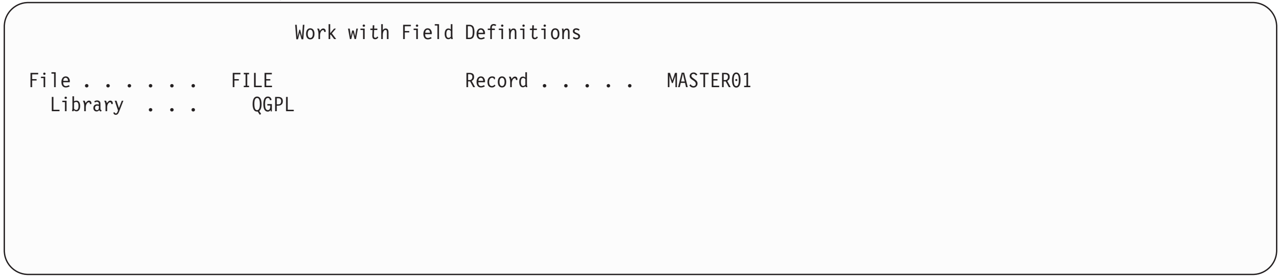
The following source shows how help is defined for the sample data area shown above. This is an example of providing area level help. When the cursor is positioned anywhere on the line containing the file name, help for the file name and library are displayed.

Reference numbers (n) are used in this example to show the relationship between a reference to a help module using the HELP attribute of a tag, and the definition of the help module using the HELP tag.

```
   ⋮
:data depth=2 help='file_and_library' layout=horiz.   1
:datai var=filename usage=out.File
:datai var=library usage=out.Library
:edata.
   ⋮
:help name='file_and_library'.File and Library - Help   1
:xh3.File and library
```

```
:p.The qualified name of the file whose members are being displayed.
:ehelp.
   ⋮
```

The following example panel shows a qualified file name and a record name in a data area with a two-column, vertical layout.



```
                        Work with Field Definitions

 File . . . . . .   FILE                Record . . . . .   MASTER01
   Library  . . .     QGPL
```

The following source shows how help is defined for the sample data area shown above. This includes providing group level help for the qualified file name. When the cursor is positioned on the file or library, help for the qualified file name is displayed. When the cursor is positioned on the record name, help for the record name is displayed.

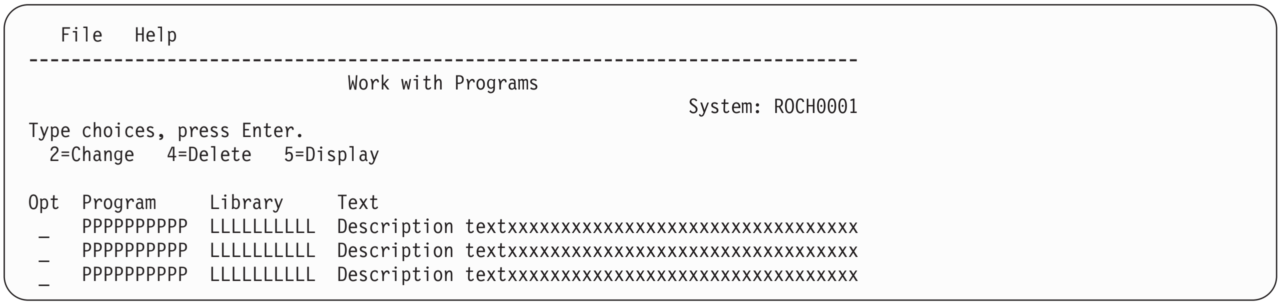
Reference numbers (**n**) are used in this example to show the relationship between referring to a help module using the HELP attribute of a tag, and the definition of the help module using the HELP tag.

```
   ⋮
:data depth=3 layout=2.
:datacol width=16.
:datacol width='*'.
:datagrp grpsep=qindent help='library/file' compact.   1
:datai var=filename usage=out.File
:datai var=library usage=out.Library
:edatagrp.
:datai var=record help=record usage=out.Record   2
:edata.
   ⋮
:help name='library/file'.File and Library - Help   1
:xh3.File and library
:p.The qualified name of the file whose field definitions are displayed.
:ehelp.
:help name=record.Record - Help   2
:xh3.Record
:p.The name of the record whose field definitions are displayed.
:ehelp.
```

Help in a Menu Bar Area

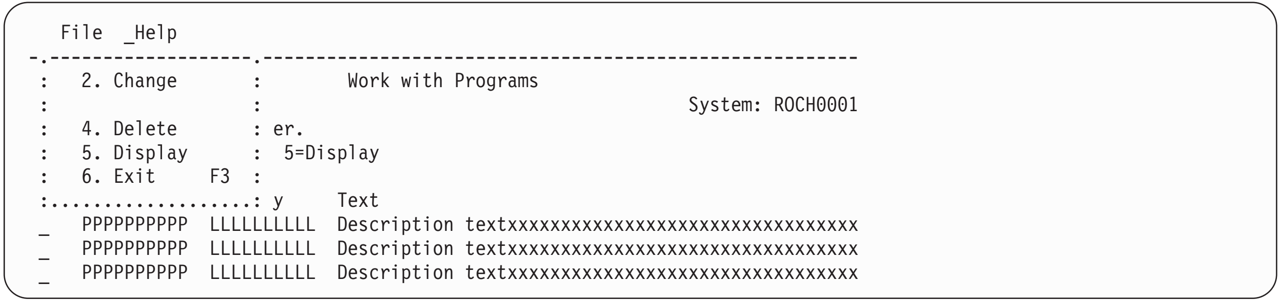
A menu bar area is located at the top of the panel and contains choices which give the user access to actions available for the panel. To select a choice in the menu bar, the user positions the cursor on the desired choice and presses the Enter key. After the user selects a choice in a menu bar, a pull-down menu appears below the menu bar containing choices for actions against the panel.

This partial screen contains an example of a menu bar area. The menu bar is the line near the top of the panel, listing *File* and *Help* as choices for the user. The user may select one of the choices to display the pull-down menu for that choice. The second screen in this example shows the pull-down menu displayed after selecting *File*.



```
   File   Help
 ------------------------------------------------------------------------------
                              Work with Programs
                                                          System: ROCH0001
 Type choices, press Enter.
   2=Change   4=Delete   5=Display

 Opt  Program     Library     Text
  _   PPPPPPPPPP  LLLLLLLLLL  Description textxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
  _   PPPPPPPPPP  LLLLLLLLLL  Description textxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
  _   PPPPPPPPPP  LLLLLLLLLL  Description textxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
```

When *File* is selected, a pull-down menu appears containing choices for actions for the panel. The actions available in the pull-down menu are *Change*, *Delete*, *Display*, and *Exit*. A pull-down menu with another list of actions appears when the user selects the *Help* menu bar choice while in the menu bar.



```
   File  _Help
 -.------------------.---------------------------------------------------------
  :   2. Change      :        Work with Programs
  :                  :                                    System: ROCH0001
  :   4. Delete      : er.
  :   5. Display     :  5=Display
  :   6. Exit     F3  :
  :..................: y     Text
  _   PPPPPPPPPP  LLLLLLLLLL  Description textxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
  _   PPPPPPPPPP  LLLLLLLLLL  Description textxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
  _   PPPPPPPPPP  LLLLLLLLLL  Description textxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
```

Contextual help for a menu bar choice includes help for the choice on the menu bar, as well as help for each active choice in the pull-down menu for that menu bar choice. Help in a menu bar area must be defined at the menu bar choice level and at the pull-down choice level.

Menu Bar Choice Level Help
To define help at the menu bar choice level, specify the HELP attribute on the menu bar choice (MBARC) tag. This attribute identifies the help module which explains the purpose of the menu bar choice.

Pull-Down Choice Level Help
To define help at the pull-down choice level, specify the HELP attribute on the pull-down field choice (PDFLDC) tag. This attribute identifies the help module which explains the purpose of the pull-down choice.

Coding Help

The following source shows how help is defined for the first menu bar choice and the pull-down menu shown on page 395.

Reference numbers (**n**) are used in this example to show the relationship between referring to a help module using the HELP attribute of a tag, and the definition of the help module using the HELP tag.

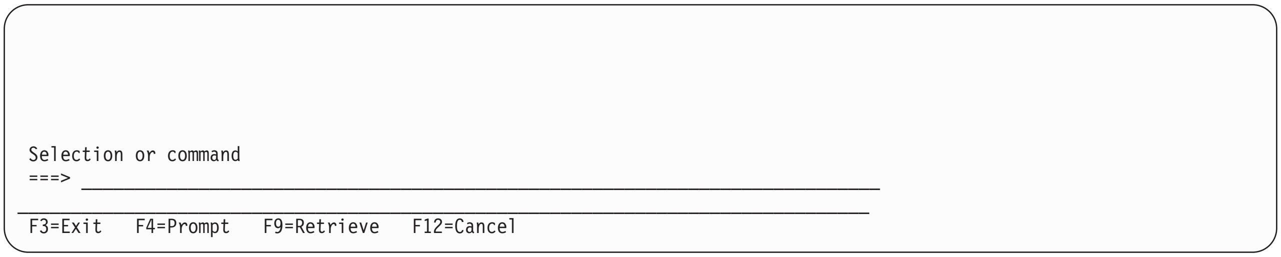
```
   ⋮
:mbarc help='mbarpgm/file'.File   1
:pdfld.
:pdfldc option=2 help='mbarpgm/file/change'   2
        action='cmd ?CHGPGM PGM(&var2./&var1.)'
        actfor=list.Change
:pdfldc option=4 help='mbarpgm/file/delete'   3
        action='cmd DLTPGM PGM(&var2./&var1.)'
        actfor=list confirm=confpgm usrexit='call exitpgm'.Delete
:pdfldc option=5 help='mbarpgm/file/display'   4
        action='cmd DSPPGM PGM(&var2./&var1.)'
        actfor=list.Display
:pdfldc option=6 help='mbarpgm/file/exit' action='exit set'   5
```

```
        varupd=no.Exit
:pdaccel.F3
:epdfld.
:embarc.
   ⋮
:help name='mbarpgm/file'.File - Help  1
:xh3.File
:p.Select this choice to display a pull-down menu containing options
to perform against selected programs.
:ehelp.
:help name='mbarpgm/file/change'.Change - Help  2
:parml.
:pt.Change
:pd.Choose this option to change attributes of the selected
programs.
This choice is not available if no programs were selected
from the list.
:eparml.
:ehelp.
:help name='mbarpgm/file/delete'.Delete - Help  3
:parml.
:pt.Delete
:pd.Choose this option to delete the selected programs.
This choice is not available if no programs were selected
from the list.
:eparml.
:ehelp.
:help name='mbarpgm/file/display'.Display - Help  4
:parml.
:pt.Display
:pd.Choose this option to display the attributes of the selected
programs.
This choice is not available if no programs were selected
from the list.
:eparml.
:ehelp.
:help name='mbarpgm/file/exit'.Exit - Help  5
:parml.
:pt.Exit
:pd.Choose this option to end the current task and return
to the display from which the task was started.
:eparml.
:ehelp.
```

Help in a Function Key Area

A function key area is located at the bottom of the panel and contains descriptions of the actions available to the user when the user presses a function key. The function key area appears on most panels.

The following partial screen contains an example of a function key area for function keys F3, F4, F9, and F12. The descriptions for these keys are listed in the function key area, which contains the text `F3=Exit`, `F4=Prompt`, `F9=Retrieve`, and `F12=Cancel`.



```
Selection or command
===> ____________________________________________________________________
__________________________________________________________________________
F3=Exit   F4=Prompt   F9=Retrieve   F12=Cancel
```

Contextual help for the function key area includes help for the entire list of keys, as well as help for each active function key. This includes function keys which do not have text displayed on the panel. Help in a function key area can be defined at the area level and must be defined at the item level.

Area Level Help
: To define help at the area level, specify the HELP attribute on the key list (KEYL) tag. This attribute identifies the help module which explains the purpose of the function key area as a whole.

Item Level Help
: To define help at the item level, specify the HELP attribute on the key item (KEYI) tag. This attribute identifies the help module which explains the purpose of the function key.

Coding Help

The following source shows how help is defined for the function key area shown on page 396. The help module, *fkey*, contains no help information. This help module only provides a title for contextual help for the function key area and a heading for the function keys in extended help. There is no title or extended help heading tag in the help for each function key because the contextual help is for the entire function key area, not for an individual key.

Reference numbers (n) are used in this example to show the relationship between referring to a help module using the HELP attribute of a tag, and the definition of the help module using the HELP tag.

```
   ⋮
:keyl name=keys help=fkey.  1
:keyi key=f3 help='fkey/exit' action='exit set' varupd=no.F3=Exit   2
:keyi key=f4 help='fkey/prompt' action=prompt.F4=Prompt  3
:keyi key=f9 help='fkey/retrieve' action=retrieve.F9=Retrieve  4
:keyi key=f12 help='fkey/cancel' action='cancel set' varupd=no.F12=Cancel   5
:keyi key=enter help='fkey/enter' action=enter.  6
:keyi key=help help='fkey/help' action=help.  7
:keyi key=pagedown help='fkey/pagedown' action=pagedown.  8
:keyi key=pageup help='fkey/pageup' action=pageup.  9
:keyi key=print help='fkey/print' action=print.  10
:ekeyl.
   ⋮
:help name='fkey'.Function Keys - Help  1
:xh3.Function keys
:ehelp.
:help name='fkey/exit'.  2
:parml.
:pt.F3=Exit
:pd.Ends the current task and returns you to the display from
which the task was started.
:eparml.
:ehelp.
:help name='fkey/prompt'.  3
:parml.
:pt.F4=Prompt
:pd.Provides assistance in entering or selecting a command.
:eparml.
:ehelp.
:help name='fkey/retrieve'.  4
:parml.
:pt.:F9=Retrieve
:pd.Shows the last command you entered on the command line,
along with any parameters you included.  By pressing this key
once, you receive the last command you ran.  By pressing this
key twice, you receive the next to last command that you ran,
and so on.
:eparml.
:ehelp.
:help name='fkey/retrieve'.  5
:parml.
:pt.F12=Cancel
```

```
:pd.Returns to the previous menu or display.
:eparml.
:ehelp.
:help name='fkey/enter'   6
:parml.
:pt.Enter
:pd.Submits information on the display for processing.
:eparml.
:ehelp.
:help name='fkey/help'   7
:parml.
:pt.Help
:pd.Provides more information about using the display.
:eparml.
:ehelp.
:help name='fkey/pagedown'   8
:parml.
:pt.Page Down or Roll Up
:pd.Moves forward to show additional information for this display.
:eparml.
:ehelp.
:help name='fkey/pageup'  9
:parml.
:pt.Page Up or Roll Down
:pd.Moves backward to show additional information for this display.
:eparml.
:ehelp.
:help name='fkey/print'  10
:parml.
:pt.Print
:pd.Prints the information currently shown on the display.
:eparml.
:ehelp.
   ⋮
```

Chapter 20. Defining Online Help Information

When the Help key is pressed and the DDS source for your display file specifies that online help information is available for the display, the system shows the information referred to by the panel group or document or, if you used DDS, contained in the record.

The following sections in this chapter tell you how to create the actual information that the user sees:

- "Defining Online Help Information in a Panel Group"

  For more information on creating online help using the UIM, see Chapter 19, "Making Online Help Accessible for Your Panel Group," on page 385 and Part 3, "Programming Application Displays Using Panel Groups," on page 271.
- "Defining Online Help Information in a DDS Record" on page 412

If the DDS source for your display does not specify that online help information is accessible for your display, go to Chapter 18, "Making Online Help Information Accessible for Your Display File."

Defining Online Help Information in a Panel Group

The user interface manager (UIM) uses panel groups to access online help information. The online help information in panel groups may be used in several ways:

- Display help
- Command help
- Index search topics

To use panel groups for online help information, you must specify them in the DDS source for your application display. If you plan to use panel groups for help but have not yet specified them in your DDS source, see "Specifying Panel Groups for Help in Your Display File" on page 373.

Entering the UIM Source for a Panel Group for Help

Like display files that are created by compiling DDS source, panel groups are created by compiling UIM source. The UIM source for a panel group is entered in a source file member. The source type for all panel groups is *PNLGRP. Steps for creating and entering data in a source file member are found in Chapter 1, "Building a Sample Display with Online Help Information."

UIM tags, which always begin with a colon (:) and end with a period (.), are used to help format and identify the information. Detailed information about UIM tags is found in Appendix A, "UIM Panel Group Definition Language."

Organizing a Panel Group with Help Modules

The UIM source for every panel group, whether used for display help, command help, or an index search topic, starts with a :PNLGRP tag and ends with an :EPNLGRP tag.

Units of help information, known as **help modules**, are defined in the panel group between the :PNLGRP and :EPNLGRP tags. Each help module starts with a named :HELP tag and ends with an :EHELP tag, as follows:

```
:PNLGRP.
:HELP name=firsthelp.Title of First Help Module
:P.
Information for first help module
:EHELP.
:HELP name=secondhelp.Title of Second Help Module
```

```
:P.
Information for second help module.
:EHELP.
:EPNLGRP.
```

A panel group may contain one or more help modules. The help module name, which is the value for the *name* attribute on the :HELP tag, identifies the help module and must be unique for each help module in the panel group. The text that follows the period (.) on the :HELP tag is used as the title when the online help information is displayed.

Using the Information in a Help Module More Than Once

The :IMHELP tag allows information to be imbedded, which means that information that is used more than once can be defined in one help module and then used within another help module, as follows:

```
:PNLGRP.
:HELP name=pacific.Pacific Ocean
:P.
The Pacific Ocean is the largest ocean in the
world.
:IMHELP name=ocean.
:EHELP.
:HELP name=atlantic.Atlantic Ocean
:P.
The Atlantic Ocean separates North and South America from
Europe and Africa.
:IMHELP name=ocean.
:EHELP.
:HELP name=ocean.
:P.
An ocean is one of the five large bodies of salt water, which
together cover nearly three-fourths of the world.
:EHELP.
:EPNLGRP.
```

Note: The :P tag used in the previous UIM source indicates the start of a paragraph. More information on the :P tag is available in Appendix A, “UIM Panel Group Definition Language.”

When the user sees the online help information in the previous example, the definition of *ocean* is the second sentence for the definitions of both the Pacific Ocean and the Atlantic Ocean.

Using a Help Module Contained in a Different Help Panel Group

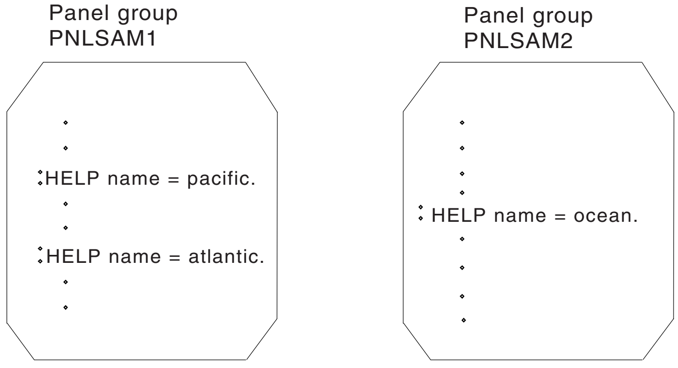
If the help module that contains the repeated information is contained in a different panel group, an :IMPORT tag is needed. The :IMPORT tag identifies the panel group that contains the needed help module and makes that panel group available for use within the current panel group. The :IMPORT tag is placed after the :PNLGRP tag and before the *first* :HELP tag in the panel group that refers to the help module.

First panel group (named *PNLSAM1*):

```
:PNLGRP.
:IMPORT pnlgrp=pnlsam2 name=ocean.
:HELP name=pacific.Pacific Ocean
:P.
The Pacific Ocean is the largest ocean in the
world.
:IMHELP name=ocean.
:EHELP.
:HELP name=atlantic.Atlantic Ocean
:P.
The Atlantic Ocean separates North and South America from
Europe and Africa.
:IMHELP name=ocean.
:EHELP.
:EPNLGRP.
```

Second panel group (named *PNLSAM2*):

```
:PNLGRP.
:HELP name=ocean.
:P.
An ocean is one of the five large bodies of salt water, which
together cover nearly three-fourths of the world.
:EHELP.
:EPNLGRP.
```



Emphasizing and Formatting the Text within a Help Module

Panel markup tags are UIM tags used within help modules to help format and emphasize the information. More detailed information about the panel markup tags, including examples that show how the tags are used, is found in Appendix A, “UIM Panel Group Definition Language.”

Defining Paragraphs and Notes: The following panel markup tags format blocks of text into paragraphs or notes:

Function	UIM Tags	Use
Notes	:NT and :ENT :NOTE and :ENOTE	A note with one or more paragraphs. A note begins with `Note:` and usually refers to something in the text that precedes it. Displayed note text is indented.
Paragraphs	:P	A block of text that forms a paragraph. Paragraph text is separated from other paragraphs and text by a blank line.
Paragraph continuation	:PC	Continuation of a paragraph that has been interrupted by another panel markup tag, such as a figure, example, or list. When you specify paragraph continuation for text, paragraph formatting returns.

Adding Headings: The following panel markup tags identify associated text as headings:

Function	UIM Tags	Use
Extended help headings	:XH1 through :XH4	Conditional headings for main topic and subtopics of information. (When displaying contextual help, the first extended help heading tag is ignored so that its text is not shown.) Headings are useful for separating and organizing sections of text. Displayed heading text is highlighted and placed on its own line.
Headings	:H1 through :H4	Main topics and subtopics of information. Headings are useful for separating and organizing sections of text. Displayed heading text is highlighted and placed on its own line.

Highlighting Text: The following panel markup tags allow you to highlight text:

Function	UIM Tags	Use
Highlighted phrases	:HP0 through :HP9 and :EHP0 through :EHP9	A word or phrase to be highlighted. Text may be highlighted to emphasize it from the surrounding text. Depending on the tag, displayed highlighted text may be shown normal, underlined, brightened or colored differently, in reverse image, or a combination of these effects.
Title Citation	:CIT and :ECIT	The title of a publication. Publication titles are underlined when displayed.

Making Lists: You may organize your text in lists using the following panel markup tags:

Function	UIM Tags	Use
Definition List	:DL and :EDL (starts and ends list); :DTHD and :DDHD (provides column headings): :DT and :DD (identify terms and their definitions)	A list of words or phrases and their corresponding definitions, descriptions, or explanations. When definition lists are displayed, each term appears in the left column with its definition across from it in the right column.
List part	:LP	A comment or explanation that applies to a part of a list. A list part allows you to enter text after a list item without making the text part of the previous list item and without interrupting the list. List part text is not indented when displayed.
Ordered list of items	:OL and :LI and :EOL	An ordered list of items. Items in an ordered list are preceded by numbers or alphabetic letters to show sequence or order. Displayed list items in an ordered list are indented.
Parameter list	:PARML and :EPARML	Parameter terms and descriptions. The parameter list is commonly used to define programming keywords and variables. The parameter term is shown in the left column; its definition appears indented on the line following the term.
Simple list	:SL and :LI and :ESL	List of items. Simple lists are commonly used when list items are contained on one line and no sequence is required. List items in a simple list are indented only and not preceded by numbers, letters, hyphens, or dashes.
Unordered list	:UL and :LI and :EUL	Unordered list of items. Unordered lists are commonly used when sequence is not required for the items in the list. List items in an unordered list are indented and preceded by the lowercase letter o, a hyphen, or a dash.

Identifying Programming Keywords and Variables: The following panel markup tags allow you to highlight programming keywords and variables, and are often used with the parameter list tags:

Function	UIM Tags	Use
Programming keyword	:PK and :EPK	A programming keyword. Programming keywords are used to explain the elements of programming syntax. Programming keywords are highlighted and, if specified as a default also, underlined.
Programming variable	:PV and :EPV	A programming variable. Programming variables are used to explain the elements of programming syntax. Programming variables are underlined when displayed.

Indicating Structured Text: If you want text to be displayed as it is entered, you can use the following panel markup tags:

Function	UIM Tags	Use
Example	:XMP and :EXMP	An area of text that is displayed exactly as it is entered. Examples are commonly used to show sample computer input or output. Displayed example text is indented.
Figures	:FIG and :EFIG	A diagram, chart, or other illustration. If desired, figures may be displayed with captions.
Unformatted lines	:LINES and :ELINES	An area of text that is displayed exactly as it is entered. Unformatted lines are used for any text that needs to be displayed without being concatenated.

For more information on the language tags, see Appendix A, "UIM Panel Group Definition Language," on page 465.

Adding Index Search Tags to a Help Panel Group

Help panel groups may also contain index search modules. Index search supplements the help information that is provided for each display. To use the information in help panel groups for the index search function, you need to add the appropriate UIM tags to your help modules.

Understanding How Index Search Works: Users can access the index search function from any display help that specifies that the index search function is available.

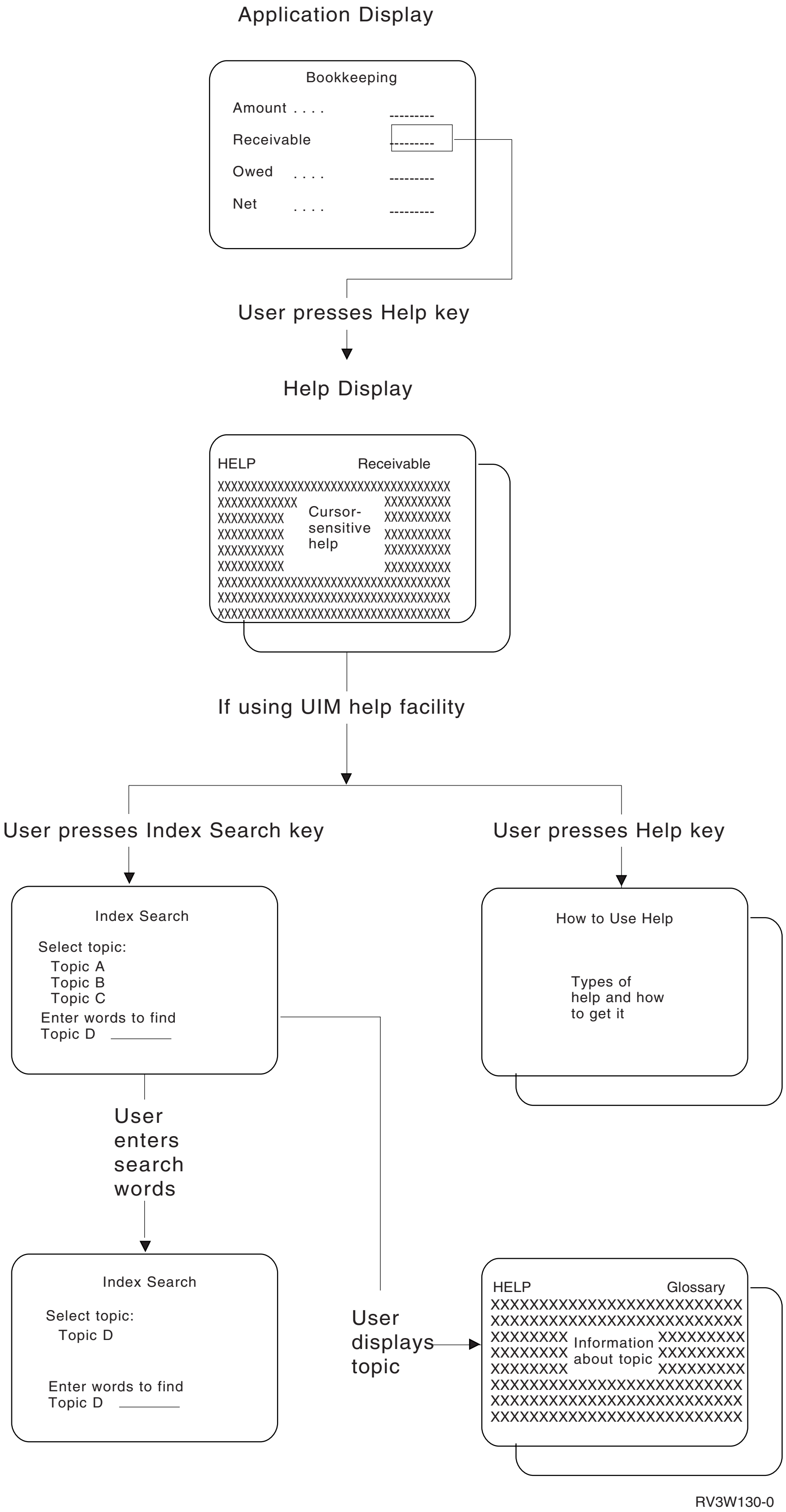
When the user presses the Help key from a working display, help information is displayed. On all help displays that support the index search function, F11=Index search is active. When the user presses F11, the Index Search display appears. This display shows a list of main topics in the topic hierarchy. It has an input field at the bottom for search words.

The user can browse or print any topic in the list or type a word (or words) on the input field. To view one or more topics, the user types a `5` in the option field beside the topic title and presses the Enter key. To print one or more topics, the user types a `6` in the option field.

A plus sign (+) next to a topic means that the user can expand the topic to show its subtopics in the hierarchy using option 7 (Expand topic). Or, the user can use option 8 (Compress topic) to compress a topic so that the subtopics are no longer shown.

If the user types a word or words on the input field, the index search function matches the words with synonym tables, and presents a list of topics that match the search words entered.

The following illustration shows how the index search function is accessed from display help:



The user may also reach the index search function by using the Start Search Index (STRSCHIDX) command. The STRSCHIDX command lets a user access search indexes without using the Help key or F11.

Deciding Which Topics to Put in Index Search: Index search topics usually consist of conceptual, procedural, and reference information.

Conceptual information usually answers the question “Why?” Conceptual information includes overviews, rationales, advantages, comparisons and contrasts, and diagrams.

Procedural information answers the question “How do I . . . ?” Procedural information always consists of a series of steps.

Reference Information typically answers the question “What is . . . ?” Reference information includes definitions, examples, commands, tables, prerequisites, and cautions.

Defining Index Search Topics and Root Words: The :ISCH tag defines the title of a topic in the index and specifies the root words that serve as the link between the topic and the search words (synonyms) entered by the user. The tag appears immediately after the :HELP tag to which it refers. There can only be one :ISCH tag within a single help module.

For each :ISCH tag, there can be several lines of root words, provided that the total number of root words is no more than 50. If more than one line of root words is used, `ROOTS=` must be repeated at the beginning of the second line and subsequent lines:

```
:PNLGRP.
:HELP name=entry1.
:ISCH ROOTS='root1 root2 root3 root4 root5'
ROOTS='root6 root7 root8 root9 root10'
ROOTS='root11 root12 root13 ... root50'.
Title of First Topic
:P.
This is the first index search module in this panel group.
:EHELP.
:EPNLGRP.
```

The root words on all lines must be enclosed in apostrophes and a period must be placed only at the end of the last line of root words. The topic title follows the period on the :ISCH tag and may be placed on the line immediately following the period.

Designating Synonyms for Root Words: The :ISCHSYN tag defines the words (synonyms) that, if entered by a user, match a specific root word. If a word that is entered by a user is a synonym for a root word, then a match is found for each topic whose :ISCH tag contains that root.

If you want a word that is used as a root word to be used as a synonym as well, you must include the word as a synonym on the :ISCHSYN tag. For example:

```
:ISCHSYN ROOT='ocean'.ocean water sea
```

The synonyms for the :ISCHSYN tag must be entered on one line, and at least one :ISCHSYN tag must exist for each root word. If more than one line is needed, more :ISCHSYN tags may be entered for the same root word.

UIM does not differentiate between synonyms entered in uppercase, lowercase, or mixed case. For this reason, it is not necessary to repeat synonyms to cover all the different cases.

You may use alphabetic or numeric characters for synonyms; however, the following characters (including their hexadecimal equivalents) are not allowed to be used as a synonym or part of a synonym:

. (period)

((left parenthesis)

) (right parenthesis)

; (semicolon)

, (comma)

? (question mark)

: (colon)

The :ISCHSYN tags may be placed anywhere in the panel group, but to make maintenance easier, place them all in one area (such as at the beginning of your panel group or in a panel group object that contains only :ISCHSYN tags).

The following example shows some :ISCHSYN tags and the :ISCH tags that use them:

```
:PNLGRP.
:ISCHSYN ROOT='ocean'.ocean water sea
:ISCHSYN ROOT='lake'.lake water pond
:ISCHSYN ROOT='definition'.definition define description what
:ISCHSYN ROOT='definition'.summary concept information explanation
:HELP name='defocean'.
:ISCH ROOTS='definition ocean'.
Definition of ocean
:P.
An ocean is one of the five large bodies of salt water, which
together cover nearly three-fourths of the world.
:EHELP.
:HELP name='deflake'.
:ISCH ROOTS='definition lake'.
Definition of lake
:P.
A lake is a body of standing water that is enclosed by land.
:EHELP.
:EPNLGRP.
```

Choosing Root Words and Synonyms for Index Search Topics: The following tips help you decide which words to use as root words and synonyms for your index search topics:

- Design the root words as building blocks that can be put together in different combinations for different topics.
- The significant words in the title of a topic provide a good start on an appropriate set of root words for the topic.
- Use only one form of a word as a root word if all forms of the word have the same synonyms.
- If two similar words have the same synonyms, use only one of the words as a root word, not both.
- There will be cases where it is appropriate to use two separate root words that have many common synonyms.
- Include all synonyms that seem reasonable for the root word.
- Remember that the root words are only used to connect synonyms with help topics.
- You may have to create a special root word that applies to only one topic or a few specific topics, but these special root words should be the exception, not the rule.
- For topics whose titles include an abbreviation, the abbreviation and the major words making up the abbreviation should all be used as root words.
- Either uppercase or lowercase may be used.
- Use comment lines liberally to clarify the context in which particular root words are to be used.
- If the root word is a real word, include the same word as a synonym for the root word.
- You may want to include misspellings of commonly misspelled words as synonyms.

Defining an Index Search Hierarchy: The index search subtopic (ISCHSUBT) tag identifies the help modules within the same panel group that are subtopics under the preceding topic specified on an index search (ISCH) tag. The ISCHSUBT tag must appear after the ISCH tag. Any help module that is not identified by an ISCHSUBT tag is a primary topic in the index search hierarchy. Therefore, if no ISCHSUBT tags are used, all help modules are primary topics and there is no hierarchy.

The TOPIC attribute on the ISCHSUBT tag is used to define the subtopics for a topic. The order in which the help modules appear on the TOPICS attribute is the order in which they are displayed in the index search hierarchy. For more information on the rules for help module names, see “Name Syntax” on page 469.

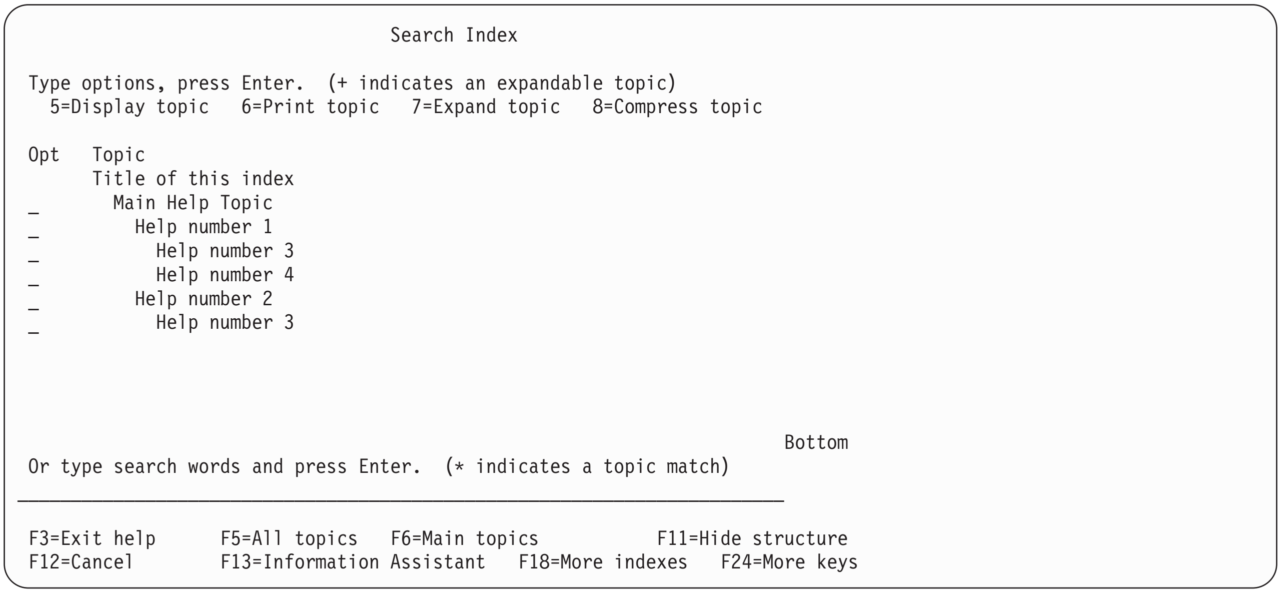
A topic can be the subtopic of more than one topic.

Topics can be nested to no more than 16 levels.

The following example shows how the ISCH tags and ISCHSUBT tags work together to form an index search hierarchy:

```
:PNLGRP.
:HELP name=mainhelp.
:ISCH roots='root words'.
Main Help Topic
:ISCHSUBT topics='help1'
          topics='help2'.
   ⋮
:EHELP.
:HELP name=help1.
:ISCH roots='root words'.
Help number 1
:ISCHSUBT topics='help3 help4'.
   ⋮
:EHELP.
:HELP name=help2.
:ISCH roots='root words'.
Help number 2
:ISCHSUBT topics='help3'
   ⋮
:EHELP.
:HELP name=help3.
:ISCH roots='root words'.
Help number 3
   ⋮
:EHELP.
:HELP name=help4.
:ISCH roots='root words'.
Help number 4
   ⋮
:EHELP.
:EPNLGRP.
```

This UIM source creates the following index search hierarchy:



```
                              Search Index

Type options, press Enter.  (+ indicates an expandable topic)
  5=Display topic   6=Print topic   7=Expand topic   8=Compress topic

Opt   Topic
      Title of this index
_       Main Help Topic
_         Help number 1
_           Help number 3
_           Help number 4
_         Help number 2
_           Help number 3




                                                                        Bottom
Or type search words and press Enter.  (* indicates a topic match)
_________________________________________________________________________

F3=Exit help      F5=All topics   F6=Main topics           F11=Hide structure
F12=Cancel        F13=Information Assistant   F18=More indexes   F24=More keys
```

National Language Considerations: The index search function can be used with either double-byte character support (DBCS) or single-byte character support (SBCS) data. When DBCS data is used, the device from which it is requested must be capable of entering and presenting the data in DBCS. The object which contains the index search data is marked as containing DBCS data when appropriate. The system determines if the device is capable of handling the DBCS data.

When the data is being prepared for DBCS format and the index search function is used with that data, consider the following:

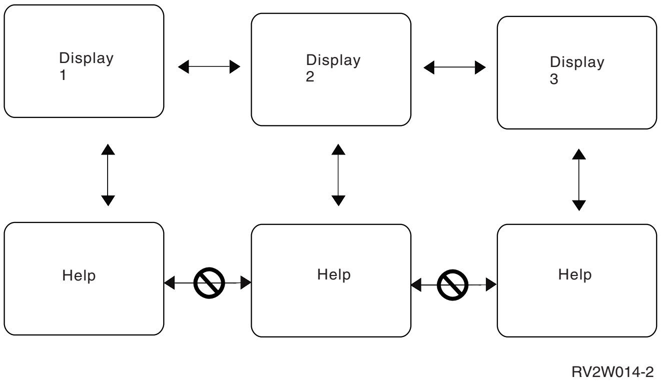
- When the index search data is prepared for a DBCS system, the synonyms entered on the ISCHSYN tag must be in double-byte character mode. That is, the first byte after the tag must be a shift-out character and the last byte of the data must be a shift-in character. The system does not convert data on the ISCHSYN tag to double-byte.
- Words must be separated by a single-byte blank. From 1 to 19 double-byte characters may be combined to form a word. Intervening shift-out and shift-in characters are allowed, but are ignored by index search.
- The words that are used to link the ISCH and ISCHSYN tags (the `ROOTS` attribute of the ISCH tag and the `ROOT` attribute of ISCHSYN tag) must be identical and should not be entered in DBCS.
- Search words can be entered in either single-byte mode or double-byte mode. Single-byte blanks can be entered to separate the words.

When the search words are shown on the screen, the double-byte character representation (the character that was actually used in the search) is shown. Special processing takes place so that index search is not case sensitive. The search words from the ISCHSYN tag are uppercased using a translation table for the code page that is specified with the `TXTCHRID` attribute of the PNLGRP tag. If the search words are DBCS, they are not uppercased. Shift-out and shift-in characters are treated as blanks during parsing; leading and trailing blanks are removed. All SBCS words are uppercased using a translate table for the code page of the device description. For more information about DBCS, see Double-byte character set support in the iSeries Information Center.

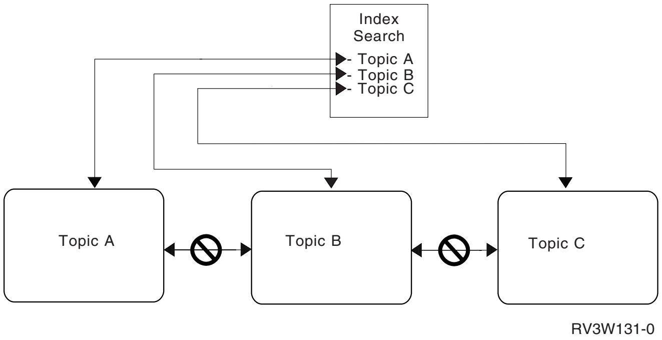
Linking Help Modules

Hypertext is a structure of related help modules that are linked together by their common areas. The linking allows users to select one or more help modules of interest from another help module. A help module that links to one or more help modules is called an information **node**.

Without hypertext, the only way you can access help is through the associated display or the command prompter. You cannot go directly from one help module to another unless a link, an association between two information nodes, exists that makes each help module a node in a hypertext network.



Similarly, the only way you can access an index search topic is through the index search function. You cannot go directly from one index search topic to another.



These restrictions make it difficult for you, first, to determine what related information exists and, second, to access the information conveniently. Hypertext makes it possible for you to identify relationships among information nodes so that you and other users can easily access the information you need.

Designing Your Links: The structure of your hypertext nodes determines the relationships among the different nodes.

Relationships that involve hierarchy (such as those between a task and its subtasks or between a command and its parameters) can be expressed as subordinate nodes below a larger node that they relate to, in the same way that an organization chart represents the management structure.

Relationships that do not involve hierarchy (such as those between a procedural node and a reference node or between two similar procedures) can be expressed as clusters of nodes.

If you are designing a complex hypertext structure, the following questions may help you make design decisions about what to link to what:

- "What don't I understand here?"
- "What words are unfamiliar?"
- "What conceptual information is assumed?"
- "What else do I need to know to complete my task?"
- "What is the next task I want to do after this one?"
- "What other tasks are similar to this one?"

- “What would a graphical representation of this information look like?”
- “What is a specific example of this general information?”
- “What other displays or commands are related to this one?”

You are allowed as many links as you want from any one information node; however, it is not essential that each information node be linked to another node. A link implies a clear logical relationship. If the relationship is not clear and logical, no link should exist.

Creating Links: A **hypertext reference phrase** is a word or phrase that identifies a selectable hypertext link. The reference phrase is emphasized (with high intensity or color or underlining or a combination of these, depending on the display station). The hypertext reference phrase tells the user that there is more information at the other end of the link.

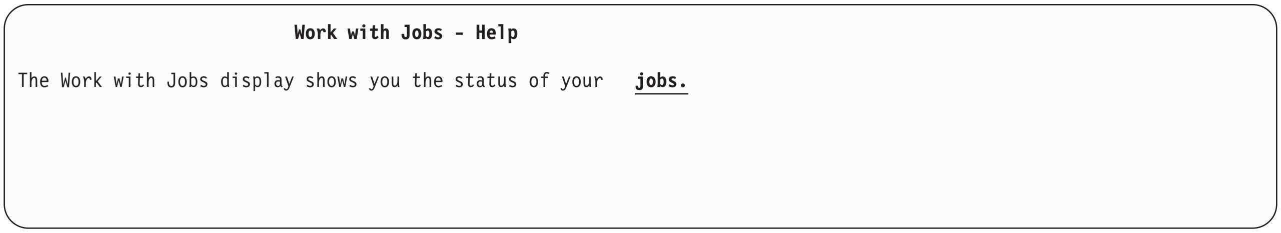
To create a hypertext link, use the :LINK tag to mark the beginning of the reference phrase. Use the corresponding :ELINK tag to mark the end of the reference phrase.

You define a link in one direction only, from node A to node B. The link back from node B to node A is not defined by a :LINK tag. However, the user can return from node B to node A by pressing F12. The user can also press F6 to display a list of the titles of nodes previously displayed, then position the cursor next to a title on that list and press Enter to return to and display the selected node again.

The following example uses the :LINK tag to create a link from one help module to another help module:

```
:HELP NAME='wrkjob'.Work with Jobs - Help
:XH3.Work with jobs
:P.
The Work with Jobs display shows you the status of your
:LINK PERFORM='DSPHELP job pnlgrp1'.
jobs.
:ELINK.
:EHELP.
```

The following display shows how the preceding example would look to the user:



```
                         Work with Jobs - Help

 The Work with Jobs display shows you the status of your   jobs.
```

Note that the reference phrase is emphasized when displayed. It is also preceded by three blanks for an attribute byte, a one-character field, and another attribute byte. By using the Tab key to place the cursor on this field, the user can see the additional information about the highlighted word or phrase by pressing the Enter key.

Creating and Deleting Panel Groups

The Create Panel Group (CRTPNLGRP) command creates panel groups for help displays. In the following example, the panel group named PNLSAM in library LIBSAM is created by using source file member HDMSAM in the same library.

```
CRTPNLGRP PNLGRP(LIBSAM/PNLSAM) SRCFILE(LIBSAM/SRCSAM) SRCMBR(HDMSAM)
```

The Delete Panel Group command (DLTPNLGRP) deletes panel groups from the system. In the following example, the panel group named PNLSAM in library LIBSAM is deleted:

```
DLTPNLGRP PNLGRP(LIBSAM/PNLSAM)
```

Assigning Panel Groups as Help for Commands

Panel groups can be created as help for command parameters that may be prompted.

The first help module, which contains extended help for the command, is specified for the Help identifier parameter for the Create Command (CRTCMD) command. Each additional help item in the panel group corresponds to a parameter in the command. Only one help module can exist for each parameter that can be prompted.

For a help module to be associated with a parameter, the :HELP name must use the following convention:

```
:HELP name='help-identifier/parameter-name'.
```

where `help-identifier` is the name specified for the Help identifier parameter in the Create Command (CRTCMD) command, and `parameter-name` is the name of the parameter that the help module describes.

The following example shows one way to organize a panel group for command help:

```
:PNLGRP.
:HELP name=startcmd.
:P.
The text for this help module is used as the
extended help for the command.
:EHELP.
:HELP name='startcmd/parameter1'.
:P.
This help item is used for information about the first
parameter.
:EHELP.
:HELP name='startcmd/parameter2'.
:P.
This help item is used for information about the second
parameter.
:EHELP.
:EPNLGRP.
```

Using Panel Groups in a Search Index

Panel groups that contain :ISCH, :ISCHSYN, and :ISCHSUBT tags can be added as entries in a search index. To find out how to access the index search function, see “Understanding How Index Search Works” on page 403.

Creating a Search Index

The Create Search Index (CRTSCHIDX) command creates a search index object. Search index entries that provide the reference to the online help information contained in one or more panel group objects may be added to this object. These referenced panel groups can be added and removed from the search index.

In the following example, a search index object named ACCOUNTING is created in the current library:

```
CRTSCHIDX SCHIDX(ACCOUNTING)
          TITLE('Accounting Help Index')
          TEXT('Accounting Help Index')
```

Adding Entries to a Search Index

The Add Search Index Entry (ADDSCHIDXE) command causes a search index object to build references to the root words and synonyms for each help module in a panel group that contains an ISCH tag. The help modules in the panel group then become entries in the search index object.

When a search index object refers to several panel groups as entries, the order that the panel groups are added into the search index object determines the order the entries are displayed when search index function is shown. Similarly, the order of the help modules in a panel group define the order in which those topics appear in the search index. If the panel group defines a hierarchy, then the topics appear in the order defined by the hierarchy.

A limit of 1000 panel groups may be added to a search index object. Only one type of panel group may be added to a search index object. That is, a search index object cannot contain panel groups that use the ISCHSUBT tag and panel groups that do not use the ISCHSUBT tag.

In the following example, the panel group named PAYROLL is added to the search index named ACCOUNTING. Both the panel group object and the search index object must exist in the library list.

```
ADDSCHIDXE   SCHIDX(ACCOUNTING)   PNLGRP(PAYROLL)
```

Removing Entries from a Search Index

The Remove Search Index Entry (RMVSCHIDXE) command removes the references to a panel group from the search index object. The RMVSCHIDXE command removes references to a panel group object that was added using the ADDSCHIDXE command.

In the following example, entries for the panel group PAYROLL are removed from the search index ACCOUNTING. The search index object is found by searching the library list.

```
RMVSCHIDXE    SCHIDX(ACCOUNTING)   PNLGRP(PAYROLL)
```

Deleting a Search Index

The Delete Search Index (DLTSCHIDX) command deletes a search index object from the system. The DLTSCHIDX command does not delete any panel groups that are referred to by the search index object.

In the following example, a search index object named ACCOUNTING is deleted from the library where the object is first found in the library list.

```
DLTSCHIDX    SCHIDX(ACCOUNTING)
```

Copying QUSRTOOL Examples That Define Help in a Panel Group

The QUSRTOOL library provides sample DDS-described displays that access online help information using panel groups. You can copy these examples into a library of your choosing and then tailor them for your own use. For more information about these sample displays, see “Using the Displays Example in the QUSRTOOL Library” on page 415.

Defining Online Help Information in a DDS Record

You can use DDS records for the online help information for your display.

The record formats that give the actual online help information may be included in the same member as the DDS source for the application display. The record formats may also be contained in a different display file. An example of DDS source used for online help information is shown in “Entering the Records That Contain the Help Information” on page 378.

Part 5. Guidelines for IBM i5/OS-Style Displays

Chapter 21. Designing IBM i5/OS-Style Displays 415
Using the Displays Example in the QUSRTOOL Library 415
Recognizing the Example Objects. 415
Installing the Example Objects. 417
Viewing the Sample Displays, Command, and Online Help Information 417
Copying the Source for the Example Objects for Your Own Use 420
Defining Special Functions and Attributes for All Displays 421
Designing the Single-Choice Menu Display . . . 421
Title 422
Instruction Line 422
Menu Options 422
Menu Selection Entry Field 423
Function Keys 423
Online Help Information 423
General Menu Display Operation. 423
Designing the Entry Display 423
Title 424
Instruction Line 424
Prompt Area 425
Field Prompts 425
Function Keys 428
Online Help Information 428
General Entry Display Operation 429
Designing the Information Display 429
Title 430
Location Information 430
Prompt Area 1 431
Prompt Area 2 431
Instruction Line 432
Function Keys 432
Online Help Information 432
General Information Display Operation. . . . 432
Designing the List Display 432
Title 433
Prompt Area 433
Instruction Line 434
Options Line 434
Column Headings. 435
Extended Action Entry Area 435
List Fields 435
Paging Location Information 436
Function Keys 436
Online Help Information 436
General List Display Operation 437
Operating the List Control Field 437
Positioning the List 437
Positioning to Lowercase Names in a List 438
Changing the List Control Field and Positioning the List 438
Operating the Extended Action Entry Area 438
List Operation When Options Are Specified 440
Cursor Positioning Rules 441
Error Condition Rules 441
List Where Only One Item Can Be Selected 442
List Format in Empty List Situation 442
Defining the Function Key Area for All Displays 443
Optional Command Line and Identifier Field 443
Available Command Line Tool. 446
Common Key Assignments. 446
Defining Help Information for All Displays . . . 450
Help for the Menu Display. 450
Help for the Entry Display 451
Help for the Information Display. 452
Help for the List Display 453
Defining and Presenting Messages 454
Designing Common User Access (CUA) Entry Level Models 456
Entry Dialog Actions 456
Function Key Area and Message Line Relationship. 457
Single-Choice Selection (Menu) 457
Selection Choices and Choice Entry Field . . 458
Guidelines for Single Selection Field Operation 458
Entry Display 459
Entry Fields 459
Information Display 460
List Display 460
Help Information 461

Chapter 21. Designing IBM i5/OS-Style Displays

If you use data description specifications (DDS) to implement your displays, this chapter shows the format of each i5/OS system display type, giving the position and characteristics of all key display elements, both constants and data fields. In addition, this chapter describes the common actions assigned to function keys across the system. However, if you use the user interface manager (UIM) to implement your displays, the UIM provides the correct formatting and placement for you. For more information on UIM, see Part 3, “Programming Application Displays Using Panel Groups,” on page 271.

By using the guidelines in this chapter, your displays will be compatible with the Systems Application Architecture® (SAA®) environment, which makes significant use of the IBM Systems Application Architecture Common User Access (CUA) rules and guidelines that apply to display stations.

Using the Displays Example in the QUSRTOOL Library

The purpose of the QUSRTOOL library, which is optionally installable on the base operating system, is to provide you access to examples of various tools and programming techniques that may help you with application development and management of your system.

Before you can use any of these program examples, the save files in which they are packaged must first be changed into source physical files. This may have already been done for you; if not, an UNPACKAGE tool is provided to do it for you. For instructions for unpackaging the files, see member TTTINFO in source physical file QATTINFO. (You can use the DSPPFM command to display the physical file member.)

Source physical file QATTINFO contains all the information you’ll need to get started. For each example program in the QUSRTOOL library, there is a member in QATTINFO that describes the tool and how to install it.

The displays example in the QUSRTOOL library gives you access to four sample displays and a sample command. Online information is available for the displays and the command and includes the index search function and hypertext links.

Recognizing the Example Objects

The sample displays tool in QUSRTOOL is made up of several members that will help you install, create, and delete the objects used by the example:

Table 40. Source Members for Displays Example in QUSRTOOL (Install, Create, and Delete)

Source Member	Source File	Object Type	Description
T0011INF	QATTINFO	N/A	Primary documentation member that tells you what to do to install the example objects in a library of your choosing. The documentation may be viewed using the DSPPFM command, CPYF to a printer, or using the SEU display function.
T0011CRT	QATTCL	*PGM	Installation program for creating all the example #1 objects
T0011CR2	QATTCL	*PGM	Installation program for creating all the example #2 objects
T0011CR5	QATTCL	*PGM	Installation program for creating all the example #3 objects

Table 40. Source Members for Displays Example in QUSRTOOL (Install, Create, and Delete) (continued)

Source Member	Source File	Object Type	Description
T0011DCL	QATTCL	*PGM	Program that shows the example display file with its four sample displays, command, and online help information
T0011DLT	QATTCMD	*CMD	Command to delete all the example objects for example #1
T0011DC2	QATTCMD	*CMD	Command to delete all the example objects for example #2
T0011DC5	QATTCMD	*CMD	Command to delete all the example objects for example #3
T0011DEL	QATTCL	*PGM	Program used by the T0011DLT command to delete the example objects for example #1
T0011DL2	QATTCL	*PGM	Program used by the T0011DC2 command to delete the example objects for example #2
T0011DL5	QATTCL	*PGM	Program used by the T0011DC5 command to delete the example objects for example #3

The sample displays, command, and online help information are contained in the following members of QUSRTOOL:

Table 41. Source Members for Displays Example in QUSRTOOL (Sample Displays, Command, and Online Help Information)

Source Member	Source File	Object Type	Description
T0011CMD T0011CM2 T0011CM3 T0011CM5	QATTCMD	*CMD	Sample commands
T0011CLP T0011CP2 T0011CP3 T0011CP5	QATTCL	*PGM	Command processing program (CPP) used by the sample command
T0011CHL	QATTUIM	*PNLGRP	Online help information for the sample command
TOO11DDS	QATTDDS	*FILE	Example display file that contains the DDS source for a sample i5/OS-style menu, entry display, information display, and list display
T0011DHL	QATTUIM	*PNLGRP	Online help information for the example display file for example #1
T0011IDX	QATTUIM	*PNLGRP	Online help information referred to by the index search object
T0011DD5	QATTDD5	*FILE	Source for physical file for example #3.
T0011HL2	QATTUIM	*PNLGRP	Online help for example #2.
T0011MN2	QATTUIM	*MENU	Menu for example #2.
T0011PNI	QATTUIM	*PNLGRP	Panel group for example #3 (contains various declarations).
T0011PN2	QATTUIM	*PNLGRP	Panel group for example #2.
T0011PN5	QATTUIM	*PNLGRP	Panel group for example #3.
T0011PN6	QATTUIM	*PNLGRP	Online help for example #3.
T0011RA5	QATTRPG	*PGM	List option processing program for example #3.
T0011RE5	QATTRPG	*PGM	General panel exit program for example #3.

Table 41. Source Members for Displays Example in QUSRTOOL (Sample Displays, Command, and Online Help Information) (continued)

Source Member	Source File	Object Type	Description
T0011RF5	QATTRPG	*PGM	Program to process F4 for example #3.
T0011RF6	QATTRPG	*PGM	Function key processor for example #3.
T0011RI6	QATTRPG	*PGM	RPG PLISTs used by programs for example #3.
T0011RI7	QATTRPG	*PGM	VARRCD definitions used by programs for example #3.
T0011RL5	QATTRPG	*PGM	Incomplete list processing exit program for example #3.
T0011RP5	QATTRPG	*PGM	Main program for example #3.
T0011RP6	QATTRPG	*PGM	F4 prompt processor for example #3.

The other objects are not contained in the QUSRTOOL library but are created when you create the example objects into your library:

Table 42. Objects Created When Creating Example Objects

Source Member	Object Type	Description
T0011IDX	*SCHIDX	Search index object
T0011MSGFL	*MSGF	Message file used when creating the sample command and display file

Installing the Example Objects

To install the displays example, follow the installation instructions in T0011INF. The install programs place all the example objects in a library of your choosing.

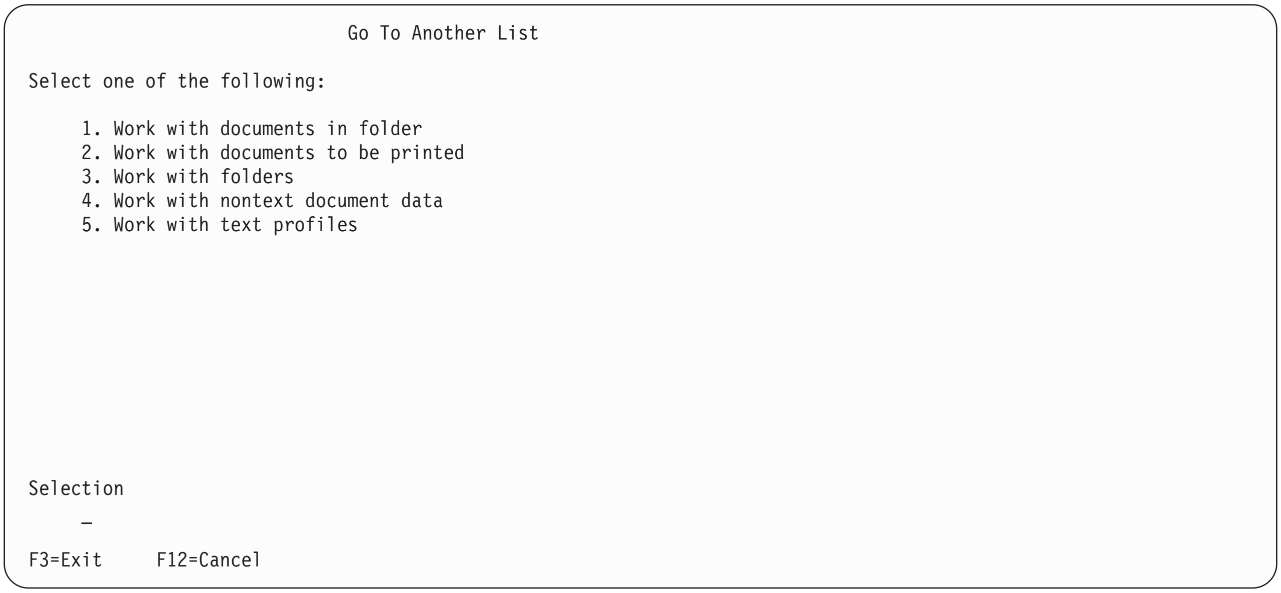
Viewing the Sample Displays, Command, and Online Help Information

You can view the sample displays, command, and online help information after installing the example objects. To view the samples, enter the following commands:

- To view the sample menu, enter:

  ```
  CALL  PGM(your-library/T0011DCL)  PARM(MENU)
  ```

  The sample menu appears:



```
                              Go To Another List

Select one of the following:

     1. Work with documents in folder
     2. Work with documents to be printed
     3. Work with folders
     4. Work with nontext document data
     5. Work with text profiles










Selection
     _

F3=Exit     F12=Cancel
```

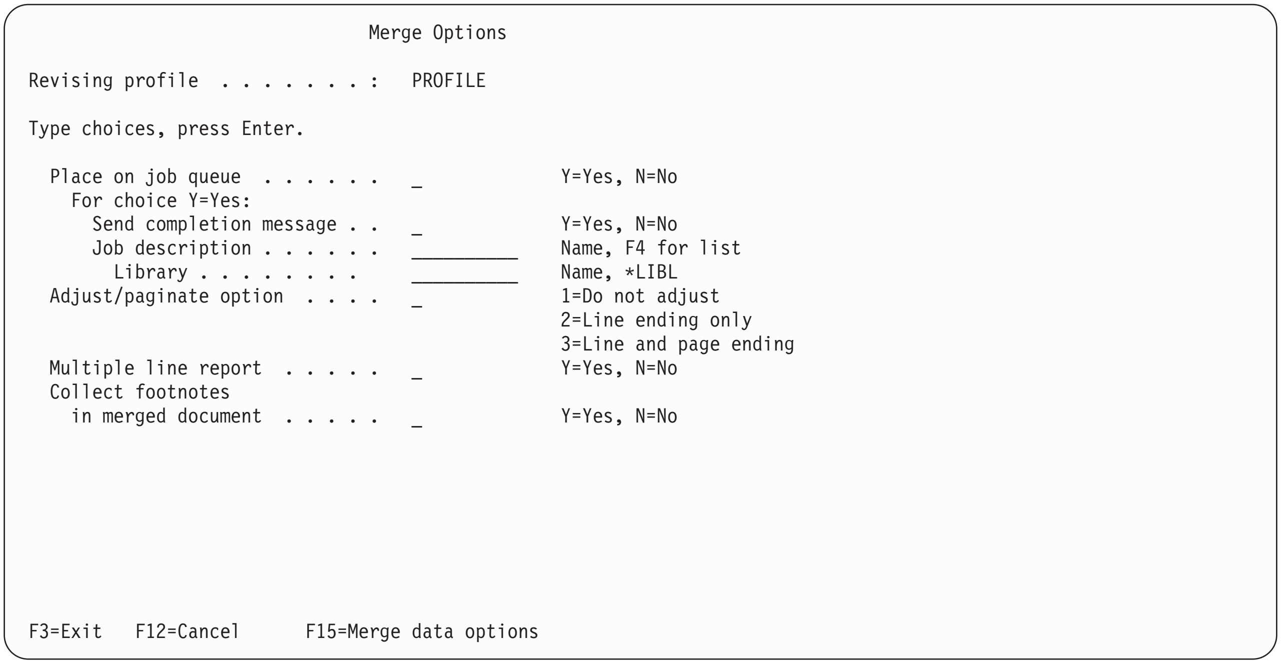
Figure 125. Sample Menu in QUSRTOOL

Press the Help key to view the online help information for the display. The online help information shown depends on where the cursor is located when you press the Help key.

- To view the sample entry display, enter:

  ```
  CALL  PGM(your-library/T0011DCL)  PARM(ENTRY)
  ```

  The sample entry display appears.



```
                                Merge Options

Revising profile  . . . . . . . :   PROFILE

Type choices, press Enter.

  Place on job queue  . . . . . .   _             Y=Yes, N=No
    For choice Y=Yes:
      Send completion message . .   _             Y=Yes, N=No
      Job description . . . . . .   __________    Name, F4 for list
        Library . . . . . . . .     __________    Name, *LIBL
  Adjust/paginate option  . . . .   _             1=Do not adjust
                                                  2=Line ending only
                                                  3=Line and page ending
  Multiple line report  . . . . .   _             Y=Yes, N=No
  Collect footnotes
    in merged document  . . . . .   _             Y=Yes, N=No








F3=Exit   F12=Cancel      F15=Merge data options
```

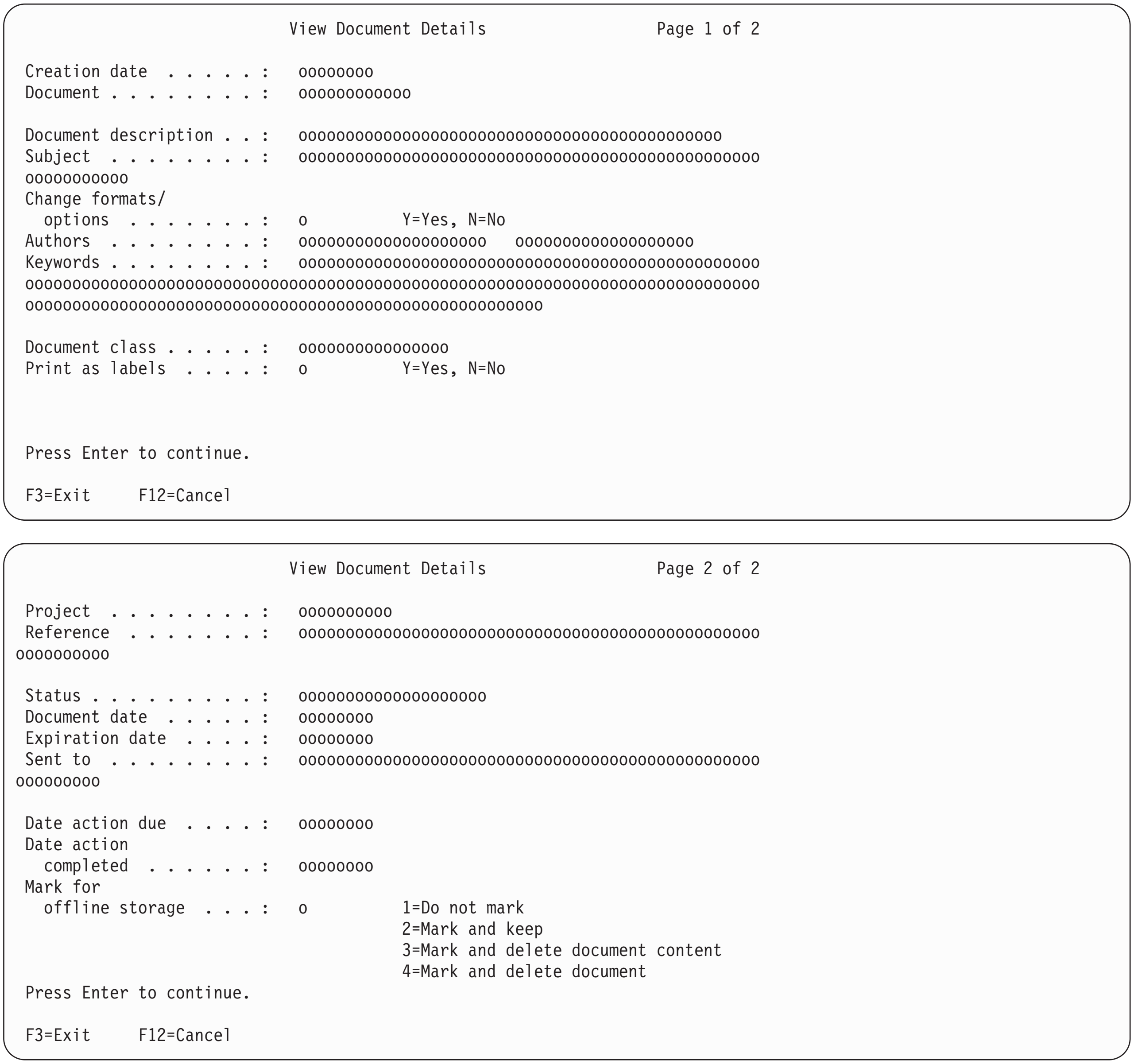
Figure 126. Sample Entry Display in QUSRTOOL

Press the Help key to view the online help information for the display. The online help information shown depends on where the cursor is located when you press the Help key.

- To view the sample information display, enter:

  `CALL  PGM(your-library/T0011DCL)  PARM(INFO)`

  The sample information display appears:



```
                            View Document Details                  Page 1 of 2

Creation date  . . . . . :   00000000
Document . . . . . . . . :   000000000000

Document description . . :   000000000000000000000000000000000000000000000
Subject  . . . . . . . . :   0000000000000000000000000000000000000000000000000
00000000000
Change formats/
  options  . . . . . . . :   0          Y=Yes, N=No
Authors  . . . . . . . . :   00000000000000000000   0000000000000000000
Keywords . . . . . . . . :   0000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000

Document class . . . . . :   0000000000000000
Print as labels  . . . . :   0          Y=Yes, N=No



Press Enter to continue.

F3=Exit     F12=Cancel
```

```
                             View Document Details                  Page 2 of 2

 Project  . . . . . . . . :   0000000000
 Reference  . . . . . . . :   0000000000000000000000000000000000000000000000000
0000000000

 Status . . . . . . . . . :   00000000000000000000
 Document date  . . . . . :   00000000
 Expiration date  . . . . :   00000000
 Sent to  . . . . . . . . :   0000000000000000000000000000000000000000000000000
000000000

 Date action due  . . . . :   00000000
 Date action
   completed  . . . . . . :   00000000
 Mark for
   offline storage  . . . :   0          1=Do not mark
                                         2=Mark and keep
                                         3=Mark and delete document content
                                         4=Mark and delete document
 Press Enter to continue.

 F3=Exit     F12=Cancel
```

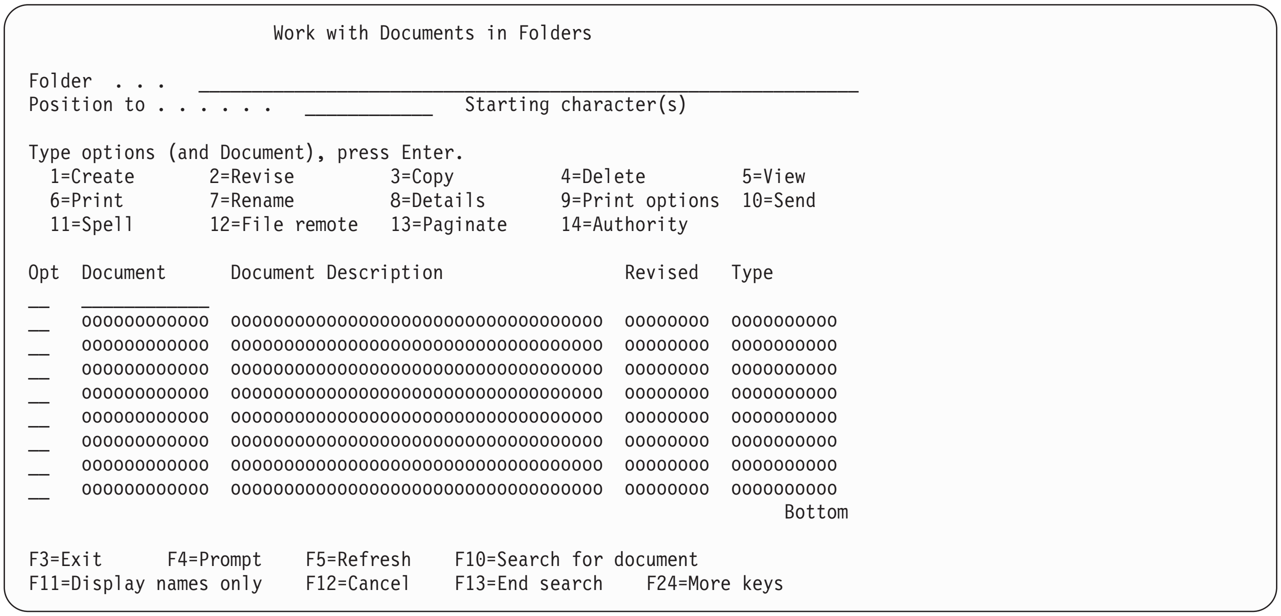
Figure 127. Sample Information Display (Two Pages) in QUSRTOOL

  Press the Help key to view the online help information for the display. The online help information shown depends on where the cursor is located when you press the Help key.

- To view the sample list display, enter:

  `CALL  PGM(your-library/T0011DCL)  PARM(LIST)`

  The sample list display appears:



```
                       Work with Documents in Folders

Folder  . . .   ______________________________________________________________
Position to . . . . . .   ____________   Starting character(s)

Type options (and Document), press Enter.
  1=Create       2=Revise         3=Copy          4=Delete         5=View
  6=Print        7=Rename         8=Details       9=Print options  10=Send
  11=Spell       12=File remote   13=Paginate     14=Authority

Opt  Document      Document Description                 Revised   Type
__   ____________
__   oooooooooooo  ooooooooooooooooooooooooooooooooooo  oooooooo  oooooooooo
__   oooooooooooo  ooooooooooooooooooooooooooooooooooo  oooooooo  oooooooooo
__   oooooooooooo  ooooooooooooooooooooooooooooooooooo  oooooooo  oooooooooo
__   oooooooooooo  ooooooooooooooooooooooooooooooooooo  oooooooo  oooooooooo
__   oooooooooooo  ooooooooooooooooooooooooooooooooooo  oooooooo  oooooooooo
__   oooooooooooo  ooooooooooooooooooooooooooooooooooo  oooooooo  oooooooooo
__   oooooooooooo  ooooooooooooooooooooooooooooooooooo  oooooooo  oooooooooo
__   oooooooooooo  ooooooooooooooooooooooooooooooooooo  oooooooo  oooooooooo
                                                                       Bottom

F3=Exit      F4=Prompt    F5=Refresh    F10=Search for document
F11=Display names only    F12=Cancel    F13=End search    F24=More keys
```

Figure 128. Sample List Display in QUSRTOOL

Press the Help key to view the online help information for the display. The online help information shown depends on where the cursor is located when you press the Help key.

- To view the index search function, do the following:
  1. Type the command to show any of the displays shown in Figure 125 on page 418, Figure 126 on page 418, Figure 127 on page 419, or Figure 128.
  2. Press the Help key to show the help for the display.
  3. Press F11 to use the index search function.
  4. Press F5 to show all the topics available in index search.
  5. You can choose to view any topic shown in the list. The second topic in the list contains a hypertext link. The fourth topic in the list uses the same panel group as the help for the command.
- To view the sample command, enter `T0011CMD`:

  Press the Help key to view the online help information for the command. The online help information shown depends on where the cursor is located when you press the Help key.

Copying the Source for the Example Objects for Your Own Use

You can tailor the example objects for your own use after you copy the source from the QUSRTOOL source file. There are basically two different ways that you can copy the source from the QUSRTOOL source file:

- You can copy one member at a time using the browse/copy services function of SEU.
- You can copy all members from a QUSRTOOL source file (for example, all CL source members contained in QATTCL) at one time using the Copy Source File (CPYSRCF) command.

Defining Special Functions and Attributes for All Displays

The following functions and attributes are *required* for all i5/OS-style displays:

Table 43. Required Functions and Attributes of All i5/OS-Style Displays

Function	Description
Online help information	Available from every display using the Help key and a command attention (CAnn) key as an alternative Help key. The default for the alternative Help key is CA01, which assigns the help function to F1 (the CUA designated Help key).
Blinking cursor	Blinks as long as the record appears on the display screen.
Command function keys CF03 and CF12	Allows the corresponding function keys (F3 and F12) to be active on the display
Help clearing	Ensures that only the current help is available.

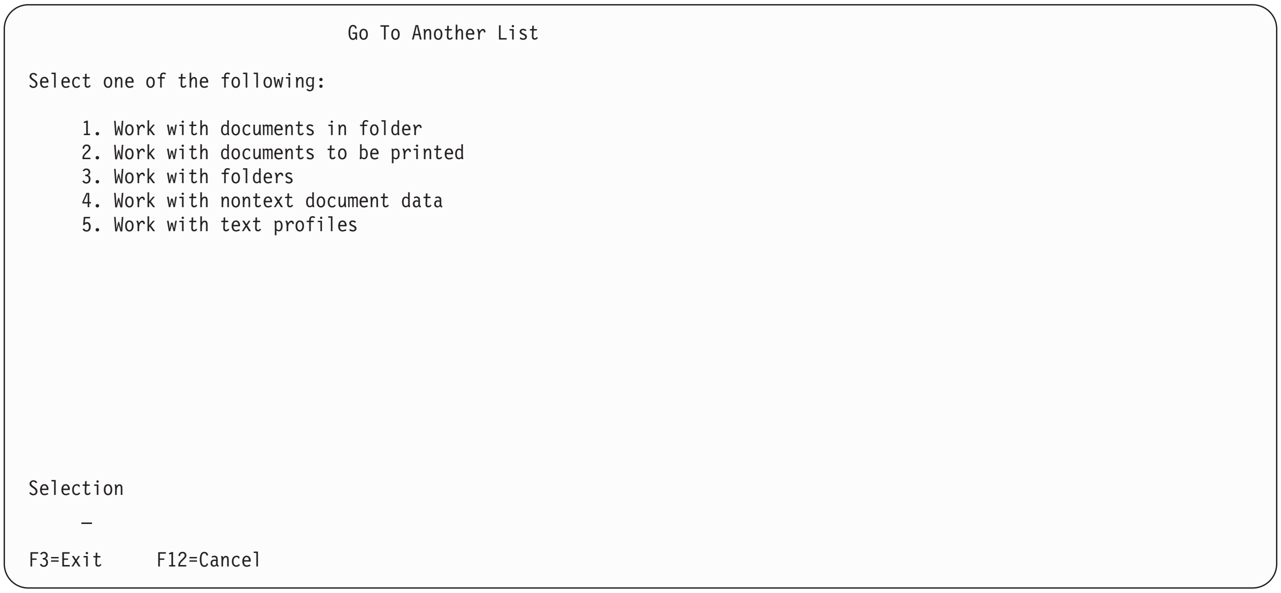
The following functions and attributes are *optional* for all i5/OS-style displays:

Table 44. Optional Functions and Attributes of All i5/OS-Style Displays

Function	Description
Print key	Allows the user to print a display
Page Up and Page Down (Roll Up or Roll Down) keys	Allows the user to page the message subfile
Alternative Page Up and Alternative Page Down (Alternative Roll Up and Alternative Roll Down) keys	Assigns a command function (CFnn) key as an alternative paging key (default keys are CF07 and CF08, respectively)
Keyboard locking	Avoids unlocking the keyboard until the system has finished writing the display and is ready for input.

Designing the Single-Choice Menu Display

An i5/OS system menu shows a list of choices from which the user can select one choice. It always has a title, an instruction, a list of choices, and a labeled field for typing the number of the choice selected. Figure 129 on page 422 shows an example of a menu.



```
                            Go To Another List

Select one of the following:

     1. Work with documents in folder
     2. Work with documents to be printed
     3. Work with folders
     4. Work with nontext document data
     5. Work with text profiles











Selection
     _

F3=Exit     F12=Cancel
```

Figure 129. Sample Application Menu

Before you continue designing this display, define the required and, if desired, optional functions and attributes found in “Defining Special Functions and Attributes for All Displays” on page 421.

For information about viewing this sample display or copying the source for your own use, see “Using the Displays Example in the QUSRTOOL Library” on page 415.

Title

The title should be centered on line 1, in mixed case, and displayed in high intensity.

Instruction Line

The guidelines for an instruction line are as follows:

- The instruction line is located on line 3 and begins in position 2. It tells the user to choose between the options listed on the display.
- The instruction line format for a menu is:

  ```
  Select one of the following:
  ```

  Note that the instruction line ends in a colon.
- All instruction lines on system displays are specified as blue on color displays.
- Leave line 4 blank and begin the menu options on line 5.

Menu Options

The guidelines for menu options are as follows:

- Begin the options following one blank line after the instruction line.
- List the options (one per line) starting in position 7 for options 1 through 9 and position 6 for options 10 and above. The option number is followed by a period and is not highlighted.
- Limit the number of options to a maximum of 10 wherever possible.
- Options are single-spaced and in numeric order. A gap in the numeric sequence is indicated by one blank line, regardless of how many numbered options are missing.
- Capitalize the first letter of the first word of the option text; the remaining words should be capitalized as appropriate for a sentence.

- There is no punctuation at the end of each line of the option text.
- If Sign-off is an option on a menu, it should have option number 90.
- Existing option numbers should not change when new options are added.

Menu Selection Entry Field

The guidelines for the menu selection entry field are as follows:

- The length of a menu selection entry field is one position for option numbers 1 through 9, and two positions for option numbers 10 or greater (up to 99).
- The menu selection entry field is located in a fixed position relative to the function key area. It begins in position 7 on the second line above the top (or only) line of the function key area.
- The selection prompt is on the line above the menu selection entry field and begins in position 2.
- The text of the selection prompt is:

  ```
  Selection
  ```
- The line immediately above the top line of the function key area is blank.

Function Keys

Guidelines for defining the function key area are available in "Defining the Function Key Area for All Displays" on page 443.

Online Help Information

Online help information for this sample display using the user interface manager (UIM) is available in QUSRTOOL. For information on how to use the source in QUSRTOOL, see "Using the Displays Example in the QUSRTOOL Library" on page 415.

Guidelines for defining the online help information are available in "Help for the Menu Display" on page 450.

General Menu Display Operation

Following is a summary of what happens when the user presses the Enter key with or without making any entries on the menu:

- If there is no entry in the selection entry field, the menu remains on the display (no-operation instruction) and an informational message is displayed, if the keyboard is not locked. For example:

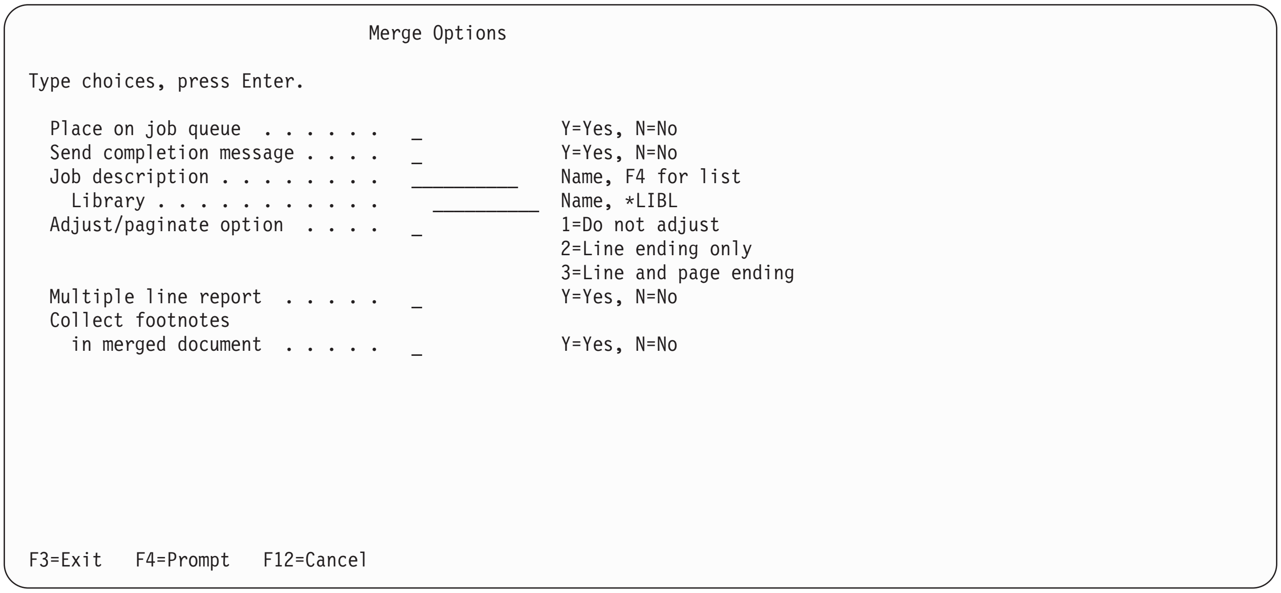
  ```
  Enter option number or command.  Press Help for details.
  ```
- If the value entered was not a valid menu option, a message such as:

  ```
  Value entered is not a valid menu selection
  ```

  should be issued.
- If there is an entry that is determined to be a valid option, the requested action is processed.

Designing the Entry Display

Entry displays let users type in entry field, and are typically used to provide the system parameters and options associated with an action request. Figure 130 on page 424 shows an example of an entry display.



```
                                Merge Options

Type choices, press Enter.

  Place on job queue  . . . . . .   _             Y=Yes, N=No
  Send completion message . . . .   _             Y=Yes, N=No
  Job description . . . . . . . .   __________    Name, F4 for list
    Library . . . . . . . . . . .     __________  Name, *LIBL
  Adjust/paginate option  . . . .   _             1=Do not adjust
                                                  2=Line ending only
                                                  3=Line and page ending
  Multiple line report  . . . . .   _             Y=Yes, N=No
  Collect footnotes
    in merged document  . . . . .   _             Y=Yes, N=No









F3=Exit   F4=Prompt   F12=Cancel
```

Figure 130. Sample Entry Display

Before you continue designing this display, define the required and, if desired, optional functions and attributes found in “Defining Special Functions and Attributes for All Displays” on page 421.

For information about viewing this sample display or copying the source for your own use, see “Using the Displays Example in the QUSRTOOL Library” on page 415.

Title

The title should be centered on line 1, in mixed case, and displayed in high intensity.

Instruction Line

The guidelines for an instruction line are as follows:

- The first character of each instruction line begins in position 2.
- The instruction line for an entry display is a mandatory field in mixed case characters. The format is one of the following, depending on what is to be entered:

```
Type choices, press the Enter key.
Type changes, press the Enter key.
Type information, press the Enter key.
```

- The top instruction line for an entry display ends in a period.
- All instruction lines on system displays are specified as blue on color displays.
- Although a single instruction (one sentence) is recommended, two instructions are allowed. Each instruction should fit on a single line with the first character of each line beginning in position 2. There is no blank line between the instructions. For example,

```
Type choices, press Enter.
Press F16 to send.
```

- A single instruction can have a maximum length of two lines. The line that is carried over is indented two spaces (begins in position 4).

```
Type information to define the..............
  of the distribution list, press Enter.
```

Prompt Area

In general, the columns containing the field prompts and entry fields should be on the left half of the display, and the column containing the list of possible choices should be on the right half of the display. If all the prompts for a request cannot be shown in the prompt area of a single display, make paging of the prompt area possible.

Field Prompts

The guidelines for field prompts are as follows:

- Field prompts are in normal sentence capitalization and are located to the left of the field they identify.
- The first letter of the first word of the prompt should be uppercase; the remaining text should be lowercase, unless grammatically incorrect to do so.
- There is no punctuation at the end of the prompt.
- A series of periods (dots) is used to connect the field prompt and the input field.
- Dots are spaced every other character. The farthest right dot is in the last position within the column width specified for the field prompt, and the dots are placed every other position back to the prompt text. There must be three blank spaces between the farthest right dot and the input field the prompt identifies.
- The closest dot to the prompt text is two or three positions after the prompt (minimum of one blank between prompt text and dot).
- For input fields, there should be room for two or more dots; otherwise dots should not be shown.

  ```
  Prompt . . .   _______        (room for at least 2 dots)
  Long prompt    _______        (no room for 2 dots)
  ```

- Indentation of field prompts and their corresponding input fields is allowed to show a level of hierarchy.
- The field prompt column is separated from the column containing the input field by 3 character positions (blanks).
- When two or more lines of field prompts are presented, the starting positions of the prompts, the dots, and the input fields are aligned.
- Field prompts on entry displays are preceded by an instruction line beginning in position 2. The field prompts start in position 4.

  For example:

  ```
  Type changes, press Enter.
    Customer name . . . . . . . . .   ______________________________
    Customer address  . . . . . . .   ______________________________
    Telephone number  . . . . . . .   _____________
  ```

  The instruction line is separated from the first field prompt by one blank line.

- If field prompt text carries over to a second line, it is indented an additional two spaces. The input field is after the last line.

  ```
  Record format
    of file . . . . . . .   ______________________________
  ```

- The use of leader dots with an unformatted (continuous) entry field that carries over to multiple lines is the same as with standard formatting. If the field carries over to another line, however, it must extend to position 80 and carry over to position 1 on the following line.

  ```
    Cause . . . . .   _____________________________________________________
  __________________________________________________________________________
  ```

- Entry fields formatted on a line-by-line basis should occupy individual lines beginning with the line following the prompt. The entry fields should be indented two positions from the beginning of the prompt.

  The prompt is not followed by leader dots. Never use leader dots unless the field being identified is on the same line.

```
Memo
  ______________________________________________________________________
  ______________________________________________________________________
  ______________________________________________________________________
  ______________________________________________________________________
```

- When a single piece of data to be entered has multiple parts, the parts can be put on one line if they are understood by the user and considered one piece of data.

  ```
  Social security number . . . .   ___   __   ____
  Row/column . . . . . . . . . .   ___   ___
  ```

  A From/To date in a field prompt should be formatted as two fields on one line as follows:

  ```
  Date filed . . . . . . . . From  __/__/__  To  __/__/__  MM/DD/YY
  Document date . . . . . . . From  __/__/__  To  __/__/__  MM/DD/YY
  ```

- Any indication of the type of value (for example, Name), range of value (for example, 1 to 99), or specific value (for example, Y=Yes, N=No) should not appear with the field prompt, but should be shown only in the possible choices area.
- For a qualified name hierarchy, indent *both* the prompt text and the entry field by two positions.
- Indent prompts under a group heading by two positions. A group heading has no corresponding input field. The group heading should be followed by a colon. It can extend to more than one line, but only one level of indent is allowed.
- Do not show an entry field if the value on the display is not meaningful. If entry fields are conditionally meaningful, place conditionally valid prompts on a display separate from the prompt they are dependent on.

  As an alternative, the prompt can be indented under a group heading that defines the conditions. For example:

  ```
        Place on job queue . . . . . . . . .  _          Y=Yes, N=No
          For choice Y=Yes:
            Send completion message  . . . .  _          Y=Yes, N=No
  ```

Entry Fields: A *required* entry field requires the user to type a value. Such a field cannot be supported by a default. An *optional* entry field does not require the user to type a value. The program can always get a value for an optional entry field by defining a default.

Entry fields take two forms. The first form requests a user-supplied value, like a name, descriptive text, or address. Frequently, these fields can accept a value from a list of values of variable length and contents. When such a list of valid choices exists, F4 should be supported to allow the user to request the list and simply choose from it.

```
    File  . . . . . . . . .    __________   Name, F4 for list
```

The second type of entry field supports a selection from a fixed set of choices. CUA calls this a *Selection* field. If only one choice can be selected, the choices should be numbered. For example:

```
    Type style . . . . .  1   1=Prestige elite (12 pitch)
                              2=Courier (10 pitch)
                              3=Essay standard
```

The only exceptions to numbering the choices are when the answers are Yes or No and when the choice value has significance to the user by itself.

If the list of values is longer than five lines, F4 should be supported.

When the prompt requires a Yes or No response, Y and N should be valid entries. For example:

```
    Duplex . . . . . . .  Y   Y=Yes, N=No
```

When a display first appears, the cursor should be positioned on the first entry field that the user will type into. This is usually the first field unless a field specified on an earlier display is carried over to this display.

On some entry displays it is desirable to indicate previous user entries in an entry field. For example, if a user typed some command values prior to requesting prompting, those user-supplied values would be flagged on the prompt.

The greater than symbol (>) is used as the indication. In the entry display format, there are three blank spaces between the last leader dot and the beginning of the input field. The > is placed in the middle space of those three blank spaces.

Rules for Entry Fields: The following rules apply to the entry fields themselves:

- Required fields cannot have a default. A subvalue of a required specification can have a default. For example, a file name parameter is required. No default file name is defined but a default library qualification of the file name can be provided.
- Required entry fields must be displayed in high intensity. The field prompt and possible choices information are in normal intensity. If a value is entered in error the value should be shown again in reverse image and underlined.
- Optional entry fields should be normal intensity.
- Wherever possible, optional entry fields should have a defined and displayed default value.
- Starting positions for entry fields must be left-justified and aligned (unless there are hierarchical fields).
- If defined, default values should appear in the input fields. Values shown in fields are left-justified, including numbers that are displayed as character values (see Figure 130 on page 424). When values shown in fields are specified as numeric, they are right-justified.
- User-supplied values typed in the entry fields override the default values.
- Use the underline attribute to indicate the length of the entry field. The indicated length should be the same as the maximum number of characters that can be entered.
- The entry field can extend into the possible choices information and even carry over into the field prompt area on subsequent lines if needed. If an unformatted, or continuous, field carries over to another line, it must extend to position 80 and carry over to position 1 on the following line. For example:

  ```
   Message text . . . . . .   ______________________________________
  __________________________________________________________________
  __________________________________________________________________
  ```

  If a list of possible choices is appropriate for an extended length input field (see "Descriptive Text Area (Possible Choices Information)" on page 427), the choices may be shown after the input field or accessed through F4.
- Echoed errors (incorrect values in input fields) are displayed again in reverse image. The keyboard should be unlocked so that the user can correct the error without having to press Error Reset before making the correction. The cursor should be positioned at the first field in error.

  Note: If reverse image is used in a highlighted input field (for example, a required field), the highlight attribute should be turned off (changed to normal intensity) to avoid a no display situation with 5250 display stations.
- If a field supports F4 for a list of choices, F4 must remain available if an error is made in the field. This is important in helping the user correct the error.

  If the entry display can be paged and the first error is on the currently displayed fields, show the display as is without special positioning. If the first error is not on the currently displayed fields, position the first error at the top of the area that can be paged.

Descriptive Text Area (Possible Choices Information): Possible Choices information should be shown in the description area to the right of entry fields for which only a predefined set of values or a limited range of values is allowed . If the number of values is too great to be shown, or if the values are dynamic, as in a list of document names, F4 should be supported. The phrase *F4 for list* must appear as

Possible Choices information for that field. (If F4 is supported for all fields on the display, *F4 for list* does not have to appear for each field.) A description of the values should also be provided in the Help information supporting the entry field.

The following rules apply to Possible Choices information:

- Information for all entry fields on a display should have the same starting position.
- Choices are separated by one comma and one blank if shown in a horizontal character string.

```
Y=Yes, N=No
QDKT, QTAPE1, QTAPE2, *SAVF
Name, *ALL
```

  Choices are aligned on the left and on separate lines if shown vertically. No comma or other punctuation is shown at the end of any line.

```
Y=Yes
N=No
1=Prestige elite (12 pitch)
2=Courier (10 pitch)
3=Essay standard (proportional)
4=Essay bold (proportional)
```

- Capitalize the first word on either side of the equal sign, regardless of whether it is the first value, last value, or only value. For example:

```
1-255, Blank=Entire instruction
Blank=Entire instruction, *NONE=None of instruction
```

- Specific values that can be typed are shown in uppercase.

```
       Files . . . . . . . . .   __________   Name, *NONE
```

- Yes and No should be specified by Y=Yes, N=No.
- Use ’generic*’ to indicate that generic names must be followed by an asterisk.

  For example:

```
       Files . . . . . . . . .   __________   Name, generic*
```

- Identify key numeric ranges. For example:

```
       0-99
```

  If the range is not important, do not include it. For example, if there is no limit to the value, do not show anything.

- If a numeric value is being prompted and the value must be in specific units, identify the type of units. For example:

```
  Time in seconds
```

- In a series of possible choices without space to show all the values, show the most useful beginning with the default and ending with an ellipsis.
- Show a word in quotation marks only when the value being prompted needs to be in quotation marks.

Function Keys

Guidelines for defining the function key area are available in “Defining the Function Key Area for All Displays” on page 443.

Online Help Information

Online help information for this sample display using the user interface manager (UIM) is available in QUSRTOOL. For information on how to use the source in QUSRTOOL, see “Using the Displays Example in the QUSRTOOL Library” on page 415.

Guidelines for defining the online help information are available in “Help for the Entry Display” on page 451.

General Entry Display Operation

The guidelines for entry display operations are as follows:

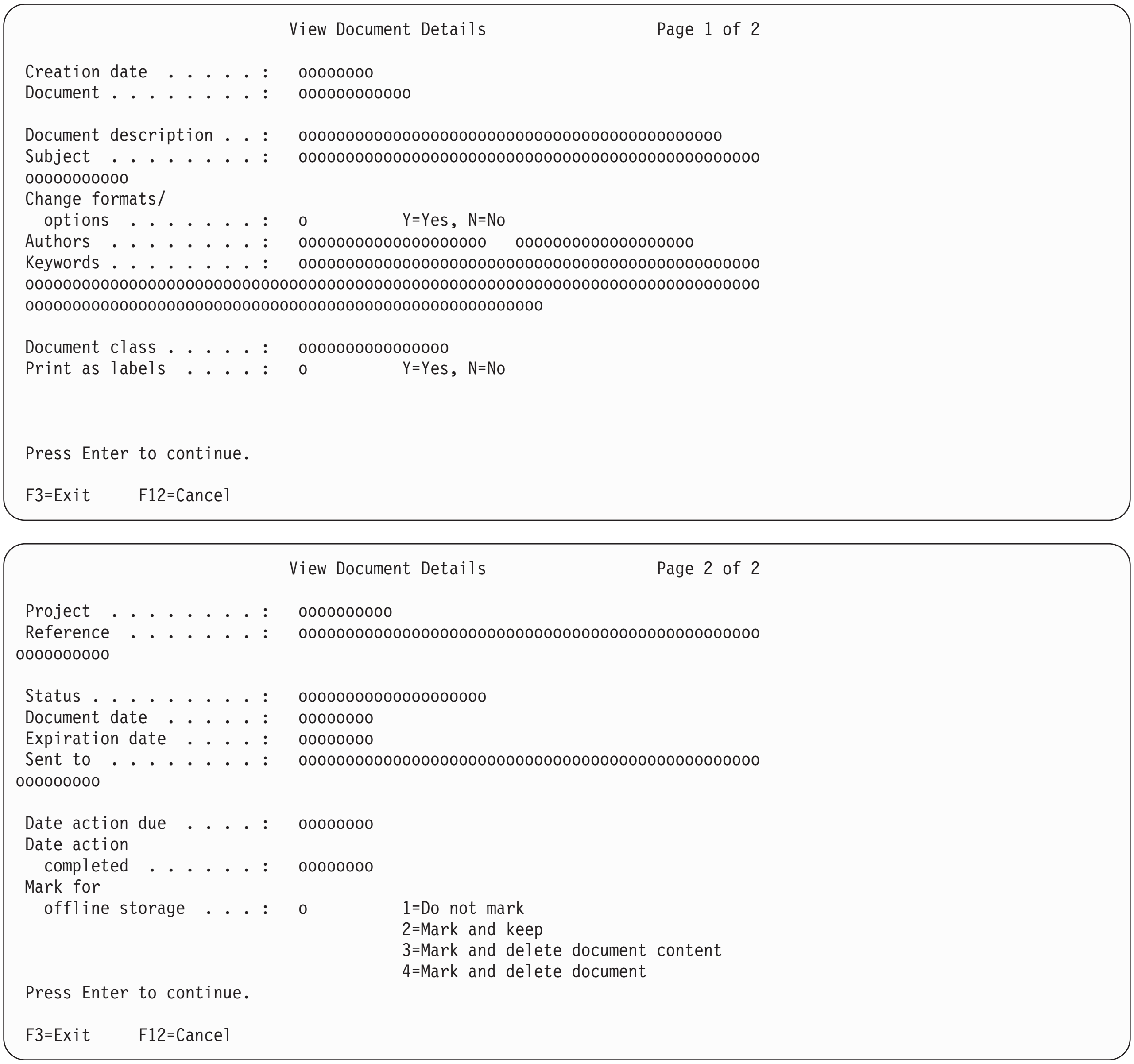
- If a user blanks out a field that has a default, return the default to the field when you show the display again after other checking is completed. *Do not issue an error message*. The function of the display is not performed so the user can accept or change the returned default.
- If a user blanks out an optional field that does not have a default, leave the field blank. *Do not issue an error message*. Processing can occur as appropriate.
- If a user blanks out a required field that has no default, a message such as

  ```
  Library name is required
  ```

  is displayed.
- The other fields on the display screen are checked for errors when a field is blanked out and the Enter key is pressed. In some cases, returning the default may cause an error, depending on another field on the display screen. When these errors occur, error messages should be displayed.
- In all cases, if required values are not supplied or there is no valid entries, the display is shown again with an error message indicating what is needed. Fields that have an entry that is not valid or that require an entry are shown in reverse image. If all fields have valid entries or defaults, the display is processed and the dialog proceeds to the next logical display defined for the Enter action.

Designing the Information Display

Information displays show protected information. Figure 131 on page 430 shows a two-part information display. This example shows a series of output fields that are identified or labeled by field prompts.



```
                             View Document Details                Page 1 of 2

 Creation date  . . . . . :   00000000
 Document . . . . . . . . :   000000000000

 Document description . . :   000000000000000000000000000000000000000000000
 Subject  . . . . . . . . :   0000000000000000000000000000000000000000000000000
 00000000000
 Change formats/
   options  . . . . . . . :   o          Y=Yes, N=No
 Authors  . . . . . . . . :   00000000000000000000   0000000000000000000
 Keywords . . . . . . . . :   0000000000000000000000000000000000000000000000000
 000000000000000000000000000000000000000000000000000000000000000000000000000000
 0000000000000000000000000000000000000000000000000000000

 Document class . . . . . :   0000000000000000
 Print as labels  . . . . :   o          Y=Yes, N=No



 Press Enter to continue.

 F3=Exit     F12=Cancel
```

```
                             View Document Details                Page 2 of 2

 Project  . . . . . . . . :   0000000000
 Reference  . . . . . . . :   0000000000000000000000000000000000000000000000000
0000000000

 Status . . . . . . . . . :   00000000000000000000
 Document date  . . . . . :   00000000
 Expiration date  . . . . :   00000000
 Sent to  . . . . . . . . :   0000000000000000000000000000000000000000000000000
000000000

 Date action due  . . . . :   00000000
 Date action
   completed  . . . . . . :   00000000
 Mark for
   offline storage  . . . :   o          1=Do not mark
                                         2=Mark and keep
                                         3=Mark and delete document content
                                         4=Mark and delete document
 Press Enter to continue.

 F3=Exit     F12=Cancel
```

Figure 131. Sample Information Display (Two Pages)

Before you continue designing this display, define the required and, if desired, optional functions and attributes found in "Defining Special Functions and Attributes for All Displays" on page 421.

For information about viewing this sample display or copying the source for your own use, see "Using the Displays Example in the QUSRTOOL Library" on page 415.

Title

The title should be centered on line 1, in mixed case, and displayed in high intensity.

Location Information

Location information for a multiple-part display is right-aligned on line 1 and is in the form:

```
Page xx of xx
```

Prompt Area 1

The guidelines for field prompts are as follows:

- Field prompts are in normal sentence capitalization and are located to the left of the field they identify.
- Field prompts for output fields are not preceded by an instruction line and begin in position 2.
- A series of periods (dots) and a colon are used to connect the field prompt and the output field.
- The colon is in the last position within the column width specified for the field prompt, and the dots are placed every other position back to the prompt text. The closest dot to the prompt text is two or three positions after the prompt (minimum of one blank between prompt text and dot).
- The field prompt column is separated from the column containing the output field by 3 character positions (blanks).
- When two or more lines of field prompts are presented, the dots and colons on each line are aligned vertically.
- Output fields are always preceded by a colon, and have no underline attributes marking the field length.
- If the prompt requires more than one line, indent the second line two positions. The output field is after the last line.

```
Record format
  of file . . . . . . :   ______________________________
```

- For output fields, dots should be used if there is room for one or more in addition to the colon. The colon is required. There must be one blank space before the colon in the output field format.

```
Prompt . . . :   Xxxxxxx         (room for more than 1 dot)
Med prompt . :   Xxxxxxx         (room for 1 dot only)
Long prompt  :   Xxxxxxx         (no room for 1 dot)
```

- Indentation of field prompts and their corresponding output fields is allowed to show a level of hierarchy.
- When two or more lines of field prompts are presented, the starting positions of the prompts, the dots and colons, and the output fields are aligned.
- The use of leader dots and the colon with an unformatted (continuous) output field that carries over to multiple lines is the same as with standard formatting. The second and subsequent lines, in the case of output fields, are indented by two positions from the beginning of the prompt text.

```
Cause . . . . :   Xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
  xxxxxxxxxxxxx. Xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx.
```

- Output fields formatted on a line-by-line basis should occupy individual lines beginning with the line following the prompt. The output fields should be indented two positions from the beginning of the prompt.

  The prompt is followed by a colon with no intervening blank. The prompt is not followed by leader dots. Never use leader dots unless the field being identified is on the same line.

```
Buckslip:
  Xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
  Xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
  Xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
  Xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
```

- When the information to be shown has multiple parts, the parts can be put on one line if they are understood by the user and considered as one piece of data.

```
Social security number  . . . :    XXX   XX   XXXX
Row/column  . . . . . . . . . :    27    43
```

Prompt Area 2

The prompts on the second part of the information display follow the same format as those on the first part of the display.

Instruction Line

Multipart information displays use an instruction line that tells the user how to continue or end the dialog with the display. For example:

```
Press Enter to continue.
```

This bottom instruction line is placed one blank line after the last line of information or one blank line above the first function key line.

Function Keys

Guidelines for defining the function key area are available in “Defining the Function Key Area for All Displays” on page 443.

Online Help Information

Online help information for this sample display using the user interface manager (UIM) is available in QUSRTOOL. For information on how to use the source in QUSRTOOL, see “Using the Displays Example in the QUSRTOOL Library” on page 415.

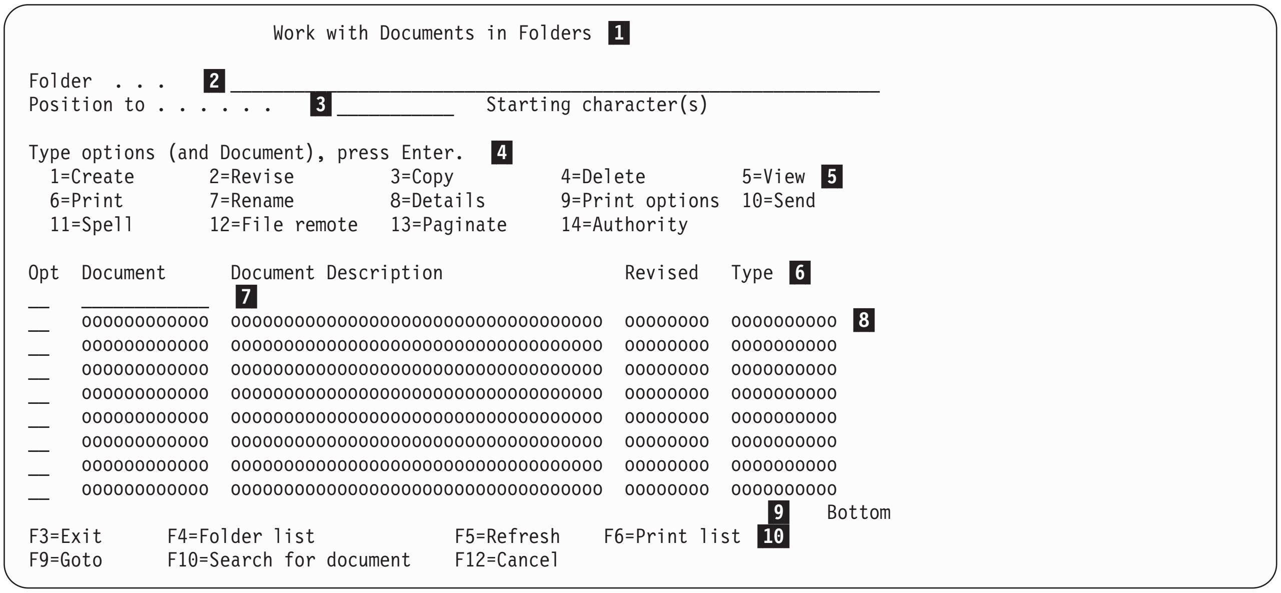
Guidelines for defining the online help information are available in “Help for the Information Display” on page 452.

General Information Display Operation

This output information has been set up as separate information displays that the user progresses through by pressing the Enter key. A paging model can also be used. When the Enter key is pressed, the dialog proceeds to the next logical display. The instruction `Press Enter to continue` should be shown on the display.

Designing the List Display

List displays show a list of items from which users may select one or more and then specify one or more actions to perform on those items. Figure 132 on page 433 is an example of a list display with entry fields. This is called a mixed display. A **mixed display** is a display that combines different types of display elements.



```
                        Work with Documents in Folders 1

Folder  . . .   2 _____________________________________________________________
Position to . . . . . .   3 ___________   Starting character(s)

Type options (and Document), press Enter.  4
  1=Create       2=Revise         3=Copy          4=Delete        5=View  5
  6=Print        7=Rename         8=Details       9=Print options  10=Send
  11=Spell       12=File remote   13=Paginate     14=Authority

Opt  Document      Document Description                 Revised   Type  6
__   ____________  7
__   000000000000  00000000000000000000000000000000000  00000000  0000000000  8
__   000000000000  00000000000000000000000000000000000  00000000  0000000000
__   000000000000  00000000000000000000000000000000000  00000000  0000000000
__   000000000000  00000000000000000000000000000000000  00000000  0000000000
__   000000000000  00000000000000000000000000000000000  00000000  0000000000
__   000000000000  00000000000000000000000000000000000  00000000  0000000000
__   000000000000  00000000000000000000000000000000000  00000000  0000000000
__   000000000000  00000000000000000000000000000000000  00000000  0000000000
                                                                    9    Bottom
F3=Exit      F4=Folder list           F5=Refresh    F6=Print list  10
F9=Goto      F10=Search for document  F12=Cancel
```

Figure 132. Sample List Display

Reference Key	**Display Element**
1	Title
2	List control field
3	*Position to* field
4	Instruction line
5	Options line
6	Column heading
7	Extended action entry area
8	List field
9	Paging location information
10	Function keys

Before you continue designing this display, define the required and, if desired, optional functions and attributes found in “Defining Special Functions and Attributes for All Displays” on page 421.

For information about viewing this sample display or copying the source for your own use, see “Using the Displays Example in the QUSRTOOL Library” on page 415.

Title

The title should be centered on line 1, in mixed case, and displayed in high intensity.

Prompt Area

In Figure 132, the prompt area consists of the list control field and the *Position to* field.

The guidelines for field prompts are as follows:

- Field prompts are in normal sentence capitalization and are located to the left of the field they identify.

- A series of periods (dots) is used to connect the field prompt and the input field. The farthest right dot is in the last position within the column width specified for the field prompt, and the dots are placed every other position back to the prompt text. The closest dot to the prompt text is two or three positions after the prompt (minimum of one blank between prompt text and dot).
- The field prompt column is separated from the column containing the input or output field by 3 character positions (blanks).
- When two or more lines of field prompts are presented, the farthest right dots should be vertically aligned if possible.
- Field prompts that are not preceded by an instruction line (on some mixed displays) begin in position 2.
- Input fields have underline attributes marking their length.

Instruction Line

The guidelines for the instruction line are as follows:

- The first character of an instruction line begins in position 2.
- There is no blank line between an instruction line and a following options line.
- The top instruction line for a list display ends in a period.
- If the instruction line is preceded by any information, such as an entry field, it must be separated from that information by one blank line.
- The instruction line for a standard action list is as follows:

  ```
  Type options, press Enter.
  ```
- The instruction line for an action list with an extended action entry is as follows:

  ```
  Type options (and Xxxx), press Enter.
  ```

  where Xxxx identifies the type of item the list contains, such as Dictionary, Document, and so on.
- When an instruction line followed by an options line is used with a list, the instruction and option combination is separated from the list column headings by one blank line.
- All instruction lines on system displays are specified as blue on color displays.

Options Line

The guidelines for the options line are as follows:

- The options line begins in position 4.
- Each listed option has its first letter in uppercase and is preceded by a number and equal sign with no intervening spaces. A minimum of two spaces must be left between listed options.
- Three lines of options is the maximum (it is recommended that the options line be held to two lines). When options extend to more than one line, the digits are vertically aligned. List displays can have up to three lines of options if preferable to a switch to the F23=More options function.
- Use full word names for option labels (for example, 5=Display jobs, 8=Save library).
- If more options are not shown, put an ellipsis after the last option shown and support and show F23=More options to switch a second set of option descriptions into the instruction area.
- Use the following standardized list action codes:

  1 Select (used on the display resulting from pressing F4). Selects for return of value to entry field where F4 was pressed. Do not use to select an action. The other action codes for the actual action should be used. This is only used when an object or item in the list is just being chosen.

  1 Create. Used on lists with an extended action entry in the list, reserved for Create on all Work with... displays.

  2 Change or Edit.

  3 Copy or Hold.

4 Delete or End.

5 Display or Work with the actual object represented and its content. If it is an object with content, display or work with the content. If it is a description, display or work with the attributes it describes.

6 Print or Release.

7 Rename.

8 Display attributes or Detail (the secondary display option) Work with ... attributes or Detail (the secondary work with).

9 Run.

- Work with... actions should be assigned to 5 or 8 if they are not needed for Display actions. If both are used, 9 or 12 may be used for the work with option.
- The option numbers supported for any list display must be listed in order, but gaps are allowed in the sequence.
- If the option field is 2 characters (because of the number of options), typing a one-digit selection code in either position of the field should be accepted.

Column Headings

The guidelines for column headings are as follows:

- Column headings are required on list displays and must be in mixed case.
- Column headings should be left-justified over the corresponding data if the data is character or character and numeric. Column headings over numeric data only is right-justified. When the heading is longer than the data, it should be centered over the data.
- The column heading for the option field should be *Option*. (*Option* can be abbreviated *Opt* when space is limited.)
- Columns should be separated by no fewer than two spaces and no more than seven spaces.
- Column headings can occupy a maximum vertical size of three lines.

Extended Action Entry Area

In the extended action entry area, the first line under the column headings is an entry area that cannot be paged. The entry area contains an option field and an identifier field. The identifier field is an input field with its length indicated by underline attributes and is the same length as the corresponding output fields on those lines of the list that can be paged. For example:

```
Opt  Document      Document Description                 Revised   Type
__   ___________
```

Both fields have the attributes of entry fields: underline plus reverse image and cursor positioning in case of errors.

List Fields

The farthest left column in the list area is the *Option* column. It contains input fields (underlined) 1 or 2 characters long, depending on the number of options available. One-character entries are accepted in either position for two-character input fields.

The column to the right of the *Option* column should contain the information for identifying the list item or for sequencing (if the list is ordered). This column usually names the item.

Lists can be described using subfiles (see Chapter 4, "Displaying Groups of Records Using Subfiles"). Use of subfiles, however, does not support the descriptive paging information used by the system interface at the bottom of the area to be paged.

In the sample list display shown in Figure 132 on page 433, the fields on the available list lines are described using five fields on a line and nine lines in each field. The program then loads the successive nine-line blocks of data as the user pages through the file.

The fields in the *Option* column have the attributes of entry fields. The fields in the other columns, being defined as input/output (B), have the protect attribute to prevent user access.

Paging Location Information

Whenever the user is on any part other than the last part of an area that can be paged, the phrase `More...` appears highlighted (high intensity) in the location information field on the bottom separator line. When the user is on the last display of an area that can be paged, `More...` is replaced by `Bottom`.

The location information field is defined as the seven farthest right display positions of the bottom separator line. `More...` and `Bottom` are right-aligned in this field.

When multiple areas on a mixed display can be paged, each requires its own paging information. Figure 133 shows the layout of a display with location information.

```
01                                    Display Title
02  Top Separator Line
03  Display Body
04
05  - - - - - - - - - - - - start of paging area - - - - - - - - - - -   - - - -
06
07
08
09
10
11
16
12
13
14
15
17
18
19
20
21  - - - - - - - - - - - - -end of paging area- - - - - - - - - - - -   - - - -
22  Bottom Separator Line                                                  More...
23  Function Key Area
24  Message Area
```

Figure 133. Layout of Display with Location Information

Each allowable constant is conditioned by an indicator. The program must display the correct constant, depending on the status of the paging.

Function Keys

Guidelines for defining the function key area are available in “Defining the Function Key Area for All Displays” on page 443.

Online Help Information

Online help information for this sample display using the user interface manager (UIM) is available in QUSRTOOL. For information on how to use the source on QUSRTOOL, see “Using the Displays Example in the QUSRTOOL Library” on page 415.

Guidelines for defining the online help information are available in “Help for the Information Display” on page 452.

General List Display Operation

The processing priority of a mixed list display when the Enter key is pressed is as follows:

- The list control field (folder, dictionary, and so on) at the top of the display
- The *Position to* prompt
- The extended action entry area
- Options within the list proper

Error diagnosis follows this same sequence.

On this type of mixed list display, the list control field at the top of the display normally contains the name of the object (folder, dictionary, and so on) whose contents are being displayed. By changing the name, for example to another valid folder name, the user is essentially issuing a display command for the contents of the newly specified folder. When the user presses the Enter key, the list display area is then replaced with a new list display.

Operating the List Control Field

The list control field specifies what the list contains. For example, by changing the list control field from FOLDER1 to FOLDER2 and pressing the Enter key, the user gets a new list that displays the contents of FOLDER2.

The guidelines for the operation of the list control field are as follows:

- Once a list of items is displayed, changing the name of the list in the list control field while selections within the list are pending causes an error message. The pending selections are saved and the list control field with the changed name is in reverse image. The error message must include the name that was changed so the user can change the name back, if desired.

  Blanking out the name simply results in the name being restored when the list is displayed again after pending operations are completed.
- If a list of items from folder A is being displayed and there are no pending selections entered in the list, the user may obtain a list of items from folder B by changing folder A to folder B in the list control field and pressing the Enter key.
- While a list of items from folder A is being displayed with no pending requests, the user may perform a specific function on a particular item in folder B by changing folder A to folder B, typing in the selected option number and the identifier in the extended action entry area, and then pressing the Enter key. After the option is performed, the folder B list is displayed.
- If folder A is changed to folder B in the list control field and *the Enter key is not pressed*, and then a page/roll operation is attempted (with or without selections pending), the page/roll is not performed, an error message is shown, and the list control field is in reverse image.

Positioning the List

The guidelines for positioning the list are as follows:

- The Position to function is used for quick repositioning of the list.
  - If the user specifies the character *S* as the starting point, the resulting list starts with the first S item and includes all items after that as well, not just the items beginning with S. When positioning to S, if the list contains AA, BB, SA, SB, the list is positioned to SA.
  - Once the list has been repositioned, the Page Up and Page Down keys can still be used to page to any other items in the list.
  - The special values *TOP and *BOT are supported to position the list to the beginning or end.
  - If no items are found that begin with the characters entered, the list is positioned beginning with the item immediately preceding the requested position.

- After a Position to function, the item to which the list is positioned should be at the top of the list area, the cursor should be on the first option field that follows the input-capable field (if one exists), and the *Position to* field should be blanked out.
- The *Position to* field is not checked for being a valid syntax name or a direct match on a name in the list. Instead, the user is positioned in the list where the entered string would fit based on collating sequence rules (the user can enter `A&` to position to where `A&` is in the collating sequence).
- *Position to* field entries in the list option column are ignored. If the user types characters into this prompt and presses the Enter key while operations are pending in the list area of the list, the list is repositioned and the selections are saved, just as if a page/roll key had been pressed.

Positioning to Lowercase Names in a List

If you support a *Position to* prompt for your list, and the items in the list have lowercase characters, you should follow these rules to let the user position the list to those items:

- If the input in the *Position to* field does not begin with double quotation marks (″), the input should be folded to uppercase and positioned to the name closest to that uppercase name.
- If the input in the *Position to* field begins with double quotation marks (″b or ″b″), the list should be positioned to the name closest to ″b.

  If the user types the ending quotation marks (″b″), you must strip them off so that list positions to the name beginning with the double quotation marks (″b). Otherwise, it will look for ″b″ in the collating sequence.
- If double quotation marks are used on a string that has no special or lowercase characters that would require quotation marks, they are removed, and the list is positioned to the name that is not marked by quotation marks.

  Using the following list,

  ```
  "aa"
  "bb"
  AAAAA
  BBBBB
  ```

  – Position to ″b and Position to ″b″ should both put ″bb″ at the top of the list.

  – Position to ″AAA,″ Position to ″AAA, and Position to AAA should all put AAAAA at the top of the list.

Changing the List Control Field and Positioning the List

Options for positioning the list include:

- If the list control field is changed to a valid value (a new list can be shown), and a value is typed in the *Position to* field, show the new list positioned according to the *Position to* field.
- If the list control field is changed to a value that is not valid and a value is typed in the *Position to* field:

  – Display the same list

  – Reverse image the list control field and show the message

  – *Position to* field remains as typed; list is not positioned

Operating the Extended Action Entry Area

The guidelines for the operation of the extended action entry area are as follows:

- The cursor is initially positioned on the option field of the extended action entry area. The cursor returns to this position after either of the following operations:

  – Position to

  – Any paging operation
- Figure 134 on page 439 shows the processing priority used by the system when the Create function is used with an extended action entry area:

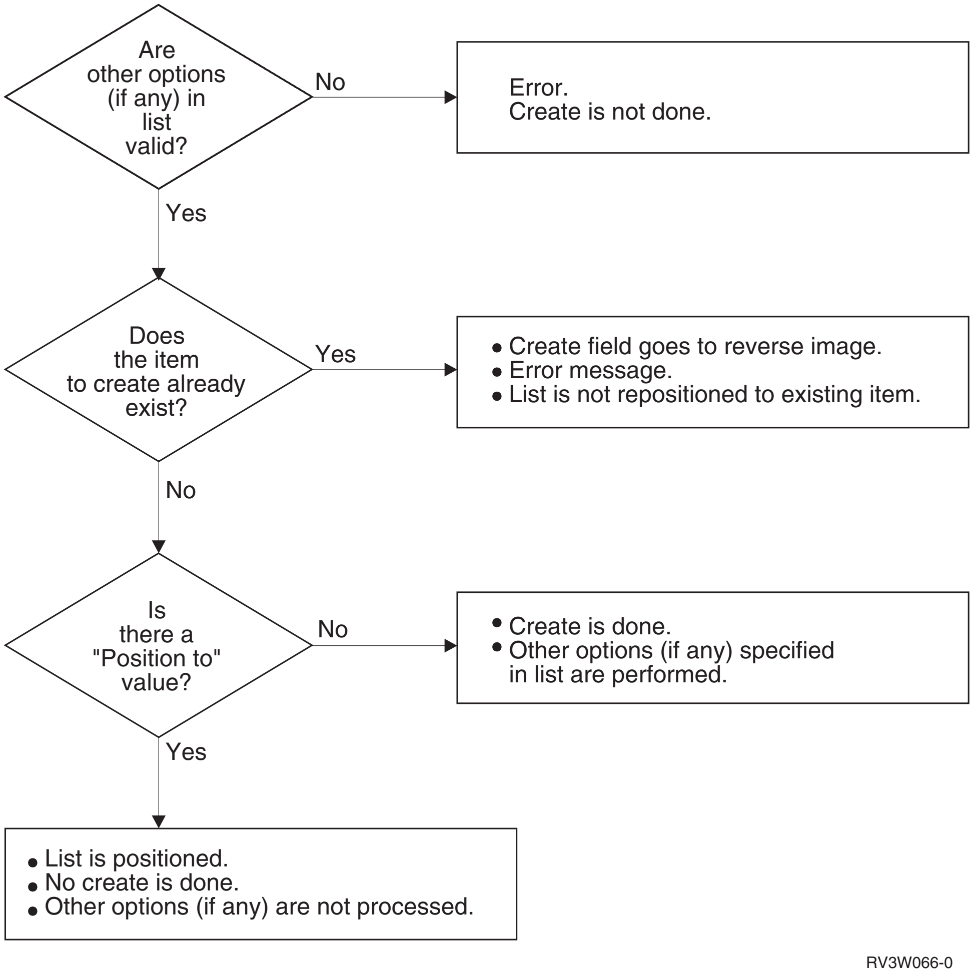

Are other options (if any) in list valid?
No
Error.
Create is not done.
Yes
Does the item to create already exist?
Yes
● Create field goes to reverse image.
● Error message.
● List is not repositioned to existing item.
No
Is there a "Position to" value?
No
● Create is done.
● Other options (if any) specified in list are performed.
Yes
● List is positioned.
● No create is done.
● Other options (if any) are not processed.
RV3W066-0

Figure 134. Example of Processing Priority with List Display

- Specifying the Create option on any list line other than the extended action entry area results in an error.
- Specifying the Create option creates a single new list item identified by the user (using the extended action entry area). If the program requires other user-supplied information to create the list item, it must present the user with an entry display on which to supply that information.
- The user can enter any of the other options valid for that list using the extended action entry area. The user must identify the list item the action is to be performed against by typing the name of that item in the input field on the extended action entry area.
- Options entered in the extended action entry area are valid against any item in the list proper. The user can perform an option against any item identified in the list proper.
- In the extended action entry area, if an option is entered and an identifier is not, an error message should state that no function can be performed unless an identifier is specified. There are two exceptions:
  - The identifier is one of the items prompted for on an accompanying entry display that is always presented to the user
  - The program supports creating a temporary object (for example, a query) that can be named later
- If an identifier is entered and an option is not, no operation is performed against the contents of the identifier field.
- The identifier field can contain a defaulted value. When list processing is performed, the option field in the extended action entry area is restored to blanks. The identifier field, however, can keep a defaulted value.

List Operation When Options Are Specified

The following rules cover the general operation of lists that have options (actions) specified against items (objects) in the list.

List processing should follow these general rules:

- The sequence of list processing should match the sequence in which the list items are presented on the display screen; that is, proceeding from the top line to the bottom line (and from left to right if there is more than a single item on a line).
- After processing, the same view of the list that was presented before processing should be shown again (unless an error occurs).

Specific rules for list operations are as follows:

- The user can choose the same or different options for more than one list item. The choices can be made on any of the item displays that can be paged. Options are not processed until the Enter key is pressed (on the list display itself or on an entry display that was displayed after the Prompt key was pressed).
- There should not be a limit on the number of options that can be typed on a list (except the Select option).
- If the user does not have the proper authority to perform the operation, display such a message on the message line.
- Operations are performed in the order shown in the list area.

  Note: Some operations supported on list displays (Delete is an example) can be grouped together on a separate display when presented for confirmation. Confirmation is given on the group of operations, but then each operation is performed in the order shown in the list.
- When list processing has completed, after having started from a page other than the first, the user is returned to the same page of the list, and the list position does not change (unless an error occurs). However, if the top entry of the page was removed as a result of a delete operation, the list is positioned such that the next remaining entry *prior* to the deleted entry is at the top of the list.

  When processing is completed, after being started from the first page, the list is positioned with the current top entry in the list, even if entries were added to or deleted from the beginning of the list.
- The page/roll keys are used to move forward and backward through the list. Pressing a page/roll key causes a full page roll (all items are replaced).
- Even if a list allows a user to perform only one operation, the user should be allowed to roll even after one item has been chosen, because processing will not occur until the Enter key is pressed.
- When a page/roll key is pressed, any selections made should be saved (the operations should not be performed).
- Lists that can page will not wraparound at the beginning or end of the list.
- If multiple display operations are being performed, they should be done one at a time, with completion of each indicated by the user pressing the Enter key on the last display.
- If the user interrupts list processing before it is finished (for example, by pressing F3 or F12 from an interim display), the list is shown with all processed options blanked out (in the *Option* column). All unprocessed options are still marked. The option that was being processed when F3 or F12 was pressed (the one where the cursor was positioned), is considered processed and, therefore, is unmarked.
- If no processing errors occur, all operations are completed prior to the list being shown again.
- If a processing error is detected, processing is interrupted at that point to allow the user to handle the error. The list can be displayed again with the option field corresponding to the error in reverse image and the cursor positioned to that field. A separate display may also be presented that allows the user to handle the error. The appropriate error message is always presented on the message line. When the user has resolved the error condition, list processing is started again.
- If multiple errors are detected, the list is displayed again with all option fields corresponding to the errors in reverse image. The cursor is positioned to the first field in error.

If the list display can be paged and the first error is on the currently displayed fields, show the display as is without special positioning. If the first error is not on the currently displayed fields, position the first error at the top of the area that can be paged.

- You can elect to have applications either dynamically update the list as the operations in the *Option* column are performed, or wait to display the updated list until the user presses F5=Refresh. Each option is described as follows:
  - A dynamic update of the list can be done if the number of entries in the list is not large and the update can occur as operations are performed without a significant delay.
  - If the number of entries is large and updating the list would result in a noticeable delay, the list should be updated only when the user presses F5=Refresh. If dynamic updating is not used, the entries that the user has operated on (by entering an option next to one or more), and therefore changed, should be annotated as to what operation has been performed on them. The annotation should be made in a status field to the right of the entry. Annotation should only be made by type of operation (for example, operations that change the item or its status, such as Delete or Change, should be annotated; operations such as Display should not.) No annotation should be made on an entry where the operation failed. The annotation should indicate the type of operation performed on the item. Show `(Ended)` if the entry was ended, `(Held)` if held, `(Changed)` if changed, or `(Released)` if released. Abbreviations that correspond to the operation performed, such as `Cnl`, `Hld`, `Chg`, `Rls`, can be used if needed.

Cursor Positioning Rules

Cursor positioning rules are as follows:

- When a list display is initially shown, position the cursor to the option field for the first item in the list. Follow this rule even when there are some secondary fields (for example *Position to*) at the top of the display.
- When a list is presented after a position list or a page/roll function, position the cursor to the first option field in the list area. After a position list function, the item to which the list is to be positioned should be at the top of the list area and the cursor should be in the option field for that item.
- When another view of the list is presented, the cursor should remain on the entry where it was (unless the cursor was not in the list area originally; then it is positioned at the first item in list).
- After selected operations have been performed and the list reappears, the following rules apply:
  - The user should always return to the same page of the list from which the Enter key was pressed. This is the primary rule.
  - Place the cursor by the last item selected. If that item is not on the current display, place the cursor by the first item on the current display.
  - If the list was positioned at the top before the Enter key was pressed, the list should stay at the top when processing is complete. For example, if an item was added to the beginning of the list, position the list to start with the new item. The user expects to see the top of the list if positioned to the top.
  - When errors are detected, position the cursor in the first option in error (options in error should be in reverse image). The list should be positioned again only if the entry that caused the error is not on the current display page.

Error Condition Rules

Error condition rules are as follows:

- When a page/roll key is pressed, option number fields are checked for entries that are valid.
- When an Exit or Refresh function is performed, the entries on the display are not processed, and errors are not returned to the user.
- If errors occur, the option field corresponding to the error is in reverse image, the cursor is positioned to that field, and the appropriate error message appears on the message line.

If the list display is can be paged and the first error is on the currently displayed fields, show the display as is without special positioning. If the first error is not on the currently displayed fields, position the first error at the top of the area that can be paged.

- If the first error is on the currently displayed fields, show the display as is without special positioning. If the first error is not on the currently displayed fields, position the first error at the top of the area that can be paged.
- Each product can have an internal limit for the number of selections that can be made from a list at one time. If the user exceeds this limit, display an error message. This error message should include these items:
  - The fact that the maximum number of allowed selections has been exceeded.
  - The amount by which this maximum number was exceeded (so the user knows how many selections to remove).

  When this message is presented, *the pending selections should not be in reverse image*.
- Issue the following errors if the user attempts to page past the top or bottom of an area that can be paged:
  - Top message

    Message Text: `Already at top of area.`

    Cause: You pressed a key to move backward in an area. However, you cannot move in that direction because you are already at the top of that area.

    Recovery: If you want to move displayed information backward in another area, move the cursor to that area and press either the Page Up key or the Roll Down key again.
  - Bottom message

    Message Text: `Already at bottom of area.`

    Cause: You pressed a key to move forward in an area. However, you cannot move in that direction because you are already at the bottom of that area.

    Recovery: If you want to move displayed information forward in another area, move the cursor to that area and press either the Page Down key or the Roll Up key again.

List Where Only One Item Can Be Selected

If more than one item is selected and the Enter key is pressed:

- List is shown again with all selections still marked
- All selections are in reverse image
- Issue the message: `Only one selection allowed`

If one or more items are selected and a Roll key is pressed:

- No message is issued
- Roll is performed

List Format in Empty List Situation

If the user requests a list of items and there are no items available, the rules are as follows:

1. Start the text in column 4 on the second line following the list area column headers (leave one blank line). If only one list line is available, place the text on the line immediately following the column headers.

```
Opt    Document      Date    Text

 _     _________
          ...blank line...
  (No documents in folder)
```

2. The attributes of the text should be of normal emphasis and normal color (green).
3. If the Help key is pressed while the cursor is anywhere in the list area, processing should be done as though the list is not empty (online help information is available for the columns).

4. The text should be generic and state that the list is empty, but should not state the reason. The text should be in parentheses.
5. The text should begin with No xxxxxxxx where xxxxxxxx identifies what is not displayed. The phrase should not be followed with a period, because it is not a full sentence. For example:

   ```
   (No objects in library)
   ```
6. If information about why the list is empty is desired, present it in a message on the message line.

Defining the Function Key Area for All Displays

The guidelines for the function key area are as follows:

- List active key assignments from left to right in numeric order, beginning at position 2. Active F keys cannot be omitted (with the exception of F1) unless they are displayed by the More keys function, in another set of keys. Capitalize only the first letter of the first word in the function descriptions.
- Leave three spaces between key assignments unless more are needed for alignment.
- When function keys are displayed on two lines, align the key assignments on the left (on F) if possible.

  ```
  F3=Xxxxxxxxx   F4=Xxxxx      F5=Xxxxxxxxxxx   F6=Xxxxxxxxxxxx
  F7=Xxxxxxx     F12=Xxxxxxx   F13=Xxxxxxx
  ```
- Display the keys in ascending numeric sequence (gaps in the sequence can exist).
- If one line of a two-line function key area contains a function key description that is very long (that is, forces a gap of up to seven blanks), this description may span two or more function key descriptions on the other line for the purpose of alignment. For example:

  ```
  F3=Exit   F4=Prompt   F9=Retrieve   F12=Cancel
  F13=Information Assistant           F16=System main menu
  ```
- If all the active keys cannot be displayed in two lines, support F24=More keys. The guidelines for what to display are:
  – Enter, Help, and the Roll or Page keys should not be displayed.
  – Display active function keys in multiple one-line or two-line sets. Use F24=More keys to see the next set of function keys in the function key area of a display. All sets must reserve the same number of lines, one or two.

    ```
    F3=Xxxxxxxxx   F4=Xxxxx      F5=Xxxxxxxxxxx   F6=Xxxxxxxxxxxx
    F7=Xxxxxxx     F12=Xxxxxxx   F24=More keys
    ```
  – F3=Exit and F12=Cancel must be shown in the primary set (and only in that set). Only F24 is displayed in all sets.
  – Although the function keys should be in numeric sequence in each set, the application can determine which keys are in each set. The functions displayed in the primary set should be those that are used the most.
- Displayed function keys on system displays are specified as blue on color displays.

Optional Command Line and Identifier Field

You have probably seen other *system* menus with a slightly different format. Figure 129 on page 422 shows a menu from within an application, and is the type of menu you should create. System menus have an additional *system identifier* in the upper left corner. This is used as a name in conjunction with the GO command. Such names have to be present as object names in system tables.

The system uses names in this identifier field only on menus accessed by the GO command. Using this identifier field on application menus could confuse the user because the names would not work with the GO command. A user can access the system menus, given the menu identifier. Using the GO command, a user can specify a particular menu, by the menu identifier, or, if a particular menu is not known, a generic identifier can be used. In this case, the user is shown the Work with Menus display and from this list, can specify a menu to run. For more information on how to create display file menus, see Chapter 9, “Creating and Accessing Menus Using Display Files,” on page 235.

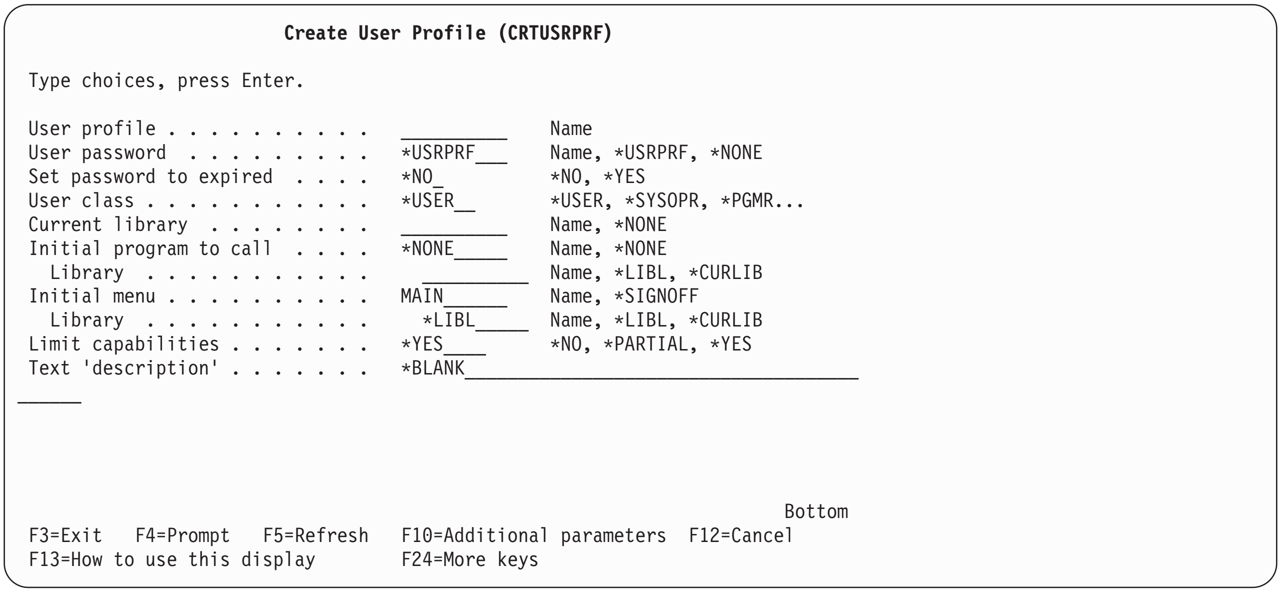
System menus also use a command line as the entry field for menu selection. The only command area support DDS provides is to allow you to define an entry field (with accompanying field prompt) into which a command could be entered. You write all code to support this entry area as a command area, including passing the command to the system, handling the Prompt function, handling the Retrieve function, and handling help for the command. “Available Command Line Tool” on page 446 provides information on a command line function available in the QUSRTOOL library to assist in putting a command line on an application display.

To support commands from within an application, make one or more system displays with a command line available or provide a command line on one or more application displays.

Note: If the commands change something the application is dependent on, results that cannot be predicted can occur.

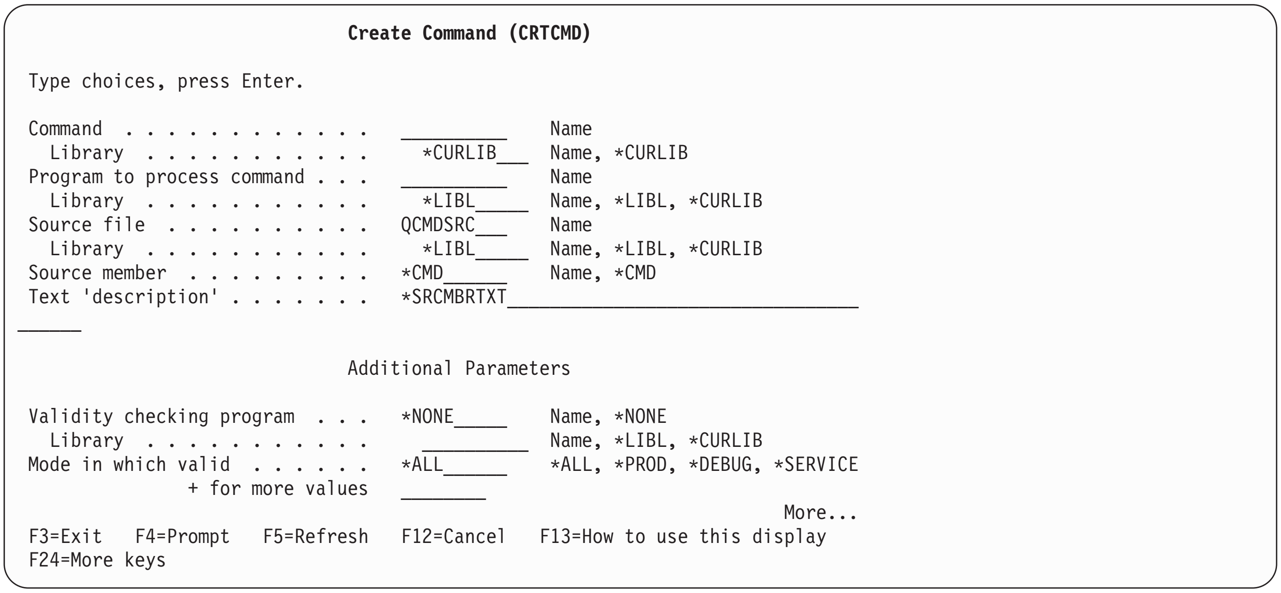
With the i5/OS system you can limit command entry to the system commands and application commands you want. Typically, this is the best approach for the user. You can limit the user to special commands by specifying *YES for the *Limit capabilities* prompt on the Create User Profile (CRTUSRPRF) display, as in Figure 135. You can then create commands with the Create Command (CRTCMD) command. Specify *YES for the *Allow limited users* prompt to make the command available to limited users as in Figure 136 on page 445 and Figure 137 on page 445.

Note: Most commands are shipped with ALWLMTUSR(*NO). See the *Limit capabilities* (LMTCPB) prompt on the Create User Profile (CRTUSRPRF) display for a list of ALWLMTUSR(*YES) commands. You can change IBM-supplied commands with the Change Command (CHGCMD) command. If you change an IBM-supplied command you must keep track of your changes in a control language (CL) program because the commands are replaced with each release.



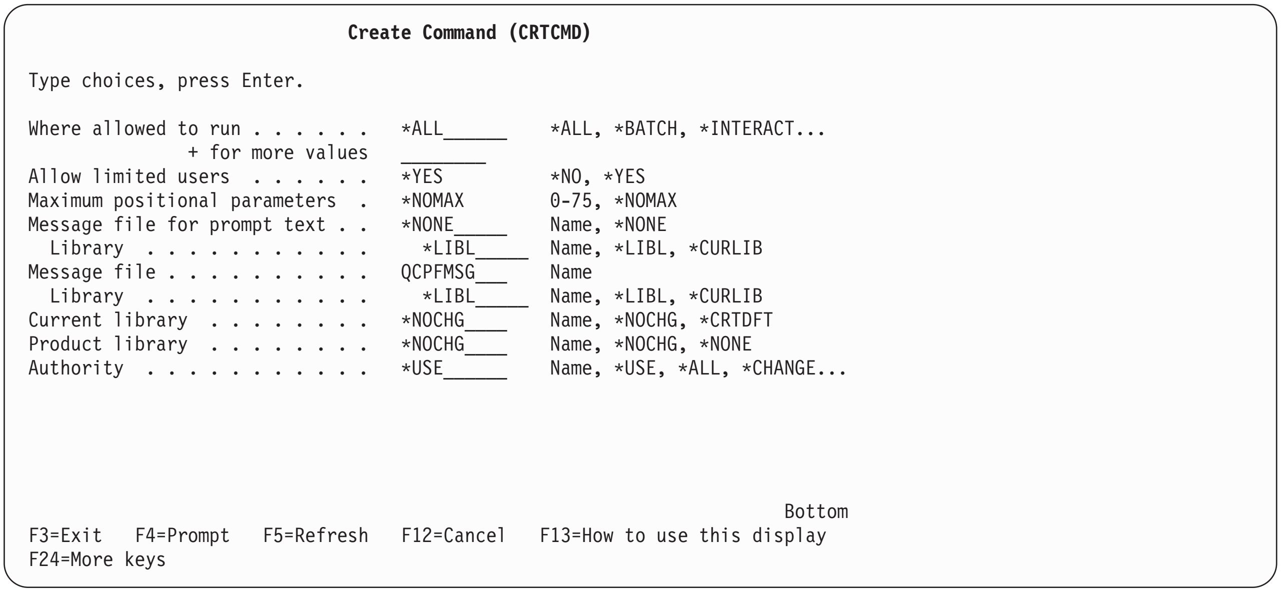
```
                          Create User Profile (CRTUSRPRF)

 Type choices, press Enter.

 User profile . . . . . . . . . .   __________    Name
 User password  . . . . . . . . .   *USRPRF___    Name, *USRPRF, *NONE
 Set password to expired  . . . .   *NO_          *NO, *YES
 User class . . . . . . . . . . .   *USER__       *USER, *SYSOPR, *PGMR...
 Current library  . . . . . . . .   __________    Name, *NONE
 Initial program to call  . . . .   *NONE_____    Name, *NONE
   Library  . . . . . . . . . . .     __________  Name, *LIBL, *CURLIB
 Initial menu . . . . . . . . . .   MAIN______    Name, *SIGNOFF
   Library  . . . . . . . . . . .     *LIBL_____  Name, *LIBL, *CURLIB
 Limit capabilities . . . . . . .   *YES____      *NO, *PARTIAL, *YES
 Text 'description' . . . . . . .   *BLANK_____________________________________
______




                                                                       Bottom
 F3=Exit   F4=Prompt   F5=Refresh   F10=Additional parameters  F12=Cancel
 F13=How to use this display        F24=More keys
```

Figure 135. Create User Profile Entry Display



```
                              Create Command (CRTCMD)

 Type choices, press Enter.

 Command  . . . . . . . . . . . .   __________    Name
   Library  . . . . . . . . . . .     *CURLIB___  Name, *CURLIB
 Program to process command . . .   __________    Name
   Library  . . . . . . . . . . .     *LIBL_____  Name, *LIBL, *CURLIB
 Source file  . . . . . . . . . .   QCMDSRC___    Name
   Library  . . . . . . . . . . .     *LIBL_____  Name, *LIBL, *CURLIB
 Source member  . . . . . . . . .   *CMD______    Name, *CMD
 Text 'description' . . . . . . .   *SRCMBRTXT_________________________________
______

                              Additional Parameters

 Validity checking program  . . .   *NONE_____    Name, *NONE
   Library  . . . . . . . . . . .     __________  Name, *LIBL, *CURLIB
 Mode in which valid  . . . . . .   *ALL______    *ALL, *PROD, *DEBUG, *SERVICE
               + for more values    ________
                                                                        More...
 F3=Exit   F4=Prompt   F5=Refresh   F12=Cancel   F13=How to use this display
 F24=More keys
```

Figure 136. Create Command Display with Additional Parameters Selected



```
                              Create Command (CRTCMD)

 Type choices, press Enter.

 Where allowed to run . . . . . .   *ALL______    *ALL, *BATCH, *INTERACT...
               + for more values    ________
 Allow limited users  . . . . . .   *YES          *NO, *YES
 Maximum positional parameters  .   *NOMAX        0-75, *NOMAX
 Message file for prompt text . .   *NONE_____    Name, *NONE
   Library  . . . . . . . . . . .     *LIBL_____  Name, *LIBL, *CURLIB
 Message file . . . . . . . . . .   QCPFMSG___    Name
   Library  . . . . . . . . . . .     *LIBL_____  Name, *LIBL, *CURLIB
 Current library  . . . . . . . .   *NOCHG____    Name, *NOCHG, *CRTDFT
 Product library  . . . . . . . .   *NOCHG____    Name, *NOCHG, *NONE
 Authority  . . . . . . . . . . .   *USE______    Name, *USE, *ALL, *CHANGE...




                                                                        Bottom
 F3=Exit   F4=Prompt   F5=Refresh   F12=Cancel   F13=How to use this display
 F24=More keys
```

Figure 137. Second Display of Additional Parameters

If your application uses commands, you should define an entry field (with accompanying field prompt) on the display. To match the i5/OS system, command lines should appear as follows:

```
Selection or command
===> _________________________________________________________________________
```

- The one-line command line begins in position 7 on the line immediately above the function key area and ends in position 79 on the same line.
- The command line is identified by a command line prompt, beginning in position 2. If the command line is used to enter either the menu selection or a command, the text of the prompt is:

  `Selection or command`

- The command entry arrow (===>) is in positions 2 through 5 of the first line of the command line.
- The command entry arrow (===>) is normal intensity like the command line.

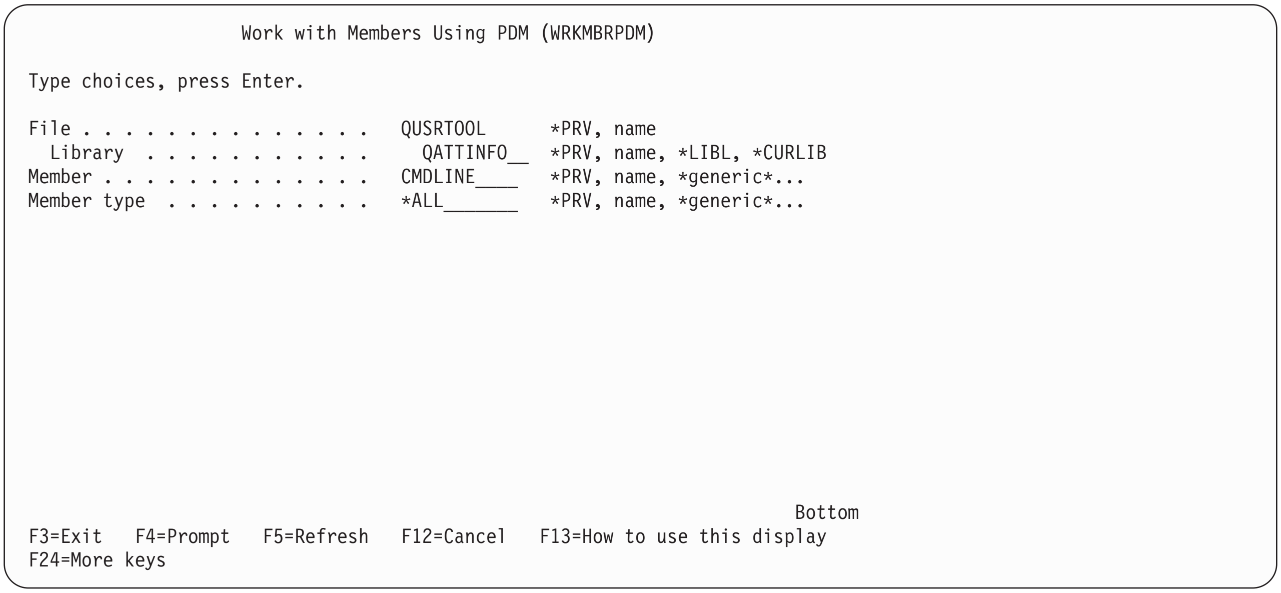
Available Command Line Tool

General-Use Programming Interface

The QUSCMDLN program can be used to display a pop-up window that contains a command line. The command line can be used to enter system commands. QUSCMDLN can be called by any user program. More information about QUSCMDLN is available in the **APIs** topic

End of General-Use Programming Interface

The QUSRTOOL library on the i5/OS system contains a command line function and documentation to assist in putting a command line function on an application display (member CMDLINE, in file QATTINFO, in QUSRTOOL). One way to access this function is by using the Work with Members Using PDM (WRKMBRPDM) display, shown in Figure 138.



```
                    Work with Members Using PDM (WRKMBRPDM)

Type choices, press Enter.

File . . . . . . . . . . . . . .   QUSRTOOL      *PRV, name
  Library  . . . . . . . . . . .     QATTINFO__  *PRV, name, *LIBL, *CURLIB
Member . . . . . . . . . . . . .   CMDLINE____   *PRV, name, *generic*...
Member type  . . . . . . . . . .   *ALL_______   *PRV, name, *generic*...

















                                                                        Bottom
F3=Exit   F4=Prompt   F5=Refresh   F12=Cancel   F13=How to use this display
F24=More keys
```

Figure 138. Work with Members Using PDM Entry Display

Common Key Assignments

Table 45 on page 447 summarizes common function key assignments and indicates when each key is used. The keys identified as CUA can be used only for the function defined. If the function is not available, the key should not be used for another function.

Non-CUA keys are keys commonly used on the system for a particular function. If the function is available on a display, the key assignment given should be used for compatibility with system key assignments. If the function is not being used, you can use the key assignment for an application-specific function.

Table 45. Function Key Assignments

Name	Key	CUA	Usage/Function
Help	F1	Yes	Active on all displays, but not shown on the display. F1 is also mapped to the Help function but is not displayed in the function key area.
Set 1/Set 2	F2	Yes	Required when applications need more function keys than can be assigned to the 24 numbered keys. F1=Help, F2=Set n, F3=Exit, F9=Retrieve/command, and F12=Cancel are required in both sets of function keys.
Exit	F3	Yes	Usually active and shown on all displays. F3 exits the application or a unit of work within the application. If the application is exited, the user is returned to the display from which the application was requested. If a unit of work within an application is exited, the user is returned to a predefined point within the application that serves as a point of focus for requesting work (usually a primary menu). User input is normally discarded; however, if pressing the Exit key could cause loss of data and require extensive user effort to retrieve or reconstruct, an exit display should be provided to give the user the option of saving the data before exiting. The Exit function must be available on all displays except: • Displays used for confirmation of the delete action • List display shown as a result of pressing F4 • Exit displays provided to give the user the option of saving the data before exiting The following describes the use of the Exit key for specific display types: • **Menu**. If the menu is part of a dialog that began with a primary menu or the initial menu after sign-on (for example, the primary or initial menu is the first menu on the current menu stack), returns to this primary or initial menu. Pressing the Exit key from the primary menu exits the current dialog and returns to the display from which the dialog was entered. • **List display**. Returns to the menu or other display from which the list display was requested. User input is discarded. • **Entry display**. Returns to the menu or other display from which the entry display (or set of entry displays) was requested. User input is discarded. However, an exit display can be provided to give the user the option of saving the entries or changes. • **Information display**. Returns to the menu or other display from which the information display (or set of information displays) was requested. • **Any help display**. Returns to the display from which help was originally requested.

Table 45. Function Key Assignments (continued)

Name	Key	CUA	Usage/Function
Prompt	F4	Yes	Should be supported whenever the display contains an entry field for which the valid entries are known but are not shown on the display. F4 provides a list of selectable values (usually a list of names) related to the field where the cursor is currently positioned. When the user selects one or more values from the list, those values are used as input values for the field. If F4 is pressed and the function is not supported for the field containing the cursor, the user receives an error message. If the function is supported for the field containing the cursor and either a default value exists or the user makes an entry in the field, the value in the field is kept and the list is displayed. If the user selects a value from the list, this value replaces the previous value in the field. If the user does not select a value, the previous value is kept. If the user leaves the field blank and presses F4, the list is also displayed for a value to be selected. When a command is typed on a system command line and F4 is pressed, a prompt for the command (an entry display with input fields for the command parameters) is displayed.
Refresh	F5	Yes	Should be used where the Refresh function is appropriate. F5 can start resetting defaults on an entry display, or update (refresh) system output being displayed. The using program determines what is appropriate depending on the display type: • **Menu and help display**. Not usually used. • **Entry display**. Clears user input and shows the display again with the original values provided at the time the display was first shown. The cursor is repositioned at the first logical input field on the display. If the entry fields are on a display that can be paged, Refresh should position the cursor at the beginning of the list of entry fields. • **List display**. Clears user input and updates the display to show the current status. Refresh should maintain the same position in the list. The list should be shown again with the entry that was at the top of the list page when F5 was pressed still at the top of the list page, and with the cursor positioned at the top list item. • **Display with output-only fields (data output items)**. Updates the information variable fields to show the current values. If the information variable fields are in a display that can be paged, Refresh should maintain the same position in the list of fields.

Table 45. Function Key Assignments (continued)

Name	Key	CUA	Usage/Function
			Summary of list positioning rules with the Refresh function: 1. The display is shown again with the entry that was at the top of the display still at the top of the display. 2. If that entry no longer exists, the entry following it is shown at the top of the display. 3. When the first entry in the list was at the top, the first entry is shown at the top again (even though it may be a new first entry). 4. If the entry that was at the top of the display no longer exists and there are no entries following it, the display is shown with the first entry of the entire set of values at the top of the display. 5. If the location-determining attributes of the entry at the top of the display have changed, the list is shown with the entries starting where the old entry would have been.
Create	F6	No	Optional. F6 is used for the Create function on a list display that has not converted to the combination list with 1=Create. However, using a combination list is preferred on the i5/OS system. Summary of uses for F6: • May be requested with the cursor in any position. • Go to separate display in which additional object(s) can be defined. • On completion, show list display from which Create request was made again. • Show created object(s) sorted into the list. • Requests that were pending at the time F6 was pressed should not be processed, but should still be pending. **Note:** When a very large number of new entries can be added, such as with Add Entries to a Distribution List, it may be a good application decision to not allow an Add or Create function until other pending requests are completed. In this case, a message can be issued when F6 is requested and other requests are pending.
Retrieve	F9	Yes	Required on all panels or pop-up windows with a command line.
Command	F9	Yes	Used to display a pop-up window containing a command line.
Cancel	F12	Yes	Must be active and shown on all displays. F12 returns to the previous logical display. It provides the basic navigation function of backing up one step in the interface, as described for the following displays: • **Menu**. Backs up to the previous menu or other display from which the menu was directly started. Any user input is discarded. • **Entry and List displays**. Backs up to the previous display. User input is discarded. (If significant data would be lost, an exit display should be shown.) • **Information displays**. Backs up to the display previously shown. **Note:** If a set of information (including input or output fields) can logically be presented in a paging format, use a display that can be paged instead of the Enter and Cancel keys for forward and backward navigation through the information. This function must be available on all displays, except when a logical Cancel function is precluded by other factors. An example is when only the Exit function is appropriate because the user must first verify (using an exit display) whether data is to be saved or not.

Table 45. Function Key Assignments (continued)

Name	Key	CUA	Usage/Function
XXXX Main Menu	F16	No	Optional for applications. F16 must be active and shown on system menus. It provides a direct path to the system or application primary menu. A target menu other than the system main menu must be explicitly stated. This key would normally be used within an application to provide a direct path to the application main menu. When this key is pressed, any user input is discarded. If F3 or F12 is pressed on the target menu, the user is returned to the display where F16 was pressed.
Top of list	F17	No	Optional on displays that can be paged. F17 pages to the top or beginning (same as *TOP in the *Position to* field). This key should be supported when the *Position to* field is not present.
Bottom of list	F18	No	Optional on displays that can be paged. F18 pages to the bottom or end (same as *BOT in the *Position to* field). This key should be supported when the *Position to* field is not present.
Left	F19	Yes	Should be operational on all displays with a horizontal area that can be paged. There are no common columns of information between successive views.
Right	F20	Yes	Should be operational on all displays with a horizontal area that can be paged. There are no common columns of information between successive views.
More options	F23	No	Optional. List displays only. Use F23 to display the next set of options in the instruction area of a list display. Use when the first set of options shown in up to three lines is not complete. It is recommended that an ellipsis be shown after the last option in the options area if other options exist that cannot be seen until F23 is pressed. The most important functions are shown first, and the less frequently used options are shown after F23 is pressed.
More keys	F24	No	Optional. Use F24 to see more active function keys in the function key area of a display. Use when all key descriptions will not fit in the two-line function key area.

Defining Help Information for All Displays

Help for i5/OS system displays consists of specific information on fields where the cursor is positioned, and, when the cursor is positioned anywhere else, general information on what the display is for and how the user interacts with it.

Help on the system also includes a system help index. This section describes the type of help support for fields provided by the system for each display type.

Help for the Menu Display

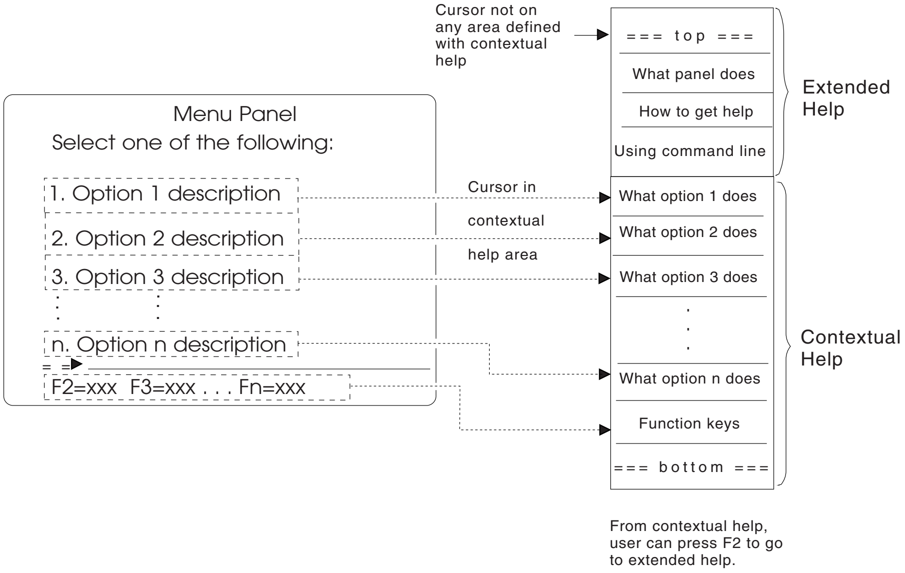
The following table and illustration describe the help information for each menu help area:

Table 46. Type of Help for Each Help Area-Menu Display

Cursor Position	Help Information	UIM Tag
On line containing option number and description	What function each option performs	HELP element of MENUI tag
In function key area	Function performed by each function key turned on for the display	HELP element of KEYL and KEYI tag

Table 46. Type of Help for Each Help Area-Menu Display (continued)

Cursor Position	Help Information	UIM Tag
Any other display position	Beginning of general help for display (user can page through all help for display, which includes description of how to use the entry line)	HELP element of PANEL tag
On command line with no entries	Entry at beginning of general help for panel (user can page through all help for display)	HELP element of PANEL tag
On command line with option number entered	What function each option performs	HELP element of MENUI tag



RV2W027-4

The following table gives the DDS considerations for help on menu displays:

Table 47. DDS Considerations-Help on Menu Displays

Item Help Area	Area Covered
Each menu option	From position 1 to the farthest right display position on the line containing the option number and the option text
Function key area	From position 1 on the first line of the function key descriptions to the farthest right display position on the last line of the function key descriptions

The remainder of the display is defined as the general help area.

Help for the Entry Display

The following table and illustration describe the help information for each entry display help area:

Table 48. Type of Help for Each Help Area-Entry Display

Cursor Position	Help Information	UIM Tag
On input field or descriptive lines associated with input fields (data entry item)	Use of each possible entry in input field	HELP element of DATAI tag

Table 48. Type of Help for Each Help Area-Entry Display (continued)

Cursor Position	Help Information	UIM Tag
In function key area	Function performed by each function key turned on for the display	HELP element of KEYL tag
Any other display position	Beginning of general help for display (user can page through all help for display)	HELP element of PANEL tag

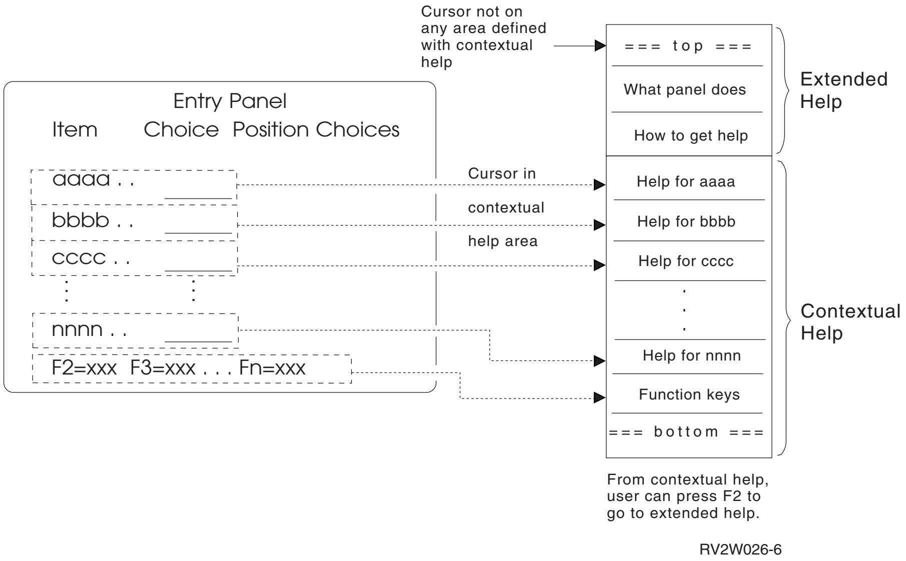


Figure 139. Help Areas for Entry Displays

The following table gives the DDS considerations for help on menu displays:

Table 49. DDS Considerations-Help on Entry Displays

Item Help Area	Area Covered
Each item	From position 1 on the first line of the item text to the farthest right display position on the last line of either (1) the input field(s) or (2) the possible choices text, whichever occupies the most lines (see Figure 139).
Function key area	From position 1 on the first line of the function key descriptions to the farthest right display position on the last line of the function key descriptions

The remainder of the display is defined as the general help area.

Help for the Information Display

The following table and illustration describe the help information for each information display help area:

Table 50. Type of Help for Each Help Area-Information Display

Cursor Position	Help Information
In output field or descriptive lines associated with output field.	Meaning of output field. (A **data output field** is an area on a display consisting of descriptive text and an associated output field into which variable data is written at display time.)
In function key area	Function performed by each function key turned on for the display
Any other display position	Beginning of general help for display (user can page through all help for display)

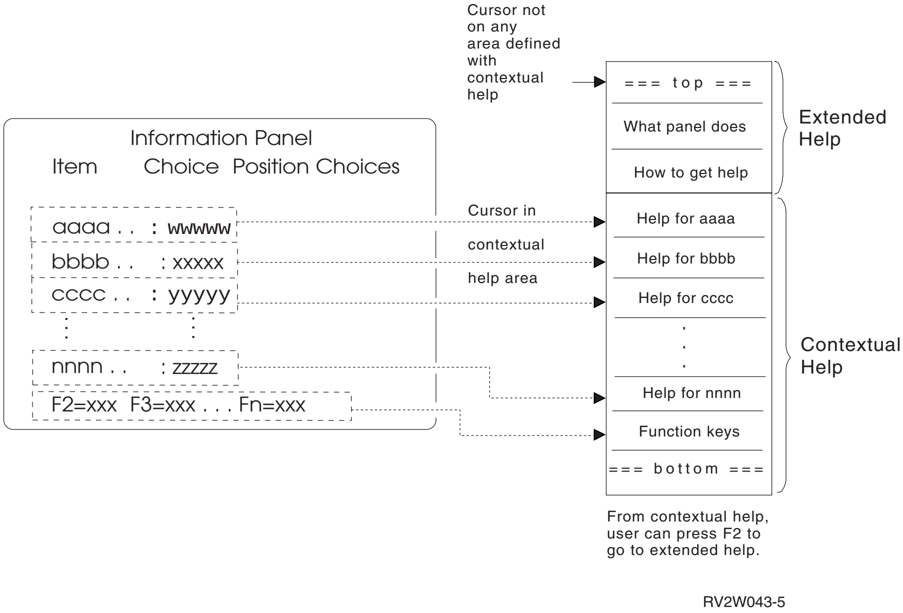


Figure 140. Help Areas for Information Displays

The following table gives the DDS considerations for help on information displays:

Table 51. DDS Considerations-Help on Information Displays

Item Help Area	Area Covered
Data output items	From position 1 to the farthest right display position on the line (see Figure 140)
Function key area	From position 1 on the first line of the function key descriptions to the farthest right display position on the last line of the function key descriptions

The remainder of the display is defined as the general help area.

Help for the List Display

The following table and illustration describe the help information for each list display help area:

Table 52. Type of Help for Each Help Area-List Display

Cursor Position	Help Information	UIM Tag
In input field or descriptive lines associated with input fields	Use of each possible entry in input field	HELP element of LISTCOL tag
In specific column	Meaning and use of column. This includes how to use that entry field (for example, creating a list entry or performing an action against a list entry without paging to it) for the column containing the extended action entry.	HELP element of LISTCOL tag
In function key area	Function performed by each function key turned on for the display	HELP element of KEYL tag
Any other display position	Beginning of general help for display (user can page through all help for display)	HELP element of PANEL tag

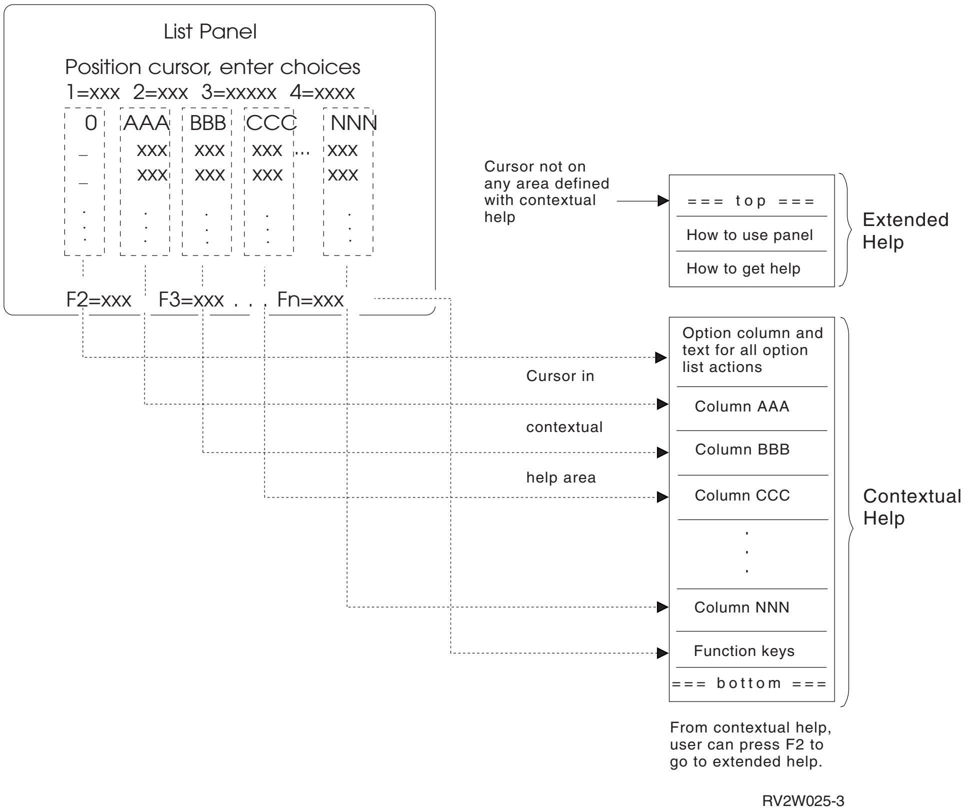

List Panel
Position cursor, enter choices
1=xxx 2=xxx 3=xxxxx 4=xxxx
0
AAA
BBB
CCC
NNN
xxx
F2=xxx F3=xxx . . . Fn=xxx
Cursor not on any area defined with contextual help
=== top ===
How to use panel
How to get help
Extended Help
Cursor in contextual help area
Option column and text for all option list actions
Column AAA
Column BBB
Column CCC
Column NNN
Function keys
=== bottom ===
Contextual Help
From contextual help, user can press F2 to go to extended help.
RV2W025-3

Figure 141. Help Areas for List Displays

The following table gives the DDS considerations for help on list displays:

Table 53. DDS Considerations-Help on List Displays

Item Help Area	Area Covered
Each column	From the farthest left position on the first line of the column heading to the farthest right position on the last line of column data. The farthest left position is defined as the farthest left character position of either the longest heading line or the data column, whichever is wider. The farthest right position is defined as the farthest right character position of either the longest heading line or the data column, whichever is wider. (See Figure 141).
Function key area	From position 1 on the first line of the function key descriptions to the farthest right display position on the last line of the function key descriptions

The remainder of the display is defined as the general help area.

Defining and Presenting Messages

For compatibility with the system, an application should present an error message any time a user types a selection or actual value that is not allowed. The message itself should provide as much information as possible to allow users to continue processing. On the system, messages appear on the bottom line of the display.

Messages should be supported by help information that gives an expanded description of the error (if necessary) and a remedy to the problem. In general, the help information should state what happened and, if possible, what to do if an action is required. The highest priority should be given to a statement of action required.

The user can request help for the message being displayed by moving the cursor to the message line and pressing the Help key. Message help is presented on a separate display and can extend to more than one display if needed.

The DDS keyword ERRMSG is easy to use, but locks the keyboard and lets the user see only one message at a time. This option is acceptable if there are only a few messages. However, to keep the user's interaction with the message easy, the goals are to:

- Not make the user unlock the keyboard to respond to the message
- Provide a formatted display that can be paged for additional message information

To accomplish this, you should use a subfile containing messages from a program message queue (SFLMSGKEY, SFLMSGRCD, and SFLPGMQ keywords) rather than use the ERRMSG, ERRMSGID, SFLMSGID, and SFLMSG keywords.

To provide messages that tell the user what is correct, not simply that an entry is not valid, validity checking in your program is recommended over using the CHECK, RANGE, VALUES, and COMP keywords in DDS. When these keywords are used, a message tells the user that what is entered is not valid, but no explanation or indication of what is valid is given.

Message presentation on the system adheres as closely as possible to the following rules:

- Detected input errors result in an error message and the function is not performed.
- Error messages are shown on the display where the values that are not valid or options were entered. The error message does not appear on a separate display started by the data entered and the Enter action.
- When errors are detected on data entered on a display, the display is shown again with:
  - The first value(s) in error visible. In the case of a display that cannot be paged, the display is shown again with the erroneous values still in the fields where they were entered. For the display that can be paged where the user only entered values on one display page of values, the display is shown again as it appeared when the Enter key was pressed. For the display that can be paged where the user has paged and entered values on multiple pages, the display is shown again with the first page of values that contains an error.
  - The erroneous values are shown in reverse image (if the display station supports this).

    Note: If reverse image is used in a highlighted input field, the highlight attribute should be turned off (changed to normal intensity) to avoid a no display situation with 5250 display stations.
  - The cursor is positioned on the first field with a value in error.
  - The keyboard is not locked.
- When multiple errors are detected on data entered on a display:
  - Error messages for all errors on the display can be viewed, one at a time, on the message line using the roll/page key.
  - If multiple messages are waiting, the last three positions of the message line contain "b+b" (b indicates blank) in high intensity to indicate additional messages are present. The roll/page keys allow viewing the other messages. To page through the messages, the user moves the cursor to the message line and presses the appropriate roll/page key.
  - When multiple messages are presented, the messages are in the same order as the values in error appear on the display.

- When the user takes the appropriate corrective action (pressing a key, for example), the error message is removed. Only messages for errors that have not been corrected are present on the message line or waiting to be viewed.

- If a user requests help for an item other than the message (the cursor is not on the message line when the user presses the Help key), the message remains on the display when the user returns from the Help displays.

Designing Common User Access (CUA) Entry Level Models

CUA has defined the entry model to guide the programmer of an existing simple, transaction-oriented application in redesigning the user interface of that application to use some of the standard interface components defined by CUA. The key difference between entry and the other CUA models is that entry is action oriented and follows an action-object process sequence. Other CUA models have an object-action orientation.

The CUA entry model closely resembles the existing i5/OS interface. The user is typically asked to select an action or task from a menu or list and then indicate the object to which that action applies.

Entry Dialog Actions

Certain CUA entry dialog actions must be shown in the function key area when they are available on the display area. Table 54 summarizes CUA entry dialog actions indicating the function key assignments and when each key is used. If the function is not available, the key should not be used for another function.

Table 54. CUA Entry Dialog Actions

Function	Key	Usage Requirements
Enter	Enter	Active on all displays, but not shown in function key area
Help	F1, Help	Help must be active on all displays, but is not shown on the display. F1 must also be mapped to the Help function. F1 is not shown in the function key area. CUA screens should have F1 turned on for help; however, DDS also allows the Help key to be assigned to this function.
Exit	F3	Must be active and shown on all displays except: • Confirmation displays. • List display that appear as a result of the user pressing F4. • Exit displays that result from the user pressing F3=Exit where the decision is made as to save or not. • Pop-up windows
Prompt	F4	Must be supported whenever the display contains an entry field for which the valid entries are known but cannot be shown in the possible choice description area. Required on any display with a command line.
Refresh	F5	Required on action list displays (except selection lists) and strongly recommended on other list displays. Should be used on other displays where the refresh function is appropriate.
Backward	Page Up, F7	Must be operational on all displays with an area that can be paged. Not shown in function key area. Page up shows the previous page of data. CUA displays should have Page up turned on for paging; however, DDS also allows F7 to be assigned to this function.

Table 54. CUA Entry Dialog Actions (continued)

Function	Key	Usage Requirements
Forward	Page Down, F8	Must be operational on all displays with an area that can be paged. Not shown in function key area. Page down shows the next page of data. CUA displays should have the Page Down key turned on for paging; however, DDS also allows F8 key to be assigned to this function.
Retrieve	F9	Required on all displays with a command line.
Command	F9	Used to display a pop-up command line.
Cancel	F12	Must be active and shown on all displays except: • When the previous display was a prompt for a function that had already been processed. • On a display where only the Exit function is appropriate because the user must not be allowed to exit without first verifying (by way of an Exit display) whether data is to be saved.
Left	F19	Must be operational on all displays with a horizontal area that can be paged. There are no common columns of information between these successive views.
Right	F20	Must be operational on all displays with a horizontal area that can be paged. There are no common columns of information between these successive views.

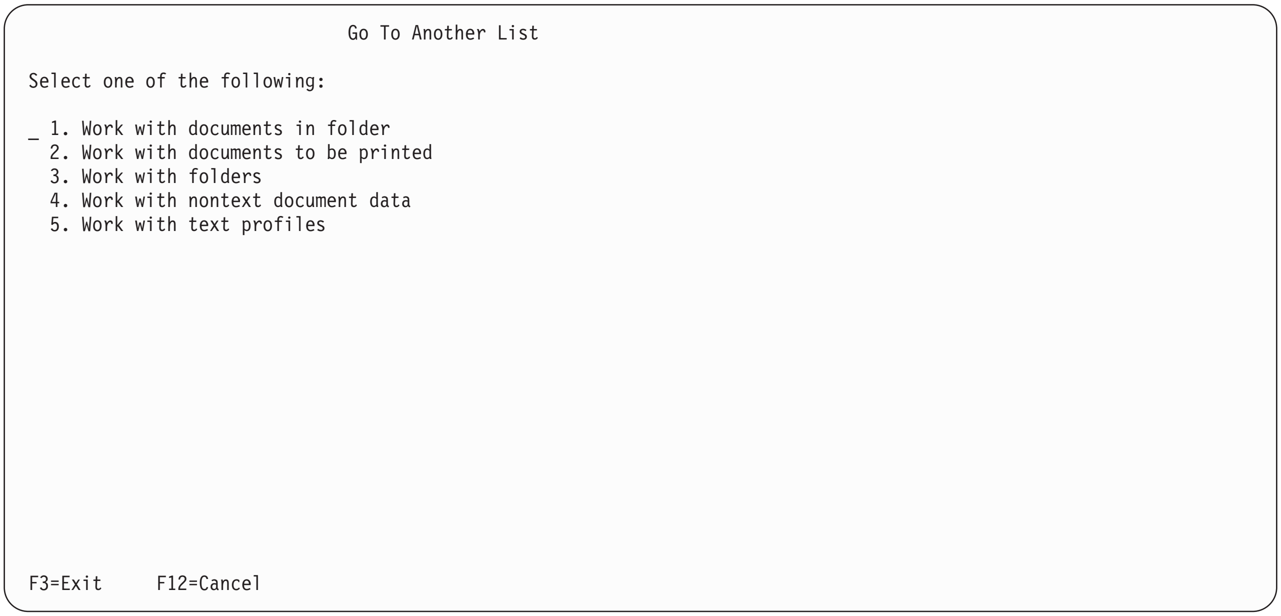
Function Key Area and Message Line Relationship

The entry model has the message line placed as a separator between other display areas and the function key area, and command area if used. A command line would be above the function key area and below the message line.

The guidelines for the function key area do not change, except for placement. A single line of function keys is on line 24. Two lines of function keys occupy lines 23 and 24. The message area is on the line immediately above the function keys.

Single-Choice Selection (Menu)

A CUA single-choice selection field displays a list of choices from which the user can select one choice. The display always has a title, an instruction, a list of choices, and a labeled field for typing the number of the choice selected. Figure 142 on page 458 shows an example of a CUA single-choice selection field on a display.



```
                          Go To Another List

Select one of the following:

_ 1. Work with documents in folder
  2. Work with documents to be printed
  3. Work with folders
  4. Work with nontext document data
  5. Work with text profiles















F3=Exit     F12=Cancel
```

Figure 142. Example of an Application Menu

Selection Choices and Choice Entry Field

The guidelines for selection choices and the choice entry field are as follows:

- Begin the choices one blank line after the instruction line.
- A choice entry field is on line 5 preceding the first choice. The length of a choice entry field is one position if the number of choices is less than 10, two positions if any choice numbers are 10 or greater (up to 99).
- The choice entry field starts in position 2.
- List the choices (one per line) starting in position 4 if the number of options is nine or less. The number is followed by a period and is not highlighted.
- If any choice number exceeds 9, the choice entry field requires 2 characters (in positions 2 and 3). Option numbers 1 through 9 start in position 6, and numbers 10 and higher start in position 5.

Guidelines for Single Selection Field Operation

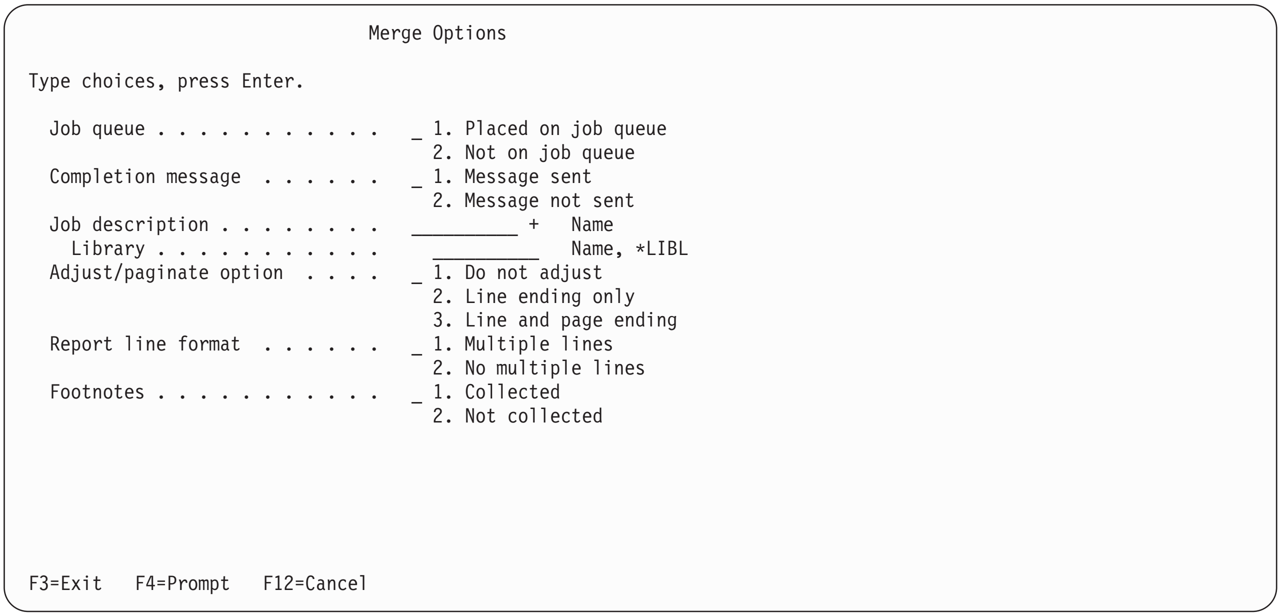
The single selection field allows two methods of interaction. The user can enter the number of the choice in the choice entry field and press the Enter key, or can move the cursor directly to the desired choice and press the Enter key.

Use the following rules:

- If the cursor is in the choice entry field, or 1 character position to the right (position 3 or 4), process the choice in the choice entry field.
- If the cursor is not in or immediately to the right of the choice entry field, and the choice entry field contains a blank, process the choice on the same line as the cursor.
- If the cursor is not in or immediately to the right of the choice entry field, the choice entry field is not blank (a value has been entered), and the value in the field is not the same as the choice being selected, present a message that asks the user to choose between the choice being pointed at and the choice indicated by the value in the choice entry field.

Entry Display

Entry displays let users type in entry fields, and are typically used to indicate the system parameters and options associated with an action request. Figure 143 shows an example of an entry display.



```
                                Merge Options

Type choices, press Enter.

  Job queue . . . . . . . . . . .   _ 1. Placed on job queue
                                      2. Not on job queue
  Completion message  . . . . . .   _ 1. Message sent
                                      2. Message not sent
  Job description . . . . . . . .   __________ +   Name
    Library . . . . . . . . . . .     __________   Name, *LIBL
  Adjust/paginate option  . . . .   _ 1. Do not adjust
                                      2. Line ending only
                                      3. Line and page ending
  Report line format  . . . . . .   _ 1. Multiple lines
                                      2. No multiple lines
  Footnotes . . . . . . . . . . .   _ 1. Collected
                                      2. Not collected





F3=Exit   F4=Prompt   F12=Cancel
```

Figure 143. Entry Display

Entry Fields

CUA entry fields take two forms. The first form is an entry field that requests a user-supplied value, like a name, descriptive text, or address. Frequently, these fields can accept a value from a list of values that is of variable length and contents. When such a list of valid choices exists, support F4 to allow the user to request the list and simply choose from it.

Using descriptive text (F4 for list) is preferred method for identifying a field supporting F4. As an alternative, CUA specifies a plus sign (+) to follow the entry field (with one intervening blank for an attribute byte) if the field supports F4.

```
File  . . . . . . . . .    __________ +   Name
```

The second type of entry field supports a selection from a fixed set of choices. CUA calls this a Selection field. If only one choice can be selected, number the choices. For example:

```
Type style . . . . .   _ 1. Prestige elite (12 pitch)
                         2. Courier (10 pitch)
                         3. Essay standard
```

The user selects by either moving the cursor or typing in the entry field (see “Guidelines for Single Selection Field Operation” on page 458).

If the list of values is more than can be shown in one to five lines, support F4.

CUA discourages prompts that are phrased to require selection of a Yes or No response. For example, for setting printing options, instead of using:

```
Duplex . . . . . . . .   _ 1. Yes
                           2. No
```

CUA recommends:

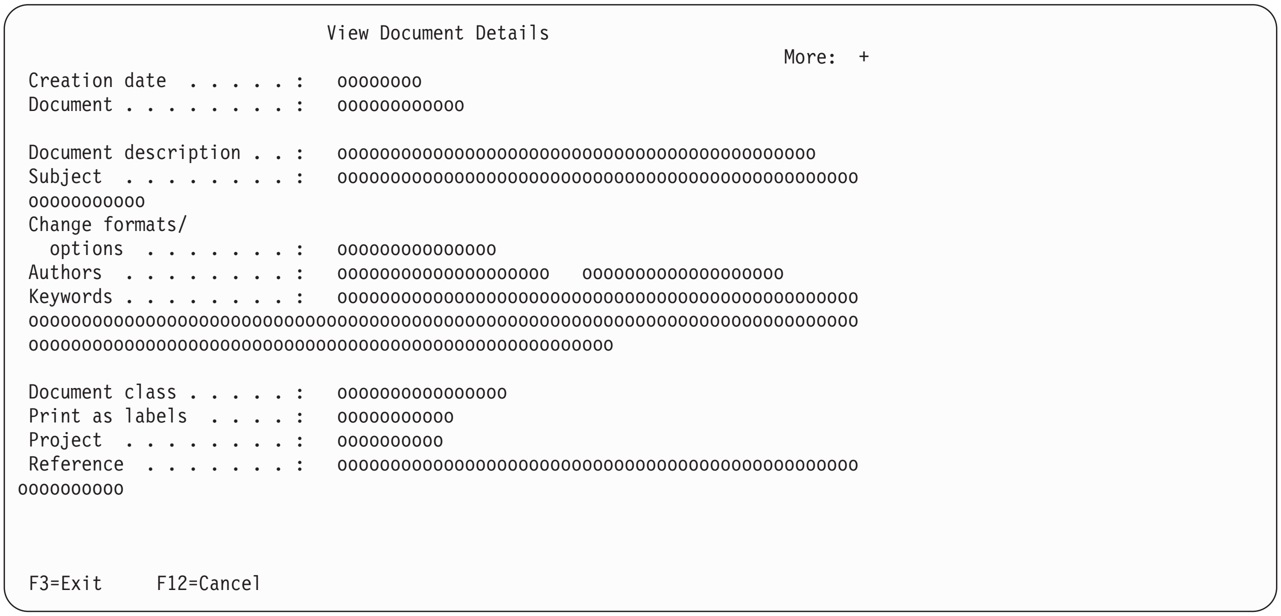
```
    One or both sides  . .   _ 1. Both sides
                               2. One side
```

Selection choices are aligned on the left and shown on separate lines. No comma or other punctuation is shown at the end of any line.

```
  _ 1. Prestige elite (12 pitch)
    2. Courier (10 pitch)
    3. Essay standard (proportional)
    4. Essay bold (proportional)
```

Information Display

A CUA version of the information display shown could be paged instead of being in two parts as the example shown in Figure 131 on page 430. Also, the explanatory information for Yes and No would not be required because Yes and No would not have been used for the entry function for which this information display is based. See “List Display” for a discussion of CUA paging information. Figure 144 shows an example of the sample information display in CUA format.



```
                             View Document Details
                                                                        More:  +
 Creation date  . . . . . :   oooooooo
 Document . . . . . . . . :   oooooooooooo

 Document description . . :   ooooooooooooooooooooooooooooooooooooooooooooo
 Subject  . . . . . . . . :   ooooooooooooooooooooooooooooooooooooooooooooooooo
 ooooooooooo
 Change formats/
   options  . . . . . . . :   ooooooooooooooo
 Authors  . . . . . . . . :   oooooooooooooooooooo   ooooooooooooooooooo
 Keywords . . . . . . . . :   ooooooooooooooooooooooooooooooooooooooooooooooooo
 oooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooo
 ooooooooooooooooooooooooooooooooooooooooooooooooooooooo

 Document class . . . . . :   oooooooooooooooo
 Print as labels  . . . . :   ooooooooooo
 Project  . . . . . . . . :   oooooooooo
 Reference  . . . . . . . :   ooooooooooooooooooooooooooooooooooooooooooooooooo
oooooooooo



 F3=Exit     F12=Cancel
```

Figure 144. Example of an Information Display

List Display

Like the information display, a CUA version of the sample list display would differ primarily in the display of paging information. Figure 145 on page 461 shows an example of a list display.

CUA uses the word `More` followed by a colon and then paging symbols to indicate additional information exists outside the visible area, and the direction to page to see that information.

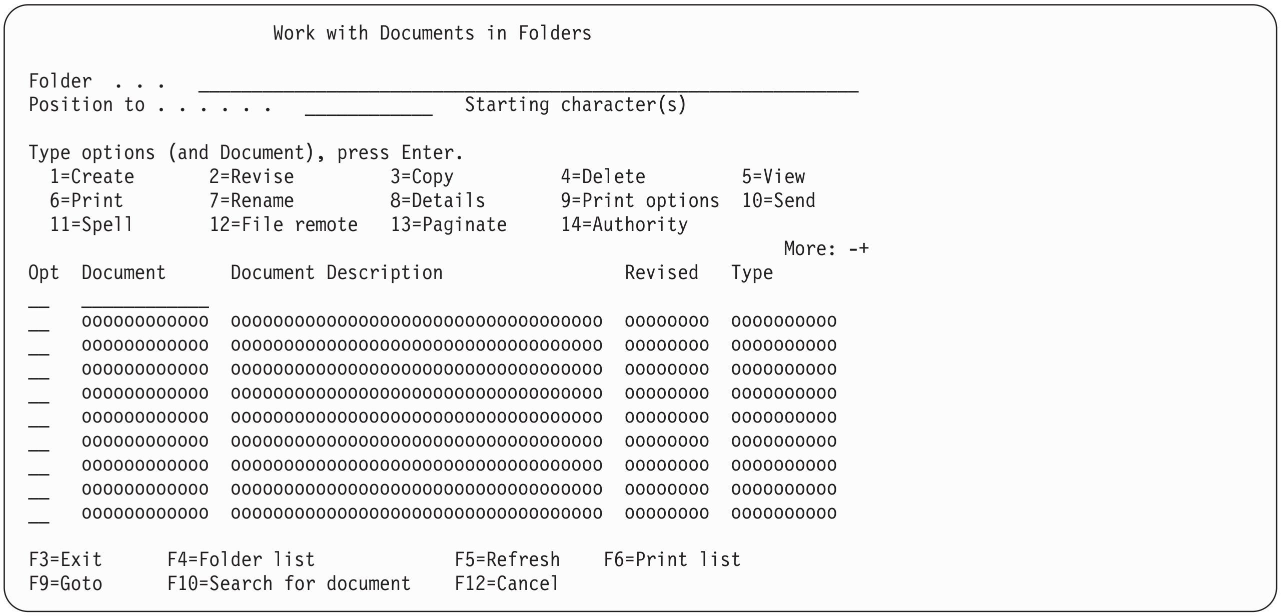
If the user can page in all four directions, backward, forward, left, and right, reserve space for all four paging symbols. If the user can page in only two directions, reserve space for two. Symbols are not shown if the user cannot page in the direction they represent; blanks are shown instead.

The four paging symbols are:

<	indicates there is information to the left
–	indicates there is information backward

+ indicates there is information forward

> indicates there is information to the right



```
                       Work with Documents in Folders

Folder  . . .   _______________________________________________________________
Position to . . . . . .   ____________   Starting character(s)

Type options (and Document), press Enter.
  1=Create       2=Revise         3=Copy          4=Delete         5=View
  6=Print        7=Rename         8=Details       9=Print options  10=Send
  11=Spell       12=File remote   13=Paginate     14=Authority
                                                                       More: -+
Opt  Document      Document Description                Revised   Type
__   ____________
__   oooooooooooo  ooooooooooooooooooooooooooooooooooo  oooooooo  oooooooooo
__   oooooooooooo  ooooooooooooooooooooooooooooooooooo  oooooooo  oooooooooo
__   oooooooooooo  ooooooooooooooooooooooooooooooooooo  oooooooo  oooooooooo
__   oooooooooooo  ooooooooooooooooooooooooooooooooooo  oooooooo  oooooooooo
__   oooooooooooo  ooooooooooooooooooooooooooooooooooo  oooooooo  oooooooooo
__   oooooooooooo  ooooooooooooooooooooooooooooooooooo  oooooooo  oooooooooo
__   oooooooooooo  ooooooooooooooooooooooooooooooooooo  oooooooo  oooooooooo
__   oooooooooooo  ooooooooooooooooooooooooooooooooooo  oooooooo  oooooooooo
__   oooooooooooo  ooooooooooooooooooooooooooooooooooo  oooooooo  oooooooooo

F3=Exit      F4=Folder list            F5=Refresh    F6=Print list
F9=Goto      F10=Search for document   F12=Cancel
```

Figure 145. Example of a List Display

According to CUA, the following paging indicators can be used instead of `More: - +` for alternative text paging indicators:

Top Above the area that is paged when users view the beginning of the information

More Above the area that is paged when users can page backward

Bottom Below the area that is paged when users view the end of the information

More Below the area that is paged when users can page forward

Note: This technique takes extra lines above and below the area that will be paged.

Using CUA defined paging information is not possible when using subfiles. The application program itself must handle the paging. See “List Fields” on page 435 for more information.

The paging method used should be cursor independent when the information is paged in fixed amounts.

Help Information

CUA help information resembles i5/OS help information by having a number of defined functions available from every help display via function keys. These function keys and their functions are:

F1=Help for help
F2=Extended help
F3=Exit
F5=Tutorial
F9=Keys help
F11=Help index
F14=Tutorial

These functions cannot be provided by DDS using function keys from help displays. For information on DDS capabilities for help information, see “Defining Help Information for All Displays” on page 450.

Part 6. Appendixes

Appendix A. UIM Panel Group Definition Language

This appendix describes the UIM panel group definition language. Following some introductory material on the nature of the language syntax and the structure of the panel group definitions, the actual statements of the language, the **tags**, are defined. The tag definitions in this chapter are organized alphabetically.

A sample syntax diagram is shown below:

```
►►──:TAGNAME──VATTR──=──value──────────────────────────────────────.─────────────────────────►◄
                              └─KATTR──=──┬─DEFAULT─┬──VALUE1──VALUE2─┘  └─tag-content─┘
                                          └─────────┘
```

The TAGNAME specifies the name of the tag. This sample tag has two attributes: VATTR, for value attribute; and KATTR, for keyword attribute. The second attribute in this example is shown on the alternate branch to indicate that it is an optional attribute. This attribute can have one of three values: DEFAULT, VALUE1, and VALUE2.

DEFAULT is the value used when you do not specify a value for the KATTR attribute. If there is a default value, it appears on the group choice line above the attribute name, as DEFAULT appears on the line above the KATTR attribute name. Instead of using the DEFAULT value, you may choose either VALUE1 or VALUE2. These values are entered as shown in the syntax diagram.

The *value* is a value you enter, as specified for that attribute. An attribute *value* must be enclosed with apostrophes (') when it contains characters other than A through Z or 0 through 9, and it must be contained on one source line. To enter an apostrophe in a value surrounded by apostrophes, type the apostrophe twice.

The *tag-content* is text associated with that tag that may be displayed or printed for the user. It is designed to allow national language translation.

The period at the end of the tag and before the tag content is known as the **markup/content separator**. This separator is required for all tag markup.

The source for a panel group can be entered in either lowercase, uppercase, or mixed case characters. The UIM converts the tag names, attribute names, and attribute values to uppercase as appropriate during the compile process. For example, the following uses of the example tag, TAGNAME, are equivalent specifications:

```
:TAGNAME VATTR=VALUE.
:tagname vattr=value.
:Tagname Vattr=Value.
```

Tag attributes do not need to be specified in the order specified in the syntax diagram. For example, the following uses of the example tag, TAGNAME, are equivalent specifications:

```
:tagname vattr=value kattr=value1.
:tagname kattr=value1 vattr=value.
```

Sometimes it is necessary to continue the attribute of a tag to another source line. However, not all attributes can be continued. The attributes in the following table are the only ones that can be continued.

Table 55. Tag Attributes That Can Be Continued

Attribute	Tag
ACTION	KEYI
ACTION	MENUI
ACTION	PDFLDC
COLHEAD	LISTCOL
COLS	LISTVIEW
CONDS	TT
EMPHASIS	LISTDEF
ENTER	LISTACT
ENTER	PANEL
EXPR	COND
EXTENTER	LISTACT
EXTPROMPT	LISTACT
LINKWHEN	LINK
NOGET	VARRCD
NOPUT	VARRCD
PERFORM	LINK
PROMPT	LISTACT
PROTECT	LISTDEF
RANGE	CHECK
REL	CHECK
ROOTS	ISCH
TOPICS	ISCHSUBT
UNLESS*n*	LINK
VALUES	CHECK
VALUES	TTROW
VARS	LISTDEF
VARS	VARRCD

To continue an attribute, repeat the attribute on the next source line, as in the following example:

```
:TAGNAME
        VATTR='value1 value2 value3'
        VATTR='value4 value5 value6'
        VATTR='value7 value8 value9'.
tag-content
```

A maximum of fifty attribute names can be specified for any tag, including each repeated attribute name.

Tag Content Formatted as Paragraphs

The period (.), question mark (?), and exclamation point (!) are sentence-ending characters. Depending on the national language requirements, one or more blanks are placed after each of the sentence-ending characters that end a UIM source line when the panel group or menu object is created. The number of blanks can be from one to six, and is specified in the message text of the CPI6AB9 message in the message file QCPFMSG.

The following example shows how the CPI6AB9 message works.

```
:p.This is the first sentence.
This is the second!   This is the third.
```

In this example, `:p.` is the tag name. The `:p.` tag does not require attributes. There are three spaces between the second sentence and the third sentence because of national language requirements. If the CPI6AB9 message is set to `xx`, this paragraph appears this way to the UIM after it is compiled:

```
This is the first sentence.  This is the
second!   This is the third.
```

There are two spaces between the first and second sentences. There are three spaces between the second and third sentences.

Because the CPI6AB9 message affects the amount of space that follows the end of a source line, if the CPI6AB9 message is changed to xxx, there will be three spaces between each sentence in the previous example.

Note: The CPI6AB9 message should already be translated to the proper number of x's for your country or region.

Panel Areas

The five types of panel areas are menu, information, data, list, and text areas. A mixed panel is constructed by using two or more of these areas within the same panel.

The restrictions associated with mixed panels are listed in Table 56.

Table 56. Restrictions Associated With Mixed Panels

	Menu	**Info**	**Data**	**Non-Action List**	**Action List**	**Text**
Menu	No	Yes	Yes	Yes	No	No
Info	Yes	Yes	Yes	Yes	Yes	No
Data	Yes	Yes	Yes	Yes	Yes	No
Non-Action List	Yes	Yes	Yes	Yes	Yes	No
Action List	No	Yes	Yes	Yes	No	No
Text	No	No	No	No	No	No

All of these area types can be scrolled. Scrolling, as well as presentation of the appropriate scroll location information, is handled by the UIM for menu, information, data, and list areas. Scrolling for text areas is handled by the text area user exit program.

Panels

A panel is made up of one or more panel areas, each of which has a specific layout and function. Specific panels are usually categorized according to the nature of their application areas, such as information panels, menu panels, list panels, and text panels.

Panel Group Objects

A panel group is a logical grouping of panels that can correspond to a software product, a command, or a related collection of commands or other services. It can contain panels of any type. Panel groups provide a context for names of panels, variables, list definitions, and help modules.

Help on Panels

Help for a panel is constructed from help modules referred to in the panel definition. Extended help for a panel is a concatenation of the help for the panel, the menu bar, items in all areas in the order of their definitions, and function keys in the order of their definitions.

Panel Group Organization

The following list shows the required order of tags in the UIM source for panel groups and menus.

Panel Group Outline: Panel Group (PNLGRP)

- Prolog section

  Copyright statement (COPYR)
  Import statements (IMPORT)
  Class definitions (CLASS)
  Variable definitions (VAR)
  Variable Record Definitions (VARRCD)
  List definitions (LISTDEF)
  Condition definitions (COND)
  Truth table definitions (TT)
  Menu bar definitions (MBAR)
  Key list definitions (KEYL)
- Body section

  Display panel definitions (PANEL) [3]
  - Area definitions (MENU, INFO, DATA, LIST, TEXT)
  - Command and option line definitions (CMDLINE, OPTLINE)

  Print head panel definitions (PRTHEAD) [3]
  - Area definitions (INFO, DATA)
  - Print trailer message (PRTTRAIL)

  Print panel definitions (PRTPNL) [3]
  - Area definitions (INFO, DATA, LIST)

  Help module definitions (HELP) [3]

A UIM menu, created with the Create Menu (CRTMNU) command, is a special form of panel group. These menus are limited to the following panel group tags:

UIM Menu Outline: Panel Group (PNLGRP)

- Prolog section

  Copyright statement (COPYR)
  Import statements (IMPORT)
  Variable definitions (VAR)
  Condition definitions (COND)
  Truth table definitions (TT)
  Menu Bar (MBAR)
  Key list definitions (KEYL)

[3] PANEL, PRTHEAD, PRTPNL, and HELP tags can be used in any order.

[4] The PANEL and HELP tags can be used in any order.

- Body section
  - Panel definitions (PANEL) [4]
    - Area definitions (MENU, INFO, DATA)
    - Command and option line definitions (CMDLINE, OPTLINE)
  - Help module definitions (HELP) [4]

Name Syntax

There are several elements, identified with tags, within a panel group that can be named so that they can be referred to from other elements within that panel group, such as classes, dialog variables, lists, and conditions. The application program can also refer to named elements by passing the name to the UIM as a parameter on an application programming interface (API) call. These elements have a `NAME` attribute on the tags that define them. These names may be up to 10 characters long and can contain only the characters A through Z and 0 through 9. The first character of a name must be an A through Z character.

Names of help modules defined by the HELP tag can be up to 32 characters long and are defined in the same name space as other elements. As a result, a help module and another element cannot have the same name in a panel group. They can contain the characters A through Z, 0 through 9, slash (/), and underscore (_). If the name contains a slash or an underscore, it must be enclosed between apostrophes. The first character of a help name must be an A through Z character, a slash, or an underscore.

Because all names are stored in the panel group object in uppercase format, lowercase alphabetic characters (a through z) are converted to uppercase alphabetic characters (A through Z). When the name of an element is passed to the UIM through an application program interface, the name must be passed in uppercase characters.

Because these names stay within the panel group and reside in the same name space, a dialog variable and a UIM list cannot have the same name.

There is an occasional need to refer to object names in the panel group source. These names must obey the rules for i5/OS object names. They must be enclosed between apostrophes when they contain characters other than A through Z and 0 through 9.

Symbols

A symbol is a name that can be replaced with something else while the panel group is compiling. All symbols use an ampersand (&), followed by the symbol name, followed by a period. Symbols must appear in the text following the period of a tag.

Symbols can be entered in either uppercase or lowercase.

The following symbols are defined by the UIM:

&amp. Creates an ampersand (&) without triggering symbol processing. This symbol is used when an ampersand is needed in the text of a tag. If this symbol is not used, the UIM replaces the symbol name with its actual value.

```
►►──&amp.─────────────────────────────────────────────────────────────►◄
```

&colon.

Creates a colon (:) without triggering tag processing. This symbol is used when a colon is needed in the text of a tag.

```
►►──&colon.───────────────────────────────────────────────────────────►◄
```

&cont. Forces the concatenation of the next source line with the current source line. Concatenation occurs only when the symbol is the last character in a source line. No visible symbol is created for the &cont. symbol.

When the &cont. symbol is used to continue the two parts of a double-byte character set (DBCS) word, both the ending shift-in character of the first line and the starting shift-out character of the second line are discarded.

The &cont. symbol is useful with a figure, a line, or an example. The symbol can be used when a CIT, HP*n*, or LINK tag would cause the resulting source line to be longer than the length of the source record.

Here is an example of how to use the &cont. symbol.

```
:LINES.
Text to be continued &cont.
:LINK perform='dsphelp hyper'
linkwhen='chkusrcls("*PGMR")'.&cont.
reference phrase&cont.
:ELINK.&cont.
more text on line.
:ELINES.
```

```
►►──&cont.──────────────────────────────────────────────────────────────►◄
```

&msg(msgid,msgf,lib,NOSUB).
The compiler extracts the first-level message text for the message MSGID from the message file MSGF in library LIB and substitutes the text in place of this symbol. The values *LIBL and *CURLIB, without single quotation marks, can be specified for the library. The message text is retrieved in the CCSID of the panel group source file without any substitution variables.

The message file defaults to the message file specified on the SUBMSGF attribute of the PNLGRP tag. If SUBMSGF is not specified, the message file must be specified in the symbol.

The NOSUB option causes the compiler to not substitute blanks for replacement variables found in the extracted message.

```
►►──&msg(msgid──┬───────────────────────────────────┬───────────────────►◄
                └─,msgf──┬──────────────────────┬─).┘
                         └─,lib──┬───────────┬──┘
                                 └─,NOSUB────┘
```

&period.
Creates a period (.) in the text. This symbol is used when a period is needed in column one of the text without triggering control word processing.

```
►►──&period.────────────────────────────────────────────────────────────►◄
```

&slr. Creates a right slash (/) in the text. This symbol is used if a right slash is needed in column one of the tag source. If this symbol is not used, right slashes in column one of the source file can cause problems if the source file is part of the input stream for a batch job.

```
►►──&slr.───────────────────────────────────────────────────────────────►◄
```

Comments

In addition to markup done with tags, comments can also be inserted into the source of the panel group. Typically, comments are used to describe the content of the file, or to give information as to the structure, format, date, author, and so on.

Comments begin with a period followed by an asterisk (.*). They must be used in columns one and two of the source record. They are ignored by the compiler, but are listed in the source listing. They are not preserved in the panel group object. An example using comments follows:

```
.*****************************************
.* panel group      xxx.xxx
.* author           X. X. Xxxxxxx
.* change history   9/18/92 (xxx) Created
.*****************************************
:pnlgrp ...
   ⋮
:epnlgrp.
```

Imbeds

Source from other files can be brought into a panel group at the time the panel group is compiled. This allows you to create a panel group or menu from multiple source file members. Imbeds are done with the `.IM` control word. An example using imbeds follows:

```
.im member1
```

The member (member1) in the file named in the include file (INCFILE) parameter on the Create Panel Group (CRTPNLGRP) or Create Menu (CRTMNU) command is opened and processed as if it appeared in the original source file. Imbeds can be nested up to 10 deep; attempting to nest deeper than 10 results in an error and the panel group or menu object is not created.

DBCS Graphic Literals

DBCS graphic literals are supported by the UIM to support processing of strings of literal characters of DBCS graphic data. The syntax of the literals follows:

```
                           ┌────────┐
                  (1)      ↓  (2)   │           (3)
►►─"─shift-out───────────D──────────┴─shift-in──────"─g──────────────────────────────────────────►◄
```

Notes:

1 *Shift-out* is X'0E'

2 *D* is one DBCS character

3 *Shift-in* is X'0F'

The value must be enclosed in apostrophes because it contains special characters.

All strings of literal characters coded on UIM tags that are compared against a DBCS graphic field must be in the form of DBCS graphic literals. The shift-out and shift-in characters are removed from the graphic strings before the comparison is performed.

Caution should be used when literals are compared to input-capable graphic fields. You must realize that on a nongraphic DBCS device, there are two fewer bytes available for input. As a result, you should always use literals that are two bytes less than the maximum length of the entry field of the DBCS graphic variable.

When DBCS graphic characters are used after the period of a tag, (such as on the translation list item (TI) tag) a graphic literal should not be used. What should be used is everything between and not including the quotation marks in the diagram above.

Hexadecimal Literals

Hexadecimal data can be entered in attributes wherever a character string literal is allowed. The syntax of hexadecimal literals follows:

```
            ┌──────────┐
                (1)
►►──"──────▼─xx────────┴──"──x──────────────────────────────────────────────────►◄
```

Notes:

1 *xx* is a pair of hexadecimal digits (0 through 9, a through f, or A through F)

The following example shows how to code a hexadecimal value for a translation item.

```
:TI value='"01"x'.*YES
:TI value='"02"x'.*NO
```

The value must be enclosed in apostrophes because it contains special characters.

APPFMT (Application Formatted Area)

```
►►──:APPFMT──VAR──=──dialog-variable-name──WIDTH──=──area-width──DEPTH──=──area-depth──────────►

►──┬────────────────────────────────────────┬──────────────────────────────────────────────►◄
   └─USREXIT──=──'CALL program-reference' ──.─┘
```

The application formatted area (APPFMT) tag is an optional tag which defines an area in the upper right corner of a menu area to be formatted by the application. Only one APPFMT tag can be specified in each menu area. It must immediately follow the menu area (MENU) and top instruction (TOPINST) tags.

The application formatted area is part of the menu area, appearing at the right edge of the menu area and ending one character position before the right edge of the panel. The width is determined by the `WIDTH` attribute of this tag. The application formatted area extends from the blank line following the top instruction lines of the menu area to the number of lines defined in the `DEPTH` attribute of this tag. A two-byte separator is maintained between the menu item descriptions and the application formatted area.

If the depth of the application formatted area is less than the depth of the menu area, the area below the application formatted area and to the right of the menu area is blank. The depth of the application formatted area may not be greater than the depth of the menu area.

If the menu area is scrollable, the scroll indicators appear to the left of the separator between the menu area and the application formatted area. You cannot scroll the application formatted area, even if you can scroll the menu area.

The application formatted data is described in "Application Formatted Data" on page 473.

Extended help for the panel is displayed if the Help key is pressed while the cursor is in the application formatted area.

Required Attributes

VAR=*dialog-variable-name*
The name of the dialog variable containing the application formatted area. Define the dialog variable using a class definition (CLASS) tag with a BASETYPE of CHAR *x*, where *x* equals the product of the area width times the area depth.

You can use the exit program for application formatted data, specified on the USREXIT attribute of this tag, to update this dialog variable each time the panel is displayed. The UIM uses the value of the dialog variable as the application formatted data, described in "Application Formatted Data."

WIDTH=*area-width*
The width, in characters, for the application formatted area associated with the menu area. The width specified must be an integer in the range of 1 to 40, or the panel width minus 17, whichever is less.

DEPTH=*area-depth*
The depth, in lines, of the application formatted area. This depth must be no greater than the number of lines available for the body of the menu. The body of the menu is the area between and excluding the top instruction line and the bottom instruction line or bottom separator.

Optional Attribute

USREXIT='*CALL program-reference***'**
The exit program for application formatted data, called to update the value of the dialog variable containing the application formatted data each time the panel is displayed.

For a description of the CALL dialog command, see Appendix B, "UIM Dialog Commands," on page 641.

For a description of the interface between the UIM and the exit program for application formatted data, see the **APIs** topic in the iSeries Information Center.

Application Formatted Data

The dialog variable containing the application formatted data is specified on the VAR attribute of this tag. The contents of this variable is viewed by the UIM as a *d* by *w* array of characters, where *d* is the depth and *w* is the width of the application formatted area. Each row of the array is displayed as one line of the application formatted area.

Any character within the data can be a selection character for a highlighting class. Only output and text classes are allowed in an application formatted area; no input fields are allowed. The class selection applies to the following characters up to another class selection character or the end of the row, whichever occurs first. The UIM sets the highlighting class to normal text at the beginning of each row of the area.

The selection characters for highlighting are in the range of X'01' through X'06'. The following selection characters for highlighting are recognized by the UIM:

X'01'	Normal variable output
X'02'	De-emphasized variable output
X'03'	Emphasized variable output
X'04'	Normal text
X'05'	De-emphasized text
X'06'	Emphasized text

The UIM replaces each class selection character with the appropriate display attributes for the class. Other character values in the ranges X'01' through X'06', and X'10' through X'3F', as well as X'FF', are converted to X'1F', appearing as a reverse image box on the screen. Characters X'00' (null), X'0E' (shift-out for double-byte), X'0F' (shift-in for double-byte), and X'40' through X'FE' (normal, displayable characters), are passed unchanged to the screen.

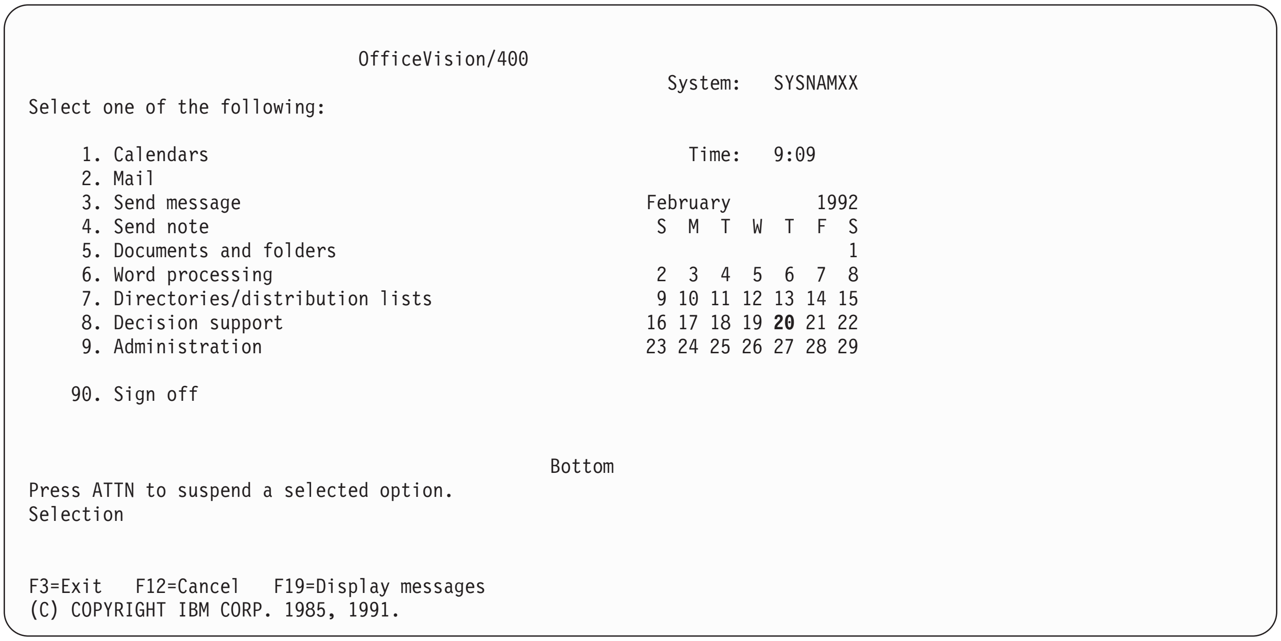
The UIM inserts a normal text attribute before each line of the application formatted area and a field-ending attribute after each line of the application formatted area. This effectively isolates the application formatted area from the rest of the panel with respect to highlighting attributes.

No character set and code page conversion is done on the application formatted data. If conversion is necessary, it must be done by the application.

When the UIM calls the exit program to format the application area of a menu, it passes a value which identifies the BIDI attribute on the panel group (PNLGRP) tag and the code page number of the display device.

The NBRSHAPE and SYMSWAP attributes of the CLASS tag used for the application formatted area are determined from the class of the dialog variable containing the application formatted data.

Example: Application Formatted Area



```
                               OfficeVision/400
                                                            System:   SYSNAMXX
Select one of the following:

     1. Calendars                                             Time:   9:09
     2. Mail
     3. Send message                                      February        1992
     4. Send note                                          S  M  T  W  T  F  S
     5. Documents and folders                                                1
     6. Word processing                                    2  3  4  5  6  7  8
     7. Directories/distribution lists                     9 10 11 12 13 14 15
     8. Decision support                                  16 17 18 19 20 21 22
     9. Administration                                    23 24 25 26 27 28 29

    90. Sign off


                                                 Bottom
Press ATTN to suspend a selected option.
Selection


F3=Exit   F12=Cancel   F19=Display messages
(C) COPYRIGHT IBM CORP. 1985, 1991.
```

BOTINST (Bottom Instruction)

```
►►──:BOTINST──┬─────────────────────────────────────────────────┬──────────────────────────►◄
              └─INST──=──dialog-variable-name──.─┬────────────────────┬─┘
                                                  └─instruction-text─┘
```

The bottom instruction (BOTINST) tag specifies the bottom instruction lines for an area of the panel. This tag appears immediately before the ending tag for the area. You can use multiple bottom instruction tags to present multiple instruction lines if the tag contains instruction text after the period. If multiple tags are coded, no blank lines appear between the text on different tags. Only one bottom instruction tag is allowed per area if the INST attribute is used on this tag.

If you do not specify a BOTINST tag for an area, no instruction line is allocated for the area.

For menus and information areas, a blank line is always left between the body of the area and the instruction lines. If you can scroll the area, a blank line is reserved for the scroll information between the body of the area and the bottom instruction. For information about how instruction lines are formatted with respect to the body of data and list areas, see the BODYSEP attribute in "DATA (Data Presentation Area)" on page 496 and "LIST (List Area)" on page 552.

Optional Attribute

INST=*dialog-variable-name*

The name of a dialog variable that contains the bottom instruction text to be displayed. The dialog variable must be defined with a width less than or equal to the width specified on the panel tag minus 2. If the INST attribute is used, no *instruction-text* can be specified for this tag.

Dialog variables must be defined with a BASETYPE of CHAR, IGC, or BIN on the class definition (CLASS) tag.

The error state of the dialog variable is not used for determining the highlighting of the text.

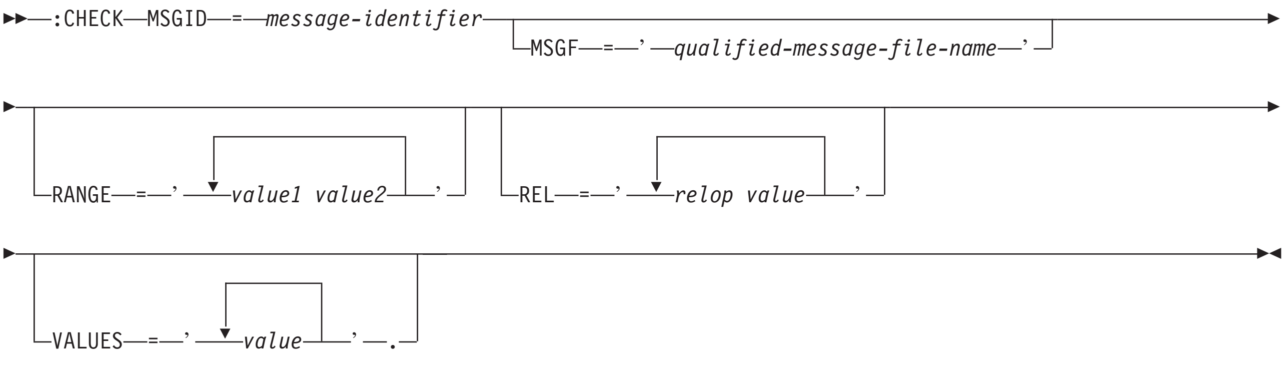
Special formatting for IGC. (The abbreviation IGC is used in commands and keywords to represent double-byte character set functions.) When a dialog variable with a BASETYPE of IGC is specified on the CLASS tag, the UIM does special formatting. If the dialog variable value begins with a shift-out character (X'0E'), the UIM shifts the value one byte to the left to preserve vertical alignment with other lines.

Optional Text

instruction-text

The text appearing as bottom instructions for the area. The text is an implied paragraph. When the display is formatted, any text that does not fit onto one display line is formatted on multiple lines as necessary and indented two display positions. The text can be a maximum of 255 characters, and can contain only the reverse text (RT) tag. The *instruction text* is required unless the INST attribute is specified on this tag.

CHECK (Validity Checking)



The validity checking (CHECK) tag is used within a class definition to specify validity checks associated with variables of that class.

Validity checking occurs on input of the variable value and after translation list processing. If a translation list is specified for this class and can be applied to an input value, no validity checking is performed on the translated input value. If the translation list is not applicable and indicates no error, the validity checks defined by this tag are applied to the input value. For more information on using the CHECK tag with translation lists, see "TI (Translation List Item)" on page 628 and "TL (Translation List)" on page 629.

At least one RANGE, REL, or VALUE attribute of this tag must be coded. If more than one validity check is coded on the tag, the value must pass only one of the check attributes to be accepted for this check. If more than one CHECK tag is used, a value must pass all the check tags to be accepted. Operations from multiple attributes are ORed together.

Required Attribute

MSGID=*message-identifier*

The message identifier of the error message displayed to the user if a validity check fails. The message should have no substitution variables defined, because the UIM does not supply replacement text when the message is sent.

Optional Attributes

MSGF='*qualified-message-file-name***'.**

The message file that contains the message specified on the MSGID attribute of this tag. If the DFTMSGF attribute is not specified on the panel group (PNLGRP) tag, this attribute must be specified.

RANGE='*value1 value2***'**

A range check is valid for both character and numeric types. The check passes if the value entered by the user is greater than or equal to *value1* and less than or equal to *value2*. This attribute can be specified more than once.

Each value must be numeric or character to match the BASETYPE of the variable as specified on the class definition (CLASS) tag, and cannot be a dialog variable. All character values must be enclosed in quotation marks. If the BASETYPE of this class is TIME, a time zone value must not be specified.

REL='*relop value***'**

A relational check is valid for both numeric and character types. This attribute can be specified more than once.

Each value must be numeric or character to match the BASETYPE of the variable as specified on the CLASS tag, and cannot be a dialog variable. All character values must be enclosed in quotation marks. If the BASETYPE of this class is TIME, a time zone value must not be specified.

Relational operators are listed below. All operators are valid for both character and numeric types. The check is affected by the CASE and BLANKS attributes of the CLASS tag.

= **Equal**. The check passes if the value entered by the user is equal to this literal value.

¬= **Not equal**. The check passes if the value entered by the user is not equal to this literal value.

> **Greater than**. The check passes if the value entered by the user is greater than this literal value.

< **Less than**. The check passes if the value entered by the user is less than this literal value.

>= **Greater than or equal**. The check passes if the value entered by the user is greater than or equal to this literal value.

<= **Less than or equal**. The check passes if the value entered by the user is less than or equal to this literal value.

The relational operator may be specified as =, ¬=, >, <, >=, or <=, if the code page of the source maps the not character ([) to X'5F'. Otherwise, it is suggested that the special values *EQ, *NE, *GT, *LT, *GE, or *LE be used, respectively, for the relational operator.

VALUES='*value1 value2 ... value-n***'**

A values check is valid for both character and numeric types. The check passes if the value entered by the user is equal to one of the values in the list. Checking on character values is affected by the CASE and BLANKS attributes of the CLASS tag. The VALUES attribute can be specified more than once, and up to 50 values can be specified for this tag.

Each value must be numeric or character to match the BASETYPE of the variable as specified on the CLASS tag, and cannot be a dialog variable. The values in the list are separated by blanks. All character values must be enclosed in quotation marks. If the BASETYPE of this class is TIME, a time zone value must not be specified.

Example: Validity Checking

UIM Source

```
:CHECK RANGE='1 10' REL='= 99' MSGID=USR0234.
:CHECK REL='< 5' REL='> 10' MSGID=USR1234.
:CHECK VALUES='"*YES" "*NO"'
MSGID=MMM0001 MSGF=USRMSGF.
```

The range check allows values 1 through 10, or 99, for any variables of the class to which it applies. The relational check allows values less than 5 or greater than 10. The values check accepts two special values.

CIT (Title Citation)

```
►►──:CIT──.─┬────────────┬─:ECIT.──────────────────────────────────────────────►◄
            └─title-text─┘
```

The title citation (CIT) tag identifies the title of a publication and requires a matching end tag. This tag is only allowed in information areas and help areas. Title citations are underscored for online and printed text, and can occur anywhere in the text.

The CIT and ECIT tags must be specified on word boundaries. If the two characters immediately following the ECIT tag are a punctuation mark and a blank, the UIM extends the emphasis attribute to include the punctuation mark. This allows the punctuation mark and the title to be displayed using the same emphasis.

Optional Text

title-text
Although the title to be cited is not required, the tag has no meaning when no title is specified.

Example: Title Citations

UIM Source

```
:P.For more information about the UIM,
see the :cit.Application Display Programming:ecit. book.
```

Results

```
For more information about the UIM,
see the Application Display Programming book.
```

CLASS (Class Definition)

```
►►─:CLASS──NAME──=──class-name──BASETYPE──=──┬─'──CHAR──  ──n──'─────────────────────┬──────────►
                                             ├─'──IGC──  ──n──┬────────────┬──'──────┤
                                             │                ├─OPEN───────┤         │
                                             │                └─EITHER─────┘         │
                                             ├─'──GRAPHIC──  ──w c──┬────────┬──'────┤
                                             │                      ├─CHAR───┤       │
                                             │                      └─POS────┘       │
                                             ├─'──NAME──  ──n──┬─────────────┬──'────┤
                                             │                 ├─SIMPLE──────┤       │
                                             │                 └─GENERIC─────┘       │
                                             ├─'──OBJNAME──  ──n──┬────────────┬──'──┤
                                             │                    ├─SIMPLE─────┤     │
                                             │                    ├─GENERIC────┤     │
                                             │                    └─POSTO──────┘     │
                                             ├─'──BIN──  ──b──'──────────────────────┤
                                             ├─'──PACKED──  ──t f──'─────────────────┤
                                             ├─'──ZONED──  ──t f──'──────────────────┤
                                             ├─'──PTR──'─────────────────────────────┤
                                             ├─'──DATE──'────────────────────────────┤
                                             ├─'──TIME──┬────────┬──'────────────────┤
                                             │          └─ZONE───┘                   │
                                             └─'──ACTION──'──────────────────────────┘

►──┬──────────────────────────────┬──┬──────────────────────────┬──┬──────────────────────────┬──────►
   └─WIDTH──=──display-width──────┘  └─CHRID──=──┬─NONE───┬─────┘  └─SHIFT──=──┬─NONE───┬─────┘
                                                 └─PNLGRP─┘                    └─UPPER──┘

►──┬──────────────────────────┬──┬────────────────────────────┬──┬─────────────────────────────┬────►
   └─CASE──=──┬─MIXED──┬──────┘  └─BLANKS──=──┬─RESPECT──┬────┘  └─SUBST──=──┬─DISPLAY──┬──────┘
              └─UPPER──┘                      └─IGNORE───┘                   └─QUOTED───┘

►──┬──────────────────────────┬──┬─────────────────────────────┬──┬─────────────────────────────┬───►
   └─BIDI──=──┬─PNLGRP──┬─────┘  └─CONTXTREV──=──┬─NO───┬──────┘  └─NBRSHAPE──=──┬─PNLGRP──┬─────┘
              ├─LTR─────┤                        └─YES──┘                        ├─ARABIC──┤
              └─RTL─────┘                                                        └─HINDI───┘

►──┬───────────────────────────────────────────┬───────────────────────────────────────────────────►◄
   └─SYMSWAP──=──┬─NO───┬──.──:ECLASS.─────────┘
                 └─YES──┘
```

The class definition (CLASS) tag defines variable types in variable pools and on panels. Attributes of the CLASS tag define the data type for the class of the dialog variable and the validity checking necessary for all variables of that class. The attributes of this tag also determine the methods for mapping an internal dialog variable of a particular class to or from a displayed form.

Other tags can be nested within the CLASS tag. These tags are listed in the following table. The table defines the order in which the tags must appear and specifies on which page more information can be found about each tag.

When more than one tag is listed with the same order number, all tags of that number can be mixed in any order. However, a tag with a higher order number cannot precede a tag with a lower order number. For example, a tag with an order number of three cannot precede a tag with an order number of one or two.

Table 57. Tags Allowed Between the CLASS and ECLASS Tag

Tag Name	Order	Page
TL (Translation list)	1	629
CHECK (Validity checking)	2	475

Required Attributes

NAME=*class-name*
The name assigned to the class. For more information on the rules for naming, see "Name Syntax" on page 469.

BASETYPE
The base data type for variables in the class. The BASETYPE for a class determines the internal storage representation and basic input and output editing characteristics for variables of a class.

CHAR *n*
Variables of this class are character strings of a fixed length. The number of bytes, *n*, is from 1 to 32 767.

IGC *n* **[OPEN | EITHER]**
Variables of this class are fixed-length character strings which may contain double-byte data surrounded by shift-out and shift-in characters. The number of bytes, *n*, is from 4 to 32 767.

OPEN indicates that variables defined with this class can contain both single-byte and double-byte data.

EITHER indicates that variables defined with this class can contain single-byte or double-byte data, but not both. If EITHER is specified, the number of bytes, *n* must be an even number.

The CASE attribute on this tag is not allowed for IGC classes. For IGC classes, processing for CHRID=PNLGRP on this tag depends on whether or not the field value contains double-byte character set (DBCS) data.

A dialog variable with a BASETYPE of IGC can be presented on a device that is not DBCS-capable, as long as the panel group object in which the variable is declared does not have TEXTMODE=DBCS specified on the panel group (PNLGRP) tag. This allows the same panel group object and application to support both single-byte character set (SBCS) and DBCS field values as long as the predefined text in the panel group is only SBCS.

GRAPHIC *w c* **[CHAR | POS]**
Variables of this class can contain only DBCS characters which are not surrounded by shift-out and shift-in characters or UCS-2 characters. The number of DBCS characters or UCS-2 characters, *w*, is from 1 to 16 383 (1 DBCS character equals 2 bytes). The CCSID, *c*, should only contain a CCSID using the UCS-2 encoding scheme.

CHAR indicates that the values specified for the optional width value are specified in number of UCS2 characters. If the optional width value is not specified, the default is set to 2 times *w*. This parameter is only allowed when a UCS2 CCSID has been specified.

POS indicates that the values specified for the optional width value are specified in number of display positions. If the width attribute is not specified, the default width is set to the value specified for *w*. This parameter is only allowed when a UCS2 CCSID has been specified.

The contents of a GRAPHIC dialog variable must only be DBCS data or UCS-2 data. When GRAPHIC is specified on this tag and the CCSID parameter is not specified, the field will contain graphic DBCS data. When GRAPHIC is specified on this tag and the CCSID parameter is specified with a CCSID using the UCS-2 encoding scheme, the field will contain UCS-2 data. In both cases, shift-out and shift-in characters are not placed in the variable pool by either the application or the UIM. On output, the UCS-2 data will be converted from the specified UCS-2 CCSID to the CCSID of the device. On input, the data will be converted from the device CCSID to the specified UCS-2 CCSID.

When the CCSID parameter is specified, several restrictions are enforced. First, all graphic fields using the CCSID parameter will not be allowed to span multiple lines on a print panel. The COND tag will be limited to comparing other variables of the same UCS-2 Level 1 encoding scheme. UCS-2 dialog variables will not be allowed on the CMD dialog command. Also, the following tags will not be allowed.

- CHECK tag
- TI tag

Like current graphic DBCS support, each UCS2 character is two bytes long. Since the length of data displayed on the device after a conversion from a UCS2 CCSID to the device CCSID is dependent upon what CCSID the device is capable of, the WIDTH attribute can be used to provide a more accurate field width. This allows you to bypass the default field width to either avoid truncation of field data or to increase the display space on the screen by decreasing the field length.

When the optional WIDTH attribute is not specified in conjunction with the GRAPHIC `basetype` with the CCSID parameter and the POS special value is not specified, the number of characters on the screen is two times the number of UCS2 characters. When the WIDTH attribute is specified in conjunction with the GRAPHIC `basetype` with the CCSID parameter, this value is used for the field length instead of default field width.

If POS has been specified in conjunction with the CCSID parameter, the value entered for the WIDTH attribute represents the number of display positions required to display the value.

If CHAR has been specified in conjunction with the CCSID parameter, the value entered for the WIDTH attribute represents the number of UCS2 characters that would be displayed on the screen. The same is true if neither CHAR nor POS was specified.

For example, a panel group contains the following line:

```
:class name=example
basetype='graphic 10 X' width=Y.
```

- X is the UCS-2 CCSID that the data is stored as. Y is the width of this field. If Y was not specified, then the length of the field on the screen is 20 (two times the number of UCS-2 characters).
- If you know that the UCS-2 data is constructed from single-byte data, then the field width, Y, could be specified as five UCS-2 characters so that the field would have a width of 10 single-byte characters on the screen.
- If you know that the UCS-2 data is constructed from double-byte data, then the field width, Y, could be specified as 11 UCS-2 characters so that the field would have a length of 22 single-byte characters on the screen. This would allow space for the shift-out and shift-in characters.

Consider the following equivalent definitions:

```
:class name=example1 basetype='graphic 10 13488 pos' width=20.
:class name=example2 basetype='graphic 10 13488 char' width=10.
:class name=example3 basetype='graphic 10 13488'.
```

The definitions create classes that contain 10 UCS2 characters, which would be displayed in 20 positions on the panel.

Consider another set of definitions for the same output:

```
:class name=example1 basetype='graphic 10 13488 pos' width=10.
:class name=example2 basetype='graphic 10 13488 char' width=5.
:class name=example3 basetype='graphic 10 13488'      width=5.
```

The definitions in this example define classes that contain 10 UCS2 characters, which would be displayed in 10 display positions on the panel.

On output, field data that is longer than the specified field length will be truncated. On input, if too many characters are entered into the UCS-2 field, then the field will be reverse image and an error appears on the error line stating that too many characters were entered. The maximum number of characters to enter will be displayed in the error message.

NAME ***n*** **[SIMPLE | GENERIC]**

Use a BASETYPE of NAME for System/38™ names and names which follow the same rules as System/38 object names, such as names of record formats.

Lowercase characters (a through z) are converted to upper case characters (A through Z) according to the same rules that commands follow. For more information on converting lowercase characters to uppercase characters, see the **CL** topic in the iSeries Information Center.

The number of bytes, *n*, is from 1 to 255.

SIMPLE indicates that the name entered must be allowed as a simple i5/OS name.

GENERIC indicates that the name must be a syntactically correct i5/OS generic name.

A translation list must be specified in the class definition if blanks or any other name value that is not valid is allowed as an input value for a dialog variable of this BASETYPE.

OBJNAME ***n*** **[SIMPLE | GENERIC | POSTO]**

Use a BASETYPE of OBJNAME for i5/OS object names. SIMPLE indicates that the name entered must be a single i5/OS object name.

GENERIC indicates that the name must be a syntactically correct i5/OS generic object name.

POSTO indicates that the name is used for a position-to field, and does not have to be a syntactically correct i5/OS name. Lowercase characters (a through z) are converted to uppercase characters (A through Z) unless the name is enclosed in double quotation marks. The number of bytes, *n*, is from 1 to 255.

When POSTO is specified, if the first and last character of the name is a double quotation mark, the last double quote is removed from the name. If the first character is not a quotation mark, all lowercase characters (a through z) are converted to uppercase characters (A through Z). When POSTO is specified, the name does not have to be a valid object name.

A translation list must be specified in the class definition if blanks or any other name value that is not valid is allowed as an input value for a dialog variable of this BASETYPE. For example, a translation list is necessary when *LIBL should be an allowed value for a variable containing the library name. For an example of a translation list, see “TL (Translation List)” on page 629.

The CASE attribute on this tag cannot be specified for an object name class. The UIM automatically adds or removes delimiter characters (quotation marks) and converts lowercase input values to uppercase as necessary.

BIN ***b***

Variables of this BASETYPE are signed and unsigned binary numbers. The number of bits, *b*, is either 15, 16, 31, 32, or 64.

’BIN 15’ and ’BIN 31’ are signed binary numbers. ’BIN 16’ and ’BIN 32’ are unsigned binary numbers.

’BIN 15’ is the same as BINARY(2) and ’BIN 31’ is the same as BINARY(4), as described in the **APIs** topic in the iSeries Information Center.

’BIN 64’ is an unsigned binary number.

PACKED *t f*
Variables of this class are packed decimal numbers. The number of digits, *t*, is from 1 to 63. The number of decimal positions, *f*, is from 0 to *t*.

ZONED *t f*
Variables of this class are zoned decimal numbers. The number of digits, *t*, is from 1 to 63. The number of decimal positions, *f*, is from 0 to *t*.

PTR
Variables of this class are pointers. Any variables of this `BASETYPE` cannot be displayed or printed. The data pointed to by a PTR variable can be displayed in a text area. For more information on text areas, see “TEXT (Text Area)” on page 623.

DATE *n*
The variable is a 7- or 8-byte string representing the date in the form *cyymmdd* or *yyyymmdd*. The representation is determined by the number *n*. A value of 2 for *n* indicates the *cyymmdd* format (the default value). A value of 4 for *n* indicates the *yyyymmdd* format. The date abbreviations are represented as follows.

For two-digit years:

c Century. Zero indicates years 19xx and one indicates the years 20xx.

yy Year

mm Month

dd Day

For four-digit years:

yyyy Year

mm Month

dd Day

TIME [ZONE]
A value of the base TIME class is a 6-byte string representing the time in the form *hhmmss*. The time abbreviations are represented as follows:

hh Hours

mm Minutes

ss Seconds

The optional `ZONE` attribute indicates that the time value supports an additional 10-byte time zone value.

zzzzzzzzzz
Time zone

With the optional `ZONE` attribute, the value is then a 16-byte string that represents the time in the form *hhmmsszzzzzzzzzz*

Time zones are not validated, no kind of time zone conversion is supported, and the time zone can be entirely blank. Literal TIME values specified in the `CHECK` or `COND` expressions, or on the `VALUES` attribute of the `TI` tag cannot have a time zone value. Time zones are specified only by the application, or from user input.

In addition, two special values are supported:

***LCL** The local time zone is used.

***SYS** The time zone of the system is used.

For example, a variable can contain the value `135900*LCL`, indicating a time of 13 hours, 59 minutes, and 0 seconds in the local time zone.

These special values are resolved before the value is updated in the variable pool.

ACTION
The variable is a 2-byte binary number with the same internal form as BIN 15, used as the option column for an action list or the selection column for a selection list.

When a variable with this `BASETYPE` is specified for the `VAR` attribute on a list column (LISTCOL) tag for an action list, the UIM determines the width of the input field based on the maximum `OPTION` attribute value specified for any list action (LISTACT) tag in the area.

Table 58. Attribute Summary for Each BASETYPE

BASETYPE	**Default WIDTH Attribute**	**Minimum WIDTH Attribute**	**Maximum WIDTH Attribute**	**Default SHIFT Attribute**	**Default CASE Attribute**	**Default BLANKS Attribute**
BIN 15	6	1	255	UPPER	UPPER	IGNORE
BIN 16	5	1	255	UPPER	UPPER	IGNORE
BIN 31	11	1	255	UPPER	UPPER	IGNORE
BIN 32	10	1	255	UPPER	UPPER	IGNORE
BIN 64	20	1	255	UPPER	UPPER	IGNORE
CHAR *n*	*n*	*n*	32767	NONE	MIXED	RESPECT
IGC *n*	*n*	*n*	32767	NONE	MIXED	RESPECT
GRAPHIC *w*	*w*	*w*	16383	NONE	MIXED	RESPECT
GRAPHIC *w c*	*w*	*w*/2	16383	NONE	MIXED	RESPECT
NAME *n*	*n*	*n*	255	UPPER	UPPER	IGNORE
OBJNAME *n*	*n*	*n*	255	NONE	See text	IGNORE
PACKED *t f* where *t*=*f*	*t*+3	*f*+3	255	UPPER	n/a	n/a
PACKED *t f* where *t*[=*f* and *f*>0	*t*+2	*f*+2	255	UPPER	n/a	n/a
PACKED *t f* where *t*[=*f* and *f*=0	*t*+1	1	255	UPPER	n/a	n/a
ZONED *t f* where *t*=*f*	*t*+3	*f*+3	255	UPPER	n/a	n/a
ZONED *t f* where *t*[=*f* and *f*>0	*t*+2	*f*+2	255	UPPER	n/a	n/a
ZONED *t f* where *t*[=*f* and *f*=0	*t*+1	1	255	UPPER	n/a	n/a
DATE 2	8	8	255	UPPER	n/a	n/a
DATE 4	10	10	255	UPPER	n/a	n/a
TIME	8	8	255	UPPER	n/a	n/a
ACTION	6 (See text)	1	255	UPPER	UPPER	IGNORE

Optional Attributes

WIDTH=*display-width***.**
The width of a display field in which values of this class are presented. For a `BASETYPE` of GRAPHIC, this value is the number of DBCS characters or UCS-2 characters. The defaults and allowable values for the display width depend on the `BASETYPE` attribute, as specified in Table 58.

For variables that use this class, a value larger than the default for the BASETYPE may be needed to support the external display form of values appearing in a translation list. For more information on translation lists, see "TL (Translation List)" on page 629.

CHRID=NONE | PNLGRP

If the CHRID parameter on the CRTPNLGRP command is *JOBCCSID, the compiler issues a warning message and the CHRID attribute on the CLASS tag is ignored. All variables are converted from the job CCSID to the device CHRID when *JOBCCSID is specified.

If NONE is specified, the data is assumed to be in the correct character set and code page for the display or printer device.

If PNLGRP is specified, the data associated with this class is interpreted with the character set and code page specified on the CHRID parameter on the CRTPNLGRP control language command. For devices that do not allow the downloading of this code page to handle the display of such data, character conversion occurs on both output and input for variables of this class. The conversion table for such cases is determined by the following rules:

```
                Character set    Code page
Panel group      xxx               yyy
Device           aaa               bbb
Output conversion table = QUSRSYS/Qyyyaaabbb
Input conversion table  = QUSRSYS/Qbbbxxxyyy
```

If the conversion table is not found, no conversion takes place. This conversion is done before any validity checking or inbound processing of a translation list.

For a BASETYPE of GRAPHIC, CHRID=PNLGRP is not allowed on this tag. The UIM does not perform character set and code page conversion on GRAPHIC fields containing DBCS data because there is no single-byte part of a GRAPHIC field. For GRAPHIC fields containing UCS-2 data, conversions will occur only between the specified UCS-2 CCSID and the device CCSID.

When CHRID=PNLGRP is specified for a class with BASETYPE of IGC, the UIM converts only the single-byte portion of the field. No conversion is done on the DBCS portion of the field.

SHIFT=NONE | UPPER

If NONE is coded, no special keyboard shift is used for fields of this class.

If UPPER is coded, the keyboard is shifted to uppercase for data entry for display input fields associated with variables of this class. The default keyboard shift is determined by the BASETYPE. For a summary of the BASETYPE and corresponding default, see Table 58 on page 483.

CASE=MIXED | UPPER

MIXED indicates that characters from any field of this class are preserved when assigned to the variable pool.

UPPER indicates that lowercase characters (a through z) are uppercased before assignment to the pool. This translation takes place after any processing for the CHRID attribute on this tag and before any validity checking or inbound processing of a translation list. For a BASETYPE of GRAPHIC, CASE=UPPER is not allowed. The default case is determined by the BASETYPE. For a summary of the BASETYPE and corresponding default, see Table 58 on page 483.

BLANKS=RESPECT | IGNORE

RESPECT indicates that blanks in the input field are preserved when the value is assigned to the variable pool.

IGNORE indicates that the value is left justified when it is assigned to the variable pool. The default is determined by the BASETYPE. The processing to remove leading blanks takes place after any processing for the CHRID and CASE attributes on this tag, and before validity checking or inbound processing of a translation list. For a summary of the BASETYPE and corresponding default, see Table 58 on page 483.

When BLANKS=IGNORE is specified for a BASETYPE of IGC, only leading SBCS blanks are removed from the field value. The processing to remove leading blanks stops at the first shift-out character in the field.

SUBST=DISPLAY | QUOTED

If DISPLAY is specified, the substitution values used on the CMD dialog command for this class of variables is the same as their displayed values.

If QUOTED is specified, the substitution values used on the CMD dialog command for this class of variables are enclosed in apostrophes, and apostrophes within the values are doubled. Values translated with the translation list (TL) tag are not quoted.

BIDI=PNLGRP | LTR | RTL

Sets the left-to-right or right-to-left orientation for variables of this class. This attribute is ignored when BIDI=NONE is specified on the PNLGRP tag.

PNLGRP indicates that the parameter from the BIDI attribute on the PNLGRP tag should be used to set the orientation for variables of this class.

LTR indicates that a left-to-right orientation should be used for variables of this class.

RTL indicates that a right-to-left orientation should be used for variables of this class.

The orientation defaults to PNLGRP for classes with a BASETYPE of CHAR and IGC. Orientation defaults to LTR for classes with a BASETYPE of NAME, OBJNAME, BIN, PACKED, ZONED, DATE, TIME, and ACTION.

The cursor direction is reversed for any input field which has an orientation opposite the panel's orientation. The cursor direction can be changed later when the CONTXTREV attribute of this tag is processed.

CONTXTREV=NO | YES

Specifies that a context-sensitive reversal is performed for bidirectional variables of this class. This attribute is ignored when BIDI=NONE is specified on the PNLGRP tag.

For CONTXTREV=NO, when the orientation of a variable does not match the orientation of the panel group, the value for each display line of the variable is reversed before it is displayed. On entry, the contents of each display line of a field that has an orientation opposite to the panel is also reversed before it is moved from the display field to the variable pool. NO is the default.

For CONTXTREV=YES, the contents of dialog variables are checked to determine the ultimate display orientation of the variable. The dialog variable value is checked before it is displayed. If the dialog variable is used in an input field, the value is checked again before the input value is copied to the variable pool. A description of the checking follows:

- When BIDI=LTR is specified on the PNLGRP tag,
  - The dialog variable is inverted if and only if the dialog variable value contains at least one Arabic or one Hebrew alphabetic character, *or* if it contains no Latin characters and BIDI=RTL is specified on the CLASS tag.
- When BIDI=RTL is specified on the PNLGRP tag,
  - The dialog variable is inverted if and only if the dialog variable value does not contain any Arabic or Hebrew alphabetic characters, and if either it contains Latin characters or BIDI=LTR is specified on the CLASS tag.

The Arabic and Hebrew code pages used to determine if the variable must have its display orientation flipped are described as follows:

- For code page 420
  - The following hex values represent Arabic characters: 42 through 49, 51, 52, 55 through 59, 62 through 69, 70 through 79, 80, 8A through 8F, 90, 9A through 9F, A0, AA through AF, B0 through B5, B8 through BF, CB, CD, CF, D0, and DA through DE.

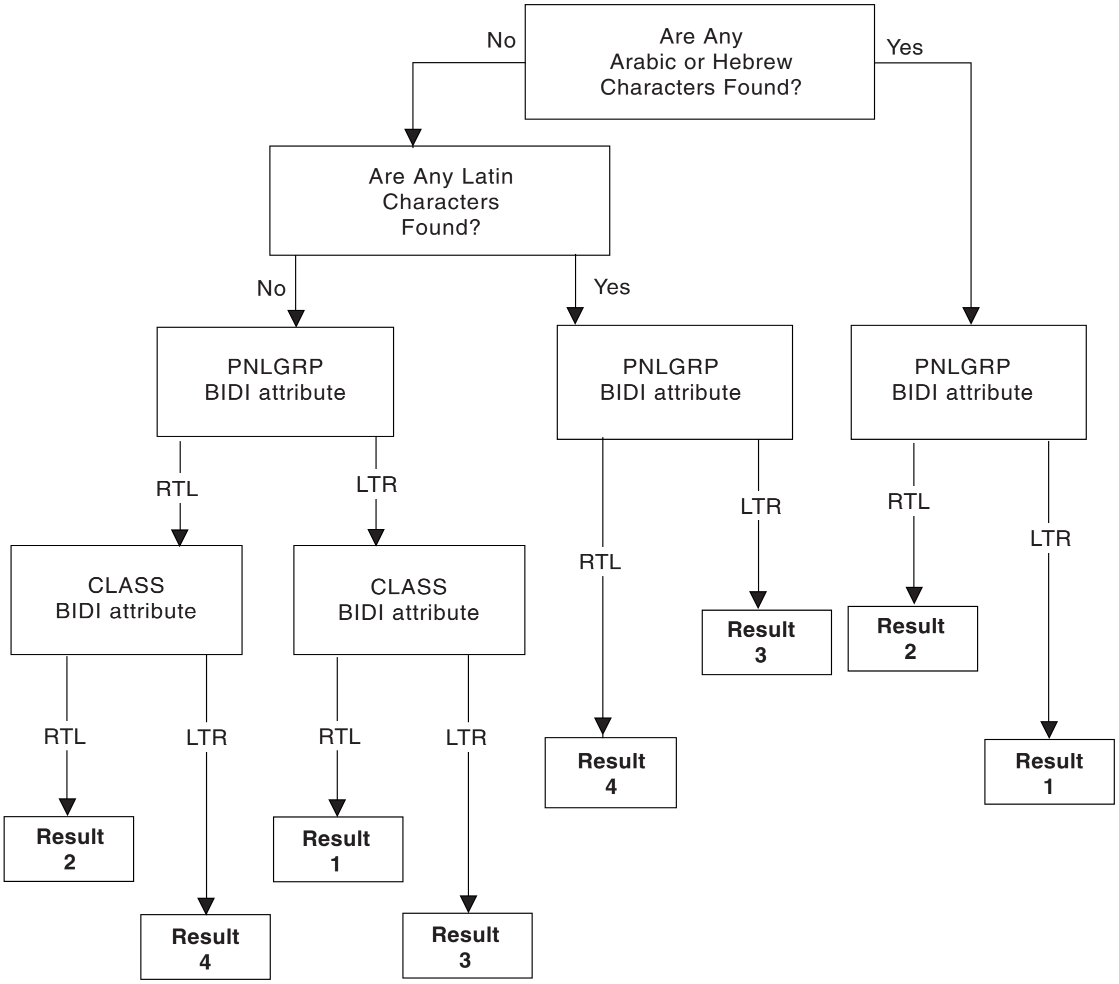
- The following hex values represent Latin characters: 6F, 81 through 89, 91 through 99, A2 through A9, C1 through C9, D1 through D9, DF, E2 through EB, ED, EE, EF, F0 through F9, and FB through FE.
- The remaining characters are not considered to be Arabic and Latin.

- For code page 424
  - The following hex values represent Hebrew characters: 41 through 49, 51 through 59, 62 through 69, and 71.
  - The following hex values represent Latin characters: 81 through 89, 91 through 99, A2 through A9, C1 through C9, D1 through D9, E2 through E9, and F0 through F9.
  - The remaining characters are not considered to be Hebrew and Latin.

For a CONTXTREV=YES input field, the initial cursor direction is determined the same way as for a CONTXTREV=NO input field. Then it is further processed and redetermined, based on the results of the following content inversion:

- BIDI=LTR is specified on the PNLGRP tag:
  - When BIDI=LTR is specified on the CLASS tag:
    - The cursor direction is inverted if and only if the content of the dialog variable is inverted.
  - When BIDI=RTL is specified on the CLASS tag:
    - The cursor direction is inverted if and only if the content of the dialog variable is *not* inverted *and* if it contains at least one Latin alphabetic character. (This prevents the inversion of the cursor direction in an empty field.)
- BIDI=RTL is specified on the PNLGRP tag:
  - When BIDI=LTR is specified on the CLASS tag:
    - The cursor direction is inverted if and only if the content of the dialog variable is *not* inverted.
  - When BIDI=RTL is specified on the CLASS tag:
    - The cursor direction is inverted if and only if the content is inverted *and* if the variable contains at least one Latin alphabetic character. (This prevents the inversion of the cursor direction in an empty field.)

For a detailed summary of how CONTXTREV=YES works for various fields and panels, see the following decision tree.



Result 1: Right-to-left cursor motion. Invert value of dialog variable.

Result 2: Right-to-left cursor motion.

Result 3: Left-to-right cursor motion.

Result 4: Left-to right cursor motion. Invert value of dialog variable.

RV2W060-0

NBRSHAPE=PNLGRP | ARABIC | HINDI

Controls the display shape of application-generated numbers for bidirectional variables of this class. This attribute is ignored when BIDI=NONE is specified on the PNLGRP tag, or when any code page other than 420 (Arabic) is used by the display device.

When PNLGRP is specified, the parameter from the NBRSHAPE attribute on the PNLGRP tag should be used for variables of this class. PNLGRP is the default.

When ARABIC is specified, the normal digits X'F0' through X'F9' are used for numbers.

When HINDI is specified, the translation of digits is performed with a set of two translation tables defined for each national language. The first table is used for output operations, and the second table is used for input operations.

The NBRSHAPE attribute allows the application program to not know the actual code points used to display the numeric digits.

SYMSWAP=NO | YES

Controls the exchange of symmetric characters for bidirectional variables of this class. This attribute is ignored when BIDI=NONE is specified on the PNLGRP tag. An example of two pairs of symmetric characters are:

’(’ with ’)’

’<’ with ’>’

When YES is specified, the symmetric character exchange is performed for both input and output operations on any field displayed with a right-to-left orientation. The situations where this can occur are:

- CONTXTREV=NO is specified on this tag for a variable that has a right-to-left orientation.
- CONTXTREV=YES is specified on this tag for a variable that has a right-to-left orientation, and the contents of the field causes the resulting field orientation to stay right-to-left.
- CONTXTREV=YES is specified on this tag for a variable that has a left-to-right orientation, and the contents of the field causes the resulting field orientation to change to right-to-left.

When SYMSWAP=NO is specified, symmetric character exchanges are not performed. NO is the default.

The complete list of symmetric characters for Hebrew (code page 424) is:

(with) Right and left parentheses, 4D with 5D

{ with } Right and left braces, C0 with D0

[with] Right and left square brackets, BA with BB

< with > Less than and greater than signs, 4C with 6E

≪ with ≫ Double less than and double greater than signs, 8A with 8B

The complete list of symmetric characters for Arabic (code page 420) is:

(with) Right and left parentheses, 4D with 5D

< with > Less than and greater than signs, 4C with 6E

All symmetric characters generated by the UIM are processed as if SYMSWAP=YES is specified. At this time, the symmetric characters include:

> Greater than symbol. Used to mark menu options and used in the command line prompt

( ) Right and left parentheses symbols. Used around an empty list message.

Example: Class Definitions

UIM Source

```
.* Class of options whose only legal values
.* are Y, N, and blank.
.* These values are translated to an
.* internal form.
 :class name=option
        basetype='bin 15'
        case=upper width=1.
 :tl msgid=MMM1234.
 :ti value=1.Y
 :ti value=2.N
 :ti value=3.
 :etl.
 :eclass.
 .*
 .* class of object names or the special
 .* value *all
 :class name=object
```

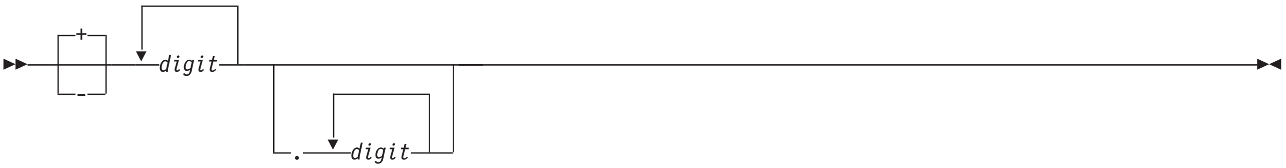
```
         basetype='objname 10 generic'.
:tl.
:ti value='"*ALL"'.*ALL
:etl.
:eclass.
.*
.* Class of options from 1-5 or 7-9,
.* in a field of width 1.
:class name=listoption
         basetype='bin 15' width=1.
:check range='1 5' range='7 9'.
:eclass.
```

Display Forms of Numeric Values

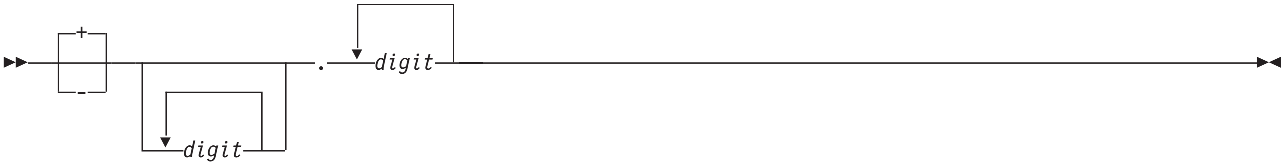
Numbers are usually displayed without leading zeros. If the absolute value of the number is less than one, a leading zero before the decimal point is supplied or suppressed, based on the value of the job’s decimal format separator attribute. Numbers are signed only if the number is negative. A decimal point is displayed only if there are fractional digits. The decimal point character used is the one defined by the job’s decimal format separator attribute. Numbers are displayed preserving all fraction digits.

If a value is too large to fit in the display field on output, the field is filled with plus signs. If a field consists entirely of plus signs on entry, no validity checks are performed on the field and the value of the variable in the variable pool is left unchanged.

Numeric literals must be of the form:



or



A comma may be used in place of a period for decimal point on input. If conversion to the internal form loses significant digits, the field is considered in error and the UIM displays an error message to the user without altering the contents of the variable pool. All nonzero digits after the decimal point are considered significant, and attempts to truncate result in an error. Leading zeros before the decimal point and trailing zeros after the decimal point are not considered significant. If a sign is not supplied, the value is assumed to be positive.

Binary values can also be translated to character strings on output and from character strings on entry. In this case, the value specified for the WIDTH attribute on this tag must allow for the width of the translated character string specified on the translation list item (TI) tag. For more information on translation lists, see “TL (Translation List)” on page 629 and “TI (Translation List Item)” on page 628.

Any variable with a BASETYPE of ACTION is handled with editing rules for a numeric value, except that a zero value is displayed as blanks, and a zero input value is not valid. Plus signs are not allowed on entry. Translation lists and validity check processing are not allowed for variables whose BASETYPE is ACTION.

Some examples of the display for numeric values are shown in the following table.

BASETYPE	WIDTH	Internal Value	Displayed Value	Notes
BIN 15	1	1	1	
BIN 15	1	-1	+	(Value too large to display)
BIN 15	5	1	1	
BIN 15	5	-1	-1	
BIN 15	5	-3456	-3456	
BIN 15	5	-32768	+++++	(Value too large to display)
BIN 16	5	65535	65535	
BIN 31	10	2147483645	2147483645	
BIN 31	9	2147483645	+++++++++	(Value too large to display)
BIN 32	10	4294967295	4294967295	
BIN 32	9	4294967295	+++++++++	(Value too large to display)
BIN 64	20	18446744073709551615	18446744073709551615	
BIN 64	20	18446744073709551615	+++++++++	(Value too large to display)
PACKED 2 2		.45	.45	Calculated field width is 5
PACKED 2 2		-.45	-.45	Calculated field width is 5
PACKED 5 2		000.45	0.45	Calculated field width is 7
PACKED 5 2		-000.45	-0.45	Calculated field width is 7
PACKED 5 2	8	000.45	0.45	
PACKED 5 2	8	-000.45	-0.45	
PACKED 5 2	8	123.45	123.45	
PACKED 5 2	8	-123.45	-123.45	
PACKED 5 2	4	-123.45	++++	(Value too large to display)
PACKED 5 2	4	003.45	3.45	
PACKED 5 0		00001	1	Calculated field width is 6
PACKED 5 0		00000	0	Calculated field width is 6
PACKED 5 0		-00000	0	Calculated field width is 6
PACKED 5 0		-00001	-1	Calculated field width is 6

Display Forms of Character, Date, and Time Values

Character values are displayed as they appear from the variable pool, padding with nulls on the right as necessary. If the value entered into the input field is not translated using a translation list, an error is reported to the user and the dialog variable is not updated if the value is longer than the BASETYPE of the variable after trailing blanks are removed. Only trailing SBCS blanks are removed; DBCS blanks are considered significant.

Dates are always displayed in the form specified by the date format and date separator of the current job. When a date is entered by a user, it must be in the format specified by the date format of the current job, but the separator characters specified by the date separator of the current job are optional. Because the date formats do not allow the entry of the century digit, it is assumed to be 0 if the year number is greater than 40. If the year number is less than or equal to 40, the century digit is assumed to be 1.

For those fields that are defined as ’DATE 4’, the user must enter the date with the full four-digit year. If the user enters a date with only a two-digit year, the date will be flagged as not valid.

Times are always displayed in a 24-hour time format, with the time separator as specified by the time separator of the current job. The separator characters specified by the time separator of the current job are optional. If a time is entered without a time separator, either four or six digits must be supplied.

If the time value contains a time zone, the time zone value is separated from the rest of the value by a single blank character. Upon input, the time zone value must be separated from the rest of the time value with a single space; otherwise, an error is signaled.

CMDLINE (Command Line)

```
►►──:CMDLINE──SIZE──=──SHORT──LONG──┬─────────────────────────────┬──.──┬──────────────────┬──────────►◄
                                    └─NAME──=──command-line-name──┘     └─instruction-text─┘
```

The command line (CMDLINE) tag specifies that the panel has a command line and provides additional command line information. This tag is allowed only for display panels. A command line may appear on any panel and may be independent of all areas on the panel, or it may be associated with an action list or menu area on the panel.

Only system commands can be entered on the command line, unless the panel contains a menu area or an action list area. If the panel contains a menu area, the user enters either a menu option or a command on the command line. For an action list, the user enters a string that is either run as a command or acts as parameter information for action list processing.

This tag must be the final tag in the panel, and is placed just before the end of the display panel. The CMDLINE tag and the option line (OPTLINE) tag are mutually exclusive.

Required Attribute

SIZE=SHORT | LONG

A SHORT command line takes up only one line of the display and fills up that line.

A LONG command line takes up two lines on the display.

SIZE=SHORT must be specified if a panel width other than 80 bytes (or 132 bytes) is specified on the display panel (PANEL) tag. SIZE=LONG is valid only for a 27-row by 132-byte panel or a 24-row by 80-byte panel.

When BIDI=RTL is specified on the panel group (PNLGRP) tag, all SIZE=LONG command lines are automatically formatted by the compiler to a one-line command line with a blank line separator between the command line and the function key area.

Optional Attribute

NAME=*command-line-name*

The name associated with the command line. This name can be used later with the Add Pop-Up Window (QUIADDPW) API to position a window near the command line.

For more information on the rules for naming, see “Name Syntax” on page 469.

Optional Text

instruction-text

The text appearing as instructions for the command line. The text is an implied paragraph.

When the display is formatted, any text that does not fit on one display line is formatted on multiple lines as necessary and indented two display positions. The text can be a maximum of 255 characters and can only contain the reverse text (RT) tag. If no text is provided, no instruction line is displayed above the command line.

COND (Condition Definition)

```
►►──:COND──NAME──=──condition-name──EXPR──=──'──conditional-expression──'──────────────►

►──┬──────────────────────────┬──────────────────────────────────────────────────────►◄
   └─EVAL──=──┬─ALWAYS─┬──.───┘
              └─ONCE───┘
```

The condition definition (COND) tag defines a condition that must be true if certain processing is to take place. Other tags may refer to this condition, which is evaluated before a panel is displayed. If the condition is true, the action or formatting specified by the referring tag is performed.

Required Attributes

NAME=*condition-name*
The name of the condition specified in the expression. The name must be unique within the panel group.

For more information on the rules for naming, see "Name Syntax" on page 469.

EXPR='*conditional-expression***'.**
A conditional expression is a true or false expression in the following form:

```
      ┌─*AND─────────────────┐
      │ ┌─*OR──────────────┐ │
      ▼ ▼                  │ │
►►──────────┬─A────────────┴─┴─────────────────────────────────────►◄
            ├─(──A──)──────┤
            └─*NOT──(──A──)┘
```

where A is an operand which can be one of the following:

- A conditional expression
- A comparison between a dialog variable and a literal value
- A comparison between two dialog variables
- A built-in function

When operand A is a conditional expression, the *AND and *OR logic becomes more complex.

The logical OR character (|) can be used in place of *OR, the ampersand character (&); can be used in place of *AND, and the logical NOT character (¬) can be used in place of *NOT. Since the logical OR and logical NOT characters are not in the invariant character set, their use is not recommended. For code page 00037, the common USA code page, the hexadecimal value of the logical OR character is X'4F', and the hexadecimal value of the logical NOT character is X'5F'. The UIM compiler uses these hexadecimal values regardless of the code page of the source.

When operand A is a comparison between a dialog variable and a literal value, or between two dialog variables, it must be in the following form:

```
►►──┬─dialog-variable-name─┬──relational-operator──┬─dialog-variable-name─┬──────────►◄
    └─literal-value────────┘                       └─literal-value────────┘
```

The relational operator may be specified as =, ¬=, >, <, >=, or <=, if the code page of the source maps the not character ([) to X'5F'. Otherwise, it is suggested that the special values *EQ, *NE, *GT, *LT, *GE, or *LE be used, respectively for the relational operator.

Comparisons may take place only between items of matching BASETYPE and precision as defined on the class definition (CLASS) tag. Comparisons between two literals are not allowed.

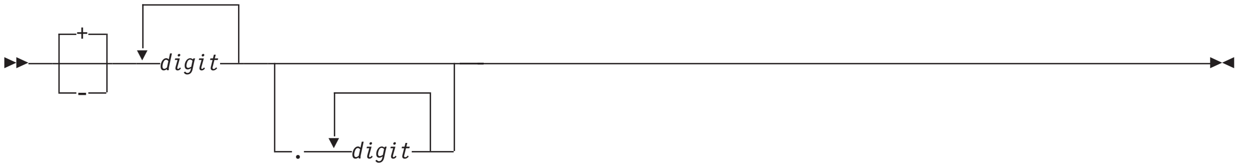
Character string literals must be enclosed in double quotation marks ("). The UIM does not automatically uppercase character strings.

If the BASETYPE of the class is TIME, a time zone value must not be specified.

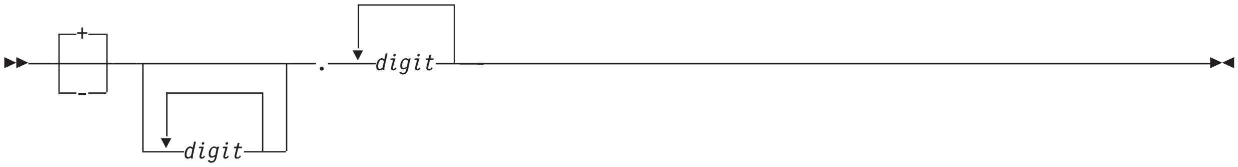
Hexadecimal strings are allowed as character strings. Hexadecimal strings cannot be used for comparison with dialog variables defined with a BASETYPE of GRAPHIC on the CLASS tag. For more information about specifying hexadecimal strings, see "Hexadecimal Literals" on page 472.

For a comparison between a dialog variable whose BASETYPE is GRAPHIC (as defined on the CLASS tag) and a literal, a double-byte character set (DBCS) graphic literal must be used. For more information about specifying a DBCS graphic literal, see "DBCS Graphic Literals" on page 471.

Numeric literals must be of the form:



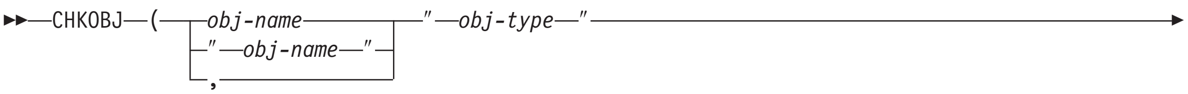
or



where *d* is a digit from 0 through 9.
This attribute is repeatable so that an expression can span several lines.
When operand A is a built-in function, it can be used as follows:

CHKOBJ

Evaluates to true if the object is found on the system and the current job possesses at least the level of authorization to the object specified by the authorities. Arguments must be character strings enclosed in double quotation marks ("). The object name follows i5/OS object naming conventions. The object type is any of the allowable object types for the DSPOBJD command and the authorities must be a single value or a list of authorizations to be checked for. The values of these authorizations are separated by blanks. The following syntax diagram illustrates the authorities and how to use them:



```
►─┬────────────────────────────────────────┬──────────────────────────────────────────────►◄
  └─,─"──┬─*CHANGE────────────────────┬─"─)─┘
         ├─*ALL───────────────────────┤
         ├─*USE───────────────────────┤
         ├─*EXCLUDE───────────────────┤
         ├─*AUTLMGT───────────────────┤
         │ ┌────────────────────────┐ │
         │ ▼  (1)                   │ │
         └─────────┬─*OBJEXIST───┬──┴─┘
                   ├─*OBJMGT─────┤
                   ├─*OBJOPR─────┤
                   ├─*OBJALTER───┤
                   ├─*OBJREF─────┤
                   ├─*ADD────────┤
                   ├─*DLT────────┤
                   ├─*READ───────┤
                   ├─*UPD────────┤
                   └─*EXECUTE────┘
```

Notes:

1 Each value can be used only once, with a maximum of 7.

If no authorities are specified, no authorization check is performed and the function becomes an existence check.

CHKPGM

The CHKPGM function accepts a qualified program name to be called as an exit program from UIM. If the program is not able to be called, the function evaluates to false. The exit program will determine if the function should be set to true or false and return an indicator to UIM. The program name must be enclosed in double quotation marks ("). The *LIBL special value can be used in place of the library name. If no library name is entered, *LIBL is the default.

The following syntax diagram illustrates the valid argument values for the CHKPGM function:

```
►►──CHKPGM──(──"──pgm-name──"──)──────────────────────────────────────────────────────►◄
```

CHKUSRCLS

The user class argument specified for the function is compared to the user class parameter from the user profile of the current job. The function evaluates to true if the user profile has the same or greater value than the function argument. The function argument must be enclosed in double quotation marks (").

The following syntax diagram illustrates the valid argument values for the user class:

```
►►──CHKUSRCLS──(──┬─*SECOFR──┬──)──────────────────────────────────────────────────────►◄
                  ├─*SECADM──┤
                  ├─*PGMR────┤
                  ├─*SYSOPR──┤
                  └─*USER────┘
```

```
CHKOBJ("OBJECT","*FILE","*USE")

CHKOBJ("PANELGRP","*PNLGRP")
          *AND CHKUSRCLS("*PGMR")

CHKOBJ("DOCUMENT","*DOC","*READ *UPD")
          *OR CHKUSRCLS("*SYSOPR")
```

```
*NOT(CHKOBJ("PROGRAM","*PGM"))
CHKPGM("*LIBL/PROGRAM")
CHKPGM("PROGRAM")
```

Optional Attribute

EVAL=ALWAYS | ONCE

Indicates whether or not the expression specified on the EXPR attribute of this tag must be evaluated every time the condition is referred to, or if it needs to be evaluated only once.

If EVAL=ALWAYS is used, the expression is evaluated to determine a new truth value every time the condition is referred to. All EVAL=ALWAYS conditions in a panel group or menu are evaluated each time a panel or menu is displayed. ALWAYS is the default.

If EVAL=ONCE is specified, the expression is evaluated only once for every open application that uses the panel group. The UIM only evaluates the expression for a condition of this type the first time the condition is referred to. Every reference after the first one uses the same truth value determined by the first reference. ONCE may be specified to improve performance for conditions that do not change while the application is open, particularly if the expression requires the evaluation of a function (like CHKOBJ) that may be costly.

Example: Conditioning an Option

The following example shows how the COND tag might be used to condition menu items. The first condition uses a complex expression which is split onto two source lines. The second condition checks for the application user's authority to use a menu on the system, and the third condition checks for the user class of the application user.

UIM Source

```
  ⋮
:cond name=a
      expr='(var1 = 100 *AND var2 > 0)'
      expr='*OR *NOT(var3 = 0)'.
:cond name=b
      expr='chkobj("MENU2","*MENU","*READ, *UPD")'.
:cond name=c
      expr='chkusrcls("*PGMR")'.

  ⋮
:menui
       cond=a
       help='option1/help'
       action='CMD CALL PGM01'
       option=1.Run report
:menui
       cond=b
       help='option2/help'
       action='MENU MENU2'
       option=2.Administration
:menui
       cond=c
       help='option30/help'
       action='MENU UTIL'
       option=30.Utilities
```

⋮

COPYR (Copyright)

```
►►──:COPYR──.──notice-text──────────────────────────────────────────────────►◄
```

The copyright (COPYR) tag provides a way to present a copyright notice. Only one COPYR tag is allowed. The copyright notice is displayed in the message area of the first panel displayed after the panel group is opened. This tag must be coded immediately after the panel group (PNLGRP) tag.

Required Text

notice-text
The text of the copyright notice that is displayed. The text must be between 1 to 76 characters.

DATA (Data Presentation Area)

Syntax for Display Panels:

```
►►──:DATA──DEPTH──=──┬─area-depth─┬──┬──────────────────────────┬──────────────►
                     └─'──*──'────┘  └─HELP──=──help-module-name─┘

►──┬──────────────────────┬──┬────────────────────┬─────────────────────────►
   │              ┌─SPACE─┐│  │           ┌─NO──┐  │
   └─BOTSEP──=────┼─NONE──┼┘  └─SCROLL──=──┴─YES─┴──┘
                  └─RULE──┘

►──┬──────────────────────┬──┬─────────────────┬────────────────────────────►
   │              ┌─1─────┐│  │            ┌─0─┐│
   └─LAYOUT──=────┼─2─────┼┘  └─MAXHEAD──=──┼─1─┼┘
                  └─HORIZ─┘                 ├─2─┤
                                            ├─3─┤
                                            └─4─┘

►──┬────────────────────────┬──┬─────────┬──.──┬────────────┬──:EDATA.──────►◄
   │                ┌─SPACE──┐│  └─COMPACT─┘     └─area-title─┘
   └─BODYSEP──=─────┼─INDENT─┼┘
                    ├─BOTH───┤
                    └─NONE───┘
```

Syntax for Print Panels:

```
►►──:DATA──┬──────────────────────┬──┬──────────────────────┬──────►
           │            ┌─SPACE─┐ │  │           ┌─1─────┐  │
           └─BOTSEP──=──┼─NONE──┼─┘  └─LAYOUT──=──┼─2─────┼─┘
                        └─RULE──┘                 └─HORIZ─┘

►──┬─────────────────────┬──┬──────────────────────────────────────┬──►
   │             ┌─0─┐   │  │             ┌─SPACE──┐                 │
   └─MAXHEAD──=──┼─1─┼─┘  └─BODYSEP──=──┼─INDENT─┼──┬─────────┬──┘
                 ├─2─┤                      ├─BOTH───┤  └─COMPACT─┘
                 ├─3─┤                      └─NONE───┘
                 └─4─┘

►──┬──────────────────────┬──.──┬────────────┬──:EDATA.──────────►◄
   │          ┌─NORMAL─┐  │     └─area-title─┘
   └─TYPE──=──┼─PROLOG─┼─┘
```

The data presentation area (DATA) tag describes a data presentation area in a panel. This tag is allowed for display panels and print panels. This area can be used for data entry items or output data items, or any combination of the two.

Other tags can be nested within the DATA tag. These tags are listed in the following table. The table defines the order in which the tags must appear, indicates which tags can be used in display panels only, print panels only, or both (specified by a D, P, or B, respectively), and specifies on which page more information can be found for each tag.

When more than one tag is listed with the same order number, all tags of that number can be mixed in any order. However, a tag with a higher order number cannot precede a tag with a lower order number. For example, a tag with an order number of three cannot precede a tag with an order number of one or two.

Table 59. Tags Allowed Between the DATA and EDATA Tags

Tag Name	Order	Use	Page
TOPINST (Top instruction line)	1	D	630
DATACOL (Data column)	2	B	507
DATAGRP (Data item group)	3	B	508
DATAI (Data item)	3	B	510
DATASLT (Data selection field)	3	D	518
BOTINST (Bottom instruction line)	4	D	474

Required Attribute

DEPTH=*area-depth* **| '*'**

The depth of the area in lines, including separators if any are specified. This attribute is required for display panels but is not allowed for print panels. If '*' is specified, the space remaining on the display after all else is allocated is given to this area. Only one area in the panel may have '*' coded.

Optional Attributes

HELP=*help-module-name*

Identifies online information explaining all items in the area. This attribute is allowed only for display panels. The help module name may be a name imported from another panel group, but must follow the rules for names outlined earlier in this chapter. For more information on the rules for naming, see "Name Syntax" on page 469.

A single help module name must be associated with every item in the area. If the HELP attribute is specified on the DATA tag, the help applies to all groups and items in the area, and the HELP attribute is not allowed on any data group (DATAGRP), data item (DATAI), data selection field (DATASLT), and data selection field choice (DATASLTC) tags in the area.

If no HELP attribute is specified on the DATA tag, a help module name must be associated with each item in the area by specifying the HELP attribute on the DATAGRP tags or the DATAI and DATASLT tags.

To provide the user with easy access to the online information for a data area with the LAYOUT=HORIZ attribute specified on this tag, specify the HELP attribute on the DATA tag instead of the HELP attribute on the individual DATAI tags.

BOTSEP=SPACE | NONE | RULE

Defines the bottom separator for the data presentation area. If SPACE is specified, a line of spaces is used. SPACE is the default.

NONE indicates that no separator line exists.

RULE indicates that a line of underscored spaces is used.

SCROLL=NO | YES

Specifies whether or not the area is scrollable. This attribute is allowed only for display panels. NO indicates that the area is not scrollable. NO is the default.

YES indicates that the data presentation area is scrollable. SCROLL=YES is not allowed when LAYOUT=HORIZ is specified. A line of spaces is used by the UIM to provide a line for the scroll information. If BOTSEP=SPACE, only one line of spaces is used unless this area also contains bottom instructions.

LAYOUT=1 | 2 | HORIZ

Indicates either the number of layout columns for a vertically formatted area or that a horizontal format should be used.

For LAYOUT=1, there is only one column of data items. For panels with WIDTH=80 bytes on the panel definition (PANEL) tag, the single column of data items is in positions 2 through 79. For panels with WIDTH=132, the single column of data items is in positions 2 through 131.

For LAYOUT=2, two layout columns of equal width are used for data items. The data items specified within the area appear in order, from top to bottom within a layout column and from left to right between layout columns. For panels with WIDTH=80, the two layout columns are in positions 2 through 38 and 43 through 79. For panels with WIDTH=132, the two layout columns of data items are in positions 2 through 64 and 69 through 131.

For LAYOUT=HORIZ, the data fields are listed horizontally across the width of the panel, with five spaces between items. This layout is useful if only two or three short values are necessary. It should be limited to a single display line, because the UIM does not vertically align items appearing on different lines within the area.

MAXHEAD=0 | 1 | 2 | 3 | 4

The maximum number of lines that can be used for column headings. The column headings are specified using the data column (DATACOL) tag. From zero to four lines can be specified. Zero is the default, meaning that no column headings are allowed. Only MAXHEAD=0 is valid for a data presentation area with a horizontal layout, because column headings are not allowed in horizontal presentation. For more information on column headings, see "DATACOL (Data Column)" on page 507.

When column headings are used, you may want to provide expansion space for national language translation by specifying a MAXHEAD value larger than the number of heading lines required. If this is not done, expansion space must be provided within the column widths determined by the DATACOL tag.

BODYSEP=SPACE | INDENT | BOTH | NONE

The type of visual separation distinguishing the body of the area from other elements on the display, particularly any top and bottom instruction lines within the area.

SPACE leaves a blank line after the last top instruction line and before the first bottom instruction line. Items in the body of the area begin in the leftmost position of the layout column. They are not indented with respect to instruction lines. If the area contains no top or bottom instruction lines, no blank lines are reserved before or after the area body.

INDENT is used to indent items in the body of the area by two bytes from the leftmost position in the layout column where the instruction lines begin. If the area contains top or bottom instruction lines, no blank line is reserved between the instructions and the area body except if the area is scrollable. If the area is scrollable, a blank line is reserved to provide a line for the scroll information.

BOTH leaves a blank line after the last top instruction line and before the first bottom instruction line, and also indents items in the body of the area by two bytes from the leftmost position in the layout column where the instruction lines begin. If the area contains no top or bottom instruction lines, no blank lines are reserved before or after the area body.

NONE does not leave a blank line between the instruction line and the body, and does not indent the body with respect to the layout column except if the area is scrollable. If the area is scrollable, a blank line is reserved to provide a line for the scroll information.

COMPACT

If COMPACT is specified, no blank lines are left between column headings and all items and outer groups in the area. If COMPACT is not specified, a blank line appears after the area column headings and between all items and outer groups.

TYPE=NORMAL | PROLOG

Indicates whether or not this data area is a prolog area. The prolog area is printed only once after the title line on the first page. This attribute is allowed only for the print head panel (PRTHEAD) tag. NORMAL is the default value.

Optional Text

area-title

The title of the area. If no text is specified, no title line is allocated to the area. The text must appear on the same or next line as the tag, can contain only the reverse text (RT) tag, and is a maximum of 55 characters long.

Print Formatting Considerations

Printed data areas are formatted similarly to data areas that are displayed, with the following exceptions:

- When printing, there is a minimum of two lines of data entries on a page besides the data group headings and the data column headings.
- If two lines of data entries do not fit on a page, a page eject occurs and the data entries are printed on the next page.

For formatting of data areas with LAYOUT=2, the UIM balances the data entries across the page and ignores keep processing.

This format may not look good in some cases, but LAYOUT=2 data areas should only be used on small data areas or when the user is not concerned about keep processing.

If there are data column headings, they are repeated on each page if the data area is continued onto another page.

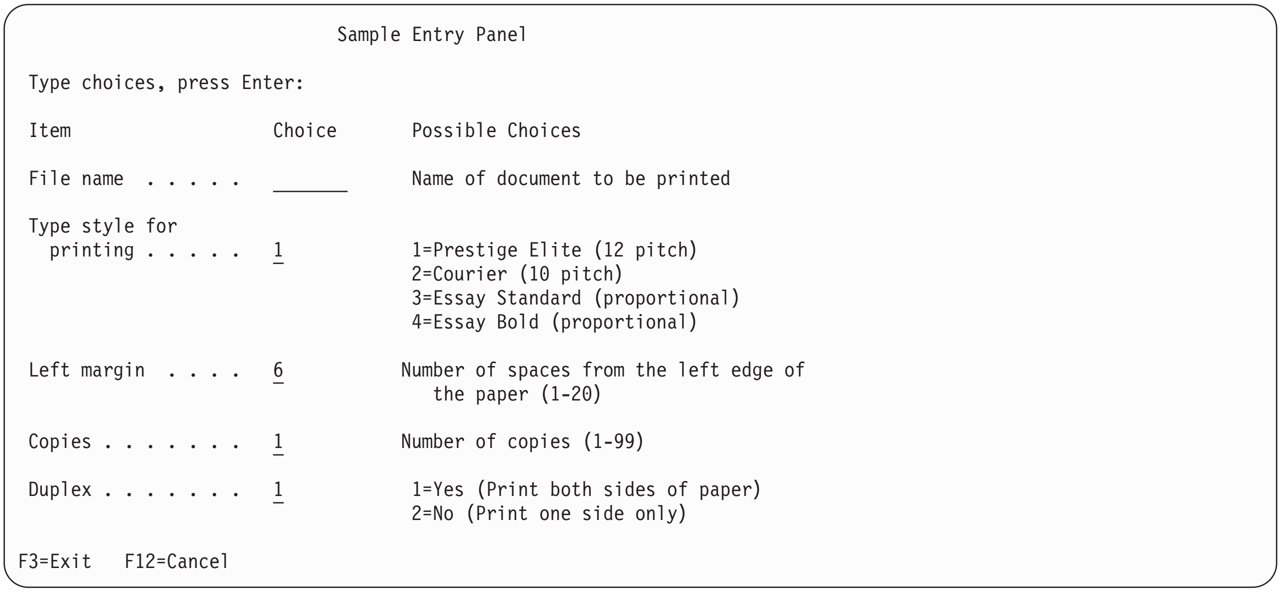
Example 1: Data Entry Panel

The following example shows a sample data entry panel. The panel has 5 prompts; 4 prompts have defaults supplied to the calling program by previously setting the dialog variables for the function.

UIM Source

```
:panel name=entry1
       help=hentry1
       topsep=space
       keyl=keys
       .Sample Entry Panel
:data depth='*'
      maxhead=2.
:topinst.Type choices, press Enter:
:datacol width=20.Item
:datacol width=10.Choice
:datacol width='*'.'Possible Choices'
:datai usage=inout
       help=hfname
       var=fname.File name
:datac.Name of document to be printed
:datai usage=inout
       help=hstyle
       var=prtstyle.Type style for printing
:datac.1=Prestige Elite (12 pitch)
:datac.2=Courier (10 pitch)
:datac.3=Essay Standard (proportional)
:datac.4=Essay Bold (proportional)
:datai usage=inout
       help=hmargn
       var=margin.Left margin
:datac.Number of spaces from the left
edge of the paper (1-20)
:datai usage=inout
       help=hcopy
       var=copies.Copies
:datac.Number of copies (1-99)
:datai usage=inout
       help=hduplx
       var=duplex.Duplex
:datac.1=Yes (Print both sides of paper)
:datac.2=No (Print one side only)
:edata.
:epanel.
```

Results



```
                             Sample Entry Panel

 Type choices, press Enter:

 Item                   Choice       Possible Choices

 File name  . . . . .   _______      Name of document to be printed

 Type style for
   printing . . . . .   1             1=Prestige Elite (12 pitch)
                                      2=Courier (10 pitch)
                                      3=Essay Standard (proportional)
                                      4=Essay Bold (proportional)

 Left margin  . . . .   6            Number of spaces from the left edge of
                                        the paper (1-20)

 Copies . . . . . . .   1            Number of copies (1-99)

 Duplex . . . . . . .   1             1=Yes (Print both sides of paper)
                                      2=No (Print one side only)

F3=Exit   F12=Cancel
```

Example 2: Two-Column Format in a Data Entry Panel

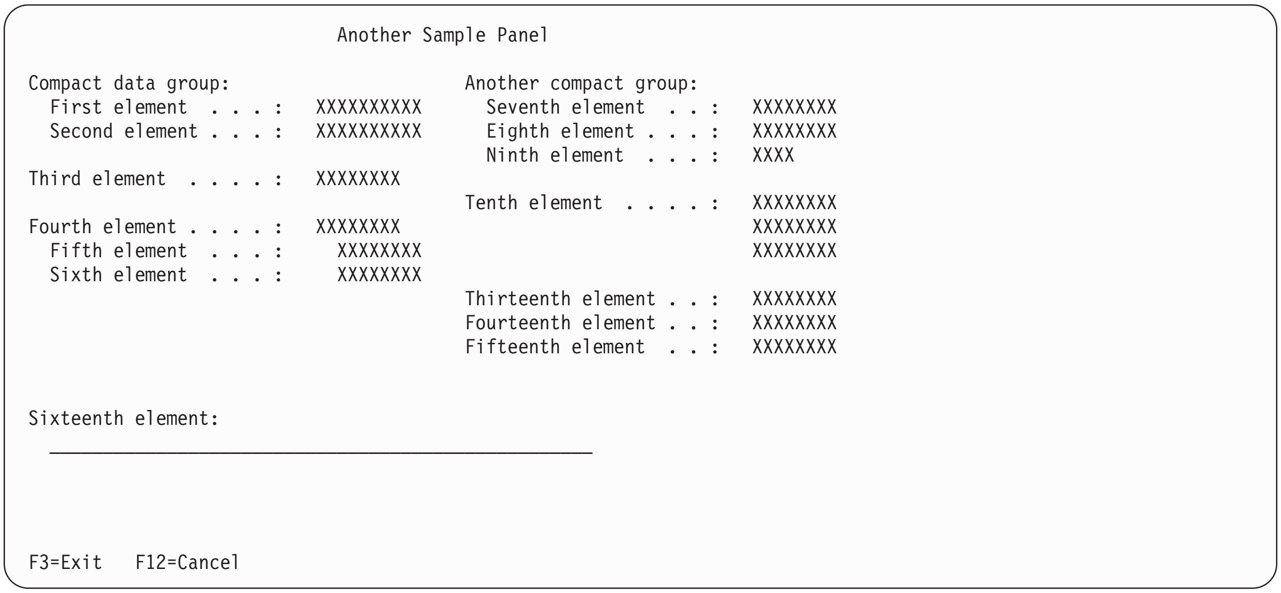
The following example shows a data area using two-column format containing output items, and a form of data entry field that has no prompt.

UIM Source

```
:panel name=panelx
       help=helpx
       topsep=space
       keyl=keys
       .Another Sample Panel
:data depth=14 layout=2.
:datacol width='24'.
:datacol width='*'.
:datagrp compact.Compact data group
:datai usage=out
       help=hvar1
       var=var1.First element
:datai usage=out
       help=hvar2
       var=var2.Second element
:edatagrp.
:datai usage=out
       help=hvar3
       var=var3.Third element
:datagrp grpsep=qindent
       compact.
:datai usage=out
       help=hvar4
       var=var4.Fourth element
:datai usage=out
       help=hvar5
       var=var5.Fifth element
:datai usage=out
       help=hvar6
       var=var6.Sixth element
:edatagrp.
:datagrp compact.Another compact group
:datai usage=out
       help=hvar7
       var=var7.Seventh element
```

```
:datai usage=out
       help=hvar8
       var=var8.Eighth element
:datai usage=out
       help=hvar9
       var=var9.Ninth element
:edatagrp.
:datagrp help=hvar1012
       grpsep=none compact.
:datai usage=out
       var=var10.Tenth element
:datai usage=out
       var=var11 cond=cond11.
:datai usage=out
       var=var12 cond=cond12.
:edatagrp.
:datagrp grpsep=none
       compact.
:datai usage=out
       help=hvar13
       var=var13.Thirteenth element
:datai usage=out
       help=hvar14
       var=var14.Fourteenth element
:datai usage=out
       help=hvar15
       var=var15.Fifteenth element
:edatagrp.
:edata.
:data depth=3
       bodysep=none.
:datacol width=0.
:datacol width='*'.
:datagrp.Sixteenth element
:datai usage=inout
        help=hvarx
        var=varx.
:edatagrp.
:edata.
:epanel.
```

Results



```
                             Another Sample Panel

Compact data group:                      Another compact group:
  First element  . . . :   XXXXXXXXXX      Seventh element  . . :   XXXXXXXX
  Second element . . . :   XXXXXXXXXX      Eighth element . . . :   XXXXXXXX
                                           Ninth element  . . . :   XXXX
Third element  . . . . :   XXXXXXXX
                                         Tenth element  . . . . :   XXXXXXXX
Fourth element . . . . :   XXXXXXXX                                 XXXXXXXX
  Fifth element  . . . :     XXXXXXXX                               XXXXXXXX
  Sixth element  . . . :     XXXXXXXX
                                         Thirteenth element . . :   XXXXXXXX
                                         Fourteenth element . . :   XXXXXXXX
                                         Fifteenth element  . . :   XXXXXXXX


Sixteenth element:
  ___________________________________________________



F3=Exit   F12=Cancel
```

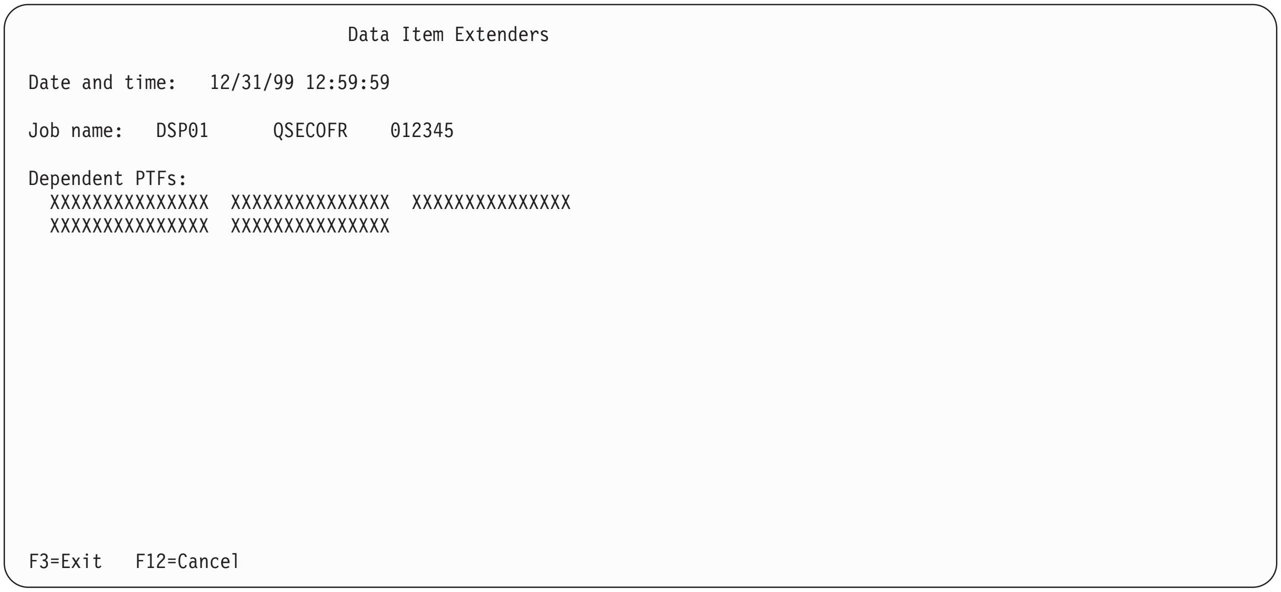
Example 3: Two Presentation Areas for Data Items

The following two examples show three data areas, which are initialized from the dialog variables in the variable pool. The first two areas are aligned horizontally, and the third area is formatted vertically.

UIM Source

```
:panel name=xmp3
       keyl=x1
       help=hxmp3
       .Data Item Extenders
:data depth=2
       layout=horiz
       botsep=space.
:datai  usage=out
       var=date
       help=hdatetime.Date and time
:dataix usage=out
       var=time
       newline=no
       itemsep=1.
:edata.
:data depth=2
       layout=horiz
       botsep=space.
:datai  usage=out
       var=jobname
       help=hdatetime.Job name
:dataix usage=out
       var=user
       newline=no
       itemsep=1.
:dataix usage=out
       var=jobnbr
       newline=no itemsep=1.
:edata.
:data depth='*'
       layout=1
       botsep=space
       compact.
:datacol width=0.
:datacol width='*'.
:datagrp grpsep=indent
       help=dependptfs
       compact.Dependent PTFs
:datai  usage=out
       var=ptf1.
:dataix usage=out
       var=ptf2.
:dataix usage=out
       var=ptf3.
:dataix usage=out
       var=ptf4
       newline=yes.
:dataix usage=out
       var=ptf5.
:edatagrp.
:edata.
:epanel.
```

Results



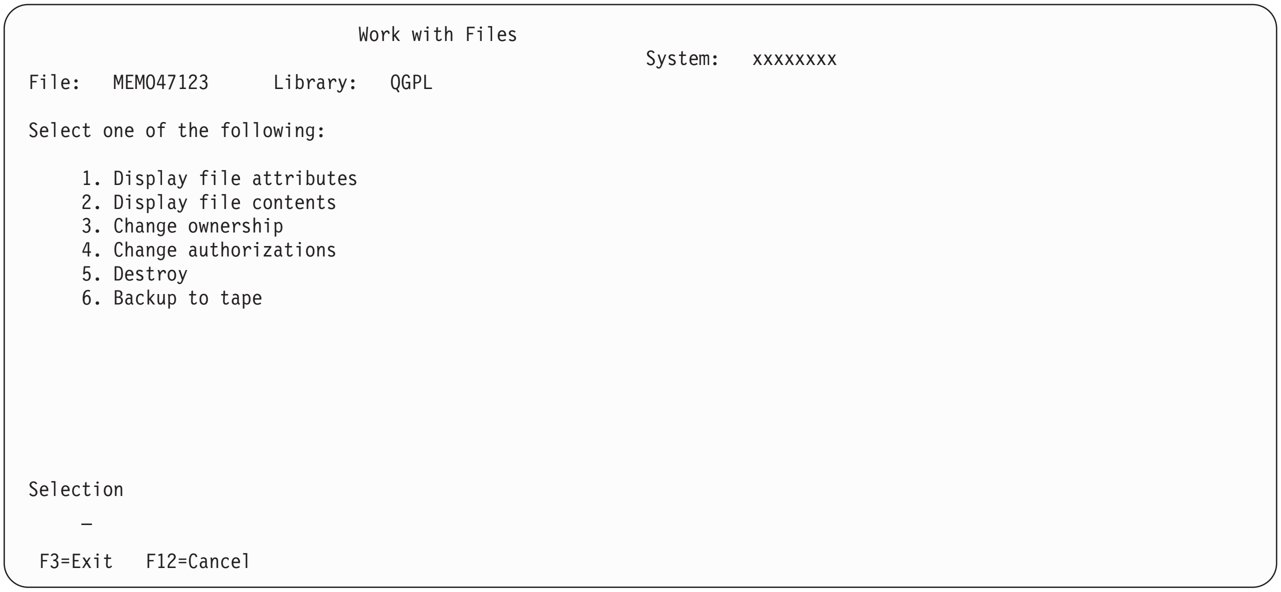
```
                              Data Item Extenders

 Date and time:   12/31/99 12:59:59

 Job name:   DSP01      QSECOFR    012345

 Dependent PTFs:
   XXXXXXXXXXXXXXX  XXXXXXXXXXXXXXX  XXXXXXXXXXXXXXX
   XXXXXXXXXXXXXXX  XXXXXXXXXXXXXXX

















 F3=Exit   F12=Cancel
```

Example 4: Data Presentation Area with a Menu Area

UIM Source

```
:panel name=xxx
       keyl=x1
       help=hxxx
       .Work with Files
:data depth=2
       layout=horiz.
:datai var=fname
       help='filename'.File
:datai var=lname
       help='libname'.Library
:edata.
:menu depth='*'.
:topinst.Select one of the following:
:menui action='return 1'
       help='hoption1'
       option=1.Display file attributes
:menui action='return 2'
       help='hoption2'
       option=2.Display file contents
:menui action='return 3'
       help='hoption3'
       option=3.Change ownership
:menui action='return4'
       help='hoption4'
       option=4.Change authorizations
:menui action='return 5'
       help='hoption5'
       option=5.Destroy
:menui action='return 6'
       help='hoption6'
       option=6.Backup to tape
:emenu.
:optline.Selection
:epanel.
```

Results



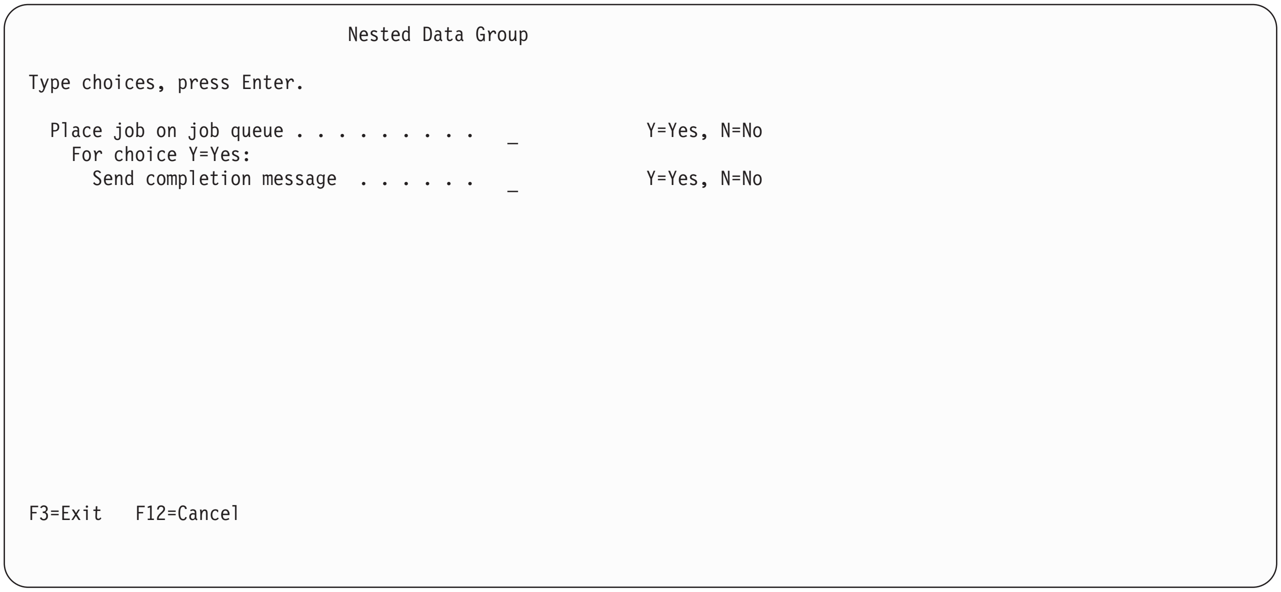
```
                                 Work with Files
                                                           System:   xxxxxxxx
 File:   MEMO47123      Library:   QGPL

 Select one of the following:

      1. Display file attributes
      2. Display file contents
      3. Change ownership
      4. Change authorizations
      5. Destroy
      6. Backup to tape








 Selection
      _

  F3=Exit   F12=Cancel
```

Example 5: Data Entry Panel with a Nested Data Group

UIM Source

```
:panel name=nestg
       help=helpn
       topsep=space
       keyl=keys
       .Nested Data Group
:data depth='*'
       scroll=no.
:topinst.Type choices, press Enter.
:datacol width=40.
:datacol width=10.
:datacol width='*'.
:datagrp compact
       grpsep=none.
:datai usage=inout
       help=hjobq
       var=jobq.Place on job queue
:datac.Y=Yes, N=No
.* The following "datagrp" tag is
.*needed to cause indentation for
.* the "For choice Y=Yes" data group.
:datagrp compact
       grpsep=indent.
:datagrp compact
       grpsep=indent.For choice Y=Yes:
:datai usage=inout
       help=hcmpmsg
       var=cmpmsg.Send completion message
:datac.Y=Yes, N=No
:edatagrp.
:edatagrp.
:edatagrp.
:epanel.
```

Results



```
                              Nested Data Group

 Type choices, press Enter.

   Place job on job queue . . . . . . . . .   _            Y=Yes, N=No
     For choice Y=Yes:
       Send completion message  . . . . . .   _            Y=Yes, N=No














 F3=Exit   F12=Cancel
```

DATAC (Data Item Choices)

```
►►──:DATAC──┬─────────────────────────────────┬─.─┬──────────────────┬──────────────────────►◄
            └─CHOICE──=──dialog-variable-name─┘   └─text-for-choices─┘
```

The data item choices (DATAC) tag provides the text for the possible choices column of a data presentation area. This tag is allowed for display panels and print panels. It is not valid if only two data column (DATACOL) tags are specified for the area, or if LAYOUT=HORIZ is specified on the data presentation area (DATA) tag for the area.

The DATAC tag must appear after the corresponding data item (DATAI) tag. More than one DATAC tag may be specified after the DATAI tag to provide several lines of text. If no DATAC tags are entered, no text exists in the possible choices column for the data item.

Optional Attribute

CHOICE=*dialog-variable-name*

The name of a dialog variable containing the choices text to be displayed. The dialog variable must be defined with a width less than or equal to the width specified on the third DATACOL tag for the data area, the possible choices column. If the CHOICE attribute is used, no *text-for-choices* can be specified after the period of this tag.

Dialog variables must be defined with a BASETYPE of CHAR, IGC, or BIN on the class definition (CLASS) tag.

The error state of the dialog variable is not used for determining the highlighting of the text.

Special formatting for IGC. (The abbreviation IGC is used in commands and keywords to represent double-byte character set functions.) When a dialog variable with a BASETYPE of IGC is specified on the CLASS tag, the UIM does special formatting. If the variable value begins with a shift-out character (X'0E'), the UIM shifts the value one byte to the left to preserve vertical alignment with data choices on other lines.

Optional Text

text-for-choices

The text is an implied paragraph. Lines are formatted as necessary onto multiple lines within the column and indented two spaces from the beginning of the column. The text is a maximum of 255 characters and can only contain the reverse text (RT) tag. The *text-for-choices* is required unless the CHOICE attribute is specified on this tag.

DATACOL (Data Column)

```
►►──:DATACOL──WIDTH──=──┬─column-width─┬──.──┬────────────────┬───────────────────────────────────►◄
                        └─'──*──'──────┘     └─column-heading─┘
```

The data column (DATACOL) tag specifies the width of the item, choice, and possible choices columns, and can also provide column headings. This tag is allowed for display panels and print panels. It is not valid if LAYOUT=HORIZ is specified for the data presentation area. It must appear after the data presentation area (DATA) tag for the area and before the first data item (DATAI), data selection field (DATASLT), or data group (DATAGRP) tag.

Two or three DATACOL tags must be specified when the LAYOUT attribute of on the DATA tag specifies 1 or 2. If only two DATACOL tags are used, the area has no possible choices column and no data item choices (DATAC) tags are allowed in the area. If three DATACOL tags are specified, all three columns are allocated, including the possible choices column.

Required Attribute

WIDTH=*column-width* **|** **'*'**

The width, in bytes, for the column. The width specified must be a positive integer, and the sum of the widths of the columns and separators in the data presentation area must not exceed the width of the layout column determined by the DATA tag. WIDTH=0 is allowed only for the first DATACOL tag; it formats the area without an item column for data item or data selection field prompt text.

If '*' is coded, the remainder of the area width is used for the column. Only one DATACOL in the area may have '*' specified. A three-column separator is maintained between columns, except when WIDTH=0 is specified for the first DATACOL tag. In this case, the item column is not allocated and there is no separator in front of the choice column.

The value for any data item may span columns, but must begin on the correct column boundary. The possible choices text for that item, assuming three DATACOL tags are specified, begins in the location assigned to the possible choices. This possible choices text may be on the same or next line as the data item field.

Optional Text

column-heading

The column heading placed above the data items in the data area. If no column heading is specified, none is displayed.

The text may appear on more than one line and can contain only the reverse text (RT) tag. Each word of the column heading is placed on a new line. If multiple words are necessary in a line of the column heading, they must be enclosed in apostrophes ('). Each word or quoted string must fit within the column width defined by the WIDTH attribute on this tag. The maximum number of words or quoted strings allowed is specified by the MAXHEAD attribute on the DATA tag. Column headings are not allowed if MAXHEAD=0 is used for the data area.

The column headings are left-justified. If no column heading text is specified on any DATACOL tag for the area, no lines are reserved on the display for heading information.

DATAGRP (Data Group)

Syntax for Display Panels:

```
►►──:DATAGRP──┬───────────────────────────┬──┬──────────────────────────┬──────►
              └─HELP──=──help-module-name─┘  └─NAME──=──data-group-name─┘

►──┬──────────────────────────┬──┬────────────────────────┬────────────────────►
   │               ┌─INDENT──┐│  └─COND──=──condition-name─┘
   └─GRPSEP──=─────┼─QINDENT─┼┘
                   └─NONE────┘

►──┬─────────┬──.──┬───────────────┬──:EDATAGRP.──────────────────────────────►◄
   └─COMPACT─┘     └─group-heading─┘
```

Syntax for Print Panels:

```
►►──:DATAGRP──┬──────────────────────────┬──┬─────────────────────────┬────────►
              │               ┌─INDENT──┐│  └─COND──=──condition-name─┘
              └─GRPSEP──=─────┼─QINDENT─┼┘
                              └─NONE────┘

►──┬─────────┬──.──┬───────────────┬──:EDATAGRP.──────────────────────────────►◄
   └─COMPACT─┘     └─group-heading─┘
```

The data group (DATAGRP) tag is used to group data items and data selection fields in a data presentation area. This tag is allowed for display panels and print panels. One data group may be nested within another data group; up to four levels of data group nesting are allowed, including the outermost group. This tag is not valid if LAYOUT=HORIZ is specified for the data presentation area (DATA) tag.

The UIM does keep processing for data item groups. When data groups are nested within other data groups, keep processing is done for all the groups. If an outer group must be split, keep processing is attempted for groups nested within it. If the number of lines required for an entire group exceeds the number of lines remaining in the layout column, the entire group is forced to the next column. In the case of single-column layout, the entire group is forced to the next scrollable page. Any data group that becomes too large for one layout column begins at the top of one column and continues onto as many columns or pages as required for all items.

Optional Attributes

HELP=*help-module-name*

Identifies online information explaining all items in the group. This attribute is allowed only for display panels. The help module name may be a name imported from another panel group, but must follow the rules for names outlined earlier in this chapter. For more information on the rules for naming, see “Name Syntax” on page 469.

A single help module name must be associated with every item in the area. If the HELP attribute is specified on the DATA tag, the help applies to all groups and items in the area, and the HELP attribute is not allowed on any data group (DATAGRP), data item (DATAI), data selection field (DATASLT), and data selection field choice (DATASLTC) tags in the area.

If the HELP attribute is specified on the DATAGRP tag, the help applies to all items in the group, and the HELP attribute is not allowed on any DATAI, DATASLT, DATASLTC, or nested DATAGRP tags within the data group.

If no HELP attribute is specified on the DATA tag or on any outer or nested DATAGRP tags, the HELP attribute is required on all DATAI and DATASLT tags within the group.

NAME=*data-group-name*
The name associated with the group. This name can be used with the Add Pop-Up Window (QUIADDPW) API to position a window near this data group. This attribute is allowed only for display panels.

For more information on the rules for naming, see "Name Syntax" on page 469.

GRPSEP=INDENT | NONE | QINDENT
Specifies what type of visual separation should be used to distinguish the items in the group.

INDENT is used to indent the left-most column, prompt, or value for every data item, group heading of a nested data group, or data selection field within the group by two positions. Only the left-most column is indented. The group heading is not indented.

QINDENT is used to indent each data item or nested data group or data selection field, except the first one in the group. The first item or imbedded group within this data item group are not indented. All other items, fields, and imbedded groups are indented. Both the prompt column and the value column of a data item or data selection field are indented by two positions. The group heading of an nested data group and all items within it are indented by two positions.

The determination of which data items, selection fields and groups are indented is made when the panel group is created and is unaffected by conditioning. This option presents data items in the same format the control language (CL) prompter uses for qualified names. Because qualified name indenting of a value would be backward for a BIDI=RTL panel group, the value column is not indented when QINDENT is specified for a BIDI=RTL panel group. The prompt text of each item after the first one is still indented.

NONE does not indent any data items, data selection fields, or nested data item groups in the group. This option is not recommended when group heading text is specified. It can be used to group related data items, data item groups, and data selection fields together without a group heading. It does this by defining a compact group of items and nested groups in a non-compact area.

COND=*condition-name*
The group is displayed or printed only if the condition specified is true. All tags within the group appear only if this data group appears. The condition must be defined in the panel group prolog with the condition definition (COND) tag.

COMPACT
If COMPACT is specified, then no blank lines are left between individual items or nested data group headings. If this data group is nested inside another data group for which COMPACT is specified, COMPACT is implied for this data group whether it is specified or not.

Optional Text

group-heading
The heading placed above the data items of the group. The text occupies only one line of the panel, but may span several columns. The text must appear on the same line or next line as the tag and can only contain the reverse text (RT) tag.

If the text is omitted, no lines are allocated on the display for a group heading, but the items between the DATAGRP and EDATAGRP tags are processed as a group with respect to keep processing on the display and other UIM operations.

A colon is added to the end of the text if one is not already specified and if there is room for it.

DATAI (Data Item)

Syntax for Display Panels:

```
►►──:DATAI──VAR──=──dialog-variable-name──USAGE──=──┬─OUT───┬────────────────────►
                                                    └─INOUT─┘

►──┬─────────────────────────────┬──┬──────────────────────────┬──────────────────►
   └─HELP──=──help-module-name───┘  └─NAME──=──data-item-name──┘

►──┬──────────────────────────┬──┬─────────────────────────┬──────────────────────►
   │                ┌─BEFORE─┐│  │               ┌─AFTER─┐ │
   └─PMTLOC──=──────┴─ABOVE──┴┘  └─CHCLOC──=─────┴─ABOVE─┴─┘

►──┬──────────────────────┬──┬──────────────────────────┬─────────────────────────►
   │            ┌─LEFT──┐ │  └─JUSTIFY──=──┬─LEFT───────┘
   └─ALIGN──=───┼─RIGHT─┼─┘                ├─RIGHT─┤
                ├─START─┤                  ├─START─┤
                └─END───┘                  └─END───┘

►──┬────────────────────────┬──┬──────────────────────┬──┬───────────────────────┬►
   │               ┌─NO──┐  │  │             ┌─YES─┐  │  │              ┌─YES─┐  │
   └─REQUIRED──=───┴─YES─┴──┘  └─CSRLOC──=───┴─NO──┴──┘  └─DISPLAY──=───┴─NO──┴──┘

►──┬────────────────────────┬──┬──────────────────────────┬───────────────────────►
   │               ┌─NO──┐  │  └─COND──=──condition-name──┘
   └─AUTOENTR──=───┴─YES─┴──┘

►──┬───────────────────────────────┬──┬────────────────────────────────────┬──────►
   └─PROMPT──=──' ──action-text──' ┘  └─DSPVALUE──=──dialog-variable-name──┘

►──.──┬────────────────┬──────────────────────────────────────────────────────────►◄
      └─data-item-text─┘
```

Syntax for Print Panels:

```
►►──:DATAI──VAR──=──dialog-variable-name──┬─────────────────┬─────────────────────►
                                          └─USAGE──=──OUT───┘

►──┬──────────────────────────┬──┬─────────────────────────┬──────────────────────►
   │                ┌─BEFORE─┐│  │               ┌─AFTER─┐ │
   └─PMTLOC──=──────┴─ABOVE──┴┘  └─CHCLOC──=─────┴─ABOVE─┴─┘

►──┬──────────────────────┬──┬──────────────────────────┬─────────────────────────►
   │            ┌─LEFT──┐ │  └─JUSTIFY──=──┬─LEFT───────┘
   └─ALIGN──=───┼─RIGHT─┼─┘                ├─RIGHT─┤
                ├─START─┤                  ├─START─┤
                └─END───┘                  └─END───┘

►──┬──────────────────────────┬──.──┬────────────────┬────────────────────────────►◄
   └─COND──=──condition-name──┘     └─data-item-text─┘
```

The data item (DATAI) tag defines an item placed in a data presentation area. This tag is allowed for display panels and print panels.

The data item extender (DATAIX) tag can be used to specify additional dialog variables, which are displayed as part of the data item. The data selection choices (DATAC) tag can be used to specify possible choices text, which is displayed as part of the data item. The DATAIX and DATAC tag are considered part of the DATAI tag with respect to formatting, conditioning, scrolling, and help.

Note that the prompt, field and extenders, and possible choices text must fit within the data area. The UIM does not allow an individual data item to be split while scrolling.

Required Attributes

VAR=*dialog-variable-name*

The name of the dialog variable constructing the data item. The current value of the dialog variable is displayed or printed.

If the item appears in an area with LAYOUT=1 specified on the data presentation area (DATA) tag, its value can exceed the width of the choice column. In this case, the value is presented as a single field that wraps onto as many display lines as necessary. If LAYOUT=2 is specified on the DATA tag, the width of the value must fit between the starting position determined from the second data column (DATACOL) tag and the ending position for the layout column.

For a data presentation area specified with LAYOUT=HORIZ, if there are already data items on the display line and if the length of the prompt plus the value does not fit on the current line and allow a five column separator between data items, the prompt and values are placed on the next display line. For horizontal layout, the prompt plus the value must fit on a single line of the panel.

USAGE=OUT | INOUT

Indicates the display use of the data item. This attribute is required for display panels and is optional for print panels.

USAGE=OUT defines an output data item. OUT indicates that the variable displayed is for output only and cannot be changed by the user.

USAGE=INOUT defines a data entry item. INOUT indicates that the variable is for data entry and can be changed by the user.

Optional Attributes

HELP=*help-module-name*

Identifies online information which explains the data item. This attribute is allowed only for display panels. The name of the help module can be a name imported from another panel group, but must follow the rules for names outlined earlier in this chapter. For more information on the rules for names, see “Name Syntax” on page 469.

A single help module name must be associated with every item in the area. If the HELP attribute is specified on the DATA tag, the help applies to all groups and items in the area, and the HELP attribute is not allowed on any data group (DATAGRP) and DATAI tags in the area.

If the HELP attribute is specified on the DATAGRP tag, the help applies to all data items in the group and the HELP attribute is not allowed on DATAI tags within the group.

If no HELP attribute is specified on the DATA tag or on DATAGRP tags containing a data item, the HELP attribute is required on the DATAI tag.

To provide the user with easy access to the online information for a LAYOUT=HORIZ data area, use the HELP attribute on the DATA tag instead of the HELP attribute on the individual DATAI tags in the horizontal area.

The HELP attribute applies to all DATAIX tags associated with this data item.

NAME=*data-item-name*

The name associated with the item. This name can be used with the Add Pop-Up Window (QUIADDPW) API to position a window near this data item. This attribute is allowed only for display panels.

For more information on the rules for naming, see “Name Syntax” on page 469.

PMTLOC=BEFORE | ABOVE

Governs the placement of the variable value in relationship to the prompt text for the data item.

BEFORE indicates that the prompt text is placed before (to the left of) the variable value. The variable value begins on the line the prompt text ended on.

ABOVE indicates that the prompt text is placed above the data item variable. The variable is indented two spaces from the beginning of the prompt text. ABOVE is not allowed when LAYOUT=HORIZ is specified on the DATA tag.

If the prompt text is placed above the data item variable as a result of ABOVE, the ALIGN attribute on this tag is not used for formatting.

All data item extenders placed on a new line as a result of NEWLINE=CALC or NEWLINE=YES on the DATAIX tag begin in the same column as the data item variable.

CHCLOC=AFTER | ABOVE

Governs the placement of the data choices text in relationship to the variable value for the data item.

AFTER indicates that the data choices text is placed after the variable value. This is either on the same line or on the line below the end of the variable value.

ABOVE indicates that the data choices text is placed above the data item variable. ABOVE is allowed only if PMTLOC=ABOVE is specified. This attribute is not allowed when LAYOUT=HORIZ is specified on the DATA tag.

This attribute governs placement of data choices text only for DATAC tags immediately following the DATAI tag. DATAC tags following a DATAIX tag are not affected by this attribute.

ALIGN=LEFT | RIGHT | START | END

Governs how the display value of the variable is positioned within the choice column defined by the second DATACOL tag.

ALIGN=LEFT positions the leftmost character of the display value with the left edge of the choice column.

ALIGN=RIGHT positions the rightmost character of the display value with the right edge of the choice column. ALIGN=RIGHT is not allowed when LAYOUT=HORIZ is specified on the DATA tag. If the width of the display value, set by the WIDTH attribute on the class definition (CLASS) tag, exceeds the width of the choice column, ALIGN=RIGHT will work the same way as ALIGN=LEFT.

START is a synonym for LEFT, and END is a synonym for RIGHT.

JUSTIFY=LEFT | RIGHT | START | END

Governs how the dialog variable is edited into the display value. The default for this attribute is the same value as was specified on the ALIGN attribute for this tag.

For JUSTIFY=LEFT, the dialog variable is left-justified into the display value and leading blanks are preserved.

For JUSTIFY=RIGHT, the dialog variable is right-justified into the display value and trailing blanks are stripped.

START is a synonym for LEFT, and END is a synonym for RIGHT.

REQUIRED=NO | YES

Indicates whether or not the item is required. This attribute is allowed only for display panels. NO indicates that the item is not required.

YES indicates that the item is required on the display and that the field is highlighted accordingly. YES is only valid if USAGE=INOUT is specified on this tag for the data item.

When REQUIRED=YES is coded, no explicit checks are made for user entry. However, REQUIRED=YES causes the UIM to do input editing and validity check processing, even if the user does not enter anything into the field. This allows you to use the validity checking (CHECK) tag to ensure that the user enters data into a required field.

CSRLOC=YES | NO

Indicates whether or not the cursor is allowed on the input field when the UIM does default cursor positioning. This attribute is allowed only for display panels.

YES does not guarantee that the cursor is placed on the input field; it only indicates to the UIM that the cursor may be placed on the input field when the UIM does default cursor positioning.

NO indicates that the cursor should not be placed on the field. CSRLOC=NO is intended for input fields such as the position-to field found in a data area above a list area. The cursor should not be placed on a position-to field when the panel is first displayed or after a position-to operation completes. When CSRLOC=NO is coded, it is still possible for the UIM to place the cursor on the field when default cursor positioning is performed. This may occur in the following situations:

- All input fields on the current page of the panel are marked with CRSLOC=NO.
- The input field is marked in error.
- The cursor positioning under an application program's control specifies that the cursor should be placed on the field.

The CSRLOC attribute applies to all DATAIX tags associated with this data item.

DISPLAY=YES | NO

Indicates whether or not the contents of the field are visible when the panel is displayed. This attribute is allowed only for display panels.

YES indicates that the field is visible.

NO indicates that the field is not visible. DISPLAY=NO is intended for input fields, such as a password field, which should not be visible.

AUTOENTR=NO | YES

Indicates whether or not the field is an automatic enter input field. This attribute is allowed only for display panels. An automatic enter input field returns from the device to the host when the user enters a character, including a blank, into the last position of the field. This has the same effect as the user pressing the Enter key.

NO indicates that the field is not an automatic enter field.

YES indicates that the field is an automatic enter field. If YES is specified, USAGE=INOUT must also be specified on this tag.

Although the UIM does not restrict its usage, AUTOENTR=YES is intended for input fields that are one character wide.

COND=*condition-name*

The item is in effect on a display only if the condition specified is true. All associated DATAC and DATAIX tags are displayed only if this data item appears on the display. The condition must be defined in the panel group with the condition definition (COND) tag.

PROMPT='*action-text***'**

The action occurring when F4=List is requested through the PROMPT dialog command. This attribute is allowed only for display panels. The PROMPT attribute is only allowed when USAGE=INOUT is specified for this data item. The valid forms of action-text are:

- *'CALL program-reference'* For a description of the interface between the UIM and the exit program for the cursor-sensitive prompt, see the **APIs** topic in the iSeries Information Center.
- *'RETURN positive-integer'*

For a description of these actions, see Appendix B, "UIM Dialog Commands," on page 641.

DSPVALUE=*dialog-variable-name*

The dialog variable containing the current data entered in this data item. This attribute is allowed only for display panels and when the PROMPT attribute is coded on this tag. This variable is updated regardless of whether or not VARUPD processing is performed. The variable pool is updated based on the VARUPD attribute used to define the function key. This updating is independent of the display value variable processing.

The dialog variable specified must be defined on the CLASS tag as a CHAR or IGC variable whose length is the same as the width of the dialog variable specified on the VAR attribute of this tag. No translation list processing or value checking is performed for the value before it is placed in this variable. Character set and code page conversion are performed for this variable if the class of the variable named on the VAR attribute of this tag specifies that character set and code page conversion should be performed.

Optional Text

data-item-text

The text describing the data item. The text may appear on more than one line and can only contain the reverse text (RT) tag.

If the item appears in a data item group, it is indented as specified by the GRPSEP attribute on the DATAGRP tag. If the text is too long to fit in the prompt column, it is formatted onto additional lines as necessary to fit within the item column and is indented two spaces.

If no text is specified, the data value appears without a prompt. Prompt text should be specified for all DATAI tags, except for the following special cases:

- When the data item is described completely by the instruction or group heading text appearing above the item.
- When the data item contains a value that is a logical continuation of the value for a data item appearing immediately before it in the panel. Conditioning may be used to present only as many lines on the panel as there are values available.

  The set of DATAI tags presenting this type of information should always be specified in a data group to ensure that the UIM keeps related values together on the display, and to present appropriate information in extended help for the panel. Prompt text should be specified for the first data item in the group, or heading text should be specified on the DATAGRP tag to describe the entire group. Help should be specified for the group or for the entire area.

  When data groups are not used, the UIM may present items in different layout columns or even on different scrollable pages. If the same help module is specified for multiple data items, extended help for the panel repeats the online information for each item on the panel.

DATAIX (Data Item Extender)

Syntax for Display Panels:

```
►►─:DATAIX─VAR─=─dialog-variable-name─USAGE─=──┬─OUT───┬─────────────────►
                                               └─INOUT─┘

►──┬─────────────────────────┬─┬──────────────────────────────────────┬──►
   │              ┌─CALC─┐   │ │             ┌─2────────────────────┐ │
   └─NEWLINE─=────┼─NO───┼───┘ └─ITEMSEP─=───┴─item-separator-value─┴─┘
                  └─YES──┘

►──┬─────────────────────────┬─┬──────────────────────────┬──────────────►
   │           ┌─LEFT──┐     │ └─JUSTIFY─=────┬─LEFT──┬───┘
   └─ALIGN─=───┼─RIGHT─┼─────┘                ├─RIGHT─┤
               ├─START─┤                      ├─START─┤
               └─END───┘                      └─END───┘

►──┬──────────────────────────┬─┬──────────────────────────┬─────────────►
   │               ┌─NO──┐    │ │              ┌─YES─┐     │
   └─REQUIRED─=────┴─YES─┴────┘ └─DISPLAY─=────┴─NO──┴─────┘

►──┬──────────────────────────────┬─┬───────────────────────────────────┬─►
   └─PROMPT─=─' ─action-text─' ───┘ └─DSPVALUE─=─dialog-variable-name───┘

►──┬─────────────────────────────┬──────────────────────────────────────►◄
   │               ┌─NO──┐       │
   └─AUTOENTR─=────┴─YES─┴───.───┘
```

Syntax for Print Panels:

```
►►─:DATAIX─VAR─=─dialog-variable-name──┬─────────────────┬───────────────►
                                       └─USAGE─=─OUT─────┘

►──┬─────────────────────────┬─┬──────────────────────────────────────┬──►
   │              ┌─CALC─┐   │ │             ┌─2────────────────────┐ │
   └─NEWLINE─=────┼─NO───┼───┘ └─ITEMSEP─=───┴─item-separator-value─┴─┘
                  └─YES──┘

►──┬─────────────────────────┬─┬──────────────────────────────┬─────────►◄
   │           ┌─LEFT──┐     │ └─JUSTIFY─=────┬─LEFT──┬───.───┘
   └─ALIGN─=───┼─RIGHT─┼─────┘                ├─RIGHT─┤
               ├─START─┤                      ├─START─┤
               └─END───┘                      └─END───┘
```

The data item extender (DATAIX) tag provides an additional dialog variable to be associated with a data item (DATAI) tag. This tag is allowed for display panels and print panels.

The DATAIX tags must appear after the corresponding DATAI tag. More than one DATAIX tag may be specified after the DATAI tag to provide several dialog variables. If no DATAIX tags are entered, only one dialog variable, specified on the DATAI tag, appears for the data item.

The DATAIX and data choices (DATAC) tags can be specified in any order. The order in which they are specified determines where they appear in relationship to the dialog variable of the DATAI tag.

Required Attributes

VAR=*dialog-variable-name*

The name of the additional dialog variable displayed with the previous DATAI tag. The current value of the dialog variable is displayed or printed.

For a LAYOUT=1 data presentation area, the display value can exceed the width of the panel. In this case, the value is presented as a single field that wraps onto as many lines as necessary. For a LAYOUT=2 data presentation area, the display value must fit on one line.

For a LAYOUT=HORIZ data presentation area, the data item prompt, value, and all DATAIX values must all fit on a single line of the panel. If there are already data items on one line of the panel and if the prompt text and dialog variable for the DATAI tag plus the value of this dialog variable do not fit on the current line and still allow a five-column separator between data items, the entire data item, including this item extender, are placed on the next line of the panel.

USAGE=OUT | INOUT

The display use of the data item. This attribute is required for display panels but is optional for print panels.

USAGE=OUT defines an output data item. OUT indicates that the variable displayed is for output only and cannot be changed by the user.

USAGE=INOUT defines a data entry item. INOUT indicates that the variable is for data entry and can be changed by the user.

Optional Attributes

NEWLINE=CALC | NO | YES

Governs the placement of the variable for the data item extender in relationship to the variable value for the data item, possible choices text, and previously-defined data item extenders.

NEWLINE=CALC indicates that the UIM determines the placement of the variable. The variable is placed on the same line as the previous dialog variable if all of the following conditions are true:

- The variable fits within the value column, defined by the second data column (DATACOL) tag of the current line.
- The DATAIX tag is not immediately followed by a DATAC tag.
- The current line is not wrapped from the previous line

If any of the above conditions are not true, the data item extender is placed on the next display or print line. It is positioned according to the ALIGN attribute of this tag.

When NEWLINE=CALC is specified in a data presentation area with LAYOUT=HORIZ specified, the value is placed on the same line as the previous value.

NEWLINE=NO indicates that the value should be placed on the same line and to the right of the previous value. The previous value is the dialog variable for either the DATAI tag or a previous DATAIX tag. If the DATAIX tag appears immediately after a DATAC tag, NEWLINE=NO is not allowed.

NEWLINE=YES indicates that the value should be placed on the next display or print line. NEWLINE=YES is not allowed for data presentation areas with LAYOUT=HORIZ specified.

ITEMSEP=2 | *item-separator-value*

Indicates how many spaces separate the dialog variable of this data item extender and the previous dialog variable.

For data presentation areas with LAYOUT=1 or LAYOUT=2, the maximum value for this attribute is the width of the choice column, as defined by the second DATACOL tag. For data presentation areas with LAYOUT=HORIZ specified, the maximum value for this attribute is 5. The default value is 2, and the minimum value is 1.

The ITEMSEP attribute is not allowed when NEWLINE=YES is specified on this tag, and has no effect when NEWLINE=CALC is specified and the extender variable is placed on a new line.

ALIGN=LEFT | RIGHT | START | END

Governs how the display value is positioned within the choice column defined by the second DATACOL tag.

If the dialog variable is preceded by the dialog variable for the DATAI tag or by a previous DATAIX tag, the ALIGN attribute has no effect on the position of the display value within the choice column.

If the dialog variable is the first variable on the display or print line, the ALIGN attribute formats text as follows:

- ALIGN=LEFT positions the leftmost character of the display value with the left edge of the choice column.
- ALIGN=RIGHT positions the rightmost character of the display value with the right edge of the choice column. If the width of the display value exceeds the width of the choice column, ALIGN=RIGHT works the same as ALIGN=LEFT.

ALIGN=RIGHT is not allowed for data presentation areas with LAYOUT=HORIZ specified.

START is a synonym for LEFT and END is a synonym for RIGHT.

JUSTIFY=LEFT | RIGHT | START | END

Governs how the dialog variable is edited into the display value. The default for this attribute is the same value as was specified or defaulted to on the ALIGN attribute of this tag.

For JUSTIFY=LEFT, the dialog variable is left-justified into the display value. Leading blanks are preserved.

For JUSTIFY=RIGHT, the dialog variable is right-justified into the display value and trailing blanks are stripped.

START is a synonym for LEFT and END is a synonym for RIGHT.

The JUSTIFY attribute is ignored for variables defined as UCS-2.

REQUIRED=NO | YES

NO indicates that the item is not required. This attribute is allowed only for display panels.

YES indicates that the item is required on the display and that the field is highlighted accordingly. YES is only valid if USAGE=INOUT is specified for the data item extender.

When REQUIRED=YES is coded, no explicit checks are made for user entry. REQUIRED=YES causes the UIM to perform input editing and validity check processing, even if the user does not enter anything into the field. This allows you to use the validity checking (CHECK) tag to ensure that the user enters data into a required field.

DISPLAY=YES | NO

Indicates whether or not the field is visible when the panel is displayed. This attribute is allowed only for display panels. YES indicates that the field is visible.

NO indicates that the field is not visible. DISPLAY=NO is intended for input fields such as a password field, which should not be visible.

PROMPT='*action-text*'

The action occurring when F4=List is requested through the PROMPT dialog command. This attribute is allowed only for display panels, and when USAGE=INOUT is specified on this tag for this data item extender. The valid forms of action text are:

- *'CALL program-reference'* For a description of the interface between the UIM and the exit program for the cursor-sensitive prompt, see the **APIs** topic in the iSeries Information Center.
- *'RETURN positive-integer'*

For a description of these actions, see Appendix B, “UIM Dialog Commands,” on page 641.

DSPVALUE=*dialog-variable-name*

The dialog variable containing the current data entered into this data item extender. This attribute is allowed only for display panels and when the PROMPT attribute is coded on this tag.

This variable is updated regardless of whether or not VARUPD processing is performed. The variable pool is updated based on the VARUPD attribute used to define the function key. This updating is independent of the display value variable processing.

The dialog variable specified must be defined on the class definition (CLASS) tag as a CHAR or IGC variable whose length is the same as the width of the dialog variable specified on the VAR attribute of this tag.

No translation list processing or value checking is performed for the value before it is placed in this variable. Character set and code page conversion are performed for this variable if the class of the variable named on the VAR attribute of this tag specifies that character set and code page conversion should be performed.

AUTOENTR=NO | YES

Indicates whether or not the field is an automatic enter input field. This attribute is allowed only for display panels. An automatic enter input field returns from the device to the host when the user enters a character, including a blank, into the last position of the field. This has the same effect as the user pressing the Enter key.

NO indicates that the field is not an automatic enter field. This is the default value.

YES indicates that the field is an automatic enter field. If YES is specified, USAGE=INOUT must also be specified on this tag. Although the UIM does not restrict its usage, AUTOENTR=YES is intended for input fields that are one character wide.

DATASLT (Data Selection Field)

```
►►──:DATASLT──TYPE──=──┬─SINGLE─┬──┬──────────────────────────────┬──┬───────────────────────────┬──────────►
                       └─MULTI──┘  └─VAR──=──dialog-variable-name─┘  └─HELP──=──help-module-name─┘

►──┬────────────────────────────────┬──┬──────────────────────────┬──┬───────────────────────┬──────────────►
   └─NAME──=──selection-field-name──┘  └─PMTLOC──=──┬─BEFORE─┬──┘  └─REQUIRED──=──┬─NO──┬──┘
                                                    └─ABOVE──┘                    └─YES─┘

►──┬───────────────────────┬──┬───────────────────────────┬─────────────────────────────────────────────────►
   └─AUTOENTER──=──┬─NO──┬─┘  └─COND──=──condition-name───┘
                   └─YES─┘

►──.──┬──────────────────────────────┬──:EDATASLT.───────────────────────────────────────────────────────►◄
      └─selection-field-prompt-text──┘
```

The data selection field (DATASLT) tag defines a selection field in the data presentation area. This tag is allowed only for display panels.

The DATASLT may be a single- or multiple-choice selection field. The selection field is fixed in content and number of choices. A data selection field can only be specified when LAYOUT=1 or 2 on the data presentation area (DATA) tag.

The data selection field choice (DATASLTC) tag specifies the choices for the selection field. The DATASLTC tag is part of the DATASLT tag with respect to formatting, conditioning, scrolling, and help.

The prompt, field, and choices must fit within the data area. The UIM does not allow a selection field to be split while scrolling.

Other tags can be nested within the DATASLT tag. These tags are listed in the following table. The table defines the order in which the tags must appear, indicates which tags can be used in display panels only, print panels only, or both (specified by a D, P, or B, respectively), and specifies on which page more information can be found for each tag.

When more than one tag is listed with the same order number, all tags of that number can be mixed in any order. However, a tag with a higher order number cannot precede a tag with a lower order number. For example, a tag with an order number of three cannot precede a tag with an order number of one or two.

Table 60. Tag Allowed Between the DATASLT and EDATASLT Tags

Tag Name	Order	Use	Page
DATASLTC (Data selection field choice)	1	D	523

Required Attributes

TYPE=SINGLE | MULTI
Specifies whether or not the selection field is a selection field for single or multiple choices.

SINGLE indicates that the selection field is for a single choice.

MULTI indicates that the selection field is for multiple choices.

Optional Attributes

VAR=*dialog-variable-name*
The name of the dialog variable used to construct the single-choice selection field. This attribute is required for TYPE=SINGLE. The variable is presented on the display if TYPE=SINGLE, and must be declared with a BASETYPE of BIN(31) on the class definition (CLASS) tag. This variable contains the option number of the choice selected by the user. If no choices are selected, the variable returns a zero. The current value of the dialog variable is presented on the display.

If TYPE=MULTI is specified on this tag, this attribute is not allowed.

HELP=*help-module-name*
Identifies online information explaining the purpose of the data selection field. The name of the help module may be a name imported from another panel group, but the name must follow the rules for names outlined earlier in this chapter. For more information on the rules for names, see “Name Syntax” on page 469.

A single help module name must be associated with every item in the area. If the HELP attribute is specified on the DATA tag, the help applies to all groups and items in the area, and the HELP attribute is not allowed on any data group (DATAGRP) and DATASLT tags in the area.

If the HELP attribute is specified on the DATAGRP tag, the help applies to all data selection fields in the group, and the HELP attribute is not allowed on DATASLT tags within the group.

If no HELP attribute is specified on the DATA tag nor on DATAGRP tags containing a data selection field, the HELP attribute is required on the DATASLT tag.

NAME=*selection-field-name*
The name associated with the selection field. This name can be used later with the Add Pop-Up Window (QUIADDPW) API to position a window associated with this selection field. For more information on the rules for naming, see “Name Syntax” on page 469.

PMTLOC=BEFORE | ABOVE
Governs the placement of the prompt text in relationship to the input field associated with the selection field.

BEFORE indicates that the prompt text is placed before (to the left of) the variable value of the data field. The variable value begins on the same line on which the prompt text ended. BEFORE is the default.

ABOVE indicates that the prompt text is placed above the data item variable. The variable is indented two spaces from the beginning of the prompt text.

REQUIRED=NO | YES
This attribute is allowed only for TYPE=SINGLE. NO indicates that the field is not required. NO is the default value.

YES indicates that the field is required on the display, and the field is highlighted accordingly. When REQUIRED=YES is coded, no explicit checks are made for user entry. However, YES causes the UIM to perform input editing and validity check processing even if the user does not enter anything into the field. This allows you to use the validity checking (CHECK) tag to ensure that the user enters data into a required field. REQUIRED=YES cannot be specified when TYPE=MULTI is specified on this tag.

AUTOENTR=NO | YES
Indicates whether or not the field is an automatic enter input field. An automatic enter input field returns the screen from the device to the host when the user enters a character, including a blank, into the last position of the field. This has the same effect as the user pressing the Enter key.

NO indicates that the field is not an automatic enter field. This is the default value.

YES indicates that the field is an automatic enter field. Although the UIM does not restrict its usage, AUTOENTR=YES is intended to be used only on input fields that are one character wide. YES is not allowed when TYPE=MULTI is specified on this tag.

COND=*condition-name*
The selection field is displayed only if the condition specified is true. Associated DATASLTC tags are displayed only if this selection field appears on the display. The condition must be defined in the panel group with the condition definition (COND) tag.

Optional Text

selection-field-prompt-text
The text which describes the selection field. The text may appear on more than one line and can contain only the reverse text (RT) tag. The length of the text is 255 bytes or less.

If the selection field appears in a data item group, it is indented as specified by the GRPSEP attribute on the DATAGRP tag. If the text is too long to fit in the prompt column, it is formatted on the following lines as necessary to fit within the column and indented two spaces from the beginning of the first line.

If no text is specified, the selection field appears without a prompt. Prompt text should be specified for all DATASLT tags, except for when the selection field is described completely by the instruction or group heading text appearing above the item.

Example 1: Data Entry Panel

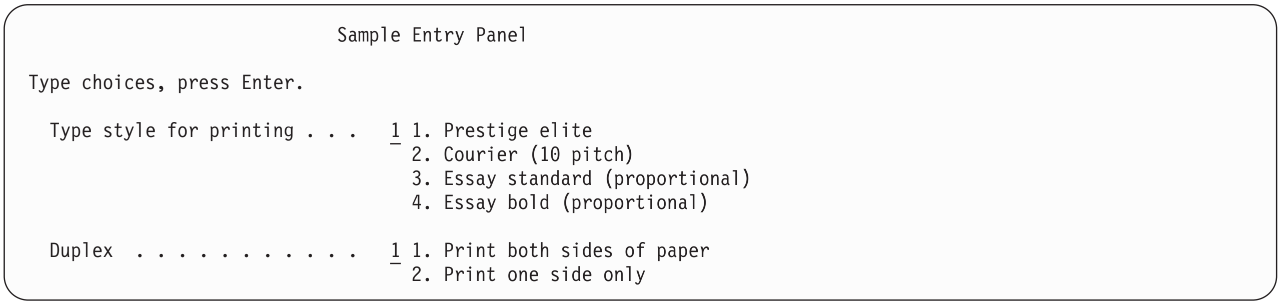
The example shows a sample data entry panel. The panel has 2 selection fields with the defaults supplied by the calling program by previously setting the dialog variables for the function.

UIM Source

```
:class name=select
      basetype='bin 31'.
:var name=prtstyle
      class=select.
```

```
:var name=duplex
      class=select.
   ⋮
:panel name=entry1
      help=hentry1
      topsep=space
      ...
      .Sample Entry Panel
:data depth='*'
      bodysep=both.
:topinst.Type choices, press Enter.
:datacol width=30.
:datacol width='*'.
:dataslt help=hstyle
       var=prtstyle
       type=single
       .Type style for printing
:datasltc help=hstylep
       option=1
       .Prestige elite (12 pitch)
:datasltc help=hstylec
       option=2
       .Courier (10 pitch)
:datasltc help=hstylees
       option=3
       .Essay standard (proportional)
:datasltc help=hstyleeb
       option=4
       .Essay bold (proportional)
:edataslt.
:dataslt type=single
       help=hduplx
       var=duplex
       .Duplex
:datasltc help=hduplxy
       option=1
       .Print both sides of paper
:datasltc help=hduplxn
       option=2
       .Print one side only
:edataslt.
:edata.
:epanel.
:invellip.
```

Results



```
                            Sample Entry Panel

 Type choices, press Enter.

   Type style for printing . . .   1 1. Prestige elite
                                     2. Courier (10 pitch)
                                     3. Essay standard (proportional)
                                     4. Essay bold (proportional)

   Duplex  . . . . . . . . . . .   1 1. Print both sides of paper
                                     2. Print one side only
```

In this example, the *PRTSTYLE* and *DUPLEX* dialog variables must be BIN 31, and the option value selected is returned to the calling program in these variables.

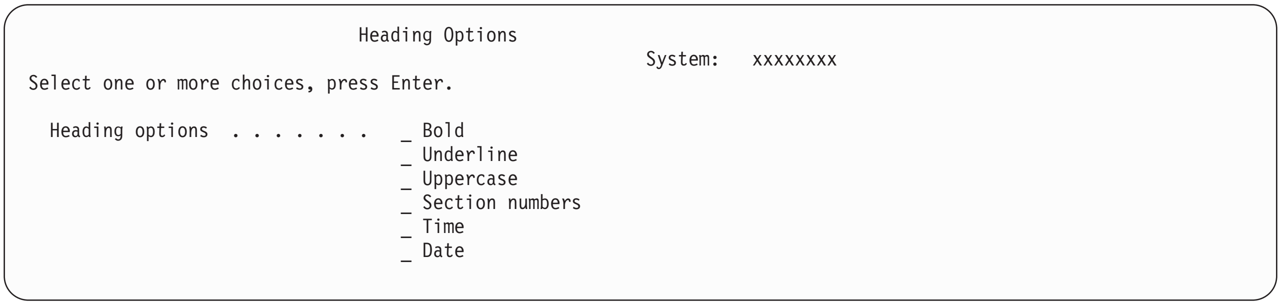
Example 2: Multiple-Selection Field

This example shows a multiple-selection field. The user selects the values by typing a slash (/) or the country-designated character into the entry field preceding the desired options. More than one heading option may be selected.

UIM Source

```
:class name=sfield
       basetype='char 1'.
:var name=hdoptsb
       class=sfield.
:var name=hdoptsu
       class=sfield.
:var name=hdoptsuc
       class=sfield.
:var name=hdoptssn
       class=sfield.
:var name=hdoptstm
       class=sfield.
:var name=hdoptsdt
       class=sfield.
:panel name=xxx
       keyl=x1.Heading Options
:data depth='*'
       bodysep=both.
:topinst.Select one or more
choices, press Enter.
:datacol width=30.
:datacol width='*'.
:dataslt type=multi
       help=haward.Heading options
:datasltc var=hdoptsb
       help=hbold.Bold
:datasltc var=hdoptsu
       help=hundl.Underline
:datasltc var=hdoptsuc
       help=hupperc.Uppercase
:datasltc var=hdoptssn
       help=hsecn.Section numbers
:datasltc var=heoptstm
       help=htime.Time
:datasltc var=hdoptsdt
       help=hdate.Date
:edataslt.
:edata.
```

Results



```
                              Heading Options
                                                        System:   xxxxxxxx
Select one or more choices, press Enter.

  Heading options  . . . . . . .   _ Bold
                                   _ Underline
                                   _ Uppercase
                                   _ Section numbers
                                   _ Time
                                   _ Date
```

If the user chooses *Underline*, *Section numbers*, and *Date* in this example, the value in the dialog variables are:

- hdoptsb=0
- hdoptsu=1

- hdoptsuc=0
- hdoptssn=1
- hdoptstm=0
- hdoptsdt=1

DATASLTC (Data Selection Field Choice)

```
►►─:DATASLTC──┬──────────────────────────────────┬──┬───────────────────────────┬──────────────►
              └─CHOICE──=──dialog-variable-name──┘  └─OPTION──=──option-number──┘

►─┬───────────────────────────────┬──┬────────────────────────────┬──┬──────────────────────────┬──►
  └─VAR──=──dialog-variable-name──┘  └─HELP──=──help-module-name──┘  └─COND──=──condition-name──┘

►─┬───────────────────────────┬──┬───────────────────────────────────┬──────────────────────────►
  └─AVAIL──=──condition-name──┘  └─AVLMSGID──=──message-identifier──┘

►─┬───────────────────────────────────────────────┬─.─┬──────────────────┬────────────────────►◄
  └─AVLMSGF──=──'──qualified-message-file-name──'─┘   └─text-for-choices─┘
```

The data selection field choice (DATASLTC) tag defines a possible choice for a single- or multiple-choice selection field. This tag is allowed only for display panels. It is not valid if LAYOUT=HORIZ is specified on the data presentation area (DATA) tag for the area.

The DATASLTC tags must appear after the corresponding data selection field (DATASLT) tag and at least one DATASLTC tag must be specified between the DATASLT and the EDATASLTC tags. The DATASLTC tag defines the choices for the selection field.

Optional Attributes

CHOICE=*dialog-variable-name*

The name of a dialog variable containing the possible choices text to be displayed. The dialog variable must be defined so that the text fits on a single line.

Dialog variables must be defined with a BASETYPE of CHAR, IGC, or BIN on the class definition (CLASS) tag.

The error state of the dialog variable is not used for determining the highlighting of the text.

If the CHOICE attribute is specified, the *text-for-choices* cannot be specified.

Special formatting for IGC. (The abbreviation IGC is used in commands and keywords to represent double-byte character set functions.) When a dialog variable with a BASETYPE of IGC is specified on the CLASS tag, the UIM does special formatting. If the variable value begins with a shift-out character (X'0E'), the UIM shifts the value one character to the left to preserve vertical alignment with data choices on other lines.

OPTION=*option-number*

The number assigned to this option. Option numbers are integers in the range of 1 to 99. This attribute is required if specified within a DATASLT tag that has TYPE=SINGLE specified, but cannot be used if TYPE=MULTI is specified.

Selection field choices are displayed in the order defined in the selection field. If the numbering of two choices is not consecutive, a blank line is automatically placed between the two choices. If two choices have the same option number and both are conditioned-on at the same time, the choice defined first is displayed.

VAR=*dialog-variable-name*
The name of the dialog variable used to indicate whether or not the selection field choice is selected. This attribute is required for TYPE=MULTI. The variable must be declared with a BASETYPE of CHAR 1 on the CLASS tag.

If the choice is not selected, the value of the dialog variable is ′0′. If the choice is selected, a value of ′1′ is returned in the dialog variable. If the current value of the variable is ′1′ when the screen is displayed, a slash (/) is displayed on the screen. The user may enter either a slash or the country-designated character and a ′1′ is returned to the user. Whichever character is used for selection by the user is shown when the panel is redisplayed. If the current value of the dialog variable is something other than ′1′ when the panel is displayed, the variable is presented as unselected.

The VAR attribute is required if TYPE=MULTI on the DATASLT tag. If TYPE=SINGLE is specified, the VAR attribute is not allowed.

HELP=*help-module-name*
Identifies online information explaining the purpose of the selection field choice. The name of the help module may be a name imported from another panel group, but must follow the rules for names outlined earlier in this chapter. For more information on the rules for names, see “Name Syntax” on page 469.

This attribute is allowed only when the HELP attribute is specified on the DATASLT tag that contains this DATASLTC tag.

If the HELP attribute is specified on a DATASLTC tag within a selection field, all DATASLTC tags within that selection field must have the HELP attribute specified.

For multiple-choice selection fields with SELECT=MULTI specified on the DATASLT tag, the online information identified by this attribute is included as part of the contextual help displayed when the cursor is positioned anywhere within the selection field.

For single-choice selection fields with SELECT=SINGLE specified on the DATASLT tag, the online information identified by this attribute is displayed when help is requested while the cursor is positioned on the choice text for this tag. This online information is also included as part of the contextual help displayed when the cursor is positioned within the selection field but not on the text for one of the choices within the field. This includes when the cursor is positioned on the prompt text for the selection field or in the entry field for the selection field. If the cursor is in the entry field and a valid choice is entered, when help is requested, the help for that choice is displayed.

COND=*condition-name*
The selection field choice is in effect on the panel only if the condition specified is true. The condition must be defined in the panel group prolog with the condition definition (COND) tag. When the choice is conditioned-off, the selection field choice does not appear in the selection field and the help for the choice is not included in requests to display help.

AVAIL=*condition-name*
The name of a condition indicating whether or not the selection field choice is available. The condition must be defined in the panel group prolog with the COND tag.

When the condition is true, the selection field choice is available. When the condition is false, the selection field choice is not available. Any condition specified on the COND attribute on this tag takes precedence over this attribute.

Unavailable choices are displayed with a color change and an asterisk (*) overlaying the first part of the choice option number.

The AVAIL attribute cannot be specified if the selection field is specified as TYPE=MULTI on the DATASLT tag.

AVLMSGID=*message-identifier*
The message identifier of the message displayed when the selection field choice is selected when it is not available as specified by the `AVAIL` attribute on this tag. This attribute is allowed only when the `AVAIL` attribute is specified.

If this attribute is not specified, the UIM displays a default message stating that the choice is not currently available.

AVLMSGF=*'qualified-message-file-name'*
The message file name containing the message identifier. This attribute is allowed when the `AVLMSGID` attribute on this tag is specified. If the `DFTMSGF` attribute is not specified on the panel group (PNLGRP) tag and the `AVLMSGID` attribute on this tag is specified, this attribute must be specified.

Optional Text

text-for-choices
This text is an implied paragraph. When the display is formatted, any text that does not fit onto one line is formatted on the following lines and indented two columns. The text can be a maximum of 255 characters and can only contain the reverse text (RT) tag. *Text-for-choices* is required unless the `CHOICE` attribute is specified on this tag.

DL (Definition List)

```
►►──:DL──┬─────────┬──.──┬──────────────────────────────────┬──┬─────────────────────────────┬──────►
         └─COMPACT─┘     └─:DTHD──.──definition-term-header─┘  └─:DDHD──.──definition-header─┘

   ┌────────────────────────────────────────────────────┐
   ▼                                                    │
►──:DT──.──definition-term──:DD──.──definition──:EDL.───┴─────────────────────────────────────────►◄
```

The definition list (DL) is a list of words or phrases and their corresponding definitions, descriptions, or explanations. This tag requires a matching end tag. A definition list is only allowed in information areas and help areas.

The terms being defined and their definitions are identified by the definition term (DT) tag and the definition description (DD) tag. A heading for the column of terms and the column of definitions can be identified by the definition term heading (DTHD) tag and the definition description heading (DDHD) tags, respectively.

Definition lists can occur anywhere in text; they can be nested within other lists or definition lists, and other lists can be nested within definition lists.

Two DT or DD tags cannot be used consecutively.

Other tags can be nested within the DL tag. These tags are listed in the following table. The table defines the order in which the tags must appear, indicates which tags can be used in display panels only, print panels only, or both (specified by a D, P, or B, respectively), and specifies on which page more information can be found for each tag.

When more than one tag is listed with the same order number, all tags of that number can be mixed in any order. However, a tag with a higher order number cannot precede a tag with a lower order number. For example, a tag with an order number of three cannot precede a tag with an order number of one or two.

Table 61. Tags Allowed Between the DL and EDL Tags

Tag Name	Order	Use	Page
FIG (Figure)	1	B	527
LINES (Unformatted lines)	1	B	546
XMP (Example)	1	B	639
NT (Note)	1	B	590
P (Paragraph)	1	B	593
PC (Paragraph continuation)	1	B	603
LP (List part)	1	B	579
OL (Ordered list)	1	B	591
SL (Simple list)	1	B	622
UL (Unordered list)	1	B	633
PARML (Parameter list)	1	B	602
DL (Definition list)	1	B	525

Optional Attribute

COMPACT
Formats the list without the blank line between the items.

Required Tags

DT.*definition-term*
The text for the definition term. The term formats the way it is entered, in highlight phrase 2 (HP2).

The term is assumed to be 10 single-byte character set (SBCS) characters or 4 double-byte character set (DBCS) characters, and starts at position 2 relative to the left margin. If the term is longer than the assumed number of characters, it is extended into the description line.

DD.*definition*
The text for the definition. The description of the term is an implied paragraph and can contain text items. Additional paragraphs can be inserted following the description paragraph, using the paragraph (P) tag.

The description starts at position 14 relative to the left margin, unless the term extends into the description area.

Optional Tags

DDHD.*definition-header*
The text for the definition header. The header formats the way it is entered, in highlight phrase 2 (HP2).

DTHD.*definition-term-header*
The text for the definition term header. The header formats the way it is entered, in highlight phrase 2 (HP2).

Example 1: Definition List

This example illustrates how a definition list with headings will format.

UIM Source

```
:DL.
:DTHD.Term
:DDHD.Description
:DT.DL
:DD.This is a sample
definition list term
and description.
:DT.EDL
:DD.Here is another.
:EDL.
```

Results

Term	**Description**
DL	`This is a sample definition list term and description.`
EDL	`Here is another.`

Example 2: Compact Definition List

This is a definition list using the `COMPACT` attribute.

UIM Source

```
:DL compact.
:DT.COMPACT
:DD.This causes the list
to be compacted, so the
blank lines are removed.
:DT.ANOTHER
:DD.Here is another.
:EDL.
```

Results

COMPACT	`This causes the list to be compacted, so the blank lines are removed.`
ANOTHER	`Here is another.`

Care should be taken when using unformatted lines (LINES), figure (FIG), and example (XMP) tags within a definition list, because text that does not fit on one line wraps to column one of the next line. Lines and figures start at the current left margin and examples are indented four spaces from the current left margin. The current left margin changes when nested lists are formatted. If there is an information source containing the LINES, FIG, or XMP tag imbedded at various locations, including within lists, it may not look the same each time.

FIG (Figure)

```
►►──:FIG──┬──────────────────────┬──.──figure-content──┬───────────────────────────────────┬──:EFIG.──────►◄
          │            ┌─RULE─┐  │                      └─:FIGCAP──.──figure-caption-text─┘
          └─FRAME──=───┴─NONE─┴──┘
```

The figure (FIG) tag identifies a diagram, chart, or other illustration. This tag requires a matching end tag. The figure tag is only allowed in information areas and help areas. A figure can also contain a figure caption, which is identified by the figure caption (FIGCAP) tag.

Figures turn off automatic formatting, so the text is formatted the same way it is entered. The figure formats where it is entered, across the full width of the screen or window.

Care should be taken when using figures within lists, because figure text that does not fit on one line wraps to column one of the next line.

Figures are always formatted with the left margin set to column 2. The formatting width is two columns less than the width specified on the help module. When the FIG tag appears in an information area, the formatting width initially is two bytes less than the width specified on the panel tag.

The figure caption begins at the left margin. There are two spaces between the figure tag, ″Figure:″, and the figure caption text if either the figure tag or the figure caption text are single-byte characters. There are four spaces between the figure tag and the figure caption if both are double-byte characters. If the figure caption does not fit on one line, it wraps onto the second line starting at position 4 relative to the left margin.

Optional Attribute

FRAME=RULE | NONE

The type of frame to put around the figure. RULE is the default. A row of hyphens is displayed above and below the figure.

NONE indicates that no frame is used; a blank line is placed ahead of and behind the figure.

Optional Tag

FIGCAP.*figure caption text*

A caption for the figure, if needed. The caption may appear on more than one line in the source. The FIGCAP tag must appear immediately before the EFIG tag.

Example: Sample Figure

This example shows how text formats the way it is entered with a FIG tag, and how a FIG tag can specify a frame to set the figure off from the rest of the text.

UIM Source

```
:FIG frame=rule.
some
   sample
      text
:FIGCAP.A Sample Figure
:EFIG.
```

Results

```
some
   sample
      text
Figure:  A Sample Figure
```

HELP (Help Module)

```
►►──:HELP──NAME──=──help-module-name──┬────────────────────────────────────┬──────────►
                                      │              ┌─'*'──────────────┐  │
                                      └─WIDTH──=─────┴─help-window-width─┴──┘

►──┬──────────────────────────────────────────────────────────────────────────┬──────►◄
   │          ┌─'*'──────────────────┐                                         │
   └─DEPTH──=─┴─help-window-depth────┴──.──┬───────────────────┬──:EHELP.────┘
                                           └─help-title-text───┘
```

The help module (HELP) tag indicates the beginning of a help module. An EHELP tag must be specified at the end of the help module. A help module can be used for help displays and for index search topics. For more information on the index search function, see "ISCH (Index Search)" on page 538. Help modules may also be linked to each other. For more information on how help modules may be linked together, see "LINK (Hypertext Link Definition)" on page 547.

Any tag that has a `HELP` attribute may refer to this module.

Required Attribute

NAME=*help-module-name*

The name of this help module. For more information on the rules for naming, see "Name Syntax" on page 469.

This name can be referred to from:

- The `HELP` attribute on UIM panel definition tags.
- The `NAME` attribute on an import (IMPORT) tag.
- The `NAME` attribute on an imbed help (IMHELP) tag within another help module.
- The `TOPICS` attribute on an index search subtopic (ISCHSUBT) tag.
- The HLPPNLGRP keyword in data description specifications (DDS) specifications.
- A control language (CL) command definition.
- The help identifier array parameter on the Display Help (QUHDSPH) API.
- The DSPHELP dialog command.

The maximum length of the value for the `NAME` attribute is 32 characters. A slash (/) can be used in the name as a delimiter or separator character. The slash is used to separate a command name from a parameter name when naming a help module for commands. If the slash is used, the name must be enclosed in apostrophes ('). Help modules may not be nested.

Optional Attributes

WIDTH='*' | help-window-depth

The width of the help window when it is displayed. A value specified for the `WIDTH` attribute must include 2 bytes of space for the right and left margin. The minimum width is 32 bytes, and the maximum width is 74 bytes.

Help modules that are displayed as full screen or in an extended help window are indented by 4 bytes. Therefore, the formatting width is 6 bytes less than the width. For example, a help module formatted with the maximum width of 74 bytes has a formatting width of 68 bytes.

The UIM does not increase or decrease the width of the window if a value other than '*' is specified. The width should be specified only if the text between the HELP and EHELP tags depends on a window of a particular width.

If the help is for the function keys or a list area and if WIDTH='*', the UIM uses a width of 38 bytes. If the help is for a menu area or data area, the UIM uses a width of 60 bytes. The UIM finds a location for the window using this width, then expands the window to a size which displays the online information in the most appropriate fashion.

The width used is that of the first HELP tag found when the UIM assembles the online help information.

DEPTH='*' | help-window-depth

The depth of the help window when it is displayed. The value specified for the depth may not be less than nine lines, and must include the following:

- One line for a help title
- One line for a top separator line
- One line for a bottom separator
- Two lines for the function keys
- One line for the message line

For help displayed in a pop-up window, the maximum value for the DEPTH attribute is 21.

The UIM does not increase or decrease the depth of a window if a value other than '*' is specified. A specific depth should be specified only if the text between the HELP and the EHELP tags depends on a particular window width.

If the help is for a data area, menu area, or the function keys, and if DEPTH='*', the UIM chooses a depth of 9. If the help is for a list area, the UIM chooses a depth of 14. The UIM finds a location for the window using this depth, then expands the window to a size designed to display the online help information in the most appropriate fashion.

The depth used is that of the first HELP tag found when the UIM assembles the online help information.

Optional Text

help-title-text

The title used on the help panel. The text must appear on the same or next line as the tag and can only contain the reverse text (RT) tag. If the *help-title-text* is longer than the pop-up window width, the title text is truncated to the width of the pop-up window. If the value for the WIDTH attribute on this tag is greater than 57 or equal to '*', the maximum length of the *help-title-text* is 55 characters. Otherwise, the maximum length is the value specified for the WIDTH attribute minus two.

The help panel title displayed depends on whether contextual or extended help is requested and whether the help is displayed as a full screen or in a pop-up window.

Help information presented in pop-up windows is titled as follows:

- For contextual help, the title of the HELP tag for the associated item is used. If no title is defined on the HELP tag, a default title of "Help" is used.
- For extended help, the title from the first HELP tag for the panel is used. If no title is defined on the HELP tag, the default title or the full display title of the QUHDSPH API is used.

If the text for the help title is longer than the pop-up window width, the title text is truncated at a word boundary to the width of the pop-up window. The default title or the full display title of the QUHDSPH API is used.

Help information presented in full screen mode is titled as follows:

- For contextual help, the title from the HELP tag is used.
- For extended help, the title from the first HELP tag is used.
- If no title is defined on the HELP tag, the default title or the full display title of the QUHDSPH API is used.

Help modules must be coded so that each module can be individually formatted. Help modules must begin with one of the following tags:

DL	Definition list
FIG	Figure
H1-H4	Headings
IMHELP	Imbed help
ISCH	Index search
LINES	Unformatted lines
NT	Note
OL	Ordered list
P	Paragraph
PARML	Parameter list
SL	Simple list
UL	Unordered list
XH1-XH4	Extended help headings
XMP	Example

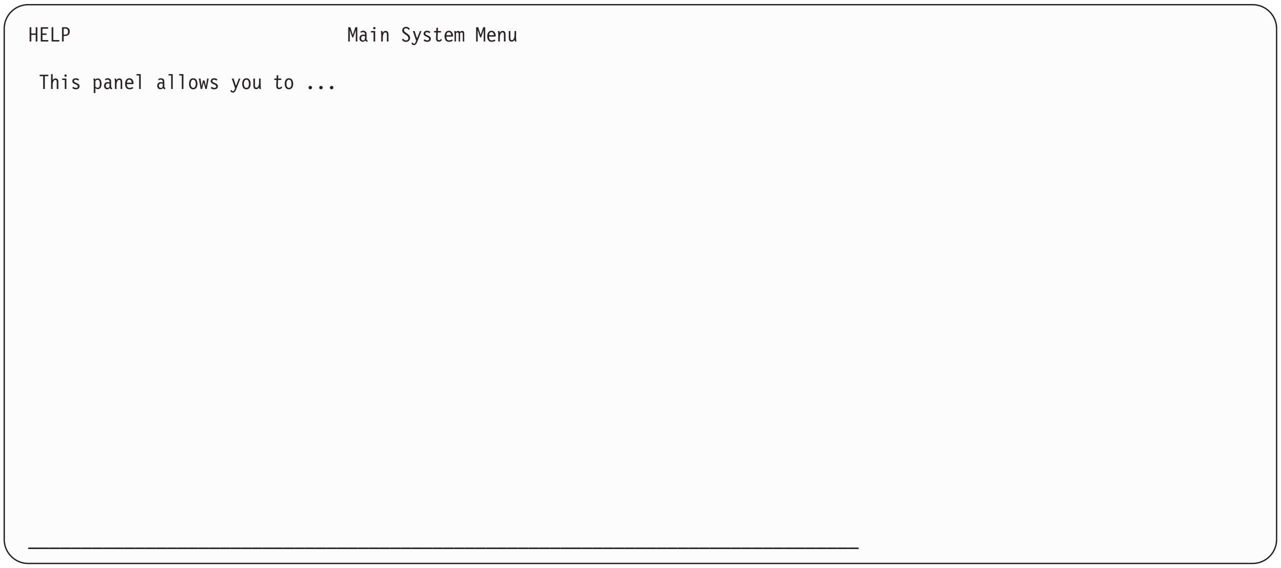
Example: Help Panel Definition

This example shows how a help panel is defined and how the text is displayed.

UIM Source

```
:help name=hmain.Main System Menu - Help
:p.
This panel allows you to ...
  (extended description of the panel)
:ehelp.
```

Results



```
HELP                        Main System Menu

 This panel allows you to ...
```

HP0 through HP9 (Highlighted Phrase)

```
►►──┬─:HP0.──┬──────┬─:EHP0.─┬─────────────────────────────────────────►◄
    │        └─text─┘        │
    ├─:HP1.──┬──────┬─:EHP1.─┤
    │        └─text─┘        │
    ├─:HP2.──┬──────┬─:EHP2.─┤
    │        └─text─┘        │
    ├─:HP3.──┬──────┬─:EHP3.─┤
    │        └─text─┘        │
    ├─:HP4.──┬──────┬─:EHP4.─┤
    │        └─text─┘        │
    ├─:HP5.──┬──────┬─:EHP5.─┤
    │        └─text─┘        │
    ├─:HP6.──┬──────┬─:EHP6.─┤
    │        └─text─┘        │
    ├─:HP7.──┬──────┬─:EHP7.─┤
    │        └─text─┘        │
    ├─:HP8.──┬──────┬─:EHP8.─┤
    │        └─text─┘        │
    └─:HP9.──┬──────┬─:EHP9.─┘
             └─text─┘
```

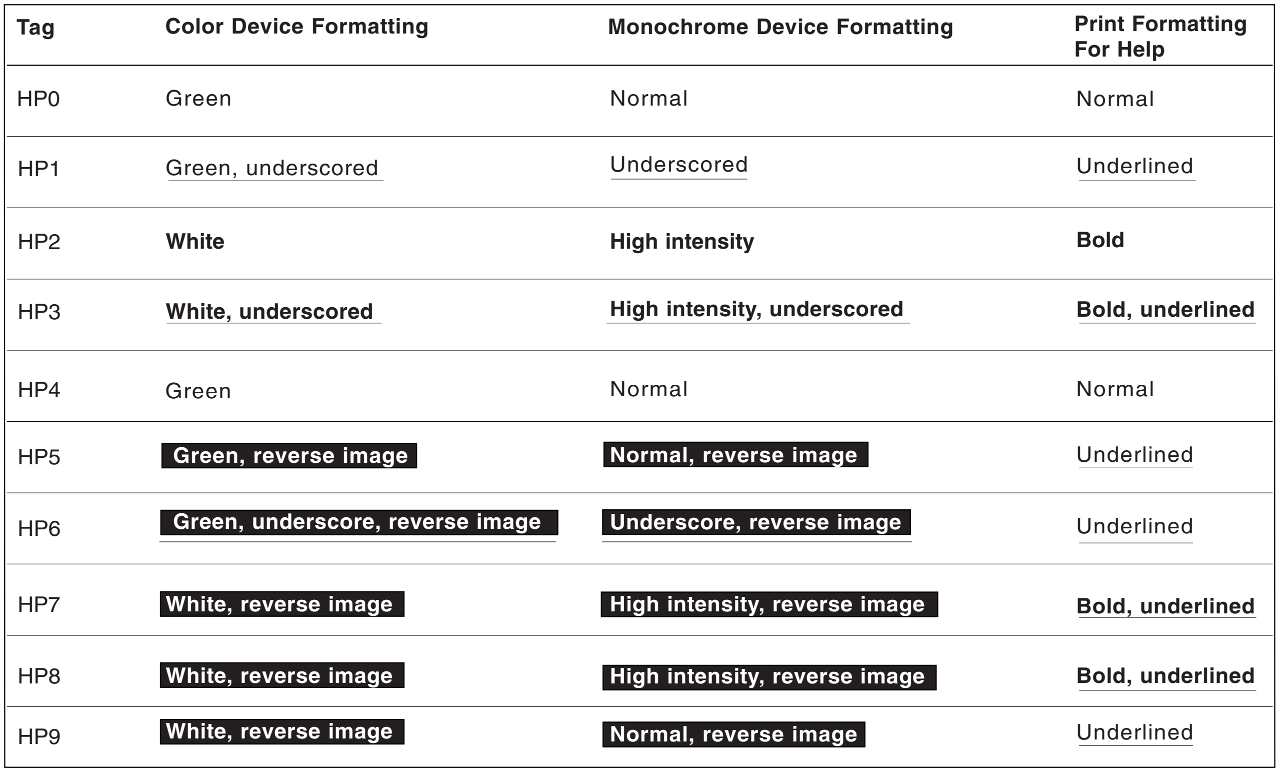
These highlighted phrase (HP0-HP9) tags identify a word or phrase which is highlighted. All HP*n* tags require matching end tags. These tags are only allowed in help areas and in information areas that are not in the print head (PRTHEAD) or print panel (PRTPNL) tags.

The HP*n* and EHP*n* tag phrase should be specified on word boundaries. If the two characters immediately following the EHP*n* tag are a punctuation mark and a blank, the UIM automatically extends the emphasis attribute to include the punctuation mark. This allows the punctuation mark and the text associated with it to be displayed using the same emphasis.

Highlighting tags may be nested. For example, `:HP1.text:HP2.text2:EHP2.text3:EHP1.` is valid.

Optional Text

text
: Although the word or phrase to be highlighted is not required, the tag has no meaning when no text is specified.



Tag	Color Device Formatting	Monochrome Device Formatting	Print Formatting For Help
HP0	Green	Normal	Normal
HP1	Green, underscored	Underscored	Underlined
HP2	**White**	**High intensity**	**Bold**
HP3	**White, underscored**	**High intensity, underscored**	**Bold, underlined**
HP4	Green	Normal	Normal
HP5	**Green, reverse image**	**Normal, reverse image**	Underlined
HP6	**Green, underscore, reverse image**	**Underscore, reverse image**	Underlined
HP7	**White, reverse image**	**High intensity, reverse image**	**Bold, underlined**
HP8	**White, reverse image**	**High intensity, reverse image**	**Bold, underlined**
HP9	**White, reverse image**	**Normal, reverse image**	Underlined

RV2W062-0

H1 through H4 (Heading)

```
►►──┬─:H1─┬──.──heading-text──────────────────────────────────────────────►◄
    ├─:H2─┤
    ├─:H3─┤
    └─:H4─┘
```

The heading (H1-H4) tags identify main topics and subtopics of information. These tags are only allowed in information areas and help areas.

In a help area, text after the headings is indented four spaces from the margin to separate the headings from the text. In an information area, text formats flush left, against the left margin.

Headings have one blank line formatted before and after the heading text.

Specific formatting rules are as follow:

H1 Centers and formats text as underscored and highlighted, like the HP3 tag. This tag causes a page eject when it appears in a printed help module.

H2 Left justifies and formats text as underscored and highlighted, like the HP3 tag.

H3 Left justifies and formats text as highlighted text, like the HP2 tag.

H4 Left justifies and formats text as underscored, like the HP1 tag.

Required Text

heading-text
The text for the heading. The text of a heading must be entered on a single line and is formatted as

entered. It cannot contain any other tags. A common practice is to enter the text according to the current publishing style, with significant words in initial caps.

Example: Heading Tags

This example illustrates how the different headings are justified and formatted.

UIM Source

```
:H1.A One Heading
:p.Here's a paragraph.
:H2.A Two Heading
:p.Another paragraph.
:H3.A Three Heading
:p.Still another paragraph.
:H4.A Four Heading
:p.Still another paragraph.
```

Results

A One Heading

Here’s a paragraph.

A Two Heading

Another paragraph.

A Three Heading

Still another paragraph.

A Four Heading

Still another paragraph.

IMHELP (Imbed Help)

```
►►──:IMHELP──NAME──=──help-module-name──.─────────────────────────────────────────►◄
```

The imbed help (IMHELP) tag imbeds a help module within a help module. A help module begins with a help (HELP) tag, and ends with an end help (EHELP) tag.

Required Attribute

NAME=*help-module-name*
The name of a help module imbedded where this tag occurs. For more information on the rules for naming, see “Name Syntax” on page 469. The name must be the name of another help module defined within the same panel group, or it must be imported from another panel group using the import (IMPORT) tag. IMHELP tags have a nesting limit of 16.

If the index search (ISCH) or index search synonym (ISCHSYN) tags are a part of the embedded help module, they are not part of the module where the IMHELP tag occurs.

An active highlight phrase (HP*n*) tag is not allowed when an IMHELP tag is coded. Any active HP*n* tags must be ended with EHP*n* tags before coding the IMHELP tag.

The imbed help (IMHELP) tag can be imbedded with the following tags:

- Definition list (DL). The IMHELP tag must be preceded by a DD (definition) tag, or the text of the DD tag.
- Imbedded help (IMHELP).
- Note (NT).
- Ordered list (OL). The IMHELP tag must be preceded by a LI (list item) tag, or the text of the LI tag.
- Parameter list (PARML). The IMHELP tag must be preceded by a PD (parameter definition) tag, or the text of the PD tag.
- Simple list (SL). The IMHELP tag must be preceded by a LI (list item) tag, or the text of the LI tag.
- Unordered list (UL). The IMHELP tag must be preceded by a LI (list item) tag, or the text of the LI tag.

Text cannot directly follow the IMHELP tag. Any tag allowed in the construct, except for the HP*n* (highlighted phrase), RT (reverse text), or PK (programming keyword) tags, can be used to satisfy this requirement.

Example: Imbedded Help

This example uses imported help for the help on function keys. A second help panel imports commonly used information within a list item.

UIM Source

```
:HELP name='menu1'.
:H2.Purpose of MENU1
Menu1 is intended for the use of...
   ⋮
:IMHELP name='keydefs'.
   ⋮
:EHELP.
   ⋮
:HELP name='keydefs'.
:PARML.
:PT.F1=Help
:PD.This key...
:PT.F3=Exit
:PD.This key...
:EPARML.
:EHELP.
:HELP name='lib1'.
:P.The valid values for library are:
:ol.
:li.APP1LIB
:li.APP2LIB
:li.Any IBM-supplied library.
:imhelp name='ibmlib'.
:p.
Some of the above libraries ...
:li.APP3LIB
   ⋮
:eol.
:ehelp.


:HELP name='ibmlib'.
:ol.
:li.QSYS
:li.QUSRSYS
   ⋮
:eol.
:ehelp.
```

IMPORT (Import)

```
►►──:IMPORT──NAME──=──┬─imported-name─┬──PNLGRP──=──panel-group-name──────────────────►
                      └─'──*──'───────┘

►──┬─────────────────────────────────────────────────────────────────────────┬──────►◄
   │            ┌─SAME─────────┐                                            │
   └─NEWNAME──=─┴─private-name─┴─┬──────────────────────────────────┬──.───┘
                                 └─PRDLIB──=──product-library-name─┘
```

The import (IMPORT) tag makes help modules defined in another panel group object available in the current panel group. This tag must appear in the prolog section of the panel group source following the panel group (PNLGRP) tags and the copyright (COPYR) tag. An IMPORT tag must be specified before any references by other tags to the help modules it defines.

This tag may also assign a private name to the imported help module. If a private name is assigned, the private name replaces the imported name in the scope of the current panel group. This allows names to be imported that would have conflicted with other names within the current panel group.

Required Attributes

NAME=*imported-name* | *'*'*

The internal name of the help module imported for use within the current panel group. For more information on the rules for naming, see “Name Syntax” on page 469.

If '*' is coded, all unresolved names are assumed to be imported from this panel group. Only one IMPORT tag may be coded with NAME='*'.

PNLGRP=*panel-group-name*

The name of the panel group that contains the help module specified by NAME. This is an i5/OS object name which obeys all object name rules. This name may be fully qualified.

Optional Attributes

NEWNAME=SAME | *private-name*

A new name used instead of the imported name within the current panel group. The new name suppresses the visibility of the imported name and serves as its replacement. Only NEWNAME=SAME is allowed when NAME='*' is specified.

PRDLIB=*product-library-name*

The name of the library added into the library search list as a product library, used to locate the panel group defining the help module specified by the NAME attribute on this tag. This attribute is not allowed when the PNLGRP attribute is fully qualified.

The product library is only used when imbedding help information using the imbed help (IMHELP) tag. The product library is ignored when referring to a help module using the HELP attribute on other UIM tags.

INFO (Information Area)

Syntax for Display Panels:

```
►►──:INFO──DEPTH──=──┬─area-depth─┬──┬──────────────────────────┬──────►
                     └─'──*──'────┘  │              ┌─SPACE─┐   │
                                     └─BOTSEP──=──┼─NONE──┼───┘
                                                    └─RULE──┘

►──┬──────────────────────┬──.──┬────────────┬──:EINFO.──────────►◄
   │            ┌─NO──┐   │     └─area-title─┘
   └─SCROLL──=──┴─YES─┴───┘
```

Syntax for Print Panels:

```
►►──:INFO──┬──────────────────────────┬──┬───────────────────────┬──────►
           │             ┌─SPACE─┐    │  │           ┌─NORMAL─┐  │
           └─BOTSEP──=──┼─NONE──┼────┘  └─TYPE──=──┴─PROLOG─┴──┘
                         └─RULE──┘

►──.──┬────────────┬──:EINFO.──────────────────────────►◄
      └─area-title─┘
```

An information area (INFO) tag provides textual information to explain the operation of an application or panel. This tag is allowed for display panels and print panels. It can construct a detailed instruction area on a panel.

Information areas are formatted with a width of 72 columns unless the width attribute on the panel tag is less than 74 columns. In this case, the information area is formatted with a width of two columns less than the width attribute on the panel tag.

Required Attribute

DEPTH=*area-depth* | '*'

The depth of the area in lines, including separators if any are specified. This attribute is required for display panels, but not allowed for print panels. If '*' is specified, the space remaining on the display after other panel elements are allocated is given to this area. Only one area in the panel may have '*' coded.

Optional Attributes

BOTSEP=SPACE | NONE | RULE

Defines the bottom separator for the information area. If SPACE is specified, a line of spaces is used.

NONE indicates that no separator line exists.

If RULE is specified, a line of underscored spaces is used.

SCROLL=NO | YES

Indicates whether or not this area is scrollable. This attribute is allowed only for display panels. NO indicates that the area is not scrollable.

YES indicates that the area is intended to be scrollable. For a `SCROLL`=YES area, a line of spaces is used by the UIM to provide a line for the scroll information. If `BOTSEP`=SPACE is specified also, one line is used for both the separator and the scroll information line.

TYPE=NORMAL | PROLOG
Indicates whether or not this information area is a prolog area. The prolog area is printed only once after the title line on the first page. This attribute is allowed only for the print head (PRTHEAD) tag. This attribute is valid only when the INFO tag appears between the PRTHEAD and the EPRTHEAD tags for information areas in a PRTHEAD panel. NORMAL is the default value.

Optional Text

area-title
The title of the area. If no text is specified, no title line is allocated to the area. The text must appear on the same or next line as the tag, can contain only the reverse text (RT) tag, and cannot exceed a maximum length of 55 characters long.

Print Formatting Considerations

Printed information areas are formatted like displayed areas, except when four lines of an information area do not fit on a page when printing. If this is the case, a page eject occurs and the information area is printed on the next page. Information areas are formatted to a width of 72 columns, as is done for information areas on display panels, regardless of the width specified on the PRTHEAD or print panel (PRTPNL) tag.

ISCH (Index Search)

```
►►—:ISCH—ROOTS—=—'root-word-list' —.—index-entry-text——————————————————————————————►◄
```

The index search (ISCH) tag identifies the text displayed by the index search function if the user enters a word that matches a synonym. The ISCH tag is only allowed in source code for panel group objects and is not allowed in the source for menu objects. The placement of the ISCH tag determines the help module displayed when the index entry is selected. The ISCH tags should appear immediately after the help module (HELP) tag it refers to.

Required Attribute

ROOTS=*'root-word-list'*
A list of up to 50 root words that apply to the index entry. Each root word may be up to 20 characters and can contain only the characters A through Z, a through z, and 0 through 9. One or more blanks must be specified between words, so the entire list must be enclosed in apostrophes. The `ROOTS` attribute may be repeated, allowing you to define more root words for an index entry than fits on one source line.

The root words serve as a link between the ISCH tags and the index search synonym (ISCHSYN) tags, and do not appear to the user. To allow the user to search for an index entry, each root word specified here must have a matching root word on an ISCHSYN tag.

Required Text

index-entry-text
The text of the index entry presented when one of the synonym words is selected. The text may contain up to 72 characters but can not contain other tags. It must appear on the same or next line as the period which ends the tag definition and cannot span two lines in the source.

If the index entry is a subtopic, the index-entry-text is indented two spaces from the text of the first higher index entry in the hierarchy of index entries.

The text which is provided is used as the topic title when topics are presented for selection. If no text is specified on the HELP tag of the topic, the text provided on the ISCH tag is also used as the panel title when the topic is selected for presentation. When used as a panel title, no more than 55 characters of the text is shown. If the text contains more than 55 characters, it is truncated at a blank and an ellipse (...) is placed after the text to indicate that truncation has occurred.

Example: Index Search

This example shows some ISCHSYN tags and the ISCH tags that use them:

UIM Source

```
:ISCHSYN ROOT='copy'.copy copying copies
:ISCHSYN ROOT='delete'.delete deleting deletes
:ISCHSYN ROOT='delete'.remove removes removing
:ISCHSYN ROOT='folder'.folder folders
:ISCHSYN ROOT='folder'.document documents
   ⋮
:help name=fldcpy.
:ISCH ROOTS='copy folder'.Copying folders
   ⋮
:ehelp.
:help name=flddlt.
:ISCH ROOTS='delete folder'.
Deleting folders
:ehelp.
```

ISCHSUBT (Index Search Subtopic)

```
►►—:ISCHSUBT—TOPICS—=—'help-module-name-list' —.——————————————————————————————►◄
```

The index search subtopic (ISCHSUBT) tag identifies the help modules within the same panel group that are subtopics under the preceding topic specified on an index search (ISCH) tag. The ISCHSUBT tag must appear after the ISCH tag. This tag is repeatable.

Any help module with an ISCH tag that is not identified by an ISCHSUBT tag is a primary topic in the index search hierarchy. Therefore, if no ISCHSUBT tags are used, all help modules are primary topics and there is no hierarchy in the index search. The ISCHSUBT tag is allowed only in source code for panel group objects; it is not allowed in the source code for menu objects.

Required Attribute

TOPICS=*'help-module-name-list'*

Identifies the help modules within the same panel group that are subtopics under the preceding topic specified on an index search (ISCH) tag. The order in which the help modules appear on the TOPICS attribute is the order in which they are displayed in the index search hierarchy. A help topic name cannot be specified twice in this list. For more information on the rules for help modules names, see "Name Syntax" on page 469. This attribute is repeatable.

A topic can be the subtopic of more than one topic.

Topics can be nested to no more than 16 levels.

Example: Index Search Hierarchy

The following example shows how the ISCH tags and ISCHSUBT tags work together to form an index search hierarchy:

UIM Source

```
:HELP name=mainhelp.
:ISCH roots='root words'.
Main Help Topic
:ISCHSUBT topics='help1'
          topics='help2'.
   ⋮
:EHELP.
:HELP name=help1.
:ISCH roots='root words'.
Help number 1
:ISCHSUBT topics='help3 help4'.
   ⋮
:EHELP.
:HELP name=help2.
:ISCH roots='root words'.
Help number 2
:ISCHSUBT topics='help3'
   ⋮
:EHELP.
:HELP name=help3.
:ISCH roots='root words'.
Help number 3
   ⋮
:EHELP.
:HELP name=help4.
:ISCH roots='root words'.
Help number 4
   ⋮
:EHELP.
```

This UIM source creates the following index search hierarchy:

```
Title of this index
  Main Help Topic
    Help number 1
      Help number 3
      Help number 4
    Help number 2
      Help number 3
```

ISCHSYN (Index Search Synonym)

```
►►──:ISCHSYN──ROOT──=──root-word──.──synonym-words──────────────────────────────►◄
```

The index search synonym (ISCHSYN) tag identifies the variations and synonyms for the root words used in the index search function. The ISCHSYN tag is only allowed in source for panel group objects and is not allowed in the source for menu objects.

The text must be entered on one line. If more than one line is needed, multiple ISCHSYN tags may be coded. This text, when combined with the index search (ISCH) tag, determines the index entries displayed when the user enters words for the index search function.

The ISCHSYN tags build a table of synonyms, which serves as a link to the ISCH tags. As words are entered for an index search, they are matched with the words in the synonym table to link to the entries displayed.

There is no restriction on the placement of the ISCHSYN tags, but to make maintenance and translation easier, they should be placed in one area, such as the beginning of the panel group or in a panel group object which contains only ISCHSYN tags.

Required Attribute

ROOT='*root-word***'**
The root word to which the synonyms apply. If a real word is used for the root word, like “copy”, that word should also be entered in the *synonym-words* field.

The root word is used in the `ROOTS` attribute of the ISCH tag and may be up to 20 characters. A root word can contain only the characters A through Z, a through z, and 0 through 9.

Required Text

synonym-words
Special Format. Specifies variations and synonyms for the root word. The synonym words must be separated by blanks. If additional synonyms are needed for a root word, additional ISCHSYN tags should be entered specifying the same root word. The additional synonyms are added to the previous synonyms.

Each synonym word must be 40 characters or less and can contain no spaces. It cannot contain a period, left or right parenthesis, a semicolon, a comma, question mark, or colon.

Different languages have a larger or smaller number of synonyms for each word. To provide meaningful results, the translation of a root word must be done by providing a list of synonym words and by not translating each English synonym.

The UIM automatically handles lowercase, single-byte character set (SBCS) synonym words the same way as uppercase words, using the code page specified on the `TXTCHRID` attribute of the panel group (PNLGRP) tag.

Example: Index Search Synonyms

This example shows several ISCHSYN tags and how they are referred to by ISCH tags.

UIM Source

```
:pnlgrp...
   ⋮
:ISCHSYN ROOT='copy'.copy copying
:ISCHSYN ROOT='copy'.duplicate duplicating
:ISCHSYN ROOT='copy'.model
:ISCHSYN ROOT='root1'.remove removing
:ISCHSYN ROOT='root1'.delete deleting
:ISCHSYN ROOT='root1'.trash discard
:ISCHSYN ROOT='folder'.folder folders
:ISCHSYN ROOT='folder'.document documents
:ISCHSYN ROOT='folder'.data information
   ⋮
:help name=fldcpy.
:ISCH ROOTS='copy folder'.Copying a folder
:h2.Copying a Folder
   ⋮
:ehelp.
   ⋮
:help name=flddlt.
:ISCH ROOTS='root1 folder'.Deleting a folder
:h2.Deleting a folder
   ⋮
:ehelp.
```

In this example, the ISCHSYN tags builds a synonym table containing the following entries:

Root Word	Synonym Words
copy	`copy copying duplicate duplicating model`
root1	`remove removing delete deleting trash discard`
folder	`folder folders document documents data information`

The ISCH tag for the "Copying a folder" entry points to the "copy" and "folder" search synonym lists.

When the user requests ″copy folders,″ the entry is found and displayed because ″copy″ and ″folders″ are both listed as matches in the ISCHSYN table. If the user entered ″copy folders,″ ″folders, copy,″ or ″copying documents,″ the entry ″Copying a folder″ is displayed.

KEYI (Key List Item)

```
►►──:KEYI──KEY──=──key-name──HELP──=──help-module-name──┬───────────────────────────────┬──────►
                                                        └─ACTION──=──'──action-text──'──┘

►──┬────────────────────────────┬──┬──────────────────────┬──┬──────────────────────────────────┬──►
   └─COND──=──condition-name────┘  │                ┌─YES─┐│  └─PRIORITY──=──priority-number───┘
                                   └─VARUPD──=──────┴─NO──┴┘

►──.──┬─────────────────────────┬──────────────────────────────────────────────────────────────►◄
      └─key-description-text────┘
```

The key list item (KEYI) tag defines a single function key. This tag must occur between the KEYL and EKEYL tags. It assigns displayable text for a specified key and identifies the action to take place when that key is pressed.

Required Attributes

KEY=*key-name*
The names for the engraved or software defined function keys are Attn, Enter, F1-F24, Help, Home, Print, Pagedown, Pageup, and Sysreq.

HELP=*help-module-name*
The associated help module for the key description. The help module name may be a name imported from another panel group, but it must follow the rules for names outlined earlier in this chapter. For more information on the rules for naming, see "Name Syntax" on page 469.

ACTION=*'action-text'*
The action occurring when the function key is pressed. This attribute is conditionally required. For certain keys, the `ACTION` attribute must not be specified since the UIM does not handle that particular key. For all other keys, the `ACTION` attribute is required.

The valid forms of action text are:

- *'ACTIONS'*
- *'CALL program-reference'* For a description of the interface between the UIM and the function key CALL program, see the **APIs** topic in the iSeries Information Center.
- *'CANCEL'*
- *'CHGVIEW'*
- *'CMD command-string'* Any dialog variable in the command string must be preceded by an ampersand to denote variable substitution.
- *'CMDLINE'*
- *'ENTER'*

- *'EXIT'*
- *'HELP'*
- *'HOME'*
- *'MENU qualified-menu-name RTNPNT|NORTNPNT'*
- *'MOREKEYS'*
- *'MOVETOP'*
- *'PRINT'*
- *'PROMPT'*
- *'RETRIEVE'*
- *'RETURN positive-integer'*
- *'PAGEUP'*
- *'PAGEDOWN'*

For a description of each of these actions, see Appendix B, "UIM Dialog Commands," on page 641.

Engraved key assignments are enforced by the compiler. These keys are the keys listed, except for F1 through F24. If engraved keys are assigned anything but their corresponding dialog function (or no action in certain cases), a compile error results. The following is a list of the engraved keys and their corresponding actions:

Key	**Action**
Attn	No action
Enter	'ENTER'
Help	'HELP'
Home	'HOME'
Print	'PRINT'
Pageup	'PAGEUP'
Pagedown	'PAGEDOWN'
Sysreq	No action

Optional Attributes

COND=*condition-name*
The key is in effect on the panel only if the condition specified is true. The condition must be defined in the panel group with the condition definition (COND) tag.

VARUPD=YES | NO
If YES is coded, validity checking occurs and the variable pool is updated with values entered by the user before the action takes place.

If NO is coded, the action is performed immediately and no variable pool updating occurs. If the `VARUPD` attribute is not specified and the EXIT or CANCEL dialog command is specified on the `ACTION` attribute of this tag, a warning message is generated at compile time stating that `VARUPD`=YES is assumed. This is done as a reminder to you, because you may not want to check for valid input data before allowing the user to exit a screen through these dialog commands. To avoid this message, specify `VARUPD`=YES or `VARUPD`=NO on the key assignment.

If no `ACTION` attribute is specified, this attribute is ignored.

PRIORITY=*priority-number*
The order in which keys should be displayed when the MOREKEYS dialog command is performed.

Key items with a lower number priority are displayed before key items with a higher number priority. A specified priority number must be greater than or equal to one and less than or equal to 99.

If a priority is not specified for a key, the default value depends on the action specified:

Action	**Default**
EXIT	1
CANCEL	2
All others	99
No action	99

The `PRIORITY` attribute is used only within a key list that has `ACTION=`MOREKEYS specified on one of the KEYI tags. Otherwise, the priority attribute is ignored.

Optional Text

key-description-text

The displayable description for the associated key. For example, `"F5=Refresh"` or `"F12=Cancel"`. If no text is specified, a description of the key is not displayed on the panel, although the key is active. The text must appear on the same or next line as the tag and can only contain the reverse text (RT) tag.

The function key area is formed by concatenating the text from the various key list items, with each key separated by at least three spaces. If the text does not fit on the rest of a line, it is placed on the next line.

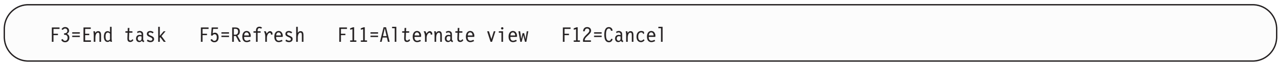
Example: Key Definitions

The following example defines the function keys F3, F5, F11, and F12.

UIM Source

```
:keyl name=keylist
          help=hkeylist.
:keyi key=f3
          action='exit set'
          help=exit
          .F3=End task
:keyi key=f5
          action='return 5'
          help=refresh
          .F5=Refresh
:keyi key=f11
          action=chgview
          help=chgview
          .F11=Alternate view
:keyi key=f12
          action='cancel set'
          help=cancel
          .F12=Cancel
:ekeyl.
```

Results



```
F3=End task   F5=Refresh   F11=Alternate view   F12=Cancel
```

KEYL (Key List)

```
►►──:KEYL──NAME──=──key-list-name──┬──────────────────────────────┬──.──:EKEYL.──────────────────────►◄
                                   └─HELP──=──help-module-name─┘
```

The key list (KEYL) tag indicates the beginning of a list of key definitions. Each list may be referred to by one or more panel definitions. An EKEYL tag must be specified at the end of the key list.

Other tags can be nested within the KEYL tag. These tags are listed in the following table. The table defines the order in which the tags must appear and specifies on which page more information can be found for each tag.

Table 62. Tag Allowed Between the KEYL and EKEYL Tags

Tag Name	Order	Page
KEYI (Key list item)	1	545

Required Attribute

NAME=*key-list-name*
The name assigned to the key list. For more information on the rules for naming, see “Name Syntax” on page 469.

Optional Attribute

HELP=*help-module-name*
Identifies online help information explaining the purpose of the function keys. The name of the help module can be a name imported from another panel group, but must follow the rules for names outlined earlier in this chapter. For more information on the rules for names, see “Name Syntax” on page 469.

Example: Key List

This is an example of a key list.

UIM Source

An example of a key list follows:

```
:keyl name=keylist.
:keyi key=f1
      help='key/helpf1'
      action=help.
:keyi key=f3
      help='key/exit'
      action='exit set'.F3=Exit
:keyi key=f12
      help='key/cancel'
      action='cancel set'.F12=Cancel
:keyi key=f24
      help='key/morekeys'
      action=morekeys.F24=More keys
:keyi key=enter
      help='key/enter'
      action=enter.
:keyi key=help
      help='key/help'
      action=help.
:keyi key=pagedown
      help='key/pagedown'
      action=pagedown.
:keyi key=pageup
      help='key/pageup'
      action=pageup.
:keyi key=print
      help='key/print'
      action=print.
:ekeyl.
```

LINES (Unformatted Lines)

```
►►─:LINES.──┬───────────────────┬─:ELINES.──────────────────────────────────────────►◄
            └─unformatted-lines─┘
```

The unformatted lines (LINES) tag identifies an area of user-controlled line entry where the lines are not automatically concatenated. Text is formatted the same way it is entered. This tag is only allowed in information areas and help areas. The LINES tag requires a matching end tag.

Note: Care should be taken when using unformatted lines within lists, because text that does not fit on one line wraps to column one of the next line. The current left margin changes when nested lists are being formatted. If there is online help information containing unformatted lines imbedded at various locations, including within lists, it may not look the same each time.

Help modules displayed in full-screen format or in an extended help window are indented by 4 bytes. Therefore, the formatting width initially is 6 bytes less than the width specified on the help module. When the LINES tag appears in an information area, the formatting width initially is two bytes less than the width specified on the panel tag.

Optional Text

unformatted-lines
Although the text for the unformatted lines is not required, the tag has no meaning when no text is specified.

Example: Unformatted Lines

This example illustrates how text is formatted the same way it is entered.

UIM Source

```
:LINES.
First line
Second line
:ELINES.
```

Results

```
First line
Second line
```

LINK (Hypertext Link Definition)

```
►►──:LINK──PERFORM──=──'action-text' ─┬─────────────────────────────────────────┬──────────────────────►
                                      └─┬─UNLESS1─┬──=──'conditional-expression' ┘
                                        ├─UNLESS2─┤
                                        ├─UNLESS3─┤
                                        └─UNLESS4─┘

►─┬──────────────────────────────────┬─┬─────────────────────────────────────────┬────────────────────►
  └─┬─THENDO1─┬──=──'action-text' ───┘ └─LINKWHEN──=──'conditional-expression' ──┘
    ├─THENDO2─┤
    ├─THENDO3─┤
    └─THENDO4─┘

►─.─┬──────────────────┬─:ELINK.──────────────────────────────────────────────────────────────────────►◄
    └─hypertext-phrase─┘
```

The hypertext link definition (LINK) tag identifies the reference phrase text which is the anchor of a static hypertext link. A reference phrase may appear in any UIM help area, but may not appear in an information area. This tag requires a matching end tag. It also identifies action taken when the reference phrase is selected. The only type of action supported is to display a help module.

The LINK and ELINK tag must be specified on word boundaries. If the two characters immediately following the ELINK tag are a punctuation mark and a blank, the UIM automatically extends the emphasis attribute to include the punctuation mark. This allows the punctuation mark and the text associated with it to be displayed using the same emphasis.

On a monochrome device, the reference phrase text is displayed as highlighted, underscored text. On a color device, the reference phrase text is displayed as underscored, yellow text. Because of hardware limitations, the underscore is not visible on a PS/2 computer or on a personal computer with a color device.

The only tag that can be used within a link tag is the reverse text (RT) tag. LINK tags may not be used within other LINK tags, but may be used within any of the following tags:

CIT	Title citation
DD	Definition
FIG	Figure
FIGCAP	Figure caption text
HP*n*	Highlighted phrase
LI	List item
LINES	Unformatted lines
LP	List part
NT	Note
P	Paragraph
PC	Paragraph continuation
PD	Parameter description
PK	Programming keyword
PV	Programming variable

XMP Example

Required Attribute

PERFORM=*'action-text'*

The action occurring when the reference phrase is selected and all of the condition expressions for the `UNLESS`*n* attributes have evaluated to false. The valid form of action text is:

- *'DSPHELP help-module-name [panel-group-name]'*

Note: The brackets in the action text above indicate that the panel group name is optional. They are *not* required in the UIM source.

For a description of the DSPHELP action, see Appendix B, "UIM Dialog Commands," on page 641.

Optional Attributes

UNLESS*n***='***conditional-expression***'.**

The only allowed values for *n* are 1 through 4.

The `UNLESS`*n* and `THENDO`*n* attributes must be coded as pairs. The expressions on the `UNLESS`*n* attributes are evaluated in numeric order. If one evaluates to true, the corresponding `THENDO`*n* action is performed and all higher-numbered `UNLESS`*n* and `THEN`*n* attributes and the `PERFORM` attribute on this tag are ignored.

For information on using conditional expressions with the `UNLESS`*n* attribute, see "Conditional Expressions."

THENDO*n***=***'action-text'*

The only allowed values for *n* are 1 through 4.

This attribute specifies the action occurring when the reference phrase is selected and the conditional expression for the corresponding `UNLESS`*n* attribute evaluates to true and all the conditional expressions for the lower-numbered `UNLESS`*n* attributes evaluate to false.

The `UNLESS`*n* and `THENDO`*n* attributes must be coded as pairs. The `UNLESS`*n* attributes are evaluated in numeric order. If one evaluates to true, the corresponding `THENDO`*n* action is performed and all higher-numbered `UNLESS`*n* attributes and the `PERFORM` attribute on this tag are ignored.

The valid form of action text is:

- *'DSPHELP help-module-name [panel-group-name]'*

Note: The brackets in the option text above indicate that the panel group name is optional. They are *not* required in the UIM source.

For a description of the DSPHELP action, see Appendix B, "UIM Dialog Commands," on page 641.

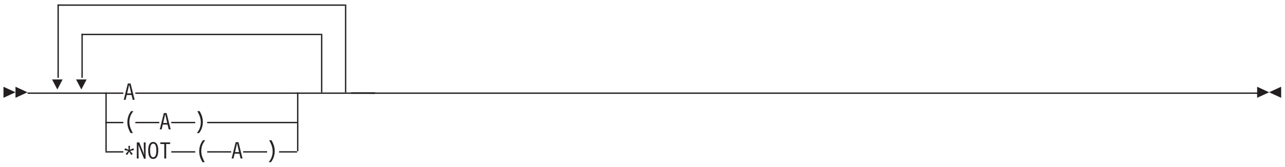
LINKWHEN='*conditional-expression***'.**

When the `LINKWHEN` expression evaluates to true or is not coded, the reference phrase is made available and a user is allowed to select it. When the `LINKWHEN` expression evaluates to false, the reference phrase is not activated and may not be selected.

For more information on using conditional expressions with the `LINKWHEN` attributes, see "Conditional Expressions."

Conditional Expressions

Although the set of conditions that can be formed with the `LINKWHEN` attribute is identical to the set of conditions that can be formed with the `UNLESS`*n* attributes, the significance of those conditions is different. The `LINKWHEN` attribute activates or deactivates the LINK tag, while the `UNLESS`*n* attribute selects the action performed when an active LINK tag is selected. A conditional expression is a true or false expression in the following form:



where A is an operand which can be one of the following:

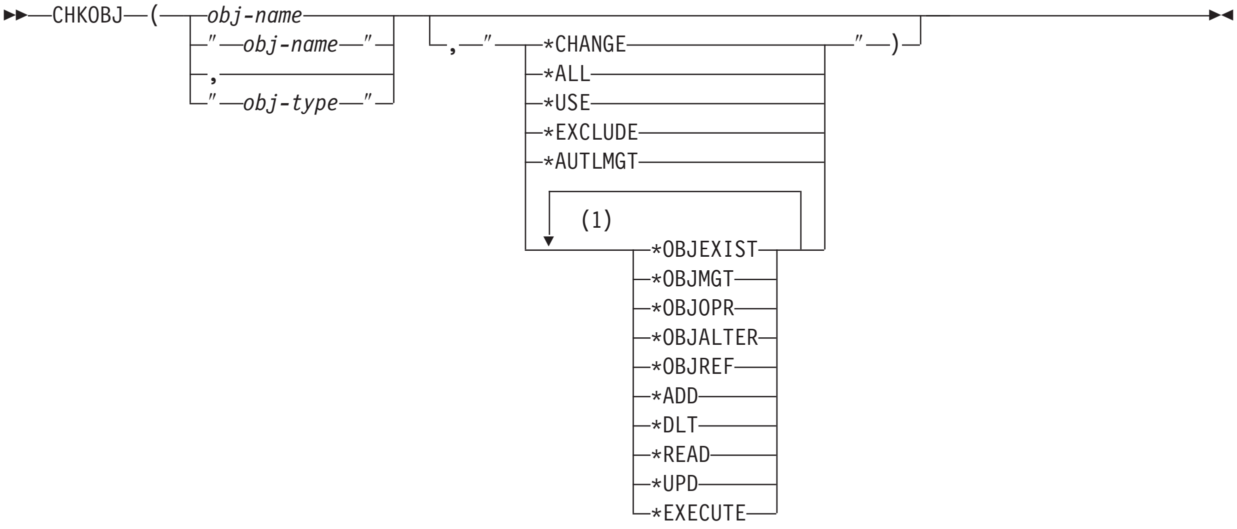
- A conditional expression
- A built-in function

The logical OR character (|) can be used in place of *OR, the ampersand character (&); can be used in place of *AND, and the logical NOT character (¬) can be used in place of *NOT. Because the logical OR and logical NOT characters are not in the invariant character set, their use is not recommended. For code page 00037, the common USA code page, the hexadecimal value of the logical OR character is X'4F', and the hexadecimal value of the logical NOT character is X'5F'. The UIM compiler uses these hexadecimal values regardless of the code page of the source.

There are three built-in functions:

CHKOBJ

Evaluates to true if the object is found on the system and the current job possesses at least the level of authorization to the object specified by the authorities. Arguments must be character strings enclosed in double quotation marks ("). The object name follows i5/OS object naming conventions. The object type is any of the allowable object types for the DSPOBJD command and the authorities must be a single value or a list of authorizations to be checked for. The values of these authorizations are separated by blanks. The following syntax diagram illustrates the authorities and how to use them:



Notes:

1 Each value can be used only once, a maximum of 7

If no authorities are specified, no authorization check is performed and the function becomes an existence check.

When a program adopts the authority of its owner, that authority is normally used to authorize operations performed by that program. However, when help is requested, those adopted authorities are ignored while the UIM displays the online help information. Therefore, no adopted authorities are used when the CHKOBJ function is used on the UNLESS or LINKWHEN attributes on the LINK tag.

For more information about programs adopting the authority of the owner, see the USRPRF(*OWNER) parameter on the CRTCLPGM command.

CHKPGM

The CHKPGM function accepts a qualified program name to be called as an exit program from UIM. If the program is not able to be called, the function evaluates to false. The exit program will determine if the function should be set to true or false and return an indicator to UIM. The program name must be enclosed in double quotation marks (″). The *LIBL special value can be used in place of the library name. If no library name is entered, *LIBL is the default.

The following syntax diagram illustrates the valid argument values for the CHKPGM function:

```
►►──CHKPGM──(──″──pgm-name──″──)──────────────────────────────────────────────►◄
```

CHKUSRCLS

The user class argument specified for the function is compared to the user class parameter from the user profile of the current job. The function evaluates to true if the user profile has the same or greater value than the function argument. The function argument must be enclosed in double quotation marks (").

The following syntax diagram illustrates the valid argument values for the user class:

```
►►──CHKUSRCLS──(──┬─*SECOFR─┬─)───────────────────────────────────────────────►◄
                  ├─*SECADM─┤
                  ├─*PGMR───┤
                  ├─*SYSOPR─┤
                  └─*USER───┘
```

Examples of conditional expressions follow:

```
CHKOBJ("OBJECT","*FILE","*USE")

CHKOBJ("PANELGRP","*PNLGRP")
          *AND CHKUSRCLS("*PGMR")

CHKOBJ("DOCUMENT","*DOC","*READ *UPD")
          *OR CHKUSRCLS("*SYSOPR")

*NOT(CHKOBJ("PROGRAM","*PGM"))
CHKPGM("*LIBL/PROGRAM")
CHKPGM("PROGRAM")
```

Bidirectional Considerations

The value of the BIDI attribute on the panel group (PNLGRP) tag of all possible panel groups that can be reached by taking hypertext links should be the same. If they are not the same, it is possible that the resulting screen or pop-up window is displayed with the opposite orientation when a reference phrase is selected.

Example: Hypertext Link

This example shows how hypertext links can be used to provide definitions for terms.

UIM Source

```
:HELP NAME='hyper/help'.Hypertext in i5/OS
:P.
Hypertext lets users explore
the online help information in a way that is most
natural for them.
Hypertext links can be used within :LINK
PERFORM='DSPHELP item/specific/help'
```

```
        .item specific help:ELINK.,
:LINK PERFORM='dsphelp extended/help'
        .extended help:ELINK., and
:LINK PERFORM='dsphelp index/search'
        .index search help:ELINK..
:EHELP.

:HELP NAME='item/specific/help'
        .Definition of Item Specific Help
:P.
Item specific help is . . .
:EHELP.

:HELP NAME='extended/help'
         .Definition of Extended Help
:P.
Extended help is . . .
:EHELP.

:HELP NAME='index/search'
         .Definition of Index Search
:P.
Index search allows you to tell the
system to search for specific information.
Index search information is made more
useful by the addition of
:LINK PERFORM='dsphelp hyper/help'
         .hypertext:ELINK. because
it allows you to link to additional help
topics.
:EHELP.
```

Results

```
..................................................................
:                        Hypertext on i5/OS                      :
:                                                                 :
:  Hypertext lets users explore the online information in a way   :
:  that is most natural for them.  Hypertext links can be used    :
:  within   item specific help,  extended help, and               :
:  index search help.                                             :
:                                                         Bottom  :
:  F2=Extended help   F10=Move to top    F11=Index search         :
:  F12=Cancel         F13=User support   F24=More keys            :
:                                                                 :
:.................................................................:
```

When the cursor is moved to *INDEX SEARCH HELP* and the Enter key is pressed, the following screen is shown.

```
..................................................................
:                    Definition of Index Search                   :
:                                                                 :
:  Index search allows you to tell the system to search for specific :
:  information.  Index search information is made more useful by the :
:  addition of > hypertext because it allows you to link to        :
:  additional help topics.                                        :
:                                                         Bottom  :
:  F6=Viewed topics   F10=Move to top    F11=Index search         :
:  F12=Cancel         F13=User support   F24=More keys            :
:                                                                 :
:.................................................................:
```

LIST (List Area)

Syntax for Display Panels:

```
►►──:LIST──DEPTH──=──┬─area-depth─┬──LISTDEF──=──list-name──────────────────►
                     └─'*'────────┘

►──┬──────────────────────────┬──┬─────────────────────┬──┬────────────────────┬──►
   │              ┌─SPACE─┐   │  │             ┌─YES─┐ │  │              ┌─0─┐ │
   └─BOTSEP──=────┼─NONE──┼───┘  └─SCROLL──=───┴─NO──┴─┘  └─MAXHEAD──=───┼─1─┼─┘
                  └─RULE──┘                                              ├─2─┤
                                                                         ├─3─┤
                                                                         └─4─┘

►──┬──────────────────────────┬──┬───────────────────────────────┬──►
   │                ┌─SPACE──┐ │  └─VIEW──=──dialog-variable-name─┘
   └─BODYSEP──=─────┼─INDENT─┼─┘
                    ├─BOTH───┤
                    └─NONE───┘

►──┬────────────────────────┬──┬─────────────────────┬──►
   │              ┌─NONE───┐ │  │             ┌─NO──┐ │
   └─ACTOR──=─────┼─UIM────┼─┘  └─EXTACT──=───┴─YES─┴─┘
                  └─CALLER─┘

►──┬─────────────────────────┬──┬─────────────────────┬──►
   │              ┌─NONE───┐ │  │               ┌─2─┐ │
   └─SELECT──=────┼─SINGLE─┼─┘  └─MAXACTL──=────┼─1─┼─┘
                  └─MULTI──┘                    └─3─┘

►──┬────────────────────────────────┬──┬───────────────────────────────────┬──►
   └─PARMS──=──dialog-variable-name─┘  └─HEADSIZE──=──dialog-variable-name─┘

►──.──┬────────────┬──:ELIST.──────────────────────────────────────►◄
      └─area-title─┘
```

Syntax for Print Panels:

```
►►──:LIST──LISTDEF──=──list-name──┬──────────────────────────┬──►
                                  │              ┌─SPACE─┐   │
                                  └─BOTSEP──=────┼─NONE──┼───┘
                                                 └─RULE──┘

►──┬─────────────────────┬──┬──────────────────────────┬──►
   │                ┌─0─┐ │  │                ┌─SPACE──┐ │
   └─MAXHEAD──=─────┼─1─┼─┘  └─BODYSEP──=─────┼─INDENT─┼─┘
                    ├─2─┤                     ├─BOTH───┤
                    ├─3─┤                     └─NONE───┘
                    └─4─┘

►──┬───────────────────────────────┬──┬───────────────────────────────────┬──►
   └─VIEW──=──dialog-variable-name─┘  └─HEADSIZE──=──dialog-variable-name─┘

►──.──┬────────────┬──:ELIST.──────────────────────────────────────►◄
      └─area-title─┘
```

The list area (LIST) tag defines a list area on a panel.

This tag is allowed for display panels and print panels, except for the print head panel (PRTHEAD) tag. A **list area** consists of an arbitrary number of rows of like columns, and can be scrolled up and down if the number of rows exceeds the area in which the list is displayed.

A list area presents a view of a UIM list, which is manipulated by the UIM application programming interfaces (APIs). UIM lists are named and described by the list definition (LISTDEF) tag. The UIM displays the entries in the list, handling the scroll operations which allow the user to see all the entries.

The area may have multiple views, which are selected by the CHGVIEW dialog command. The CHGVIEW dialog command can be assigned to a function key which alternates between the views.

Other tags can be nested within the LIST tag. These tags are listed in the following table. The table defines the order in which the tags must appear, indicates which tags can be used in display panels only, print panels only, or both (specified by a D, P, or B, respectively), and specifies on which page more information can be found for each tag.

When more than one tag is listed with the same order number, all tags of that number can be mixed in any order. However, a tag with a higher order number cannot precede a tag with a lower order number. For example, a tag with an order number of three cannot precede a tag with an order number of one or two.

Table 63. Tags Allowed Between the LIST and ELIST Tags

Tag Name	Order	Use	Page
TOPINST (Top instruction line)	1	D	630
LISTACT (List action)	2	D	562
LISTGRP (List column group)	3	B	575
LISTCOL (List column)	3	B	568
LISTVIEW (List view)	4	B	578
BOTINST (Bottom instruction line)	5	D	474

Required Attributes

DEPTH=*area-depth* **| '*'**
This attribute is required for display panels but is not allowed for print panels. The depth of the area in lines, including separators if any are specified. If '*' is specified, the space remaining on the display after all other panel elements are allocated is given to this area. Only one area in the panel may have '*' coded.

LISTDEF=*list-name*
The name of the UIM list from which the data for this list area is taken. The list must be defined in the panel group using the LISTDEF tag.

Optional Attributes

BOTSEP=SPACE | NONE | RULE
Defines the bottom separator for the list area. If SPACE is specified, a line of spaces is used.

NONE indicates that no separator line exists.

If RULE is specified, a line of underscored spaces is used as a separator line.

SCROLL=YES | NO
Indicates whether or not the list area is scrollable. This attribute is allowed only for display panels.

YES indicates that the list area is intended to be scrollable. For SCROLL=YES, a line of spaces is used by the UIM to provide a line for the scroll information.

NO indicates that the list is not scrollable. Ordinarily, NO would not be used, but for short lists it can be specified to disable the scrolling keys for the list area.

If BOTSEP=SPACE is specified and there are no bottom instructions for both the separator and scroll information line, one line is used.

MAXHEAD= 0 | 1 | 2 | 3 | 4
The maximum number of lines for column headings. From 0 to 4 lines can be specified. The default is 0, indicating that no column headings are allowed. For more information on column headings, see "LISTCOL (List Column)" on page 568. This attribute includes any heading line required by the column group headings.

When column headings are used, provide expansion space for national language translation by specifying a MAXHEAD value larger than the number of heading lines required. If this expansion space is not provided, it must be provided within the column width determined by the MAXWIDTH attribute of each list column (LISTCOL) tag.

This attribute cannot be specified if the HEADSIZE attribute on this tag is specified for use with variable column headings.

BODYSEP=SPACE | INDENT | BOTH | NONE
The type of visual separation distinguishing the body of the area from other elements in the area, particularly any top and bottom instruction lines within the area.

SPACE leaves a blank line after the top instruction lines and before the bottom instruction lines. List columns in the body of the area begin in the leftmost position of the layout column and are not indented with respect to instruction lines. If the area contains no top or bottom instruction lines, no blank lines are reserved before or after the area body.

INDENT is used to indent list columns in the body of the area by two positions from the leftmost position in the layout column where the instruction lines begin. If the area contains top or bottom instruction lines, no blank line is reserved between the instructions and the area body except if the area is scrollable. If the area is scrollable, a blank line is reserved for the scroll information.

BOTH leaves a blank line after the top instruction lines and before the bottom instruction lines, and indents the body two positions from the leftmost position in the layout column where the instruction lines begin. If the area contains no top or bottom instruction lines, no blank lines are reserved before or after the area body.

NONE does not leave a blank line between instruction lines and the body and does not indent the body with respect to the layout column except if the area is scrollable. If the area is scrollable, a blank line is reserved to provide a line for the scroll information.

VIEW=*dialog-variable-name*
This is a BIN 15 dialog variable which determines the view of the list which appears to the user. Valid values are 0 through one less than the number of views defined. The UIM changes this variable to match the number of the view that is active (where 0 is the first view) when the value is not valid or when the CHGVIEW dialog command is run for the list area.

The first list view (LISTVIEW) tag for the list area defines view 0, the second LISTVIEW tag for the list area defines view 1, and so on for each LISTVIEW tag in the list area.

For more information on list views, see "LISTVIEW (List View)" on page 578. The VIEW attribute must be specified if more than one LISTVIEW tag is specified for the list area.

ACTOR=NONE | UIM | CALLER.
If ACTOR=NONE is used, the list is not an action list. This attribute is allowed only for display panels. No list action (LISTACT) tags can be specified and the UIM does not do action list processing for the list area.

If ACTOR=UIM is specified, the actions indicated by the LISTACT tags are performed by the UIM and must have the ENTER attribute specified.

If ACTOR=CALLER, the actions indicated by the LISTACT tags must be performed by the calling program of the display panel (QUIDSPP) API.

EXTACT=NO | YES

Specifies whether or not an action list has extended action capability. This attribute is allowed only for display panels. With an extended action list, the first line below the column headings is used for an extended action entry. This line contains the action option column, along with input-capable fields for additional list columns.

The user can enter a list action option in the option field for the extended action entry, along with data in the other columns. The action identified by the list action tag is performed using the data entered for the other list columns. The data entered into the extended action entry columns does not have to match the data in an existing list entry.

EXTACT=NO indicates that the list does not have extended action capability.

EXTACT=YES indicates that the list does have extended action capability. This is allowed only if ACTOR=UIM or ACTOR=CALLER is specified on this tag. The action list option column and at least one other list column defined in each list view must be defined for extended action use with EXTACT=YES on the LISTCOL tag. Any LISTACT tags which can operate on the extended action field must have ACTFOR=BOTH or ACTFOR=EXTACTE specified.

When a panel is displayed which has an extended action list but the list is not currently active in the open application, the list is activated by the QUIDSPP API.

SELECT=NONE | SINGLE | MULTI

If SELECT=NONE is used, the list is not a selection list. This attribute is allowed only for display panels.

If SELECT=SINGLE is specified, the list area is a single-choice selection list. When a single-choice selection list is displayed, a period (.) precedes each list entry. The user can select only one list entry by typing the slash (/) or the country-designated character over the period.

If SELECT=MULTI is specified, the list is a multiple-choice selection list. Multiple list entries may be chosen by entering the slash (/) or the country-designated character into the entry field preceding each list entry to select. To deselect a choice, the user should type a blank over, or delete the slash or country-designated character, in the list entry.

A value of 1000 is set in the action variable for the selected list entries.

ACTOR=NONE must be specified or made a default on this tag if the value of SELECT is either SINGLE or MULTI. An action variable must be declared for each view of the list.

MAXACTL= 2 | 1 | 3

The maximum number of lines used for list action descriptions. From 1 to 3 lines can be specified; 2 is the default. Only as many list action lines as needed are used. This attribute is allowed only for display panels.

The only time that MAXACTL=1 would be useful is when there are two or more list actions and you do not want two lines of list actions after national language translation.

When the UIM formats the first two list action lines, it attempts to align the start of each action with the line above or below it. This is done by moving text a few columns to the right on either the first or second line. When the text for the third list action line is formatted, the text for the first two lines is not moved, but the UIM tries to align the text on the third line with the text on the second line.

PARMS=*dialog-variable-name*

This attribute must name a CHAR 255 dialog variable, used by the UIM to store parameter information for action list processing. This attribute is allowed only for display panels and is only valid when the ACTOR attribute on this tag has a value other than NONE. It must be used when the list area operates in conjunction with a command line on the same panel. This provides a way for the user to specify parameters that affect action list processing.

The UIM stores the contents of the command line in this variable when the command line contains parameters for action list processing. The UIM sets the variable to blanks before action list processing when the command line does not contain parameter information.

This dialog variable is intended for use in the action string for a CMD dialog command specified on the `ENTER`, `EXTENTER`, `PROMPT`, or `EXTPROMPT` attribute on the LISTACT tag.

Note: This variable only contains command line contents when the command line is interpreted as specifying parameters for action list processing. It does not provide generalized access to the contents of the command line.

HEADSIZE=*dialog-variable-name*

This dialog variable specifies the number of dialog variables specified on the `COLHEAD` attribute of the LISTCOL tag that should be used for the list column headings. This attribute is not allowed if `MAXHEAD` is specified.

This dialog variable can only contain the values 0 through 10, as these are the valid number of lines for heading text. If this dialog variable is not a valid value from 0 through 10, the maximum of 10 is used. When using the dialog variables from the list specified on the `COLHEAD` attribute of the LISTCOL tag, the variables are used in the order defined on the `COLHEAD` attribute. This dialog variable must be defined with a `BASETYPE` of 'BIN 15' on the class definition (CLASS) tag.

The dialog variables on the `HEADSIZE` attribute and the `COLHEAD` attribute of the LISTCOL tag are evaluated like normal dialog variables. It is recommend that you set the *HEADSIZE* dialog variable before calling the QUIDSPP API and not change it when changing from one view to another. It is up to the application programmer to not let the list and headings shift up and down on panels.

This attribute may not be specified if the list area contains list column groups.

Optional Text

area-title

The title of the area. If no text is specified, no title line is allocated to the area. The text must appear on the same or next line as the tag, can only contain the reverse text (RT) tag, and cannot exceed a maximum length of 55 characters.

Print Formatting Considerations

Printed list areas are formatted like those displayed with the following changes:

- Printing always starts with the first entry in the list.
- For lists which are incomplete at the top, the UIM starts printing with the first entry in the list and does not require the list to be marked complete at the top.
- For lists which are incomplete at the bottom, the UIM formats and prints until it runs out of list entries, then calls the exit program for processing the incomplete list for more entries. In general, the exit program is asked for a large number of list entries, and if the list is not marked complete at the bottom, the exit program is called again for more list entries.
- For layouts greater than one, the UIM formats as many entries as can be printed on one page, then prints that page. If there are not enough entries to fill up a page, the entries are balanced across the layout columns. There is always a minimum of two entries in the first column before putting entries into subsequent columns. This helps the user understand how to read the list. For example, a list with `LAYOUT=2` and only two list entries would print as follows:

```
Column 1          Column 2
xxxxxxxx
xxxxxxxx
```

For list panels, there must be room for a minimum of two lines of list entries on a page besides the column headings. If two lines of list entries do not fit on a page, a page eject occurs and the list entries are printed on the next page. The following example would need to fit on one page, or a new page would be used.

```
Column 1          Column 2
xxxxxxxx         xxxxxxxxxx
xxxxxxxx         xxxxxxxxxx
```

List column headings and group headings are repeated on each page if the list area is continued onto another page.

Example 1: List Area

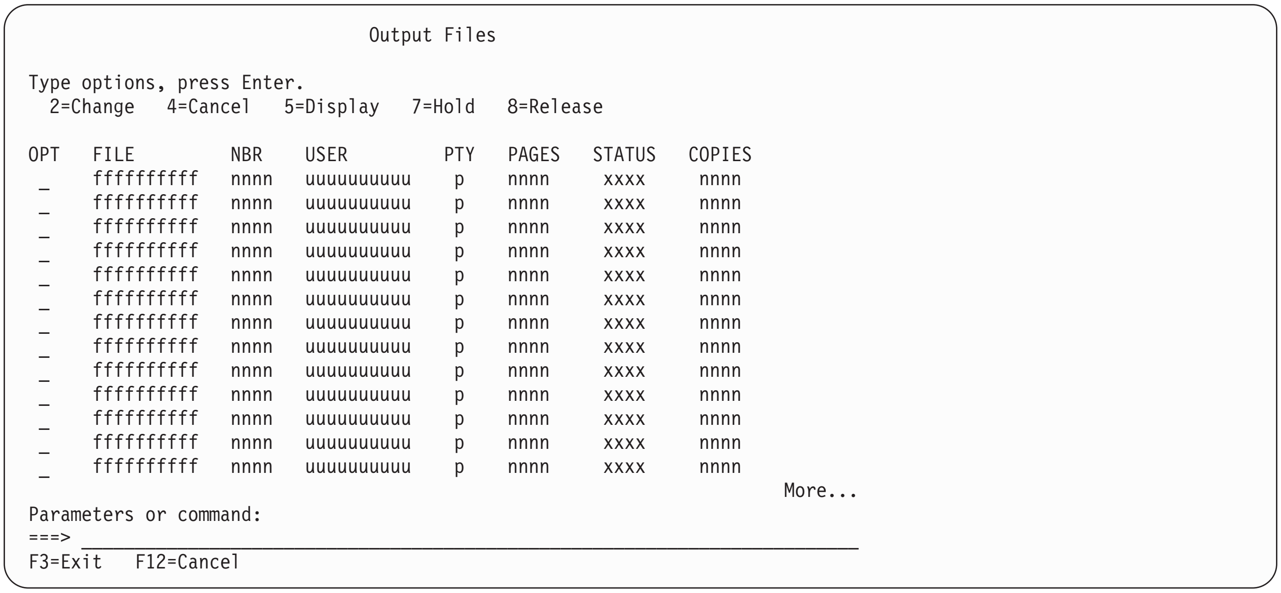
This example shows an action where the UIM performs the action requested by the user.

UIM Source

```
:listdef name=outflist
         vars='opt fil nbr usr pri pg sts co'.
:panel topsep=space
         ...
         .Output Files
:list depth='*'
         listdef=outflist
         maxhead=2
         actor=uim
         parms=pvar.
:topinst.Type options, press Enter.
:listact
  enter='CMD CHGSPLFA FILE(&FIL)'
  enter='JOB(&USR/&SPID) SPLNBR(&NBR) &PVAR'
  help=opt2
  option=2.2=Change
:listact
  enter='CMD CNLSPLF FILE(&FIL)'
  enter='JOB(&USR/&SPID) SPLNBR(&NBR)'
  help=opt4
  option=4.4=Cancel
:listact
  enter='CMD DSPSPLF FILE(&FIL)'
  enter='JOB(&USR/&SPID) SPLNBR(&NBR)'
  help=opt5
  option=5.5=Display
:listact
  enter='CMD HLDSPLF FILE(&FIL)'
  enter='JOB(&USR/&SPID) SPLNBR(&NBR)'
  help=opt7
  option=7.7=Hold
:listact
  enter='CMD RLSSPLF FILE(&FIL)'
  enter='JOB(&USR/&SPID) SPLNBR(&NBR)'
  help=opt8
  option=8.8=Release
:listcol var=opt
        usage=inout
        maxwidth=6
        help=hopt.Opt
:listcol var=fil
        usage=out
        maxwidth=10
        help=hfil.File
:listcol var=nbr
        usage=out
        maxwidth=6
        help=hnbr.Nbr
:listcol var=usr
```

```
        usage=out
        maxwidth=10
        help=huser.User
:listcol var=pri
        usage=out
        maxwidth=6
        help=hpri.Pty
:listcol var=pg
        usage=out
        maxwidth=8
        help=hpg.Pages
:listcol var=sts
        usage=out
        maxwidth=10
        help=hsts.Status
:listcol var=co
        usage=out
        maxwidth=10
        help=hco.Copies
:listview layout=1
        cols='opt fil nbr usr pri pg sts co'.
:elist.
:cmdline size=short.Parameters or command:
:epanel.
```

Results



```
                             Output Files

 Type options, press Enter.
   2=Change   4=Cancel   5=Display   7=Hold   8=Release

 OPT   FILE         NBR    USER         PTY   PAGES   STATUS   COPIES
  _    ffffffffff   nnnn   uuuuuuuuuu    p    nnnn     xxxx     nnnn
  _    ffffffffff   nnnn   uuuuuuuuuu    p    nnnn     xxxx     nnnn
  _    ffffffffff   nnnn   uuuuuuuuuu    p    nnnn     xxxx     nnnn
  _    ffffffffff   nnnn   uuuuuuuuuu    p    nnnn     xxxx     nnnn
  _    ffffffffff   nnnn   uuuuuuuuuu    p    nnnn     xxxx     nnnn
  _    ffffffffff   nnnn   uuuuuuuuuu    p    nnnn     xxxx     nnnn
  _    ffffffffff   nnnn   uuuuuuuuuu    p    nnnn     xxxx     nnnn
  _    ffffffffff   nnnn   uuuuuuuuuu    p    nnnn     xxxx     nnnn
  _    ffffffffff   nnnn   uuuuuuuuuu    p    nnnn     xxxx     nnnn
  _    ffffffffff   nnnn   uuuuuuuuuu    p    nnnn     xxxx     nnnn
  _    ffffffffff   nnnn   uuuuuuuuuu    p    nnnn     xxxx     nnnn
  _    ffffffffff   nnnn   uuuuuuuuuu    p    nnnn     xxxx     nnnn
  _    ffffffffff   nnnn   uuuuuuuuuu    p    nnnn     xxxx     nnnn
                                                                         More...
 Parameters or command:
 ===> _________________________________________________________________________
 F3=Exit   F12=Cancel
```

Example 2: List Area with Three Layout Columns

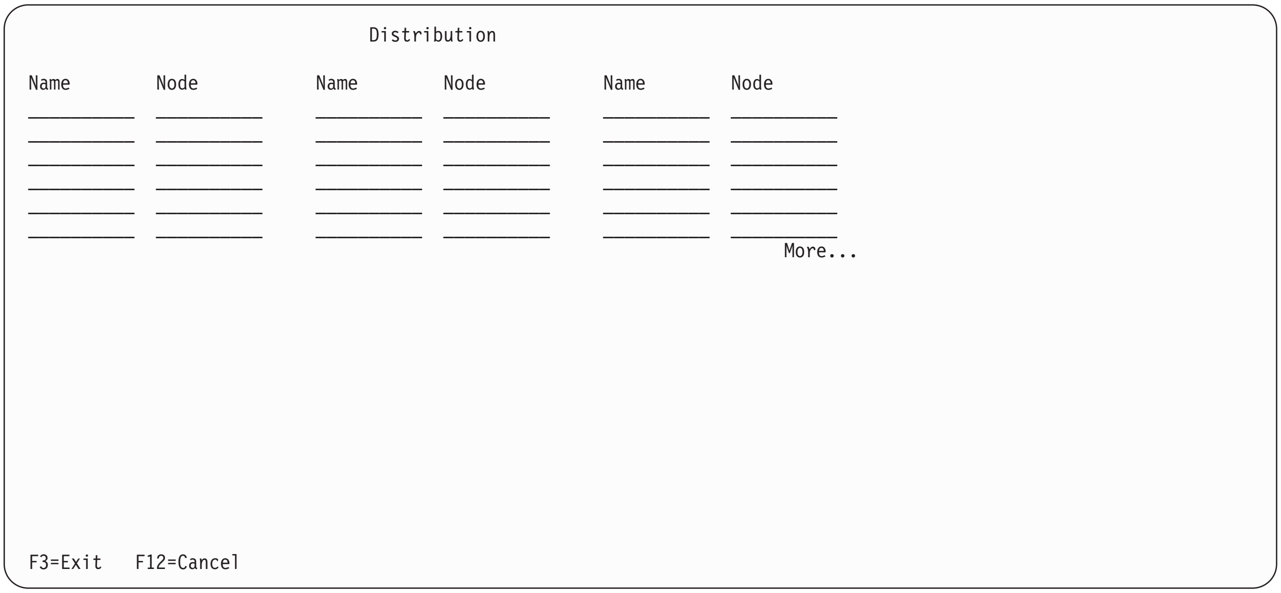
This example shows a list area where three entries appear in each row of the display.

UIM Source

```
:listdef name=dist
         vars='name node'.
:panel panel-attributes.Distribution
:list listdef=dist
         depth=8
         maxhead=1.
:listcol var=name
         usage=inout
         maxwidth=10
         help=hxxyy1.Name
:listcol var=node
```

```
         usage=inout
         maxwidth=10
         help=hxxyy2.Node
:listview layout=3
         cols='name node'.
:elist.
:epanel.
```

Results



```
                                Distribution

Name        Node           Name        Node           Name        Node
__________  __________     __________  __________     __________  __________
__________  __________     __________  __________     __________  __________
__________  __________     __________  __________     __________  __________
__________  __________     __________  __________     __________  __________
__________  __________     __________  __________     __________  __________
                                                                       More...











F3=Exit   F12=Cancel
```

Example 3: List Area with List Column Groups

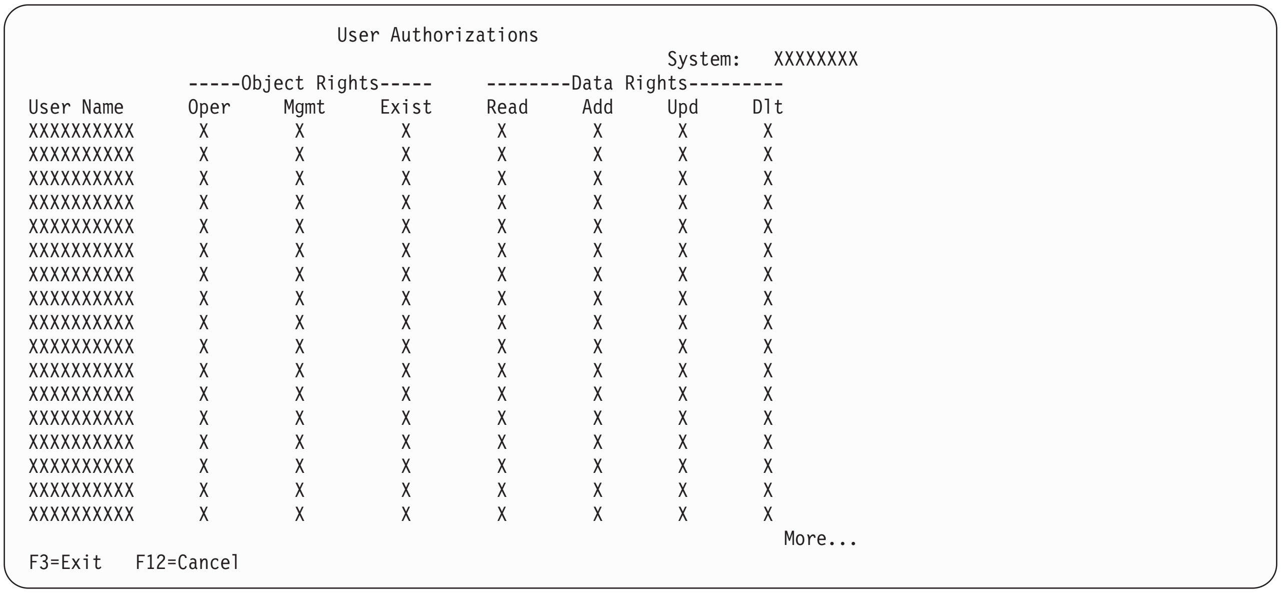
This example shows how list columns can be grouped together.

UIM Source

```
:listdef name=auth
         vars='user oper mgmt exist'
         vars='read add update delete'.
:panel panel-attributes.User Authorizations
:list listdef=auth
         depth='*'
         maxhead=3.
:listcol var=user
         usage=out
         maxwidth=10
         help=huser
         .User Name
:listgrp col=objrights
         help=hobjaut.Object Rights
:listcol var=oper
         usage=out
         maxwidth=7
         .Oper
:listcol var=mgmt
         usage=out
         maxwidth=7
         .Mgmt
:listcol var=exist
         usage=out maxwidth=7
         .Exist
:elistgrp.
:listgrp col=dtarights
         help=hdtaaut.Data Rights
```

```
:listcol var=read
         usage=out
         maxwidth=7
         .Read
:listcol var=add
         usage=out
         maxwidth=7
         .Add
:listcol var=update
         usage=out
         maxwidth=7
         .Upd
:listcol var=delete
         usage=out
         maxwidth=7
         .Dlt
:elistgrp.
:listview layout=1
         cols='user objrights dtarights'
:elist.
:epanel.
```

Results



```
                             User Authorizations
                                                            System:   XXXXXXXX
               -----Object Rights-----     --------Data Rights---------
User Name      Oper     Mgmt     Exist     Read     Add     Upd     Dlt
XXXXXXXXXX      X        X         X        X        X       X       X
XXXXXXXXXX      X        X         X        X        X       X       X
XXXXXXXXXX      X        X         X        X        X       X       X
XXXXXXXXXX      X        X         X        X        X       X       X
XXXXXXXXXX      X        X         X        X        X       X       X
XXXXXXXXXX      X        X         X        X        X       X       X
XXXXXXXXXX      X        X         X        X        X       X       X
XXXXXXXXXX      X        X         X        X        X       X       X
XXXXXXXXXX      X        X         X        X        X       X       X
XXXXXXXXXX      X        X         X        X        X       X       X
XXXXXXXXXX      X        X         X        X        X       X       X
XXXXXXXXXX      X        X         X        X        X       X       X
XXXXXXXXXX      X        X         X        X        X       X       X
XXXXXXXXXX      X        X         X        X        X       X       X
XXXXXXXXXX      X        X         X        X        X       X       X
XXXXXXXXXX      X        X         X        X        X       X       X
XXXXXXXXXX      X        X         X        X        X       X       X
                                                                       More...
F3=Exit   F12=Cancel
```

Example 4: Dynamic List Column Heading Formatting

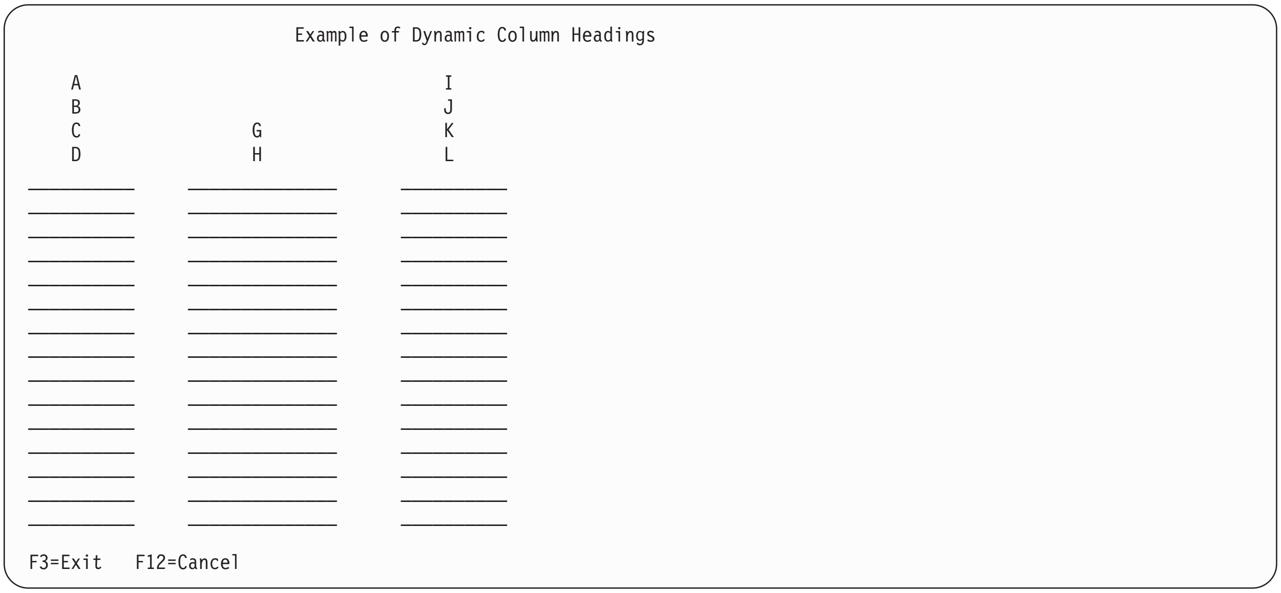
This example shows how the column headings are contained in dialog variables.

UIM Source

```
:listdef name=xmp4
         vars='var1 var2 var3'.
:panel panel-attributes.
Example of Dynamic Column Headings
:list listdef=xmp4
         depth='*'
         headsize=colhsize.
:listcol var=var1
         usage=inout
         maxwidth=10
         colhead='A B C D E F'.
:listcol var=var2
         usage=inout
         maxwidth=14
         colhead='G H'.
:listcol var=var3
         usage=inout
         maxwidth=10
         colhead='I J K L'.
:listview layout=1
         cols='var1 var2 var3'.
:elist.
:epanel.
```

Note: In this example, A B C D E F G H I J K and L are dialog variables that would each be set to one line of column heading text. In the screen that follows, the actual text appears instead of the dialog variable names. *COLHSIZE* is also a dialog variable, and in this example it is set equal to 4.

Results



```
                         Example of Dynamic Column Headings

    A                                  I
    B                                  J
    C                G                 K
    D                H                 L
__________     ______________      __________
__________     ______________      __________
__________     ______________      __________
__________     ______________      __________
__________     ______________      __________
__________     ______________      __________
__________     ______________      __________
__________     ______________      __________
__________     ______________      __________
__________     ______________      __________
__________     ______________      __________
__________     ______________      __________
__________     ______________      __________
__________     ______________      __________
__________     ______________      __________

F3=Exit   F12=Cancel
```

LISTACT (List Action)

```
►►─:LISTACT──HELP──=──help-module-name──OPTION──=──number──┬──────────────────────────────┬──────────►
                                                            │                 ┌─BOTH────┐  │
                                                            └─ACTFOR──=──────┼─LISTE───┼──┘
                                                                              └─EXTACTE─┘

►─┬─────────────────────────┬─┬───────────────────────────────┬─┬──────────────────────────────────┬─►
  └─CONFIRM──=──panel-name──┘ └─ENTER──=──' ──action-text──' ─┘ └─EXTENTER──=──' ──action-text──' ─┘

►─┬────────────────────────────────┬─┬──────────────────────────────────┬─┬──────────────────────────┬─►
  └─PROMPT──=──' ──action-text──' ─┘ └─EXTPROMPT──=──' ──action-text──' ─┘ │              ┌─ENTER──┐  │
                                                                          └─NOCMD──=──────┴─PROMPT─┴──┘

►─┬────────────────────────┬─┬──────────────────────────────────────────────────┬─────────────────────►
  │            ┌─ENTER──┐  │ └─USREXIT──=──' ──CALL── ──program-reference──' ─┘
  └─NOEXT──=──┼─PROMPT─┼──┘
               └─MSG────┘

►─┬───────────────────────────────────────┬─┬─────────────────────────────────────────────────┬─────►
  └─EXTMSGID──=──message-identifier───────┘ └─EXTMSGF──=──' ──qualified-message-file-name──' ─┘

►─┬────────────────────────────┬─┬─────────────────────────────┬─┬───────────────────────────────────┬─►
  └─COND──=──condition-name────┘ └─AVAIL──=──condition-name────┘ └─AVLMSGID──=──message-identifier────┘

►─┬─────────────────────────────────────────────────┬─.─┬──────────────────────┬─────────────────►◄
  └─AVLMSGF──=──' ──qualified-message-file-name──' ─┘   └─action-description───┘
```

The list action (LISTACT) tag defines an operation which takes place on individual items of a list. This tag is allowed only for display panels. Options entered by a user are not actually run until an Enter or Prompt function key is pressed, allowing the user to select several list items while using the Page Up or Page Down keys.

If ACTOR=UIM is specified on the list area (LIST) tag, all actions indicated by LISTACT tags are performed by the UIM. These actions must have the ENTER attribute specified, and may also have the PROMPT and USREXIT attributes specified. If ACTOR=CALLER is specified on the LIST tag, the actions are handled by the program which called the Display Panel (QUIDSPP) API. In this case, none of the following attributes can be specified on the LISTACT tag:

- CONFIRM
- ENTER
- PROMPT
- USREXIT
- EXTENTER
- EXTPROMPT
- NOEXT
- EXTMSGID
- EXTMSGF

There is no way to have the UIM perform some actions and the calling program perform others.

When the CMD dialog command is used as the action performed for the extended action entry in an action list, the command text should be a Control Language command, not a System/36 Environment OCL command. The value *N is substituted in the command text for any dialog variable from the extended action entry that has a blank value. An exception to this is when NOEXT=MSG is specified on this tag, indicating that the command should not be submitted when any of the values for the extended action entry are blank. This consideration applies to the following attributes of this tag:

- ENTER
- PROMPT
- EXTENTER
- EXTPROMPT

Required Attributes

HELP=*help-module-name*

Identifies the associated help module for this list action. The online help information for all actively-conditioned list actions is always added to the online help information for the action column.

OPTION=*number*

The value associated with the option selected for this item in the list. The value must be an integer value in the range of 1 through 999. The action field variable must be defined with a BASETYPE of ACTION on the class definition (CLASS) tag. The values of the action field are automatically limited to the valid options by the UIM. An error occurs when the user enters a number in the action field for which there is no active LISTACT tag.

Optional Attributes

ACTFOR=BOTH | LISTE | EXTACTE

Indicates whether the list action defined is allowed for the existing list entries, the extended action entry, or both. The default is BOTH.

If LISTE is specified, the action defined is valid only in the action fields associated with the existing list entries on the display.

EXTACTE indicates that the action defined is valid only in the action field for the extended action entry. For example, ACTFOR=EXTACTE might be used for Option 1=Create, because the create option is not allowed for an existing entry.

CONFIRM=*panel-name*

The confirmation panel displayed before the list action is performed. The confirmation panel must give the user the option to confirm or not confirm the list action. If the action is confirmed, the appropriate actions from the ENTER, EXTENTER, and USREXIT attributes of this tag are performed. If the action is not confirmed, none of those actions are performed.

The actions coded on the PROMPT and EXTPROMPT attributes of this tag are not confirmed. The prompt screen that results from the PROMPT and EXTPROMPT actions serves as the confirmation panel, allowing users to change their minds and cancel the action. The confirmation panel must be another panel defined within this panel group.

The CONFIRM attribute is only allowed when ACTOR=UIM is specified in the LIST tag.

When the CONFIRM attribute is specified, EXTACT=YES is specified on the LIST tag and if ACTFOR=BOTH or ACTFOR=EXTACTE is specified on this tag, NOEXT=MSG must also be specified on this tag.

A maximum of 20 different LISTACT tags for a single action list may specify the CONFIRM attribute. This does not impose any restriction on the number of list entries that any one of the actions may be applied to.

For more information on required and recommended conventions for this attribute, see “Confirmation Panel Requirements” on page 566 and “Confirmation Panel Conventions” on page 567.

ENTER='*action-text***'**

Specifies the action occurring when the list action is requested through the ENTER dialog command. This attribute is required if ACTOR=UIM is specified on the LIST tag. The valid forms of action-text are:

- *′CALL program-reference′* For a description of the interface between the UIM and the exit program for an action list option, see the **APIs** topic in the iSeries Information Center.
- *′CMD command-string′* Any dialog variable name in the command string must be preceded by an ampersand and should be ended with a period to denote variable substitution.

For a description of each of these actions, see Appendix B, “UIM Dialog Commands,” on page 641.

This attribute is repeatable, allowing the construction of long command strings.

The PARMS attribute on the LIST tag identifies a dialog variable that can be substituted into a command string on the CMD dialog command. Additional functions are provided for list actions which use the CMD dialog command. When parameters from the command line are substituted into a command string, the parameters from the command line override the same parameter defined in the command string if the parameter in the command string is preceded by the ″?<″ or ″??″ selective prompting characters.

This function allows a prompt for a command to show the parameter values from the command line, or to show the current parameter values for the object when an overriding parameter is not on the command line. When the ″?<″ or ″??″ selective prompting characters are used, it is your responsibility to have the current parameter values substituted into the command string.

EXTENTER='*action-text***'**
Specifies the action occurring when the list action for the extended action field is requested through the ENTER dialog command. This attribute is not allowed when ACTFOR=LISTE is specified on the LISTACT tag, or when EXTACT=NO is specified on the LIST tag.

The syntax and usage of this attribute is the same as the ENTER attribute for this tag. If the EXTENTER attribute is not specified, the action specified on the ENTER attribute is used.

PROMPT='*action-text***'**
Specifies the action occurring when the list action is requested through the PROMPT dialog command. The valid forms of action text are:

- *′CALL program-reference′* For a description of the interface between the UIM and the exit program for an action list option, see the **APIs** topic in the iSeries Information Center.
- *′CMD command-string′* Any dialog variable name in the command string must be preceded by an ampersand to denote variable substitution.

For a description of these actions, see Appendix B, “UIM Dialog Commands,” on page 641.

This attribute is repeatable, allowing the construction of long command strings.

The PARMS attribute on the LIST tag identifies a dialog variable that can be substituted into a command string on the CMD dialog command. Additional command submission support is activated for list actions which use the CMD dialog command. When parameters from the command line are substituted into a command string, the parameters from the command line override the same parameter defined in the command string if the parameter in the command string is preceded by the ″?<″ or ″??″ selective prompting characters.

This function allows a prompt for a command to show the parameter values from the command line, or to show the current parameter values for the object when an overriding parameter is not on the command line. When the ″?<″ or ″??″ selective prompting characters are used, it is your responsibility to have the current parameter values substituted into the command string.

EXTPROMPT='*action-text***'**
Specifies the action occurring when the list action for the extended action field is requested through the PROMPT dialog command. The EXTPROMPT attribute is not allowed when ACTFOR=LISTE is specified on the LISTACT tag, when EXTACT=NO is specified on the LIST tag, or when the PROMPT attribute is not specified on this tag.

The syntax and use of this attribute is the same as for the PROMPT attribute on this tag. If the EXTPROMPT attribute is not specified, the action specified on the PROMPT attribute is used.

NOCMD=ENTER | PROMPT

Specifies the action the UIM performs for the ENTER dialog command when the command line is blank.

ENTER indicates that the action on the ENTER attribute of this tag should be performed.

PROMPT indicates that the action on the PROMPT attribute of this tag should be performed.

This allows you to have a command prompted when the command line is empty, but not prompted if the command line contains parameters.

For an explanation of how this attribute interacts with the NOEXT attribute of this tag, see Table 64.

NOEXT=ENTER | PROMPT | MSG

Specifies the action the UIM should perform for the ENTER dialog command when one or more of the input fields on the extended action entry are blank.

ENTER indicates that the action on the EXTENTER attribute of this tag should be performed.

PROMPT indicates that the action on the EXTPROMPT attribute of this tag should be performed.

MSG indicates that no action should be performed and that the message specified on the EXTMSGID attribute of this tag should be sent to the user. When MSG is specified, the EXTMSGID attribute is required.

For an explanation of how this attribute interacts with the NOCMD attribute, see Table 64. This table shows how the NOCMD and NOEXT attributes operate when the command line is blank and one or more of the input fields on the extended action entry are blank.

Table 64. NOCMD and NOEXT Attribute Interaction

NOCMD	NOEXT	Result
ENTER	ENTER	EXTENTER attribute is used
PROMPT	ENTER	EXTPROMPT attribute is used
ENTER	PROMPT	EXTPROMPT attribute is used
PROMPT	PROMPT	EXTPROMPT attribute is used
ENTER	MSG	EXTMSGID message is displayed
PROMPT	MSG	EXTMSGID message is displayed

This table is used only when all of the following are true:

1. A list action option is being processed for the extended action entry.
2. The command line is blank.
3. One or more of the input fields on the extended action entry are blank.

USREXIT='*CALL program-reference***'**

Specifies the list exit program the UIM calls to update or delete this list entry after the action defined in either the ENTER, EXTENTER, PROMPT, or EXTPROMPT attribute of this tag is performed. The program is passed information that includes an indication of whether the option succeeded or failed.

For a description of the CALL dialog command, see Appendix B, “UIM Dialog Commands,” on page 641.

For a description of the interface between the UIM and the program for an action list, see the **APIs** topic in the iSeries Information Center.

EXTMSGID='*message-identifier***'**

The message identifier of the message displayed when the action defined by this tag cannot be performed because one or more input fields on the extended action entry are blank. This attribute is required when NOEXT=MSG is specified on this tag; it is not allowed otherwise.

EXTMSGF='*qualified-message-file-name***'**
The message file name that contains the message identifier. The attribute is allowed only when the EXTMSGID attribute is specified on this tag. If the DFTMSGF attribute is not specified on the panel group (PNLGRP) tag, this attribute must be specified when the EXTMSGID attribute on this tag is specified.

COND=*condition-name*
The list action is in effect on the panel only if the condition specified is true. The condition must be defined in the panel group prolog with the condition definition (COND) tag. If the conditions for all LISTACT tags are false, the action column still appears on the display, but no option numbers are valid. It is up to you to ensure that some options remain valid for the list.

AVAIL=*condition-name*
The list action is available only if the condition specified evaluates to true. The condition must be defined in the panel group prolog with the COND tag.

Unavailable actions are displayed with an asterisk (*) overlaying the first part of the action description. Online help information is displayed for unavailable list actions.

When the condition is true, the list action is available. When the condition is false, the list action is not available. Any condition specified on the COND attribute takes precedence over this attribute.

AVLMSGID=*message-identifier*
The message identifier of the message displayed when a number is entered and is not available as specified by the AVAIL attribute of this tag. This attribute is allowed only when the AVAIL attribute is specified.

If this attribute is not specified, the UIM displays a default message stating that the option number is not currently available.

AVLMSGF=*'qualified-message-file-name'*
The message file name that contains the message identifier. The attribute is allowed only when the AVLMSGID attribute of this tag is specified. If the DFTMSGF attribute is not specified on the PNLGRP tag, this attribute must be specified when the AVLMSGID attribute on this tag is specified.

Optional Text

action-description
The text shown as part of the top instructions for the list area. If no text is supplied, none is listed in the instruction part of the list area for this action. The text must appear on the same or next line as the tag and can only contain the reverse text (RT) tag.

The instruction line is formed by concatenating the text from the various list actions, separated by three spaces. If the text does not fit on the rest of a line, it is placed on the next line.

Confirmation Panel Requirements

The panel named on the CONFIRM attribute of this tag must follow these conventions:

- The ENTER attribute on the display panel (PANEL) tag of the confirmation panel must have 'RETURN 100' coded. This indicates to the UIM that the action is confirmed by the user.
- No ACTION attribute on a key list item (KEYI) tag can be assigned to the RETURN dialog command, because the UIM is handling the display of the panel and is not able to interpret the returned value.
- The confirmation panel must have a list area.
- The list used as the confirmation list must not be the same list displayed in the action list panel.
- The confirmation panel must not have a menu bar, action list, command line, or menu area.

Confirmation Panel Conventions

The panel named in the CONFIRM attribute of this tag should follow these conventions:

- It should have title text and top instructions to describe the panel.
- It should have a single list area with a list definition containing the same (or a subset of) the columns defined in the original action list.
- The list should be scrollable.
- If the action list has multiple views, the confirmation list should also have multiple views and the same dialog variable should be used for the view variable on both lists. The confirmation panel should have a function key defined for the CHGVIEW dialog command. When the same view variable is used for both lists, it allows a change view operation on the confirmation panel to change the view of the action list panel when it is redisplayed.
- It should have function key definitions for the ENTER, CANCEL, PAGEDOWN, PAGEUP, HELP, and PRINT dialog commands.
- If a general exit program is defined, it should avoid making changes to the action list and avoid changing condition values that affect the redisplay of the original action list panel.
- The list used on the confirmation panel should not be used on anything but a confirmation panel, because the UIM deletes and changes the contents of that list. If multiple confirmation panels are used within a UIM application, they use the same list definition.
- Most of the help information for the action list panel can be used for the confirmation panel, but separate help information should be provided for the action column and the extended help.
- The class definition for the list option variable defined with a BASETYPE of ACTION on the CLASS tag should have a WIDTH attribute on the CLASS tag to accommodate the largest list action option on the action list panel. For example, if the largest option number on the action list panel is 12, WIDTH=2 should be specified on the class definition for the action variable. This ensures that the option number is displayed in the same position under the column heading on the confirmation panel as it was on the action list panel.
- Only the values for the input capable fields of the extended list action entry will be displayed on the confirmation panel. All other columns will be shown as blanks even though there may be a value there.

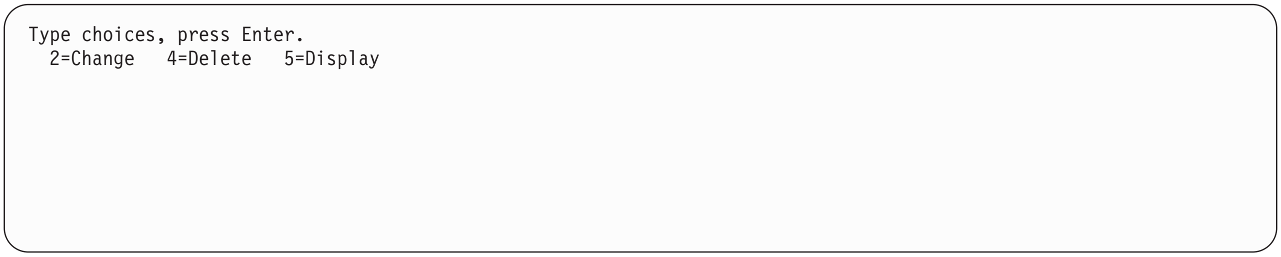
Example: List Actions

This example shows how the action options on a display are defined.

UIM Source

```
:listact option=2
         help='option/2'
         enter='CALL CHGPGM'
         .2=Change
:listact option=4
         help='option/4'
         enter='CALL DLTPGM'
         confirm=CONFDLT
         .4=Delete
:listact option=5
         help='option/5'
         enter='CALL DSPPGM'
         .5=Display
```

Results



```
Type choices, press Enter.
  2=Change   4=Delete   5=Display
```

LISTCOL (List Column)

Syntax for Display Panels:

```
►►──:LISTCOL──VAR──=──list-object-variable-name──────────────────────────►

►─MAXWIDTH──=──┬─column-width─┬──USAGE──=──┬─OUT───┬─────────────────────►
               └─'*'──────────┘            └─INOUT─┘

►─┬─────────────────────────────┬─┬──────────────────────────────┬───────►
  └─HELP──=──help-module-name───┘ └─COL──=──column-identifier────┘

►─┬─────────────────────────────┬─┬───────────────────────────┬──────────►
  └─NAME──=──list-column-name───┘ └─JUSTIFY──=──┬─LEFT──┬─────┘
                                                ├─RIGHT─┤
                                                ├─START─┤
                                                └─END───┘

►─┬──────────────────────────┬─┬─────────────────────────────────┬───────►
  │              ┌─NO──┐     │ └─PROMPT──=──'──action-text──'────┘
  └─EXTACT──=────┴─YES─┴─────┘

►─┬──────────────────────────────────────┬───────────────────────────────►
  └─DSPVALUE──=──dialog-variable-name────┘

►─┬────────────────────────────────────────────────┬─┬──────────────────────┬──►
  └─COLHEAD──=──'──dialog-variable-name-list──'────┘ │               ┌─YES─┐│
                                                     └─DISPLAY──=────┴─NO──┴┘

►─┬────────────────────────────────────────────────┬─────────────────────►◄
  │                ┌─YES─┐                         │
  └─AUTOSKIP──=────┴─NO──┴──.──┬────────────────┬──┘
                               └─column-heading─┘
```

Syntax for Print Panels:

```
►►──:LISTCOL──VAR──=──list-object-variable-name──────────────────────────────►

►─MAXWIDTH──=──┬─column-width─┬──┬─────────────────┬──────────────────────────►
               └─'*'──────────┘  └─USAGE──=──OUT─┘

►─┬───────────────────────────┬──┬──────────────────────────┬────────────────►
  └─COL──=──column-identifier─┘  └─JUSTIFY──=──┬─LEFT──┬─┘
                                               ├─RIGHT─┤
                                               ├─START─┤
                                               └─END───┘

►─┬──────────────────────────────────────────────┬─.─┬────────────────┬──────►◄
  └─COLHEAD──=──'──dialog-variable-name-list──'─┘   └─column-heading─┘
```

The list column (LISTCOL) tag defines a column that may belong to any of several list views. This tag is allowed for display panels and print panels. It specifies the dialog variable, whether the field allows the user to enter data or is for output only, the column width, and the column heading.

Required Attributes

VAR=*list-object-variable-name*
The name of the variable from the underlying list object to display in this column.

MAXWIDTH=*column-width* **|** **'*'**
The maximum width of the column. The width may allow for translation of the column heading into other languages. The minimum size for a column heading is four characters. Specification of a width longer than that defined in the class definition (CLASS) tag allows more room for the formatting of column headings, but does not allow more data to be entered into the field. Specification of a width shorter than the class width on the CLASS tag is allowed only for CHAR, IGC, or certain TIME variables, and is not allowed for input columns.

Only the time zone portion of a TIME variable can be truncated. So the ZONE option must have been specified on the BASETYPE of the class definition for the TIME variable and the width specified for the LISTCOL must be greater than or equal to 8.

If '*' is coded, the remainder of the area width is used for the column, and values in that column can be truncated. Only one column may have '*' coded. A warning message is put in the compiler listing if the column with '*' specified is not the last column of all views. A two- to five-character separator is maintained between columns.

USAGE=OUT | INOUT
The display use of the column. This attribute is required for display panels, but is optional for print panels.

USAGE=OUT defines an output data column. OUT indicates that the variable displayed is for output only and cannot be changed by the user.

USAGE=INOUT defines a data entry column. INOUT indicates that the variable is for data entry and can be changed by the user. USAGE=INOUT fields are not allowed for print panels, but are required for the action column of an active list.

Optional Attributes

HELP=*help-module-name*
Identifies online help information which explains the purpose of the column in this list. This attribute is allowed only for display panels. The help module name may be a name imported from another

panel group, but must follow the rules for names outlined earlier in this chapter. For more information on the rules for names, see “Name Syntax” on page 469.

The HELP attribute is not allowed if this column is part of a list column group defined by the list group (LISTGRP) tag, but is required if the column is not part of a group.

COL=*column-identifier*

The identifier of the column, which can be referred to in the COLS attribute on the list view (LISTVIEW) tag. If the identifier is not supplied, it defaults to the dialog variable name specified on the VAR attribute of this tag. These names are in a separate name space from all other names and are meaningful only within the list area, not to the panel group.

A list column identifier cannot be specified and no default column name is assumed if the column is part of a list column group defined by the LISTGRP tag.

NAME=*list-column-name*

The name associated with the column. For more information on the rules for naming, see “Name Syntax” on page 469.

This name can be used with the Add Pop-Up Window (QUIADDPW) API to position a window near this list column. This attribute is allowed only for display panels.

If the list column is part of a list column group defined by the LISTGRP tag and both the LISTCOL and LISTGRP tags have the NAME attribute specified, the list column name is set in the dialog variable specified on the CSRNAME attribute of the display panel (PANEL) tag when the cursor is anywhere in the list column. The list group name is set for the CSRNAME attribute on the PANEL tag when the cursor is positioned between columns or on an unnamed list column within the named list group. For a description of the CSRNAME attribute, see “PANEL (Display Panel)” on page 595.

JUSTIFY=LEFT | RIGHT | START | END

Indicates whether the dialog variable values should be left-justified or right-justified within the list column. If the ALIGN attribute of this tag is omitted, the default right-justifies values for numeric dialog variables and left-justifies nonnumeric dialog variables.

Leading blanks are preserved when the dialog variable is left-justified. Right-justification strips all trailing blanks. The UIM rules for editing a numeric dialog variable produces a displayable value which does not contain any leading blanks.

A common case where you might want to override the alignment default is for a column of values whose BASETYPE is defined as numeric on the CLASS tag, but whose displayable values are sometimes special values generated through a translation list. For more information on translation lists, see “TL (Translation List)” on page 629.

START is a synonym for LEFT and END is a synonym for RIGHT.

EXTACT=NO | YES

Specifies whether or not this list column is input-capable for the extended action entry. This attribute is allowed only for display panels.

EXTACT=NO specifies that this list column does not have an input-capable field in the extended action entry.

EXTACT=YES specifies that this list column does have an input-capable field in the extended action entry. When EXTACT=YES is specified on the list area (LIST) tag, EXTACT=YES must be specified on at least two columns in each list view, including the action column.

Any column with EXTACT=YES specified must be in every view of the list area.

PROMPT='*action-text***'**

The action occurring when F4=List is requested through the PROMPT dialog command. This attribute is allowed only for display panels. This attribute is not allowed for a dialog variable defined with a BASETYPE of ACTION on the CLASS tag or for any USAGE=OUT list columns on this tag. The valid forms of action text are:

- *'CALL program-reference'* For a description of the interface between the UIM and the exit program for a cursor-sensitive prompt, see the **APIs** topic in the iSeries Information Center.
- *'RETURN positive-integer'*

For a description of these actions, see Appendix B, "UIM Dialog Commands," on page 641.

DSPVALUE='*dialog-variable-name***'**
The dialog variable containing the current data entered in the list entry which contains the cursor. This attribute is allowed only for display panels and when the PROMPT attribute is coded on this tag. This variable is updated regardless of whether or not the variable pool is updated. The variable pool is updated based on the VARUPD attribute of the key item (KEYI) or pull-down field choice (PDFLDC) tag. This updating is independent of the normal updating of the variable pool.

The dialog variable specified must be defined on the CLASS tag as a CHAR or IGC variable whose length is the same as the width of the dialog variable specified on the VAR attribute of this tag. No translation processing or validity checking is performed for the value before it is placed in this variable. Character set and code page conversion is performed for this variable if the class of the variable named on the VAR attribute of this tag specifies that the character set and code page conversion should be performed.

COLHEAD='*dialog-variable-name-list***'**
A list of dialog variables separated by blanks, specifying the column heading placed above the list items in the list area. Up to ten dialog variables can be specified. These dialog variables must have a BASETYPE of CHAR or IGC specified on the CLASS tag, and must have a width greater than or equal to the class width of the dialog variable specified on the VAR attribute of this tag, and less than or equal to the width specified on the MAXWIDTH attribute of this tag. All variables specified on a single COLHEAD attribute must be defined with the identical class.

Each dialog variable is used as a line of column heading text. The UIM does no aligning or justification on the contents of these dialog variables.

If this attribute is used, column heading text cannot be specified after the period for this tag, the MAXHEAD attribute must not be specified on the LIST tag, but the HEADSIZE attribute of the LIST tag must be specified.

This attribute may not be used on list columns that are part of list column groups.

DISPLAY=YES | NO
Indicates whether or not the fields in the list column are visible when the panel is displayed. This attribute is allowed only for display panels. YES indicates that the fields in the list column are visible.

NO indicates that the fields in the list column are not visible. NO can only be specified for a column in the list when USAGE=INOUT is specified on this tag and ACTOR=NONE is specified on the LIST tag.

AUTOSKIP=YES | NO
Indicates whether or not the cursor should automatically move to the next input field when data is entered in the last position of the list column field. This attribute is allowed only for display panels. YES is the default.

YES indicates that the cursor automatically moves to the next input field.

NO indicates that the cursor does not automatically move to the next input field. NO can only be specified for a column in the list when USAGE=INOUT is specified on this tag and ACTOR=NONE is specified on the LIST tag.

Optional Text

column-heading
The column heading placed above the column in the reserved part of the list area. If no column heading or the COLHEAD attribute is specified, no column heading is displayed.

The text may appear on more than one line and can only contain the reverse text (RT) tag.

Each word of the column heading is placed on a new line. If multiple words are necessary in a line of the column heading, they must be enclosed in apostrophes ('). Each word or quoted string must fit within the maximum column width defined by the MAXWIDTH attribute of this tag. The maximum number of words or quoted strings allowed is specified by the MAXHEAD attribute on the LIST tag. Column headings are not allowed if MAXHEAD=0 is used for the list area.

Column headings are justified according to the JUSTIFY attribute of this tag only if they are specified as text. No justification is done when the COLHEAD attribute is specified on this tag.

If no column heading text or COLHEAD attribute is specified on any LISTCOL tag for the area, no lines are reserved on the display for heading information.

Formatting Considerations

When the width of the column data is less than the width of the column heading text, the column data is centered under the column heading text. If the data does not center evenly under the heading, the additional space is placed on the right for JUSTIFY=LEFT and on the left for JUSTIFY=RIGHT. The following example shows all three situations:

```
Center Data    Align Left    Align Right
  xxxxxxx       xxxxxxx        nnnnnnnn
  xxxxxxx       xxxxxxx        nnnnnnnn
  xxxxxxx       xxxxxxx        nnnnnnnn
  xxxxxxx       xxxxxxx        nnnnnnnn
```

When the width of the column data is greater than the width of the column heading text, the column heading text is adjusted to the left for JUSTIFY=LEFT and to the right for JUSTIFY=RIGHT. The following example shows these situations:

```
     Align Left        Align Right
     xxxxxxxxxxxx    nnnnnnnnnnnnn
     xxxxxxxxxxxx    nnnnnnnnnnnnn
     xxxxxxxxxxxx    nnnnnnnnnnnnn
     xxxxxxxxxxxx    nnnnnnnnnnnnn
```

If the column heading text requires more than one line, the shorter lines of text are centered with respect to the longest line. If the shorter lines do not center evenly under the longer lines, the additional space is on the right for JUSTIFY=LEFT and on the left for JUSTIFY=RIGHT. The following example shows all situations:

```
           Line Up   Line Up   Line Up   Line Up
 Line Up    Head      Head      Head      Head
 Right      Right     Right     Right     Left
 nnnnnnnn   nnnnnnn   nnn     nnnnnnnnn    xx
 nnnnnnnn   nnnnnnn   nnn     nnnnnnnnn    xx
 nnnnnnnn   nnnnnnn   nnn     nnnnnnnnn    xx
 nnnnnnnn   nnnnnnn   nnn     nnnnnnnnn    xx
```

LISTDEF (List Definition)

```
►►──:LISTDEF──NAME──=──list-name──VARS──=──'──dialog-variable-list──'──────────────────────────►

►─┬──────────────────────────────────┬──┬────────────────────────────────┬─────────────────────►
  └─CHGVAR──=──dialog-variable-name──┘  └─MSGID──=──message-identifier───┘

►─┬────────────────────────────────────┬──┬──────────────────────────────────────────────┬──────►
  └─MSGIDVAR──=──dialog-variable-name──┘  └─MSGF──=──'──qualified-message-file-name──'───┘

►─┬───────────────────────────────────┬──┬───────────────────────────────────────────────────┬─►
  └─PRTFLAG──=──dialog-variable-name──┘  └─EMPHASIS──=──'──dialog-variable-emphasis-list──'──┘

►─┬──────────────────────────────────────────────────┬─.───────────────────────────────────────►◄
  └─PROTECT──=──'──dialog-variable-usage-list──'─────┘
```

The list definition (LISTDEF) tag defines UIM lists which are data structures maintained by the UIM. These lists contain the data for list areas presented on the display. UIM lists consist of a variable number of rows. Each row contains one or more columns, and each column in the row contains a copy of one dialog variable value. These lists are manipulated by the program using the UIM application programming interfaces (APIs). Lists may be shared among panels which contain list areas.

Required Attributes

NAME=*list-name*
The name of the list object. This name must be unique within the panel group. For more information on the rules for naming, see “Name Syntax” on page 469.

VARS='*dialog-variable-list***'**
The list of up to 50 variables which make up the columns of the list. All dialog variables in this list must be previously defined using the variable definition (VAR) tag in this panel group. Not all of these variables need to be displayed in a view defined with the list view (LISTVIEW) tag. The variable names in the list are separated by blanks.

The UIM automatically determines what column is used for action list or selection list processing by identifying the variable specified on the VARS attribute and defined with a BASETYPE of ACTION on the class definition (CLASS) tag. An action column is required for each list view when an ACTOR or SELECT value other than NONE is specified, or when SELECT=SINGLE or MULTI is specified on the list area (LIST) tag. Only one column of a list may be specified as an action column.

Optional Attributes

CHGVAR=*dialog-variable-name*
Specifies that any list entry values entered by the user must be compared to values already in the list entry. This attribute also indicates where the comparison result should be stored. The dialog variable specified must be one of the variables listed on the VARS attribute of this tag, and must be defined with a BASETYPE of CHAR 1 on the CLASS tag.

List values are compared using the internal form of each dialog variable rather than its display form. If any values in the list entry are changed, the list value associated with the dialog variable specified on this attribute is set to '1'. If all the new list values are equal to the old list values, no special processing is performed; the dialog variable specified for this attribute is not changed.

Note: The dialog variable specified on this attribute is set to '1' each time the user changes the list entry, scrolls the list, or uses some other function key. The list value associated with the dialog variable specified on this attribute is set to '1' twice if a value in the list is changed, the list is scrolled, scrolled back again to the original position, and the list entry is changed back to its original value.

An application program can use the list value for the dialog variable specified on this attribute to determine which list entries are changed by the user. The application is responsible for setting and resetting the list entry value to something other than '1'. It does this by updating the list entry after the input values are processed and before the list is redisplayed to make the comparison reliable.

MSGID=*message-identifier*
The message identifier of a message that is displayed to the user when the UIM list is empty. If no message identifier is specified with either the MSGID or MSGIDVAR attribute on this tag, no message is displayed when the list is empty. The MSGID and the MSGIDVAR attributes cannot both be used on the same LISTDEF tag.

MSGIDVAR=*dialog-variable-name*
A dialog variable containing the message identifier of a message displayed to the user when the UIM list is empty. The dialog variable specified must be defined with a BASETYPE of ’CHAR 7’ on the CLASS tag. If no message identifier is specified with either the MSGID or MSGIDVAR attribute on this tag, no message is displayed when the list is empty. The MSGID and the MSGIDVAR attributes cannot both be used on the same LISTDEF tag.

MSGF='*qualified-message-file-name***'**
The message file name containing the message identifier specified on the MSGID or MSGIDVAR attribute. The attribute is allowed only when the MSGID or MSGIDVAR attribute is specified on this tag. If the DFTMSGF attribute is not specified on the panel group (PNLGRP) tag, this attribute must be specified when the MSGID or MSGIDVAR attribute on this tag is specified.

PRTFLAG=*dialog-variable-name*
Specifies the name of a dialog variable in the list definition containing a character printed on the first character of each list entry. For each list entry printed, the character stored in the dialog variable is printed on the first column. The dialog variable specified must be one of the variables listed on the VARS attribute of this tag, and must be defined on the CLASS tag with a BASETYPE of CHAR 1 and a width of 1. This attribute is intended to be used for output listing applications where a special character needs to be printed in column one.

EMPHASIS='*dialog-variable-emphasis-list.***'**
The list of dialog variables containing the emphasis or color associated with each of the variables listed on the VARS attribute of this tag. The number of items in the list must be the same as the number of dialog variables specified on the VARS attribute. Only one emphasis is allowed for each variable listed, but a particular emphasis value may apply to more than one variable. A value of ’*’ should be specified to indicate that normal emphasis should be used with the corresponding variable listed on the VARS attribute, and also used for list columns not displayed in any list view.

The dialog variables specified on this attribute cannot be one of the variables listed on the VARS attribute of this tag.

Each variable in the list must be defined with a BASETYPE of CHAR 1 on the CLASS tag. When an entry is added to the list or an existing entry is updated, the values for these variables determine the emphasis used when that entry is displayed. The following table shows the values that can control the emphasis.

Table 65. Emphasis Values

Value	Output Field	Input Field
0	Normal	Normal
1	Emphasize	Emphasize
2	De-emphasize	Normal

Normal and de-emphasis are the same on a monochrome device.

If a dialog variable listed in this attribute has a value specified other than one of the above values, normal emphasis is used. If this attribute is not specified, normal emphasis is used.

This attribute does not control emphasis during printing.

PROTECT='*dialog-variable-usage-list***'**

The list of dialog variables containing the value used to override the USAGE attribute of the list column (LISTCOL) tag for each of the variables listed on the VARS attribute of this tag. The number of items in the list must be the same as the number of dialog variables specified on the VARS attribute. Only one protection value is allowed for each variable listed, but a particular protection value may apply to more than one variable.

A value of '*' should be specified to indicate that there is no override for the USAGE attribute on the LISTCOL tag for the corresponding variable listed on the VARS attribute. A value of '*' is also used for list columns not displayed in any list view.

The dialog variables specified on this attribute cannot be one of the variables listed on the VARS attribute.

Each variable in the list must be defined with a BASETYPE of CHAR 1 on the CLASS tag. When an entry is added to the list or an existing entry is updated, the values for these variables determine the protection whenever that entry is displayed. This attribute overrides the USAGE attribute from the LISTCOL tag. The valid values for the dialog variables specified on this attribute are as follows:

'1' The variable use is output-only.

'0' The use is controlled by the USAGE attribute on the LISTCOL tag.

Changing a list option field for an action list or selection list to output-only does not affect list processing. If the action field contains an option number or selection value but is displayed as output-only, the option is processed.

If a dialog variable listed in this attribute is not specified or has a value specified other than '1' or '0', the use is controlled by the USAGE attribute of the LISTCOL tag.

LISTGRP (List Column Group)

Syntax for Display Panels:

```
►►─:LISTGRP─COL─=─column-identifier─HELP─=─help-module-name──────────────►

►─┬──────────────────────────────┬───────────────────────────────────────►
  └─NAME─=─list-column-group-name─┘

►─┬─────────────────────────────────────────────────────────┬────────────►◄
  │             ┌─*─┐                                       │
  └─COLSEP─=────┴─N─┴──.──┬───────────────┬──:ELISTGRP.─────┘
                          └─group-heading─┘
```

Syntax for Print Panels:

```
►►─:LISTGRP─COL─=─column-identifier──────────────────────────────────────►

►─┬─────────────────────────────────────────────────────────┬────────────►◄
  │             ┌─*─┐                                       │
  └─COLSEP─=────┴─N─┴──.──┬───────────────┬──:ELISTGRP.─────┘
                          └─group-heading─┘
```

LISTGRP Tag

The list column group (LISTGRP) tag groups columns in a list area together under a single heading that applies to all the columns. This tag is allowed for display panels and print panels. Columns may be defined within the list area that do not belong to any list column group. This tag cannot be used in a list area if `MAXHEAD=0` is specified on the list area (LIST) tag for the area.

There are some restrictions on the kinds of columns that can exist within a column group.

- The action column of the list cannot be in a column group.
- All or none of the columns can have column headings.
- Column groups cannot be nested.
- At least one column must be defined within a column group.

Required Attributes

COL=*column-identifier***.**

The identifier of the column group, which is referred to in the `COLS` attribute on the list view (LISTVIEW) tag. These names are in a separate name space from all other names and are meaningful only within the list area, not within the panel group.

HELP=*help-module-name*

Identifies online help information which explains the purpose of the column group in this list. This attribute is required for display panels; it is not allowed for print panels. The help module name may be a name imported from another panel group, but must follow the rules for names outlined earlier in this chapter. For more information on the rules for names, see "Name Syntax" on page 469.

Optional Attribute

NAME=*list-column-group-name*

The name associated with the group. This attribute is allowed only for display panels. For more information on the rules for naming, see "Name Syntax" on page 469. This name can be used with the Add Pop-Up Windows (QUIADDPW) API to position a window near this list column group.

COLSEP=* | N

The number of spaces by which to separate columns in a list column group. `COLSEP=*` causes UIM to calculate the width of the column separator. The calculated width is between 2 and 5 spaces. `COLSEP=N` can be any positive number. When choosing a value for N, keep in mind the widths of the columns. This attribute does not affect the space between list column groups and other list columns. It also does not affect the space following selection columns in list column groups.

Optional Text

group-heading

The group heading text placed above the column group in the heading part of the list area. This heading must fit on one line over the columns in the group. It is centered over the group of columns and has a graphic indication of the number of columns over which it spans. The headings of the columns in a column group may span different numbers of lines. The group heading appears on the line immediately above the tallest column heading of the group.

No column in a column group may have `MAXWIDTH='*'` specified on the list column (LISTCOL) tag. The columns and the separators between the columns define the width allocated for the group heading. The group heading must be no longer than this width.

If no column headings are specified on any of the columns of the group, the group heading appears immediately above the column data.

The text must appear on the same or next line as the tag and can only contain the reverse text (RT) tag.

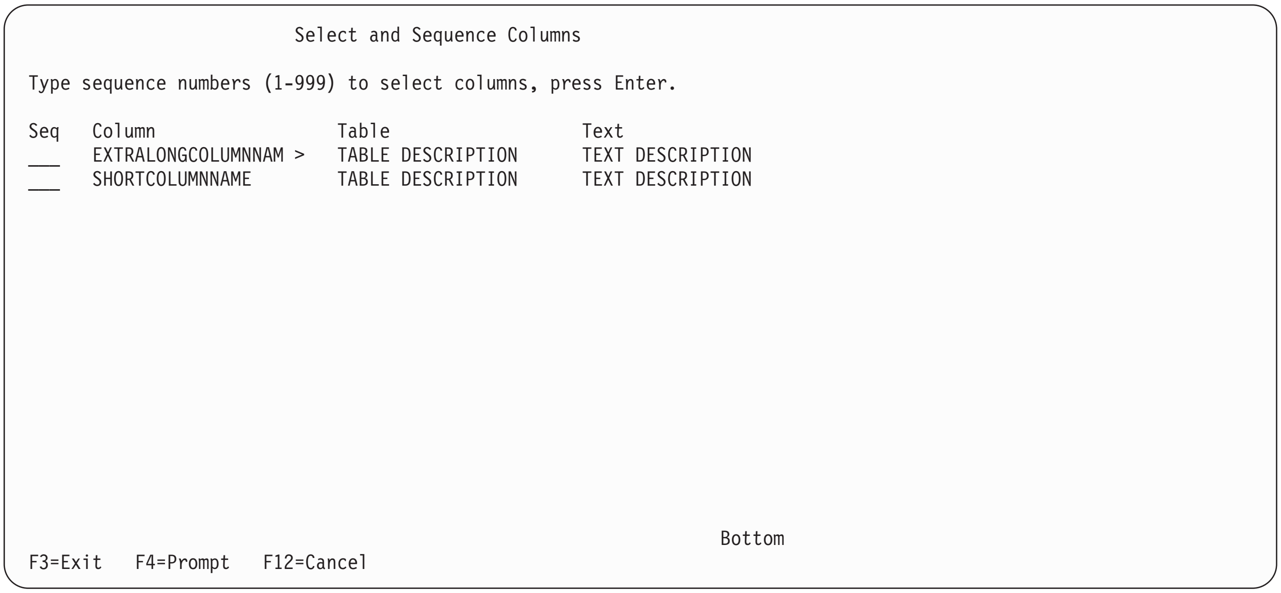
Example: List Column Group

This example uses a column separator of 1.

UIM Source

```
:listdef name=list1
  vars='seq colname more table text'.
:panel panel-attributes.Select and Sequence Columns
:list listdef=list1
      depth='*'
      maxhead=3.
:topinst.Type sequence numbers (1-999) to select
columns, press Enter.
:listcol var=seq
         usage=inout
         maxwidth=3
         help=hseq.Seq
:listgrp col=column
         colsep=1
         help=hlistgrp.
:listcol var=colname
         usage=out
         maxwidth=18.Column
:listcol var=more
         usage=out
         maxwidth=1.
:elistgrp.
:listcol var=table
         usage=out
         maxwidth=20
         help=htable.Table
:listcol var=text
         usage=out
         maxwidth=25
         help=htext.Text
:listview layout=1
          cols='seq column table text'.
:elist.
:epanel.
```

Results



```
                        Select and Sequence Columns

Type sequence numbers (1-999) to select columns, press Enter.

Seq   Column                 Table                  Text
___   EXTRALONGCOLUMNNAM >   TABLE DESCRIPTION      TEXT DESCRIPTION
___   SHORTCOLUMNNAME        TABLE DESCRIPTION      TEXT DESCRIPTION











                                                                    Bottom
F3=Exit   F4=Prompt   F12=Cancel
```

LISTVIEW (List View)

```
►►─:LISTVIEW──COLS──=──'column-identifier-list' ─┬──────────────────────────────────────────┬──────►◄
                                                 │            ┌─1──────────────────────┐     │
                                                 └─LAYOUT──=──┴─number-of-layout-columns─┴──.─┘
```

The list view (LISTVIEW) tag defines one of the views of a list presented in a list area. This tag is allowed for both display panels and print panels.

Multiple list views can be defined in an area and can be changed using the CHGVIEW dialog command, which can be assigned to a function key or to a pull-down choice. If there is more than one list view on the panel and the CHGVIEW dialog command is performed, the list area with the cursor in it changes views.

The view may also be established in a program by changing the value of the dialog variable specified on the `VIEW` attribute on the list area (LIST) tag. Views are numbered consecutively from zero to one less than the number of LISTVIEW tags defined in this area. The first LISTVIEW tag for the list area defines view zero, the second LISTVIEW tag defines view one, and so on for each LISTVIEW tag in the list area.

If the `ACTOR` attribute on the LIST tag has a value other than NONE, the action column of the list must appear in all views of the list area for the panel. If the `SELECT` attribute on the LIST tag has a value other than NONE or MULTI, the action column of the list must appear in all views of the list area.

Although multiple views can be defined for a print panel, only the view identified by the `VIEW` attribute on the LIST tag is printed when the Print Panel (QUIPRTP) API is called. To print more than one view, the QUIPRTP API must be called multiple times, updating the view dialog variable between each call to QUIPRTP.

Required Attribute

COLS='*column-identifier-list*'

A list of column or column group identifiers which are separated by blanks. This list identifies columns and groups that are part of this view.

All variables used in this view must be distinct.

Optional Attribute

LAYOUT=1 | *number-of-layout-columns*

Indicates whether more than one set of columns should be presented on the display at one time. If the width of columns defined by the `COLS` attribute of this tag is small, it may be possible to position more than one set of the columns on the display or page.

The format used for multiple-column layout divides the entire display or page width into layout columns of equal size, with a separator of at least three characters between layout columns. All list columns specified on the `COLS` attribute of this tag must fit within a single layout column.

For all panels with `WIDTH=132` on the display panel (PANEL) or print panel (PRTPNL) tags, a value from 1 through 10 is allowed. For panels with `WIDTH` less than 132, a value from 1 through 6 is allowed.

The following tables show the available column width and what positions are used for each layout column, depending on the value of the `LAYOUT` attribute. The first table applies to all panels with `WIDTH=80` specified on the PANEL or PRTPNL tags. The second table applies to panels with `WIDTH=132` specified.

Table 66. Layout Values for Width=80. Layout column widths and positions for panels with WIDTH=80.

Layout	Layout Width	Layout 1 Column Positions	Layout 2 Column Positions	Layout 3 Column Positions	Layout 4 Column Positions	Layout 5 Column Positions	Layout 6 Column Positions
1	78	2-79					
2	37	2-38	43-79				
3	24	2-25	29-52	56-79			
4	17	2-18	22-38	42-58	62-78		
5	13	2-14	18-30	34-46	50-62	66-78	
6	10	2-11	15-24	28-37	41-50	54-63	67-76

Table 67. Layout Values for WIDTH=132. Layout column widths and positions for panels with WIDTH=132.

Layout	Layout Width	Layout 1 Column Positions	Layout 2 Column Positions	Layout 3 Column Positions	Layout 4 Column Positions	Layout 5 Column Positions	Layout 6 Column Positions
1	130	2-131					
2	63	2-64	69-131				
3	41	2-42	46-86	90-130			
4	30	2-31	35-64	68-97	101-130		
5	23	2-24	28-50	54-76	80-102	106-128	
6	19	2-20	24-42	46-64	68-86	90-108	112-130

Table 68. Layout Values for WIDTH=132. Layout column widths and positions for panels with WIDTH=132.

Layout	Layout Width	Layout 7 Column Positions	Layout 8 Column Positions	Layout 9 Column Positions	Layout 10 Column Positions
7	16	97-112			
8	13	66-78	82-94		
9	11	86-96	100-110	114-124	
10	10	80-89	93-102	106-115	119-128

If a `WIDTH` of less than 80 or a `width` between 81 and 131 is specified on the PANEL tag, the following algorithm can be used to determine the layout width:

1. Subtract 2 from the width of the panel.
2. Subtract 3*(layout-1) from the result of step 1. This is the total separator space between the layouts.
3. Divide the result of step 2 by the layout; do not round up. This is the layout width.
4. If the remainder from step 3 is greater than (layout-1), the separator between the layouts is 4. Otherwise, the separator space is 3.

LP (List Part)

```
►►—:LP—.—list-part-text——————————————————————————————————————►◄
```

The list part (LP) tag identifies a comment or an explanation applying to a part of a list. This tag is only allowed in information areas and help areas. It can be placed anywhere within the list.

LP Tag

The text following the LP tag starts at the left margin of the current level of the list. It is not numbered or lettered. When used with the ordered list (OL) tag, the LP tag does not interrupt or increase the ordered sequence.

Example: List Part

This example uses a list part tag to keep the flow of the list from being interrupted.

UIM Source

```
:ol.
:li.First item
:lp.This is a list part.
:li.Second item
:eol.
```

Results

1. First item

This is a list part.

2. Second item

MBAR (Menu Bar)

```
►►──:MBAR──NAME──=──menu-bar-name──┬────────────────────────────┬──┬──────────────────────────────┬──────────►◄
                                   └─HELP──=──help-module-name─┘  │          ┌─1─┐                │
                                                                  └─MAXBARL──┼─2─┼──.──:EMBAR.──┘
                                                                             └─3─┘
```

The menu bar (MBAR) tag defines a menu bar area of a panel. The menu bar appears as the topmost element in a panel and consists of one or more menu bar choices. When the user positions the cursor at a menu bar choice and requests the ENTER dialog command, the pull-down menu for the selected menu bar choice is displayed below the text for the menu bar choice.

The resulting pull-down menu contains a selection field which has one or more pull-down choices. The user is allowed to choose only one pull-down menu choice. Each selection field has one or more pull-down menu choices.

All menu bars must be defined after any conditional statements and before the key lists.

When a menu bar is specified for a panel with a list action column, a slash (/) or country-designated selection character may be entered on the list action column to indicate the elements the menu bar choice is to act against. The slashes are processed in the same top-to-bottom processing as in normal list action processing. The slash (/) is not allowed on the extended action entry of an action list.

Other tags can be nested within the MBAR tag. These tags are listed in the following table. This table defines the order in which the tags must appear and specifies on what page more information can be found for each tag.

Table 69. Tag Allowed Between the MBAR and EMBAR Tags

Tag Name	Order	Page
MBARC (Menu bar choice)	1	583

Required Attribute

NAME=*menu-bar-name*
The name assigned to the menu bar. For more information on the rules for naming, see “Name Syntax” on page 469.

Optional Attributes

HELP=*help-module-name*
The name of the help information explaining the purpose of the menu bar. The help module name may be a name imported from another panel group, but must follow the rules for names outlined earlier in this chapter. For more information on the rules for names, see “Name Syntax” on page 469.

MAXBARL=1 | 2 | 3
The maximum number of lines that can be used for the descriptions of the menu bar. From 1 to 3 lines can be specified; 1 is the default. Only as many menu bar lines as needed up to the maximum number allowed by MAXBARL are used. If all the menu bar choices do not fit in the number of lines specified, a compiler error results.

Example: Menu Bar

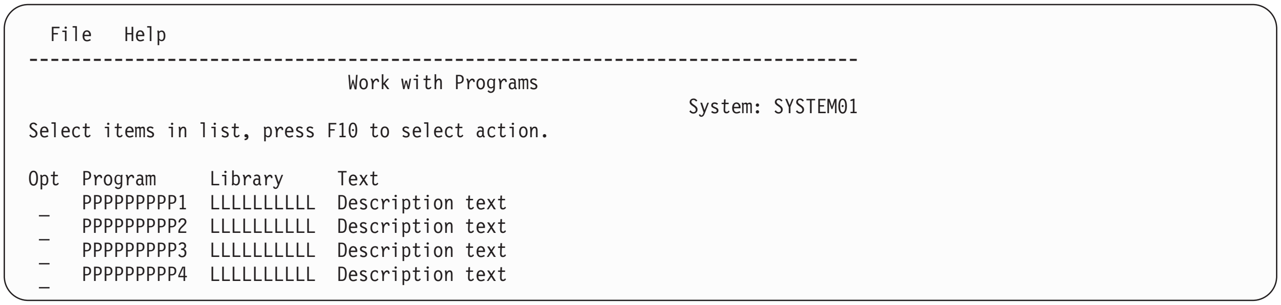
This example defines the menu bar seen on the Work with Programs panel.

UIM Source

```
.* Menu bar for Work with Programs panel
:mbar  name=mbarpgm
        help='mbarpgm/'
        .
.*
.* Menu bar choice for the "File" pull-down
:mbarc    help='mbarpgm/file'
        .File
:pdfld.
:pdfldc    option=2
        help='mbarpgm/file/change'
        action='cmd ?CHGPGM PGM(&var2/&var1)'
        actfor=list
        .Change
:pdfldc    option=4
        help='mbarpgm/file/delete'
        action='cmd DLTPGM PGM(&var2/&var1)'
        actfor=list
        confirm=confpgm
        usrexit='call exitpgm'
        .Delete
:pdfldc    option=5
        help='mbarpgm/file/display'
        action='cmd DSPPGM PGM(&var2/&var1)'
        actfor=list
        .Display
:pdfldc    option=6
        help='mbarpgm/file/exit'
        action=exit
        varupd=no
        .Exit
:pdaccel.F3
:epdfld.
:embarc.
.*
.* Menu bar choice for the "Help" pull-down
:mbarc    help='mbarpgm/help'
        .Help
:pdfld.
:pdfldc    option=1
```

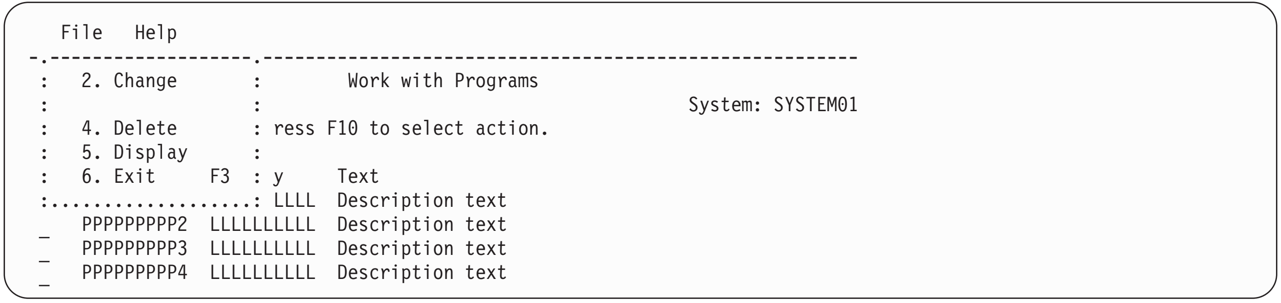
```
        help='mbarpgm/help/helphelp'
        action=helphelp
        varupd=no
        .Help for help...
:pdfldc    option=2
        help='mbarpgm/help/help'
        action=exthelp
        varupd=no
        .Extended help...
:pdfldc    option=3
        help='mbarpgm/help/keyshelp'
        action=keyshelp
        varupd=no
        .Keys help...
:pdfldc    option=4
        help='mbarpgm/help/schidx'
        action=helpidx
        varupd=no
        .Help index...
:pdfldc    option=5
        help='mbarpgm/help/about'
        action='call logopgm'
        varupd=no
        .About...
:epdfld.
:embarc.
:embar.
```

Results



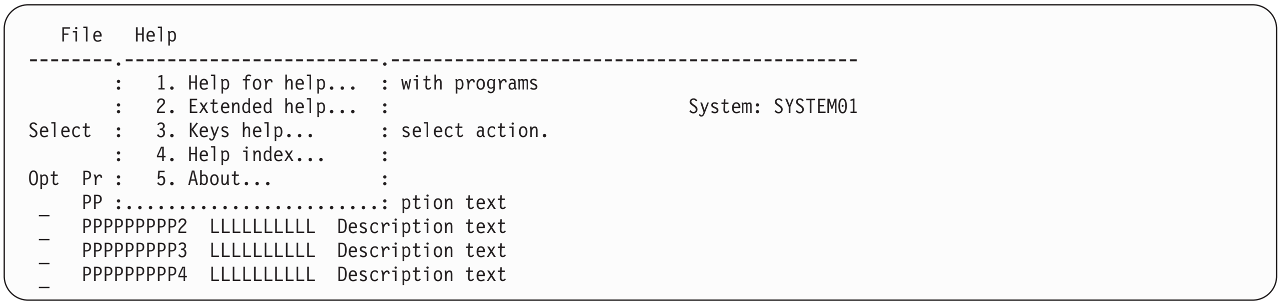
```
  File   Help
------------------------------------------------------------------------------
                              Work with Programs
                                                               System: SYSTEM01
Select items in list, press F10 to select action.

Opt  Program     Library     Text
 _   PPPPPPPPP1  LLLLLLLLLL  Description text
 _   PPPPPPPPP2  LLLLLLLLLL  Description text
 _   PPPPPPPPP3  LLLLLLLLLL  Description text
 _   PPPPPPPPP4  LLLLLLLLLL  Description text
```

The *File* pull-down is displayed as shown below.



```
   File   Help
-.------------------.----------------------------------------------------------
 :   2. Change      :        Work with Programs
 :                  :                                      System: SYSTEM01
 :   4. Delete      : ress F10 to select action.
 :   5. Display     :
 :   6. Exit     F3  : y     Text
 :..................: LLLL  Description text
 _   PPPPPPPPP2  LLLLLLLLLL  Description text
 _   PPPPPPPPP3  LLLLLLLLLL  Description text
 _   PPPPPPPPP4  LLLLLLLLLL  Description text
```

The *Help* pull-down is displayed as shown below.



```
   File   Help
--------.-----------------------.------------------------------------------
        :   1. Help for help...  : with programs
        :   2. Extended help...  :                          System: SYSTEM01
Select  :   3. Keys help...      : select action.
        :   4. Help index...     :
Opt  Pr :   5. About...          :
 _   PP :.......................: ption text
 _   PPPPPPPPP2  LLLLLLLLLL  Description text
 _   PPPPPPPPP3  LLLLLLLLLL  Description text
 _   PPPPPPPPP4  LLLLLLLLLL  Description text
```

When F10 (ACTIONS) is pressed, the cursor is positioned in the space before the first menu bar choice.

When a menu bar choice is selected, the cursor is located at the pull-down field and the tab key moves to the first unselected menu bar choice. Repeated pressing of the tab key moves the cursor through the menu bar area and then back to the pull-down menu.

MBARC (Menu Bar Choice)

```
►►──:MBARC──HELP──=──help-module-name──.──menu-bar-choice-text──:EMBARC.──────────────────────────────►◄
```

The menu bar choice (MBARC) tag defines one choice within a menu bar. The end of the menu bar choice must be indicated with an EMBARC tag.

Other tags can be nested within the MBARC tag. These tags are listed in the following table. This table defines the order in which the tags must appear and specifies on what page more information can be found for each tag.

Table 70. Tags Allowed Between the MBARC and EMBARC Tags

Tag Name	Order	Page
PDFLD (Pull-down field)	1	604
PDFLDC (Pull-down field choice)	2	605

Required Attributes

HELP=*help-module-name*
The name of the help information explaining the purpose of the menu bar choice. The help module name may be a name imported from another panel group, but must follow the rules for names outlined earlier in this chapter. For more information on the rules for names, see “Name Syntax” on page 469.

Required Text

menu-bar-choice-text
The displayable description for the menu bar choice. The text must appear on the same line or the next line as the tag, the text can contain only the reverse text (RT) tag, and the text must be no longer than 32 bytes.

The menu bar is formed by concatenating the text from the various menu bar choices. If the text does not fit on the rest of a line, it is placed on the next line. The menu bar is limited to the number of lines specified on the MAXBARL attribute on the MBAR tag.

The text is displayed with a tabable space in front of it. When the tab keys are used to position the cursor at one of the menu bar choices, the cursor appears one character to the left of the text. Two spaces are placed between each text field for the menu bar. The visual result has three spaces between choices.

MENU (Menu Area)

```
►►──:MENU──DEPTH──=──┬─area-depth─┬──┬──────────────────────────┬──────────────────►
                     └─'*'────────┘  │                ┌─SPACE─┐ │
                                     └─BOTSEP──=──────┼─NONE──┼─┘
                                                      └─RULE──┘

►─┬──────────────────────┬───────────────────────────────────────────────────────►◄
  │           ┌─NO──┐    │
  └─SCROLL──=─┴─YES─┴────┘.──┬────────────┬──:EMENU.─
                             └─area-title─┘
```

The menu area (MENU) tag defines a menu area. This tag is allowed only for display panels. The end of the menu area must have a EMENU tag.

The menu area is included in the panel with all the other areas that are defined on the panel. Only one menu area may be present on any panel.

Either the command line (CMDLINE) tag or the option line (OPTLINE) tag must be specified for a menu panel. The user selects a menu option by entering the option number on the command line or option line.

Other tags can be nested within the MENU tag. These tags are listed in the following table. The table defines the order in which the tags must appear, indicates which tags can be used in display panels only, print panels only, or both (specified by a D, P, or B, respectively), and specifies on which page more information can be found for each tag.

When more than one tag is listed with the same order number, all tags of that number can be mixed in any order. However, a tag with a higher order number cannot precede a tag with a lower order number. For example, a tag with an order number of three cannot precede a tag with an order number of one or two.

Table 71. Tags Allowed Between the MENU and EMENU Tags

Tag Name	Order	Use	Page
TOPINST (Top instruction line)	1	D	630
APPFMT (Application formatted area)	2	D	472
MENUGRP (Menu column group)	3	D	587
MENUI (Menu item)	3	D	588
BOTINST (Bottom instruction line)	4	D	474

Required Attribute

DEPTH=*area-depth* **| '*'**
The depth of the area in lines, including separators if any are specified.

If '*' is specified, the space remaining on the display after everything else is allocated is given to this area. Only one area in the panel may have '*' coded.

Optional Attribute

BOTSEP=SPACE | NONE | RULE
Defines the bottom separator for the menu area. If SPACE is specified, a line of spaces is used.

NONE indicates that no separator line exists.

If RULE is specified, a line of underscored spaces is used as a separator line.

SCROLL=NO | YES
NO indicates that the menu is not scrollable.

YES indicates that the menu area is intended to be scrollable. Ordinarily YES should not be used, but longer, scrollable menus can be constructed with SCROLL=YES. For a SCROLL=YES area, a line of spaces is used by the UIM to provide a line for the scroll information. If BOTSEP=SPACE is also specified on this tag, only one line of spaces is used unless this area also contains bottom instructions.

Optional Text

area-title
The title of the area. If no text is specified, no title line is allocated to the area. The text must appear on the same or next line as the tag, the text can contain only the reverse text (RT) tag, and cannot exceed a maximum length of 55 characters.

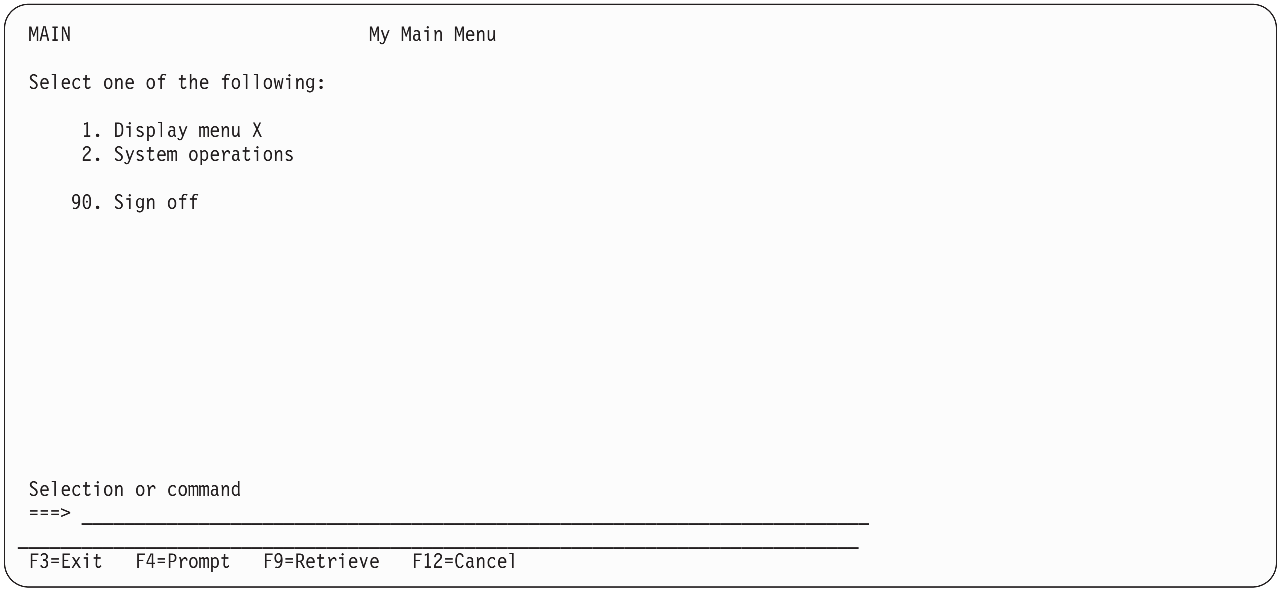
Example 1: Simple Menu Area

This example defines a menu area with three options.

UIM Source

```
:panel name=main
       help='menu/main'
       keyl=small
       panelid=zmenu
       topsep=space
       .My Main Menu
:menu depth='*'.
:topinst.Select one of the following:
:menui help=MNUSER
       option=1
       action='MENU X'
       .Display menu X
:menui help=MNSYS
       option=2
       action='MENU SYSOPR'
       .System operations
:menui help=MNOFF
       option=90
       action='CMD SIGNOFF'
       .Sign off
:emenu.
:cmdline size=long.Selection or command
:epanel.
```

Results



```
MAIN                         My Main Menu

Select one of the following:

     1. Display menu X
     2. System operations

    90. Sign off













Selection or command
===> ___________________________________________________________________________
 _______________________________________________________________________________
F3=Exit   F4=Prompt   F9=Retrieve   F12=Cancel
```

Example 2: Menu Area with Groups

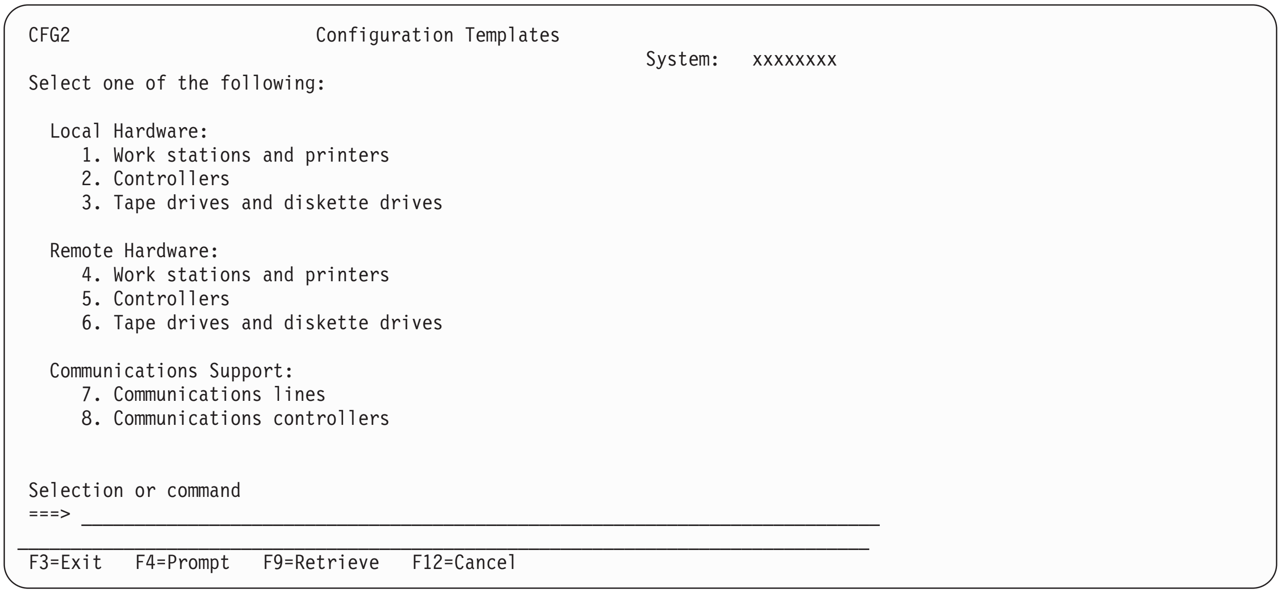
This example has three menu areas, each of which has options that can be chosen.

UIM Source

```
:panel name=cfg2
       help='menu/cfg2'
       panelid=zmenu
       keyl=small
       .Configuration Templates
:menu depth='*'.
:topinst.Select one of the following:
:menugrp.Local Hardware:
:menui help='opt/lcl/wsprt'
       option=1
       action='MENU LCLWSPRT'
       .Work stations and printers
:menui help='opt/lcl/ctl'
       option=2
       action='MENU LCLCTL  '
       .Controllers
:menui help='opt/lcl/tapdkt'
       option=3
       action='MENU LCLTAPDKT'
       .Tape drives and diskette drives
:emenugrp.
:menugrp.Remote Hardware:
:menui help='opt/rmt/wsprt'
       option=4
       action='MENU RMTWSPRT'
       .Work stations and printers
:menui help='opt/rmt/ctl'
       option=5
       action='MENU RMTCTL  '
       .Controllers
:menui help='opt/rmt/tapdkt'
       option=6
       action='MENU RMTTAPDKT'
       .Tape drives and diskette drives
:emenugrp.
:menugrp.Communications Support:
```

```
:menui help='opt/cmn/lines'
       option=7
       action='MENU CMNLINES'
       .Communications lines
:menui help='opt/cmn/ctl'
       option=8
       action='MENU CMNCTL  '
       .Communications controllers
:emenugrp.
:emenu.
:cmdline size=long.Selection or command
:epanel.
```

Results



```
CFG2                       Configuration Templates
                                                         System:   xxxxxxxx
Select one of the following:

  Local Hardware:
     1. Work stations and printers
     2. Controllers
     3. Tape drives and diskette drives

  Remote Hardware:
     4. Work stations and printers
     5. Controllers
     6. Tape drives and diskette drives

  Communications Support:
     7. Communications lines
     8. Communications controllers


Selection or command
===> ____________________________________________________________________________
________________________________________________________________________________
F3=Exit   F4=Prompt   F9=Retrieve   F12=Cancel
```

MENUGRP (Menu Group)

```
►►──:MENUGRP──┬──────────────────────────┬──.──┬──────────────────┬──:EMENUGRP.──────────────────────────►◄
              └─COND──=──condition-name──┘     └──group=heading───┘
```

The menu group (MENUGRP) tag groups menu items of a menu area together and supplies a heading for the group. This tag is allowed only for display panels.

If the number of lines required to display the first two items of the menu group exceeds the number of lines remaining in the current page, the entire group is forced to the next page. When all items in the menu group are not completely displayed on one page, the group is continued onto as many pages as required to display all items.

Other tags can be nested within the MENUGRP tag. These tags are listed in the following table. The table defines the order in which the tags must appear, indicates which tags can be used in display panels only, print panels only, or both (specified by a D, P, or B, respectively), and specifies on which page more information can be found for each tag.

When more than one tag is listed with the same order number, all tags of that number can be mixed in any order. However, a tag with a higher order number cannot precede a tag with a lower order number. For example, a tag with an order number of three cannot precede a tag with an order number of one or two.

Table 72. Tag Allowed Between the MENUGRP and EMENUGRP Tags

Tag Name	Order	Use	Page
MENUI (Menu item)	1	D	588

Optional Attribute

COND=*condition-name*
The group is displayed and active on the panel only if the condition specified is true. If the condition is false, the entire group is not shown. The condition must be defined in the panel group prolog with the condition definition (COND) tag.

Optional Text

group-heading
The heading placed above the menu items of the group. The text must appear on the same or next line as the tag and can only contain the reverse text (RT) tag. The group heading always appears on one display line beginning in column four.

MENUI or MI (Menu Item)

```
►►──:MENUI or MI──HELP──=──help-module-name──OPTION──=──option-number──ACTION──=──'action-text' ─────────►

►─┬──────────────────────────┬──┬───────────────────────┬──┬────────────────────────┬──────────────────►
  └─NAME──=──menu-item-name──┘  └─COND──=──condition-name─┘  └─AVAIL──=──condition-name─┘

►─┬───────────────────────────────┬──┬────────────────────────────────────────────┬──────────────────►
  └─AVLMSGID──=──message-identifier─┘  └─AVLMSGF──=──qualified-message-file-name─┘

►─┬─────────────────────────────┬──┬──────────────────────────────────┬────────────────────────────────►
  └─MARKER──=──condition-name─┘  └─ITEM──=──dialog-variable-name─┘

►─.─┬──────────────────────────────┬─────────────────────────────────────────────────────────────────►◄
    └─menu-item-description-text─┘
```

Required Attributes

HELP=*help-module-name*
Identifies online information explaining the purpose of the menu item. The help module name may be a name imported from another panel group, but must follow the rules for names outlined earlier in this chapter. For more information on the rules for names, see “Name Syntax” on page 469.

The online information identified by this attribute is displayed when help is requested while the cursor is positioned on the text for this tag or while the cursor is on the command or option line and a valid menu item number has been entered.

OPTION=*option-number*
The number to assign this option. Option numbers are integers in the range 0 through 999.

The menu items are displayed in the order defined for the menu area. If the numbering of two items is not consecutive, a blank line is automatically placed between the two items.

ACTION='*action-text***'**
The action occurring when the menu item is selected by the user. The valid forms of action-text are:

- *'CALL program-reference'*

- *'CANCEL'*
- *'CMD command-string'* Any dialog variable name in the command string must be preceded by an ampersand and should be ended with a period to denote variable substitution.
- *'EXIT'*
- *'MENU qualified-menu-name RTNPNT|NORTNPNT'*
- *'RETURN positive-integer'*

For a description of each of these actions, see Appendix B, "UIM Dialog Commands," on page 641.

Optional Attributes

NAME=*menu-item-name*
The name associated with the menu item. This name can be used with the Add Pop-Up Window (QUIADDPW) API to position a window near this menu item. For more information on the rules for naming, see "Name Syntax" on page 469.

COND=*condition-name*
The menu item is in effect on the panel only if the condition specified is true. The condition must be defined in the panel group prolog with the condition definition (COND) tag.

AVAIL=*condition-name*
The name of a condition indicating to the UIM whether or not the menu item is available for selection by the user. The condition must be defined in the panel group prolog with the COND tag.

Unavailable menu items are displayed with an asterisk (*) overlaying the first part of the menu item number.

When the condition is true, the menu item is available. When the condition is false, the menu item is not available. Any condition specified on the `COND` attribute of this tag takes precedence over this attribute.

AVLMSGID=*message-identifier*
The message identifier of the message displayed when the pull-down choice is selected but is not available as specified by the `AVAIL` attribute of this tag. This attribute is allowed only when the `AVAIL` attribute is specified.

If this attribute is not specified, the UIM displays a default message stating that the choice is currently not available.

AVLMSGF=*'qualified-message-file-name'*
The message file name containing the message identifier. This attribute is allowed only when the `AVLMSGID` attribute is specified on this tag. If the `DFTMSGF` attribute is not specified on the panel group (PNLGRP) tag, and if the `AVLMSGID` attribute is specified on this tag, then this attribute must be specified.

MARKER=*condition-name*
The name of a condition indicating whether or not the menu item is marked with a greater than sign (>). The marker is displayed only if the condition specified is true. The condition must be defined in the panel group prolog with the COND tag.

When the condition is true, a greater than sign is displayed to the left of the option number of the menu item. When the condition is false, the line is blank before the option number of the menu item. If this attribute is omitted, no marker appears on the menu item.

ITEM=*dialog-variable-name*
The name of a dialog variable containing the text for the menu item description when the panel is displayed. The dialog variable may be defined with a `BASETYPE` of CHAR, IGC, or BIN on the class definition (CLASS) tag. The declared length of the dialog variable must be no longer than what fits on one line of the panel, starting in column 10. Allowance must be made if there is an application formatted area.

If this attribute is used, no *menu-item-description-text* can be specified after the period for this tag.

Optional Text

menu-item-description-text
The descriptive text accompanying the option and shown on the menu. The text may appear on more than one line in the source. The text can contain only the reverse text (RT) tag, and cannot exceed 70 characters in length.

The item description text begins in column 10 of the panel. The option number, a period, and one blank appear immediately before the item description text. The item number begins in column 7, 6, or 5, depending on whether the option number has 1, 2, or 3 digits. The text is formatted onto additional lines as necessary, and the additional lines are not indented two spaces to fit on the panel.

NT or NOTE (Note)

```
►►──:NT. or :NOTE.──┬───────────┬──:ENT. or :ENOTE.──────────────────────────────────────────►◄
                    └─note-text─┘
```

The note (NT) tag identifies a single- or multiple-paragraph note. This tag must have a matching end tag. You can also use :NOTE and :ENOTE in place of the :NT and :ENT tags. These tags are only allowed in information areas and help areas. Notes cannot be nested.

Notes can occur anywhere in text that a paragraph is allowed. They can contain any basic text items, but cannot contain headings, figures, lines, examples, or any type of list. If this tag has not been ended when a heading tag is encountered, the note ends and a warning message is issued by the compiler.

The note is an implied paragraph. It is formatted as a block and indented 6 spaces from the current margin. The word *Note*, formatted in highlight phrase 2 (HP2) and followed by a colon, begins the paragraph. There are two spaces between the note tag and the text if either the note tag or the text are single-byte characters. There are four spaces between the note tag and the text if both are double-byte characters.

A note is aligned with the text of a list item when used within a list.

Other tags can be nested within the NOTE tag. These tags are listed in the following table. The table defines the order in which the tags must appear, indicates which tags can be used in display panels only, print panels only, or both (specified by a D, P, or B, respectively), and specifies on which page more information can be found for each tag.

When more than one tag is listed with the same order number, all tags of that number can be mixed in any order. However, a tag with a higher order number cannot precede a tag with a lower order number. For example, a tag with an order number of three cannot precede a tag with an order number of one or two.

Table 73. Tags Allowed Between the NOTE and ENOTE Tags

Tag Name	Order	Use	Page
P (Paragraph)	1	B	593
PC (Paragraph continuation)	1	B	603

Optional Text

note-text
Although the text of the note is not required, the tag has no meaning when text is not specified.

Example: Using a Note

The note tag makes important information stand out in a paragraph, as in this example.

UIM Source

```
:P.Here's a paragraph.
:NT.
Here's the first paragraph
of the note.
:P.
Here's the second paragraph
of the note.
:ENT.
:P.Here's a paragraph.
```

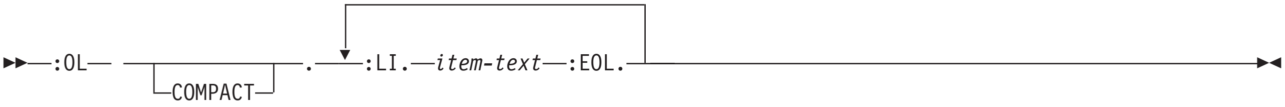
Results

`Here's a paragraph.` **Note:** Here’s the first paragraph of the note.

Here’s the second paragraph of the note.

Here’s a paragraph.

OL (Ordered List)



The ordered list (OL) tag identifies an ordered list of items. It requires a matching end tag. These tags are only allowed in information areas and help areas.

Ordered lists can occur anywhere in text and can be nested within other lists.

Note: Care should be taken when using unformatted lines (LINES), figure (FIG), and example (XMP) tags within ordered lists, because text that does not fit on one line wraps to column one of the next line. Lines and figures start at the current left margin, and examples are indented four spaces from the current left margin. The current left margin changes when nested lists are formatted. If online help information has the LINES, FIG, or XMP tag imbedded at various locations, including within lists, it may not look the same each time.

The OL tag is formatted as a hanging, indented list, with the item identifier (1,2,...a,b,...) at position four relative to the left margin. The text begins at position eight relative to the left margin.

Ordered lists are identified in a formatted document by sequential numbers or letters, depending on the definition for each level. The levels of definition are as follows:

1. 1., 2., 3., . . .,11., 12., . . ., 99.
2. a., b., c., . . ., z., aa., bb., . . ., zz.
3. 1), 2), 3), . . .,11), 12), . . ., 99)
4. a), b), c), . . ., z), aa), bb), . . ., zz)

Any additional levels repeat the sequence from the beginning.

Other tags can be nested within the OL tag. These tags are listed in the following table. The table defines the order in which the tags must appear, indicates which tags can be used in display panels only, print panels only, or both (specified by a D, P, or B, respectively), and specifies on which page more information can be found for each tag.

When more than one tag is listed with the same order number, all tags of that number can be mixed in any order. However, a tag with a higher order number cannot precede a tag with a lower order number. For example, a tag with an order number of three cannot precede a tag with an order number of one or two.

Table 74. Tags Allowed Between the OL and EOL Tags

Tag Name	Order	Use	Page
FIG (Figure)	1	B	527
LINES (Unformatted lines)	1	B	546
XMP (Example)	1	B	639
NT (Note)	1	B	590
P (Paragraph)	1	B	593
PC (Paragraph continuation)	1	B	603
LP (List part)	1	B	579
OL (Ordered list)	1	B	591
SL (Simple list)	1	B	622
UL (Unordered list)	1	B	633
PARML (Parameter list)	1	B	602
DL (Definition list)	1	B	525

Optional Attribute

COMPACT
This attribute formats the list without a blank line between items.

Required Tag

:LI.*item-text*
The text for the list item. The text is preceded by a number or a letter and a period.

Example: Ordered List

This example has two ordered lists, one imbedded within the other. The second ordered list uses the `COMPACT` attribute.

UIM Source

```
Some normal text...
:ol.
:li.First item (number)
:ol.
:li.First item (letter)
:li.Second item (letter)
:eol.
:li.Second item (number)
:eol.
```

Results

```
Some normal text...

1. First item (number)

   a. First item (letter)
   b. Second item (letter)

2. Second item (number)
```

OPTLINE (Option Line)

```
►►──:OPTLINE──┬──────────────────────────┬──.──┬──────────────────┬──────────────────────────────►◄
              └─NAME──=──option-line-name─┘     └─instruction-text─┘
```

The option line (OPTLINE) tag specifies that a panel has a menu option field as opposed to a command line which allows commands or options to be entered. This tag is allowed only for display panels. The OPTLINE tag can be used only on a panel that has a menu area.

Any text associated with this tag is used as an instruction line, appearing immediately above the option field on the panel.

This tag must be the final tag in a panel, placed just before the EPANEL tag. The command line (CMDLINE) tag and the OPTLINE tag are mutually exclusive.

Optional Attribute

NAME=*option-line-name*

The name associated with the option line. This name can be used with the Add Pop-Up Window (QUIADDPW) API to position a window near the option line. For more information on the rules for naming, see “Name Syntax” on page 469.

Optional Text

instruction-text

The text appearing as instructions for the option line. This text is an implied paragraph.

When the display is formatted, any text that does not fit onto one display line is formatted on the following blank lines and indented two columns. The text can be a maximum of 255 characters and can contain only the reverse text (RT) tag. If no text is provided, no instruction line is allocated or displayed above the option line.

P (Paragraph)

```
►►──:P.──paragraph-text──────────────────────────────────────────────────────────────────────────►◄
```

The paragraph (P) tag identifies a paragraph, which is one or more sentences related by their subject matter. This tag is only allowed in information areas and help areas.

Paragraphs can occur anywhere in text and can contain text items. A matching end tag is not allowed; a paragraph is ended by another paragraph or by a higher-level element.

Each paragraph is formatted as a block of text without indenting the first line. One blank line separates paragraphs from other text items.

Paragraphs inserted within a list align with the text of the list item.

Optional Text

Although the paragraph text is not required, the tag has no meaning when text is not specified.

Example: Paragraph Tag

This example illustrates how the paragraph tag is used.

UIM Source

```
:P.Here's a paragraph.
Lines are formatted to
fill the column.
:P.Here's another
paragraph.
```

Results

```
Here's a paragraph. Lines are formatted to fill the column.

Here's another paragraph.
```

PANEL (Display Panel)

```
►►──:PANEL──NAME──=──panel-name──HELP──=──help-module-name──KEYL──=──key-list-name──────────►

►──┬─────────────────────────────────┬──┬──────────────────────────────┬────────────────────►
   └─PANELID──=──dialog-variable-name─┘  └─TITLE──=──dialog-variable-name─┘

►──┬──────────────────────────────┬──┬───────────────────────────────┬──┬────────────────────────┬──►
   │                ┌─80──────────┐ │  │                ┌─FIT─────────┐ │  └─MBAR──=──menu-bar-name─┘
   └─WIDTH──=───────┴─panel-width─┴─┘  └─DEPTH──=───────┴─panel-depth─┴─┘

►──┬─────────────────┬──┬────────────────────────────┬──┬──────────────────────────────┬──────►
   │          ┌─1─┐  │  │                ┌─SYSNAM───┐ │  └─DATE──=──dialog-variable-name─┘
   └─MSGL──=──┼─2─┼──┘  └─TOPSEP──=──────┼─SPACE────┼─┘
              ├─3─┤                      ├─RULE─────┤
              └─4─┘                      ├─DATETIME─┤
                                         └─NONE─────┘

►──┬──────────────────────────────┬──┬──────────────────────────┬──┬────────────────────────────┬──►
   └─TIME──=──dialog-variable-name─┘  │                ┌─PNLGRP─┐ │  └─ENTER──=──'──action-text──'─┘
                                      └─ENBGUI──=──────┼─NO─────┼─┘
                                                       └─YES────┘

►──┬─────────────────────────────┬──┬─────────────────────────────────────────────┬──────────►
   └─SELECT──=──'──action-text──'─┘  └─USREXIT──=──'──CALL── ──program-reference──'─┘

►──┬─────────────────────────┬──┬────────────────────────────────┬────────────────────────────►
   └─TT──=──truth-table-name─┘  └─CSRVAR──=──dialog-variable-name─┘

►──┬────────────────────────────────┬──┬────────────────────────────────┬─────────────────────►
   └─CSRPOS──=──dialog-variable-name─┘  └─CSRLST──=──dialog-variable-name─┘

►──┬────────────────────────────────┬──┬─────────────────────────────────┬────────────────────►
   └─CSREID──=──dialog-variable-name─┘  └─CSRNAME──=──dialog-variable-name─┘

►──.──┬──────────────────┬──:EPANEL.──────────────────────────────────────────────────────────►◄
      └─panel-title-text─┘
```

The display panel (PANEL) tag is opened using the PANEL tag and closed using the EPANEL tag. It contains tags to define one or more panel area definitions.

Other tags can be nested within the PANEL tag. These tags are listed in the following table. The table defines the order in which the tags must appear, indicates which tags can be used in display panels only, print panels only, or both (specified by a D, P, or B, respectively), and specifies on which page more information can be found for each tag.

When more than one tag is listed with the same order number, all tags of that number can be mixed in any order. However, a tag with a higher order number cannot precede a tag with a lower order number. For example, a tag with an order number of three cannot precede a tag with an order number of one or two.

Table 75. Tags Allowed Between the PANEL and EPANEL Tag

Tag Name	Order	Use	Page
DATA (Data presentation area)	1	D	496
INFO (Information area)	1	D	537

Table 75. Tags Allowed Between the PANEL and EPANEL Tag (continued)

Tag Name	Order	Use	Page
LIST (List area)	1	D	552
MENU (Menu area)	1	D	584
TEXT (Text area)	1	D	623
CMDLINE (Command line)	2	D	491
OPTLINE (Option line)	2	D	593

Required Attributes

NAME=*panel-name*
The name of the panel. For more information on the rules for naming, see "Name Syntax" on page 469.

HELP=*help-module-name*
The name of a help module containing the beginning of the extended help for the panel.

KEYL=*key-list-name*
The name of a key list for the panel.

Optional Attributes

PANELID=*dialog-variable-name*
The name of a dialog variable containing the text for the panel identifier when the panel is displayed. The dialog variable must be defined on a class definition (CLASS) tag with BASETYPE='CHAR 10', 'NAME 10', or 'OBJNAME 10'. The UIM displays the contents of the variable in the upper left corner of the panel. If this attribute is omitted, no panel identifier appears on the display.

To specify a panel identifier for a menu created by using the Create Menu (CRTMNU) command, the dialog variable for ZMENU should be declared with PANELID=ZMENU on the PANEL tag. The value of the *ZMENU* dialog variable is displayed as the panel identifier.

TITLE=*dialog-variable-name*
The name of a dialog variable containing the text for the panel title. The dialog variable must be defined with a display width of 55 or the width of the panel minus 2, whichever is less. A translation list may be used to provide the panel title.

The UIM trims trailing blanks from the string and centers the resulting text in the panel title area of the display. If this attribute is used, no *panel-title-text* can be specified.

For a pop-up panel, the declared length of the dialog variable must be equal to the width of the pop-up panel minus 2 bytes, and less if PANELID or TOPSEP=DATETIME is specified on this tag for the panel.

WIDTH=80 | *panel-width*
The width of the panel. If the width is not specified, the panel width defaults to 80 bytes. The specified width must contain 2 bytes of space for the left and right margins. A panel with WIDTH=80 results in 78 bytes of space for panel data.

The maximum width supported for a panel displayed as a full screen is 132 bytes. The maximum width supported for a pop-up window depends on what size screen the device supports. On a device that supports 24 rows by 80 bytes, 74 bytes is the maximum width of a pop-up window. On a device that supports 27 rows by 132 bytes, 126 bytes is the maximum width of a pop-up window. The minimum width for a pop-up window is 20 bytes.

DEPTH=FIT | *panel-depth*
The depth of the panel. If the depth is not specified, the panel depth defaults to FIT. If DEPTH=FIT, the

UIM uses the depth of the device at display time to determine the depth of the panel. When DEPTH=FIT, the compiler verifies that all panel elements fit within 24 lines.

The specified depth includes:

- One line for a panel title
- One line of area data
- One or two lines for function key descriptions
- One line for the message line

The maximum supported depth is 27 lines for a panel displayed in a full display. The maximum supported depth for a pop-up panel depends on what the display device can support. On a device that can only support 24 rows by 80 bytes, the maximum depth of a pop-up window is 21 lines. On a device that can support 27 rows by 132 bytes, the maximum depth of a pop-up window is 24 lines.

The minimum allowed depth is 5 lines.

MBAR=*menu-bar-name*

The name of the menu bar used on the panel. This attribute is not allowed on confirmation panels or panels that will be used as pop-up windows.

Menu bars are only allowed on panels defined with one of the following sizes:

- WIDTH=80 or 132 bytes, and DEPTH=FIT
- WIDTH=80 bytes, DEPTH=24 lines
- WIDTH=132 bytes, DEPTH=27 lines

MSGL=1 | 2 | 3 | 4

The number of message lines that should appear on this panel. One message line is the default.

TOPSEP=SYSNAM | SPACE | RULE | DATETIME | NONE

Defines the top separator for the panel. If SYSNAM is specified, the separator line contains only the system name.

If SPACE is specified, a line of spaces is used.

If RULE is specified, a line of underscored spaces is used. RULE cannot be specified if the MBAR attribute is specified on this tag.

If DATETIME is specified, the separator line contains the date and time from the dialog variables identified on the DATE and TIME attributes of this tag. DATETIME also causes the system name to be on the right side of the title line. When date and time is used as the top separator for the panel, it should **not** be used to provide a current time clock. This function should be used to show the date and time in which the important data on the panel is collected or generated.

NONE indicates that no separator line exists.

Values other than TOPSEP=SPACE or NONE may cause undesirable results when the panel width is less than 80.

DATE=*dialog-variable-name*

The name of a dialog variable containing a date displayed on the top separator for the panel. This attribute is required when TOPSEP=DATETIME is specified on this tag. The dialog variable must be defined with BASETYPE='DATE' on the CLASS tag.

TIME=*dialog-variable-name*

The name of a dialog variable containing a time displayed on the top separator for the panel. This attribute is required when TOPSEP=DATETIME is specified. The dialog variable must be defined with BASETYPE='TIME' on the CLASS tag.

If the dialog variable can have a time zone value associated with it, the time value will be trimmed of blanks on the right and then right-justified on the screen. If the dialog variable's time zone value is blank, no time zone value is displayed.

ENBGUI=PNLGRP | NO | YES

Specifies whether the panel is enabled for conversion to a graphical user interface (GUI) by a client program.

When ENBGUI=YES is specified, the UIM includes information about the layout and content of the panel in the 5250 data stream. This information is used by client programs to create the graphical interface on the client. Table 76 describes the information that is included in the 5250 data stream:

Table 76. Layout of UIM finger print

Byte	Value and Description
0	Hex 27 - Non-display attribute
1	Hex 40 - Blank
2	Hex 20 - Terminating attribute
3	Reserved
4	Hex 00 - Panel BIDI=NONE or LTR Hex 40 - Panel BIDI=RTL
5	Hex 00 - Panel does not contain a list area Hex 40 - Panel might contain a list area
6	Hex 00 - Panel does not contain a menu area Hex 40 - Panel might contain a menu area
7	Hex 00 - Panel does not contain a help area Hex 40 - Panel might contain a help area This includes the UIM Help (UH) panels as well as application panels containing a :TEXT area.
8	Hex 00 - Panel does not contain any other areas Hex 40 - Panel might contain one or more other areas Other types of areas are :DATA (input and/or output), :INFO (similar to help).
9	Hex 27 - Non-display attribute
10	Hex 40 - Blank
11	Hex 20 - Terminating attribute

When ENBGUI=NO is specified, the UIM does not include the extra information in the data stream.

ENTER=*'action-text'*

The default enter action for the panel, occurring when the Enter key is pressed and no UIM-defined action, such as menu or action list processing, is necessary. If this attribute is omitted, the panel is redisplayed to the user with no message or error indication. The valid forms of action text are:

- *'CALL program-reference'* For a description of the interface between the UIM and the program for a function key CALL, see the **APIs** topic in the iSeries Information Center.
- *'CMD command-string'* Any dialog variable name in the command string must be preceded by an ampersand to denote variable substitution.
- *'MSG message-identifier [qualified-message-file-name]'*
- *'RETURN positive-integer'*

For a description of each of these dialog commands, see Appendix B, "UIM Dialog Commands," on page 641.

SELECT=*'action-text'*

The default selection action for the panel, occurring when items are selected in a selection list or an action list and no action is selected from the menu bar. If this attribute is not specified, the panel is displayed to the user with no message or error indication. The valid forms of action text are:

- *'ACTIONS'*
- *'MSG message-identifier [qualified-message-file-name]'*
- *'PULLDOWN'*
- *'RETURN positive-integer'*

The PULLDOWN and ACTIONS dialog commands cannot be specified if the MBAR attribute is not specified on this tag.

USREXIT='*CALL program-reference***'**
A general exit program called each time the user presses a function key or the Enter key after dialog variables and list entries are updated and before any action specified on the key list item (KEYI), list action (LISTACT), pull-down field choice (PDFLDC), or menu item (MENUI) tag is performed.

The program is passed parameters that include what function is requested by the user. The exit program may do application-defined validity checks, change the display position attribute of lists that appear on the screen, or both. If the exit program sends a CPF6A02 status message to the UIM, the UIM redisplays the panel without performing the normal action processing.

The panel is redisplayed without calling this exit program if validity check errors associated with the CLASS, validity checking (CHECK), or translation list (TL) tags are detected, or if the user presses an inactive function key. If the user presses a function key defined with VARUPD=NO on the KEYI tag, the values entered on the panel by the user are not available to this program. They are only stored internally in the UIM.

For a description of the CALL dialog command, see Appendix B, “UIM Dialog Commands,” on page 641.

For a description of the interface between the UIM and the general exit program, see the **APIs** topic in the iSeries Information Center.

TT=*truth-table-name*
The name of a truth table defined by the truth table (TT) tag, specifying what combinations of truth values may occur for conditions defined by condition definition (COND) tags while the application is running.

The table specified may contain any subset or superset of the conditions referred to by tags in the panel definition. Only the truth value combinations specified in the table, augmented by worst-case assumptions for any truth values not specified in the table, are considered in evaluating whether or not the panel definition is usable. The table should not exclude any truth value combination that could occur when the panel is displayed. If a valid truth value combination is omitted, a panel group object may create without error and have undesirable results when panels are displayed. For example, the UIM may fail to show panel elements that are conditioned-on for display. For more information about truth tables, see “TT (Truth Table)” on page 631.

If this attribute is omitted, all combinations of truth values are considered possible. This causes the tag language compiler to make worst-case assumptions for all conditions in evaluating whether or not the panel definition is valid.

CSRVAR=*dialog-variable-name*
This attribute must name a CHAR 10 dialog variable which is used by the UIM to contain either the name of the dialog variable or the name of the command line where the cursor is or will be positioned.

The UIM updates this variable after the panel is displayed and before the general exit program or any action routines are called. It contains the name of the dialog variable associated with the field, or the name of the command line (if named) where the cursor is positioned on exit from the panel. If the cursor is not in a field or on a named command line, the *CSRVAR* dialog variable is set to blanks.

The Display Panel (QUIDSPP) API supports an option that allows the application program to control the initial cursor position by setting the dialog variables associated with the CSRVAR, CSRPOS, CSRLST, and CSREID panel attributes before the panel is displayed. If the *CSRVAR* dialog variable does not contain the name of a dialog variable or command line displayed on the panel, the UIM default cursor positioning is used and no error is reported.

If a data item is defined using data item extenders, the CSRVAR attribute contains the name of the extender dialog variable if the cursor is positioned within the extender field. If the cursor is positioned between fields, CSRVAR is set to the name of the dialog variable specified on the VAR attribute of the data presentation area (DATA) tag, and the *CSRPOS* dialog variable is set to zero.

CSRPOS=*dialog-variable-name*

This attribute must name a BIN 15 dialog variable used by the UIM to contain the character position within the field identified by the CSRVAR attribute on this tag where the cursor is or will be positioned.

The UIM updates this variable after the panel is displayed, before the panel exit program or any action routines are called. It contains the position within a field where the cursor is positioned on exit from the panel. If the cursor is not in a field, the dialog variable is set to zero. If the cursor is positioned in a text area associated with the field, such as a prompt label or column heading for a list area, the CSRVAR attribute contains the name of the dialog variable for the corresponding data item or list column, and the *CSRPOS* dialog variable is set to zero.

The QUIDSPP API supports an option which allows the using program to control the initial cursor position by setting the dialog variables associated with the CSRVAR, CSRPOS, CSRLST, and CSREID attributes on this tag before the panel is displayed. If the CSRPOS attribute identifies a position that is not valid for the display field, the UIM default cursor positioning is used and no error is reported.

CSRLST=*dialog-variable-name*

This attribute must name a CHAR 10 dialog variable used by the UIM to contain the name of the list where the cursor is or will be positioned.

The UIM updates this variable after the panel is displayed, before the panel exit program or any action routines are called. It contains the name of the list associated with the list area where the cursor is positioned on exit from the panel. If the cursor is not in a list area, the dialog variable is set to blanks.

The QUIDSPP API supports an option that allows the using program to control initial cursor position by setting the dialog variables associated with the CSRVAR, CSRPOS, CSRLST, and CSREID attributes of this tag before the panel is displayed. If the CSRLST attribute is not blank and does not contain the name of a list displayed on the panel, the UIM default cursor positioning is used and no error is reported.

CSREID=*dialog-variable-name*

This attribute must name a CHAR 4 dialog variable used by the UIM to contain the list entry handle within the list identified by the CSRLST attribute where the cursor is or will be positioned.

The UIM updates this variable after the panel is displayed, before the panel exit program or any action routines are called. It contains the list entry identifier where the cursor was positioned on exit from the panel. If the cursor was not in any list entry, the *CSREID* dialog variable is set to X'00'. If the cursor was positioned in a text area associated with a list column, such as a column heading, the CSRLST attribute contains the name of the corresponding list and the *CSREID* dialog variable is set to X'00'.

The QUIDSPP API supports an option that allows the using program to control the initial cursor position by setting the dialog variables associated with the CSRVAR, CSRPOS, CSRLST, and CSREID attributes of this tag before the panel is displayed. If the CSREID attribute identifies an unusable position that does not specify a list entry displayed on the panel, the UIM default cursor positioning is used and no error is reported.

CSRNAME=*dialog-variable-name*

This attribute must name a CHAR 10 dialog variable used by the UIM to contain the name of the item where the cursor is positioned.

The UIM updates this variable after the panel is displayed, before the panel exit program or any action routines are called. It contains the name of the item where the cursor is positioned on exit from the panel. If the cursor was not on any item, the *CSRNAME* dialog variable is set to blanks.

The value of this variable may be used to position pop-up windows adjacent to a field using the Add Pop-Up Window (QUIADDPW) API.

Optional Text

panel-title-text
The title used on this panel. The text must appear on the same or next line as the tag. The text cannot contain other tags, and is limited in length based on the width of the panel. The *panel-title-text* may be no more than 55 bytes long and is required unless the `TITLE` attribute is specified on this tag.

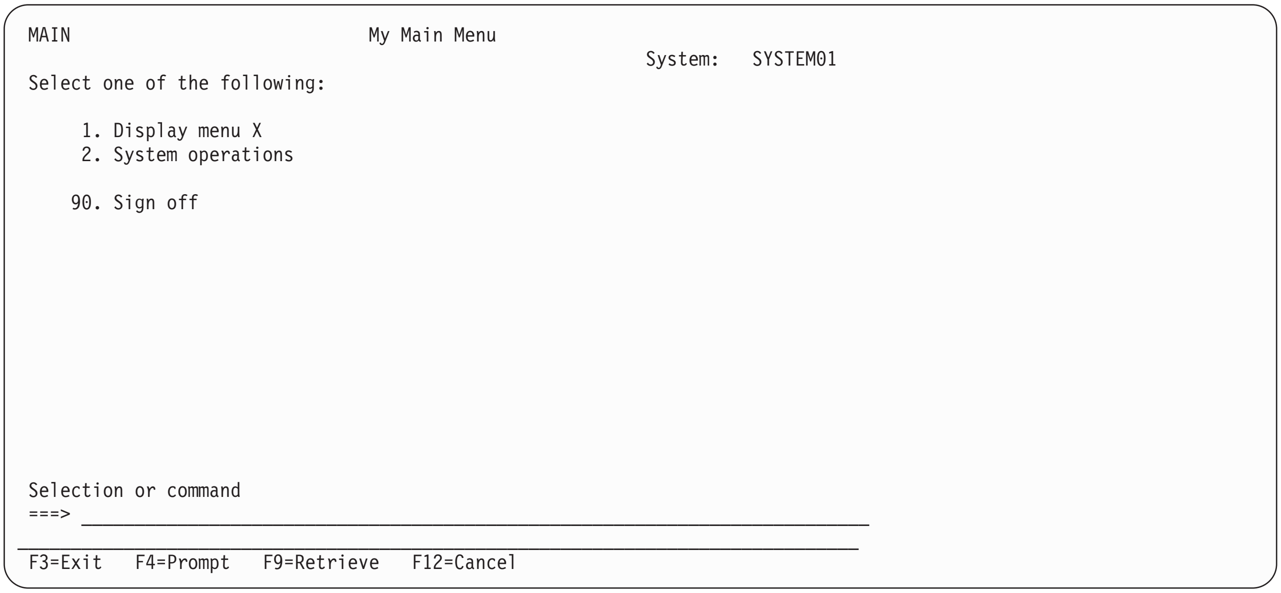
Example: Panel Definition

This example defines a panel and sets up the elements used within the panel.

UIM Source

```
:panel name=main
       help=hmain
       panelid=zmenu
       keyl=small
       .My Main Menu
:menu depth='*'.
:topinst.Select one of the following:
:menui help=MNUSER option=1
       action='CMD MENU X'
       .Display menu X
:menui help=MNSYS
       option=2
       action='CMD MENU SYSOPR'
       .System operations
:menui help=MNOFF
       option=90
       action='CMD SIGNOFF'
       .Sign off
:emenu.
:cmdline size=long.Selection or command
:epanel.
```

Results



```
MAIN                            My Main Menu
                                                          System:   SYSTEM01
Select one of the following:

     1. Display menu X
     2. System operations

    90. Sign off











Selection or command
===> __________________________________________________________________________
_______________________________________________________________________________
F3=Exit   F4=Prompt   F9=Retrieve   F12=Cancel
```

PARML (Parameter List)

```
              ┌──────────────────────────────────────┐
              ▼                                      │
►►──:PARML.────:PT.──parameter-term──:PD.──description─┴──:EPARML.────────────────────►◄
```

The parameter list (PARML) tag identifies parameter terms and descriptions. These tags are only allowed in information areas and help areas. This tag requires a matching end tag. The terms defined and their definitions are identified by the parameter term (PT) tag and the parameter definition (PD) tag, respectively.

Note: Care should be taken when using unformatted lines (LINES), figure (FIG), and example (XMP) tags within parameter lists because text that does not fit on one line wraps to column one of the next line. Lines and figures start at the current left margin, and examples are indented four spaces from the current left margin. The current left margin changes when nested lists are formatted. If there is online help information containing lines, figures, or examples imbedded at various locations, including within lists, it may not look the same each time.

Parameter lists can occur anywhere in text; they can be nested within other lists, and other lists can be nested within parameter lists. Two PT or two PD tags cannot be used consecutively.

Nested PARML tags increase the current left margin by four.

Other tags can be nested within the PARML tag. These tags are listed in the following table. The table defines the order in which the tags must appear, indicates which tags can be used in display panels only, print panels only, or both (specified by a D, P, or B, respectively), and specifies on which page more information can be found for each tag.

When more than one tag is listed with the same order number, all tags of that number can be mixed in any order. However, a tag with a higher order number cannot precede a tag with a lower order number. For example, a tag with an order number of three cannot precede a tag with an order number of one or two.

Table 77. Tags Allowed Between the PARML and EPARML Tags

Tag Name	Order	Use	Page
FIG (Figure)	1	B	527
LINES (Unformatted lines)	1	B	546
XMP (Example)	1	B	639
NT (Note)	1	B	590
P (Paragraph)	1	B	593
PC (Paragraph continuation)	1	B	603
LP (List part)	1	B	579
OL (Ordered list)	1	B	591
SL (Simple list)	1	B	622
UL (Unordered list)	1	B	633
PARML (Parameter list)	1	B	602
DL (Definition list)	1	B	525

Required Tags

PT.*parameter-term*
The parameter term. The term formats in highlight phrase 2 (HP2) and can contain the programming keyword (PK) or programming variable (PV) tags.

The term appears on a line by itself, starting at position four relative to the current left margin.

PD.*parameter-description*
The description of the parameter. It can also contain the PK and PV tags.

The definition is printed on the following line, with the paragraph indented eight spaces from the current left margin. It is an implied paragraph and can contain any text items. Additional paragraphs can be inserted following the description paragraph by using the paragraph tag.

Example: Parameter List

This example uses a parameter list to define terms or parameters.

UIM Source

```
:PARML.
:PT.TERM
:PD.This is a description of
the term.  :PK.Term:EPK. is
a programming keyword.
:PT.:PK def.DEFAULT:EPK.
:PD.This is a sample default
parameter.
:PT.:PV.variable:EPV.
:PD.This is a parameter variable.
:EPARML.
```

Results

TERM
`This is a description of the term.` **`Term`** `is a programming keyword.`

DEFAULT
`This is a sample default parameter.`

variable
`This is a parameter variable.`

PC (Paragraph Continuation)

```
►►──:PC.──paragraph-continuation-text──────────────────────────────────►◄
```

The paragraph continuation (PC) tag identifies the continuation of a paragraph that has been interrupted by another document element. This tag is allowed only in information areas and help areas.

A paragraph continuation usually occurs after a figure, example, or a list. It indicates that the following text is a continuation of the paragraph interrupted by a figure, example, or list.

A matching end tag is not allowed. A paragraph continuation is implicitly ended by another paragraph or by a higher-level element.

Example: Paragraph Continuation

The paragraph continuation tag continues the paragraph after the interruption of an example.

UIM Source

```
:p.If you enter the following
command
:xmp.
WRKSPLF
:exmp.
:pc.a listing of all my spool files
appears on the
terminal.
```

Results

```
If you enter the following command

WRKSPLF

a listing of all my spool files appears on the terminal.
```

PDACCEL (Pull-Down Accelerator)

```
►►──:PDACCEL──.──accelerator-text──────────────────────────────────────────►◄
```

The pull-down choice accelerator (PDACCEL) tag defines text to be displayed as the accelerator key for a pull-down choice.

This tag defines only the text displayed. It does not automatically define the function key. It is your responsibility to define the function key with the key list item (KEYI) tag to perform the action defined for the pull-down choice accelerator.

Required Text

accelerator-text
The displayable description for the accelerator key. The text must appear on the same or next line as the tag. The text cannot contain any other tags, and cannot exceed 4 bytes in length.

The accelerator text is displayed to the right of its pull-down choice. The first character of each accelerator in the pull-down field is left-aligned two spaces after the longest text for any pull-down choice.

PDFLD (Pull-Down Field)

```
►►──:PDFLD──┬─────────────────────────────┬──.──:EPDFLD.──────────────────►◄
            └─NAME──=──pull-down-field-name─┘
```

The pull-down field (PDFLD) tag defines a selection field within the menu bar pull-down. A pull-down field consists of one or more pull-down choices. The pull-down fields are formatted according to the IBM Systems Application Architecture (SAA) basic interface.

A single pull-down field can be defined for one menu bar choice between the menu bar choice (MBARC) and the EMBARC tags. The largest option number allowed for a pull-down choice is 99. The number of pull-down choices allowed to be specified between the PDFLD tag and the EPDFLD tag is determined by how many choices fit on the screen, based on conditioning. The maximum number of pull-down choices active at one time is determined by what fits on a screen.

Other tags can be nested within the PDFLD tag. These tags are listed in the following table. The table defines the order in which the tags must appear, indicates which tags can be used in display panels only, print panels only, or both (specified by a D, P, or B, respectively), and specifies on which page more information can be found for each tag.

When more than one tag is listed with the same order number, all tags of that number can be mixed in any order. However, a tag with a higher order number cannot precede a tag with a lower order number. For example, a tag with an order number of three cannot precede a tag with an order number of one or two.

Table 78. Tags Allowed Between the PDFLD and EPDFLD Tags

Tag Name	Order	Use	Page
PDACCEL (Pull-down choice accelerator)	1	D	604
PDFLDC (Pull-down field choice)	1	D	605

Optional Attribute

NAME=*pull-down-field-name*
The name associated with the pull-down field. For more information on the rules for naming, see “Name Syntax” on page 469.

PDFLDC (Pull-Down Field Choice)

```
►►──:PDFLDC──OPTION──=──option-number──HELP──=──help-module-name──ACTION──=──'──action-text──'──────────►

►──┬─────────────────────────┬──┬──────────────────────────────────┬──────────────────────────────────►
   │              ┌─PANEL─┐  │  └─CONFIRM──=──internal-panel-name──┘
   └─ACTFOR──=────┴─LIST──┴──┘

►──┬──────────────────────────────────────────────┬──┬─────────────────────────┬──┬─────────────────────────────┬──►
   └─USREXIT──=──'──CALL program-reference──'──┘  └─VARUPD──=──┬─YES─┬───┘  └─COND──=──condition-name──┘
                                                                 └─NO──┘

►──┬─────────────────────────────┬──┬─────────────────────────────────────┬──────────────────────────────►
   └─AVAIL──=──condition-name──┘  └─AVLMSGID──=──message-identifier──┘

►──┬──────────────────────────────────────────────────┬──┬────────────────────────────────────┬─────────►
   └─AVLMSGF──=──'──qualified-message-file-name──'──┘  └─CHOICE──=──dialog-variable-name──┘

►──.──┬────────────────────────┬────────────────────────────────────────────────────────────────────────►◄
      └─pull-down-choice-text─┘
```

The pull-down field choice (PDFLDC) tag defines one choice within a pull-down field. A pull-down field consists of one or more pull-down choices.

Required Attributes

OPTION=*option-number*
The number assigned to this option. Option numbers are integers in the range of 1 through 99.

The pull-down choices are displayed in the order defined in the pull-down field. If the numbering of two choices is not consecutive, a blank line is automatically placed between the two choices.

HELP=*help-module-name*

Identifies help information explaining the purpose of the pull-down field choice. The name of the help module may be a name imported from another panel group, but must follow the rules for names outlined earlier in this chapter. For more information on the rules for names, see “Name Syntax” on page 469.

The online information identified by this attribute is displayed when help is requested while the cursor is positioned on text for this tag. This online information is also included as part of the contextual help displayed when the cursor is positioned on the text for the menu bar choice for this pull-down menu, or when the cursor is positioned within the pull-down menu but not on the text for one of the choices. If the cursor is in the entry field and a valid choice has been entered when help is requested, the help for that choice is displayed.

ACTION=*'action-text'*

The action occurring when the pull-down choice is selected.

Table 79. Valid Action Text for ACTION Values

Dialog Command	ACTFOR=PANEL	ACTFOR=LIST
CALL	X	X
CHGVIEW	X	
CMD	X	X
CMDLINE	X	
DSPHELP	X	
EXIT	X	
EXTHELP	X	
HELPHELP	X	
HELPIDX	X	
KEYSHELP	X	
MENU	X	
RETRIEVE	X	
RETURN	X	X

For a description of each of these actions, see Appendix B, “UIM Dialog Commands,” on page 641.

Optional Attributes

ACTFOR=PANEL | LIST

If PANEL is specified, the pull-down choice performs actions against the whole panel.

If LIST is specified, the pull-down choice performs actions against items in a selection list or in an action list.

When LIST is specified on a pull-down choice and if the user has not selected a list to be processed, the pull-down choice is displayed with an asterisk (*) overlaying the option number to indicate that the pull-down choice is not available.

CONFIRM=*internal-panel-name*

The name of the confirmation panel displayed before the action is performed. This attribute is allowed only when ACTFOR=LIST and ACTION=CALL or ACTION=CMD is specified on this tag. The confirmation panel must give the user the option to confirm or not confirm the action. If the action is confirmed, the appropriate actions from this tag are performed. If the action is not confirmed, none of those actions are performed.

The confirmation panel must be another panel defined within this panel group.

For a list of the recommended and required conventions for this attribute, see “Confirmation Panel Requirements” on page 608 and “Confirmation Panel Conventions” on page 608.

USREXIT='*CALL program-reference***'**

Specifies the exit program for the action list program called to update the list entries after the action defined in the ACTION attribute is performed. This attribute is allowed only when ACTFOR=LIST and ACTION=CALL or ACTION=CMD is specified on this tag. The program is passed information that includes the name of the list and an indication of whether the option succeeded or failed.

For a description of the CALL dialog command, see Appendix B, “UIM Dialog Commands,” on page 641.

For a description of the interface between the UIM and the exit program for action lists, see the **APIs** topic in the iSeries Information Center.

VARUPD=YES | NO

This attribute is allowed only when PANEL is specified on the ACTFOR attribute for this tag. When ACTFOR=LIST is specified, variable pool updating is performed. If YES is specified, validity checking occurs and the variable pool is updated before the action takes place.

If NO is specified, the action is performed immediately and no variable pool updating occurs.

Dialog Command	VARUPD Values
CALL	YES | NO
CHGVIEW	YES
CMD	YES | NO
CMDLINE	NO
DSPHELP	NO
EXIT	NO | YES
EXTHELP	NO
HELPHELP	NO
KEYSHELP	NO
HELPIDX	NO
MENU	YES
RETRIEVE	NO
RETURN	YES | NO

Note: The default value is based on the ACTION attribute. The first value is the default value.

COND=*condition-name*

The pull-down choice is in effect on the panel only if the condition specified evaluates to true. The condition must be defined in the panel group prolog with the condition definition (COND) tag. When the choice is conditioned-off, it does not appear in the pull-down and the help for the choice is not included in requests to display help.

AVAIL=*condition-name*

The name of a condition used by the UIM to determine whether or not the pull-down choice is available. The condition must be defined in the panel group prolog with the COND tag.

When the condition is true, the pull-down choice is available. When the condition is false, the pull-down choice is not available. Any condition specified on the COND attribute of this tag takes precedence over this attribute.

Unavailable choices are displayed with a color change on color devices and an asterisk (*) overlaying the first part of the choice option number.

AVLMSGID=*message-identifier*

The message identifier of the message displayed when the pull-down choice is selected when it is not available as specified by the AVAIL attribute. This attribute is allowed only when the AVAIL attribute is specified on this tag. If this attribute is not specified, the UIM displays a default message stating that the choice is currently not available.

AVLMSGF='*qualified-message-file-name*'

The message file name containing the message identifier. This attribute is allowed only when the AVLMSGID attribute is specified on this tag. If the DFTMSGF attribute is not specified on the panel group (PNLGRP) tag, and if the AVLMSGID attribute on this tag is specified, this attribute must be specified.

CHOICE=*dialog-variable-name*

The name of a dialog variable containing the pull-down choice text to be displayed. The dialog variable must be defined with a width less than or equal to 32 bytes. If this attribute is used, no *pull-down-choice-text* can be specified.

Dialog variables must be defined with a BASETYPE of CHAR, IGC, or BIN on the class definition (CLASS) tag.

The error state of the dialog variable is not used for determining the highlighting of the text.

Special formatting for IGC. (The abbreviation IGC is used in commands and keywords to represent double-byte character set functions.) When a dialog variable with a BASETYPE of IGC is specified on the CLASS tag, the UIM does special formatting. If the variable value begins with a shift-out character (X'0E'), the UIM shifts the value 1 byte to the left to preserve vertical alignment with choices on other lines.

Optional Text

pull-down-choice-text

The descriptive text shown in the pull-down field. The text may appear on a single line in the source. The text can contain only the reverse text (RT) tag, and cannot exceed 32 bytes in length. The *pull-down-choice-text* is required unless the CHOICE attribute is specified on this tag.

For a pull-down choice, the option number, a period, and one blank appears immediately before the choice description text. The NBRSHAPE attribute of the PNLGRP tag explains how option numbers are presented when BIDI=LTR or when BIDI=RTL is specified on the PNLGRP tag.

Confirmation Panel Requirements

The panel named on the CONFIRM attribute of this tag must follow these conventions:

- The ENTER attribute on the display panel (PANEL) tag of the confirmation panel should have 'RETURN 100' coded. This indicates to the UIM that the action is confirmed by the user.
- No ACTION attribute on a key list item (KEYI) tag can be assigned to the RETURN action, because the UIM is handling the display of the panel and is unable to interpret the returned value.
- The confirmation panel must have a list area.
- The list used as the confirmation list must not also be used on the action list panel.
- The confirmation panel must not have an action list, a command line, a menu area, or a menu bar.

Confirmation Panel Conventions

The panel named on the CONFIRM attribute of this tag should follow these conventions:

- It should have title text and top instructions as appropriate to describe the panel.
- It should have a single list area with a list definition that contains the same (or a subset of) columns defined in the original action or selection list.
- The list should be scrollable.
- If the action list has multiple views, the confirmation list should also have multiple views and the same dialog variable should be used for the view variable on both lists. The confirmation panel should also have a function key defined for the CHGVIEW dialog command. This allows a change view operation on the confirmation panel to change the view of the action or selection list panel when it is redisplayed.
- The confirmation panel should have function key definitions for the ENTER, CANCEL, PAGEUP, PAGEDOWN, HELP, and PRINT dialog commands.

- If a general exit program is defined, it should avoid making changes to the list or condition values that affect the redisplay of the original panel.
- The list used on the confirmation panel should not be used on anything but a confirmation panel, because the UIM deletes and changes the contents of that list. If multiple confirmation panels are used within a UIM application, they can share the same list.
- Most of the online help information for the action list panel can be used for the confirmation panel, but separate help information should be provided for the action column and for the extended panel help.

PK (Programming Keyword)

```
►►──:PK──┬─────┬──.──programming-keyword-text──:EPK.──────────────────────────►◄
         └─DEF─┘
```

The programming keyword (PK) tag identifies a programming keyword. This tag is only allowed in information areas and help areas. It requires a matching end tag.

A programming keyword can occur anywhere in text. It helps explain the elements of programming syntax and is frequently used within parameter lists. For more information on parameter lists, see “PARML (Parameter List)” on page 602.

The PK and EPK tag phrase must be specified on word boundaries. If the two characters immediately following the EPK tag are a punctuation mark and a blank, the UIM automatically extends the emphasis attribute to include the punctuation mark. This allows the punctuation mark and the text associated with it to be displayed with the same emphasis.

Optional Attribute

DEF
Specifies that the programming keyword is a default value. The programming keyword formats in highlight phrase 2 (HP2) unless the `DEF` attribute is specified. If the `DEF` attribute is specified, the programming keyword formats as highlight phrase 3 (HP3).

Required Text

programming-keyword-text
Specifies the programming keyword.

PNLGRP (Panel Group)

```
►►──:PNLGRP──┬──────────────────────────────────┬──┬──────────────────────┬──┬──────────────────────────┬──►
             └─SCHIDX──=──qualified-object-name─┘  │               ┌─NO──┐ │  │                 ┌─SBCS─┐ │
                                                   └─ENBGUI──=──┴─YES─┴─┘  └─TXTMODE──=──┴─DBCS─┴─┘

►──┬─────────────────────────────────────────────┬──┬─────────────────────┬──┬──────────────────────────────┬──►
   └─TXTCHRID──=──' ──character-set code-page──' ─┘  │           ┌─NONE─┐ │  │                   ┌─ARABIC─┐ │
                                                     └─BIDI──=──┼─LTR──┼─┘  └─NBRSHAPE──=──┴─HINDI──┴─┘
                                                                 └─RTL──┘

►──┬───────────────────────────────────────────┬──┬─────────────────────────────────────────────────┬──►
   └─DFTMSGF──=──qualified-message-file-name─┘  └─SUBMSGF──=──' ──qualified-message-file-name──' ─┘

►─.──:EPNLGRP.────────────────────────────────────────────────────────────────────────────────────►◄
```

PNLGRP Tag

The panel group (PNLGRP) tag begins the definition of a panel group. Only one PNLGRP tag is allowed and a matching EPNLGRP tag is required.

Other tags can be nested within the PNLGRP tag. These tags are listed in the following table. This table defines the order in which the tags must appear and specifies on which page more information can be found for each tag.

When more than one tag is listed with the same order number, all tags of that number can be mixed in any order. However, a tag with a higher order number cannot precede a tag with a lower order number. For example, a tag with an order number of three cannot precede a tag with an order number of one or two.

Table 80. Tags Allowed Between the PNLGRP and EPNLGRP Tags

Tag Name	Order	Page
COPYR (Copyright)	1	496
ISCHSYN (Index search synonym)	any	540
IMPORT (Import)	2	536
CLASS (Class definition)	3	478
VAR (Variable definition)	4	634
VARRCD (Variable record definition)	5	637
LISTDEF (List definition)	6	573
COND (Condition definition)	7	492
TT (Truth table)	8	631
MBAR (Menu bar)	9	580
KEYL (Key list)	10	545
PANEL (Display panel)	11	595
PRTHEAD (Print head panel)	11	613
PRTPNL (Print panel)	11	618
HELP (Help module)	11	529

Optional Attributes

SCHIDX=*qualified-object-name*

The index search object used when the index search function is requested from the help modules for panels defined in this panel group. If SCHIDX is not specified, the index search function is not accessible when help modules are displayed for panels in this panel group. This attribute has no effect on help modules defined in the panel group unless panels in the panel group use the help modules.

ENBGUI=NO | YES

Specifies the default value for enabling a client program for all panels in the panel group. This attribute establishes the value when ENBGUI=PNLGRP is specified or defaulted on the PANEL tag.

When ENBGUI=YES is specified for a panel, the UIM includes information about the layout and content of the panel in the 5250 data stream. This information is used by a client program to create the graphical interface on the client. Table 76 on page 598 describes the information that is included in the 5250 data stream.

When ENBGUI=NO is specified for a panel, the UIM does not include the extra information in the data stream.

TXTMODE=SBCS | DBCS

Specifies whether a single-byte character set (SBCS) or double-byte character set (DBCS) is used in the tag text. If DBCS is specified, the tag text must have the shift-in or shift-out characters to indicate the start and end of DBCS text.

A panel group object for which TXTMODE=DBCS is specified can only be used on a display station and system capable of handling DBCS information. An attempt to display a DBCS menu, open a DBCS panel group object, or present DBCS help information on a device that does not support DBCS results in an exception.

TXTCHRID='*character-set code-page***'**

Specifies the character set and code page for SBCS text data in the panel group source. This information is used for synonyms defined using the index search synonym (ISCHSYN) tag. For more information on this tag, see "ISCHSYN (Index Search Synonym)" on page 540. If this attribute is omitted, all text must be on the index search (ISCH) and ISCHSYN tags to be in the character set and code page specified by the QCHRID system value at the time the panel group is created.

The TXTCHRID attribute specifies the code page and character set of the text information in the panel group source. It must not be confused with the CHRID parameter on the CRTPNLGRP command, which controls the way dialog variables with CHRID=PNLGRP specified on the class definition (CLASS) tag are handled when the dialog variable value is sent to or from the display station.

BIDI=NONE | LTR | RTL

Indicates the orientation of the panels in the panel group. NONE indicates that the panels in the panel group should be displayed in a single direction with a left-to-right orientation. NONE is the default and is required when TXTMODE=DBCS is specified on this tag.

When BIDI=NONE is coded, the following items are ignored:

- The NBRSHAPE attribute on this tag
- The BIDI, NBRSHAPE, CONTXTREV, and SYMSWAP attributes on the CLASS tag
- All reverse text (RT) tags

LTR indicates that the panels in the panel group are bidirectional and should be displayed with a left-to-right orientation.

RTL indicates that the panels in the panel group are bidirectional and should be displayed with a right-to-left orientation.

When either LTR or RTL is specified for a panel group, the work station controller enables the hot key to flip the panel so that the user can flip the panel if necessary. When RTL is specified, the controller automatically flips all panels of the panel group as they are displayed.

The normal use of LTR is for panels written in a left-to-right language with a small amount of right-to-left language text in it. The normal use of RTL is for panels written in a right-to-left language with a small amount of text in a left-to-right language.

The UIM does not allow a LTR or RTL panel to be used on a display device that does not have the correct code page for bidirectional languages. Hebrew uses code page 424 and Arabic uses code page 420.

The value of this attribute of all possible panel groups that can be reached through hypertext links should be the same. If they are not the same when a reference phrase is selected, the resulting screen or pop-up window may be displayed with the opposite orientation.

NBRSHAPE=ARABIC | HINDI

This attribute is ignored when BIDI=NONE is specified on this. The NBRSHAPE attribute controls the display shape of UIM-generated numbers. This attribute is ignored except when code page 420 (Arabic) is being used.

When ARABIC is specified, the normal digits, X'F0' through X'F9', are used for numbers. ARABIC is the default.

When HINDI is specified, the translation of digits is performed with a translation table defined for each national language.

Digits that come from the panel group source or from messages are not affected by the `NBRSHAPE` attribute because they are expected to already be translated to the correct character. The numbers that are affected by the `NBRSHAPE` attribute are:

- Numbers generated by the UIM for menu items
- Numbers generated by the UIM for ordered list numbers
- Numbers generated by the UIM for pull-down field choices
- Page number information on a print panel and printed help information

Numbers that are not affected by the `NBRSHAPE` attribute are:

- The digits used in the system name
- Messages displayed on the message line
- Copyright dates on the message line
- Product number information on a print panel and in printed help information

Because the Hindi zero looks like a period, the format of the number in front of a menu item or pull-down choice is 'n)' instead of 'n.', where 'n' is the menu item or choice option number.

DFTMSGF= *qualified-message-file-name*
The default message file used when a message file is not specified for a message identifier on any of the following tags:

CHECK	Validity checking
LISTDEF	List definition
TL	Translation list
DATASLTC	Data selection field
PDFLDC	Pull-down field choice
LISTACT	List action
MENUI	Menu item

If the corresponding message file attribute is not specified on any of these tags when the `MSGID` attribute is specified for this tag, this attribute is required.

SUBMSGF= *qualified-message-file-name*
The default message file used when a message file is not specified on the `&msg` symbol.

PRTHEAD (Print Head Panel)

```
►►──:PRTHEAD──NAME──=──panel-name──┬──────────────────────────────────────┬────────────────────►
                                   └─PRODINFO──=──dialog-variable-name──┘

►──┬─────────────────────────────────┬──┬─────────────────────────────────┬────────────────────►
   └─PRTDATE──=──dialog-variable-name─┘  └─PRTTIME──=──dialog-variable-name─┘

►──┬───────────────────────────────┬──┬─────────────────────────┬──┬──────────────────────┬────►
   └─TITLE──=──dialog-variable-name─┘  └─TT──=──truth-table-name─┘  │             ┌─80──┐  │
                                                                   └─WIDTH──=──┴─132─┴──┘

►──┬─────────────────────────────┬──┬────────────────────────────────┬─────────────────────────►
   └─OBJ──=──dialog-variable-name─┘  └─OBJLIB──=──dialog-variable-name─┘

►──.──┬─────────────────┬──:EPRTHEAD.──────────────────────────────────────────────────────────►◄
      └─page-title-text─┘
```

A print head panel (PRTHEAD) tag contains tags defining heading information and data used with a UIM print application. A print heading panel establishes the prolog section printed after the title line on the first page and the header data printed at the top of every page. The prolog and header sections can be made up of data and information areas. A PRTHEAD panel causes an automatic page eject.

Other tags can be nested within the PRTHEAD tag. These tags are listed in the following table. The table defines the order in which the tags must appear, indicates which tags can be used in display panels only, print panels only, or both (specified by a D, P, or B, respectively), and specifies on which page more information can be found for each tag.

When more than one tag is listed with the same order number, all tags of that number can be mixed in any order. However, a tag with a higher order number cannot precede a tag with a lower order number. For example, a tag with an order number of three cannot precede a tag with an order number of one or two.

Table 81. Tags Allowed Between the PRTHEAD and EPRTHEAD Tags

Tag Name	Order	Use	Page
DATA (Data presentation area)	1	P	496
INFO (Information area)	1	P	537
PRTTRAIL (Print trail)	2	P	619

Required Attribute

NAME=*panel-name*
The name of the print head panel. For more information on the rules for naming, see "Name Syntax" on page 469.

Optional Attributes

PRODINFO=*dialog-variable-name*
The name of a dialog variable containing the product information printed on the second title line of the print output.

The dialog variable must be defined with a `BASETYPE` of CHAR or IGC on the class definition (CLASS) tag, and the maximum width is 22 bytes.

PRTDATE=*dialog-variable-name*
The name of a dialog variable containing the date printed on the page title. If this attribute is specified, the PRTTIME attribute must also be specified on this tag. If this attribute is not specified, the UIM uses the current system date when the Open Print Application (QUIOPNPA) API or the Add Print Application (QUIADDPA) API is called.

The dialog variable must be defined with a class that has a BASETYPE of DATE on the CLASS tag.

PRTTIME=*dialog-variable-name*
The name of a dialog variable containing the time printed on the page title. The dialog variable must be defined with a class that has a BASETYPE of TIME on the CLASS tag. If this attribute is specified, the PRTDATE attribute must also be specified on this tag. If this attribute is not specified, the UIM uses the local time when the QUIOPNPA API or the QUIADDPA API is called.

For an 80 column print width and a dialog variable that can contain a time zone, the time zone portion of this dialog variable is ignored. However, the time zone value can be displayed if the OBJ and OBJLIB attributes are omitted from this tag.

TITLE=*dialog-variable-name*
The name of a dialog variable containing the text for the page title line. The dialog variable must be defined on a CLASS tag with a display width of 55. A translation list may be used to provide the panel title or the width of the panel minus two, whichever is less. The dialog variable may also be defined with a BASETYPE of CHAR 55 or IGC 55 on the CLASS tag. The UIM trims trailing blanks from the string and centers the resulting text in the page title area. If this attribute is used, no *title-text* can be specified.

TT=*truth-table-name*
The name of a truth table specifying what combinations of truth values may occur at run time when the panel head is formatted.

The table specified on this tag may contain a subset of the conditions referred to by tags in the print head panel. Only the truth value combinations specified in the table, augmented by worst-case assumptions for any truth values not specified in the table, are considered in evaluating whether or not the print head panel is valid. The table should not exclude any truth value combination that could occur when the print head panel is formatted at run time. If a valid truth value combination is omitted, a panel group object may create without error and produce undesirable results when panels are printed.

If this attribute is omitted, all combinations of truth values are considered possible. This causes the tag language compiler to make worst-case assumptions for all conditions in evaluating whether or not the print head panel is valid.

WIDTH=80 | 132
Defines the width, in bytes, of the PRTHEAD panel. The default value is 80 bytes.

OBJ=*dialog-variable-name*
The name of a dialog variable containing the name of the object whose information is printed on the title line of the printout. If this attribute is not specified, an object name is not printed.

If the OBJLIB attribute on this tag is specified, the class for this dialog variable must be defined with a width of 10 characters. If the OBJLIB attribute is not specified, the class for this dialog must be defined with a width of 1 to 21 characters.

OBJLIB=*dialog-variable-name*
The name of a dialog variable containing the name of the library where the specified object from the OBJ attribute on this tag is found. This library name is printed on the title line of the printout. If this attribute is not specified, a library name is not printed.

You are responsible for the validity of the data contained in the OBJ and OBJLIB attributes. The UIM does not perform validity checking on these variables. If only the OBJ attribute is specified, only the OBJ is printed. However, if only the OBJLIB attribute is specified, a message is issued at compile time.

The class for this dialog variable must be defined with a width of 10 characters.

Optional Text

page-title-text
The title used on the title line. The text must appear on the same line or next line as the tag. The text can only contain the reverse text (RT) tag, and cannot exceed 55 characters in length. The *page-title-text* is required unless the TITLE attribute is specified on this tag.

Layout of the Title Lines

The title lines for the PRTHEAD tag appear on the top two lines of each printed page. The placement of fields in the title lines depends on the width specified on this tag.

Table 82. First Line of Heading with Print Width 132

Data	Start Column	Length
Title	See note.	Max. 55
Separator	See note.	4
″Page″ (with expansion)	See note.	Max. 11
Separator	125	2
Page number	127	4
Separator	131	2

Note: The start column for this data is variable. The title is centered between the left margin and the separator following the title.

Table 83. First Line of Heading with Print Width 80

Data	Start Column	Length
Title	See note.	Max. 55
Separator	See note.	4
″Page″ (with expansion)	See note.	Max. 11
Separator	73	2
Page number	75	4
Separator	79	2

Note: The start column for this data is variable.

Table 84. Second Line of Heading with Print Width 132

Data	Start Column	Length
Product information	1	22
Separator	23	Min. 55
Library	See note.	Max. 10
Slash	See note.	1
Object	See note.	Max. 10
Separator	99	4
System name	103	8
Separator	111	2
Date	113	8
Separator	121	2

Table 84. Second Line of Heading with Print Width 132 (continued)

Data	Start Column	Length
Time	123	8
Separator	131	2

Note: The library and object are formatted as OBJLIB/OBJ with trailing blanks stripped from the OBJLIB and OBJ values. The OBJLIB/OBJ aggregate is right-justified against the separator before the system name.

Table 85. Second Line of Heading with Print Width 132 and time zone

Data	Start Column	Length
Product information	1	22
Separator	23	Min. 55
Library	See notes 1 and 2.	Max. 10
Slash	See notes 1 and 2.	1
Object	See notes 1 and 2.	Max. 10
Separator	See note 2.	4
System name	See note 2.	8
Separator	See note 2.	2
Date	See note 2.	8
Separator	See note 2.	2
Time	See note 2.	8
Separator	See note 2.	1
Time zone	See note 2.	Max. 10
Separator	131	2

Notes:

1. The library and object are formatted as OBJLIB/OBJ with trailing blanks stripped from the OBJLIB and OBJ values. The OBJLIB/OBJ aggregate is right-justified against the separator before the system name.
2. Trailing blanks are stripped from the time zone and all information starting with the library up to and including the time zone is right-justified against the separator before the end of the heading.

Table 86. Second Line of Heading with Print Width 80

Data	Start Column	Length
Product code	1	Max. 22
Separator	23	Min. 3
Library	See note.	Max. 10
Slash	See note.	1
Object	See note.	Max. 10
Separator	47	4
System name	51	8
Separator	59	2
Date	61	8
Separator	69	2
Time	71	8
Separator	79	2

Note: The library and object are formatted as OBJLIB/OBJ with trailing blanks stripped from the OBJLIB and OBJ values. The OBJLIB/OBJ aggregate is right-justified against the separator before the system name.

In order to print a time zone on a panel with an 80 column print width, the OBJ and OBJLIB attributes must be omitted.

Table 87. Second Line of Heading with Print Width 80 and time zone

Data	Start Column	Length
Product code	1	Max. 22
Separator	23	Min. 1
System name	See note.	8
Separator	See note.	2
Date	See note.	8
Separator	See note.	2
Time	See note.	8
Separator	See note.	1
Time zone	See note.	Max. 10
Separator	See note.	2

Note: Trailing blanks are stripped from the time zone, and all information starting with the system name up to and including the time zone is right-justified against the separator before the end of the heading.

Example: Print Title Line

The following example has a title line for output with a width of 132 bytes. The title is from the print head panel or the first print panel. The page text and page number are provided by the UIM.

Either the UIM or the application can provide the date and time. The application must use the PRTDATE and PRTTIME attributes on the PRTHEAD tag to specify a date and time. If the PRTDATE and PRTTIME attributes are not specified, the UIM uses the system date and time when the QUIOPNPA API or the QUIADDPA API is called. The date and time are formatted using the job attributes.

```
                        This is an Example of a Print Title Line                         Page     1
5722SS1 V5R4M0  060210                             QNETUSER/EXAMPLE    SYSTEM01  02/10/06  11:09:04
```

PRTPNL (Print Panel)

```
►►─:PRTPNL──NAME──=──panel-name──┬──────────────────────────────┬──┬────────────────────────┬──────────►
                                 └─TITLE──=──dialog-variable-name─┘  └─TT──=──truth-table-name─┘

►─┬────────────────────────────────────────────────────┬─────────────────────────────────────────►◄
  │            ┌─80──┐                                 │
  └─WIDTH──=───┴─132─┴───.──┬────────────┬──:EPRTPNL.─┘
                            └─title-text─┘
```

A print panel (PRTPNL) tag contains tags to define one or more area definitions for a print panel. It requires a matching EPRTPNL tag.

Other tags can be nested within the PRTPNL tag. These tags are listed in the following table. The table defines the order in which the tags must appear, indicates which tags can be used in display panels only, print panels only, or both (specified by a D, P, or B, respectively), and specifies on which page more information can be found for each tag.

When more than one tag is listed with the same order number, all tags of that number can be mixed in any order. However, a tag with a higher order number cannot precede a tag with a lower order number. For example, a tag with an order number of three cannot precede a tag with an order number of one or two.

Table 88. Tags Allowed Between the PRTPNL and EPRTPNL Tags

Tag Name	Order	Use	Page
DATA (Data presentation area)	1	P	496
INFO (Information area)	1	P	537
LIST (List area)	1	P	552

Required Attribute

NAME=*panel-name*
The name of the print panel. For more information on the rules for naming, see "Name Syntax" on page 469.

Optional Attributes

TITLE=*dialog-variable-name*
The name of a dialog variable containing the text for the panel title when the panel is printed. The dialog variable must be defined with a class that has a width of 55. The UIM trims trailing blanks from the string and centers the resulting text in the title line of the print panel. If this attribute is used, no *title-text* can be specified.

TT=*truth-table-name*
The name of a truth table that specifies what combinations of truth values may occur at run time when the panel is printed.

The table specified on this tag may contain a subset of the conditions referred to by tags in the print panel. Only the truth value combinations specified in the table, augmented by worst-case assumptions for any truth values not specified in the table, are considered in evaluating whether or not the print panel definition is valid. The table should not exclude any truth value combination that could occur when the print panel is printed. If a valid truth value combination is omitted, a panel group object may create without error and produce undesirable results when panels are printed.

If this attribute is omitted, all combinations of truth values are considered possible. This causes the tag language compiler to make worst-case assumptions for all conditions in evaluating whether or not the print panel is valid.

WIDTH=80 | 132

Defines the width, in bytes, of the print panel. The default value is 80 bytes.

Optional Text

title-text

The title used on this print panel. The text must appear on the same or next line as the tag. The text can only contain the reverse text (RT) tag, and cannot exceed 55 characters in length. The *title-text* is not allowed if the TITLE attribute is specified on this tag.

PRTTRAIL (Print Trailer Message)

```
►►──:PRTTRAIL.──trailer-message-text──────────────────────────────────────────────►◄
```

The print trailer message (PRTTRAIL) tag is allowed only within the print head panel (PRTHEAD) tag, appearing immediately before the EPRTHEAD tag. The trailer message is printed at the end of the print listing. If no trailer is specified, no default trailer is printed.

Note: The trailer is only printed when the printer file is closed by using the option for a normal close on either the Close Application (QUICLOA) API or the Remove Application (QUIRMVPA) API. The trailer is not printed if the application is closed using the option for an abnormal close.

Required Text

trailer-message-text

The text appearing at the end of the print listing. This text can be entered without leading or trailing asterisks (*). The UIM adds asterisks before and after the text.

The text can be a maximum of 106 bytes for WIDTH=132 and a maximum of 54 bytes for WIDTH=80, specified on a PRTHEAD tag. The text can contain only the reverse text (RT) tag.

Example: Trailer Message

This example shows what kind of text a print trailer message might have.

UIM Source

```
:prttrail.E N D  O F  L I S T I N G
```

PV (Programming Variable)

```
►►──:PV.──programming-variable──:EPV.─────────────────────────────────────────────►◄
```

The programming variable (PV) tag identifies a programming variable. It requires a matching end tag. These tags are only allowed in information areas and help areas. It is frequently used within parameter lists. For more information on parameter lists, see “PARML (Parameter List)” on page 602.

A programming variable can occur anywhere in the text to help explain the elements of programming syntax.

The PV and EPV tag phrase must be specified on word boundaries. If the two characters immediately following the EPV tag are a punctuation mark and a blank, the UIM automatically extends the emphasis attribute to include the punctuation mark. This allows the punctuation mark and the text associated with it to be displayed using the same emphasis.

Required Text

programming-variable
The programming variable. The text formats in highlight phrase 1 (HP1).

RT (Reverse Text)

```
►►—:RT.—reverse-direction-text—:ERT.——————————————————————————————————►◄
```

The reverse text (RT) tag indicates that the enclosed text has an orientation opposite to the orientation of the panel group. This tag is ignored for panel groups with BIDI=NONE specified on the panel group (PNLGRP) tag. On a BIDI=RTL panel group, text in a left-to-right language is placed between the RT and ERT tags, while on a BIDI=LTR panel group, text in a right-to-left language is placed between the RT and ERT tags.

With the import (IMPORT) tag, it is possible to combine help information from panel groups that have BIDI=RTL specified with help information from panel groups that have BIDI=LTR specified. This is not recommended, but when it does occur, the user needs to use the hot key sequence for the work station controller to flip the screen to read the help information with the opposite orientation. Any text within the RT and ERT tags is formatted correctly for its panel group orientation. Therefore, when the RT tag is used for text that is imported into help information with the opposite orientation, the imported text is also readable when the screen is flipped.

Unlike the highlight phrase tags, the RT tag is allowed within the text of most tags. The following tags allow an imbedded RT tag:

BOTINST	Bottom instructions
CIT	Title citation
CMDLINE	Command line
DATA	Data area
DATASLTC	Data selection choices
DATAGRP	Data group
DATAI	Data item
DD	Definition
DDHD	Definition header
DT	Definition term
DTHD	Definition term header
FIGCAP	Figure caption text
Hn	Headings
HPn	Highlighted phrase
INFO	Information area
KEYI	Key list item

LINK	Hypertext link definition
LIST	List area
LISTACT	List action
LISTGRP	List group
LI	List item
LP	List part
MBARC	Menu bar choice
MENU	Menu area
MENUI	Menu item
NT	Note
OPTLINE	Option line
P	Paragraph
PC	Paragraph continuation
PD	Parameter description
PT	Parameter term
PRTHEAD	Print head panel
PRTPNL	Print panel
PRTTRAIL	Print trailer message
PDFLDC	Pull-down field choice
TOPINST	Top instruction line

The RT tag cannot be used within a figure (FIG), unformatted lines (LINES), or an example (XMP) tag. No tag is allowed between an RT tag and its matching ERT tag.

Blanks found between an RT tag and its matching ERT tag are preserved. This includes the trailing blank at the end of each source line. Therefore, starting a source line with an ERT tag is not recommended.

Example 1: Left-to-Right Formatting on a Right-to-Left Panel

This example illustrates left-to-right source formatted and displayed on a right-to-left panel.

UIM Source

```
:P.For more information see the
:CIT.:RT.IBM i5/OS User Interface
Manager Reference Manual:ERT.:ECIT.
which you can find in your library.
```

Results

```
IBM i5/OS User eht ees noitamrofni erom roF
uoy hcihw Interface Manager Reference Manual
                   .yrarbil ruoy ni dnif nac
```

Example 2: Left-to-Right Formatting on a Left-to-Right Panel

This example illustrates left-to-right source formatted and displayed on a left-to-right panel.

UIM Source

```
:P.This is an example of some
:RT.PRETEND HEBREW TEXT:ERT.
imbedded within some real
English text.
```

Results

```
This is an example of some DNETERP
TXET WERBEH imbedded within some
real English text.
```

SL (Simple List)

```
                         ┌──────────────────────────┐
                         ▼                          │
►►──:SL──┬─────────┬──.───:LI.──item-text──:ESL.──┴──────────────────────────────►◄
         └─COMPACT─┘
```

The simple list (SL) tag identifies a list of items. It requires a matching end tag. These tags are only allowed in information areas and help areas.

Simple lists can occur anywhere in text and can be nested within other lists.

Care should be taken when using the unformatted lines (LINES), figure (FIG), and example (XMP) tags within simple lists, because text that does not fit on one line wraps to column one of the next line. Lines and figures start at the current left margin, and examples are indented four spaces from the current left margin. The current left margin changes when nested lists are formatted. If there is help information containing the LINES, FIG, or XMP tags imbedded at various locations, including within lists, it may not look the same each time and can cause undesirable results.

Simple lists are formatted as hanging, indented lists, with no item identifier.

Other tags can be nested within the SL tag. These tags are listed in the following table. The table defines the order in which the tags must appear, indicates which tags can be used in display panels only, print panels only, or both (specified by a D, P, or B, respectively), and specifies on which page more information can be found for each tag.

When more than one tag is listed with the same order number, all tags of that number can be mixed in any order. However, a tag with a higher order number cannot precede a tag with a lower order number. For example, a tag with an order number of three cannot precede a tag with an order number of one or two.

Table 89. Tags Allowed Between the SL and ESL Tags

Tag Name	Order	Use	Page
FIG (Figure)	1	B	527
LINES (Unformatted lines)	1	B	546
XMP (Example)	1	B	639
NT (Note)	1	B	590
P (Paragraph)	1	B	593
PC (Paragraph continuation)	1	B	603
LP (List part)	1	B	579

Table 89. Tags Allowed Between the SL and ESL Tags (continued)

Tag Name	Order	Use	Page
OL (Ordered list)	1	B	591
SL (Simple list)	1	B	622
UL (Unordered list)	1	B	633
PARML (Parameter list)	1	B	602
DL (Definition list)	1	B	525

Optional Attribute

COMPACT
The list is formatted with no blank line between the items.

Required Tag

:LI.*item-text*
The text for the list item. The text is indented four spaces from the current margin.

Example: Simple Lists

This example uses two simple lists, one imbedded within the other. The second simple list uses the `COMPACT` attribute.

UIM Source

```
Some normal text...
:sl.
:li.First item
:sl.
:li.First item
:li.Second item
:esl.
:li.Second item
:esl.
```

Results

```
Some normal text...

First item

    First item
    Second item

Second item
```

TEXT (Text Area)

```
►►──:TEXT──VAR──=──dialog-variable-name──USREXIT──=──'CALL program-reference' ──────────────────────►

►─┬──────────────────────────────────────────────────────────────────┬─.─────────────────────────►◄
  └─ROW──=──dialog-variable-name──  ──COL──=──dialog-variable-name─┘
```

The text area (TEXT) tag defines a text area in a panel to be formatted by the application. Only one TEXT tag can be specified in each panel. Menu bars and command lines are allowed, however, no other areas can be defined in the panel.

The width of this area is determined by the WIDTH attribute of the PANEL tag which this area is defined in.

The depth of this area is the depth of the panel. The depth of the panel is determined by the DEPTH attribute of the PANEL tag which this area is defined in. The DEPTH value is reduced by any lines used for other parts of the panel such as a menu bar lines, the panel title line, the command lines, the function key line(s), and the message line(s). See the PANEL tag for a description of these dimensions.

The data passed to the UIM to be displayed in this area is described in the "Text Data" on page 625 section below.

If the Help key is pressed while the cursor is in the text area extended help for the panel is displayed.

Required Attribute

VAR=*dialog-variable-name*

The name of the dialog variable containing the data, or a pointer to the data, for the text area. Define the dialog variable using a class definition (CLASS) tag with basetype of CHAR X or PTR. The CHRID=PNLGRP attribute is not allowed on the class definition for this dialog variable.

The data in the dialog variable is viewed by the UIM as a *d* by *w* array of characters, where *d* is the depth and *w* is the width of the text area. Each row of the array is displayed as one line of the text area.

If the *dialog-variable-name* has a BASETYPE of CHAR on its CLASS tag, this dialog variable contains the data for the text area.

If the *dialog-variable-name* has a BASETYPE of PTR on its CLASS tag, this dialog variable contains a pointer to the data for the text area. The data pointed to by this pointer is viewed by UIM the same as for a dialog variable with a BASETYPE of CHAR. Any exception the UIM receives while accessing this data will result in an escape message sent to the calling program.

The dialog variable can be declared as any size. If there is not enough data to fill the text area the remainder of the text area will be blank. If there is more than enough data to fill the text area only the data needed to fill the text area will be used.

You can use the exit program for the text area, specified on the USREXIT attribute of this tag, to update this dialog variable each time the panel is displayed. The UIM uses the value of the dialog variable to work with the data for the text area as described in "Text Data" on page 625. The value of the dialog variable depends on the BASETYPE attribute on the CLASS tag of the dialog variable.

The size of the text area can vary depending on the number of rows taken up by the function keys. This is important to remember when coding for multiple languages or conditioning function keys with the condition (COND) tag.

USREXIT='*CALL program-reference***'**

The exit program called to update the value of the dialog variable on the VAR attribute each time the panel is displayed.

This area will be considered scrollable by the UIM. If the user presses a key assigned to a scrolling dialog command and the text area should be scrolled (as opposed to other scrollable areas on the panel) the UIM will call the user exit program. On return the UIM will redisplay the panel. For a description of the user exit program structure passed by the UIM, see the **APIs** topic in the iSeries Information Center.

A general panel exit program should be used to diagnose if the user has scrolled too far. If the user has scrolled too far the general panel exit should send an appropriate message followed by the special message to cancel the determined action. For a description of the general panel exit program structure passed by the UIM, see the **APIs** topic in the iSeries Information Center. If this is done the message telling the user he has scrolled to far will only be seen once. If the text area exit sends the message it may be displayed on the screen longer than desired.

For a description of the CALL dialog command, see Appendix B, "UIM Dialog Commands," on page 641.

Optional Attributes

ROW=*dialog-variable-name*
The row of the cursor. Define the dialog variable using a class definition (CLASS) tag with BASETYPE of BIN(31).

The dialog variables for the row and column can be set by the application before calling the UIM to set the location of the cursor within the text area. The first character of the first row in the text area would be row 1 and column 1. A value of zero in either of the dialog variables tells the UIM to use default cursor positioning.

When the UIM returns to the application, it will set these dialog variables to the row and column where the cursor was positioned in the text area when the user exited the panel. If the cursor was not within the text area when the user exited the panel the row and column dialog variables will both be zero.

COL=*dialog-variable-name*
The same as ROW except this is the column of the cursor.

Cursor positioning

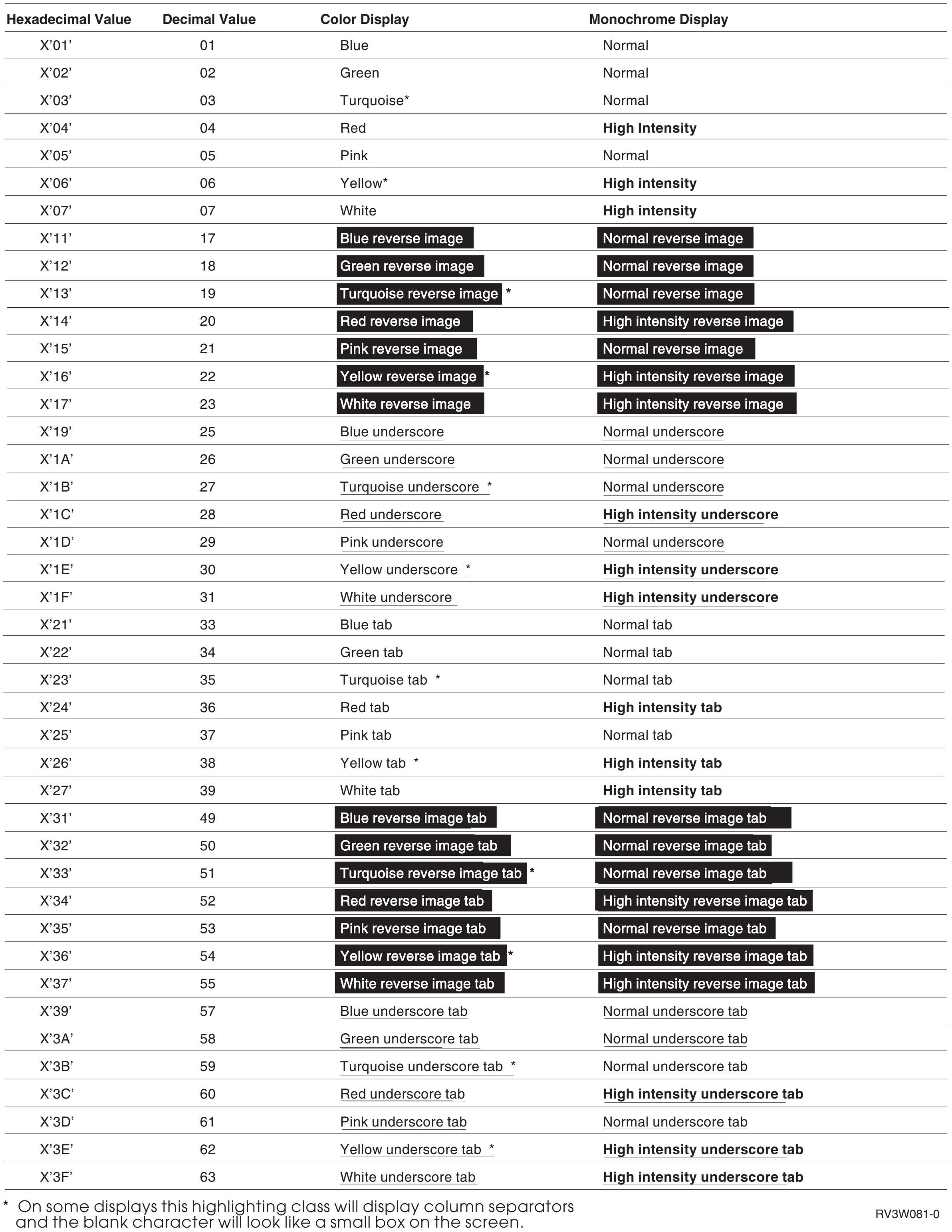
The cursor position in a panel with a text area depends on many things. The following is a prioritized list for where the cursor will be positioned. The first item that holds true will position the cursor.

1. If the ROW and COL attributes are used on the TEXT tag and the value of the dialog variables are valid when the panel is displayed, the cursor is positioned at that row and column.
2. If the panel has a command line the cursor is positioned at the command line.
3. If the panel has a menu bar the cursor is positioned at the first menu bar choice.
4. If the text area has tabable highlighting classes the cursor is positioned at the first one.
5. If none of the above position the cursor the cursor will be positioned at the upper left corner of the panel or pop-up window.

Text Data

Any character within the data can be a selection character for a highlighting class. Only output data is allowed in a text area; no input fields are allowed. The highlighting of the text applies to the characters following the class selection character up to another class selection character or the end of the area, whichever occurs first.

The following table describes the selection characters for highlighting that are recognized by the UIM:



Hexadecimal Value	Decimal Value	Color Display	Monochrome Display
X'01'	01	Blue	Normal
X'02'	02	Green	Normal
X'03'	03	Turquoise*	Normal
X'04'	04	Red	High Intensity
X'05'	05	Pink	Normal
X'06'	06	Yellow*	High intensity
X'07'	07	White	High intensity
X'11'	17	Blue reverse image	Normal reverse image
X'12'	18	Green reverse image	Normal reverse image
X'13'	19	Turquoise reverse image *	Normal reverse image
X'14'	20	Red reverse image	High intensity reverse image
X'15'	21	Pink reverse image	Normal reverse image
X'16'	22	Yellow reverse image *	High intensity reverse image
X'17'	23	White reverse image	High intensity reverse image
X'19'	25	Blue underscore	Normal underscore
X'1A'	26	Green underscore	Normal underscore
X'1B'	27	Turquoise underscore *	Normal underscore
X'1C'	28	Red underscore	High intensity underscore
X'1D'	29	Pink underscore	Normal underscore
X'1E'	30	Yellow underscore *	High intensity underscore
X'1F'	31	White underscore	High intensity underscore
X'21'	33	Blue tab	Normal tab
X'22'	34	Green tab	Normal tab
X'23'	35	Turquoise tab *	Normal tab
X'24'	36	Red tab	High intensity tab
X'25'	37	Pink tab	Normal tab
X'26'	38	Yellow tab *	High intensity tab
X'27'	39	White tab	High intensity tab
X'31'	49	Blue reverse image tab	Normal reverse image tab
X'32'	50	Green reverse image tab	Normal reverse image tab
X'33'	51	Turquoise reverse image tab *	Normal reverse image tab
X'34'	52	Red reverse image tab	High intensity reverse image tab
X'35'	53	Pink reverse image tab	Normal reverse image tab
X'36'	54	Yellow reverse image tab *	High intensity reverse image tab
X'37'	55	White reverse image tab	High intensity reverse image tab
X'39'	57	Blue underscore tab	Normal underscore tab
X'3A'	58	Green underscore tab	Normal underscore tab
X'3B'	59	Turquoise underscore tab *	Normal underscore tab
X'3C'	60	Red underscore tab	High intensity underscore tab
X'3D'	61	Pink underscore tab	Normal underscore tab
X'3E'	62	Yellow underscore tab *	High intensity underscore tab
X'3F'	63	White underscore tab	High intensity underscore tab

* On some displays this highlighting class will display column separators and the blank character will look like a small box on the screen.

RV3W081-0

Figure 146. Highlighting Classes Allowed in TEXT Area

The UIM replaces each class selection character with the appropriate display attributes for the class. Other character values in the ranges X'10' through X'3F', as well as X'FF', are converted to X'1F', appearing as a reverse image box on the screen. Characters X'00' (null), X'0E' (shift-out for double-byte), X'0F' (shift-in for double-byte), and X'40' through X'FE' (normal, displayable characters), are passed unchanged to the screen.

For best appearance you should not place text in the first and last column of a panel. This also allows you to highlight the first word in any row easily.

If you are trying to create a tabable phrase of a particular color do not use blanks between the words of the phrase. You should use the non-tabable highlighting class of the same color. This will give you the most effective display on all work stations.

When a panel is displayed with tabable highlighting classes in the text area the cursor is not automatically located on the first one. If you want this to happen you must use the ROW and COL attributes of this tag to set the location of the cursor.

If one or more tabable highlight classes are placed next to each other in the data only the last one will allow tabbing. The others will be ignored. The maximum number of tabable fields is 240. It may be less if there are other input capable or tabable fields on the display such as the command line or menu bar choices.

If any highlighting class is placed in the last column of the last row in a full screen panel or in the last column of any row in a pop-up window, it will be ignored.

For pop-up windows highlighting classes never continue on the next line. You must use another highlighting class if you want it to continue.

If a panel smaller than the device is displayed without adding a pop-up window first, highlighting classes continue into the blank space to the right of the panel on the screen. To turn off these highlighting classes use other highlighting classes at the end of each row that look like blank space.

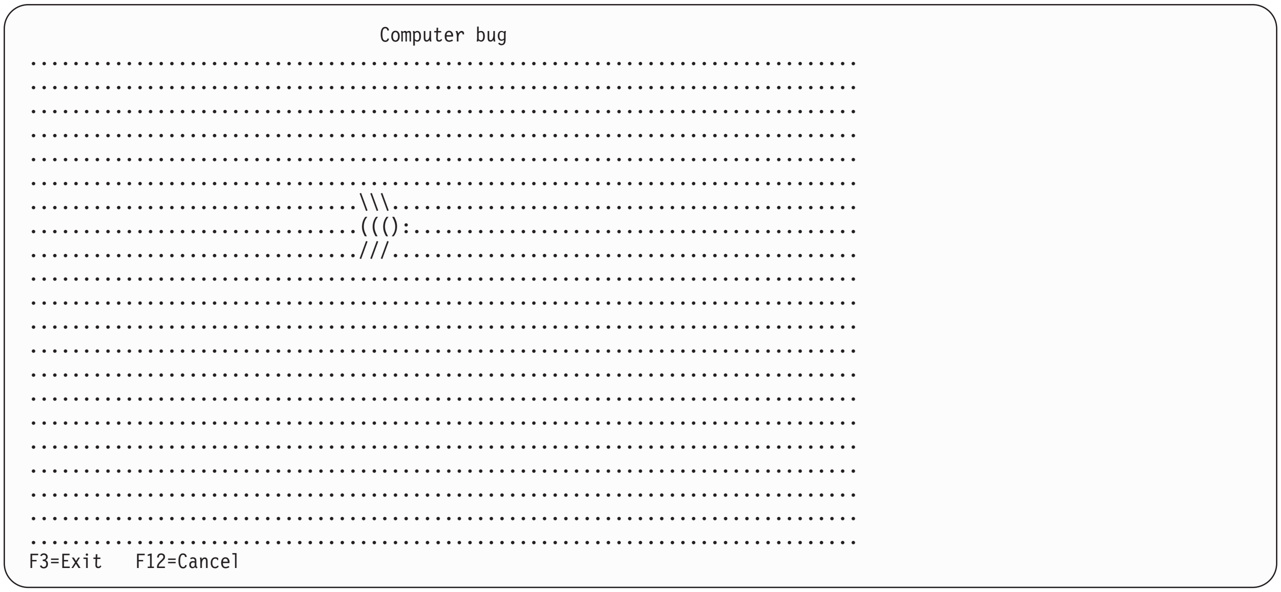
No character set and code page conversion is done on the text area data. The application should provide the data in the correct character set and code page for the device.

When the UIM calls the exit program to format the text area, it passes a value which identifies the BIDI attribute on the panel group (PNLGRP) tag and the code page number of the display device.

If the NBRSHAPE and SYMSWAP attributes are used on the CLASS tag for the text area dialog variable they will be ignored.

If DBCS data is used the application is fully responsible to make sure it displays properly. An error will occur if you attempt to display half of a DBCS character or if one of the IGC shift characters (X'0E', X'0F') is missing.

Example: Text area



```
                                 Computer bug
..............................................................................
..............................................................................
..............................................................................
..............................................................................
..............................................................................
..............................................................................
...............................\\\............................................
...............................((():..........................................
...............................///............................................
..............................................................................
..............................................................................
..............................................................................
..............................................................................
..............................................................................
..............................................................................
..............................................................................
..............................................................................
..............................................................................
..............................................................................
..............................................................................
..............................................................................
F3=Exit   F12=Cancel
```

TI (Translation List Item)

```
►►──:TI──┬─────────────────────────────┬──.──┬───────────────┬──────────────────────────────►◄
         └─VALUE──=──'internal-value'──┘     └─displayed-value─┘
```

The translation list item (TI) tag specifies a list item in a translation list. This tag must occur between translation list (TL) and ETL tags. The item is translated on output to the panel and on input to the variable pool.

Optional Attribute

VALUE='*internal-value***'.**

The internal value for the dialog variable in the variable pool. This internal value is not subject to validity checks defined by the validity checking (CHECK) tag for this class. If the same internal value is specified more than once in the list, the first item in the list is chosen for translation on output.

If the `BASETYPE` of this class is BIN, as specified on the class definition (CLASS) tag, the value may be specified in either integer or hexadecimal notation. If the `BASETYPE` specified on the CLASS tag is CHAR, IGC, DATE, TIME, NAME, or OBJNAME, the value may be specified in either string or hexadecimal string notation. Strings are implicitly padded on the right with blanks, but no implicit padding is done for hexadecimal notation.

If the internal value is specified in string or hexadecimal notation, the value must be enclosed in quotation marks in addition to apostrophes. For example, an internal value of ***YES** must be specified as `VALUE=`'"*YES"'.

| If the `BASETYPE` of this class is TIME, a time zone value must not be specified.

Omitting the `VALUE` attribute leaves the internal dialog variable value unchanged when the user enters the *displayed-value* specified as tag content after the period. A TI tag of this type has no effect on the way internal values are presented to the user for display.

Optional Text

displayed-value

The value appearing on the panel or entered by the user. The comparisons done by the UIM to determine what internal value to use for an external value specified by the user may be changed by the CASE and BLANKS attributes of the CLASS tag. If the same external value appears more than once in the list, the first item is used for translation on entry. If no value is specified, a display value of blanks is used.

TL (Translation List)

```
►►──:TL──┬─────────────────────────┬──┬────────────────────────────────┬──────────►
         │             ┌─UPPER─┐   │  └─MSGID──=──message-identifier─┘
         └─CASE──=─────┴─MIXED─┴───┘

►──┬────────────────────────────────────────────┬──.──:ETL──────────────────────►◄
   └─MSGF──=──'qualified-message-file-name'─────┘
```

The translation list (TL) tag allows the specification of a translation list for a class, shielding the variable pool from the values actually entered on the display. Values with a BASETYPE of CHAR, IGC, DATE, TIME, NAME, OBJNAME, or that are numeric on the class definition (CLASS) tag can have a translation list. Only one translation list is allowed for each class. Translation occurs before any validity checks are performed.

Other tags can be nested within the TL and ETL tags. These tags are listed in the following table. This table defines the order in which the tags must appear and specifies on which page more information can be found for each tag.

Table 90. Tag Allowed Between the TL and ETL Tags

Tag Name	Order	Page
TI (Translation list item)	1	628

Optional Attributes

CASE=UPPER | MIXED

UPPER indicates that lowercase characters (a through z) in an input field value are converted to uppercase characters (A through Z) before comparison to the external value for each translation list item. This conversion to uppercase occurs after any conversion for the CASE attribute on the CLASS tag, but has no effect on the value assigned to the dialog variable or list entry if no matching translation list item is found. It also does not change the external display value specified on each translation list item (TI) tag; the text on the TI tag must be specified in uppercase to get a match.

MIXED indicates that an input value is compared as it is entered with the external value specified on each TI tag.

MSGID=*message-identifier*

The message identifier of the message sent if the translation fails because the user entered a value not specified in the list. If this attribute is not specified, the UIM assumes that the entered value is valid as it is entered and passes the value on to validity checking for the class.

MSGF='*qualified-message-file-name***'**

The message file containing the message sent for this error if the translation fails. If the DFTMSGF attribute is not specified on the panel group (PNLGRP) tag, this attribute must be specified if the MSGID attribute is specified on this tag.

Examples: Translation List

This example shows how a translation list is defined.

UIM Source

```
:tl msgid=MMM0001 msgf='*LIBL/QCPFMSG'.
:ti value=1.Blue
:ti value=1.BLUE
:ti value=2.Red
:ti value=2.RED
:ti value=3.Green
:ti value=3.GREEN
:ti value=4.Turquoise
:ti value=4.TURQUOISE
:etl.
```

On entry, *Blue*, due to the default CASE=UPPER and the uppercase version of the translation item, is mapped to the value 1. On output, 1 is mapped to the value *Blue*, which is mixed case, because the first matching translation list item is always used. Notice that because the integer notation is used for the value, the BASETYPE on the CLASS tag that this translation list applies to must be numeric.

This is an example of a class definition and a translation list for a dialogue variable that allows a valid i5/OS library name (*LIBL, *CURLIB, or blanks).

UIM Source

```
:class name=objlibc
      basetype='objname 10'.
:tl.
:ti value='"*LIBL"'.*LIBL
:ti value='"*CURLIB"'.*CURLIB
:ti value='"      "'.
:etl.
:eclass.
```

TOPINST (Top Instruction)

```
►►─:TOPINST──┬─────────────────────────────┬─.─┬──────────────────┬──────────────►◄
             └─INST──=──dialog-variable-name─┘   └─instruction-text─┘
```

The top instruction (TOPINST) tag of an area specifies the top instruction lines. This tag is valid for all areas, but is allowed only for display panels. It appears immediately after the introducing tag for the area.

Multiple top instruction tags may be used to present multiple lines of instructions if the TOPINST tag contains *instruction-text*. If multiple TOPINST tags are coded, no blank lines appear between the text on different tags. Only one top instruction tag is allowed per area if an INST attribute variable is used for this tag.

For menus, a blank line is always left between the body and the instruction lines area. For more information about how instruction lines are formatted with respect to the body of certain areas, see the BODYSEP attribute in "DATA (Data Presentation Area)" on page 496.

If no TOPINST tag is specified for an area, no text is used for the instruction line and the line is not allocated in the area.

Optional Attribute

INST=*dialog-variable-name*

The name of a dialog variable containing the top instruction text displayed. The dialog variable must be defined with a width less than or equal to the width specified on the panel tag minus two. If this attribute is used, no *instruction-text* can be specified for this tag.

Dialog variables must be defined with a BASETYPE of CHAR, IGC, or BIN on the class definition (CLASS) tag.

The error state of the dialog variable is not used for determining the highlighting of the text.

Special formatting for IGC. The abbreviation IGC is used in commands and keywords to represent double-byte character set (DBCS) functions. When a dialog variable with a BASETYPE of IGC is specified on the CLASS tag, the UIM does special formatting. If the variable value begins with a shift-out character (X'0E'), the UIM shifts the value 1 byte to the left to preserve vertical alignment with other lines.

Optional Text

instruction-text

The text appearing as top instructions for the area. The text is an implied paragraph. When the display is formatted, any text that does not fit onto one display line is formatted onto additional lines as necessary and indented two positions. The text can be a maximum of 255 characters, and can only contain the reverse text (RT) tag. The *instruction text* is required unless the INST attribute is specified on this tag.

TT (Truth Table)

```
►►—:TT—NAME—=—truth-table-name—CONDS—=—'—condition-name-list—'—.—:ETT.————————————►◄
```

The truth table (TT) tag begins the definition of a truth table used by the compiler when processing panel definitions. The TT tag must occur in the prolog section of the panel group after the condition definition (COND) tags and before any menu bar (MBAR) tags. You should use a truth table to assert to the UIM which conditions are mutually exclusive.

During compilation of a panel group, a panel is formatted once for each row of a truth table. Using the truth value for each condition, the compiler checks if the panel definition is valid. If a panel element is conditioned using the COND attribute on this tag, and if that condition is not mentioned in the truth table, it is assumed to have a true value by the compiler.

The name of a truth table must be specified on the TT attribute of the display panel (PANEL), print panel (PRTPNL), or print head panel (PRTHEAD) tag for the compiler to use the truth table for panel verification.

After the compile, if the truth table is wrong and the conditioning causes more panel elements to be active in an area than can be displayed, only the elements that fit are displayed, and no exception is reported.

Other tags can be nested within the TT tag. These tags are listed in the following table. This table defines the order in which the tags must appear and specifies on which page more information can be found for each tag.

Table 91. Tag Allowed Between the TT and ETT Tags

Tag Name	Order	Page
TTROW (Truth table row)	1	632

Required Attributes

NAME='*truth-table-name***'**
The name of the truth table. For more information on the rules for naming, see “Name Syntax” on page 469. This name is referred to by all panels which use this truth table for compile-time checking. This name must be unique within the panel group.

CONDS='*condition-name-list***'.**
A list of condition names separated by blanks. Each condition in the list must have been previously defined with the COND tag. The rows of the truth table are defined with the truth table row (TTROW) tag. Each item in the VALUES attribute of the TTROW tag corresponds positionally with the condition specified in this attribute. A maximum of 50 condition names can be specified.

Example: Truth Table

If conditions A, B, and C are known to be mutually exclusive on panel X, this example would illustrate how to assert this to the compiler.

UIM Source

```
:tt name=tt1 conds='A B C'.
:ttrow      values='1 0 0'.
:ttrow      values='0 1 0'.
:ttrow      values='0 0 1'.
:ett.
```

TTROW (Truth Table Row)

```
►►—:TTROW—VALUES—=—'—condition-value-list—'—.——————————————————————————————►◄
```

The truth table row (TTROW) tag defines a row in a truth table used by the compiler when processing panel definitions. The TTROW tag must occur between the truth table (TT) and ETT tags in the prolog section of the panel group.

During compilation of a panel group, a panel is formatted once for each row of a truth table, and the truth value for each condition is used to check if the panel definition is valid. A truth table can describe which of the conditions that control the formatting of a panel are mutually exclusive. If a condition is used within the panel definition and is not mentioned in the truth table, it is assumed to be true for the purposes of compile time checking; this is the same as specifying an asterisk (*) in the VALUES attribute on this tag.

Required Attribute

VALUES='*condition-value-list***'.**
The value of each condition used by the compiler to check the panel definition. Each item in the VALUES attribute of the TTROW tag corresponds positionally with the condition specified in this attribute. The following values can be used for the condition:

0 The condition is false.

1 The condition is true

***** No truth value is specified. The compiler assumes the value is true for a worst case.

The items in the list of condition values are separated by blanks. The number of items in the list must equal the number of items in the CONDS attribute of the TT tag.

UL (Unordered List)

```
                          ┌───────────────────────────┐
                          ▼                           │
►►──:UL──┬─────────┬─.────:LI.──item-text──:EUL.──────┴────────────────────►◄
         └─COMPACT─┘
```

The unordered list (UL) tag identifies an unordered list of items. The UL tag requires a matching end tag. These tags are only allowed in information areas and help areas.

Unordered lists can occur anywhere in text and can be nested within other lists.

Note: Care should be taken when using the unformatted lines (LINES), figure (FIG), and example (XMP) tags within unordered lists, because text that does not fit on one line wraps to column one of the next line. Lines and figures start at the current left margin and examples are indented four spaces from the current left margin. The current left margin changes when nested lists are formatted. If there is help information containing the LINES, FIG, or XMP tag imbedded at various locations, including within lists, it may not look the same each time and can cause undesirable results.

Unordered lists are formatted as hanging, indented lists, with the item identifier at the current left margin. There are three levels of item identifiers for nested, unordered lists:

- Bullets (lowercase letter o)
- Hyphens (-)
- Dashes (--).

These levels are repeated for more than three levels of nested, unordered lists.

Other tags can be nested within the UL tag. These tags are listed in the following table. The table defines the order in which the tags must appear, indicates which tags can be used in display panels only, print panels only, or both (specified by a D, P, or B, respectively), and specifies on which page more information can be found for each tag.

When more than one tag is listed with the same order number, all tags of that number can be mixed in any order. However, a tag with a higher order number cannot precede a tag with a lower order number. For example, a tag with an order number of three cannot precede a tag with an order number of one or two.

Table 92. Tags Allowed Between the UL and EUL Tags

Tag Name	Order	Use	Page
FIG (Figure)	1	B	527
LINES (Unformatted lines)	1	B	546
XMP (Example)	1	B	639
NT (Note)	1	B	590
P (Paragraph)	1	B	593
PC (Paragraph continuation)	1	B	603
LP (List part)	1	B	579
OL (Ordered list)	1	B	591

Table 92. Tags Allowed Between the UL and EUL Tags (continued)

Tag Name	Order	Use	Page
SL (Simple list)	1	B	622
UL (Unordered list)	1	B	633
PARML (Parameter list)	1	B	602
DL (Definition list)	1	B	525

Optional Attribute

COMPACT
The list is formatted without a blank line between the items.

Required Tag

:LI.*item-text*
The text for the list item. This text is preceded by the item identifier; the text is indented four spaces from the current margin.

Example: Unordered Lists

This example uses two unordered lists, one imbedded within the other. The second unordered list uses the `COMPACT` attribute.

UIM Source

```
Some normal text...
:ul.
:li.First item (bullet)
:ul.
:li.First item (hyphen)
:li.Second item (hyphen)
:eul.
:li.Second item (bullet)
:eul.
```

Results

```
Some normal text...

• First item (bullet)

 - First item (hyphen)
 - Second item (hyphen)

• Second item (bullet)
```

VAR (Variable Definition)

```
►►──:VAR──NAME──=──variable-name──┬───────────────────────────┬──────────────────►
                                  └─CLASS──=──class-name─┘

►──┬──────────────────────────────────────────┬──.──────────────────────────────►◄
   └─ERRVAR──=──error-status-variable-name─┘
```

The variable definition (VAR) tag declares a dialog variable in the panel group. The VAR tag does not allow nested tags and must be placed after the last class definition (CLASS) tag and before the first variable record definition (VARRCD) tag.

A value for each dialog variable exists in the variable pool for an application opened using this panel group. The dialog variables communicate values to and from panels when the panels are presented to the user. Dialog variable values are set and retrieved using the UIM application programming interfaces (APIs).

The VAR tag defines all Z-variables, as well as other dialog variables used by an application. Only variables defined by a VAR tag in the panel group source are available in an open application.

Required Attribute

NAME=*variable-name*
The name of a dialog variable. This name must be unique within the panel group. For more information on the rules for naming, see "Name Syntax" on page 469.

Optional Attributes

CLASS=*class-name*
The name of the class to which this variable belongs. The class must have been previously defined in the panel group.

The class definition for all Z-variables is determined by the UIM, so this attribute cannot be specified on the VAR tag for a Z-variable. A class must be specified for other variables declared in the panel group.

ERRVAR=*error-status-variable-name*
The name of a dialog variable, automatically defined by the UIM with a `BASETYPE` of CHAR 1 on the CLASS tag. The error status variable must not be defined by another VAR tag. This name must also be unique within the panel group.

This dialog variable contains the error status of the variable defined by this tag. This attribute is used with the Put Dialog Variable (QUIPUTV) API and the Get Dialog Variable (QUIGETV) API.

During the QUIPUTV API, the current value of the error variable determines the error status. A value of 1 (X'F1') indicates that the dialog variable specified on the `NAME` attribute on this tag is in error. A value other than 1 (X'F1') indicates that the dialog variable is not in error. If the error status variable is updated during the same QUIPUTV API request as its associated dialog variable, the new value of the error status variable is used.

During the QUIGETV API for either the *NAME* dialog variable or the *ERRVAR* dialog variable, the value of the error status variable is set to 1 (X'F1') if the dialog variable specified on the `NAME` attribute of this tag is in error. Otherwise, the value of the error status variable is set to zero (X'F0').

Note: Simply updating the *ERRVAR* dialog variable does not set the error state of the dialog variable specified in the `VAR` attribute of this tag. The *VAR* dialog variable must be updated using the QUIPUTV API to change its error state.

The error dialog variable cannot be referred to any place in the panel group source where dialog variables can normally be specified.

Dialog Variables Defined by UIM

The UIM defines a set of dialog variables that can be automatically available in the variable pool for every open application. The variables must be defined using the variable definition (VAR) tag in order to be available in the variable pool. These variables can be referenced by user programs (see the QUIGETV API) and in panel definitions, and some can be modified by user programs (see the QUIPUTV API) to control UIM processing functions. Dialog variables are not allowed for input fields on any panel.

All dialog variables defined by the UIM have names that begin with the letter Z. If you define a dialog variable for your application with the same name as a UIM dialog variable, you receive a warning message. The message states that you do not have access to the UIM dialog variable with the same name. However, you are allowed to use the dialog variable that you defined. The following is a description of all UIM-defined dialog variables:

ZCANCEL Indicates whether the CANCEL function is requested and is set as follows:

0 The CANCEL function was not requested.

1 The CANCEL function is requested.

This Z-variable can be changed by the calling (application) program using the QUIPUTV API. Changing this dialog variable results in changing the job's cancel flag. For more information on the cancel flag, see "Folding Up a List Panel" on page 355.

ZDBCS Indicates whether the job is currently in DBCS mode, and is set as follows:

0 The job is not in DBCS mode.

1 The job is in DBCS mode.

ZDSPSIZ Indicates the maximum size of the display that the device is capable of displaying and is set as follows:

***DS3** The largest size panel the device can display is 24 x 80.

***DS4** The largest size panel the device can display is 27 x 132. This device can also display panels of size 24 x 80.

ZEXIT Indicates whether the EXIT function was requested and is set as follows:

0 The EXIT function is not requested.

1 The EXIT function is requested.

This Z-variable can be changed by the calling (application) program using the Put Dialog Variable API service. Changing this dialog variable results in changing the job's exit flag. For more information on the job's exit flag, see "Folding Up Multiple Panels When EXIT Is Requested" on page 354.

ZJOB Simple name of the current job (the first qualifier of the job name).

ZJOBNBR Job number of the current job (the last qualifier of the job name).

ZLMTCPB Indicates the limits to which the user controls the initial program, initial menu, current library, and Attention key handling program values and is set as follows:

***NO**

***PARTIAL**

***YES**

ZMENU Name of the menu object currently displayed by the UIM. This dialog variable can be used only in definitions of *MENU objects.

ZMNULIB The name of the library that contains the menu object currently displayed by the UIM. This dialog variable can be used only in definitions of *MENU objects.

ZSYSNAM Current system name retrieved from the SYSNAM network attribute.

ZUSER User profile name for the current job (which is also the middle qualifier of the job name).

Z36ENV Indicates whether the job is currently running in the System/36 environment and is set as follows:

0 The job is not running in the System/36 environment.

1 The job is running in the System/36 environment.

Table 93. Attributes of UIM-Defined Variables (Z-Variables)

Name	Base Type	Length	Program Modifiable	Description
ZCANCEL	CHAR	1	Yes	Indicates whether the CANCEL function is required
ZDBCS	CHAR	1	No	Indicates whether the job is in DBCS mode
ZDSPSIZ	CHAR	4	No	Indicates the maximum size of display
ZEXIT	CHAR	1	Yes	Indicates whether the EXIT function is requested
ZJOB	OBJNAME	10	No	Job name of the current job (first qualifier)
ZJOBNBR	CHAR	6	No	Job number of the current job
ZLMTCPB	CHAR	10	No	Limits the capabilities the user has
ZMANMNU	CHAR	1	No	Mandatory menu job flag
ZMENU	OBJNAME	10	No	Name of the menu object
ZMNULIB	OBJNAME	10	No	Name of the library containing the menu object
ZSYSNAM	CHAR	8	No	Current system name
ZUSER	OBJNAME	10	No	User profile name
Z36ENV	CHAR	1	No	Indicates if running in the System/36 environment

VARRCD (Variable Record Definition)

```
►►──:VARRCD──NAME──=──variable-record-name──VARS──=──'──variable-list──'──────────────────►

►──┬──────────────────────────────┬──┬──────────────────────────────┬──.───────────────────►◄
   └─NOPUT──=──'──variable-list──'─┘  └─NOGET──=──'──variable-list──'─┘
```

The variable record definition (VARRCD) tag does not allow nested tags, and must be placed after the last variable definition (VAR) tag and before the first list definition (LISTDEF) tag.

When data is passed between a calling program and the variable pool, a variable buffer is used. The variable buffer contains values for one or more dialog variables. A VARRCD tag defines which dialog variables are in the variable buffer as well as the sequence the variables appear in the variable buffer.

The VARRCD tag must be used to define a variable record definition for use with the Get Dialog Variable (QUIGETV) API and the Put Dialog Variable (QUIPUTV) API requests. Only variables defined by a VAR tag in the panel group source may appear in the variable record definition.

Required Attributes

NAME=*variable-record-name*
The name of a variable record definition. This name must be unique within the panel group. For more information on the rules for naming, see “Name Syntax” on page 469.

VARS='*variable-list***'**
A list of up to 256 dialog variables which make up the variable record. All dialog variables in this variable list must be previously defined using the VAR tag. The dialog variable names in the list are separated by one or more blanks.

The order in which the variable names appear in the variable list is the order the variable values must appear in the variable buffer of the calling program when the QUIPUTV and QUIGETV APIs are called.

The variables must appear contiguously in the variable buffer for the calling program. The UIM makes no boundary alignment adjustments when calculating the offsets to variables within the buffer.

If any read-only Z-variables are listed in the `VARS` attribute of this tag, they must also appear in the `NOPUT` attribute of this tag.

Optional Attributes

NOPUT='*variable-list***'**
A list of up to 256 variables which should not be processed when the QUIPUTV API is requested. All variables in this variable list must appear in the variable list specified on the `VARS` attribute of this tag. The variable names in the list are separated by blanks.

Using this attribute, you can specify that only a subset of the variables in the variable buffer should be copied to the variable pool during the QUIPUTV API.

If this attribute is not specified, all variables specified on the `VARS` attribute of this tag are copied from the variable buffer to the variable pool during the QUIPUTV API.

NOGET='*variable-list***'**
A list of up to 256 variables which should not be processed when the QUIGETV API is requested. All variables in this variable list must appear in the variable list specified on the `VARS` attribute of this tag. The variable names in the list are separated by blanks.

Using this attribute, you can specify that only a subset of the variables in the variable buffer should be copied from the variable pool during the QUIGETV API.

If this attribute is not specified, all variables specified on the `VARS` attribute of this tag are copied from the variable pool to the variable buffer during the QUIGETV API.

XH1 through XH4 (Extended Help Headings)

```
►►──┬─:XH1.─┬──extended-help-heading-text─────────────────────────────────►◄
    ├─:XH2.─┤
    ├─:XH3.─┤
    └─:XH4.─┘
```

The extended help heading (XH1-XH4) tag identifies the heading used over contextual help when it is displayed as part of an extended panel help, including help for index search topics. This tag must appear within a help module (HELP) tag and EHELP boundary and is not allowed on the information (INFO) tag. The text associated with this tag is used as the heading text.

This tag allows headings to appear with help information, based on whether the text is displayed for extended panel help or for contextual help. The text from XH*n* tags always appears in extended panel help. The text also appears in contextual help if any tag containing help information precedes the XH*n* tag, or the contextual help is displayed in a full screen and no help panel title is specified on the HELP tag.

In all cases of contextual help, with the exception of the two previously mentioned, the text from the XH*n* tag does not appear when the XH*n* tag is the first tag used for contextual help.

Required Text

extended-help-heading-text
The text associated with this tag is used as a heading when the help item is used in extended panel help. The text must be on the same or next line as the period ending the tag.

Formatting Rules

Headings have one blank line formatted before and after them. The heading tag generates a blank line before the heading, and the tag following the heading generates the space after the heading.

Specific formatting rules follow:

XH1 Centers, underscores and highlights text (HP3). This tag causes a page eject when it appears in a printed help module.

XH2 Left-justifies, underscores, and highlights text (HP3).

XH3 Left-justifies and highlights text (HP2).

XH4 Left-justifies and underscores text (HP1).

Example: Sample Headings

This example shows how extended help headings are highlighted and justified.

UIM Source

```
:XH1.Main Subject
:p.Here's a paragraph.
:XH2.Topic
:p.Another paragraph.
:XH3.Subtopic
:p.Still another paragraph.
:XH4.A Four Heading
:p.Still another paragraph.
```

Results

Main Subject

```
Here's a paragraph.
```

Topic

```
Another paragraph.
```

Subtopic

```
Still another paragraph.
```

A Four Heading

```
Still another paragraph.
```

XMP (Example)

```
▶▶—:XMP.—:EXMP.——————————————————————————————————————————————▶◀
```

The example (XMP) tag identifies an example of computer input or output. The XMP tag requires a matching end tag. These tags are only allowed in information areas and help areas.

Examples can occur anywhere in text, except in other examples. The body of an example is composed of the text between the XMP tag and the EXMP tags.

The normal formatting of text is suspended within an example; that is, the lines are not concatenated.

Note: Care should be taken when using examples within lists, because example text that does not fit on a line wraps to column one of the next line. The current left margin changes when nested lists are formatted. If there is help information containing examples imbedded at various locations, including within lists, it may not look the same each time and can cause undesirable results. An example is indented 4 spaces from the current margin. Help modules displayed in full-screen format or in an extended help window are indented by 4 bytes. Therefore, the formatting width initially is 10 bytes less than the width specified on the help module. When the XMP tag appears in an information area, the formatting width initially is 6 bytes less than the width specified on the panel tag.

Example: Formatting an Example

This example illustrates how to use an example.

UIM Source

```
Some normal text...
:xmp.
This is an example
right here
:exmp.
```

Results

```
Some normal text...

This is an example
right here
```

Appendix B. UIM Dialog Commands

Dialog commands are special functions, recognized only by the UIM, that equate actions entered by a user with screen management functions. Dialog commands differ from CL commands in that they cannot be entered on a command line nor are they valid outside the scope of the UIM.

Exactly what function the UIM performs for each dialog command depends largely on the type of screen being presented and other attributes defined by the function key, key list item (KEYI), menu item (MENUI), list action (LISTACT), or pull-down field choice (PDFLDC) language tags. For more information on the language tags, see Appendix A, “UIM Panel Group Definition Language,” on page 465.

Some dialog commands are defined in terms of the content of the current command line, the position of the cursor, and the type of screen. Dialog commands also affect messages specified by the application, by the condition evaluation, and by calling the general exit. For example, some dialog commands cause condition evaluation to be bypassed, while others cause the calling of the general exit to be bypassed.

Dialog commands can be assigned to the following:

- Menu items
- Function keys
- Options on action lists
- Hypertext links
- Pull-down field choices
- `ENTER` and `SELECT` attributes of the PANEL tag

Not all dialog commands are allowed in all of these situations. For example, the dialog commands PAGEUP and PAGEDOWN cannot be assigned to options of an action list because they apply to the entire screen, and not to an individual list entry.

For a summary of the valid uses of the dialog commands, see Table 94. For a summary of the effect of the different dialog commands, see Table 95 on page 642. Later in this appendix, there is a detailed description of each dialog command.

Table 94. Summary of the Valid Uses of Dialog Commands

Dialog Command Name	Valid Uses						
	Function Key	**Pull-down Menu Choices**	**Menu Item**	**List Action**	**ENTER on PANEL Tag**	**SELECT on PANEL Tag**	**Hypertext Link Action**
ACTIONS	X					X[3]	
CALL	X	X	X	X	X		
CANCEL	X		X				
CHGVIEW	X	X[2]					
CMD	X	X	X	X	X		
CMDLINE	X	X[2]					
DSPHELP		X[2]					X
ENTER	X						
EXIT	X	X[2]	X				
EXTHELP		X[2]					

Table 94. Summary of the Valid Uses of Dialog Commands (continued)

Dialog Command Name	Valid Uses						
	Function Key	Pull-down Menu Choices	Menu Item	List Action	ENTER on PANEL Tag	SELECT on PANEL Tag	Hypertext Link Action
HELP	X						
HELPHELP		X[2]					
HELPIDX		X[2]					
HOME	X						
KEYSHELP		X[2]					
MENU	X	X[2]	X				
MOREKEYS	X						
MOVETOP	X						
MSG					X	X	
PAGEDOWN	X						
PAGEUP	X						
PRINT	X						
PROMPT[1]	X						
PULLDOWN						X[3]	
RETRIEVE	X	X[2]					
RETURN	X	X	X		X	X	

Notes:

[1] PROMPT becomes CALL or RETURN when processing a cursor-sensitive prompt defined by the `PROMPT` attribute of a DATAI, DATAIX, or LISTCOL language tag.

[2] Valid only when ACTFOR=PANEL is specified on the PDFLDC tag.

[3] Valid only when MBAR attribute is specified on the PANEL tag.

Table 95. Summary of the Effects of Dialog Commands

Dialog Command Name	Effects			
	VARUPD on KEYI or PDFLDC Tag[1]	Stop Displaying User Messages[2]	Evaluate Conditions	Call General Exit
ACTIONS	No	No	No	No
CALL	Yes or No	Yes	Yes	Yes
CANCEL	Yes or No	Yes	N/A	Yes
CHGVIEW	Yes	Yes	Yes	Yes
CMD	Yes or No	Yes	Yes	Yes
CMDLINE	No	No	No	No
DSPHELP	No	No	No	No
ENTER	Yes	N/A[3]	N/A[3]	N/A[3]
EXIT	Yes or No	Yes	N/A	Yes
EXTHELP	No	No	No	No
HELP	No	No	No	No

Table 95. Summary of the Effects of Dialog Commands (continued)

Dialog Command Name	Effects			
	VARUPD on KEYI or PDFLDC Tag[1]	Stop Displaying User Messages[2]	Evaluate Conditions	Call General Exit
HELPHELP	No	No	No	No
HELPIDX	No	No	No	No
HOME	Yes	Yes	Yes	Yes
KEYSHELP	No	No	No	No
MENU	Yes	Yes	Yes	Yes
MOREKEYS	No	No	No	No
MOVETOP	Yes	No	No	No
MSG	N/A	Yes	Yes	Yes
PAGEDOWN	Yes[6]	No	No	Yes[6]
PAGEUP	Yes[6]	No	No	Yes[6]
PRINT	No	No	No	No
PROMPT[5]	Yes or No	Yes	Yes	Yes
PULLDOWN	No	No	No	No
RETRIEVE	No	No	No	No
RETURN	Yes or No	Yes	N/A	Yes
Default[4]	N/A	Yes	Yes	Yes

Notes:

[1] VARUPD attribute applies only when the action is assigned to a function key or pull-down choice. When ACTFOR=LIST is specified on the PDFLDC tag, VARUPD is YES.

[2] The UIM stops displaying messages generated by an application (user messages) when another function is about to start. HELP can become a function if the help is for a command and the user directly starts the prompt and processing of the command from the help screen.

[3] These ENTER values are not applicable (N/A) because ENTER always equates with another function by the time these actions are performed.

[4] The UIM displays the panel again for the default ENTER or SELECT action.

[5] PROMPT becomes CALL or RETURN when processing a cursor-sensitive prompt defined by the PROMPT attribute of a DATAI, DATAIX, or LISTCOL language tag.

[6] When PAGEUP and PAGEDOWN apply to the message area, an information area, or a menu area, these values are not applicable.

The VARUPD Attribute

One of the main qualifiers of a function is the VARUPD attribute of the KEYI (key list item) and PDFLDC (pull-down field choice) language tag. This attribute defines whether or not dialog variables and list entries should be updated with values entered by the user. It also defines whether the function associated with the key should be processed if errors are detected while validity checking the values of input fields on the display. Most dialog commands assume a particular value for VARUPD that cannot be overridden by the panel definition. The following is a summary of the effect of VARUPD:

Type of Action	Effect of VARUPD Attribute	
	VARUPD=YES	VARUPD=NO
Returning control to the application that called the QUIDSPP API.	All validity checking is performed, and dialog variables and list entries are updated when correct. When one or more variables fail the validity checking, the panel is displayed again with error messages.	No validity checking is performed and no updates are made to the dialog variables and list entries. Changes made to the input field by the user since the last operation are lost. The values entered by the user can sometimes be recovered by calling QUIDSPP and specifying yes on the redisplay parameter. For more information, see the **APIs** topic .
An action performed under the control of the UIM *without* returning to the application	All validity checking is performed, and dialog variables and list entries are updated when correct. When one or more variables fail the validity checking, the panel is displayed again with error messages.	No validity checking is performed and no updates are made to the dialog variables and list entries. Contents of the input fields are saved by the UIM and are used when displaying the panel again, providing updates have not been made that make the saved values obsolete. For example, changing a dialog variable that was displayed on the saved panel causes the saved value to be discarded and the new value shown when the panel is displayed again.

ACTIONS (Menu Bar Cursor Action)

```
►►──ACTIONS──────────────────────────────────────────────────────────────────►◄
```

Alternates the cursor position between the panel and the menu bar. The Common User Access (CUA) guidelines recommend this function be assigned to F10.

When the ACTIONS function key is pressed, the UIM saves the information where the cursor was located and moves the cursor to the first choice on the menu bar. If the ACTIONS function key is pressed again, the cursor returns to its previous location before it was moved to the menu bar. If the user moves the cursor to the menu bar area using the cursor movement keys (arrow keys) and then presses the ACTIONS function key, the UIM determines the best position for the cursor.

Messages

No messages are issued by the UIM during processing.

CALL (Call Program)

```
►►──CALL──dialog-variable-name───────────────────────────────────────────────►◄
```

Calls an application program and allows applications to link programs and screens together.

The call function calls by address, by name, or by an extended program model (EPM) call.

For best performance, use call by address. If call by address is not feasible, use the call by name or an EPM call with a library qualified program name (do not use *LIBL).

For a description of the interface between the UIM and the program being called, see the section called ″User Interface Management EXIT Program″ in the The **APIs** topic . The CALL dialog command can be used with the following:

- Menu item action
- Function key action
- Action list option
- Action list exit
- General panel exit
- Default enter action
- Application format exit
- Contextual prompt exit
- Pull-down choice action

The CALL dialog command cannot be used on menu objects created using the Create Menu (CRTMNU) command, but can be used within a panel group object created using the Create Panel Group (CRTPNLGRP) command. If calling a program is the desired result of a menu option in an external menu, use the CMD dialog command to submit the CALL CL command. The user must have the proper authority to the CALL CL command and the program being called.

Required Parameter

dialog-variable-name

Specifies a dialog variable containing the call information as described below for each of the three types of call.

Call by address

This type of call is used within i5/OS system programs to allow calls of other system modules. Define the specified dialog variable with `BASETYPE='PTR'`. When the call function is requested, the dialog variable must contain a system pointer to the program. Ensure the proper authority is contained in the pointer.

Call by name

Allows calling an application program using a qualified name of a program object. Define the specified dialog variable with `BASETYPE='CHAR 20'`. When the call function is requested, the first 10 characters of the dialog variable must contain the name of the program object. The second 10 characters must contain the name of the library where the program resides. The program and library names should be left justified and put in uppercase unless a quoted object name is used.

Call by extended program model (EPM)

This type of call allows calling an entry point in an EPM language program. The UIM uses the QPXXCALL interface to make the call. Define the specified dialog variable with `BASETYPE='CHAR 130'`. When the call function is requested, the dialog variable must contain the following:

Characters	**Contents**
1-20	The name of the EPM program object and the name of the library that the object resides in
21-120	The name of the external entry point to be called
121-130	The name of the environment to which the entry point belongs

Because UIM is simply passing this information to the QPXXCALL language interface program, ensure that the contents of the dialog variable adhere to the requirements defined by QPXXCALL. When UIM calls the QPXXCALL program, the EPM environment is implicitly started and remains active until explicitly deleted.

Note: There is a high performance overhead for calling an EPM program. If possible, call by address or call by name instead. The support for calling an EPM program is provided for cases where, due to application design, an EPM program is used as the target of the CALL dialog command.

VARUPD Value

The CALL dialog command does not have a predefined VARUPD value. When CALL is assigned to a function key, the VARUPD value of the function key dictates whether or not to perform validity checking and then updates the dialog variables.

Messages

The following errors may be encountered during processing:

- Program not found.
- Not authorized to specified program.
- Number of parameters for specified program is not valid.
- Program saved with STG(*FREE).
- Any exception signalled by the QPXXCALL language interface program.

The UIM does not stop processing if these errors occurred on a call for the general panel exit, on a call for the list action exit, and on application formatted area calls.

For all other types of calls, the UIM ends the action it was performing. In most cases, this involves displaying the panel again with an error message indicating that the program call was not successful.

CANCEL

```
           ┌─NOSET─┐
►►──CANCEL─┴───────┴─────────────────────────────────────────────►◄
           └─SET───┘
```

Backs up one screen and is used with a menu item action or a function key action.

If the current screen was called by the QUIDSPP API, the function parameter contains an indication that CANCEL was requested.

If the current screen is a menu called from the previous screen via the menu fast-path function (for example, entering GO on the command line), the UIM displays the previous screen.

Optional Parameter

NOSET | SET
Indicates whether the cancel flag is set. NOSET means that the cancel flag is not set; this is the default value. SET means that the cancel flag is set. For more information on the cancel flag, see "Folding Up a List Panel" on page 355.

VARUPD Value

CANCEL does not have a predefined VARUPD value. When CANCEL is assigned to a function key, the VARUPD value of the function key determines whether or not to perform validity checking and then updates the dialog variables.

Messages

No messages are issued during processing.

CHGVIEW (Change View)

```
►►──CHGVIEW──────────────────────────────────────────────────────────────────────►◄
```

Changes the displayed view of a list and can only be assigned to a function key or a pull-down field choice action.

If there are several columns of information that a user needs to see, and all the columns do not fit on the display, you can change the view so the user sees one set of columns and then another set of columns. This continues until all the columns of information are viewed.

You can specify several different ways to present a list area on the LISTVIEW language tag and which columns belong to each view. List views are numbered sequentially starting with 0. For example, if a list areas has five presentation views, the views are numbered from 0 to 4. The first LISTVIEW tag in the panel source for the list area defines view 0, the second LISTVIEW tag in the panel source for the list area defines view 1, and so on for all LISTVIEW tags in the list area.

With the CHGVIEW dialog command, you can switch views. The function operates in a circular fashion. For example, if the current view is 3, CHGVIEW presents view 4 (if there is a view 4), or back to view 0 if there is not a view 4. Applications can specifically select a view by changing the dialog variable coded on the `VIEW` attribute of the LIST tag. The UIM also modifies this dialog variable whenever the CHGVIEW dialog command is performed.

Which list the request applies to depends on the cursor position. The rules are very similar to those for the PAGEDOWN and PAGEUP dialog commands.

- If the cursor is in a list area defined as having multiple views, the request applies to that area.
- If the cursor is not in a list area defined as having multiple views, the request applies to the first list area on the screen that does have multiple views.

Parameters

None

VARUPD Value

The CHGVIEW dialog command always operates with VARUPD=YES and the following occurs:

- Validity checking is always performed.
- Valid values are processed and the dialog variables are updated.
- Any failures from validity checking prevent changing the view and the screen is displayed again with the appropriate error messages.

Messages

No messages are issued during processing unless there are errors from a validity check.

CMD (System Command)

```
►►──CMD──command-text────────────────────────────────────────────────────────────►◄
```

Submits a CL or OCL command to the system command.

Parameter

command-text

All text following the CMD dialog command is treated as a command and is submitted to the system for processing. Whether the command is native or System/36 environment is determined by the system, not by the UIM.

The command string may contain the names of dialog variables to be substituted into the command string before the UIM submits the command to the system. Each dialog variable name must be preceded by an ampersand (&); and followed by a period (.). Although the ending period is not required, its use is recommended to avoid ambiguity.

Before submitting the command, the UIM performs substitution for dialog variables. Each dialog variable specified as part of the command string is replaced by its displayable value. This value is always calculated from the current value of the dialog variable.

VARUPD Value

The CMD dialog command does not have a predefined `VARUPD` value. When CMD is assigned to a function key or pull-down field choice, the `VARUPD` attribute on the KEYI or PDFLDC language tag dictates whether or not to perform validity checking and then updates the dialog variables. Because substituting commands depends on the values of dialog variables, care should be taken when using VARUPD=NO.

Messages

No messages are issued during processing unless there are errors from a validity check.

Hint

Because these commands are not logged to the job log, it is sometimes difficult to discover and correct command syntax errors. One way to see the command after all dialog variables have been substituted is to make the command the value of the message on a SNDMSG command as follows:

```
SNDMSG MSG (command here) TOUSR (*REQUESTER)
```

CMDLINE (Command Line)

```
►►──CMDLINE──────────────────────────────────────────────────────────►◄
```

Displays a pop-up window containing a command line on the bottom of the display.

Parameters

None

Messages

No messages are issued during processing.

DSPHELP (Display Help)

```
►►──DSPHELP──help-module-name──┬─────────────────────────────────┬───►◄
                               └─panel-group──┬─────────────────┬┘
                                              └─product-library─┘
```

Displays a specified UIM help module.

Required Parameter

help-module-name
Specifies the name of the help module to be displayed.

Optional Parameter

qualified-panel-group-name
Specifies the name of the panel group containing the help module. If it is not specified, then 1) the help module must be from the same panel group that contains the DSPHELP dialog command, or 2) the help module must be imported from another panel group by using an IMPORT tag.

product-library
Specifies the name of a library which becomes the product library when the help module is displayed. This parameter is only used when the DSPHELP dialog command is used for a hypertext link action.

Messages

No messages are issued during processing.

ENTER

```
►►──ENTER─────────────────────────────────────────────────────────────►◄
```

Allows processing of an action and is only allowed on the Enter key.

For more information, see “Considerations for Using the ENTER, HELP, and PROMPT Dialog Commands.”

Parameters

None

VARUPD Value

The ENTER dialog command always operates with VARUPD=YES and the following occurs:

- Validity checking is always performed.
- Valid values are processed and the dialog variables are updated.
- Any failures from validity checking stop the operation. The display is shown again with the appropriate error messages.

Messages

Any messages resulting from submitting or canceling the commands are displayed on the message line. Additional messages can be sent if validity-checking errors are encountered.

Considerations for Using the ENTER, HELP, and PROMPT Dialog Commands

The actions performed for ENTER, HELP, and PROMPT depend heavily on the type of display and contents of the command line. These dialog commands can be assigned only to function keys, and they have different characteristics than other dialog commands. These dialog commands are more like action qualifiers; things used to determine or modify a more concrete action like a system command string.

For example, an action list panel describes several actions that are performed, based on the contents of various fields. The ENTER and PROMPT dialog commands, and to a lesser extent HELP dialog commands, are initially requested, but serve only to describe how the rest of the panel is interpreted. They are not sufficient by themselves to describe what should be done.

Some important aspects to remember when reading descriptions of these dialog commands are:

- A menu with a command line can contain a selection from a menu item or a command. If the first nonblank character string is unquoted and contains only numeric values, it is interpreted as a menu item. Otherwise, it is assumed to be a command.
- An action list with a command line can contain parameter strings or commands. If any entry on an action list is selected, any string in the command line is treated as a parameter list. Otherwise, it is treated as a command.

EXIT (Exit Display)

```
          ┌─NOSET─┐
►►──EXIT──┤       ├──────────────────────────────────────────────────────►◄
          └─SET───┘
```

Allows the user to back out of groups of displays.

If the current display was displayed using QUIDSPP API, the UIM sets the function requested parameter indicating that the EXIT dialog command was requested. It is up to the application to determine what must be done next.

If the current display is a menu called from a series of displays, the UIM presents the display where the MENU command was initially requested. The EXIT dialog command backs completely out of a series of menus.

Optional Parameter

NOSET | SET
Indicates whether the job’s exit flag is set. NOSET means that the exit flag is not set; this is the default. SET means that the exit flag is set. For more information on the exit flag, see “Folding Up Multiple Panels When EXIT Is Requested” on page 354.

VARUPD Value

The EXIT dialog command does not have a predefined VARUPD value although the user interface style implies VARUPD=NO should normally be used. When EXIT is assigned to a function key or pull-down choice, the VARUPD value of the function key dictates whether or not to perform validity checking and then updates the dialog variables.

Messages

No messages are issued during processing.

EXTHELP (Extended Help)

```
►►──EXTHELP──────────────────────────────────────────────────────────────►◄
```

Displays the extended help for the panel.

Parameters

None

HELP

```
►►──HELP──────────────────────────────────────────────────────────────►◄
```

Displays help information. The Common User Access (CUA) guidelines recommend assigning this to the Help and F1 keys.

The specific action taken (and help text displayed) depends on the screen type, cursor position, and so on. For the UIM rules, see "Defining Contextual Help" on page 348, and for more information on the specific attributes, see "Considerations for Using the ENTER, HELP, and PROMPT Dialog Commands" on page 649.

Parameters

None

VARUPD Value

It is always VARUPD=NO. No validity checking is performed on the current contents of the current panel and no updates to dialog variables are made.

Messages

The message `Help information is not available` can be issued during processing and can occur if the user chooses to delete or not install help text supplied by the system.

Additional messages are also possible when help is requested for a specific command on a command line. Possible messages might be that the command does not exist, or the user might not have authority to it. If these messages are issued, they appear as the first message in the message area. Messages describing other conditions on the screen can still be viewed by scrolling forward.

HELPHELP

```
►►──HELPHELP──────────────────────────────────────────────────────────►◄
```

Displays information about how to use the help facilities for contextual help, extended help, help on function keys, and the help index from help panels. The HELPHELP dialog command can be coded only on pull-down choices for menu bars.

Parameters

None

Messages

No messages are issued during processing.

HELPIDX

```
►►──HELPIDX───────────────────────────────────────────────────────────►◄
```

Begins the index search function for the SCHIDX attribute specified on the panel group tag. Users may select a topic in the index to be displayed or printed. The HELPIDX dialog command can be coded only on pull-down choices for menu bars.

Parameters

None

Messages

No messages are issued during processing.

HOME (Display Home Menu)

```
►►──HOME──────────────────────────────────────────────────────────────►◄
```

Displays the job's home menu. This menu is added to the stack of menus. It does not imply returning to the beginning of the stack of menus. The HOME dialog command can be used only on function keys.

Note: HOME is allowed only on panels containing a menu area.

Parameters

None

VARUPD Value

The HOME dialog command always operates with VARUPD=YES and the following occurs:

- Validity checking is always performed.
- Valid values are processed and the dialog variables are updated.
- Any failures from validity checking prevent the operation and the screen is displayed again with the appropriate error messages.

Messages

No messages are issued during processing.

KEYSHELP

```
►►──KEYSHELP──────────────────────────────────────────────────────────►◄
```

Displays the help for the function keys on the displayed panel. The Common User Access (CUA) guidelines recommend assigning this to the F1 key.

Messages

No messages are issued during processing.

MENU

```
                               ┌─RTNPNT───┐
►►──MENU──qualified-menu-name──┤          ├─────────────────────────────────────────►◄
                               └─NORTNPNT─┘
```

Displays a subsequent menu as a result of selecting a menu item or pressing a function key.

Required Parameter

qualified-menu-name
Specifies the qualified name of the menu to be displayed.

Optional Parameter

RTNPNT | NORTNPNT
This parameter indicates whether the current menu is considered a return point when the dialog command is requested. This parameter functions the same as the return point (RTNPNT) parameter of the GO CL command. RTNPNT indicates that the current menu is returned when the Exit key is pressed; this is the default. No return point (NORTNPNT) indicates to not return to the current menu when the Exit key is pressed.

This parameter has no effect when it is coded in a panel group object using the Create Panel Group (CRTPNLGRP) command. It is meaningful only when it is used in a menu object created using the Create Menu (CRTMNU) command.

Messages

No messages are issued during processing.

MOREKEYS (Display More Function Keys)

```
►►──MOREKEYS──────────────────────────────────────────────────────────────────────►◄
```

Displays an additional list of active function keys when they cannot all fit on the panel. The action of the MOREKEYS dialog command can be assigned only to Function Key F24. When it is, it cannot be conditioned using the COND attribute of the key item (KEYI) language tag.

If an application codes ACTION=MOREKEYS on the KEYI tag, the UIM manages the situation if all of the function key descriptions do not fit on the panel at once.

When a panel is displayed with a KEYL (Key List) language tag using the MOREKEYS dialog command, the UIM gathers a list of all currently active function keys. This is based on the priority assigned to each key item on the PRIORITY attribute of the KEYI language tag. The UIM attempts to fit as many of the function keys as possible in the function key area of the panel.

If not all of the function key descriptions fit in the area, the UIM enables the function key assigned to the MOREKEYS dialog command and places the descriptions of the function keys on the panel. If all of the descriptions of the function keys fit, the function key assigned to the MOREKEYS dialog command is not enabled.

If a user presses the function key assigned to the MOREKEYS dialog command, the UIM starts where it left off in the key list and gathers as many active descriptions of function keys that fit in the function key area in order of priority. The function key assigned to the MOREKEYS dialog command is enabled.

This process continues until the end of the key list is reached. If the MOREKEYS dialog command is selected again, the UIM starts back at the beginning of the key list and begins processing the MOREKEYS dialog command again.

Parameters

None

Messages

No messages are issued during processing.

MOVETOP (Move to Top)

```
►►──MOVETOP──────────────────────────────────────────────────────────────────►◄
```

Moves a cursor-selected line to the top of the scrollable information area.

When a function key assigned to this dialog command is pressed, the UIM moves the line containing the cursor to the top of the scrollable area. If the cursor is on a blank line inserted by the UIM (paragraph separators, and so on), the next line is moved to the top of the scrollable area.

The MOVETOP dialog command applies only to information areas, and is intended primarily to allow users of online text to position the information in the most readable manner. For example, an entire chart positioned on one display is easier to read than a chart that appears on two displays.

Parameters

None

VARUPD Value

The MOVETOP dialog command operates with VARUPD=YES (except when scrolling the message line) and the following occurs:

- Validity checking is always performed.
- Valid values are processed and the dialog variables are updated.
- Any failures from validity checking prevent the operation and the panel is displayed again with the appropriate error messages.

Messages

The UIM can issue `Cursor not positioned on valid line.`

MSG (Display Message)

```
►►──MSG──message-id──┬─────────────────────────────┬─────────────────────────►◄
                     └─qualified-message-file-name─┘
```

Displays a message on the message line. It can only be assigned as the default enter action or the default selection action to perform when the Enter key is pressed, and the UIM cannot find any other specific function to perform.

One intended use of this dialog command is to inform the user that they must select an option or press a function key.

Required Parameter

message-id
Specifies the message identifier of the message to be displayed.

Optional Parameter

qualified-message-file-name
Specifies the qualified name of the message file containing the message. It defaults to the message file specified on the `DFTMSGF` attribute of the PNLGRP tag.

VARUPD Value

Because the MSG dialog command can only be specified on the `ENTER` attribute of the PANEL tag, `VARUPD` has no meaning. The `VARUPD` value is defined by the function key assigned to the ENTER action.

PAGEDOWN

```
►►──PAGEDOWN──────────────────────────────────────────────────────────►◄
```

Scrolls forward by one screen or panel and can be assigned to only a function key.

When processing incomplete lists, selecting scroll areas, and specifying VARUPD=YES, the process is the same for the PAGEDOWN dialog command as it is for the PAGEUP dialog command.

Parameters

None

Messages

UIM can issue `Already at bottom of area`.

Additional messages can be sent if validity checking encounters errors.

PAGEUP

```
►►──PAGEUP────────────────────────────────────────────────────────────►◄
```

Scrolls backward by one screen or panel and can be assigned to only a function key.

If a full scrollable area of data is available to the UIM, or the scrollable area is complete, scrolling is performed without intervention of the application. If the scrollable area is an action list with selected actions, none are performed at this time.

A special case exists where an incomplete list is displayed and the PAGEUP scroll results in an incomplete panel. In this case, the UIM calls the program specified to add additional list entries or mark the list complete.

For screens and panels with multiple scrollable areas, the position of the cursor determines what is scrolled. The rules are as follows:

- If the cursor is in a scrollable area, that area is selected. The message line is considered a scrollable area.

The boundaries of a scrollable area are defined by the top and bottom line of a menu, information, list, or data area, although not all information in that area (such as instruction lines) is actually scrolled.

- If the cursor is outside a scrollable area, the first scrollable area on the panel (top to bottom) is selected for scrolling.

Parameters

None

VARUPD Value

PAGEUP operates with VARUPD=YES except when scrolling the message line, and the following occurs:

- Validity checking is always performed.
- Valid values are processed and the dialog variables are updated.
- Any failures from validity checking prevent scrolling and the panel is displayed again with the appropriate error messages.

When scrolling the message line, the PAGEUP dialog command operates as if VARUPD=NO is in effect. Fields need not be correct and dialog variables are not updated.

Messages

UIM can issue `Already at top of area`.

Additional messages can be sent if validity checking encounters errors.

PRINT (Print Display)

```
►►──PRINT──────────────────────────────────────────────────────────────────►◄
```

Prints the current display and can be assigned only to the 5250 PRINT key.

Parameters

None

VARUPD Value

Does not apply.

Messages

No messages are issued during processing unless errors are found during validity checking.

PROMPT

```
►►──PROMPT─────────────────────────────────────────────────────────────────►◄
```

Prompts for commands, action list options, and entry fields and can be assigned only to function keys. Its specific function depends on the following:

- Type of screen being managed
- Location of the cursor
- Contents of the command or parameter line

Parameters

None

VARUPD Value

For more information, see "Prompting an Entry Field" and "Prompting an Action List Option or Command."

Messages

Any messages resulting from submitting or canceling the commands are displayed on the message line. In addition, the UIM can issue the following messages:

- `The prompt attempt for the menu item was not valid.`
- `Cursor is in a location where the prompt function is not allowed.`

Additional messages can be sent if errors from validity checking are encountered.

Prompting an Entry Field

The prompt function becomes a CALL or RETURN function based on the prompt definition specified on the `PROMPT` attribute of the associated panel definition tag. Prompting an entry field should be assigned to the F4 key.

When prompting an entry field, `VARUPD` processing is done if it is specified on the definition of the function key.

If VARUPD=YES is specified for the function key, the following occurs:

- Validity checking is always performed
- Valid values are processed and the dialog variables are updated
- Any failures from validity checking prevent prompting and the panel is displayed again with the appropriate error messages.

If VARUPD=NO is specified for the function key, the application program can get access to the keyed data that is not verified by using the `DSPVALUE` attribute of the tag where the prompt was defined.

For more information about entry field prompting, see the description of the `PROMPT` attribute for the following tags found in Appendix A, "UIM Panel Group Definition Language," on page 465.

- Data item (DATAI)
- Data item Extender (DATAIX)
- List column (LISTCOL)

Prompting an Action List Option or Command

When prompting an option on an action list or command line, the PROMPT dialog command always operates with VARUPD=YES and the following occurs:

- Validity checking is always performed.
- Valid values are processed and the dialog variables are updated.
- Any failures from validity checking prevent prompting and the panel is displayed again with the appropriate error messages.

The prompt function results in either submitting a command string to the system or calling an application program. This occurs according to the following rules:

- If the command line contains a menu item, an error is sent. The UIM does not support prompting for menu items.

- If the cursor is on the command line or in the option column and if the panel has an action list panel with options specified, the UIM selects the PROMPT action associated with each selected option. The PROMPT action can either be to submit a command to the system or to call an application program.

  If an option is selected that does not have a defined prompt action, action list processing stops, and the panel is displayed again with an error message. The error message indicates that the user cannot prompt for the selected option. In this case, the selected option number is displayed with error emphasis.

  The contents of the command line are parameters for the list action. These parameters can be accessed by defining a dialog variable for the UIM to update. The update is with the contents of the command line when the action list is processed. For more information see the LIST tag in Appendix A, "UIM Panel Group Definition Language," on page 465

  Each individual option receives a separate prompt. The UIM does not apply information added during prompting from one command to another.

  When the prompt action is to call an application program, the option is processed without prompting by the UIM. When the UIM calls the application program, an indication is given that the user requested the prompt function. It is the responsibility of the application program to provide its own version of prompting.
- If the command line contains a command, the command is submitted for prompting and processing. The command can then be run or canceled through normal prompter functions. Depending on the environment the job is running in, the system can pass the command to the System/36 environment or REXX.

  If the command line was blank (a subclass of assuming it contains a command), the Major Command Groups menu is displayed.

For additional information about the PROMPT dialog command, see "Considerations for Using the ENTER, HELP, and PROMPT Dialog Commands" on page 649.

PULLDOWN (Display Pull-Down Menu)

```
►►──PULLDOWN──────────────────────────────────────────────────────────────►◄
```

Displays the first pull-down menu of the menu bar.

Parameters

None

RETRIEVE (Retrieve Command String)

```
►►──RETRIEVE──────────────────────────────────────────────────────────────►◄
```

Displays the previously entered command.

The UIM keeps a series of commands entered on command lines. When the RETRIEVE dialog command is issued, the previous command is displayed on the command line. If the RETRIEVE dialog command is issued again before any other function is performed, the next most recent command is displayed.

Menu item selections and parameter strings cannot be retrieved. Also, commands submitted via the CMD dialog command cannot be retrieved.

Parameters

None

VARUPD Value

The RETRIEVE dialog command always operates with VARUPD=NO. No validity checking is performed on the contents of the current panel and no updates are made to the dialog variables.

Messages

No messages are issued during processing.

RETURN (Return Control to Application)

```
►►──RETURN──n──────────────────────────────────────────────────────────────►◄
```

Provides the application with a method of handling function keys, menu items, and pull-down choices by having the UIM return control to the application with the appropriate indication.

Note: To provide the Refresh/Redisplay function for an application, use the RETURN dialog command.

The effect of the RETURN dialog command can be obtained for action lists by using the `ACTOR` attribute of the LIST tag. This attribute determines whether or not the UIM should handle the action list.

Required Parameter

n This parameter specifies a numeric value from 1 through 32767. This value is returned to the application, on the function requested parameter of QUIDSPP API, when the RETURN function is encountered.

VARUPD Value

The RETURN dialog command does not have a predefined `VARUPD` value. When RETURN is assigned to a function key, the VARUPD value of the function key determines whether or not to perform validity checking and then updates the dialog variables.

Messages

No messages are issued during processing unless errors are found during validity checking.

Appendix C. Feedback Area Layouts for Display Files

This appendix contains Product-Sensitive Programming Interface and Associated Guidance Information.

Tables in this section describe the open and I/O feedback areas associated with any opened display file. The following information is presented for each item in these feedback areas:

- Offset, which is the number of bytes from the start of the feedback area to the location of each item.
- Data Type.
- Length, which is given in number of bytes.
- Contents, which is the description of the item and the valid values for it.

For information about feedback area layouts for files other than display files, see the **Files and File Systems** collection in the iSeries Information Center.

The support provided by the high-level language you are using determines how to access this information and how the data types are represented. See your high-level language manual for more information.

Open Feedback Area

The open feedback area is the part of the open data path (ODP) that contains general information about the file after it has been opened. It also contains file-specific information for display files, plus information about each device defined for the file. This information is set during open processing and may be updated as other operations are performed.

Table 96. Open Feedback Area

Offset	Data Type	Length	Contents
0	Character	2	Open data path (ODP) type: **DS** Display, tape, ICF, save, printer file not being spooled, or diskette file not being spooled. **DB** Database member. **SP** Printer or diskette file being spooled or inline data file.
2	Character	10	Name of the file being opened. If the ODP type is DS (for display files), this is the name of the display file.
12	Character	10	Name of the library containing the file. For an inline data file, the value is *N.
22	Character	10	Not applicable to displays.
32	Character	10	Not applicable to displays.
42	Binary	2	Not applicable to displays.
44	Binary	2	Maximum record length.
46	Character	2	Not applicable to displays.
48	Character	10	Not applicable to displays.
58	Binary	4	Reserved.
62	Binary	4	Reserved.

Table 96. Open Feedback Area (continued)

Offset	Data Type	Length	Contents
66	Binary	2	File type: **1** Display **2** Printer **4** Diskette **5** Tape **9** Save **10** DDM **11** ICF **20** Inline data **21** Database
68	Character	3	Reserved.
71	Binary	2	Number of lines on a display screen.
73	Binary	2	Number of positions on a display screen.
75	Binary	4	Not applicable to displays.
79	Character	2	Not applicable to displays.
81	Character	1	Not applicable to displays.
82	Character	1	Not applicable to displays.
83	Character	10	Reserved.
93	Character	10	Reserved.
103	Binary	2	Not applicable to displays.
105	Binary	2	Maximum number of records that can be read or written in a block when using blocked record I/O.
107	Binary	2	Not applicable to displays.
109	Binary	2	Blocked record I/O record increase. Number of bytes that must be added to the start of each record in a block to address the next record in the block.
111	Binary	4	Reserved.
115	Character	1	Miscellaneous flags. **Bit 1:** Reserved. **Bit 2:** File with sharing **0** File was not opened with sharing. **1** File was opened with sharing (SHARE(*YES)). **Bits 3-5:** Not applicable to displays. **Bit 6:** Field-level descriptions **0** File does not contain field-level descriptions. **1** File contains field-level descriptions. **Bit 7:** DBCS-capable file **0** File is not DBCS-capable. **1** File is DBCS-capable. **Bit 8:** Not applicable to displays.

Table 96. Open Feedback Area (continued)

Offset	Data Type	Length	Contents
116	Character	10	For display files, this is the name of the display device description that is the requesting program device. This field is supplied only when a device name of *REQUESTER is being attached to the display file by an open or acquire operation. Otherwise, this field contains *N.
126	Binary	2	File open count. If the file has not been opened with sharing, this field contains a 1. If the file has been opened with sharing, this field contains the number of programs currently attached to this file.
128	Binary	2	Reserved.
130	Binary	2	Not applicable to displays.
132	Character	1	Miscellaneous flags. **Bits 1-4:** Not applicable to displays. **Bit 5:** Separate indicator area **0** Indicators are in the I/O buffer of the program. **1** Indicators are not in the I/O buffer of the program. The DDS keyword, INDARA, was used when the file was created. **Bit 6:** User buffers **0** System creates I/O buffers for the program. **1** User program supplies I/O buffers. **Bit 7:** Reserved. **Bit 8:** Not applicable to displays.
133	Character	2	Open identifier. The value is unique for a full open operation (SHARE (*NO)) of a file or the first open operation of a file opened with SHARE(*YES). It allows you to match this file to an entry on the associated data queue.
135	Binary	2	The field value is the maximum record format length, including both data and file-specific information such as: option indicators, response indicators, source sequence numbers, and program-to-system data. If the value is zero, then use the field at offset 44.
137	Binary	2	Not applicable to displays.
139	Character	1	Miscellaneous flags. **Bits 1-3:** Not applicable to displays. **Bit 4:** CCSID character substitution **0** No substitution characters will be used during CCSID data conversion. **1** Substitution characters may be used during CCSID data conversion. **Bits 5-8:** Reserved.
140	Character	6	Reserved.
146	Binary	2	Number of devices defined for this ODP. For displays, this is determined by the number of devices defined on the DEV parameter of the Create Display File (CRTDSPF) command.
148	Character		Device name definition list. See “Device Definition List” on page 664 for a description of this array.

Device Definition List

The device definition list part of the open feedback area is an array structure. Each entry in the array contains information about each device attached to the file. The number of entries in this array is determined by the number at offset 146 of the open feedback area. The device definition list begins at offset 148 of the open feedback area. The offsets shown for it are from the start of the device definition list rather than the start of the open feedback area.

Table 97. Device Definition List

Offset	Data Type	Length	Contents
0	Character	10	Program device name. For display files, the value is the name of the device description.
10	Character	50	Reserved.
60	Character	10	Device description name. For display files, the value is the name of the device description.

Table 97. Device Definition List (continued)

Offset	Data Type	Length	Contents
70	Character	1	Device class. For displays, the device class is hex 01.
			Device type. **hex 07** 5251 Display Station **hex 0B** 5291 Display Station **hex 0D** 5292 Display Station **hex 12** 5555-B01 Display Station **hex 13** 3270 Display Station **hex 15** Graphic-capable device **hex 16** Financial terminal **hex 17** 3180 Display Station **hex 19** 3277 DHCF Device **hex 26** 3179-2 Display Station **hex 27** 3196-A Display Station **hex 28** 3196-B Display Station **hex 33** 3197-C1 Display Station **hex 34** 3197-C2 Display Station **hex 35** 3197-D1 Display Station **hex 36** 3197-D2 Display Station **hex 37** 3197-W1 Display Station **hex 38** 3197-W2 Display Station **hex 39** 5555-E01 Display Station **hex 3E** 3476-EA Display Station **hex 3F** 3477-FG Display Station **hex 40** 3278 DHCF device **hex 41** 3279 DHCF device **hex 42** ICF finance device **hex 43** Retail communications device **hex 44** 3477-FA Display Station **hex 45** 3477-FC Display Station **hex 46** 3477-FD Display Station **hex 47** 3477-FW Display Station **hex 48** 3477-FE Display Station **hex 4D** Network Virtual Terminal display station **hex 51** 5555-C01 Display Station **hex 52** 5555-F01 Display Station **hex 56** 3476-EC Display Station

Table 97. Device Definition List (continued)

Offset	Data Type	Length	Contents
71	Character	1	**hex 58** 5555-G01 Display Station **hex 59** 5555-G02 Display Station **hex 5D** 3486-BA Display Station **hex 5F** 3487-HA Display Station **hex 60** 3487-HG Display Station **hex 61** 3487-HW Display Station **hex 62** 3487-HC Display Station
72	Binary	2	Number of lines on the display screen.
74	Binary	2	Number of positions in each line of the display screen.
76	Character	2	Bit flags. **Bit 1:** Blinking capability. **0** Display is not capable of blinking. **1** Display is capable of blinking. **Bit 2:** Device location. **0** Local device. **1** Remote device. **Bit 3:** Acquire status. This bit is set even if the device is implicitly acquired at open time. **0** Device is not acquired. **1** Device is acquired. **Bit 4:** Invite status. **0** Device is not invited. **1** Device is invited. **Bit 5:** Data available status (only if device is invited). **0** Data is not available. **1** Data is available. **Bit 6:** Not applicable to displays. **Bit 7:** Requesting program device. **0** Not a requesting program device. **1** A requesting program device.

Table 97. Device Definition List (continued)

Offset	Data Type	Length	Contents
			Bit 8: DBCS device. **0** Device is not capable of processing double-byte data. **1** Device is capable of processing double-byte data. **Bits 9-10:** Reserved. **Bit 11:** DBCS keyboard. **0** Keyboard is not capable of entering double-byte data. **1** Keyboard is capable of entering double-byte data. **Bits 12-16:** Reserved.
78	Character	1	Not applicable to displays.
79	Character	1	Not applicable to displays.
80	Character	50	Reserved.

I/O Feedback Area

The results of I/O operations are communicated to the program using i5/OS messages and I/O feedback information. The I/O feedback area is updated for every I/O operation unless your program is using blocked record I/O. In that case, the feedback area is updated only when a block of records is read or written. Some of the information reflects the last record in the block. Other information, such as the count of I/O operations, reflects the number of operations on blocks of records and not the number of records. See your high-level language manual to determine if your program uses blocked record I/O.

The I/O feedback area consists of two parts: a common area and a file-dependent area.

Common I/O Feedback Area

Table 98. Common I/O Feedback Area

Offset	Data Type	Length	Contents
0	Binary	2	Offset to file-dependent feedback area.
2	Binary	4	Write operation count. Updated only when a write operation completes successfully. For blocked record I/O operations, this count is the number of blocks, not the number of records.
6	Binary	4	Read operation count. Updated only when a read operation completes successfully. For blocked record I/O operations, this count is the number of blocks, not the number of records.
10	Binary	4	Write-read operation count. Updated only when a write-read operation completes successfully.
14	Binary	4	Other operation count. Number of successful operations other than write, read, or write-read. Updated only when the operation completes successfully. This count includes update, delete, force-end-of-data, force-end-of-volume, change-end-of-data, release record lock, and acquire/release device operations.
18	Character	1	Reserved.

Table 98. Common I/O Feedback Area (continued)

Offset	Data Type	Length	Contents
19	Character	1	Current operation. **hex 01** Read or read block or read from invited devices **hex 02** Read direct **hex 03** Read by key **hex 05** Write or write block **hex 06** Write-read **hex 07** Update **hex 08** Delete **hex 09** Force-end-of-data **hex 0A** Force-end-of-volume **hex 0D** Release record lock **hex 0E** Change end-of-data **hex 0F** Put delete **hex 11** Release device **hex 12** Acquire device
20	Character	10	Name of the record format just processed, which is either: • Specified on the I/O request, or • Determined by default or format selection processing For display files, the default name is either the name of the only record format in the file or the previous record format name for the record written to the display that contains input-capable fields. Because a display file may have multiple formats on the display at the same time, this format may not represent the format where the last cursor position was typed.

Table 98. Common I/O Feedback Area (continued)

Offset	Data Type	Length	Contents
30	Character	2	Device class: Byte 1: **hex 00** Database **hex 01** Display **hex 02** Printer **hex 04** Diskette **hex 05** Tape **hex 09** Save **hex 0B** ICF Byte 2 (if byte 1 contains hex 01 for displays): **hex 07** 5251 Display Station **hex 0B** 5291 Display Station **hex 0D** 5292 Display Station **hex 12** 5555-B01 Display Station **hex 13** 3270 Display Station **hex 15** Graphic-capable device **hex 16** Financial terminal **hex 17** 3180 Display Station **hex 19** 3277 DHCF device **hex 26** 3179-2 Display Station **hex 27** 3196-A Display Station **hex 28** 3196-B Display Station **hex 33** 3197-C1 Display Station **hex 34** 3197-C2 Display Station **hex 35** 3197-D1 Display Station **hex 36** 3197-D2 Display Station **hex 37** 3197-W1 Display Station **hex 38** 3197-W2 Display Station **hex 39** 5555-E01 Display Station **hex 3E** 3476-EA Display Station **hex 3F** 3477-FG Display Station **hex 40** 3278 DHCF device **hex 41** 3279 DHCF device **hex 42** ICF finance device **hex 43** Retail communications device **hex 44** 3477-FA Display Station **hex 45** 3477-FC Display Station **hex 46** 3477-FD Display Station **hex 47** 3477-FW Display Station **hex 48** 3477-FE Display Station **hex 4D** Network Virtual Terminal display station

Table 98. Common I/O Feedback Area (continued)

Offset	Data Type	Length	Contents
32	Character	10	Device name. For displays, the name of the device for which the operation just completed.
42	Binary	4	Length of the record processed by the last I/O operation.
46	Character	80	Reserved.
126	Binary	2	Number of records retrieved on a read request for blocked records or sent on a write or force-end-of-data or force-end-of-volume request for blocked records.
128	Binary	2	For output, the field value is the record format length, including first-character forms control, option indicators, source sequence numbers, and program-to-system data. If the value is zero, use the field at offset 42. For input, the field value is the record format length, including response indicators and source sequence numbers. If the value is zero, use the field at offset 42.
130	Character	2	Reserved.
132	Binary	4	Not applicable to displays.
136	Character	8	Reserved.

I/O Feedback Area for Display Files

Table 99. I/O Feedback Area for Display Files

Offset	Data Type	Length	Contents
0	Character	2	Flag bits. **Bit 1:** Cancel-read indicator. **0** The cancel-read operation did not cancel the read request. **1** The cancel-read operation canceled the read request. **Bit 2:** Data-returned indicator. **0** The cancel-read operation did not change the contents of the input buffer. **1** The cancel-read operation placed the data from the read-with-no-wait operation into the input buffer. **Bit 3:** Command key indicator. **0** Conditions for setting this indicator did not occur. **1** The Print, Help, Home, Roll Up, Roll Down, or Clear key was pressed. The key is enabled with a DDS keyword, but without a response indicator specified. **Bits 4-16:** Reserved.

Table 99. I/O Feedback Area for Display Files (continued)

Offset	Data Type	Length	Contents
2	Character	1	Attention indicator byte (AID). This field identifies which function key was pressed. This field will always contain the value hex F1 to imitate the Enter key being pressed on a display device. For display files, this field will contain the 1-byte hexadecimal value returned from the device. **Hex Codes** **Function Keys** **hex 31** 1 **hex 32** 2 **hex 33** 3 **hex 34** 4 **hex 35** 5 **hex 36** 6 **hex 37** 7 **hex 38** 8 **hex 39** 9 **hex 3A** 10 **hex 3B** 11 **hex 3C** 12 **hex B1** 13 **hex B2** 14 **hex B3** 15 **hex B4** 16 **hex B5** 17 **hex B6** 18 **hex B7** 19 **hex B8** 20 **hex B9** 21 **hex BA** 22 **hex BB** 23 **hex BC** 24 **hex BD** Clear **hex F1** Enter/Rec Adv **hex F3** Help (not in operator-error mode) **hex F4** Roll Down **hex F5** Roll Up **hex F6** Print **hex F8** Record Backspace **hex 3F** Automatic Enter (for Selector Light Pen)
3	Character	2	Cursor location (line and position). Updated on input operations that are not subfile operations that return data to the program. For example, hex 0102 means line 1, position 2. Line 10, position 33 would be hex 0A21.

Table 99. I/O Feedback Area for Display Files (continued)

Offset	Data Type	Length	Contents
5	Binary	4	Actual data length. The length of the record format processed by the I/O operation.
9	Binary	2	Relative record number of a subfile record. Updated for a subfile record operation. For input operations, updated only if data is returned to the program. If multiple subfiles are on the display, this offset will contain the relative record number for the last subfile updated.
11	Binary	2	Indicates the lowest subfile relative record number currently displayed in the uppermost subfile display area if the last write operation was done to the subfile control record with SFLDSP specified. Updated for roll up and roll down operations. It is reset to zero when an output operation is performed to any format other than the subfile control format that initially set this value. Not set for message subfiles.
13	Binary	2	Total number of records in a subfile. Updated on a put-relative operation to any subfile record. The number is set to zero on a write or write-read operation to any subfile control record with the SFLINZ keyword optioned on. If records are put to multiple subfiles on the display, this offset will contain the total number of records for all subfiles assuming that no write or write-read operations were performed to any subfile control record with the SFLINZ keyword optioned on.
15	Character	2	Cursor location relative to active DDS WINDOW keyword (line and position). Updated on input operations that are not subfile operations that return data to the program. The cursor location is based off the usable positions within the window. For example, hex 0C13 means line 12, position 19.
17	Character	17	Reserved.
34	Character	2	Major return code. **00** Operation completed successfully. **02** Input operation completed successfully, but job is being canceled (controlled). **03** Input operation completed successfully, but no data received. **04** Output exception. **08** Device already acquired. **11** Read from invited devices was not successful. **34** Input exception. **80** Permanent system or file error. **81** Permanent session or device error. **82** Acquire or open operation failed. **83** Recoverable session or device error.
36	Character	2	Minor return code. For the values for a display file, see Appendix D, “Display File Return Codes,” on page 677.
38	Character	8	Not applicable to displays.
46	Character	1	Not applicable to displays.
47	Character	1	Reserved.
48	Character	1	Not applicable to displays.
49	Character	10	Not applicable to displays.
59	Character	4	Reserved.
63	Character	8	Not applicable to displays.
71	Character	9	Reserved.

Get Attributes

Performing the get attributes operation allows you to obtain certain information about a specific display device.

Table 100. Get Attributes

Offset	Data Type	Length	Contents
0	Character	10	Program device name.
10	Character	10	Device description name. Name of the device description associated with this entry.
20	Character	10	User ID.
30	Character	1	Device class. For displays, the device class is D.

Table 100. Get Attributes (continued)

Offset	Data Type	Length	Contents	
31	Character	6	Device type:	
			3179	3179 Display Station
			317902	3179-2 Display Station
			3180	3180 Display Station
			3196A	3196-A1/A2 Display Station
			3196B	3196-B1/B2 Display Station
			3197C1	3197-C1 Display Station
			3197C2	3197-C2 Display Station
			3197D1	3197-D1 Display Station
			3197D2	3197-D2 Display Station
			3197W1	3197-W1 Display Station
			3197W2	3197-W2 Display Station
			3270	3270 Display Station
			3476EA	3476-EA Display Station
			3476-EC	3476-EC Display Station
			3477FA	3477-FA Display Station
			3477FC	3477-FC Display Station
			3477FD	3477-FD Display Station
			3477FE	3477-FE Display Station
			3477FG	3477-FG Display Station
			3477FW	3477-FW Display Station
			525111	5251 Display Station
			5291	5291 Display Station
			5292	5292 Display Station
			529202	5292-2 Display Station
			5555B1	5555-B01 Display Station
			5555C1	5555-C1 Display Station
			5555E1	5555-E01 Display Station
			5555F1	5555-F1 Display Station
			5555-G1	5555-G01 Display Station
			5555-G2	5555-G02 Display Station
			3486BA	3486-BA Display Station
			3486BG	3486-BG Display Station
			3487HC	3487-HC Display Station
			3487HD	3487-HD Display Station
			3487HE	3487-HE Display Station
			3487HW	3487-HW Display Station
			DHCF77	3277 DHCF device
			DHCF78	3278 DHCF device
			DHCF79	3279 DHCF device

Table 100. Get Attributes (continued)

Offset	Data Type	Length	Contents
37	Character	1	Requesting program device. This flag indicates whether this entry is defining a *REQUESTER device. **N** Not a *REQUESTER device (communications source device). **Y** A *REQUESTER device (communications target device).
38	Character	1	Acquire status. Set even if device is implicitly acquired at open time. **N** Device is not acquired. **Y** Device is acquired.
39	Character	1	Invite status. **Y** Device is invited. **N** Device is not invited.
40	Character	1	Data available. **Y** Invited data is available. **N** Invited data is not available.
41	Binary	2	Number of rows on display.
43	Binary	2	Number of columns on display.
45	Character	1	Display allow blink. **Y** Display is capable of blinking. **N** Display is not capable of blinking.
46	Character	1	Online/offline status. **O** Display is online. **F** Display is offline.
47	Character	1	Display location. **L** Local display. **R** Remote display.
48	Character	1	Display type. **A** Alphanumeric or Katakana. **I** DBCS.
49	Character	1	Keyboard type of display. **A** Alphanumeric or Katakana keyboard. **I** DBCS keyboard.
50	Character	1	Not applicable to displays.
51	Character	1	Not applicable to displays.
52	Character	1	Not applicable to displays.
53	Character	8	Not applicable to displays.
61	Character	8	Not applicable to displays.
69	Character	8	Not applicable to displays.
77	Character	8	Not applicable to displays.
85	Character	8	Not applicable to displays.
93	Character	8	Not applicable to displays.

Table 100. Get Attributes (continued)

Offset	Data Type	Length	Contents
101	Character	1	Controller information. **N** Display is not attached to a controller that supports an enhanced interface for nonprogrammable work stations. **1** Display is attached to a controller (type 1) [5] that supports an enhanced interface for nonprogrammable work stations. **2** Display is attached to a controller (type 2) [5] that supports an enhanced interface for nonprogrammable work stations. **3** Display is attached to a controller (type 3) [5] that supports an enhanced interface for nonprogrammable work stations.
102	Character	1	Color capability of display. **Y** Color display **N** Monochrome display
103	Character	1	Grid line support by display. **Y** Display supports grid lines **N** Display does not support grid lines
104	Character	1	Not applicable to displays.
105	Character	8	Not applicable to displays.
113	Character	31	Reserved.

[5]

Type 1 Controllers available at V2R2 which support such things as windows, selection fields, scroll bars, and continued cursor progression.

Type 2 Controllers available at V2R3. These support all of the V2R2 functions as well as menu bars, continued-entry fields, edit masks, and simple hotspots.

Type 3 Controllers available at V3R1. These support all of the V2R2 and V2R3 functions. They also support text in the bottom border of windows.

Appendix D. Display File Return Codes

This section contains descriptions of all major and minor return codes for display files. These return codes are set in the I/O feedback area of the display file. The return codes report the results of each operation. The appropriate return code is available to the application program that issued the operation. The program then checks the return code and acts accordingly. Refer to your high-level language manual for information about how to access these return codes.

The return code is a four-digit value: the first two digits contain the major code, and the last two digits contain the minor code. With some return codes, a message is also sent to the job log or the operator message queue (QSYSOPR). You can refer to the message for additional information.

Major Code 00

Major Code 00–Operation completed successfully.

Description: The operation issued by your program completed successfully.

Action: Continue with the next operation.

Code **Description/Action**

0000 **Description:** For input operations performed by your program, 0000 indicates that some data was received on a successful input operation.

For output operations performed by your program, 0000 indicates that the last output operation completed successfully.

Action: Your program may continue. One of the messages listed below may have been issued to warn of an unusual condition that may be significant to your program even though it is not an error.

Messages:

```
CPF4018 (Status)
CPF4019 (Diagnostic)
CPF4054 (Diagnostic)
CPF4082 (Diagnostic)
CPF4410 (Diagnostic)
CPF5003 (Status)
CPF5508 (Diagnostic)
```

Major Code 02

Major Code 02–Input operation completed successfully, but your job is being ended (controlled).

Description: The input operation issued by your program was completed successfully. However, your job is being ended (controlled).

Action: Your program should complete its display processing as soon as possible to allow your program to complete in an orderly manner. The system eventually changes a job ended (controlled) to a job ended (immediate) and forces all processing to stop for your job.

Code **Description/Action**

0200 **Description:** On a successful input operation, an indication was received that a job ended (controlled) request is pending. Also, 0200 indicates that some data was received.

Action: Your program may continue. However, the recommended action is to complete the display processing and end the program, because the system will eventually cancel your job and cause all processing to stop for your job.

Major Code 03

Major Code 03–Input operation completed successfully, but no data received.

Description: The input operation was completed successfully, but no data was received.

Action: Check the minor return code for additional information, and continue with the next operation.

Code **Description/Action**

0300 **Description:** No data was received on a successful input operation. Examples of conditions causing this are no data being available on a get-relative or a get-next-changed operation to a subfile record format.

Action: Continue with whatever processing is appropriate. For example, if your program issued a get-next-changed operation to a subfile record format, then a 0300 indicates that there are no more subfile records changed by the user and no more user input data to process.

Messages:

`CPF5017 (Notify)`

`CPF5020 (Notify)`

`CPF5037 (Notify)`

0309 **Description:** Your program is being ended (controlled). No data was received.

This return code is only applicable to the read-from-invited-devices operation.

Action: Your program can continue processing. However, the recommended action is to complete the display processing and end the program, because the system will eventually cancel your job and cause all processing to stop for your job.

Messages:

`CPF4741 (Notify)`

0310 **Description:** The time interval specified by the WAITRCD value for the display file has ended.

This return code is only applicable to the read-from-invited-devices operation.

Note: Because no device is associated with the completion of this operation, the device name in the I/O feedback area contains an *N.

Action: Issue the operation to perform the intended function after the specified time interval has ended. For example, if you are using the time interval to control the length of time to wait for data, you can then issue another read-from-invited-devices operation to receive the data.

Messages:

`CPF4742 (Status)`

`CPF4743 (Status)`

Major Code 04

Major Code 04–Output exception occurred.

Description: An output exception occurred because your program attempted to send output when it should have been receiving the data that had already been sent by the display. Your data, associated with this output operation, was not sent to the display. Your program can attempt to send its output later.

Action: Issue an input operation to receive the data.

Code **Description/Action**

0412 **Description:** An output exception occurred because your program attempted to send data when it should have been receiving the data that had already been sent by the display. Your program data was not sent and should be sent later, after the data from the display has been received.

Action: Issue an input operation to receive the data.

Messages:

```
CPF4737 (Notify)
```

Major Codes 08–11

Major Codes 08 and 11–Miscellaneous program errors occurred.

Description: The operation just attempted by your program was not successful. The operation may have failed because it was issued at the wrong time.

Action: Refer to the individual return code descriptions for the appropriate recovery actions.

Code **Description/Action**

0800 **Description:** The acquire operation just performed was not successful. It tried to acquire a device that had already been acquired.

Action: If the display device requested by the original acquire operation is the one needed, your program can begin using it because it is already available. If a different device is required, issue another acquire operation for a different device name.

Messages:

```
CPD4077 (Diagnostic)
CPF50A0 (Status)
```

1100 **Description:** The read-from-invited-devices operation was not successful because no devices were invited.

Action: Issue an invite function followed by a read-from-invited-devices operation.

Messages:

```
CPF4740 (Notify)
```

Major Code 34

Major Code 34–Input exception occurred.

Description: The input operation attempted by your program was not successful. The data received was too long for the record format specified on the input operation.

Action: Refer to the individual return code descriptions for the appropriate recovery actions.

Code **Description/Action**

3431 **Description:** The input operation issued by your program was not successful because the length of the data received from the display exceeds the receive data length specified in the user-defined data stream. The data received is truncated.

This return code is only applicable to input operations using a record format specifying a user-defined data stream (USRDFN DDS keyword).

Action: Close the device file and end your program. Then, change your program so that the input record length is at least as long as the data record to be received.

Messages:

```
CPF5062 (Notify)
```

Major Code 80

Major Code 80–Permanent system or file error (not recoverable).

Description: A file or system error that is not recoverable has occurred. Recovery is unlikely until the problem causing the error has been corrected.

Action: The following general actions can be taken by your program for each 80xx return code. Other specific actions are given in each return code description.

- Continue processing without the display.
- Close the device file and open the file again. If the operation is still unsuccessful, try it again only a limited number of times. (The number of times should be specified in your program.)
- End.

Code **Description/Action**

8081 **Description:** The operation was not successful because a system error condition was detected.

Action: Your display device may need to be varied off and then on again. Your program can either:

- Continue processing without the display device.
- Close the device file and open the file again.
- End.

Messages:

CPF4182 (Escape)

CPF4510 (Escape)

CPF5192 (Escape)

CPF5257 (Escape)

CPF5403 (Escape)

CPF5416 (Escape)

CPF5418 (Escape)

CPF5423 (Escape)

CPF5429 (Escape)

CPF5431 (Escape)

CPF5404 (Escape)

CPF5405 (Escape)

CPF5408 (Escape)

CPF5409 (Escape)

CPF5410 (Escape)

CPF5411 (Escape)

CPF5414 (Escape)

CPF5415 (Escape)

CPF5433 (Escape)

CPF5434 (Escape)

CPF5441 (Escape)

CPF5447 (Escape)

CPF5455 (Escape)

CPF5456 (Escape)

CPF5507 (Escape)

8082 **Description:** The operation was not successful because the display device is unusable. This may occur because a cancel reply has been taken to an error recovery message for the device or because the display has been held by a Hold Communications Device (HLDCMNDEV) command. No operations should be issued to the device.

Action: Communications with the display cannot be resumed until the device has been reset to a varied-on state. If the device has been held, use the Release Communications Device (RLSCMNDEV) command to reset the device. If the device is in an error state, vary the device off and then on again. Once the device is reset, normal operation can be started by opening the display device file again. Your program can either:

- Continue processing without the display device.
- Close the device file and open the file again.
- End.

Messages:

CPF4354 (Escape)

CPF5269 (Escape)

80A6 **Description:** A Systems Network Architecture (SNA) unbind operation was not successful on a close or release operation. This may be the result of a device configuration error. The device may be unusable. No operations should be issued to the device.

Action: Refer to the device response code in the accompanying error message to determine the cause of the failure. Vary the device off and then on again to reset the error. Correct the error and try your program again.

Messages:

CPF4527 (Escape)

80B3 **Description:** The open operation was not successful because the display file is not available. The file cannot be opened again until the necessary resources are available.

Action: Your program can wait for the display file to become available, then issue another open operation. Otherwise, you may continue other processing or end the program.

Consider increasing the WAITFILE parameter with the Change Display File (CHGDSPF) command or Override with Display File (OVRDSPF) command to allow more time for the file to become available.

Messages:

CPF4128 (Escape)

80C0 **Description:** An error that is not recoverable has occurred on the display device.

Action: Your display devices may need to be varied off and then on again. Your program can either:

- Continue processing without the display station.
- Close the device file and open the file again.
- End.

Messages:

CPF5103 (Escape)
CPF5192 (Escape)
CPF5412 (Escape)
CPF5413 (Escape)
CPF5419 (Escape)
CPF5420 (Escape)
CPF5421 (Escape)
CPF5430 (Escape)
CPF5437 (Escape)
CPF5439 (Escape)

80EB **Description:** An open operation was not successful because an open option that is not valid or a combination of options that is not valid was specified in your program, in the display file, or in an override command.

Action: Close the display file, correct the problem, and open the file again. See the individual messages to determine what options are not valid.

Messages:

CPF4062 (Escape)
CPF4129 (Escape)
CPF4148 (Escape)
CPF4156 (Escape)
CPF4163 (Escape)
CPF4169 (Escape)
CPF4191 (Escape)
CPF4238 (Escape)
CPF4345 (Escape)
CPF5151 (Escape)
CPF5510 (Escape)
CPF5511 (Escape)
CPF5512 (Escape)
CPF5513 (Escape)
CPF5552 (Escape)

80ED **Description:** The open operation was not successful because the record format descriptions in the file have changed since your program was compiled.

Action: Close the file and end the program. Determine whether the changes affect your application program. If they do, recompile the program. If the changes do not affect your program, the file should be changed or overridden to LVLCHK(*NO). When LVLCHK(*NO) is specified, the system does not compare the record format descriptions.

Messages:
CPF4131 (Escape)

80F8 **Description:** An operation to a file was not successful because the file is marked in error.

Action: Close the file. Refer to messages in the job log to determine what errors occurred. Take the appropriate action for those errors.

Messages:

CPF4132 (Escape)
CPF4213 (Escape)
CPF4550 (Escape)
CPF5129 (Escape)
CPF5144 (Escape)
CPF5427 (Escape)

Major Code 81

Major Code 81–Permanent device error (not recoverable).

Description: A device-related error that is not recoverable occurred during an I/O operation. Any attempt to continue using this display device will probably fail again until the cause of the problem is found and corrected, but operations directed to other display devices associated with the file should be expected to work.

Action: The following general actions can be taken for each 81xx return code. Other specific actions are given in each return code description.

- Continue processing without the display device.
- Release the device or close the file, correct the problem, and acquire the device again or open the file. If the operation is still unsuccessful, try it again only a limited number of times. (The number of times should be specified in your program.)
- End.

Several return codes indicate that an error condition must be corrected by varying the device off and on again.

Code **Description/Action**

8181 **Description:** A system error condition was detected during the I/O operation to the device.

Action: Release the device in error or close the file. You may need to vary the device off and on again to clear the error. Determine the cause of the failure from the accompanying message. Check for any system operator messages indicating that additional corrective action must be performed. Open the file again or acquire the device to continue.

Messages:

CPF4553 (Escape)

CPF4725 (Escape)

CPF5105 (Escape)

CPF5189 (Escape)

CPF5254 (Escape)

8191 **Description:** The operation was not successful because a permanent line error occurred, and the system operator took a recovery option in response to the line error message. (You can find out what type of line error occurred by asking the system operator.) The device has been marked unusable.

Action: Release the device in error or close the file. Vary the device off and on again to clear the error. Open the file again or acquire the device to continue.

Messages:

CPF4526 (Escape)

CPF4542 (Escape)

CPF4551 (Escape)

CPF5128 (Escape)

CPF5143 (Escape)

CPF5198 (Escape)

8197 **Description:** An error condition that is not recoverable was detected at the device. An example of such an error is the user turning off the display device.

Action: Release the device in error or close the file. The display device may need to be varied off and then on again to clear the error. Refer to the accompanying error message for additional information regarding the source of the specific error detected. Open the file or acquire the device again to continue.

Messages:

CPF4149 (Escape)

CPF4197 (Escape)

CPF4524 (Escape)

CPF4533 (Escape)

CPF4538 (Escape)

CPF5047 (Escape)

CPF5106 (Escape)

CPF5140 (Escape)

CPF5143 (Escape)

CPF5199 (Escape)

CPF5265 (Escape)

81C2 **Description:** The operation issued by your program was not successful because the Systems Network Architecture (SNA) session with the display is not active.

Action: Release the device or close the file. Vary the device off and on again to clear the error. Open the file or acquire the device again to continue.

Messages:

CPF5170 (Escape)

CPF5422 (Escape)

Major Code 82

Major Code 82–Open or acquire operation failed.

Description: An attempt to open the display file or acquire the display device was not successful. The error may be recoverable or permanent, but is limited to the specific display device. Recovery is unlikely until the problem causing the error has been corrected.

Action: The following general actions can be taken for each 82xx return code. Other specific actions are given in each return code description. You can either:

- Continue processing without the device.
- Release the device or close the file, correct the problem, and acquire the device or open the file again. A subsequent operation could be successful if the error occurred because of some temporary condition such as the device being in use at the time.

  If the operation is still unsuccessful, try it again only a limited number of times. (The number of times should be specified in your program.)

  – If the attempted operation was an acquire operation, release the device and issue the acquire operation again.

  – If the attempted operation was an open operation, close the file and issue the open operation again.
- End.

Several return codes indicate that an error condition must be corrected by changing a value in the file. To change a parameter value for the file, use the Change Display File (CHGDSPF) or Override with Display File (OVRDSPF) command.

Code **Description/Action**

8281 **Description:** A system error condition was detected on an open or acquire operation. The file may previously have been in error, or the file could not be opened due to a system error.

Action: Determine the cause of the failure from the accompanying message. Check for any system operator messages indicating that additional corrective action must be performed.

The display device may need to be varied off and then on again to clear the error. Your program can either:

- Continue processing without the display device.
- Release the device or close the file, correct the problem, and acquire the device or open the file again.
- End.

Messages:

CPF4168 (Escape)

CPF4182 (Escape)

CPF4221 (Escape)

CPF5105 (Escape)

CPF5254 (Escape)

CPF5257 (Escape)

CPF5410 (Escape)

CPF5411 (Escape)

CPF5424 (Escape)

CPF5447 (Escape)

CPF5455 (Escape)

CPF919E (Escape)

8282 **Description:** The open or acquire operation was not successful because the display device is unusable. This may occur because a cancel reply has been taken to an error recovery message for the device or because the display has been held by a Hold Communications Device (HLDCMNDEV) command. No operations should be issued to the device. Communications with the display cannot be resumed until the device has been reset to a varied-on state.

Action: Close the display device file. If the device has been held, use the Release Communications Device (RLSCMNDEV) command to reset the device. If the device is in an error state, vary the device off and then on again. Once the device is reset, start normal operation by opening the display device file again.

Messages:

CPF4171 (Escape)

CPF4354 (Escape)

CPF5548 (Escape)

8291 **Description:** A permanent line error occurred on an open or acquire operation. The device has been marked unusable.

Action: Release the device in error or close the file. Vary the device off and on again to clear the error. Open the file again to continue.

Messages:

CPF4146 (Escape)

CPF4179 (Escape)

CPF4193 (Escape)

CPF4291 (Escape)

CPF5198 (Escape)

CPF5260 (Escape)

8297 **Description:** The open or acquire operation has ended abnormally due to an error condition detected at the display device that is not recoverable. An example of such an error is the user turning off the display device.

Action: Release the device in error or close the file. The display device may need to be varied off and then on again to clear the error. Refer to the accompanying error message for additional information regarding the source of the specific error detected. Open the file or acquire the device again to continue.

Messages:

CPF4192 (Escape)

CPF5047 (Escape)

CPF5106 (Escape)

CPF5140 (Escape)

CPF5143 (Escape)

CPF5199 (Escape)

82A6 **Description:** The open or acquire operation failed because the Systems Network Architecture (SNA) bind command was not successful.

Action: Ensure that the display device with which your program is communicating is configured properly. Refer to the device response codes in the accompanying error message for additional information regarding the specific error detected.

Messages:

CPF4124 (Escape)

CPF4190 (Escape)

CPF5103 (Escape)

CPF5517 (Escape)

82A8 **Description:** The acquire operation was not successful because the maximum number of devices allowed for the display file has been reached.

Action: Your program can recover by releasing a different device and issuing the acquire operation again. If more devices are needed, close your file and increase the MAXDEV value in the display file.

Messages:

CPF4745 (Escape)

CPF5041 (Status)

CPD4757 (Diagnostic)

82A9 **Description:** The acquire operation was not successful because the requesting program device is not available. The requesting program device may not be available because your program is not running in an interactive job.

Action: Your program can continue without the display, attempt to use a different display device, or end.

If your program needs to use the requesting program device, make sure that it runs in an interactive job.

Messages:

CPF4366 (Escape)

CPF5381 (Escape)

82AA **Description:** The open or acquire operation was not successful because the display device description was not found.

Action: Your program can continue without the display, attempt to use a different display device, or end.

Verify that the name of the display device was correctly specified in the DEV parameter on the CRTDSPF, CHGDSPF, CRTDEVDSP, or OVRDSPF command.

Messages:

CPF4103 (Escape)

CPF4747 (Escape)

82AB

Description: The open or acquire operation was not successful because the display device was not varied on.

Action: Your program can continue without the display, attempt to acquire a different display device, or end. Vary on the display device and attempt the open or acquire operation again.

If you want your program to continue with the same display, release the device or close the file, correct the problem, and acquire the device or open the file again.

Messages:

CPF4285 (Escape)

CPF5333 (Escape)

82B3

Description: The open or acquire operation was not successful because the display device you are acquiring is in use in another process.

Action: Wait for the display device to become available, then issue the acquire operation again. Otherwise, you may continue other processing without the display, or end the program.

You can use the Work with Configuration Status (WRKCFGSTS) command to determine which job is using the display device.

Consider increasing the WAITFILE parameter of the CHGDSPF or OVRDSPF command to allow more time for the device to become available.

Messages:

CPF4109 (Escape)

CPF4130 (Escape)

CPF4282 (Escape)

CPF5217 (Escape)

CPF5332 (Escape)

82EE

Description: An open or acquire operation was attempted to a device that is not supported for a display file.

Your program is attempting to acquire a device that is not a valid display device; or it is trying to acquire the requesting program device, but the requesting program device for the job is a communications device, not a display device.

Action: Your program can continue without the display, attempt to acquire a different display device, or close the file and end.

Verify that the name of the display device was specified correctly on the CHGDSPF, CRTDEVDSP, CRTDSPF, or OVRDSPF command.

If your program was attempting to acquire the requesting program device, verify that your program is running in an interactive job so that the requesting program device is a display device.

Messages:

CPF4105 (Escape)

CPF4223 (Escape)

CPF4760 (Escape)

CPF5038 (Escape)

82EF **Description:** An open or acquire operation was attempted for a device that the user is not authorized to or that is in service mode.

Action: Your program can continue without the display, attempt to acquire a different display device, or end.

If the operation was an open operation, close the file, correct the problem, and then issue the open operation again. If the operation was an acquire operation, correct the problem and issue the acquire operation again.

For authority errors, obtain authority to the device from your security officer or device owner. If the device is in service mode, the system service tools (SST) function is currently using the device. Wait until the device is available to issue the operation again.

Messages:

CPF4104 (Escape)

CPF4186 (Escape)

CPF5278 (Escape)

CPF5279 (Escape)

82F8 **Description:** Your program attempted an acquire operation to a device that is marked in error due to a previous error during an I/O or acquire operation.

Action: Release the device or close the file, correct the previous error, and acquire the device or open the file again.

Messages:

CPF5293 (Escape)

Major Code 83

Major Code 83–Device error occurred (recoverable).

Description: An error occurred during an I/O operation, but the display device is still usable. Recovery within your program might be possible.

Action: The following general actions can be taken for each 83xx return code. Other specific actions are given in each return code description.

- Continue processing without the display device.
- Correct the problem and continue processing with the display device. If the attempt to recover from the operation is unsuccessful, try it again only a limited number of times. (The number of times should be specified in your program.)
- End.

Several return codes indicate that an error condition must be corrected by changing a value in the display file. To change a parameter value for the file, use the Change Display File (CHGDSPF) or Override with Display File (OVRDSPF) command.

Code **Description/Action**

830B **Description:** Your program has attempted to issue an input or output operation either before the device was acquired or after it was released.

Your program may have improperly handled a *permanent device* or *acquire failed* error.

Action: Verify that your program tries no input or output operation with a display device that is not connected to the file and that return codes from acquire or I/O operations are handled properly.

Messages:

CPD4079 (Diagnostic)

CPF4739 (Status)

CPF5067 (Escape)

CPF5068 (Escape)

CPF5070 (Escape)

CPF5170 (Escape)

CPF5217 (Escape)

831D **Description:** The operation just attempted by your program was rejected because a parameter was not valid, out of limits, or missing.

Action: Your program can bypass the failing step and continue, or close the file and end. Refer to the accompanying message to determine what parameter was incorrect. Correct the error in your program before attempting to try the operation again.

Messages:

CPF4912 (Notify)

CPF5002 (Notify)

CPF5008 (Notify)

CPF5012 (Notify)

CPF5014 (Notify)

CPF5021 (Notify)

CPF5218 (Escape)

CPF5302 (Escape)

CPF5303 (Escape)

CPF5398 (Escape)

831E **Description:** The operation just issued by your program was not valid or a combination of operations that is not valid was specified. The error may have been caused by one of the following:

- Either your program issued an operation with an unrecognized code, or the operation specified by the code or DDS keyword is not supported by the display.
- A combination of operations or keywords that is not valid was requested.
- A user-defined data stream contained a command that is not valid for the display.

Action: Your program can bypass the operation that is not valid and continue, or close the file and end. Refer to the accompanying message to determine why the operation was rejected. Correct the error in your program before attempting to try the failing operation again.

Messages:

CPF4564 (Escape)

CPF5005 (Notify)

CPF5011 (Notify)

CPF5039 (Notify)

CPF5045 (Notify)

CPF5051 (Notify)

CPF5055 (Notify)

CPF5056 (Notify)

CPF5059 (Notify)

CPF5066 (Notify)

CPF5149 (Escape)

831F **Description:** A length that is not valid was specified on the operation.

On an output operation, your program has tried to send a data record having a length that exceeds the maximum record length allowed for the display device. The data has been truncated.

Action: Issue the output operation again with a smaller output length. The record length for a non-field-level display file cannot exceed the display size. The record length for any display file must be no greater than 32 763 characters.

Messages:

CPF4010 (Diagnostic)

CPF4078 (Diagnostic)

8322 **Description:** The attempted operation is not valid in the current state. Either a write operation was attempted while your program was not in the send state or a subfile operation was attempted when the subfile was not active.

Action: Your program can bypass the operation that is not valid and continue, or close the file and end. Correct the sequence of operations in your program before attempting to run the job again.

Messages:

CPF5013 (Notify)

CPF5060 (Notify)

832D **Description:** Your program attempted an operation that is not valid when an invite operation is outstanding. Once you have issued an invite or get-no-wait operation, another invite operation cannot be issued for the same display device until the first invite has been completed by a read or read-from-invited-devices operation.

Action: Issue an input operation to receive the data that was invited before issuing another invite operation. Otherwise, close the file and end. If a coding error in your program caused the error, correct the sequence of operations in your program.

Messages:

CPF5052 (Notify)

8343 **Description:** An attempt was made to add another record to a subfile after the subfile was full.

Action: Your program can clear the subfile, or continue without adding more records to the subfile. Otherwise, close the file and end.

Increase the subfile length in the DDS statements. If a coding error in your program caused the error, correct your program.

Messages:

CPF5043 (Notify)

83E0 **Description:** Your program attempted to issue an operation using a record format that was not defined for the display file, or omitted the record format name.

Action: Check the name of the record format in your program to be sure it is correct. Then check that the record format is defined properly in the DDS for the file.

Messages:

CPF5022 (Notify)

CPF5023 (Notify)

CPF5053 (Notify)

CPF5054 (Notify)

83E1 **Description:** An error occurred during an I/O operation. The requesting program device was set to automatically disconnect your program. The same user signed on to the same display device, so the program was started up again. The display was cleared of any data present when the error occurred.

Action: Branch to a normal starting point in your program and rewrite the display. There is no need to perform a close or open operation on the display files that were active when the error occurred.

Messages:

`CPF509F (Notify)`

83E8 **Description:** Your CL program issued an End Receive (ENDRCV) command when there is no outstanding read-with-no-wait operation.

Action: Your program can issue an output operation to continue sending, issue an input operation to begin receiving, or close the file and end. Correct the error in your program before attempting to repeat the failing operation.

Messages:

`CPF4910 (Notify)`

83F6 **Description:** Your program sent data to the display that is not valid. The data type may not be correct for the field in which it is used.

Action: Check the name of the record format in your program to be sure it is correct. Verify that the data definition statements in your program match the output record defined in the DDS for the file. Correct the error in your program before attempting to repeat the failing operation.

Messages:

`CPF5063 (Notify)`

`CPF5216 (Escape)`

`CPF5301 (Escape)`

83F8 **Description:** Your program attempted an I/O operation to a device that is marked in error due to a previous error during an I/O or acquire operation.

Action: Release the device or close the file, correct the previous error, and acquire the device or open the file again.

Messages:

`CPF5293 (Escape)`

Appendix E. Edit Codes

i5/OS Edit Codes

The following table summarizes the functions provided by i5/OS edit codes.

Table 101. Summary Chart for i5/OS Edit Codes

Edit Codes	Commas[1] Displayed	Decimal Points[1] Displayed	Sign Displayed When Negative Value	Blank Value of QDECFMT System Value	I Value of QDECFMT System Value	J Value of QDECFMT System Value	Leading Zero Suppressed
1	Yes	Yes	No sign	.00 or 0	,00 or 0	0,00 or 0	Yes
2	Yes	Yes	No sign	Blanks	Blanks	Blanks	Yes
3		Yes	No sign	.00 or 0	,00 or 0	0,00 or 0	Yes
4		Yes	No sign	Blanks	Blanks	Blanks	Yes
A	Yes	Yes	CR	.00 or 0	,00 or 0	0,00 or 0	Yes
B	Yes	Yes	CR	Blanks	Blanks	Blanks	Yes
C		Yes	CR	.00 or 0	,00 or 0	0,00 or 0	Yes
D		Yes	CR	Blanks	Blanks	Blanks	Yes
J	Yes	Yes	−(Minus)	.00 or 0	,00 or 0	0,00 or 0	Yes
K	Yes	Yes	−(Minus)	Blanks	Blanks	Blanks	Yes
L		Yes	−(Minus)	.00 or 0	,00 or 0	0,00 or 0	Yes
M		Yes	−(Minus)	Blanks	Blanks	Blanks	Yes
N	Yes	Yes	−(Minus)	.00 or 0	,00 or 0	0,00 or 0	Yes
O	Yes	Yes	−(Minus)	Blanks	Blanks	Blanks	Yes
P		Yes	−(Minus)	.00 or 0	,00 or 0	0,00 or 0	Yes
Q		Yes	−(Minus)	Blanks	Blanks	Blanks	Yes
Y[2]							Yes
Z[3]							Yes

Table 101. Summary Chart for i5/OS Edit Codes (continued)

Edit Codes	Commas[1] Displayed	Decimal Points[1] Displayed	Sign Displayed When Negative Value	Blank Value of QDECFMT System Value	I Value of QDECFMT System Value	J Value of QDECFMT System Value	Leading Zero Suppressed

Notes:

[1] The QDECFMT system value determines the decimal point character (period in U.S. usage), the character used to separate groups of three digits (comma in U.S. usage), and the type of zero suppression (depending on comma and period placement). For more information on the QDECFMT system value, see the *Work Management* book.

[2] The Y edit code suppresses the farthest left zero of a date field that is three to six digits long, and it suppresses the two farthest left zeros of a field that is seven positions long. The Y edit code also inserts slashes (/) between the month, day, and year according to the following pattern:

nn/n
nn/nn
nn/nn/n
nn/nn/nn
nnn/nn/nn

If the DATE keyword is specified with EDTCDE(Y), the separator character used is the job attribute, DATSEP. The slash (/) is the default DATSEP. If, at file creation time, DATFMT is JUL (Julian), the date is normally formatted as nn/nnn and EDTCDE(Y) is not valid.

[3] The Z edit code removes the sign (plus and minus) from a numeric field. The sign of the units position is changed to a hexadecimal F before the field is written.

Note: Edit code X can be specified but has no effect on display files and is ignored. The system operates with a preferred sign of F, which is equivalent to using edit code X. Edit code X causes a blank keyboard shift (position 35) to default to numeric only (attribute Y), but has no other effect on the display file and is ignored for all other processing. The display length of the field is determined by the keyboard shift and not by edit code X (the default numeric-only Y attribute may add 1 position to the field for decimals).

Examples of Editing Using i5/OS Edit Codes

The following table shows valid edit codes with examples of unedited source data and edited output. Zero suppression and decimal characters are determined by the system value QDECFMT. The date separator character is determined by the job attribute DATSEP. In this figure, QDECFMT is assumed to equal x, and DATSEP is assumed to equal /.

Table 102. Valid Edit Codes, Source Data, and Edited Output

Edit Codes	Positive Number– Two Decimal Positions	Positive Number– No Decimal Positions	Negative Number– Three Decimal Positions[1]	Negative Number– No Decimal Positions[1]	Zero Balance– Two Decimal Positions[1]	Zero Balance– No Decimal Positions[1]
Unedited	1234567	1234567	xxxx.125–	xxxx.125–	xxxxxx	xxxxxx
1	12,345.67	1,234,567	.125	125	.00	0
2	12,345.67	1,234,567	.125	125		
3	12345.67	1234567	.125	125	.00	0
4	12345.67	1234567	.125	125		
A	12,345.67	1,234,567	.125CR	125CR	.00	0
B	12,345.67	1,234,567	.125CR	125CR		
C	12345.67	1234567	.125CR	125CR	.00	0
D	12345.67	1234567	.125CR	125CR		
J	12,345.67	1,234,567	.125–	125–	.00	0
K	12,345.67	1,234,567	.125–	125–		
L	12345.67	1234567	.125–	125–	.00	0
M	12345.67	1234567	.125–	125–		
N	12,345.67	1,234,567	–.125	–125	.00	0
O	12,345.67	1,234,567	–.125	–125		
P	12345.67	1234567	–.125	–125	.00	0
Q	12345.67	1234567	–.125	–125		
Y[2]	123/45/67	123/45/67	0/01/25	0/01/25	0/00/00	0/00/00
Z[3]	1234567	1234567	125	125		

Notes:

[1] The x represents a blank.

[2] The Y edit code suppresses the farthest left zero of a date field that is three to six digits long, and it suppresses the two farthest left zeros of a field that is seven positions long. For more information, see the second footnote in Table 101 on page 693.

[3] The Z edit code removes the sign (plus or minus) and suppresses leading zeros.

User-Defined Edit Codes

You can define five edit codes to provide more editing function than is available with the i5/OS edit codes, and to handle common editing functions that would otherwise require the use of an edit word.

You can define your own edit codes using the Create Edit Description (CRTEDTD) command.

Table 103. IBM-Supplied Edit Descriptions

Description	QEDIT5 Edit Codes[1]	QEDIT6 Edit Codes[1]	QEDIT7 Edit Codes[1]	QEDIT8 Edit Codes[1]	QEDIT9 Edit Codes[1]
Integer mask	xxx,xxx,xxx,xxx,xx0	xxx,xxx,xxx,xxx,xx0	xxx,xxx,xxx,xxx,x0x	xxx,xxx,xxx,xxx,x0x	0xx-xx-xx
Decimal point	. (period)	. (period)	. (period)	. (period)	. (period)
Fraction mask	xxxxxxxxx	xxxxxxxxx	xxxxxxxxx	xxxxxxxxx	None
Fill character	x	x	x	x	x
Floating currency symbol	None	None	None	$	None

Table 103. IBM-Supplied Edit Descriptions (continued)

Description	QEDIT5 Edit Codes[1]	QEDIT6 Edit Codes[1]	QEDIT7 Edit Codes[1]	QEDIT8 Edit Codes[1]	QEDIT9 Edit Codes[1]
Zero balance	Replace with fill character	Replace with fill character	Normal editing rules	Normal editing rules	Normal editing rules
Negative status	CR	- (minus sign)	- (minus sign)	- (minus sign)	None
Positive status	DR	None	None	None	None
Left constant	None	None	$	None	None
Right constant	None	x*	None	None	None

:

[1] The x represents a blank.

An edit description contains the following items:

Integer mask Describes the editing of the integer portion of a field. All characters except blank, zero, and ampersand (&); are treated as constants:

- Blank means to replace the blank with a digit if zero suppression has ended; otherwise, replace the blank with the fill character.
- The zero to the farthest left means to replace the zero with a digit and end zero suppression. All other zeros are treated as constants.
- Ampersand means to replace the & with a blank.

Decimal point Defines what character is used as the decimal point. By default, a period (.) is used.

Fraction mask Describes the editing of the fraction portion of a field. Ampersand is the same as for the integer mask. All zeros are treated as constants and all blanks are replaced with digits (no fill character is used).

Fill character Defines what character is used in each position of a result that is zero-suppressed. By default, a blank is used.

Floating currency symbol
Defines the floating currency symbol to be used to edit the field.

Zero balance Specifies how zero values are to be edited. They can be edited using the fill character or the integer and fraction masks.

Negative status
Defines the characters that are to follow the edited result of a field if the field is negative.

Positive status
Defines the characters that are to follow the edited result of a field if the field is positive or zero.

Left constant Defines a constant that is to be the portion to the farthest left of the edited result of a field.

Right constant
Defines a constant that is to be the portion to the farthest right of the edited result of a field.

Using User-Defined Edit Codes

The following rules apply to using edit descriptions. These rules are affected by the length and decimal positions of the field being edited.

- The field to be edited is aligned by the integer and fraction masks.
- The entire integer mask is not always used. The integer mask is truncated on the left side immediately before the farthest left significant digit or farthest left, or forced, zero.
- The decimal point immediately follows the integer mask and the fraction mask immediately follows the decimal point. If no decimal point is used, the fraction mask immediately follows the integer mask.

- The entire fraction mask is not always used. The fraction mask is truncated on the right side immediately following the farthest right significant digit.
- The width of the edited result is equal to the total of the following:
  - Length of left constant
  - Length of floating currency symbol
  - Length of truncated integer mask
  - Length of decimal point (which is always 1 unless no decimal point is used)
  - Length of truncated fraction mask
  - Length of negative or positive status value (whichever is longer)

    Note: If the text you specify for negative status is a different length than the one for positive status, the CRTEDTD command pads the shorter value with blanks. This makes the shorter value equal in length to the longer value.
  - Length of right constant
- If either the integer mask or fraction mask does not contain enough digit replace characters for the field to be edited, the field is not edited and is ignored.

Example of a User-Defined Edit Code

The following Create Edit Description (CRTEDTD) command shows how to create an edit description to edit a numeric field and indicate if the value is a credit or debit (x indicates a blank):

```
CRTEDTD  EDTD(5) INTMASK('xxx,xxx,xx')
         FRACMASK('xxxx') NEGSTS('DEBITx')
         POSSTS('CREDIT')  LFTCNS('$')
         RGTCNS('xB**')
```

The field that uses the previous edit description contains the value 001234 and has two decimal positions. The edited field looks like this:

```
$xx12.34CREDITx**
```

Note that when you create an edit description, you specify only the number (5, 6, 7, 8, or 9) that is associated with the edit code. The system automatically affixes QEDIT to the number you specify. (In the preceding example, EDTD(5) was specified. This would then be QEDIT5.)

Appendix F. System/36-Compatible Display Data Management

This appendix describes how to use i5/OS display data to provide System/36-compatible functions. Additional information on System/36 compatibility can be found in the **DDS** topic in the iSeries Information Center.

In order to migrate System/36 applications that use display devices, i5/OS display data management has functions that allow the operating system to work like System/36 work station data management. The level of compatibility with the System/36 depends on the following:

- The User Display Management (USRDSPMGT) keyword

  This keyword is automatically put in the DDS source when the $SFGR utility or the Create System/36 Display File (CRTS36DSPF) CL command converts SFGR source to DDS source. Specifying this keyword indicates that the System/36 work station data management function is to be used instead of display data management functions.

  For example, the USRDSPMGT keyword specifies that the cursor is positioned as it is on the System/36. If the USRDSPMGT keyword is not specified, the cursor is positioned according to the display data management rules.
- The programming language used to write the program

  To get application-level System/36 compatibility, the programs must be written in System/36-compatible RPG II or System/36-compatible COBOL.

  For example, if a program is written in a System/36-compatible language, the file status codes set by the high-level language and return codes set by data management are the same as the System/36. If a program is not written in a System/36-compatible language, the file status codes and return codes are the i5/OS values.
- The environment where the application runs

  To get environment-level System/36 compatibility, the programs must run in the System/36 environment.

  For example, if an application running in the System/36 environment uses the MSGID DDS keyword to display the message from a user message file, the message file used is the message file specified on the USER1 parameter of the MEMBER OCL statement. If this same application is not run in the System/36 environment, the message file used is a message file named USR1 in the job's library list.

The remainder of this appendix contains a section for each display data management topic you may need to know about if you are interested in S/36 compatibility. Where appropriate, each topic contains the following information:

- How the System/36 supports the function. More information on the System/36 functions can be found in the following manuals:
  - *System/36 Creating Displays: Screen Design Aid and System Support Program*
  - *System/36 Concepts and Programmer's Guide*
- How the default display data management supports the function.
- How to get the System/36-compatible function with i5/OS display data management, including which of the three level-of-compatibility items previously listed is required to get the compatible function.

Clearing Lines on the Display

When a System/36 application writes the first record in a job, System/36 work station data management clears the display before displaying the application data. The System/36 function is provided on the i5/OS system for the first write operation in a System/36-environment job if the program is written in a System/36-compatible language or the display file contains the USRDSPMGT keyword. The *System/36 Environment Programming* book describes starting and ending a System/36-environment job.

i5/OS display data management normally clears the display when a display file is opened; System/36 work station data management does not. The System/36 function is provided by the operating system for programs written in a System/36-compatible language or that open a display file that specifies the USRDSPMGT keyword.

Applications can clear all or just parts of the display by using the clear line (CLRL) keyword. The CLRL keyword is automatically put in the DDS source when the System/36 environment converts the SFGR source to DDS. The value generated for the number of lines to clear is based on the value in columns 19 and 20 of the SFGR S specification.

Because some of the i5/OS and System/36-compatible display-clearing functions are different, the following DDS keywords cannot be used in display files that specify the USRDSPMGT keyword:

- ERASE
- PUTRETAIN
- KEEP
- ASSUME

Also, the OVERLAY keyword is ignored for programs written in a System/36-compatible language and for display files that specify the USRDSPMGT keyword.

Input Data for Display File Records

On the System/36, only one record with input fields can be displayed. For example, if an application on the System/36 writes record REC1 and then record REC2, only the last record written (REC2) has input-capable fields. When REC1 is written, the input fields defined by REC1 are input-capable. When REC2 is written, REC1 fields are then no longer input-capable, and the input fields defined by REC2 are input-capable.

With the default i5/OS display data management support, many different records with input fields can be displayed at the same time. For example, record REC1 (which occupies lines 1 through 12) and record REC2 (which occupies lines 21 through 24) can be displayed at the same time. The application program can read data from either record. Only the data for the input-capable fields defined by the specific record is returned. In the previous example, if the application program needs to get all the data from all the input-capable fields on the display, the application has to issue two read operations: one for REC1 and one for REC2.

i5/OS display data management allows only one record with input fields for programs written in a System/36-compatible language or for display files that specify the USRDSPMGT keyword.

Input Data from the Work Station Controller

The System/36 work station controller returns the data for all input-capable fields. With the default i5/OS display data management support, the work station controller returns only data from the display for the input-capable fields that the display station operator has changed.

i5/OS display data management provides the System/36 function with the Modified Data Tag (MDT) value of the DSPATR DDS keyword. The DSPATR(MDT) keyword is automatically put in the DDS source for every input-capable field when the System/36 environment converts SFGR source to DDS. This keyword sets on an indicator in the work station controller that makes the work station controller act as if the field with DSPATR(MDT) was changed by the display station operator. Because this keyword is specified for all input-capable fields, all the input data is returned to the system and to the program when the program issues a read operation.

Self-Check

On System/36, if a field specifies modulus 10 or modulus 11 self-checking (column 30 of SFGR D specification), the work station controller verifies that valid data is entered into the field. If a display station operator enters an incorrect modulus 10 or modulus 11 number for a field, the work station controller displays a blinking four-digit error code, and the user must enter data into the field again.

i5/OS display data management support for self-check, which uses the CHECK(M10) and CHECK(M11) DDS keywords, verifies that the field has a valid modulus 10 or modulus 11 number when you press the Enter key or another function key. This verification is used in place of issuing an error message as you type data into the field. If the CHECK(M10F) or CHECK(M11F) keyword is specified, the workstation controller verifies that valid data is entered into the field as described in the previous paragraph.

If a display file contains the CHECK(M10) or CHECK(M11) DDS keyword and also the USRDSPMGT keyword, display data management, like the System/36, issues an error message as you type the data. The DDS keywords CHECK(M10F) and CHECK(M11F) are not allowed with the USRDSPMGT keyword.

Return Input

System/36 work station data management has support to indicate to an application program whether any fields on the display have been changed by the display station operator. To use this support, the SFGR source must specify N for return input (column 22) in the SFGR S specification. If N is specified and the user does not change any fields on the display, all blanks are received as input data for a read operation.

i5/OS display data management provides the System/36 support. If the CHANGE keyword is specified at the record level without a response indicator in a display file that specifies the USRDSPMGT keyword, and the program doing the read operation is written in a System/36-compatible language, the program receives a blank record when the display station operator does not change any data on the display. The CHANGE keyword is automatically put in the DDS source when SFGR source is converted to DDS by the System/36 environment if N is specified for return input.

Erase Input Fields

On System/36, if a program writes a record to the display and the erase-input-fields function (columns 31 and 32 of the SFGR S specification) is enabled for the record, the input fields on the display are erased, but the specified record is not sent to the display. For example, if a program writes record REC1 to the display and then writes record REC2 to the display (where REC2 has the erase-input-fields function enabled), all the input fields defined by REC1 are erased and REC2 is not sent to the display.

i5/OS display data management follows the System/36 handling of erase-input-fields when the program is written in a System/36-compatible language, the USRDSPMGT keyword is specified, and the ERASEINP keyword is enabled for the record.

On System/36, if a program writes a record to the display and the erase-input-fields and put-override functions are both enabled for the record, the input fields on the display are erased, and the put-override data for the specified record is sent to the display.

i5/OS display data management follows the System/36 handling of erase-input-fields when the program is written in a System/36-compatible language, the USRDSPMGT keyword is specified, and the ERASEINP and PUTOVR keywords are enabled for the record.

Display Attributes

Some fields do not have beginning display attributes on the System/36. A field has a beginning display attribute only if one or more of the following is true:

- The field is input-capable.
- The output data is based on an indicator (column 23-24 of SFGR D specification).
- The field specifies a display attribute (such as high intensity, blink, reverse image, underline, or column separators).
- The field appears on a row that is not cleared by the record.

To provide application compatibility with the System/36, i5/OS display data management follows the System/36 rules to determine if a field should have a beginning attribute when the USRDSPMGT keyword is specified in a display file. Also, if a field does not have a beginning attribute and the OVRATR keyword is specified on the field, the OVRATR keyword is ignored (no field attribute is sent).

On the System/36, some fields do not have ending display attributes. A field has an ending attribute only if one or more of the following is true:

- The field is input-capable.
- The output data is based on an indicator (columns 23-24 of SFGR D specification).
- The field specifies a display attribute (such as high intensity, blink, reverse image, underline, or column separators).

i5/OS display data management follows the System/36 rules to determine if a field should have an ending attribute when the USRDSPMGT keyword is specified in a display file.

Positioning the Cursor

System/36 uses the following rules to determine where the cursor should be positioned when a record is displayed:

1. The cursor is positioned to the first input field (defined by the SFGR source statements) that has an indicator specified for the position-cursor attribute, has this indicator set on, and where one of the following conditions is true:
   - This field does not have the protect attribute specified (columns 37 and 38 of the SFGR D specification).
   - This field has an indicator specified for the protect attribute, but the indicator is off.
   - This field has an unoptioned-protect attribute specified (columns 37 or 38 of the SFGR D specification is Y or N).
2. If the cursor is not positioned by rule 1, a check is made for fields with an unoptioned-position-cursor attribute (columns 32 or 33 of the SFGR D specification is Y). If there is a field with an unoptioned-position-cursor attribute, the cursor is positioned to this field.
3. If the cursor is not positioned by rule 1 or 2, the cursor is positioned to the first input field (as defined by the SFGR source statements) that does not have an unoptioned-protect attribute (columns 37 or 38 of the SFGR D specification are blank).
4. If the cursor is not positioned by any of the previous rules, the cursor is positioned to the upper-left-hand corner of the display.

i5/OS display data management positions the cursor following the System/36 rules for display files that have the USRDSPMGT keyword specified.

Displaying Messages

On System/36, fields can be defined that have message text automatically inserted into the field when a record is displayed. System/36 work station data management retrieves the message text defined for the field and supplies the message text as output data for the field. To display message text, System/36 applications must specify the following:

- M for constant type (column 56 of SFGR D specification).
- The message identification code (MIC) (columns 57 through 60 of SFGR D specification or output data supplied by the program).
- The message member identifier (columns 61 and 62 of SFGR D specification or output data supplied by the program).

i5/OS applications can display messages in a variety of different ways. For information on displaying messages, see “Creating and Displaying Your Own Messages” on page 223.

To provide application compatibility with System/36, the MSGID keyword can be used to display messages. The MSGID keyword is automatically generated by the System/36 environment if M is specified for constant type in the SFGR D specification.

Either of the following formats can be used to display messages that follow the System/36 conventions for sending messages:

```
MSGID(message-identifier message-file)
MSGID(USR message-identification-code message-file)
```

Message-identifier consists of two parts: a message prefix and a message-identification-code (MIC). If an application uses the first format of the MSGID keyword, the three-character message prefix should be the first three characters of the seven-character message-identifier. If an application uses the second format of the MSGID keyword, the prefix does not need to be provided. The prefix is already specified as USR in the MSGID keyword.

The message-identification-code parameter specifies the four-character message ID of the message to be displayed.

For more information on specifying display file message-identifiers, see the **DDS** topic in the iSeries Information Center.

The message-file parameter identifies the message file that contains the message to be displayed. For System/36 compatibility, this parameter can be specified in one of two formats.

The first format for specifying the message-file parameter is in a two-character field-name. The field-name must exist in the same record as the MSGID field, and the field must be defined as a character field with usage H (hidden), P (program-to-system), B (both), or O (output-only). The field identified by field-name indicates the message file that the System/36 environment uses to display a message. The values allowed for the field are:

Table 104. Message Files for MSGID

Value of Field-Name	Message File Used
U1	Message text from the message file specified on USER1 parameter on MEMBER OCL statement.
U2	Message help text from the message file specified on USER2 parameter on MEMBER OCL statement.
P1	Message text from the message file specified on PROGRAM1 parameter on MEMBER OCL statement.

Table 104. Message Files for MSGID (continued)

Value of Field-Name	Message File Used
P2	Message help text from the message file specified on PROGRAM2 parameter on MEMBER OCL statement.
M1	Message text from IBM-supplied message file ##MSG1.
M2	Message help text from IBM-supplied message file ##MSG1.

The second format for specifying the message-file parameter is a special value. The following special values can be specified for the message file:

Table 105. Message Files for MSGID

Special Value	Message File Used
*USR1	Message text from message file specified on USER1 parameter on MEMBER OCL statement.
*USR2	Message help text from message file specified on USER2 parameter on MEMBER OCL statement.
*PGM1	Message text from message file specified on PROGRAM1 parameter on MEMBER OCL statement.
*PGM2	Message help text from message file specified on PROGRAM2 parameter on MEMBER OCL statement.
*SYS1	Message text from IBM-supplied message file ##MSG1.
*SYS2	Message help text from IBM-supplied message file ##MSG1.

The following are considerations you should be aware of when using the System/36-compatible functions of the MSGID keyword:

- If a field name or a special value is specified, the processing done by i5/OS display data management depends on the environment where the application runs. For example, if the application is run in the System/36 environment, *USR1 indicates to use the message file identified by the USER1 parameter on the MEMBER OCL statement. If the same application is not run in the System/36 environment, *USR1 indicates to use message file USR1.
- If the message is to be displayed and it only has message help, the message help is displayed.
- If the message help is to be displayed but the message has no message help, only the message is displayed.
- If the message text to be displayed is longer than the length of the output field, the message text is truncated to the length of the output field. If the message text to be displayed is shorter than the length of the output field, the message text is padded with blanks.
- If the message identifier contains any characters that are not valid (valid characters are 0-9 and A-F), the message text displayed is the message identifier followed by the two-character message file identifier.
- If the message identifier or the message file is not found, the message text displayed is the message identification code followed by two question marks (??).
- The following DDS keywords cannot be specified on a field with the MSGID keyword:

  DFT

  DFTVAL

  FLTFIXDEC

  FLTPCN

  MSGCON

Put Override

On System/36, if the put-override option is enabled (columns 33 and 34 of SFGR S specification) and the number of lines to clear (columns 19 and 20 of SFGR S specification) is specified, the number of lines to clear is ignored. System/36 work station data management does not clear the display when the put-override option is enabled. i5/OS display data management also has support to ignore the clear function (CLRL keyword) for records that have the put-override option (PUTOVR keyword) enabled. This support is used for programs written in a System/36-compatible language or using a display file that specifies the USRDSPMGT keyword.

On System/36, if the put-override option is enabled, no input-field definitions are sent to the display. Input-field definitions contain information such as length of the input-field, mandatory-fill attribute, mandatory-enter attribute, modified-data-tag attribute, and protected input-field attribute. If there are input fields currently on the display, these fields remain input-capable. The default i5/OS display data management support sends input-field definitions when the put override option is enabled. i5/OS display data management provides the System/36 function of not sending input-field definitions to the display when the put-override function is enabled and the record to be displayed is in a display file that specifies the USRDSPMGT keyword. If the application uses the System/36 function of not sending input-field definitions to the display when the put-override function is enabled, attributes that are defined by the input-field definitions cannot be changed when put-override is enabled. For example, the protected input-field attribute (DSPATR(PR)) is ignored when put-override is enabled, because the input-field definitions are not sent to the display.

On System/36, the put-override option can be specified for records that are currently displayed and for records that are not currently on the display. For example, if record REC1 is currently displayed and the application program issues a write operation with the put-override function enabled for record REC2, the request to display record REC2 is processed as a put-override request. The default support in i5/OS display data management supports the put-override function only for records that are currently displayed. For example, if record REC1 is currently displayed and the application program issues a write operation with the put-override function enabled for record REC2, the put-override function is ignored and the request to display record REC2 is processed as if the PUTOVR keyword was not enabled. If the display file specifies the USRDSPMGT keyword, i5/OS display data management provides the System/36 function of allowing the put-override function even if the record is not currently displayed.

During an override operation, i5/OS display data management may ignore help specifications contained in the record format that is displayed by the application program. This occurs if the USRDSPMGT keyword is used. If the Help Cleared (HLPCLR) keyword is also specified on the record format, no online help is available.

Handling Signed Numeric Data

System/36 does no checking when incorrect data (data other than 0 through 9) is specified for a signed numeric field on a write operation. For example, if the data in a signed numeric field is character data, the character data is displayed. The default support in i5/OS display data management is to replace any incorrect signed numeric data with nulls when the field is displayed. i5/OS display data management supports the System/36 function when the USRDSPMGT keyword is specified in the display file.

System/36 also does no checking when a read operation returns incorrect data for a signed numeric field. For example, if the data returned for a signed numeric field is character data, the character data is passed to the application program. The default support in i5/OS display data management replaces any incorrect signed numeric data with zeros before the data is given to the application program. i5/OS display data management supports the System/36 function for programs written in a System/36-compatible language.

On System/36, if zeros are entered in a signed numeric field and the field-minus key is used to exit the field, a value of negative zero is returned to the application program. The default support in i5/OS display data management returns zeros to the application instead of negative zero. i5/OS display data management supports the System/36 function for programs written in a System/36-compatible language or using a display file that specifies the USRDSPMGT keyword.

Function Keys

On System/36, application programs can receive control when a function key is pressed. To use this support, the application must enable the desired function key in the SFGR source and the program must indicate that it can process the function key.

The default i5/OS display data management support does not allow a program not written in a System/36 compatible language to indicate which function keys the program can process. The function keys that are enabled in the DDS for the display file are the function keys that i5/OS display data management passes to the program to handle.

i5/OS display data management supports the System/36 function of a program indicating which function keys it can process for programs written in a System/36-compatible language. See the appropriate System/36-compatible language manual for information on enabling function-key handling in a program.

Help Key Considerations

The HELP and HLPRTN DDS keywords are used to indicate what processing should be done by the system when the Help key is pressed. The HELP keyword indicates that the application wants the Help key enabled. The HELP keyword is automatically put in the DDS source when SFGR source is converted to DDS by the System/36 environment.

The HLPRTN keyword indicates that the application program wants to process the Help key when the display station operator presses the Help key. When the Help key is enabled in the SFGR key mask (columns 64 through 79 of the SFGR S specification) the HLPRTN keyword is put in the DDS source when the System/36 environment converts SFGR source to DDS.

If the HELP and HLPRTN keywords are not specified and the display station operator presses the Help key, a message is displayed by i5/OS display data management indicating that the Help key is not allowed.

If the USRDSPMGT and HELP keywords are specified, the HLPRTN keyword is not specified, and the display station operator presses the Help key, i5/OS display data management uses the following rules to determine how the Help key is be processed:

1. If the Help key is pressed when a message is being displayed, the message help for the message is displayed.
2. If the Help key is not processed by rule 1 and there is application help defined for the record, the application help for the record is displayed.
3. If the Help key is not processed by rules 1 or 2, an error message is displayed indicating that the key is not allowed.

If the USRDSPMGT, HELP, and HLPRTN keywords are specified and the display station operator presses the Help key, i5/OS display data management uses the following rules to determine how the Help key is be processed:

1. If the application program has indicated that it can process the Help key, an indication is returned to the application program that the Help key was pressed.
2. If the Help key is not processed by rule 1 and the Help key is pressed when a message is being displayed, the message help for the message is displayed.
3. If the Help key is not processed by rules 1 or 2 and there is application help defined for the record, the application help for the record is displayed.

4. If the Help key is not processed by rules 1, 2 or 3, an error message is displayed indicating that the key is not allowed.

Using Command Keys to Exit Application Help

On System/36, if application help is displayed and the display station operator presses a command key to exit application help, an indication of what command key was pressed is given to the application program. To use this support, the application must enable the desired command key on the display where the Help key is pressed, and the key must be enabled on the application help display. If the display file containing application help specifies the USRDSPMGT keyword, i5/OS display data management returns the indication of what command key was pressed to exit application help. If the command key was not specified on both the application record and the help record, the message, `Function Key not Allowed` is displayed.

On System/36, if application help is displayed and the display station operator presses a command key to exit application help, the data from the display where the Help key is pressed is returned to the application program. To use this support, the application must indicate *restore yes* (columns 47 and 48 of the SFGR H specification) on the display where the Help key is pressed. i5/OS display data management returns the data from the display where the Help key is pressed if the display file containing application help specifies the USRDSPMGT keyword.

Cancel-Invite Operation

On System/36, if a program issues a cancel-invite operation and data has already been received by the system (for example, the display station operator pressed the Enter key before the cancel-invite was issued), the data is lost and the application is not notified.

The default i5/OS display data management support causes the cancel-invite operation to fail if the data is available. Message CPF4737 and return code 0412 are sent to the application program. The application can check the return code and issue a read operation to receive the data that was returned from the display station.

i5/OS display data management supports the System/36 function and does not send message CPF4737 or set the return code for programs written in a System/36-compatible language or for display files that have the USRDSPMGT keyword specified.

For an application running in an i5/OS display data management environment that contains the USRDSPMGT keyword and the INVITE keyword in the display file, the following will occur:

- If a file is active with a read outstanding when a record is written from another display file, a save of the currently displayed screen takes place before the write of the new record.
- The suspended file is restored when a write is done to the suspended file.

  On a System/36, the save is not done for this situation.

To circumvent the save command in the i5/OS display data management environment, do the following:

1. Define a record named RECORDD in the file FILE1 as shown in Figure 147 on page 708.
2. Write record RECORDD to program PGM1 before writing record RECORDA

By writing record RECORDD, the read outstanding for record RECORDB is canceled, and the save/restore of the record RECORDB is not required because there is no longer an outstanding invite.

```
   PGM1 (RPG PROGRAM)
                MOVE  *ON     *IN24
                WRITERECORDB
                WRITERECORDD
                WRITERECORDA
                EXFMTRECORDC
                ENDSR

   *********************************

   FILE1
                     USRDSPMGT
   R RECORDB         OVERLAY
                     CF(01)
                     INVITE
                 1 18'TESTB'
   R RECORDC         OVERLAY
                 1 18'TESTC'
   R RECORDD         OVERLAY

   *********************************

   FILE2
                     USRDSPMGT
   R RECORDA         OVERLAY
                22  5'TEST________A'
24                   DSPATR(ND)
```

Figure 147. Circumventing the Save Command

Retain Command and Function Keys

On System/36, when a program issues a write operation to display a record, the application can indicate that the command and function keys currently active for the display station should be the command and function keys that are used for the new record. For example, if an application writes record REC1 (where REC1 enables command keys 1 through 12 and all of the function keys), and then the application writes record REC2 (where REC2 keeps the command and function keys from REC1), then command keys 1 through 12 and all of the function keys are enabled when REC2 is displayed. System/36 applications indicate that the command keys should be kept by specifying an R for the enable command keys option in the SFGR source (column 28 of the SFGR S specification). System/36 applications indicate that the function keys should be kept by specifying an R for the enable function keys option in the SFGR source (column 27 of the SFGR S specification).

i5/OS application programs can specify either a list of command and function keys that are valid when a record is displayed or an indication that the command and function keys that are currently active should remain active when a record is displayed.

i5/OS display data management allows an application to keep the current command keys for a record if the RETCMDKEY keyword is specified for the record. The RETCMDKEY keyword is automatically generated by the System/36 environment if R is specified for command keys in the SFGR S specification.

Also, i5/OS display data management allows an application to keep the current function keys for a record if the RETKEY keyword is specified for the record. The RETKEY keyword is automatically generated by the System/36 environment if R is specified for function keys in the SFGR S specification.

System/36 Functions Not Supported

This section describes the System/36 work station data management functions that are not supported by i5/OS display data management.

If one program issues a write operation that invites the display and another program issues a write operation with the put-override function enabled for the same display, all the write and read operations must use record formats from the same display file. For example, if program A sets up a read-under-format (RUF) request by writing record REC1 from file FILE1, the second program must also use FILE1 for any write operations with the put-override function enabled. If a different display file is used, all the input fields from REC1 are changed to output-only fields when the put-override function is handled by i5/OS display data management.

The put-override function can be used only across different jobs (for example, a single-requester-terminal (SRT) program that gives control to a multiple-requester-terminal (MRT) program) when the application programs are using RUF. If RUF is not in progress and the second job issues a write operation with the put-override function enabled, all the input fields on the display are changed to output-only fields when the put-override function is handled by i5/OS display data management.

On System/36, application help can return the data from input fields on an application help display. This is not supported by i5/OS display data management.

Restricted DDS Keywords/Functions

If the USRDSPMGT keyword is specified in the DDS source, the following DDS functions cannot be used:

- DDS keywords that control clearing the display:
  - ASSUME
  - ERASE
  - KEEP
  - PUTRETAIN
- ERRSFL DDS keyword
- HLPCMDKEY DDS keyword
- Subfile DDS keywords:
  - SFL
  - SFLCTL

  Because System/36-compatible languages do not support subfiles, subfiles cannot be used in a display file if USRDSPMGT is specified.
- Response indicators

  Because System/36-compatible languages do not return indicators from the read operation to a display file, response indicators should not be specified if USRDSPMGT is used.
- IGC conversion

  Because only one input-capable format is maintained on the screen at on time, IGC conversion is not supported on the System/36 or on i5/OS in the System/36 environment. The IGCCNV keyword is not allowed anywhere in the file definition when you use the USRDSPMGT keyword.
- DDS keywords:
  - CHECK(M10F)
  - CHECK(M11F)

  Because specifying the DDS keywords CHECK(M10) and CHECK(M11) with USRDSPMGT provide the same function as CHECK(M10F) and CHECK(M11F) without USRDSPMGT.

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 grant you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
U.S.A.

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 106-0032, 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 OF 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.

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

IBM Corporation

Software Interoperability Coordinator, Department YBWA
3605 Highway 52 N
Rochester, MN 55901
U.S.A.

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 information and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement, IBM License Agreement for Machine Code, 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.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs.

Each copy or any portion of these sample programs or any derivative work, must include a copyright notice as follows:

© (your company name) (year). Portions of this code are derived from IBM Corp. Sample Programs. © Copyright IBM Corp. _enter the year or years_. All rights reserved.

If you are viewing this information softcopy, the photographs and color illustrations may not appear.

Programming Interface Information

This Application Display Programming publication documents intended Programming Interfaces that allow the customer to write programs to obtain the services of IBM i5/OS.

Trademarks

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

AFP
AIX
Common User Access
CUA
IBM

InfoWindow
iSeries
OS/2
PS/2
SAA
System/36
System/38
Systems Application Architecture

Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

Other company, product, or service names may be trademarks or service marks of others.

Terms and conditions

Permissions for the use of these publications is granted subject to the following terms and conditions.

Personal Use: You may reproduce these publications for your personal, noncommercial use provided that all proprietary notices are preserved. You may not distribute, display or make derivative works of these publications, or any portion thereof, without the express consent of IBM.

Commercial Use: You may reproduce, distribute and display these publications solely within your enterprise provided that all proprietary notices are preserved. You may not make derivative works of these publications, or reproduce, distribute or display these publications or any portion thereof outside your enterprise, without the express consent of IBM.

Except as expressly granted in this permission, no other permissions, licenses or rights are granted, either express or implied, to the publications or any information, data, software or other intellectual property contained therein.

IBM reserves the right to withdraw the permissions granted herein whenever, in its discretion, the use of the publications is detrimental to its interest or, as determined by IBM, the above instructions are not being properly followed.

You may not download, export or re-export this information except in full compliance with all applicable laws and regulations, including all United States export laws and regulations.

IBM MAKES NO GUARANTEE ABOUT THE CONTENT OF THESE PUBLICATIONS. THE PUBLICATIONS ARE PROVIDED ″AS-IS″ AND WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, NON-INFRINGEMENT, AND FITNESS FOR A PARTICULAR PURPOSE.

Bibliography

The following list of manuals and iSeries Information Center topics are related to this book. The manuals are listed with their full title, base order number, and a description of the content. The description of each manual includes the relationship of that manual to this book.

System Use

- The **Basic system operations** topic in the iSeries Information Center: Use this information to explore some of the introductory concepts related to your server and the i5/OS operating system. You can also use this guide to perform basic system operation tasks, such as starting and stopping your server and working with users, jobs, and devices.

Systems Management

- The **Systems Management** collection in the iSeries Information Center: This topic provides information about creating and changing the work management environment, working with system values, and collecting and using performance data to improve system performance.

Application Development

- *ADTS/400: Character Generator Utility*, SC09-1769-00

  This guide provides information about using the Application Development Tools character generator utility (CGU) to create and maintain a double-byte character set (DBCS) on the i5/OS system.

  Use this guide to add user-defined characters to DBCS font tables.
- *ADTS/400: Programming Development Manager*, SC09-1771-00

  This guide and reference provides information about using the Application Development Tools programming development manager (PDM) to work with lists of libraries, objects, members, and user-defined options to easily do such operations as copy, delete, and rename.

  Use this guide and reference to learn how to enter DDS and UIM source.
- *ADTS for AS/400: Screen Design Aid*, SC09-2604-00

  This guide and reference provides information about using the Application Development Tools screen design aid (SDA) to design, create, and maintain displays, menus, and online help information.

  Use this guide and reference if you want to let the system create the DDS as you design your own displays.
- *ADTS for AS/400: Source Entry Utility*, SC09-2605-00

  This guide provides the application programmer or programmer with information about using the Application Development Tools source entry utility (SEU) to create and edit source members. This manual explains how to start and end an SEU session and how to use the many features of this full-screen text editor.

  Use this guide to understand the functions of SEU that are available for creating and editing your DDS and CL and other code.
- *Common User Access Basic Interface Design Guide* , SC26-4583.

  This guide assists you in designing a user interface that is consistent within your application and across other applications. This guide presents the user interface style guidelines and implementation considerations that you, the designer and/or developer, must be concerned with.

Communications and Connectivity

- *ICF Programming*, SC41-5442-00

  This guide provides the information needed to write application programs that use i5/OS communications and the i5/OS intersystem communications function (i5/OS-ICF). This manual also contains information on data description specifications (DDS) keywords, system-supplied formats, return codes, file transfer support, and program examples.

  Use this guide to work with applications on remote systems.
- *IBM 5250 Information Display System Functions Reference Manual* , SA21-9247.

This reference manual provides information about using the functions of the 5250 Information Display.

Use this manual to use the 5250 Display Data Stream.

- *IBM 5494 Remote Control Unit Functions Reference Manual* SC30-3533.

  This reference manual describes how the IBM 5494 Remote Control Unit uses Systems Network Architecture (SNA), Synchronous Data Link Control (SDLC), X.25, X.21, or Token-Ring protocols to communicate with i5/OS. In addition, this book describes how the 5494 manages the attached work stations and converts network data streams into protocols for 5250 work stations and printers.

Program Enablers

- The **DDS** topic in the iSeries Information Center: This topic provides detailed descriptions of the entries and keywords needed to describe database files (both logical and physical) and certain device files (for displays, printers, and ICF) external to the user's programs.

  Use this topic to get detailed information about specific DDS keywords used in display files.
- The **CL** topic in the iSeries Information Center: This topic provides information for system programmers and system administrators who write programs using i5/OS commands and other IBM-supplied commands.

  This topic provides a description of the i5/OS control language (CL) and its commands. (Non-i5/OS commands are described in the respective licensed program publications.) It also provides an overview of *all* the CL commands for the system, and it describes the syntax rules needed to code them.

  This topic also provides a wide-ranging discussion of i5/OS programming topics, including control language (CL) programming, controlling flow and communicating between programs, working with objects in CL programs, and creating CL programs. Other topics include predefined and impromptu messages and message handling, defining and creating user-defined commands and menus, and application testing, including debug mode, breakpoints, traces, and display functions.

  Use this guide to understand how to use CL for creating application programs.
- The **Files and File Systems** collection in the iSeries Information Center: This topic provides information about using files in application programs. It includes information on fundamental structure and concepts of data management support on the system. It also includes information on overrides and file redirection (temporarily making changes to files when an application program is run), copying files by using system commands to copy data from one place to another, and tailoring a system using double-byte data. It also provides information on how to control and understand printing, including printer connectivity, PC attached printers, and advanced function printing (AFP™) printers. Finally, it provides information about data management support for tapes.

  Use this topic for general information about using files, including printer device files and tape files, in application programs.
- *System/36 Environment Programming*, SC41-4730-00.

  This guide provides information identifying the differences in the applications process in the System/36 environment on the i5/OS system. It helps the user understand the functional and operational differences (from a System/36 perspective) when processing in the System/36 environment on the i5/OS system. This includes an environment functional overview, considerations for migration, programming, communications, security, and coexistence.

Program Interfaces

- The **APIs** topic in the**Programming** category of information in the iSeries Information Center: This topic provides information and examples for experienced application and system programmers who want to use the i5/OS application programming interfaces (APIs) from high-level language programs to list data or retrieve descriptions and information.

  Use this topic to find out more information about the APIs.

Index

Special characters

*DSPATR (display attribute) parameter 171
*FLD (field) special value 186
*JOBCCSID special value 262
*PGM (program) object type
  defining *PGM menu 244
  menu object defining 244

Numerics

5250 display data stream
  See user-defined data stream (UDDS)
5250 display station
  avoiding record format problems 80
  UNLOCK results 71
5250 engraved key restriction 346

A

accelerator key
  defining 156
  definition 156
  example 157
acquire operation
  definition 42
  file resource allocation for 41
  ILE C/C++ function for 84
  ILE COBOL statement for 84
  ILE RPG operation for 84
acquired display station 41
  additional, for I/O operations 42
  releasing from I/O operations 83
action list definition 344
action list option prompting 657
action variable definition 344, 345
ACTIONS (Menu Bar Cursor Action) dialog command 644
active record format
  determining 44
  table 44
active subfile record
  definition 89
Add Message Description (ADDMSGD) command 243
Add Search Index Entry (ADDSCHIDXE) command 411
adding
  entries during incomplete processing 342
  pop-up window over another panel 360
  pop-up windows using APIs 359
  window over another panel 360
  window to display stack 360
ADDMSGD (Add Message Description) command 243
ADDSCHIDXE (Add Search Index Entry) command 411
alarm
  sounding for messages 228
ALCOBJ (Allocate Object) command 41
Allocate Object (ALCOBJ) command 41
Allow Roll (ALWROL) keyword
  effects on rolling 56
  use 55
alphabetic conversion
  name syntax 469
alternative character sets and code pages 261
alternative keywords, specifying
  Help (HELP) 26
  Page Down (PAGEDOWN) 26
  Page Up (PAGEUP) 26
ALTHELP (Alternative Help) keyword 27
ALTPAGEDWN (Alternative Page Down) keyword 27
ALTPAGEUP (Alternative Page Up) keyword 27
ALWROL (Allow Roll) keyword
  effects on rolling 56
  use 55
ampersand (&) symbol 469
analyzing
  system error messages 229
API (application programming interface)
  adding and removing windows 359
  services 279
APPFMT (application formatted area) tag
  application formatted data 473
  example
    an application formatted area 474
  overview 472
application
  help 185
  interaction with menu bar 180
application control of panel formatting 353
application display
  changing record formats on a 48
  creating a sample 3
  defining in display file 15
  determining active record formats 44
  entry
    See entry display
  erasing unprotected input and output/input fields on 61
  field, defining in a record format 21
  functions, defining in record format 21
  how the system reads input from 76
  improving system performance 267
  information
    See information display
  inviting input to 67
  keeping record or field on 62
  list
    See list displays
  order of record formats written to 49
application display *(continued)*
  overlaying and erasing record formats on 49
  placing records on 45
  printing 37
  reading input from 70
  reading invited input from 68
  rolling data between two lines 55
  single-choice menu
    See menu display
  understanding which records do not occupy space 47
  viewing record format 22
  writing output to 44
application formatted area (APPFMT) tag
  application formatted data 473
  example
    an application formatted area 474
  overview 472
application formatted data
  selection characters 473
application handle definition 335
application help
  See online help information
  caution for multiple display stations 369
  definition 369
  exiting, System/36 application compatibility 707
application pop-up window
  defining 359
application program
  applying overrides when compiling 216
  describing data inside or outside 17
  display files, use with 41
  file resource allocation for on remote system 41
  opening 335
  overriding
    display file attributes in HLL 213
    display file names and attributes in HLL 215
    display file names in HLL 214
  passing display data 255
  record format used by 21
application programming
  confirmation processing 303
  data presentation panel 317
  list panel 298
  menu bar panel 332
application programming interface (API)
  adding and removing windows 359
  services 279
application record
  MNUBARDSP 190
application window
  defining 359
area
  on a panel 467

assignments, common key
  See common key assignments
ASSUME (Assume) keyword
  considerations 255
  example DDS source 255, 256
  OVERLAY keyword, used with 256
  restriction with USRDSPMGT keyword 709
  sharing file between programs 255
  System/36 application compatibility 700
attribute
  continued
    example 465
  display 171
  display stations 43
  entry fields 31
  field
    color 34
    emphasizing 33
    System/36 application compatibility 702
  modified data tag 23
  overriding 57
attribute character 22
attribute interaction table 565
auto-enter
  single-choice selection field 169
auto-selection
  single-choice selection field 169
automatic confirmation processing 303
availability of choices
  controlling 168

B

backward field-exit processing 180
BASETYPE
  summary table 483
  values 479
BASIC statement
  I/O requests for subfiles 98
beginning attribute character
  definition 22
benefits
  programmable mouse button 198
bibliography 715
bidirectional considerations
  formatting 550
binary return codes for program menus 244
binary value
  display form 489, 490
  formatting 489
blank lines for separating 364
border of window 120
BOTINST (bottom instruction) tag
  overview 474
bottom instruction (BOTINST) tag
  overview 474
built-in function
  CHKOBJ (check object) 493, 549
  CHKPGM (COND evaluation program) 494
  CHKPGM (condition evaluation program) 550
built-in function *(continued)*
  CHKUSRCLS (check user class) 494, 550

C

CALL (Call Program) dialog command 644
call program
  commonly asked questions 365
Call Program (CALL) dialog command 644
CANCEL dialog command 646
cancel key
  defining 158
  definition 158
  menu panel 235
cancel read operation, CL command for 84
cancel-invite operation
  CL command for 84
  ILE C/C++ function for 84
  ILE COBOL statement for 84
  ILE RPG operation for 84
  System/36 application compatibility 707
  when to use 77
CAnn (command attention) key
  defining 25
  returning to program from online help information 383
  system performance considerations 269
  validity considerations 80
CAnn (Command Attention) keyword
  for keyboard control 25
CFnn (command function) key
  CHCACCEL keyword 156
  defining 25
  system performance considerations 269
  validity considerations 80
CFNN (command function) key
  returning to program from online help information 383
CFnn (Command Function) keyword 25
CHANGE (Change) keyword 72
Change Command Default (CHGCMDDFT) command 249
Change Device Description (Display) (CHGDEVDSP) command 37
Change Display File (CHGDSPF) command
  Change Display File (CHGDSPF)
    changing file description 16
  changing
    display files 19
    file description 16
  DFRWRT (defer write) parameter
    creating display file 64
    described 267
  ENHDSP (enhance display) parameter 141
  running 19
  SHARE parameter 268
Change Job (CHGJOB) command 228
Change Menu (CHGMNU) command 249
Change View (CHGVIEW) dialog command 647
changing
  display file 19
  file description 16
  file descriptions 19
  window 124
character
  selection 473, 625
  sentence-ending 466
  symmetric
    Arabic 488
    Hebrew 488
character beginning attribute 22
character ending attribute 22
character field attribute 22
Character Identifier (CHRID) keyword
  description 261
  example 261
character set and code page
  considerations when printing 338
  conversion, DDS 261
  conversion, UIM 484
character translation
  using 262
character value
  display form 490
CHECK (Check) keyword
  parameters for cursor movement 28
  RL and RLTB parameters 28
CHECK (validity checking) tag
  example, validity checking 477
  overview 475
check box
  definition 139
  example 139
CHECK(FE) (Field Exit) keyword 154
CHGCMDDFT (Change Command Default) command 249
CHGDEVDSP (Change Device Description (Display)) command 37
CHGDSPF (Change Display File) command
  changing
    display files 19
    file description 16
  DFRWRT (defer write) parameter
    described 64
    when creating display file 64
  DFRWRT parameter 267
  ENHDSP (enhance display) parameter 141
  running 19
  SHARE parameter 268
CHGJOB (Change Job) command 228
CHGMNU (Change Menu) command 249
CHGVIEW (Change View) dialog command 647
CHKOBJ (Check Object) function 493, 549
CHKPGM (COND evaluation program) 494
CHKPGM (condition evaluation program) 550

- CHKUSRCLS (check user class) function 550
- CHKUSRCLS (Check User Class) function 494
- choice entry field, Common User Access 458
- choice-level help 185
- CHRID (Character Identifier) keyword
  - description 261
  - example 261
- CHRID (Character Identifier) parameter
  - specifying 263
  - values, table of 264
- CIT (title citation) tag
  - example, title citations 477
  - overview 477
  - use in help module 402
- CL (control language) program
  - examples
    - program menu 245
    - RTNPNT(*NO) for program menu 247
  - I/O operations using 84
  - inviting input from 68
  - reading-from-invited-devices from 70
- CLASS (class definition) tag
  - BASETYPE summary table 483
  - decision tree for orientation 486
  - display forms
    - character value 490
    - date value 490
    - numeric values 489
    - time value 490
  - example
    - class definitions 488
    - numeric values table 489
  - overview 478
  - tags allowed table 478
- class definition (CLASS) tag
  - BASETYPE summary table 483
  - decision tree for orientation 486
  - display forms
    - character value 490
    - date value 490
    - numeric values 489
    - time value 490
  - example
    - class definitions 488
  - example numeric values table 489
  - overview 478
  - tags allowed table 478
- CLEAR (Clear) keyword 25
- Clear Lines (CLRL) keyword
  - example
    - differences between OVERLAY and 54
    - with (*NO) value 53
  - S/36 environment considerations 700
  - system performance considerations 268
  - when to use 53
- clearing
  - environment considerations 699
  - lines on display
    - S/36 environment considerations 699
    - using the CLRL keyword 53
- close operation
  - CL command for 84
  - definition 83
  - files shared in job 86
  - ILE C/C++ function for 84
  - ILE COBOL statement for 84
  - ILE RPG operation for 84
- CLRL (Clear Lines) keyword
  - example
    - differences between OVERLAY and 54
    - with (*NO) value 53
  - S/36 environment considerations 700
  - system performance considerations 268
  - use 53
- CMD (System Command) dialog command 647
- CMDLINE (Command Line) dialog command 648
- CMDLINE (command line) tag
  - overview 491
- code page
  - alternative display 261
  - conversion, UIM 484
  - page 420 485
  - page 424 485
  - part of CHRID keyword for display files 261
- code page and character set considerations when printing 338
- codes, major/minor return
  - *See* major/minor return code
- colon (:) symbol 469
- COLOR (Color) keyword 34
- color attribute 171
- color, adding to displays 34
- column data
  - formatting 572
- column heading
  - list display 435
- command attention (CAnn) key
  - defining 25
  - returning to program from online help information 383
  - system performance considerations 269
  - validity considerations 80
- Command Attention (CAnn) keyword
  - for keyboard control 25
- command function (CFnn) key
  - defining 25
  - returning to program from online help information 383
  - system performance considerations 269
  - validity considerations 80
- Command Function (CFnn) keyword 25
- command key
  - *See also* command attention (CAnn) key
  - *See* command function (CFnn) key
- command line
  - entered too long 351
  - interpretation 350
  - optional 443
  - pop-up window 361
- command line *(continued)*
  - restrictions 350
  - tools 446
  - window 361
  - window example 361
- Command Line (CMDLINE) dialog command 648
- command line (CMDLINE) tag
  - overview 491
- command prompting 657
- command, CL
  - Add Message Description (ADDMSGD) 243
  - Add Search Index Entry (ADDSCHIDXE) 411
  - ADDMSGD (Add Message Description) 243
  - ADDSCHIDXE (Add Search Index Entry) 411
  - ALCOBJ (Allocate Object) 41
  - Allocate Object (ALCOBJ) 41
  - canceling read operation with 84
  - Change Command Default (CHGCMDDFT) 249
  - Change Device Description (CHGDEVDSP) 37
  - Change Display File (CHGDSPF)
    - changing file description 19
    - running 19
    - SHARE parameter 268
  - Change Job (CHGJOB) 228
  - Change Menu (CHGMNU) 249
  - CHGCMDDFT (Change Command Default) 249
  - CHGDEVDSP (Change Device Description) 37
  - CHGDSPF (Change Display File)
    - changing file description 19
    - DFRWRT parameter 267
    - SHARE parameter 268
  - CHGJOB (Change Job) 228
  - CHGMNU (Change Menu) 249
  - Create Command (CRTCMD) 411
  - Create Control Language Program (CRTCLPGM)
    - program menus 246
  - Create Display File (CRTDSPF)
    - building display with 6
    - creating display files 18
    - DFRWRT parameter 267
    - display file menus 242
    - program menus 245
    - SHARE parameter 268
  - Create Duplicate Object (CRTDUPOBJ)
    - use 249
  - Create Edit Description (CRTEDTD) 35
  - Create Menu (CRTMNU)
    - display file menus 243
    - program menus 246
  - Create Message File (CRTMSGF) 243
  - Create Panel Group (CRTPNLGRP)
    - building display with 7
    - use 410
  - Create Search Index (CRTSCHIDX) 411
  - Create Table (CRTTBL) 264

- command, CL *(continued)*
  - creating your own 411
  - CRTCLPGM (Create Control Language Program)
    - program menus 246
  - CRTCMD (Create Command) 411
  - CRTDSPF (Create Display File)
    - building display with 6
    - creating display files 18
    - DFRWRT (defer write) parameter 267
    - display file menus 242
    - program menus 245
    - SHARE parameter 268
  - CRTDUPOBJ (Create Duplicate Object)
    - use 249
  - CRTEDTD (Create Edit Description) 35
  - CRTMNU (Create Menu)
    - display file menus 243
    - program menus 246
  - CRTMSGF (Create Message File) 243
  - CRTPNLGRP (Create Panel Group)
    - building display with 7
    - use 410
  - CRTSCHIDX (Create Search Index) 411
  - CRTTBL (Create Table) 264
  - Delete Menu (DLTMNU) 249
  - Delete Override (DLTOVR) 217
  - Delete Panel Group (DLTPNLGRP) 410
  - Delete Search Index (DLTSCHIDXE) 412
  - Display Edit Description (DSPEDTD) 35
  - Display File Description (DSPFD) 20
  - Display File Field Description (DSPFFD) 20
  - Display Menu Attributes (DSPMNUA) 249
  - Display Override (DSPOVR) 217
  - Display Program References (DSPPGMREF) 20
  - DLTMNU (Delete Menu) 249
  - DLTOVR (Delete Override) 217
  - DLTPNLGRP (Delete Panel Group) 410
  - DLTSCHIDXE (Delete Search Index Entry) 412
  - DSPEDTD (Display Edit Description) 35
  - DSPFD (Display File Description) 20
  - DSPFFD (Display File Field Description) 20
  - DSPMNUA (Display Menu Attributes) 249
  - DSPOVR (Display Override) 217
  - DSPPGMREF (Display Program References) 20
  - End Receive (ENDRCV) 68
  - ENDRCV (End Receive) 68
  - GO (Go to Menu)
    - generic menu specification 248
    - use 235
  - Go to Menu (GO)
    - generic menu specification 248

- command, CL *(continued)*
  - Go to Menu (GO) *(continued)*
    - use 235
  - help for, assigning 411
  - OVERLAY (Overlay) keyword
    - effect on help-list updating 380
  - Override with Display File (OVRDSPF)
    - compared to CHGDSPF 16
    - file attributes 213
    - file names 214
    - file names and attributes 215
  - overrides
    - commands that restrict 220
    - improving performance for 268
  - OVRDSPF (Override with Display File)
    - compared to CHGDSPF 16
    - file attributes 213
    - file names 214
    - file names and attributes 215
  - Remove Search Index Entry (RMVSCHIDXE) 412
  - RMVSCHIDXE (Remove Search Index Entry) 412
  - Start Programming Development Manager (STRPDM) 3
  - Start Search Index (STRSCHIDX) 404
  - Start Source Entry Utility (STRSEU) 20
  - STRPDM (Start Programming Development Manager) 3
  - STRSCHIDX (Start Search Index) 404
  - STRSEU (Start Source Entry Utility) 20
- commands that are too long 351
- comment 471
- common feedback area
  - definition 43
- common key assignments 446
- Common User Access (CUA)
  - choice entry field 458
  - description 456
  - entry dialog actions, table of 456
  - entry display 459
  - entry fields 459
  - function key area 457
  - information displays 460
  - list display, example 460
  - menu display, single-choice 457
  - message line relationship 457
  - relationship to system 415
  - selection choices 458
  - single selection field operation, guidelines 458
- compact definition list
  - example 527
- concatenation symbol 469
- COND (condition definition) tag
  - CHKOBJ (Check Object) built-in function 493
  - CHKPGM (COND evaluation program) 494
  - CHKUSRCLS (Check User Class) built-in function 494
  - example, conditioning option 495
  - overview 492

- condition definition (COND) tag
  - CHKOBJ (Check Object) built-in function 493
  - CHKPGM (COND evaluation program) 494
  - CHKUSRCLS (Check User Class) built-in function 494
  - example, conditioning option 495
  - overview 492
- condition name, screen size 36
- conditional expression 492, 548
- conditional operator 492
- conditioning
  - option
    - example 495
- confirmation list panel, creating 299
- confirmation panel source 300
- confirmation processing
  - application programming 303
  - automatically 303
- constant field
  - definition 23
  - inserting text from message description 28
  - using 23
- content separator 465
- contextual help
  - defining in UIM 348
  - defining on display 371
  - definition 386
- continued attribute
  - example 465
- continued-entry field
  - backward field-exit processing 180
  - creating 174
  - DBCS considerations 175
  - example 174
  - forward field-exit processing 179
  - keyboard function 176
  - word wrap 175
- control language (CL) program
  - examples
    - program menu 245
    - RTNPNT(*NO) for program menu 247
  - I/O operations using 84
  - inviting input from 68
  - reading-from-invited-devices from 70
- control value 168
- controller, display station 700
- controlling
  - availability of choices 168
  - list entries on list display 343
- convention
  - recommended
    - list action (LISTACT) tag 567
    - pull-down field choice (PDFLDC) tag 608
  - required
    - list action (LISTACT) tag 566
    - pull-down field choice (PDFLDC) tag 608
- conversion
  - alphabetic 469
  - character set and code page, DDS 261

- conversion *(continued)*
  - character set and code page, UIM 484
- COPYR (copyright) tag
  - overview 496
- copyright (COPYR) tag
  - overview 496
- country-designated selection character 345
- CPF4001-40FF message 230
- CPF4101-43FF message 230
- CPF4401-44FF message 230
- CPF4501-46FF message 230
- CPF4701-48FF message 230
- CPF4901-49FF message 230
- CPF5001-50FF message 230
- CPF5101-53FF message 230
- CPF5501-56FF message 230
- Create Command (CRTCMD) command 411
- Create Control Language Program (CRTCLPGM) command 246
- Create Display File (CRTDSPF) command
  - building display with 6
  - creating display files 18
  - DFRWRT (defer write) parameter
    - improving performance with 267
    - write operation 64
  - display file menus 242
  - ENHDSP (enhance display) parameter 141
  - program menus 245
  - SHARE parameter 268
- Create Duplicate Object (CRTDUPOBJ) command
  - use 249
- Create Edit Description (CRTEDTD) command
  - example 697
  - running 35
  - use 697
- Create Menu (CRTMNU) command
  - display file menus 243
  - program menus 246
- Create Message File (CRTMSGF) command 243
- Create Panel Group (CRTPNLGRP) command
  - building display with 7
  - use 410
- Create Search Index (CRTSCHIDX) command 411
- Create Table (CRTTBL) command 264
- creating
  - application display with online help information, steps for 3
  - confirmation list panel 299
  - continued-entry field 174
  - creating 167
  - data presentation panel 303
  - display file and description 18
  - edit description 697
  - graphical look 139
  - horizontal selection field 150
  - list panel 287
  - menu bar 144
  - menu bar panel 320
- creating *(continued)*
  - menu panel 280
  - message 223
  - object 277
  - pull-down menu 154
  - scroll bar 164
  - selection list 161
  - vertical multiple-choice selection field 150
  - vertical single-choice selection field 149
  - window 116
- CRTCLPGM (Create Control Language Program) command 246
- CRTCMD (Create Command) command 411
- CRTDSPF (Create Display File) command
  - building display with 6
  - creating display files 18
  - DFRWRT (defer write) parameter
    - improving performance with 267
    - write operation 64
  - display file menus 242
  - ENHDSP (enhance display) parameter 141
  - program menus 245
  - SHARE parameter 268
- CRTDUPOBJ (Create Duplicate Object) command
  - use 249
- CRTEDTD (Create Edit Description) command
  - example 697
  - running 35
  - use 697
- CRTMNU (Create Menu) command
  - display file menus 243
  - program menus 246
- CRTMSGF (Create Message File) command 243
- CRTPNLGRP (Create Panel Group) command
  - building display with 7
  - use 410
- CRTSCHIDX (Create Search Index) command 411
- CRTTBL (Create Table) command 264
- CSRINPONLY (Cursor Input Only) keyword 29
- CSRLOC (Cursor Location) keyword 66
- CUA (Common User Access)
  - description 456
  - entry dialog actions, table of 456
  - entry display 459
  - entry fields 459
  - function key area:I2.message line relationship 457
  - information displays 460
  - list display, example 460
  - menu display, single-choice 457
  - relationship to system 415
  - selection choices:I2.choice entry field 458
  - single selection field operation, guidelines 458
- cursor
  - actions 189
- cursor *(continued)*
  - movement
    - cursor keys 190
    - horizontal selection field 151
    - right-to-left 28
    - tab key 189
    - to input-capable positions only 29
    - vertical selection field 151
  - position when help text displayed 348
  - positioning
    - after output operation 66
    - during I/O operations 77
    - each page of the subfile record 105
    - rules for list displays 441
    - subfiles 106
    - System/36 application compatibility for display files 702
  - progression
    - entry field 30
  - restrict
    - PULLDOWN keyword 158
    - WINDOW keyword 119
  - returning position to an application 66
- cursor in window 119
- Cursor Input Only (CSRINPONLY) keyword 29
- cursor key
  - movement 190

D

- data 72, 256
  - ASSUME keyword considerations for passing 255
  - checking validity of 73
  - data
    - receiving 44
  - describing inside or outside your program 17
  - externally described
    - definition 17
    - using 17
  - input
    - handling negative numeric 75
    - keeping 72
  - KEEP keyword considerations for passing 255
  - levels of description 16
  - passing
    - between programs 255
    - between programs in same routing step 255
    - between routing steps 256
  - program-described
    - definition 17
    - described 17
    - using 38
  - receiving 44
  - relationship to display file 15
  - rolling between two lines on display 55
  - sending and receiving 44

- DATA (data presentation area) tag
  - example
    - data entry panel 500
    - data entry panel with nested data group 505
    - data presentation area with menu area 504
    - two presentation areas for data items 503
    - two-column format in data entry panel 501
  - overview 496
  - tags allowed table 497
- data area help 391
- data column (DATACOL) tag
  - overview 507
- data description specifications (DDS)
  - *See also* display file
  - *See also* keyword, DDS
  - CAnn keys, restrictions 26
  - changing 19
  - CHECK(FE) (Field Exit) keyword 154
  - DDS form 21
  - definition 3
  - display files
    - passing data between programs 255
    - passing data between routing steps 256
    - use in 21
  - DSPRL (Display Right-to-Left) keyword 31
  - entry displays 452
  - ERRMSG (Error Message) keyword 120
  - ERRMSGID (Error Message ID) keyword 120
  - ERRSFL (Error Subfile) keyword 120
  - example
    - 5250 display station problems 80
    - CLRL and OVERLAY keywords, differences between 54
    - display file menu 241
    - display file source 21
    - DSPATR(PC) keyword 106
    - DSPMOD keyword 65
    - DSPMOD keyword with subfiles 100
    - ERRSFL keyword 226
    - GRDATR (Grid Line Attribute) keyword 205
    - GRDBOX (Grid Box) keyword 205
    - GRDCLR (Grid Clear) keyword 205
    - GRDLIN (Grid Line) keyword 205
    - GRDRCD (Grid Record) keyword 205
    - grid line structure 205
    - HLPCMDKEY keyword 383
    - HLPCMDKEY keyword with response indicators 384
    - HLPPNLGRP keyword 375
    - HLPPNLGRP keyword with option indicators 375

- data description specifications (DDS) *(continued)*
  - example *(continued)*
    - HLPRCD keyword 377
    - HLPRCD keyword with option indicators 377
    - message subfile 105
    - mnemonic 170
    - OVERLAY and CLRL keywords, differences between 54
    - program menu 245
    - PULLINPUT keyword 183
    - push button 167
    - PUTOVR keyword 58
    - PUTOVR keyword for more efficient coding 59
    - PUTOVR keyword for repeatedly entered data 60
    - PUTRETAIN keyword, field level 63
    - PUTRETAIN keyword, record level 62
    - ROLLUP, SFLSIZ, and SFLPAG keywords 102
    - scroll bar 165
    - secondary help 381
    - selection field 148
    - selection list 160
    - SFLMSGID keyword 227
    - SFLNXTCHG keyword 104
    - SFLPAG, SFLSIZ, and ROLLUP keywords 102
    - SFLSIZ, ROLLUP, and SFLPAG keywords 102
    - subfiles for two display sizes 36
    - two display sizes 36
    - variable-length record 103
  - field
    - color (table) 34
    - emphasizing (table) 33
  - help record 120
  - information displays 453
  - keywords ignored if display modes are changed 65
  - list display 454
  - menu displays 451
  - MSGLOC (Message Location) keyword 120
  - NOCCSID (No CCSID) keyword 262
  - processing order for subfile control 94
  - program logic 113
  - programming example 128
  - RMVWDW (Remove Window) keyword 126
  - SFLMSG (Subfile Message) keyword 120
  - SFLMSGID (Subfile Message identifier) keyword 120
  - subfile, describing with 89
  - use 15
  - USRRSTDSP (User Restore Display) keyword 126
  - validity-checking keyword 120
  - WDWBORDER (Window Border) keyword 120
  - window 115

- data description specifications (DDS) *(continued)*
  - WINDOW (Window) keyword 116
- data entry panel
  - example 500
  - introduction 320
  - nested data group example 505
- data group (DATAGRP) tag
  - overview 508
- data item (DATAI) tag
  - overview 510
- data item choices (DATAC) tag
  - overview 506
- data item extender (DATAIX) tag
  - overview 515
- data item group scrolling 348
- data presentation area (DATA) tag
  - example
    - data entry panel 500
    - data entry panel with nested data group 505
    - data presentation area with menu area 504
    - two presentation areas for data items 503
    - two-column format in data entry panel 501
  - overview 496
  - tags allowed table 497
- data presentation panel application programming 317
- data presentation panel creating 303
- data presentation panel source 306
- data queue
  - waiting on display file and ICF file 259
- data selection field (DATASLT) tag
  - example
    - data entry panel 520
    - multiple-selection field 521
  - overview 518
  - tags allowed table 519
- data selection field choice (DATASLTC) tag
  - overview 523
- data stream, 5250 display
  - *See* user-defined data stream (UDDS)
- data stream, user-defined
  - *See* user-defined data stream (UDDS)
- data tag, modified 23
- DATAC (data item choices) tag
  - overview 506
- DATACOL (data column) tag
  - overview 507
- DATAGRP (data group) tag
  - overview 508
- DATAI (data item) tag
  - overview 510
- DATAIX (data item extender) tag
  - overview 515
- DATASLT (data selection field) tag
  - example
    - data entry panel 520
    - multiple-selection field 521
  - overview 518
  - tags allowed table 519

- DATASLTC (data selection field choice) tag
  - overview 523
- date value
  - display form 490
- DBCS (double-byte character set) printing 365
- DBCS and national language considerations 408
- DBCS field
  - continued-entry field 175
- DBCS grid line structure
  - *See* grid line structure
- DDS (data description specifications)
  - *See also* display file
  - *See also* keyword, DDS
  - CAnn keys, restrictions 26
  - changing 19
  - CHECK(FE) (Field Exit) keyword 154
  - DDS form 21
  - definition 3
  - display files
    - passing data between programs 255
    - passing data between routing steps 256
    - use in 21
  - DSPRL (Display Right-to-Left) keyword 31
  - entry displays 452
  - ERRMSG (Error Message) keyword 120
  - ERRMSGID (Error Message ID) keyword 120
  - ERRSFL (Error Subfile) keyword 120
  - example
    - 5250 display station problems 80
    - CLRL and OVERLAY keywords, differences between 54
    - display file menu 241
    - display file source 21
    - DSPATR(PC) keyword 106
    - DSPMOD keyword 65
    - DSPMOD keyword with subfiles 100
    - ERRSFL keyword 226
    - GRDATR (Grid Line Attribute) keyword 205
    - GRDBOX (Grid Box) keyword 205
    - GRDCLR (Grid Clear) keyword 205
    - GRDLIN (Grid Line) keyword 205
    - GRDRCD (Grid Record) keyword 205
    - grid line structure 205
    - HLPCMDKEY keyword 383
    - HLPCMDKEY keyword with response indicators 384
    - HLPPNLGRP keyword 375
    - HLPPNLGRP keyword with option indicators 375
    - HLPRCD keyword 377
    - HLPRCD keyword with option indicators 377

DDS (data description specifications) *(continued)*
  - example *(continued)*
    - message subfile 105
    - mnemonic 170
    - OVERLAY and CLRL keywords, differences between 54
    - program menu 245
    - PULLINPUT keyword 183
    - push button 167
    - PUTOVR keyword 58
    - PUTOVR keyword for more efficient coding 59
    - PUTOVR keyword for repeatedly entered data 60
    - PUTRETAIN keyword, field level 63
    - PUTRETAIN keyword, record level 62
    - ROLLUP, SFLSIZ, and SFLPAG keywords 102
    - scroll bar 165
    - secondary help 381
    - selection field 148
    - selection list 160
    - SFLMSGID keyword 227
    - SFLNXTCHG keyword 104
    - SFLPAG, SFLSIZ, and ROLLUP keywords 102
    - SFLSIZ, ROLLUP, and SFLPAG keywords 102
    - subfiles for two display sizes 36
    - two display sizes 36
    - variable-length record 103
  - field
    - color (table) 34
    - emphasizing (table) 33
  - help record 120
  - information displays 453
  - keywords ignored if display modes are changed 65
  - list display 454
  - menu displays 451
  - MSGLOC (Message Location) keyword 120
  - NOCCSID (No CCSID) keyword 262
  - processing order for subfile control 94
  - program logic 113
  - programming example 128
  - RMVWDW (Remove Window) keyword 126
  - SFLMSG (Subfile Message) keyword 120
  - SFLMSGID (Subfile Message identifier) keyword 120
  - subfile, describing with 89
  - use 15
  - USRRSTDSP (User Restore Display) keyword 126
  - validity-checking keyword 120
  - WDWBORDER (Window Border) keyword 120
  - window 115
  - WINDOW (Window) keyword 116
- DDS form 21
- DDS keyword
  - *See* keyword, DDS
- DDS record
  - *See* record, DDS
- decimal point, edit description for 695
- Default (DFT) keyword
  - use 64
  - with ASSUME keyword 255
- default value
  - fields 64
- defining 240
  - accelerator key 156
  - application help 185
  - application pop-up window 359
  - application window 359
  - attributes for entry fields 31
  - cancel key 158
  - choice colors and attributes 171
  - choice-level help 185
  - contextual help in UIM 348
  - cursor progression for entry fields 30
  - dialog variables 335
  - function keys 351
  - function keys and VARUPD value 351
  - header area 365
  - help for field 187
  - lists 341
  - menu object 280
  - menu-bar switch key 157
  - mnemonic 170
  - MNUBARDSP (Menu-Bar Display) keyword
    - application record 180, 190
    - menu-bar record 181, 193
  - panel group object 287
  - prolog area 364
  - scrollable areas 346
- definition
  - panel 467
- definition list
  - device 664
- definition list (DL) tag
  - example
    - compact definition list 527
    - definition list 526
  - overview 525
  - tags allowed table 525
  - use in help module 402
- Delete Menu (DLTMNU) command 249
- Delete Override (DLTOVR) command 217
- Delete Panel Group (DLTPNLGRP) command 410
- Delete Search Index Entry (RMVSCHIDXE) command 412
- deleting
  - menu displays 249
  - overrides 217
  - panel group 410
  - search index 412
  - window 126
- description
  - field-level
    - definition 16
    - use 16

- description *(continued)*
  - file-level
    - definition 17
    - use 17
  - record-level
    - definition 16
    - use 16
- description, file
  - changing
    - reasons for 16
    - with the CHGDSPF command 19
  - creating for display file 18
  - definition 3
  - detecting changes in 20
  - determining 16
- descriptive text area, entry displays 427
- device definition list
  - definition 43
  - part of open feedback area 664
- DFRWRT (defer write) parameter
  - improving performance with 267
  - restrictions 64
  - using 64
- DFT (Default) keyword
  - use 64
  - with ASSUME keyword 255
- DFTVAL (Default Value) keyword 64
- dialog command
  - ACTIONS (Menu Bar Cursor Action) 644
  - CALL (Call Program) 644
  - Call Program (CALL) 644
  - CANCEL (Cancel) 646
  - Change View (Change View) 647
  - CHGVIEW (Change View) 647
  - CMD (System Command) 647
  - CMDLINE (Command Line) 648
  - Command Line (CMDLINE) 648
  - Display Help (DSPHELP) 648
  - Display Home Menu (HOME) 652
  - Display Message (MSG) 654
  - Display More Function Keys (MOREKEYS) 653
  - DSPHELP (Display Help) 648
  - effects of 641
  - ENTER (Enter) 649
  - entry field prompting 657
  - EXIT (Exit Display) 650
  - Exit Display (EXIT) 650
  - Extended Help (EXTHELP) 650
  - EXTHELP (Extended Help) 650
  - HELP (Help) 651
  - HELPHELP (Help Help) 651
  - HELPIDX (Help Index) 651
  - HOME (Display Home Menu) 652
  - introduction 641
  - KEYSHELP (Keys Help) 652
  - MENU (Menu) 653
  - Menu Bar Cursor Action (ACTIONS) 644
  - MOREKEYS (Display More Function Keys) 653
  - Move to Top (MOVETOP) 654
  - MOVETOP (Move to Top) 654
  - MSG (Display Message) 654
  - PAGEDOWN (Page Down) 655
  - PAGEUP (Page Up) 655
- dialog command *(continued)*
  - PRINT (Print Display) 656
  - Print Display (PRINT) 656
  - PROMPT (Prompt) 656
  - prompting an entry field 657
  - PULLDOWN (Display Pull-down Menu) 658
  - RETRIEVE (Retrieve Command String) 658
  - Retrieve Command String (RETRIEVE) 658
  - RETURN (Return Control to Application) 659
  - Return Control to Application (RETURN) 659
  - System Command (CMD) 647
  - valid uses 641
  - variable update (VARUPD) attribute 643
  - VARUPD attribute 643
- dialog variable
  - defined by UIM 636
  - defining 335
  - definition 335
  - displaying panel field value using 337
  - error messages 337
  - incorrect display characters 337
  - initial value 336
  - restrictions 336
  - variable pool 335
  - Z36ENV 636
  - ZCANCEL 636
  - ZDBCS 636
  - ZDSPSIZ 636
  - ZEXIT 636
  - ZJOB 636
  - ZJOBNBR 636
  - ZLMTCPB 636
  - ZMENU 636
  - ZMNULIB 636
  - ZSYSNAM 636
  - ZUSER 636
- dialog variables and special values 338
- differences between pull-down menus and pop-up windows 358
- display
  - creating UIM 287
  - list 341
  - panel field value using dialog variables 337
  - UIM 287
  - user interface manager 287
- display attribute
  - choices 171
  - color fields 34
  - emphasizing fields 33
- display attribute (*DSPATR) parameter 171
- Display Attribute (DSPATR) keyword
  - example source for PC value 106
  - use 66
  - with ASSUME keyword 255
  - with program-to-system field 33
- display character
  - incorrect 337
- display device file
  - *See* display file
- Display Edit Description (DSPEDTD) command 35
- display file
  - *See also* data description specifications (DDS)
  - *See also* subfile
  - *See also* window
  - ASSUME (Assume)
    - considerations 255
  - attributes for fields, System/36 application compatibility 702
  - cancel-invite operation, System/36 application compatibility 707
  - characteristics of 15
  - closing 83
  - command key validity 80
  - creating
    - ways for 18
    - with DDS 18
    - without DDS 38
  - cursor positioning, System/36 application compatibility 702
  - defining input and output 39
  - defining your display in 15
  - definition 3
  - description
    - creating 18
    - how changes are applied to 19
  - display station controller, System/36 application compatibility 700
  - displaying messages, System/36 application compatibility 703
  - edit codes, user-defined 695
  - edit descriptions 693
  - erase-input-fields function, System/36 application compatibility 701
  - establishing 15
  - example, programming 128
  - examples
    - assigning space to program-described 38
    - building with online help 3
    - DDS source 21
    - replacing record formats on display 48
  - file entry field attributes 259
  - function keys, System/36 application compatibility 706
  - HELP and HLPRTN keywords, System/36 application compatibility 706
  - input data
    - System/36 application compatibility 700
  - input data, System/36 application compatibility 700
  - input-only, defining 39
  - lock 41
  - maximum number of record formats in 28
  - obtaining information about 43
  - online help information
    - *See* online help information
  - opening 41

- display file *(continued)*
  - operations supported by the system and high-level languages
    - CL commands 84
    - ILE C/C++ functions 84
    - ILE COBOL statements 84
    - ILE RPG operations 84
  - operator changes, System/36 application compatibility 701
  - output-only, defining 39
  - overriding
    - attributes in HLL programs 213
    - file names and attributes in HLL programs 215
    - file names in HLL programs 214
  - passing data
    - between programs 255
    - between routing steps 256
  - put with explicit override, System/36 compatibility 705
  - relationship to data 15
  - retain command and function keys, System/36 application compatibility 708
  - return codes 677
  - screen size, specifying to improve performance 267
  - self-check, System/36 application compatibility 701
  - sharing in same job 85
  - signed numeric data, System/36 application compatibility 705
  - suspended 78
  - System/36 application compatibility 699
  - System/36 functions not supported 709
  - use 41
  - waiting for data with ICF file, data queue 259
  - window use 115
- display file attribute
  - *See* file attribute
- Display File Description (DSPFD) command 20
- Display File Field Description (DSPFFD) command 20
- display file menu
  - *See* menu display
- display form
  - binary value 489
  - character value 490
  - date value 490
  - example numeric values 489
  - numeric value 489
  - time value 490
- Display Help (DSPHELP) dialog command 648
- Display Home Menu (HOME) dialog command 652
- display I/O feedback area 670
- Display Menu Attributes (DSPMNUA) command 249
- Display Message (MSG) dialog command 654
- display mode
  - changing 100
- display mode *(continued)*
  - DDS keywords ignored if changed 65
  - definition 100
  - displaying subfiles horizontally with 100
- Display Mode (DSPMOD) keyword
  - description 65
  - example for subfiles 100
  - sample subfile display for *DS3 101
  - sample subfile display for *DS4 101
  - subfile use 100
- Display More Function Keys (MOREKEYS) dialog command 653
- Display Override (DSPOVR) command 217
- display panel (PANEL) tag
  - example
    - panel definition 601
  - overview 595
  - tags allowed table 595
- display panel field value using dialog variables 337
- Display Program References (DSPPGMREF) command 20
- Display Right-to-Left (DSPRL) keyword 31
- Display Size (DSPSIZ) keyword 36
- display station 41, 42
  - acquired, releasing from I/O operations 83
  - calculating maximum number of fields 75
  - device description, changing for printers 37
  - enhanced capabilities 141
  - models
    - 3180-2 65
    - 3197 65
    - 5250, avoiding record format problems on 80
  - obtaining information about 43
  - types 116, 139
  - user-defined data streams, differences for 251
  - with UNLOCK 71
- display station controller, System/36 application compatibility 700
- display, application
  - *See* application display
- displaying
  - list 341
  - message
    - about 223
    - in field 225
    - in program message queue 225
    - on message line 224
    - on message line using message field 224
    - on message line when subfile control record written 224
    - through subfile 225
  - override 217
  - panel using request processor program 361
  - System/36 application compatibility 703
- DL (definition list) tag
  - example
    - a definition list 526
    - compact definition list 527
  - overview 525
  - tags allowed table 525
  - use in help module 402
- DLTMNU (Delete Menu) command 249
- DLTOVR (Delete Override) command 217
- DLTPNLGRP (Delete Panel Group) command 410
- DLTSCHIDXE (Delete Search Index Entry) command 412
- document, office
  - definition for help 7
- double strike 364
- double-byte character set
  - mnemonics 170
- DSPATR (Display Attribute) keyword
  - example source for PC value 106
  - use 66
  - with ASSUME keyword 255
  - with program-to-system field 33
- DSPEDTD (Display Edit Description) command 35
- DSPFD (Display File Description) command 20
- DSPFFD (Display File Field Description) command 20
- DSPHELP (Display Help) dialog command 648
- DSPMNUA (Display Menu Attributes) command 249
- DSPMOD (Display Mode) keyword
  - description 65
  - Display Mode (DSPMOD) 100
  - example for subfiles 100
  - sample subfile display for *DS3 101
  - sample subfile display for *DS4 101
  - subfile use 100
  - use 64
- DSPOVR (Display Override) command 217
- DSPPGMREF (Display Program References) command 20
- DSPRL (Display Right-to-Left) keyword 31
- DSPSIZ (Display Size) keyword
  - specifying to improve performance 267
  - use 36
- DTAQ (data queue) parameter 259
- duplicating window 124
- dynamic list column heading formatting
  - example 560

E

- edit code
  - defining your own 35
  - definition 35
  - description 695
  - IBM-supplied 693
  - use 35
- Edit Code (EDTCDE) keyword 35

- edit description
  - creating 697
    - about 697
    - example 697
  - decimal point 695
  - description 695
  - IBM-supplied 695
  - rules 696
- edit mask
  - creating 31
- edit word
  - definition 35
- Edit Word (EDTWRD) keyword 35
- EDTCDE (Edit Code) keyword 35
- EDTWRD (Edit Word) keyword 35
- elements within a panel group 277
- embed
  - *See* imbed
- emphasis values table 574
- emphasizing fields 33
- enabled selection indicator in selection list
  - example 162
- enabling graphical user interface (GUI)
  - support 346
- End Receive (ENDRCV) command 68
- ending attribute character
  - definition 22
- ENDRCV (End Receive) command 68
- engraved key restrictions 346
- enhanced display (ENHDSP) parameter
  - writing records 141
- ENHDSP (enhanced display) parameter
  - writing records 141
- ENTER dialog command 649
- entering
  - commands that are too long 351
- entry dialog actions, table of CUA 456
- entry display
  - CUA entry field 459
  - description 423
  - descriptive text area 427
  - entry fields 426, 427
  - examples
    - CUA entry field 459
    - i5/OS-style 423
  - field prompts 425
  - instruction lines 424
  - online help information
    - DDS considerations 452
    - description 451
    - help areas for 452
    - types 451
  - operating guidelines 429
  - possible choices information 427
  - prompt areas 425
  - title 424
- entry field
  - attributes for 31
  - CUA 459
  - cursor progression for 30
  - entry displays 426
  - menu selection 423
  - prompting 657
  - protecting using Edit Masks 31
  - rules for displays 427
- entry field attribute
  - creating 31
- entry pointer for list 342
- ERASE (Erase) keyword
  - erasing records 50
  - restriction with USRDSPMGT keyword 709
  - System/36 application compatibility 700
  - with OVERLAY keyword 50
- Erase Input (ERASEINP) keyword
  - system performance considerations 268
  - use 61
- erase-input-fields function, System/36 application compatibility 701
- ERASEINP (Erase Input) keyword
  - system performance considerations 268
  - use 61
- erasing input 61
- ERRMSG (Error Message) keyword 120, 224
- ERRMSGID (Error Message Identifier) keyword 120, 224
- error
  - *See also* major/minor return code
  - *See also* message
  - field validation 71
  - recovering from 231
  - recovering from run-time 86
  - while scrolling 347
- error condition 347
- error message
  - dialog variables 337
  - window 119
- Error Message (ERRMSG) keyword 120, 224
- Error Message ID (ERRMSGID) keyword 120
- Error Message Identifier (ERRMSGID) keyword 224
- Error Message Subfile (ERRSFL) keyword
  - example DDS source 226
  - message received from 120
  - restriction with USRDSPMGT keyword 709
  - use 225
- ERRSFL (Error Message Subfile) keyword
  - example DDS source 226
  - message received from 120
  - restriction with USRDSPMGT keyword 709
  - use 225
- example
  - a text area 627
  - accelerator key 157
  - alternate view of list 288
  - application formatted area 474
  - building sample display with online help 3
  - check box 139
  - CL programs
    - program menu 245
    - RTNPNT(*NO) for program menu, CL source for 247
  - class definition 488
- example *(continued)*
  - command line pop-up window 361
  - compact definition 527
  - conditioning option 495
  - continued attribute 465
  - continued-entry field 174
  - data description specifications (DDS)
    - 5250 display station problems 80
    - ALWROL keyword 56
    - CHRID keyword 261
    - CLRL and OVERLAY keywords, differences between 54
    - CLRL(*NO) keyword 53
    - display file menu 241
    - display file source 21
    - display size, for more than one 36
    - DSPATR(PC) keyword 106
    - DSPMOD keyword 65
    - DSPMOD keyword with subfiles 100
    - ERRSFL keyword 226
    - HLPCMDKEY keyword 383
    - HLPCMDKEY keyword with response indicators 384
    - HLPPNLGRP keyword 375
    - HLPPNLGRP keyword with option indicators 375
    - HLPRCD keyword 377
    - HLPRCD keyword with option indicators 377
    - INZRCD and UNLOCK keywords 72
    - message subfile 105
    - OVERLAY and CLRL keywords, differences between 54
    - program logic for subfiles 113
    - program menu 245
    - PUTOVR keyword 58
    - PUTOVR keyword for repeatedly entered data 60
    - PUTOVR keyword, more efficient coding 59
    - PUTRETAIN keyword, field level 63
    - PUTRETAIN keyword, record level 62
    - ROLLUP, SFLSIZ, and SFLPAG keywords 102
    - RSTDSP parameter 78
    - secondary help 381
    - SFLMSGID keyword 227
    - SFLNXTCHG keyword 104
    - SFLPAG, SFLSIZ, and ROLLUP keywords 102
    - SFLSIZ, ROLLUP, and SFLPAG keywords 102
    - SLNO(*VAR) keyword 51
    - subfiles for two display sizes 36
    - UNLOCK and INZRCD keywords 72
    - variable-length record 103
  - data entry panel 500, 520
  - data entry panel with nested data group 505
  - data entry panel, two-column format 501

- example *(continued)*
  - data presentation panel 303
  - data presentation panel after scrolling 305
  - defining MNUBARDSP on application record 180, 190
  - defining MNUBARDSP on menu-bar record 181, 193
  - definition list 526
  - display with output and input fields displayed in response to request 23
  - dynamic list column heading 560
  - edit description, creating 697
  - enabled selection indicator in selection list 162
  - entry displays
    - CUA 459
    - typical 423
  - formatting example 640
  - graphic image
    - HTML keyword 209
  - grid line structure 205
  - heading tags 534
  - help modules
    - command help 411
    - imbedded 400
    - imported 400
    - ISCH tag 405
    - ISCHSYN tag 405
    - LINK tag 410
    - panel group with 399
  - help panel definition 531
  - horizontal selection field 150
  - HTML keyword
    - graphic image 209
    - resolving field overlap 208
  - hypertext link 550
  - imbedded help 535
  - index search 539
  - index search synonym 541
  - information display 429
  - key definitions 544
  - key list 545
  - left-to-right formatting on left-to-right panel 621
  - left-to-right formatting on right-to-left panel 621
  - list
    - ordered 592
    - simple 623
    - translation 630
    - unordered 634
  - list actions 567
  - list area
    - example, UIM source 557
    - list column groups 559
    - three layout columns 558
  - list display
    - CUA 460
    - extended action entry area 432
  - list panel 287
  - list part 580
  - menu area with groups 586
  - menu bar
    - choices 144
    - DDS 190
    - panel 320

- example *(continued)*
  - menu bar *(continued)*
    - UIM 581
  - menu displays
    - display file menu 241
    - message file, adding messages to 243
    - message file, describing menu actions in 239
    - naming help formats in message file 239
    - program menu 245
    - program menu, CL source for 245
    - RTNPNT(*NO) for program menu, CL source for 247
    - single choice 421
    - using options and GO command 235
  - menu panel
    - creating 280
  - menu source 282
  - menu-bar choice help 185
  - menu-bar help list 187
  - message
    - ERRSFL keyword 226
    - SFLMSGID keyword 227
  - message, trailer 619
  - mnemonic, not valid DDS for 170
  - multiple-choice selection list 159
  - multiple-selection field (UIM) 521
  - note 591
  - numeric values table 489
  - online help information
    - command help 411
    - display showing help areas 372
    - HLPCMDKEY keyword 383
    - HLPCMDKEY keyword with response indicators 384
    - HLPPNLGRP keyword 375
    - HLPPNLGRP keyword with option indicators 375
    - HLPRCD keyword 377
    - HLPRCD keyword with option indicators 377
    - imbedded help modules in panel group 400
    - imported help modules in panel group 400
    - ISCH tag 405
    - ISCHSYN tag 405
    - LINK tag 410
    - panel groups with help modules 399
    - secondary help 381
  - ordered list 592
  - output operation
    - effect of OVERLAY and ERASE on 50
  - overrides
    - applying when compiling program 216
    - attribute of display file 213
    - different display file 214
    - different display file and attributes 215
  - panel definition 601

- example *(continued)*
  - panel groups
    - command help 411
    - help modules 399
    - HLPPNLGRP keyword 375
    - HLPPNLGRP keyword with option indicators 375
    - imbedded help modules 400
    - imported help modules 400
    - ISCH tag 405
    - ISCHSYN tag 405
    - LINK tag 410
  - paragraph continuation 603
  - paragraph tag 594
  - parameter list 603
  - passing data
    - between routing steps 256
    - in same routing step 255
  - print title line 617
  - program-described display file 38
  - pull-down menu 154, 190
  - PULLINPUT parameter 183
  - push button 167
  - radio button 139
  - record format, replacing 48
  - record, DDS
    - HLPCMDKEY keyword 383
    - HLPCMDKEY keyword with response indicators 384
    - HLPRCD keyword 377
    - HLPRCD keyword with option indicators 377
    - secondary help 381
  - return-field (RTNFLD) parameter 184
  - roll operation 56
  - sample figure 528
  - sample heading 639
  - scroll bar 159, 163
  - selection field 148, 149
  - selection list 160
  - simple list 623
  - simple menu area 585
  - single-choice selection field choice help 186
  - single-choice selection list 159
  - SLNO (Starting Line Number) keyword 51
  - subfile
    - *DS3 mode display 101
    - *DS4 mode display 101
    - DSPATR(PC) keyword 106
    - DSPMOD keyword 100
    - message subfile 105
    - program logic for subfiles 113
    - ROLLUP, SFLSIZ, and SFLPAG keywords 102
    - sample display for changing information 88
    - sample display for display only 87
    - sample display for input only 88
    - sample display for task combination 88
    - sample display with selection 87
    - sample displays when rolling two or more records 110

- example *(continued)*
  - subfile *(continued)*
    - SFLNXTCHG keyword 104
    - SFLPAG, SFLSIZ, and ROLLUP keywords 102
    - SFLSIZ, ROLLUP, and SFLPAG keywords 102
    - variable-length record 103
    - vertical and horizontal, displayed at same time 99
  - suppressed selection indicator in pull-down menu 156
  - suppressed selection indicator in selection field 153
  - title citation 477
  - translation list 630
  - truth table 632
  - two presentation areas
    - data items 503
    - menu area 504
  - unavailable choice 169
  - unformatted lines 546
  - unordered list 634
  - valid DDS for mnemonic 170
  - validity checking 477
  - vertical multiple-choice selection field 150
  - vertical single-choice selection field 149
  - window title 123
  - window use 128
- example (XMP) tag
  - example, formatting an example 640
  - help module 402
  - overview 639
- EXIT (Exit Display) dialog command 650
- Exit Display (EXIT) dialog command 650
- Exit key, using on menus 235
- exit program
  - commonly asked questions 365
- expression
  - conditional 492, 548
- extended action entry area
  - list displays 435
- extended alphabetics
  - definition 261
- extended help
  - definition 371, 386
- Extended Help (EXTHELP) dialog command 650
- extended help heading (XH1 through XH4) tag
  - example, sample heading 639
  - formatting rules 639
  - overview 638
  - use in help module 401
- externally described data
  - advantages of 17
  - definition 17
  - use 17
- EXTHELP (Extended Help) dialog command 650

F

- F18=More indexes
  - removing access 388
- feedback area
  - common 43
  - error information 43
  - file-dependent 43
  - function keys 25
  - get attributes 673
  - I/O
    - common 667
    - definition 43
    - display 670
    - for function keys 25
  - open
    - definition 43
    - device definition list 664
    - individual descriptions 661
- field
  - attribute
    - defining 31
  - attributes, System/36 application compatibility 702
  - calculating maximum number for display station 75
  - character, maximum length 24
  - characteristics of 16
  - choice entry, Common User Access 458
  - color 34
  - constant 23
  - cursor progression
    - defining 30
  - default values, specifying 64
  - defining
    - attribute 31
    - cursor progression 30
  - definition 16
  - display 21
  - displaying 22
  - emphasizing 33
  - entry
    - *See* entry field
  - erasing unprotected input and output/input 61
  - hidden 22
  - input 22
  - input and output, displayed in response to request 23
  - input-capable, limitations on number of 75
  - keeping on display 62
  - list
    - *See* list fields
  - maximum number in record format 23
  - menu selection entry 423
  - message 23
  - numeric
    - maximum length 24
  - output 22
  - output/input 22
  - overriding attributes or content of 57
  - previously described 24
  - program-to-system
    - understanding how to use 23
    - with DSPATR keyword 33
- field *(continued)*
  - record format 22
  - reference 24
  - validation errors 71
  - zoned decimal, maximum length 24
- field (*FLD) special value 186
- field attribute character 22
- Field Cursor Progression (FLDCSRPRG) keyword 30
- Field Exit (CHECK(FE)) keyword 154
- field location
  - input and output records 22
  - when to specify 24
- field prompt, entry displays 425
- field value for display panel using dialog variables 337
- field-exit processing
  - backward 180
  - forward 179
- field-level description
  - definition 16
  - use 16
- field, graphic 471
- FIG (figure) tag
  - example, sample figure 528
  - overview 527
  - use in help module 402
- figure (FIG) tag
  - example, sample figure 528
  - overview 527
  - use in help module 402
- file
  - definition 17
  - display
    - *See* display file
  - shared in job
    - close operation for 86
    - input/output operation for 85
    - open operation for 85
  - source
    - *See* source file
- file attribute
  - building 213
  - overriding
    - along with display file name in HLL programs 215
    - display files in HLL programs 213
- file description
  - changing 19
    - reasons for 16
    - with the CHGDSPF command 19
  - creating for display file 18
  - definition 3
  - detecting changes in 20
  - determining 16
- file redirection
  - definition 217
  - different file types 218
  - level checking 217
  - same file types 217
  - table of defaults for combinations 218
  - valid for input and output 218
- file resource
  - how allocated 41
- file-dependent feedback area 43

file-level description 17
FLDCSRPRG (Field Cursor Progression) keyword 30
folding
  list panel 355
  multiple panels when EXIT requested 354
  when EXIT requested 354
font
  highlighting 364
format, record
  See record format
formatting
  bidirectional considerations 550
  column data 572
  display form 489
    binary value 489
    character value 490
    date value 490
    time value 490
  example 640
  function keys 351
  IGC (ideographic characters)
    BOTINST (bottom instruction) tag 475
    DATAC (data item choices) tag 506
    PDFLDC (pull-down field choice) tag 608
    TOPINST (top instruction) tag 631
  numeric value
    example values table 489
  panel
    concepts 352
    user control 354
  panels 352
  printed
    information area 538
    list area 556
forward field-exit processing 179
full screen mode
  help information 530
function key
  assignments, table 446
  common assignments on i5/OS-style displays 446
  CUA entry dialog actions, table of 456
  DDS keywords for 25
  defining
    in DDS 24
    in UIM 351
  formatting in UIM 351
  handling in UIM 351
  optional for i5/OS-style displays, table of 421
  required for i5/OS-style displays, table of 421
  retain command, System/36 application compatibility 708
  scrolling in UIM 347
  System/36 application compatibility 706
  VARUPD value 351
function key area
  CUA 457
function key area *(continued)*
  menu displays, single-choice 421
function key area help 396
function, built-in
  CHKOBJ (check object) 493, 549
  CHKPGM (COND evaluation program) 494
  CHKPGM (condition evaluation program) 550
  CHKUSRCLS (check user class) 494, 550

G

get-attributes feedback area 673
get-attributes operation
  definition 43
  ILE C/C++ function for 84
  ILE COBOL statement for 84
  ILE RPG operation for 84
  table of information available from 43
get-next-changed operation
  definition 97
get-next-changed operation for subfiles
  BASIC statement for 98
  ILE C/C++ function for 98
  ILE COBOL statement for 98
  ILE RPG operation for 98
get-relative operation
  definition 97
get-relative operation for subfiles
  BASIC statement for 98
  ILE C/C++ function for 98
  ILE COBOL statement for 98
  ILE RPG operation for 98
GETRETAIN (Get Retain) keyword 72
GO (Go to Menu) command
  generic menu specification 248
  use 235
Go to Menu (GO) command
  generic menu specification 248
  use 235
graphical user interface (GUI)
  support 346
graphics
  field 471
  literal 471
GRDATR (Grid Line Attribute) keyword 205
GRDBOX (Grid Box) keyword 205
GRDCLR (Grid Clear) 205
GRDLIN (Grid Line) 205
GRDRCD (Grid Record) 205
Grid Box (GRDBOX) keyword 205
Grid Clear (GRDCLR) 205
Grid Line (GRDLIN) 205
Grid Line Attribute (GRDATR) keyword 205
grid line structure
  definition 205
  example, DDS 205
  hardware requirements 207
  overview 205
  window 206
Grid Record (GRDRCD) 205
group object, panel 467
GUI (graphical user interface)
  support 346
guideline
  CUA, single selection field operation 458
  message 455
  operating
    entry displays 429
    information displays 432
    list display 437
    menu displays 423

H

H specification
  definition 371
  effect on help-list updating 380
  entering in DDS source 372
  order in DDS source 372
  paging order 379
H1 through H4 (headings)
  example
    heading tags 534
  help module 401
  overview 533
handle on list entry 342
handling function keys and VARUPD value 351
header area
  defining 365
heading (H1 through H4) tag
  example
    heading tags 534
  overview 533
  use in help module 401
heading, column
  See column heading
help
  choice-level 185
  data area 391
  defining for application 185
  displayed in window 120, 122
  field 187
  function key area 396
  item specific 187
  list area 386
  menu area 389
  menu bar area 394
  panels 468
HELP (help module) tag
  example, help panel definition 531
  overview 529
  placement in UIM source 399
  UIM source showing panel groups with help modules 399
HELP (Help) keyword
  enabling the Help key 370
  function key use 25
  specifying alternative 26
  System/36 application compatibility 706
help area
  See online help information
Help Area (HLPARA) keyword 371
help boundary
  definition 381

Help Boundary (HLPBDY) keyword
  secondary help 381
  use 381
Help Command Key (HLPCMDKEY) keyword
  restriction with USRDSPMGT keyword 709
  use 383
HELP dialog command 651
help display
  See online help information
Help Exclude (HLPEXCLD) keyword 373
help for field 187
Help Full (HLPFULL) keyword 373
help group
  definition 381
help information
  See online help information
help key 385
Help key
  See also ALTHELP (Alternate Help) keyword
  See also HELP (Help) keyword
  enabling 370
  results when pressed 369
  returning to program after pressing 383
  returning to program without pressing 384
help list
  defining for application 187
  definition 379
help module
  contained in different help panel group, using 400
  definition 399
  emphasizing and formatting text within 401
  examples
    command help 411
    imbedded 400
    imported 400
    ISCH tag 405
    ISCHSYN tag 405
    LINK tag 410
    panel group with 399
  headings, adding 401
  highlighting text 402
  hypertext 408
  linking 408
  making lists 402
  names 400
  organizing panel group with 399
  paragraphs and notes, defining 401
  programming keywords and variables, identifying 402
  structured text, indicating 402
  using more than once 400
help module (HELP) tag
  example, help panel definition 531
  overview 529
  placement in UIM source 399
Help Panel Group (HLPPNLGRP) keyword
  use 373
Help Panel Group (HLPPNLGRP) keyword *(continued)*
  with HLPRCD and HLPDOC keywords 375
  with option indicators 375
Help Record (HLPRCD) keyword
  with HLPDOC keyword in same file 378
  with HLPPNLGRP keyword in same file 376
  with option indicators 377
Help Return (HLPRTN) keyword
  returning to your program after help 384
  System/36 application compatibility 706
Help Search Index (HLPSCHIDX) keyword 373
Help Sequencing (HLPSEQ) keyword 381
help sublist
  definition 381
help text
  See online help information
help text displayed according to cursor position 348
Help Title (HLPTITLE) keyword 373
HELPHELP dialog command 651
HELPIDX dialog command 651
hexadecimal data 472
hidden field
  definition 22
  using 22
hierarchy, index search 406
high-level language (HLL)
  overriding display file attributes in programs 213
  overriding display file names and attributes in programs 215
  overriding display file names in programs 214
  recognizing the I/O requests supported by 84
  System/36 application compatibility 699
highlight phrases table 532
highlighted phrase (HP0 through HP9) tag
  overview 532
  use in help module 402
highlighting
  selection characters 473, 625
highlighting and fonts 364
HLL (high-level language)
  overriding display file attributes in programs 213
  overriding display file names and attributes in programs 215
  overriding display file names in programs 214
  recognizing the I/O requests supported by 84
  System/36 application compatibility 699
HLPARA (Help Area) keyword 371
HLPBDY (Help Boundary) keyword
  secondary help 381
HLPBDY (Help Boundary) keyword *(continued)*
  use 381
HLPCMDKEY (Help Command Key) keyword
  response indicators, sample source with 384
  restriction with USRDSPMGT keyword 709
  sample DDS source 383
  use 383
HLPEXCLD (Help Exclude) keyword 373
HLPFULL (Help Full) keyword 373
HLPPNLGRP (Help Panel Group) keyword
  example
    option indicators, sample source with 375
    sample DDS source 375
  use 373
  with HLPRCD and HLPDOC keywords 375
  with option indicators 375
HLPRCD (Help Record) keyword
  example
    option indicators, sample source with 377
    sample DDS source 377
    secondary help 381
  with HLPDOC keyword in same file 378
  with HLPPNLGRP keyword in same file 376
  with option indicators 377
HLPRTN (Help Return) keyword
  returning to your program after help 384
  System/36 application compatibility 706
HLPSCHIDX (Help Search Index) keyword 373
HLPSEQ (Help Sequencing) keyword 381
HLPTITLE (Help Title) keyword 373
HOME (Display Home Menu) dialog command 652
HOME (Home) keyword 25
horizontal selection field
  cursor movement 151, 152
  example 150
HP0 through HP9 (highlighted phrase) tag
  overview 532
  use in help module 402
HTML (Hypertext Markup Language) keyword
  inserting 207
  resolving field overlap 208
    examples 208
  restrictions 211
hypertext
  See also online help information
  definition 408
hypertext link
  example 550

- hypertext link definition (LINK) tag
  - CHKPGM (condition evaluation program) 550
  - conditional expressions
    - CHKOBJ built-in function 548
    - CHKUSRCLS (check user class) built-in function 550
  - example, hypertext link 550
  - overview 547
  - placement in UIM source 410
- Hypertext Markup Language (HTML) keyword
  - resolving field overlap 208
  - using 207
- hypertext reference phrase
  - *See also* online help information
  - definition 410

I

- I/O feedback area
  - common 667
  - contents 43
  - definition 43
  - display 670
- I/O operation
  - acquiring a display station for 42
  - locking keyboard and positioning cursor 77
  - mapping to HLL operations
    - CL commands 84
    - ILE C/C++ functions 84
    - ILE COBOL statements 84
    - ILE RPG operations 84
  - obtaining information about 43
  - releasing an acquired display station from 83
  - requesting
    - for subfile control record formats 98
    - for subfile record formats 96
    - for subfiles 96
- I/O request
  - high-level languages, table of subfile 98
  - supported by high-level languages 84
- ICF file
  - file entry field attributes 259
  - waiting for data with display file, data queue 259
- identifier fields for menu displays 443
- identifier, level 20
- ideographic characters (IGC)
  - formatting
    - BOTINST (bottom instruction) tag 475
    - DATAC (data item choices) tag 506
    - PDFLDC (pull-down field choice) tag 608
    - TOPINST (top instruction) tag 631
- IGC (ideographic characters)
  - formatting
    - BOTINST (bottom instruction) tag 475
- IGC (ideographic characters) *(continued)*
  - formatting *(continued)*
    - DATAC (data item choices) tag 506
    - PDFLDC (pull-down field choice) tag 608
    - TOPINST (top instruction) tag 631
- ILE C/C++ functions
  - I/O requests for subfiles 98
  - mapping to I/O operations 84
- ILE COBOL statement
  - I/O requests for subfiles 98
  - mapping to I/O operations 84
- ILE RPG operation
  - I/O requests for subfiles 98
  - mapping to I/O operations 84
- imbed help (IMHELP) tag
  - example, imbedded help 535
  - overview 534
  - placement in UIM source 400
- imbedding
  - nesting source files 471
- imbedding source files
  - nesting imbeds 471
- imbedding UIM source files 471
- IMHELP (imbed help) tag
  - example, imbedded help 535
  - overview 534
  - placement in UIM source 399, 400
  - UIM source showing panel groups using 400
  - UIM source showing panel groups with help modules 399
- import (IMPORT) tag
  - overview 536
- IMPORT (import) tag
  - overview 536
  - placement in UIM source 400
  - UIM source showing panel groups using 400
- improving
  - productivity with UIM 273
  - system performance with displays 267
- improving productivity with UIM 273
- inactive subfile record
  - definition 90
- incorrect characters in dialog variable 337
- incorrect display characters 337
- incorrect printer characters 337
- INDARA (Indicator Area) keyword 27
- index search
  - *See* online help information
  - F18=More indexes
    - removing access 388
- index search (ISCH) tag
  - parameter list 538
  - placement in UIM source 405
- index search function
  - *See* online help information
- index search hierarchy 406
- Index search key (F11) 403
- index search subtopic (ISCHSUBT) tag
  - parameter list 539
  - placement in UIM source 407
- index search synonym (ISCHSYN) tag
  - parameter list 540
  - placement in UIM source 405
- indicator
  - definition 27
  - option
    - definition 27
    - for display files 27
  - passing information via 27
  - response
    - definition 25
    - for display files 27
  - selection
    - in pull-down menu 155
    - in selection field 152
    - in selection list 162
  - setting off 28
  - setting when data is changed 72
  - types for display files 27
- Indicator Area (INDARA) keyword 27
- INFO (information area) tag
  - overview 537
  - print formatting considerations 538
- information area
  - printed 538
  - scrolling 348
- information area (INFO) tag
  - overview 537
  - print formatting considerations 538
- information area scrolling 348
- information display
  - CUA 460
  - description 429
  - example 429
  - instruction lines 432
  - location information 430
  - online help information
    - DDS considerations 453
    - description 452
    - help areas for 452
  - operating guidelines 432
  - prompt areas 431
  - titles 430
- information, help
  - *See* online help information
- information, location
  - *See* location information, displays
- information, online help
  - *See* online help information
- initial menu
  - definition 237
- Initialize Record (INZRCD) keyword 71
- initializing
  - list display 341
  - output/input fields 67
- initializing list 341
- input data
  - from the display
    - how the system reads 76
    - reading 70
    - reading invited 68
  - handling negative numeric 75
  - inviting
    - from CL programs 68
    - to the display 67
  - keeping 72
  - reading while writing output 76

input data *(continued)*
System/36 application compatibility 700
input field
definition 22
erasing 61
example, display with output and 23
improving performance when sending records with 268
using 22
input file
defining 39
input not waited for, canceling 77
input operation
effect of INZRCD and UNLOCK keywords on 72
input-capable field
limitations on number of 75
input-only file
defining 39
input/output feedback area
common 667
contents 43
definition 25, 43
display 670
input/output operation
acquiring a display station for 42
files shared in a job 85
locking keyboard and positioning cursor 77
mapping to HLL operations
CL commands 84
ILE C/C++ functions 84
ILE COBOL statements 84
ILE RPG operations 84
mapping to I/O operations 84
obtaining information about 43
releasing an acquired display station from 83
requesting
for subfile control record formats 98
for subfile record formats 96
for subfiles 96
input/output request
high-level languages, table of subfile 98
supported by high-level languages 84
inserting
entry from list display 342
HTML tags 207
inserting entry from list 342
instruction line
entry displays 424
information displays 432
list display 434
menu displays, single-choice 422
interaction
between menu bar and application 180
between menu bar and pull-down menu 188
interactive response time for list displays 343
interpretation of command line 350
INVITE (Invite) keyword
system performance considerations 269
use 67
invite operation
CL command for 84
definition 67
ILE C/C++ function for 84
ILE COBOL statement for 84
ILE RPG operation for 84
system performance considerations 269
invited input, reading from the display 68
INZINP (Initialize Input) keyword
performance considerations 268
INZRCD (Initialize Record) keyword
example with UNLOCK keyword 72
use 71
ISCH (index search) tag
parameter list 538
placement in UIM source 405
ISCHSUBT (index search subtopic) tag
parameter list 539
placement in UIM source 407
ISCHSYN (index search synonym) tag
parameter list 540
placement in UIM source 405
item-specific help 187

J

job
close operation for files shared in 86
input/output operation for files shared in 85
open operation for files shared in 85

K

KEEP (Keep) keyword
considerations 255
example DDS source 256
restriction with USRDSPMGT keyword 709
sharing file between programs 255
System/36 application compatibility 700
key
Cancel and Exit, using on menus 235
CAnn and CFnn, defining 25
CFnn (command function), validity considerations 80
command
defining 25
validity considerations 80
command attention (CAnn) 25
command function (CFnn) 25
function
defining 24
System/36 application compatibility 706
key assignments, common
See common key assignments
key definition
example 544
key interaction for menu bars and pull-down menus 188
key list (KEYL) tag
example, key list 545
overview 545
tags allowed table 545
key list item (KEYI) tag
example, key definitions 544
overview 542
keyboard
locking 49
locking during I/O operations 77
unlocking while the program is processing data 71
keyboard function
continued-entry field 176
KEYI (key list item) tag
example, key definitions 544
overview 542
KEYL (key list) tag
example, key list 545
overview 545
tags allowed table 545
KEYSHELP dialog command 652
keyword, DDS 50, 64, 100, 255
Alternative Help (ALTHELP) 26
ALTHELP (Alternative Help) 26
ALTPAGEDWN (Alternative Page Down) 26
ALTPAGEUP (Alternative Page Up) 26
ALWROL (Allow Roll)
example 56
use 55
ASSUME (Assume)
example DDS source 255, 256
use 255
with DFT keyword 255
CAnn (Command Attention) 25
CFnn (Command Function) 25
CHANGE (Change) 72
CHECK (Check)
AB value 73
description 28
M10 value 73
M11 value 73
ME value 73
MF value 73
VN value 73
VNE value 73
CHRID (Character Identifier) keyword 261
CLEAR (Clear) 25
CLRL (Clear Line) 53
example with *NO value 53
example with OVERLAY 54
system performance considerations 268
coding, more efficient 59
COLOR (Color) 34
COMP (Comparison) 73
CSRINPONLY (Cursor Input Only) 29
CSRLOC (Cursor Location) 66
Cursor Input Only (CSRINPONLY) 29
data, repeatedly entered 60

keyword, DDS *(continued)*
  DFT (Default)
    description 64
    with ASSUME keyword 255
  DFTVAL (Default Value) 64
  Display Attribute (DSPATR)
    adding color to fields with 34
    emphasizing fields with 33
    PC value 66, 106
    table of values for 33, 34
    with ASSUME keyword 255
  Display Right-to-Left (DSPRL) keyword 31
  Display Size (DSPSIZ)
    use 36
  DSPATR (Display Attribute)
    adding to color to fields with 34
    emphasizing fields with 33
    PC value 66, 106
    table of color effects 34
    table of values for 33
    with ASSUME keyword 255
  DSPMOD (Display Mode)
    description 65
    example for subfiles 100
    subfile display for *DS3 101
    subfile display for *DS4 101
  DSPRL (Display Right-to-Left) keyword 31
  DSPSIZ (Display Size)
    specifying to improve performance 267
    use 36
  EDTCDE (Edit Code) 35
  EDTWRD (Edit Word) 35
  ERASE (Erase)
    effect on output operation 50
    example of effects with OVERLAY 50
  Erase Input (ERASEINP)
    system performance considerations 268
    use 61
  ERASEINP (Erase Input)
    system performance considerations 268
    use 61
  ERRMSG (Error Message) 224
  ERRMSGID (Error Message Identifier) 224
  Error Message (ERRMSG) 224
  Error Message Identifier (ERRMSGID) 224
  Error Subfile (ERRSFL)
    description 225
    example DDS source 226
  ERRSFL (Error Subfile)
    description 225
    example DDS source 226
  field
    color (table) 34
    emphasizing (table) 33
  Field Cursor Progression (FLDCSRPRG) 30
  field level 63
  FLDCSRPRG (Field Cursor Progression) 30

keyword, DDS *(continued)*
  function keys, defining with 25
  GETRETAIN (Get Retain) 72
  GRDATR (Grid Line Attribute) 205
  GRDBOX (Grid Box) 205
  GRDCLR (Grid Clear) 205
  GRDLIN (Grid Line) 205
  GRDRCD (Grid Record) 205
  Grid Box (GRDBOX) 205
  Grid Clear (GRDCLR) 205
  Grid Line (GRDLIN) 205
  Grid Line Attribute (GRDATR) 205
  Grid Record (GRDRCD) 205
  H specifications
    See H specifications
  HELP (Help)
    enabling Help key 370
    function key use 25
  Help Area (HLPARA) 371
  Help Boundary (HLPBDY)
    use 381
  Help Command Key (HLPCMDKEY)
    use 383
  Help Exclude (HLPEXCLD) 373
  Help Full (HLPFULL) 373
  Help Panel Group (HLPPNLGRP)
    description 373
    with option indicators 375
  Help Record (HLPRCD)
    description 376
    with option indicators 377
  Help Return (HLPRTN) 384
  Help Search Index (HLPSCHIDX) 373
  Help Sequencing (HLPSEQ) 381
  Help Title (HLPTITLE) 373
  HLPARA (Help Area) 371
  HLPBDY (Help Boundary)
    secondary help 381
    use 381
  HLPCMDKEY (Help Command Key)
    response indicators, sample source with 384
    sample DDS source 383
    use 383
  HLPEXCLD (Help Exclude) 373
  HLPFULL (Help Full) 373
  HLPPNLGRP (Help Panel Group)
    description 373
    option indicators, sample source with 375
    sample DDS source 375
    with option indicators 375
  HLPRCD (Help Record)
    description 376
    option indicators, sample source with 377
    sample DDS source 377
    secondary help 381
    with option indicators 377
  HLPRTN (Help Return) 384
  HLPSCHIDX (Help Search Index) 373
  HLPSEQ (Help Sequencing) 381
  HLPTITLE (Help Title) 373
  HOME (Home) 25
  INDARA (Indicator Area) 27

keyword, DDS *(continued)*
  Indicator Area (INDARA) 27
  Initialize Record (INZRCD)
    use 71
  INVITE (Invite)
    description 67
    system performance considerations 269
  INZINP (Initialize Input), system performance considerations 268
  INZRCD (Initialize Record)
    example with UNLOCK keyword 72
    use 71
  KEEP (Keep)
    description 255
    example DDS source 256
  keywords ignored if display modes are changed 65
  LOCK (Lock) 77
  MDTOFF (Modified Data Tag Off) 62
  MOUBTN (Programmable Mouse Button) 196
  MSGALARM (Message Alarm) 228
  MSGID (Message Identifier) 225
  OVERLAY (Overlay)
    description 49
    effect on output operation 50
    example of effects with ERASE 50
    example with CLRL 54
    system performance considerations 268
  Override Attribute (OVRATR)
    system performance considerations 268
    use 57
  Override Data (OVRDTA) 59
    system performance considerations 268
  OVRATR (Override Attribute)
    system performance considerations 268
    use 57
  OVRDTA (Override Data) 59
    system performance considerations 268
  Page Down (PAGEDOWN) 25
  Page Up (PAGEUP)
    defining 25
  PAGEDOWN (Page Down) 25
  PAGEUP (Page Up) 25
  PASSRCD (Passed Record) 256
  PRINT (Print)
    enabling 37
    for function key 25
    results available (table) 37
  Programmable Mouse Button (MOUBTN) 196
  PSHBTNCHC (Push Button Choice) 167
  PSHBTNFLD (Push Button Field) 167
  Push Button Choice (PSHBTNCHC) 167
  Push Button Field (PSHBTNFLD) 167

- keyword, DDS *(continued)*
  - Put with Explicit Override (PUTOVR)
    - description 57
    - system performance considerations 268
  - Put-Retain (PUTRETAIN)
    - description 62
  - PUTOVR (Put with Explicit Override) 59, 60
    - description 57
    - example 58
    - system performance considerations 268
  - PUTRETAIN (Put-Retain) 63
    - description 62
    - for record level 62
  - RANGE (Range) 73
  - Return Cursor Location (RTNCSRLOC) 66
  - Roll Down (ROLLDOWN)
    - defining 25
    - response indicators for 28
  - Roll Up (ROLLUP)
    - example with SFLSIZ and SFLPAG 102
    - for function key 25
    - improving subfile performance 267
    - response indicators for 28
  - ROLLDOWN (Roll Down)
    - defining 25
    - response indicators for 28
  - ROLLUP (Roll Up)
    - example with SFLSIZ and SFLPAG 102
    - for function key 25
    - improving subfile performance 267
    - response indicators for 28
  - RTNCSRLOC (Return Cursor Location) 66
  - Set Off (SETOF or SETOFF) 28
  - SETOF or SETOFF (Set Off) 28
  - SFL (Subfile) 89
  - SFLCHCCTL (Subfile Choice Control) 161
  - SFLCLR (Subfile Clear) 90
  - SFLCSRPRG (Subfile Cursor Progression) 30
  - SFLCSRRRN (Subfile Cursor Relative Record Number) 67
  - SFLCTL (Subfile Control) 89
  - SFLDLT (Subfile Delete) 90
  - SFLDROP (Subfile Drop) 90
  - SFLDSP (Subfile Display) 89
  - SFLDSPCTL (Subfile Display Control) 90
  - SFLEND (Subfile End) 90
  - SFLENTER (Subfile Enter) 90
  - SFLFOLD (Subfile Fold) 90
  - SFLINZ (Subfile Initialize) 90
  - SFLLIN (Subfile Line) 90
  - SFLMODE (Subfile Mode) 67
  - SFLMSG (Subfile Message)
    - defining 90
    - description 90

- keyword, DDS *(continued)*
  - SFLMSG (Subfile Message) *(continued)*
    - displaying on message line when subfile control record written 224
  - SFLMSGID (Subfile Message Identifier)
    - defining 90
    - description 90
    - displaying on message line when subfile control record written 224
    - sample DDS 227
  - SFLMSGKEY (Subfile Message Key)
    - description 90
    - displaying messages from a program message queue 225
  - SFLMSGRCD (Subfile Message Record)
    - description 90
    - displaying messages from a program message queue 225
  - SFLNXTCHG (Subfile Next Changed)
    - description 90
    - example 104
  - SFLPAG (Subfile Page)
    - example with ROLLUP and SFLSIZ 102
    - use in subfile control record format 89
  - SFLPGMQ (Subfile Program Message Queue)
    - description 90
    - displaying messages from a program message queue 225
  - SFLRCDNBR (Subfile Record Number)
    - description 90
    - improving subfile performance 267
  - SFLRNA (Subfile Records Not Active) 90
  - SFLROLVAL (Subfile Roll Value) 90
  - SFLRTNSEL (Subfile Return Selected Choice) 161
  - SFLSCROLL (Subfile Scroll) 161
  - SFLSIZ (Subfile Size)
    - example with ROLLUP and SFLPAG 102
    - use in subfile control record format 89
  - SFLSNGCHC (Subfile Single-Choice Selection) 160
  - SLNO (Starting Line Number)
    - example with *VAR value 51
    - use 50
  - Starting Line Number (SLNO)
    - example with *VAR value 51
    - use 50
  - Subfile Choice Control (SFLCHCCTL) 161
  - Subfile Clear (SFLCLR) 90
  - Subfile Control (SFLCTL) 89
  - Subfile Cursor Progression (SFLCSRPRG) 30
  - Subfile Cursor Relative Record Number (SFLCSRRRN) 67

- keyword, DDS *(continued)*
  - Subfile Delete (SFLDLT) 90
  - Subfile Display (SFLDSP) 89
  - Subfile Display Control (SFLDSPCTL) 90
  - Subfile Drop (SFLDROP) 90
  - Subfile End (SFLEND) 90
  - Subfile Enter (SFLENTER) 90
  - Subfile Fold (SFLFOLD) 90
  - Subfile Initialize (SFLINZ) 90
  - Subfile Line (SFLLIN) 90
  - Subfile Message (SFLMSG)
    - defining 90
    - description 90
    - displaying on message line when subfile control record written 224
  - Subfile Message Identifier (SFLMSGID)
    - defining 90
    - description 90
    - displaying on message line when subfile control record written 224
    - sample DDS 227
  - Subfile Message Key (SFLMSGKEY)
    - description 90
    - displaying messages from a program message queue 225
  - Subfile Message Record (SFLMSGRCD)
    - description 90
    - displaying messages from a program message queue 225
  - Subfile Mode (SFLMODE) 67
  - Subfile Next Changed (SFLNXTCHG)
    - description 90
    - example 104
  - Subfile Page (SFLPAG)
    - example with ROLLUP and SFLSIZ 102
    - use in subfile control record format 89
  - Subfile Program Message Queue (SFLPGMQ)
    - description 90
    - displaying messages from a program message queue 225
  - Subfile Record Number (SFLRCDNBR)
    - description 90
    - improving subfile performance 267
  - Subfile Records Not Active (SFLRNA) 90
  - Subfile Return Selected Choice (SFLRTNSEL) 161
  - Subfile Roll Value (SFLROLVAL) 90
  - Subfile Scroll (SFLSCROLL) 161
  - Subfile Single-Choice Selection (SFLSNGCHC) 160
  - Subfile Size (SFLSIZ)
    - example with ROLLUP and SFLPAG 102
    - use in subfile control record format 89

keyword, DDS *(continued)*
UNLOCK (Unlock)
effects on input operation 72
example with INZRCD keyword 72
unlocking while program is processing data 71
USRDFN (User-Defined) 251
validity-checking 73
VALNUM 73
VALUES (Values) 73
Window Title (WDWTITLE) 122
Word Wrap (WRDWRAP) 32, 175
Keyword, DDS
HTML (Hypertext Markup Language)
resolving field overlap 208
restrictions 211
use 207

L

layout
title lines table 615
layout and positions summary table 578, 579
left-to-right formatting on a left-to-right panel
example 621
left-to-right formatting on a right-to-left panel
example 621
level identifier
changes in 20
definition of 20
determining 20
library
QUSRSYS library 263
QUSRTOOL
definition 7
display examples 415
panel groups, using word processing functions when creating 370
limitations
panel formatter 354
line
clearing a specified number of 53
rolling data between 55
line, instruction
See instruction line
line, options 434
LINES (unformatted lines) tag
example, unformatted lines 546
overview 546
use in help module 402
LINK (hypertext link) tag
CHKPGM (condition evaluation program) 550
conditional expressions
CHKOBJ built-in function 548
CHKUSRCLS (check user class) built-in function 550
example, hypertext link 550
overview 547
placement in UIM source 410
UIM source showing panel groups using 410
list
adding entries, 342
displaying 341
incomplete processing 342
initializing 341
inserting entry from 342
key 545
managing 340
ordered 591
parameter 602
part 579
removing entry from 342
simple 622
unordered 633
updating 342
LIST (list area) tag
example
dynamic list column heading formatting 560
list area 557
list area with list group columns 559
list area with three layout columns 558
overview 552
print formatting considerations 556
tags allowed table 553
list action (LISTACT) tag
example, list actions 567
overview 562
recommended conventions 567
required conventions 566
list area
example
with list group columns 559
with three layout columns 558
help 386
printed 556
scrolling 347
list area (LIST) tag
example
dynamic list column heading formatting 560
list area 557
list area with list group columns 559
list area with three layout columns 558
overview 552
print formatting considerations 556
tags allowed table 553
list area scrolling 347
list column (LISTCOL) tag
formatting considerations 572
overview 568
list column group (LISTGRP) tag
overview 575
list defining in UIM 341
list definition (LISTDEF) tag
emphasis values table 574
overview 573
list display
changing list control field 438
column headings 435
description 432
error condition rules 441
list display *(continued)*
example
CUA 460
extended action entry area 432
extended action entry area 435
format in an empty situation 442
instruction line 434
list entries on 343
list fields 435
list format, empty list 442
list one selection item 442
online help information
DDS considerations 454
description 453
help areas for 453
types 453
operating
extended action entry area 438
guidelines 437
list control field 437
lists when options are specified 440
options line 434
paging location information 436
positioning
lists 437
lowercase names in a list 438
rules for cursor 441
prompt areas 433
response time for 343
rules, error condition 441
titles 433
list entries on a list display 343
list entry handle 342
list entry pointer 342
list fields on list displays 435
list panel
application programming 298
creating 287
source 290
list part (LP) tag
example, list part 580
overview 579
use in help module 402
list view (LISTVIEW) tag
layout and positions summary table 578, 579
overview 578
LISTACT (list action) tag
example, list actions 567
overview 562
recommended conventions 567
required conventions 566
LISTCOL (list column) tag
formatting considerations 572
overview 568
LISTDEF (list definition) tag
emphasis values table 574
overview 573
LISTGRP (list column group) tag
overview 575
LISTVIEW (list view) tag
layout and positions summary table 578, 579
overview 578
literal, graphic 471
location information, displays 430

location of window 118, 119
LOCK (Lock) keyword 77
locked display file 41
locking keyboard and positioning cursor 77
LP (list part) tag
  example, list part 580
  overview 579
  use in help module 402

M

Main Menu display 235
major/minor return code
  major return code definitions
    00 231
    02 231
    03 231
    04 231
    08 231
    11 231
    34 231
    80 231
    81 231
    82 231
    83 231
  recovering from errors
    completion with exceptions 231
    device or session error on open or acquire operation 233
    normal completion 231
    permanent device or session error on I/O operation 232
    permanent system or file errors 232
    recoverable device or session errors on I/O operation 233
  understanding 230
managing
  panel functions 345
managing list 340
managing panel functions 345
markup separator 465
MBAR (menu bar) tag
  example, menu bar 581
  overview 580
  tags allowed table 580
MBARC (menu bar choice) tag
  overview 583
  tags allowed table 583
MDT (modified data tag)
  character translation 262
  definition 23
  resetting 62
  with ASSUME keyword 255
MDTOFF (Modified Data Tag Off) keyword 62
member
  source
    definition 4
MENU (menu area) tag
  example
    menu area with groups 586
    simple menu area 585
  tags allowed table 584
menu area
  help 389
menu area *(continued)*
  scrolling 347
  with groups
    example 586
menu area (MENU) tag
  example
    menu area with groups 586
    simple menu area 585
  tags allowed table 584
menu bar
  area help 394
  creating 144
  definition 144
  example 144, 190
  interaction with application 180
  interaction with pull-down menu 188
  using 357
menu bar (MBAR) tag
  example, menu bar 581
  overview 580
  tags allowed table 580
menu bar choice (MBARC) tag
  overview 583
  tags allowed table 583
Menu Bar Cursor Action (ACTIONS) dialog command 644
menu bar panel
  application programming 332
  creating 320
  source 323
menu definition 276
MENU dialog command 653
menu display 446
  See also application display
  Cancel key, using on 235
  changing attributes 249
  choosing which to display at sign-on time 237
  creating UIM 280
  CUA 457
  deleting 249
  description 421
  determining the previous menu 235
  display file
    building 240
    creating for 242
    creating menu object for 243
    creating message file for 243
    DDS considerations 238
    DDS, specifying 241
    defining 237
    definition 235
    help for 237
    menu and help information, describing 241
    message file, adding messages to 243
    message file, describing menu actions in 239
    naming help formats for 239
    parts of 237
  displaying attributes 249
  example
    display file menu 241
    message file, adding messages to 243
menu display *(continued)*
  example *(continued)*
    message file, describing menu actions in 239
    naming help formats in message file 239
    program menu 245
    program menu, CL source for 245
    RTNPNT(*NO) for program menu, CL source for 247
    sample single-choice 421
    using options and GO command 235
  Exit key, using on 235
  function key area 421
  identifier fields 443
  initial 237
  instruction line 422
  menu displays, single-choice 422, 423
  menu selection entry field 423
  menu stack 235
  naming conflicts, avoiding
    changing command default after command duplication 249
    generic menu specifications 248
    naming, deciding how to 248
    placement in library list 248
    specifying library 248
    ways 248
  online help information
    DDS considerations 451
    description 450
    help areas for 450
    types 450
  operating guidelines 423
  optional 443
  options 422
  program
    building 244
    creating CL program for 246
    creating display file 245
    creating menu for 246
    defining 244
    definition 235
    describing 245
    entering CL source for 245
    exiting without returning to previous menu 246
    passing parameters for 244
  returning to after running the GO command 235
  selection entry field 423
  system, running 235
  user interface manager (UIM) 280
  user-defined
    running 235
    types 235
menu group (MENUGRP) tag
  tags allowed table 587
menu group (MNUGRP) tag
  overview 587
menu item (MENUI or MI) tag
  overview 588
MENU object type 235
menu object, defining
  *DSPF menu 240
  *UIM menu 280

menu-bar choice
  defining 145
  translation of 146
menu-bar choice help
  example 185
Menu-Bar Display (MNUBARDSP) keyword
  application record 180, 190
  menu-bar record 181, 193
menu-bar help list
  example 187
menu-bar record
  MNUBARDSP 193
menu-bar separator
  creating 144
  defining 145
  definition 144
  suppressing 145
menu-bar switch key
  defining 157
  definition 157
MENUGRP (menu group) tag
  overview 587
  tags allowed table 587
MENUI or MI (menu item) tag
  overview 588
message
  automatically handling error 228
  available functions for display files 223
  CPF4001-40FF 230
  CPF4101-43FF 230
  CPF4401-44FF 230
  CPF4501-46FF 230
  CPF4701-48FF 230
  CPF4901-49FF 230
  CPF5001-50FF 230
  CPF5101-53FF 230
  CPF5501-56FF 230
  creating and displaying 223
  description 454
  displaying
    in field 225
    in program message queue 225
    message line for 224
    on message line 224
    on message line when subfile control record written 224
    priorities for 224
    through subfile 225
  error
    displaying from subfiles 105
  examples
    ERRSFL keyword 226
    SFLMSGID keyword 227
  guidelines 455
  line relationship, CUA 457
  reasons for 223
  sounding alarm for 228
  system
    alert 229
    analyzing 229
    detecting 229
    diagnostic 229
    escape 229
    file status code 229
    message queues 229
message *(continued)*
  system *(continued)*
    notifying 229
    permanent 229
    reporting 229
    status 229
  System/36 application compatibility 703
  understanding 229
  window 119
Message Alarm (MSGALARM) keyword 228
message description 28
message field to display message on message line 224
message file
  adding messages to 243
  creating 243
  describing menu action in 239
message line
  definition 23
  scrolling 346
  use 23
Message Location (MSGLOC) keyword 120
message subfile 105
message text symbol 469
message, monitor
  See message
messages displayed on message line priorities 224
mixed display
  example 432
mnemonic
  coding, not valid 170
  defining 170
  definition 139
  valid coding for 170
MNUBARDSP (Menu-Bar Display) keyword
  application record 180, 190
  menu-bar record 181, 193
mode, display
  changing 100
  definition 100
  displaying subfiles horizontally with 100
modified data tag (MDT)
  character translation 262
  definition 23
  resetting 62
  with ASSUME keyword 255
Modified Data Tag Off (MDTOFF) keyword 62
monitor message
  See message, monitor
MOREKEYS (Display More Function Keys) dialog command 653
MOUBTN (Programmable Mouse Button) keyword 196
mouse button, programmable
  See Programmable Mouse Button (MOUBTN) keyword
Move to Top (MOVETOP) dialog command 654
MOVETOP (Move to Top) dialog command 654
moving
  cursor
    CSRINPONLY keyword 29
    using cursor keys 190
    using tab key 189
  window 124
MSG (Display Message) dialog command 654
MSGALARM (Message Alarm) keyword 228
MSGID (Message Identifier) keyword 225
MSGLOC (Message Location) keyword 120
multiple panel foldup when EXIT requested 354
multiple-choice selection field
  colors and display attributes 172
  definition 147
  vertical 150
multiple-choice selection list
  creating 161
  definition 159
  example 159
multiple-selection field (UIM)
  example 521

N

name syntax
  alphabetic conversion 469
  object name
    restrictions 469
  on HELP tag
    restrictions 469
national language considerations and DBCS 408
national language translation
  sentence-ending characters 466
negative numeric input data, handling 75
nesting
  imbeds
    source files 471
No CCSID (NOCCSID) keyword 262
NOCCSID (No CCSID) keyword 262
node (hypertext), definition
  See online help information, hypertext
note (NOTE) tag
  use in help module 401
NOTE (note) tag
  use in help module 401
note (NT or NOTE) tag
  overview 590
note (NT) tag
  example, using a note 591
  tags allowed table 590
  use in help module 401
NT (note) tag
  example, using a note 591
  tags allowed table 590
  use in help module 401
NT or NOTE (note) tag
  overview 590
numeric input data, handling negative 75

numeric value
- display form
  - binary 489
  - example values table 489

O

object
- creating 277
- definition 15
- panel group 467

object name
- name syntax
  - restrictions 469

ODP (open data path)
- definition 85
- for improving system performance 268
- sharing for same job 268

OL (ordered list) tag
- example, ordered list 592
- overview 591
- tags allowed table 592
- use in help module 402

online help information
- adding to a display 369
- contextual help 371
- creating a sample display with 3
- description 450
- documents, office
  - comparing with panel groups and records 370
- enabling Help key 370
- entry displays
  - DDS considerations 452
  - description 451
  - help areas for 451
  - types 451
- example
  - building with display file 3
  - command help 411
  - display showing help areas 372
  - HLPCMDKEY keyword 383
  - HLPCMDKEY keyword with response indicators 384
  - HLPPNLGRP keyword 375
  - HLPPNLGRP keyword with option indicators 375
  - HLPRCD keyword 377
  - HLPRCD keyword with option indicators 377
  - imbedded help modules in panel group 400
  - imported help modules 400
  - ISCH tag 405
  - ISCHSYN tag 405
  - LINK tag 410
  - panel groups with help modules 399
  - secondary help 381
- exiting, System/36 application compatibility 707
- extended help 371
- H specifications
  - definition 371
  - entering in DDS source 372
  - order in DDS source 372

online help information *(continued)*
- help areas
  - DDS keywords for 371
  - defining 371
  - panel groups, specifying 373
  - record 371
  - records, specifying for 376
  - required DDS source 371
- Help key
  - enabling 370
  - returning control to program after pressing 383
  - returning control to program without pressing 384
- help text displayed according to cursor position 348
- hypertext
  - creating links 410
  - definition 408
  - designing links 409
  - reference phrase, definition 410
- index search
  - accessing 387
- index search function
  - *See also* InfoSeeker function
  - deciding topics 405
  - function 403
  - root words, choosing for topic 406
  - root words, defining 405
  - root words, including as synonyms 405
  - synonyms for root words, designating 405
  - synonyms, choosing for topic 406
  - topics, defining 405
  - UIM tags for 403
- information display
  - DDS considerations 453
  - description 452
  - help areas for 452
- list display
  - DDS considerations 454
  - description 453
  - help areas for 453
- menu display
  - DDS considerations 451
  - help areas for 450
  - types 450
- overview 386
- panel groups 403, 405
  - adding entries to search index 411
  - command help, assigning 411
  - comparing with documents and records 370
  - creating 410
  - creating hypertext links 410
  - creating search index 411
  - DDS functions for 373
  - deleting 410
  - deleting search index 412
  - designing hypertext links 409
  - documents and records in same file with 375
  - emphasizing and formatting text in help modules 401

online help information *(continued)*
- panel groups *(continued)*
  - entering UIM source for 399
  - full-screen help, indicating 373
  - headings in help modules, adding 401
  - help modules, linking 408
  - help modules, using more than once 400
  - help, specifying for 373
  - highlighting text in help modules 402
  - HLPPNLGRP keyword 375
  - HLPPNLGRP keyword with option indicators 375
  - hypertext 408
  - hypertext reference phrase 410
  - index search function 403
  - index search, enabling 387
  - Index search, enabling 373
  - making lists in help modules 402
  - name of help screen, defining 373
  - option indicators, specifying with 375
  - organizing with help modules 399
  - paragraphs and notes, defining in help modules 401
  - programming keywords and variables, identifying in help modules 402
  - purpose of UIM tags 399
  - records and documents in same file with 375
  - removing entries from search index 412
  - root words, choosing for index search topic 406
  - root words, defining 405
  - root words, including as index search synonyms 405
  - secondary help, excluding as 373
  - structured text in help modules, indicating 402
  - synonyms for root words, designating 405
  - synonyms, choosing for index search topic 406
  - topics, defining 405
  - user-defined, enabling 387
  - uses for 399
  - using a help module contained in different 400
- record, DDS
  - comparing with panel groups and documents 370
  - entering source 378
  - help boundaries 381
  - help group 381
  - help list 379
  - help sublist 381
  - help, defining for 412
  - help, specifying for 376
  - HLPCMDKEY keyword 383
  - HLPCMDKEY keyword with response indicators 384
  - HLPRCD keyword 377

online help information *(continued)*
- record, DDS *(continued)*
  - HLPRCD keyword with option indicators 377
  - option indicators, specifying with 377
  - paging between help displays 379
  - paging through H specifications 379
  - primary help 379
  - restrictions on 379
  - restrictions on secondary help 382
  - secondary help 379, 381
  - secondary help, displaying 381
  - secondary, determining the sequence of 381
  - with documents in same file 378
  - with panel groups in same file 376
- returning control to program after pressing Help key 383
- returning control to program without pressing Help key 384
- user-defined panel group
  - accessing 387

open data path (ODP)
- definition 85
- for improving system performance 268
- sharing for same job 268

open feedback area
- contents 43
- definition 43
- device definition list 664
- individual descriptions 661

open operation
- definition 41
- file resource allocation for 41
- files shared in job 85
- ILE C/C++ function for 84
- ILE COBOL statement for 84
- ILE RPG operation for 84
- obtaining information about 43

opening
- UIM application 335

operating guideline
- entry displays 429
- information displays 432
- list display
  - *See* list display
- menu displays, single choice 423

operator
- conditional 492
- relational 476

option indicator
- example
  - HLPPNLGRP keyword with option indicators 375
  - HLPRCD keyword with option indicators 377
  - MNUBARCHC (Menu Bar Choice) keyword 146
  - secondary help 381
- panel groups, defining with 375
- records, defining with 377
- removing from record area 27

option line (OPTLINE) tag
- overview 593

options line, list displays 434

OPTLINE (option line) tag
- overview 593

ordered list (OL) tag
- example, ordered list 592
- overview 591
- tags allowed table 592
- use in help module 402

organization
- panel outline 468
- UIM menu outline 468

outline, panel
- organization 468
- UIM menu outline 468

outline, UIM menu
- organization 468

output data
- writing and reading input at same time 76
- writing to display 44

output field
- definition 22
- editing 34
- examples
  - displayed in response to request, display with input and 23
- using 22

output file
- defining 39

output operation
- effect of OVERLAY and ERASE on 50
- positioning cursor after 66
- returning cursor position to application 66

output-only file, defining 39

output/input field
- definition 22
- erasing 61
- using 22

OVERLAY (Overlay) keyword
- description 49
- effect on help-list updating 380
- erasing records 50
- example
  - differences between CLRL and 54
- examples
  - with ERASE keyword 50
- system performance considerations 268
- System/36 application compatibility 700

override
- applying when compiling program 216
- attributes 57
- CL programs
  - included in 213
  - QCMDEXC, use of 213
- commands that ignore or restrict 220
- Delete Override (DLTOVR) command 217
- deleting 217
- determining whether or not to use 213

override *(continued)*
- different file types 218
- display file attributes in HLL programs 213
- display file names and attributes in HLL programs 215
- display file names in HLL programs 214
- Display Override (DSPOVR) command 217
- displaying 217
- effect on other jobs 213
- entering 213
- examples
  - applying when compiling program 216
  - attribute of display file 213
  - different display file 214
  - different display file and attributes 215
- file redirection
  - definition 217
  - different file types 218
  - level checking (LVLCHK) 217
  - same file types 217
  - table of defaults for combinations 218
  - valid for input and output 218
- Override with Display File (OVRDSPF) command
  - file attributes 213
  - file names 214
  - file names and attributes 215
- performance considerations 268
- properties of 213
- same file types 217
- when in effect 213
- when to apply 213

Override Attribute (OVRATR) keyword
- system performance considerations 268
- use 57

Override Data (OVRDTA) keyword
- description 59
- system performance considerations 268

Override with Display File (OVRDSPF) command
- compared to CHGDSPF 16
- file attributes 213
- file names 214
- file names and attributes 215

overriding and sharing printer file 365

OVRATR (Override Attribute) keyword
- system performance considerations 268
- use 57

OVRDSPF (Override with Display File) command
- compared to CHGDSPF 16
- file attributes 213
- file names 214
- file names and attributes 215

OVRDTA (Override Data) keyword
- description 59
- system performance considerations 268

P

- P (paragraph) tag
  - example, paragraph tag 594
  - overview 593
  - placement in UIM source 400
  - use in help module 401
- page 420 for code page conversion 485
- page 424 for code page conversion 485
- Page Down (PAGEDOWN) keyword
  - for function key 25
  - specifying alternative 26
- page eject during printing 365
- Page Up (PAGEUP) keyword
  - for function key 25
  - specifying alternative 26
- page, code
  - *See* code page
- PAGEDOWN (Page Down) keyword
  - for function key 25
  - specifying alternative 26
- PAGEDOWN dialog command 655
- PAGEUP (Page Up) keyword
  - for function key 25
  - specifying alternative 26
- PAGEUP dialog command 655
- paging
  - help displays, DDS records 379
  - location information, list displays 436
- panel
  - areas 467
  - definition 276, 467
  - group object 467
  - help 467
  - organization 467
  - outline 467
  - UIM menu outline 467
- PANEL (display panel) tag
  - example, panel definition 601
  - overview 595
  - tags allowed table 595
- panel contents definition 353
- panel formatter limitations 354
- panel formatting
  - application control 353
  - concepts 352
  - definition 353
- panel group
  - *See also* tag, UIM
  - accessing index search 387
  - accessing user-defined 387
  - adding UIM tags for index search function 403
  - command help, assigning 411
  - comparing with documents and records 370
  - creating 410
  - definition 3, 276
  - deleting 410
  - entering UIM source for 399
  - examples
    - command help 411
    - help modules 399
    - HLPPNLGRP keyword 375
    - HLPPNLGRP keyword with option indicators 375
    - imbedded help modules 400

- panel group *(continued)*
  - examples *(continued)*
    - imported help modules 400
    - ISCH tag 405
    - ISCHSYN tag 405
    - LINK tag 410
  - heading, adding in help modules 401
  - help modules
    - emphasizing and formatting text 401
    - highlighting text 402
    - hypertext 408
    - linking 408
    - using more than once 400
    - using ones contained in different panel group 400
  - help, specifying for 373
  - hypertext, definition 408
  - index search function 405
    - function 403
    - root words, choosing for topic 406
    - root words, including as synonyms 405
    - synonyms for root words, designating 405
    - synonyms, choosing for topic 406
    - topics, deciding 405
  - making lists in help modules 402
  - organizing with help modules 399
  - paragraphs and notes, defining in help modules 401
  - programming keywords and variables, identifying 402
  - search index
    - adding entries to 411
    - creating 411
    - deleting 412
    - removing entries from a search index 412
  - structured text in help modules, indicating 402
  - topics, defining 405
  - uses for 399
- panel group (PNLGRP) tag
  - overview 609
  - placement in UIM source 399
  - tags allowed table 610
- panel group definition language
  - imbedding source files 471
  - markup/content separator 465
  - national language translation
    - sentence-ending characters 466
  - overview 466
- panel group element 277
- panel group object
  - creating 287
  - definition 467
- panel list foldup 355
- panel markup tag
  - *See also* tag
  - definition 401
- paragraph (P) tag
  - example, paragraph tag 594
  - overview 593
  - placement in UIM source 400
  - use in help module 401

- paragraph continuation (PC) tag
  - example, paragraph continuation 603
  - overview 603
  - use in help module 401
- parameter
  - data queue (DTAQ) 259
  - DTAQ (data queue) 259
  - passed for program menus 244
  - restore display (RSTDSP)
    - example 78
  - return point (RTNPNT) parameter
    - for display file menus 235
    - for program menus 235
  - RSTDSP (restore display)
    - example 78
  - RTNPNT (return point) parameter
    - for display file menus 235
    - for program menus 235, 246
  - SHARE (share) 268
- parameter list (PARML) tag
  - example, parameter list 603
  - overview 602
  - tags allowed table 602
  - use in help module 402
- PARML (parameter list) tag
  - example, parameter list 603
  - overview 602
  - tags allowed table 602
  - use in help module 402
- Passed Record (PASSRCD) keyword 256
- passing data
  - between routing steps 256
  - in the same routing step 255
- PASSRCD (Passed Record) keyword
  - example DDS source 256
  - use 256
- PC (paragraph continuation) tag
  - example, paragraph continuation 603
  - overview 603
  - use in help module 401
- PDACCEL (pull-down accelerator) tag
  - overview 604
- PDFLD (pull-down field) tag
  - overview 604
  - tags allowed table 605
- PDFLDC (pull-down field choice) tag
  - overview 605
  - recommended conventions 608
  - required conventions 608
- PDM (programming development manager)
  - definition 4
- performance 268
  - command attention keys versus command function keys 269
  - DDS windows 269
  - deferring write operation 267
  - designating primary screen size 267
  - ERASEINP keyword, considerations with 268
  - invite operation 269
  - INZINP keyword, considerations with 268
  - overlapping but not deleting records 268
  - OVERLAY keyword, considerations with 268

- performance *(continued)*
  - override commands 268
  - PUTOVR, OVRDTA and OVRATR keywords, considerations with 268
  - restoring display 268
  - RSTDSP (restore display)
    - keywords that require 268
    - system performance considerations 268
  - sending records with input fields 268
  - sharing open data paths for same job 268
  - subfile use 267
  - ways to improve using displays 267
  - window application 126
- period (.) symbol 469
- PK (programming keyword) tag
  - overview 609
  - use in help module 402
- placement of window 118, 119
- PNLGRP (panel group) tag
  - overview 609
  - placement in UIM source 399
  - tags allowed table 610
  - UIM source showing panel groups with help modules 399
- pointer device event
  - processing priority
    - other pointer device 205
    - shifted left button pressed 202
    - shifted left button released 203
    - shifted right button pressed 204
    - unshifted left button double click 202
    - unshifted left button pressed 200
    - unshifted left button released 202
  - processing states 199
  - programmable mouse button 197
- pop-up window
  - adding 360
  - adding over another panel 356
  - adding to display stack 360
  - help information 530
  - removing 360
- pop-up window and pull-down menu differences 358
- position of window 118, 119
- possible choices information, entry displays 427
- primary help, definition 379
- PRINT (Print Display) dialog command 656
- PRINT (Print) keyword
  - enabling 37
  - for function key 25
  - results available (table) 37
- Print Display (PRINT) dialog command 656
- print head panel (PRTHEAD) tag
  - example, print title lines 617
  - layout of title lines tables 615, 616
  - overview 613
  - tags allowed table 613
- print head panel printing 364
- Print key
  - enabling 37
  - results available (table) 37
- print panel (PRTPNL) tag
  - overview 618
  - tags allowed table 618
- print panel printing 364
- print title lines
  - example 617
- print trailer message (PRTTRAIL) tag
  - example, trailer message 619
  - overview 619
- printer characters, incorrect 337
- printer file sharing and overriding 365
- PRINTER parameter 37
- printing
  - concepts 362
  - display 37
  - double-byte character set (DBCS) considerations 365
  - information area 538
  - introduction 362
  - list area 556
  - page eject 365
  - print head panel 364
  - print panel 364
  - trailer 364
- priorities for displaying messages on message line 224
- Problem Handling menu 236
- productivity gains with UIM 273
- productivity improvements with UIM 273
- program menu
  - *See also* menu display
  - definition 235
  - return code values, table of 244
- program-described data
  - definition 17
  - description 17
  - example for assigning display space 38
  - program versus system view 17
  - using 38
- program-to-system field
  - understanding how to use 23
  - with DSPATR keyword 33
- program, CL
  - *See* CL programs
- program, IBM-supplied
  - QRCVDTAQ (receive data queue) 259
  - QSNDDTAQ (send data queue) 259
- Programmable Mouse Button (MOUBTN) keyword
  - AID codes to be returned 197
  - benefits 198
  - event processing priority
    - other pointer device 205
    - shifted left button pressed 202
    - shifted left button released 203
    - shifted right button pressed 204
    - unshifted left button double click 202
    - unshifted left button pressed 200
    - unshifted left button released 202
  - event processing states 199
  - NWS considerations 199
  - operation 198
  - overview 196
- Programmable Mouse Button (MOUBTN) keyword *(continued)*
  - pointer device event 197
- programming development manager (PDM)
  - definition 4
- programming example
  - window use 128
- programming keyword (PK) tag
  - overview 609
  - use in help module 402
- Programming menu 236
- programming variable (PV) tag
  - overview 619
  - use in help module 402
- prolog area
  - defining 364
- prolog section
  - panel group 468
- prompt area
  - entry displays 425
  - list display 433
- PROMPT dialog command 656
- prompt, field
  - *See* field prompts
- prompting
  - action list option 657
  - command 657
  - entry field 657
- protecting entry field 31
- PRTHEAD (print head panel) tag
  - example, print title lines 617
  - layout of title lines tables 615, 616
  - overview 613
  - tags allowed table 613
- PRTPNL (print panel) tag
  - overview 618
  - tags allowed table 618
- PRTTRAIL (print trailer message) tag
  - example, trailer message 619
  - overview 619
- PSHBTNCHC (Push Button Choice) keyword 167
- PSHBTNFLD (Push Button Field) keyword 167
- pull-down accelerator (PDACCEL) tag
  - overview 604
- pull-down field (PDFLD) tag
  - overview 604
  - tags allowed table 605
- pull-down field choice (PDFLDC) tag
  - overview 605
  - recommended conventions 608
  - required conventions 608
- pull-down menu
  - CHECK(FE) keyword 154
  - creating 154
  - definition 144
  - example 154, 190
  - interaction with menu bar 188
  - receiving input 182
  - removing after receiving input 184
  - restrict cursor 158
  - single-choice selection field 154
  - updating before displaying 184
- pull-down menu and pop-up window differences 358

PULLDOWN dialog command 658
PULLINPUT (Pull-Down Menu Input) parameter
  example 183
push button
  creating 167
  definition 166
  example, DDS 167
Push Button Choice (PSHBTNCHC) keyword 167
Push Button Field (PSHBTNFLD) keyword 167
Put with Explicit Override (PUTOVR) keyword
  for repeatedly entered data 60
  more efficient coding 59
  sample DDS source 58
put-relative operation for subfiles
  BASIC statement for 98
  definition 96
  ILE C/C++ function for 98
  ILE COBOL statement for 98
  ILE RPG operation for 98
Put-Retain (PUTRETAIN) keyword
  example
    field level 63
    record level 62
  restriction with USRDSPMGT keyword 709
  System/36 application compatibility 700
  using 62
PUTOVR (Put with Explicit Override) keyword
  description 57
  example
    more efficient coding 59
    repeatedly entered data 60
    sample DDS source 58
  system performance considerations 268
  System/36 application compatibility 705
PUTRETAIN (Put-Retain) keyword
  example
    field level 63
    record level 62
  restriction with USRDSPMGT keyword 709
  System/36 application compatibility 700
  using 62
PV (programming variable) tag
  overview 619
  use in help module 402

Q

QRCVDTAQ (receive data queue) program 259
QSNDDTAQ (send data queue) program 259
queue, data
  See DTAQ (data queue) parameter
QUSRSYS library 263
QUSRTOOL library
  definition 7
QUSRTOOL library *(continued)*
  display examples 415
  panel groups, using word processing functions when creating 370

R

radio button
  definition 139
  example 139
read operation
  canceling read operation, CL command for 84
  definition 70
  definition for subfile 98
  subfiles
    BASIC statement 98
    ILE C/C++ function for 98
    ILE COBOL statement for 98
    ILE RPG operation for 98
  with no wait
    CL command for 84
    ILE COBOL statement for 84
    ILE RPG operation for 84
  with wait
    CL command for 84
    ILE C/C++ function for 84
    ILE COBOL statement for 84
    ILE RPG operation for 84
read-from-invited-program-devices operation
  after inviting input 67
  from CL programs, reading-from-invited-devices 70
  ILE C/C++ function for 84
  ILE COBOL statement for 84
  ILE RPG operation for 84
  understanding 69
reading
  data from window 124
receiving
  data 44
receiving input
  from pull-down menu 182
  using PULLINPUT 182
recommended convention
  list action (LISTACT) tag 567
  pull-down field choice (PDFLDC) tag 608
record area, removing option and response indicators from 27
record format 44
  changing on display 48
  checking for changes to 20
  contents 21
  defining display fields and functions in 21
  definition 3
  description 21
  determining active 44
  field, how to use 22
  maximum number for display file 28
  maximum number of fields in 23
  order written to display 49
  output and input, for 44
  overlaying and erasing on display 49
  replacing 48
record format *(continued)*
  replacing on display 48
  starting on specific line 50
  viewing on display 22
record format table, active 44
record-level description 16
  definition 16
  with field-level descriptions 16
  with no field-level descriptions 16
record, DDS
  comparing with panel groups and documents 370
  containing input fields, improving performance with 268
  definition 16
  definition for help 7
  displaying secondary help 381
  displaying two or more 49
  entering for help 378
  erasing from display during overlay 50
  examples
    HLPCMDKEY keyword 383
    HLPCMDKEY keyword with response indicators 384
    HLPRCD keyword 377
    HLPRCD keyword with option indicators 377
    secondary help 381
  help
    defining for 412
    specifying for 376
  improving performance when sending 268
  initializing (diagram) 72
  keeping on display 62
  mode to display 64
  not occupying space on display 47
  overlapping but not deleting to improve system performance 268
  paging between display 379
  placing on display 45
  resetting modified data tags for 62
  restrictions on 379
  restrictions on secondary help 382
  retrieving with WAIT command 70
  rules for displaying on more than one line 45
  secondary help, determining sequence of 381
  use by I/O statements 44
  valid placement of, figure 45
recovering from errors
  See major/minor return code
reference field
  definition 24
relational operator 476
release operation
  definition 83
  ILE C/C++ function for 84
  ILE COBOL statement for 84
  ILE RPG operation for 84
Remove Search Index Entry (RMVSCHIDXE) command 412
Remove Window (RMVWDW) keyword 126

removing
  entry from list 342
  pop-up window 359
  pop-windows using APIs 359
  pull-down menu 184
  window 126, 359
removing access
  F18=More indexes 388
request processor program when displaying panel 361
required convention
  list action (LISTACT) tag 566
  pull-down field choice (PDFLDC) tag 608
resolving
  HTML field overlap 208
resource allocation
  description 41
  effect on display station error handling 41
  understanding how system allocates 41
response indicator
  CAnn key, use with 26
  CFnn key, use with 26
  definition 25
  enabling simultaneously 28
  function keys, use with 25
  HLPCMDKEY keyword, use with 384
  removing from record area 27
  restriction with USRDSPMGT 709
  ROLLDOWN keyword, use with 28
  ROLLUP keyword, use with 28
  SETOF or SETOFF keyword, use with 28
response time
  for list displays 343
  list displays 343
  window application 126
restore display (RSTDSP) parameter
  example 78
  keywords that require 268
  system performance considerations 268
  use 78
restrict cursor
  PULLDOWN keyword 158
  WINDOW keyword 119
restrictions
  5250 engraved keys 346
  command line 350
  name syntax
    HELP tag 469
    object names 469
  on command line 350
  using dialog variables 336
retain command and function keys
  System/36 application compatibility 708
Retain Command Keys (RETCMDKEY) keyword
  System/36 application compatibility 708
Retain Function Keys (RETKEY) keyword
  System/36 application compatibility 708
RETCMDKEY (Retain Command Keys) keyword
  System/36 application compatibility 708
RETKEY (Retain Function Keys) keyword
  System/36 application compatibility 708
RETRIEVE (Retrieve Command String) dialog command 658
Retrieve Command String (RETRIEVE) dialog command 658
RETURN (Return Control to Application) dialog command 659
return code
  binary program menus 244
  display file
    description 677
    major 00 677
    major 02 677
    major 03 678
    major 04 679
    major 08 and 11 679
    major 34 680
    major 80 680
    major 81 683
    major 82 684
    major 83 688
  major/minor
    See major/minor return code
Return Control to Application (RETURN) dialog command 659
Return Cursor Location (RTNCSRLOC) keyword 66
return point (RTNPNT) parameter
  for program menus 246
return-field (RTNFLD) parameter
  description 184
  example 184
returning cursor position within subfile to application 67
returning mode of subfile to application 67
returning to menu after running GO command 235
reverse text (RT) tag
  example
    left-to-right formatting on a left-to-right panel 621
    left-to-right formatting on right-to-left panel 621
  overview 620
right slash (/) symbol 469
right-to-left cursor movement 28
right-to-left display processing 31
RMVSCHIDXE (Remove Search Index Entry) command 412
RMVWDW (Remove Window) keyword 126
Roll Down (ROLLDOWN) keyword
  for function keys 25
Roll Up (ROLLUP) keyword
  example with SFLSIZ and SFLPAG 102
ROLLDOWN (Roll Down) keyword
  function keys 25
  response indicators for 28
ROLLUP (Roll Up) keyword
  example with SFLSIZ and SFLPAG 102
  function keys 25
  improving subfile performance 267
  response indicators for 28
root words for index search
  See online help information
routing step 345
  passing data between programs in same 255
  passing data between steps 256
RSTDSP (restore display) parameter
  example 78
  keywords that require 268
  system performance considerations 268
  use 78
RT (reverse text) tag
  example
    left-to-right formatting on left-to-right panel 621
    left-to-right formatting on right-to-left panel 621
  overview 620
RTNCSRLOC (Return Cursor Location) keyword 66
RTNFLD (return field) parameter
  description 184
  example 184
rule
  edit description use 696
  entry fields 427
  list display
    cursor positioning 441
    error conditions 441
run-time errors, recovering from 86
running
  system and user-defined menus 235

S

sample figure
  example 528
sample heading
  example 639
screen design aid (SDA)
  compared to direct DDS source entry 18
  definition 7
screen size
  alternate 64
  condition names 36
  examples
    DDS for subfiles for two 36
    DDS for two 36
  improving performance with 267
  primary and secondary 36
  valid 36
scroll bar
  creating 164
  definition 159, 163
  example 159
  example, DDS 165
  operation 166
  sizing 165
scrollable area definition 346

scrolling
  area definition 346
  areas that can be scrolled 346
  data item groups 348
  defining 346
  error conditions 347
  function keys 347
  incomplete list 347
  information area 348
  list area 347
  menu area 347
  message line 346
  status information 346
  support 346
  text area 348
SDA (screen design aid)
  compared to direct DDS source entry 18
  definition 7
search index
  See online help information
secondary help
  definition 379
  determining the sequence of records 381
  displaying records 381
selection
  choices, Common User Access 458
  list where singular item can be selected, list displays 442
selection character
  country-designated 345
  highlighting 473, 625
selection field
  example 147
  example, DDS 148
  horizontal 150
    cursor movement 151
  multiple-choice
    definition 147
    vertical 150
  pull-down menu 154
  selection indicator 152
  single-choice
    auto-enter 169
    auto-selection 169
    definition 147
    vertical 149
  vertical
    creating 149, 150
    cursor movement 151
selection field choice
  colors and display attributes 172
selection indicator
  definition 152
  suppressing
    in pull-down menu 155
    in selection field 152
    in selection list 162
selection list
  creating 161
  definition 159
  example, DDS 160
  multiple 161
  selection indicator 162
  single 161
selection list (UIM) 345
self-check, System/36 application compatibility 701
sending
  data 44
sentence-ending characters 466
separating with blank lines 364
separator
  content 465
  markup 465
Set Off (SETOF or SETOFF) keywords
  response indicators for 28
SETOF or SETOFF (Set Off) keyword
  response indicators for 28
SEU (source entry utility)
  definition 5
  Edit display 5
SFL (Subfile) keyword 89
SFLCHCCTL (Subfile Choice Control) keyword 161
SFLCLR (Subfile Clear) keyword
  description 90
SFLCSRPRG (Subfile Cursor Progression) keyword 30
SFLCSRRRN (Subfile Cursor Relative Record Number) keyword 67
SFLCTL (Subfile Control) keyword 89
SFLDLT (Subfile Delete) keyword 90
SFLDROP (Subfile Drop) keyword 90
SFLDSP (Subfile Display) keyword 89
SFLDSPCTL (Subfile Display Control) keyword 90
SFLEND (Subfile End) keyword 90
SFLENTER (Subfile Enter) keyword 90
SFLFOLD (Subfile Fold) keyword 90
SFLINZ (Subfile Initialize) keyword 90
SFLLIN (Subfile Line) keyword 90
SFLMLTCHC (Subfile Multiple-Choice Selection) keyword 160
SFLMODE (Subfile Mode) keyword 67
SFLMSG (Subfile Message) keyword
  description 90
  displaying on message line when subfile control record written 224
  messages resulting from 120
SFLMSGID (Subfile Message Identifier) keyword
  description 90
  displaying on message line when subfile control record written 224
  example DDS source 227
  messages resulting from 120
SFLMSGKEY (Subfile Message Key) keyword
  description 90
  displaying messages from program message queue 225
SFLMSGRCD (Subfile Message Record) keyword
  description 90
  displaying messages from program message queue 225
SFLNXTCHG (Subfile Next Changed) keyword
  description 90
  example DDS 104
SFLPAG (Subfile Page) keyword
  example with ROLLUP and SFLSIZ 102
  use in subfile control record format 89
SFLPGMQ (Subfile Program Message Queue) keyword
  description 90
  displaying messages from program message queue 225
SFLRCDNBR (Subfile Record Number) keyword
  description 90
  improving subfile performance 267
SFLRNA (Subfile Records Not Active) keyword 90
SFLROLVAL (Subfile Roll Value) keyword 90
SFLRTNSEL (Subfile Return Selected Choice) keyword 161
SFLSCROLL (Subfile Scroll) keyword 161
SFLSIZ (Subfile Size) keyword
  example with ROLLUP and SFLPAG 102
  use in subfile control record format 89
SFLSNGCHC (Subfile Single-Choice Selection) keyword
  SFLMLTCHC (Subfile Multiple-Choice Selection) 160
  Subfile Multiple-Choice Selection (SFLMLTCHC) 160
SHARE (share) parameter
  sharing display files in same job 85
  system performance considerations 268
sharing
  open data paths 268
sharing and overriding with printer file 365
sign on
  choosing sign on menu 237
signed numeric data, System/36 application compatibility 705
simple list (SL) tag
  example, simple lists 623
  overview 622
  tags allowed table 622
  use in help module 402
simple menu area
  example 585
single selection field operation, CUA guidelines 458
single-choice menu display
  See menu display
single-choice selection field
  colors and display attributes 172
  definition 147
  horizontal 150
  in a pull-down menu 154
  vertical 149
single-choice selection list
  creating 161
  definition 159
  example 159

size
  window 118
SL (simple list) tag
  example, simple lists 623
  overview 622
  tags allowed table 622
  use in help module 402
slash (/) symbol 469
SLNO (Starting Line Number) keyword
  examples 51
  restrictions 51
  table of values and results 50
  use 50
  used with other DDS keywords 51
  with *VAR value 51
source
  example confirmation panel 300
  example data presentation panel 306
  example list panel 290
  example menu 282
  example menu bar panel 323
source entry utility (SEU)
  definition 5
  Edit display 5
source file
  definition 4
source member
  definition 4
special values and dialog variables 338
Specify Members to Work with display 4
specifying
  enhanced display capability 141
  right-to-left display processing 31
Start Programming Development Manager (STRPDM) command 3
Start Search Index (STRSCHIDX) command 404
Start Source Entry Utility (SEU) display 5
Start Source Entry Utility (STRSEU) command 20
Starting Line Number (SLNO) keyword
  restrictions 51
  table of values and results 50
  use 50
  used with other DDS keywords 51
  with *VAR value 51
stream, user-defined data
  See user-defined data stream (UDDS)
STRPDM (Start Programming Development Manager) command 3
STRSCHIDX (Start Search Index) command 404
STRSEU (Start Source Entry Utility) command 20
subfile
  See also display file
  active record in, updating 96
  adding record at a specified location in 96
  checking validity of data 104
  controlling appearance of 99
  cursor, positioning
    initially 106
    when Fold or Truncate key used 109
subfile *(continued)*
  cursor, positioning *(continued)*
    when more than one record displayed 109
    when Roll key used 106
  definition 22
  describing in DDS source 89
  displaying
    and processing records at the same time 98
    error messages from 105
    horizontally 99
    horizontally with display modes 100
    records 98
    vertically 99
  example
    *DS3 mode display 101
    *DS4 mode display 101
    DDS for two display sizes 36
    DSPATR(PC) keyword, use of 106
    DSPMOD keyword, use of 100
    message subfile 105
    program logic for subfiles 113
    ROLLUP, SFLSIZ, and SFLPAG keywords 102
    sample display for changing information 88
    sample display for display only 87
    sample display for input only 88
    sample display for task combination 88
    sample display with selection 87
    sample displays when rolling two or more records 110
    SFLNXTCHG keyword, use of 104
    SFLPAG, SFLSIZ, and ROLLUP keywords 102
    SFLSIZ, ROLLUP, and SFLPAG keywords 102
    variable-length record 103
    vertical and horizontal, displayed at same time 99
  extending full 93
  functions for 90
  high-level language programs, use in 94
  maximum number active to display station 93
  maximum number allowed in display file 93
  multiple records 87
  optional functions, table of 90
  order of subfile record format and subfile control record format 89
  performance considerations 267
  placing records on display for processing 98
  positioning cursor for each page of records 105
  program logic for 113
  reading active record at specified location in 97
  reading next changed record in 97
  recognizing uses of 87
subfile *(continued)*
    changing information 88
    combination of tasks 88
    display only 87
    input only without validity checking 88
    sample display with selection 87
  records
    active subfile 89
    inactive subfile 90
  requesting I/O operations for 96
  rolling and validity checking 93
  SFL keyword, use of 89
  SFLCTL keyword, use of 89
  SFLDSP keyword, use of 89
  SFLPAG keyword, use of 89
  SFLSCROLL keyword, use of 161
  SFLSIZ keyword, use of 89
  size equal to page size 102
  size not equal to page size 103
  subfile control record format, purpose 89
  subfile control, processing order of DDS keywords for 94
  subfile record format, purpose 89
  table of I/O requests in high-level languages
    BASIC statements 98
    ILE C/C++ functions 98
    ILE COBOL statements 98
    ILE RPG operations 98
  USRDSPMGT keyword restriction 709
  validity checking before rolling 93
  window 120
Subfile Choice Control (SFLCHCCTL) keyword 161
Subfile Clear (SFLCLR) keyword
  description 90
Subfile Control (SFLCTL) keyword 89
subfile control record format
  definition 89
  placement after subfile record format 89
  read operation 98
  requesting I/O operations for 98
  required DDS keywords 89
  SFLCTL keyword, use of 89
  SFLDSP keyword, use of 89
  SFLPAG keyword, use of 89
  SFLSIZ keyword, use of 89
  write operation 98
  write-read operation 98
Subfile Cursor Progression (SFLCSRPRG) keyword 30
Subfile Cursor Relative Record Number (SFLSCRRRN) keyword 67
Subfile Delete (SFLDLT) keyword 90
Subfile Display (SFLDSP) keyword 89
Subfile Display Control (SFLDSPCTL) keyword 90
Subfile Drop (SFLDROP) keyword 90
Subfile End (SFLEND) keyword 90
Subfile Enter (SFLENTER) keyword 90
Subfile Fold (SFLFOLD) keyword 90
Subfile Initialize (SFLINZ) keyword 90
Subfile Line (SFLLIN) keyword 90

Subfile Message (SFLMSG) keyword
  description 90
  displaying on message line when subfile control record written 224
  messages resulting from 120
Subfile Message Identifier (SFLMSGID) keyword 227
  defining in display 225
  description 90
  displaying on message line when subfile control record written 224
  messages resulting from 120
Subfile Message Key (SFLMSGKEY) keyword
  description 90
  displaying messages from program message queue 225
Subfile Message Record (SFLMSGRCD) keyword
  description 90
  displaying messages from program message queue 225
Subfile Mode (SFLMODE) keyword 67
Subfile Multiple-Choice Selection (SFLMLTCHC) keyword 160
Subfile Next Changed (SFLNXTCHG) keyword
  description 90
  example DDS 104
Subfile Page (SFLPAG) keyword
  example with ROLLUP and SFLSIZ 102
  use in subfile control record format 89
Subfile Program Message Queue (SFLPGMQ) keyword
  description 90
  displaying messages from program message queue 225
subfile record format
  definition 89
  get-next-changed operation 97
  get-relative operation 97
  placement before subfile control record format 89
  put-relative operation 96
  requesting I/O operations for 96
  SFL keyword, use of 89
  update operation 96
Subfile Record Number (SFLRCDNBR) keyword
  description 90
Subfile Records not Active (SFLRNA) keyword 90
Subfile Return Selected Choice (SFLRTNSEL) keyword 161
Subfile Roll Value (SFLROLVAL) keyword 90
Subfile Scroll (SFLSCROLL) keyword 161
Subfile Single-Choice Selection (SFLSNGCHC) keyword 160
Subfile Size (SFLSIZ) keyword
  example with ROLLUP and SFLPAG 102
  use in subfile control record format 89
suppressed selection indicator in pull-down menu
  example 156
suppressed selection indicator in selection field
  example 153
symbol
  ampersand (&) 469
  colon (:) 469
  concatenation 469
  message text
    message file default 469
  period (.) 469
  right slash (/) 469
symmetric character
  Arabic 488
  Hebrew 488
synonym, index search
  See online help information
System Command (CMD) dialog command 647
system menu
  See menu display
system message
  See message
system performance
  See performance
System/36 application compatibility
  ASSUME (Assume) keyword 700
  cancel-invite operation 707
  display files 699
  ERASE (Erase) keyword 700
  function keys 706
  HELP (Help) keyword 706
  Help Return (HLPRTN) keyword 706
  high-level languages 699
  HLPRTN (Help Return) keyword 706
  input data 700
  KEEP (Keep) keyword 700
  messages 703
  OVERLAY keyword 700
  PUTOVR (Put with Explicit Override) keyword 705
  PUTRETAIN (Put-Retain) keyword 700

T

tab key
  movement 189
tag
  definition 465
  MENUGRP (menu group) 587
tag language
  See panel group definition language
tag within help module
  CIT (title citation)
    use in help module 402
  definition list (DL)
    use in help module 402
  DL (definition list)
    use in help module 402
  example (XMP)
    use in help module 402
  extended help headings (XH1-XH4)
    use in help module 401

tag within help module *(continued)*
  FIG (figure)
    use in help module 402
  figure (FIG)
    use in help module 402
  H1 through H4 (headings)
    use in help module 401
  heading (H1 through H4)
    use in help module 401
  HELP (help module)
    panel groups with help modules 399
    placement in UIM source 399
  help module (HELP)
    panel groups with help modules 399
    placement in UIM source 399
  highlighted phrase (HP0-HP9)
    use in help module 402
  HP0-HP9 (highlighted phrase)
    use in help module 402
  hypertext link (LINK)
    placement in UIM source 410
  imbed help (IMHELP)
    placement in panel group source 400
  IMHELP (imbed help)
    panel groups using 400
    placement in panel group source 400
  IMPORT (import)
    panel groups using 400
    placement in UIM source 400
  index search (ISCH)
    placement in UIM source 405
  index search synonym (ISCHSYN)
    placement in UIM source 405
  ISCH (index search)
    panel groups using 405
    placement in UIM source 405
  ISCHSYN (index search synonym)
    panel groups using 405
    placement in UIM source 405
  LINES (unformatted lines)
    use in help module 402
  LINK (hypertext link)
    panel groups using 410
    placement in UIM source 410
  list part (LP)
    use in help module 402
  LP (list part)
    use in help module 402
  note (NOTE)
    use in help module 401
  NOTE (note)
    use in help module 401
  note (NT)
    use in help module 401
  NT (note) 401
  OL (ordered list)
    use in help module 402
  ordered list (OL)
    use in help module 402
  P (paragraph)
    placement in UIM source 400
    use in help module 401

- tag within help module *(continued)*
  - panel group (PNLGRP)
    - placement in UIM source 399
  - panel markup
    - definition 401
  - paragraph (P)
    - use in help module 401
  - paragraph continuation (PC)
    - use in help module 401
  - parameter list (PARML)
    - use in help module 402
  - PARML (parameter list)
    - use in help module 402
  - PC (paragraph continuation)
    - use in help module 401
  - PK (programming keyword)
    - use in help module 402
  - PNLGRP (panel group)
    - panel groups with help modules 399
    - placement in UIM source 399
  - programming keyword (PK)
    - use in help module 402
  - programming variable (PV)
    - use in help module 402
  - purpose 399
  - PV (programming variable)
    - use in help module 402
  - simple list (SL)
    - use in help module 402
  - SL (simple list)
    - use in help module 402
  - title citation (CIT)
    - use in help module 402
  - UL (unordered list)
    - use in help module 402
  - unformatted lines (LINES)
    - use in help module 402
  - unordered list (UL)
    - use in help module 402
  - XH1 through XH4 (extended help heading)
    - use in help module 401
  - XMP (example)
    - use in help module 402
- tag, modified data 23
- tag, UIM
  - APPFMT (application formatted area) 472
  - BOTINST (bottom instruction) 474
  - CHECK (validity checking) 475
  - CIT (title citation) 477
  - CLASS (class definition) 478
  - CMDLINE (command line) 491
  - COND (condition definition) 492
  - COPYR (copyright) 496
  - DATA (data presentation area) 496
  - DATAC (data item choices) 506
  - DATACOL (data column) 507
  - DATAGRP (data group) 508
  - DATAI (data item) 510
  - DATAIX (data item extender) 515
  - DATASLT (data selection field) 518
  - DATASLTC (data selection field choice) 523
  - DL (definition list) 525
  - FIG (figure) 527
- tag, UIM *(continued)*
  - H1 through H4 (heading) 533
  - HELP (help module) 529
  - HP0 through HP9 (highlighted phrase) 532
  - IMHELP (imbed help) 534
  - IMPORT (import) 536
  - INFO (information area) 537
  - ISCH (index search) 538
  - ISCHSUBT (index search subtopic) 539
  - ISCHSYN (index search synonym) 540
  - KEYI (key list item) 542
  - KEYL (key list) 545
  - LINES (unformatted lines) 546
  - LINK (hypertext link definition) 547
  - LIST (list area) 552
  - LISTACT (list action) 562
  - LISTCOL (list column) 568
  - LISTDEF (list definition) 573
  - LISTGRP (list column group) 575
  - LISTVIEW (list view) 578
  - LP (list part) 579
  - MBAR (menu bar) 580
  - MBARC (menu bar choice) 583
  - MENU (menu area) 584
  - MENUI or MI (menu item) 588
  - NT or NOTE (note) 590
  - OL (ordered list) 591
  - OPTLINE (option line) 593
  - P (paragraph) 593
  - PANEL (display panel) 595
  - PARML (parameter list) 602
  - PC (paragraph continuation) 603
  - PDACCEL (pull-down accelerator) 604
  - PDFLD (pull-down field) 604
  - PDFLDC (pull-down field choice) 605
  - PK (programming keyword) 609
  - PNLGRP (panel group) 609
  - PRTHEAD (print head panel) 613
  - PRTPNL (print panel) 618
  - PRTTRAIL (print trailer message) 619
  - PV (programming variable) 619
  - RT (reverse text) 620
  - SL (simple list) 622
  - TEXT (text area) 623
  - TI (translation list item) 628
  - TL (translation list) 629
  - TOPINST (top instruction) 630
  - TT (truth table) 631
  - TTROW (truth table row) 632
  - UL (unordered list) 633
  - VAR (variable definition) 634
  - VARRCD (variable record definition) 637
  - XH1 through XH4 (extended help headings) 638
  - XMP (example) 639
- text (TEXT area) tag
  - overview 623
- text area
  - scrolling 348
- text area (TEXT) tag
  - example
    - a text area 627
  - text data 625
- text area scrolling 348
- text area, descriptive
  - entry displays 427
- text data
  - selection characters 625
- TI (translation list item) tag
  - overview 628
- time value
  - display form 490
- timing function on WAITRCD parameter 69
- tips
  - Word Wrap (WRDWRAP) keyword 33
- title
  - entry displays 424
  - information displays 430
  - list display 433
  - single-choice menu displays 422
  - window 122
- title citation (CIT) tag
  - example
    - title citations 477
  - overview 477
  - use in help module 402
- title lines
  - table 615
- TL (translation list) tag
  - list, translation 630
  - overview 629
  - tags allowed table 629
- tool, command line 446
- top instruction (TOPINST) tag
  - overview 630
- topics, index search
  - *See* online help information
- TOPINST (top instruction) tag
  - overview 630
- trailer message
  - example 619
- trailer printing 364
- translation
  - character set and code page, UIM 484
- translation list (TL) tag
  - list, translation 630
  - overview 629
  - tags allowed table 629
- translation list item (TI) tag
  - overview 628
- translation table 263
- truth table (TT) tag
  - overview 631
- Truth Table (TT) tag
  - example, truth table 632
  - tags allowed table 631
- truth table row (TTROW) tag
  - overview 632
- TT (truth table) tag
  - overview 631
- TT (Truth Table) tag
  - example, truth table 632
  - tags allowed table 631

TTROW (truth table row) tag
  overview 632
two presentation areas for data items
  example 503
two-column format in a data entry panel
  example 501
type
  display station 116, 139

U

UDDS (user-defined data streams)
  3270 Model 4 display station support 253
  available functions 252
  buffer format for 5250 display station 252
  command attention keys, effects on 252
  command function keys, effects on 252
  defining output buffer 252
  defining output data stream 252
  display station differences, understanding 251
  handling of field-level requests after user-defined requests 252
  help information for 252
  I/O requests, effect on 252
  input operations 253
  limitations 252
  opening display files containing 252
  output record area for record format 252
  system operations to request commands 254
  understanding limitations 252
  using 251
  USRDFN (User-Defined) keyword 251
  valid keywords with USRDFN keyword 252
  write-read function 253
  write-read operations
    CL (control language) 253
    ILE COBOL 253
    ILE RPG 253
UIM
  commonly asked questions 365
UIM (user interface manager) 341
  See also panel group
  5250 engraved key restrictions 346
  action list definition 344
  action variable definition 344, 345
  adding pop-up window over another panel 356
  adding window over another panel 356
  after scrolling example 305
  alternate view of list example 288
  application handle definition 335
  application programming
    confirmation processing 303
    data presentation panel 317
    list panel 298
    menu bar panel 332
  application window, defining 359

UIM (user interface manager) *(continued)*
  automatic confirmation processing 303
  command line
    entered too long 351
    interpretation 350
    pop-up window 361
    restrictions 350
    window 361
    window example 361
  confirmation panel source 300
  confirmation processing
    application programming 303
    automatically 303
  contextual help, defining 348
  controlling list entries on list display 343
  conversion of character set and code page 484
  creating
    menu panel 280
    object 277
  cursor position when help text displayed 348
  data entry panel 320
  data presentation panel
    application programming 317
    example 303
    source 306
  DBCS and national language considerations 408
  defining
    application window 359
    contextual help 348
    dialog variable 335
    function keys 351
    function keys and VARUPD value 351
    menu object 280
    panel group object 287
  definition 3
  details 335
  dialog variable
    definition 335
    error messages 337
    incorrect characters 337
    initial value 336
    restrictions 336
  differences between pull-down menus and pop-up windows 358
  display characters, incorrect 337
  display panel field value using dialog variables 337
  displaying list 341
  displaying panel using request processor program 361
  elements within panel group 277
  entry pointer for list 342
  error messages for dialog variables 337
  example
    alternate view of list 288
    list panel 287
    menu bar panel 320
    menu source 282
  exit flag 354

UIM (user interface manager) *(continued)*
  field value for display panel using dialog variables 337
  folding up list panel 355
  folding up multiple panels when EXIT requested 354
  folding when EXIT requested 354
  formatting function keys 351
  function key
    defining 351
    formatting 351
    scrolling 347
    VARUPD value 351
  graphical user interface (GUI) 346
  help
    hierarchy, index search 406
    index search 387
    index search hierarchy 406
    user-defined panel group 387
  help text displayed according to cursor position 348
  help window 122
  incorrect characters in dialog variable 337
  incorrect display characters 337
  incorrect printer characters 337
  index search
    giving help panels access 387
  initial value of dialog variable 336
  initializing list 341
  inserting entry from list 342
  interactive response time for list displays 343
  limitations of panel formatter 354
  list
    displaying 341
    incomplete processing 342
    initialing 341
    inserting entry from 342
    managing 340
    removing entry from 342
    updating 342
  list display
    controlling list entries 343
  list entry
    handle 342
    on list display 343
    pointer 342
  list panel
    application programming 298
    example 287
    fold up 355
    source 290
  managing panel functions 345
  menu area, scrolling 347
  menu bar panel
    application programming 332
    example 320
    source 323
  menu bars 357
  menu definition 276
  menu object 280
  menu panel
    creating 280
    example 280
  message line, scrolling 346

UIM (user interface manager) *(continued)*
national language considerations and DBCS 408
opening application 335
panel formatter limitations 354
panel formatting
application control 353
concepts 352
definition 353
when to perform 353
panel function managing 345
panel group
elements 277
panel group object, defining 287
panel list fold up 355
pop-up window
adding 359, 360
adding and removing using APIs 359
adding over another panel 356
adding to display stack 359, 360
API for adding and removing 359
command line 361
defining application 359
example in command line 361
removing 360
removing using APIs 359
using 359
pop-up window and pull-down menu differences 358
printer characters, incorrect 337
printing
blank lines for separating 364
character set and code page 338
code page and character set 338
concepts 362
DBCS (double-byte character set) 365
double-byte character set (DBCS) 365
fonts and highlighting 364
header area 365
highlighting 364
highlighting and fonts 364
introduction 362
overriding with and sharing printer file 365
page eject 365
print head panel 364
print panel 364
printer file, sharing and overriding with 365
prolog area 364
separating with blank lines 364
sharing and overriding with printer file 365
trailer 364
pull-down menu and pop-up window differences 358
removing entry from list 342
request processor program when displaying panel 361
response time for list displays 343
restrictions
5250 engraved keys 346
using dialog variables 336
routing step 345

UIM (user interface manager) *(continued)*
scrolling 346
area definition 346
areas that can be scrolled 346
data item groups 348
defining scrollable areas 346
error conditions 347
function key 347
incomplete list 347
information area 348
list area 347
menu area 347
message line 346
status information 346
text area 348
selection characters 345
selection list definition 345
source
example confirmation panel 300
example data presentation panel 306
example list panel 290
example menu 282
example menu bar panel 323
updating list 342
user-defined panel group
giving help panels access 387
variable pool services 338
VARUPD value and function keys 351
window
adding 359, 360
adding and removing using APIs 359
adding over another panel 356
adding to display stack 359, 360
API for adding and removing 359
command line 361
defining application 359
example in command line 361
removing 360
removing using APIs 359
using 359
UIM (User Interface Manager)
API services 279
confirmation list panel, creating 299
considerations before using 273
creating
confirmation list panel 299
data presentation panel 303
list panel 287
menu bar panel 320
data presentation panel
creating 303
DDS instead of 273
functions supported 276
increasing user productivity 273
introduction 275
list panel
creating 287
menu bar panel
creating 320
panel elements supported 276
panel group
definition 276
productivity
improvements 273

UIM (User Interface Manager) *(continued)*
productivity *(continued)*
increasing application programmer 273
using DDS instead of 273
when to use to gain productivity 273
UL (unordered list) tag
example, unordered list 634
overview 633
tags allowed table 633
use in help module 402
unavailable choice
example 169
unformatted lines (LINES) tag
example, unformatted lines 546
overview 546
use in help module 402
UNLOCK (Unlock Keyboard) keyword
effects on input operation 72
example with INZRCD keyword 72
unlocking keyboard while the program is processing data 71
unordered list (UL) tag
example, unordered list 634
overview 633
tags allowed table 633
use in help module 402
update operation for subfiles
BASIC statement for 98
definition 96
ILE C/C++ function for 98
ILE COBOL statement for 98
ILE RPG operation for 98
updating
list in UIM 342
updating pull-down menu 184
User Defined (USRDFN) keyword
See user-defined data stream (UDDS)
User Display Management (USRDSPMGT) keyword
DDS keywords that cannot be used with 700
use 699
user interface manager (UIM)
See panel group
5250 engraved key restrictions 346
action list definition 344
action variable definition 344, 345
adding pop-up window over another panel 356
adding window over another panel 356
alternate view of list example 288
application handle definition 335
application programming
confirmation processing 303
data presentation panel 317
list panel 298
menu bar panel 332
application window, defining 359
automatic confirmation processing 303
command line
entered too long 351
interpretation 350
pop-up window 361
restrictions 350

- user interface manager (UIM) *(continued)*
  - command line *(continued)*
    - window 361
    - window example 361
  - commonly asked questions 365
  - confirmation panel source 300
  - confirmation processing
    - application programming 303
    - automatically 303
  - contextual help, defining 348
  - controlling list entries on list display 343
  - conversion of character set and code page 484
  - creating
    - menu panel 280
    - object 277
  - cursor position when help text displayed 348
  - data entry panel 320
  - data presentation panel
    - source 306
  - DBCS and national language considerations 408
  - defining
    - application window 359
    - contextual help 348
    - dialog variable 335
    - function keys 351
    - function keys and VARUPD value 351
    - panel group object 287
  - defining panel group object 287
  - definition 3
  - details 335
  - dialog variable
    - definition 335
    - error messages 337
    - incorrect characters 337
    - initial value 336
    - restrictions 336
  - differences between pull-down menus and pop-up windows 358
  - display characters, incorrect 337
  - display panel field value using dialog variables 337
  - displaying list 341
  - displaying panel using request processor program 361
  - elements within panel group 277
  - enabling graphical user interface (GUI) 346
  - entry pointer for list 342
  - error messages for dialog variables 337
  - example
    - alternate view of list 288
    - list panel 287
    - menu bar panel 320
  - exit flag 354
  - field value for display panel using dialog variables 337
  - folding up list panel 355
  - folding up multiple panels when EXIT requested 354
  - folding when EXIT requested 354
  - formatting function keys 351

- user interface manager (UIM) *(continued)*
  - function key
    - defining 351
    - formatting 351
    - scrolling 347
    - VARUPD value 351
  - graphical user interface (GUI) 346
  - help
    - hierarchy, index search 406
    - index search 387
    - index search hierarchy 406
    - user-defined panel group 387
  - help text displayed according to cursor position 348
  - help window 122
  - incorrect characters in dialog variable 337
  - incorrect display characters 337
  - incorrect printer characters 337
  - index search
    - giving help panels access 387
  - initial value of dialog variable 336
  - initializing list 341
  - inserting entry from list 342
  - interactive response time for list displays 343
  - limitations of panel formatter 354
  - list
    - defining 341
    - displaying 341
    - incomplete processing 342
    - initialing 341
    - inserting entry from 342
    - managing 340, 341
    - removing entry from 342
    - updating 342
  - list display
    - controlling list entries 343
  - list entry
    - handle 342
    - on list display 343
    - pointer 342
  - list panel
    - application programming 298
    - example 287
    - fold up 355
    - source 290
  - managing panel functions 345
  - menu area, scrolling 347
  - menu bar 357
  - menu bar panel 320
    - application programming 332
    - example 320
    - source 323
  - menu panel
    - creating 280
    - example 280
  - message line, scrolling 346
  - national language considerations and DBCS 408
  - opening application 335
  - panel formatter limitations 354
  - panel formatting
    - application control 353
    - concepts 352
    - definition 353
    - when to perform 353

- user interface manager (UIM) *(continued)*
  - panel function managing 345
  - panel group
    - elements 277
  - panel group object, defining 287
  - panel list fold up 355
  - pop-up window
    - adding 359, 360
    - adding and removing using APIs 359
    - adding over another panel 356
    - adding to display stack 359, 360
    - API for adding and removing 359
    - command line 361
    - defining application 359
    - example in command line 361
    - removing 360
    - removing using APIs 359
    - using 359
  - pop-up window and pull-down menu differences 358
  - printer characters, incorrect 337
  - printing
    - blank lines for separating 364
    - character set and code page 338
    - code page and character set 338
    - concepts 362
    - DBCS (double-byte character set) 365
    - double-byte character set (DBCS) 365
    - fonts and highlighting 364
    - header area 365
    - highlighting and fonts 364
    - introduction 362
    - overriding with and sharing printer files 365
    - page eject 365
    - print head panel 364
    - print panel 364
    - printer file, sharing and overriding with 365
    - prolog area 364
    - separating with blank lines 364
    - sharing and overriding with printer files 365
    - trailer 364
  - pull-down menu and pop-up window differences 358
  - removing entry from list 342
  - request processor program when displaying panel 361
  - response time for list displays 343
  - restrictions
    - 5250 engraved keys 346
    - using dialog variables 336
  - routing step 345
  - scrolling 346
    - area definition 346
    - areas that can be scrolled 346
    - data item group 348
    - defining scrollable areas 346
    - error conditions 347
    - function key 347
    - incomplete list 347
    - information area 348
    - list area 347

- user interface manager (UIM) *(continued)*
  - scrolling *(continued)*
    - menu area 347
    - message line 346
    - status information 346
    - text area 348
  - selection characters 345
  - selection list definition 345
  - source
    - example confirmation panel 300
    - example data presentation panel 306
    - example list panel 290
    - example menu bar panel 323
  - updating list 342
  - user-defined panel group
    - giving help panels access 387
  - variable pool services 338
  - VARUPD value and function keys 351
  - window
    - adding 359, 360
    - adding and removing using APIs 359
    - adding over another panel 356
    - adding to display stack 359, 360
    - API for adding and removing 359
    - command line 361
    - defining application 359
    - example in command line 361
    - removing 360
    - removing using APIs 359
    - using 359
- User Interface Manager (UIM)
  - API services 279
  - confirmation list panel, creating 299
  - considerations before using 273
  - creating
    - confirmation list panel 299
    - data presentation panel 303
    - list panel 287
    - menu bar panel 320
  - data presentation panel
    - after scrolling example 305
    - creating 303
    - example 303
  - DDS instead of 273
  - defining
    - menu object 280
  - defining menu object 280
  - example
    - menu source 282
  - functions supported 276
  - increasing user productivity 273
  - introduction 275
  - list panel
    - creating 287
  - menu bar panel
    - creating 320
  - menu definition 276
  - menu object, defining 280
  - panel elements supported 276
  - panel group
    - definition 276
  - productivity
    - improvements 273
- User Interface Manager (UIM) *(continued)*
  - productivity *(continued)*
    - increasing application programmer 273
  - source for example menu 282
  - using DDS instead of 273
  - when to use to gain productivity 273
- User Restore Display (USRRSTDSP) keyword 126
- user-defined data streams (UDDS)
  - 3270 Model 4 display station support 253
  - available functions 252
  - buffer format for 5250 display station 252
  - command attention keys, effects on 252
  - command function keys, effects on 252
  - defining output buffer 252
  - defining output data stream 252
  - display station differences, understanding 251
  - handling of field-level requests after user-defined requests 252
  - help information for 252
  - I/O requests, effect on 252
  - input operations 253
  - limitations 252
  - opening display files containing 252
  - output record area for record format 252
  - system operations to request commands 254
  - understanding limitations 252
  - using 251
  - USRDFN (User-Defined) keyword 251
  - valid keywords with USRDFN keyword 252
  - write-read function 253
  - write-read operations
    - CL (control language) 253
    - ILE RPG 253
- user-defined edit code
  - about 35
  - edit descriptions 695
- user-defined menus
  - *See* menu display
- user-defined panel group
  - *See* online help information
- using note
  - example 591
- USRDFN (User-Defined) keyword
  - *See* user-defined data stream (UDDS)
- USRDSPMGT (User Display Management) keyword
  - restricted DDS keywords 700, 709
  - restrictions for response indicators 709
  - restrictions for subfiles 709
  - use 699
- USRRSTDSP (User Restore Display) keyword 126

V

- valid DDS for mnemonic
  - example 170
- validity checking
  - functions, specifying 73
  - keywords 73
  - performing 336
  - restrictions when using CAnn keys 26
  - subfile data 104
- validity checking (CHECK) tag
  - example, validity checking 477
  - overview 475
- validity-checking keyword 120
- value
  - BASETYPE 479
  - character 490
  - date 490
  - numeric 489
  - time 490
- VAR (variable definition) tag
  - overview 634
- variable definition (VAR) tag
  - overview 634
- variable pool
  - definition 335
  - services 338
- variable record definition (VARRCD) tag
  - overview 637
- variable update (VARUPD) attribute
  - dialog command 643
  - function keys 351
  - KEYI tag 543
  - PDFLDC tag 607
- variable value 337
- variable-length record 103
- VARRCD (variable record definition) tag
  - overview 637
- VARUPD (variable update) attribute
  - dialog command 643
  - function keys 351
  - KEYI tag 543
  - PDFLDC tag 607
- VARUPD value and function keys 351
- vertical
  - multiple-choice selection field
    - cursor movement 151
    - example 150
  - single-choice selection field 149

W

- WAIT (wait) command 70
- WAIT(*NO) parameter 68
- WAITFILE (wait file) parameter
  - for acquire operation 42
  - for open operation 42
  - for system allocation of resources 41
- WAITRCD (wait record) parameter
  - timing function for 69
- WDWBORDER (Window Border) keyword 120
- WDWTITLE (Window Title) keyword 122
- window
  - adding over another panel 356

window *(continued)*
- border 120
- changing 124
- creating 116
- cursor position 119
- defining 359
- definition 38
- duplicating 124
- grid line structure 206
- help displayed in 120, 122
- location 118, 119
- making one pop out of series 125
- making two active 124
- maximum displayed 116
- message in 119
- moving 124
- performance 126
- programming example 128
- reading data from 124
- removing 126
- size 118
- subfile used in 120
- system performance considerations 269
- title 122

WINDOW (Window) keyword 116
Window Border (WDWBORDER) keyword 120
window definition record 117
window reference record 117
window title
- example, DDS 122

Window Title (WDWTITLE) keyword 122
Word Wrap (WRDWRAP) keyword 32, 175
words, index search root
- *See* online help information

Work with Members Using PDM display 4
wrap
- word 32, 175

WRDWRAP (Word Wrap) keyword 32, 175
write operation 98
- CL command for 84
- deferring until a read request is made 64
- definition 44
- ILE C/C++ function for 84
- ILE COBOL statement for 84
- ILE RPG operation for 84
- subfiles
  - BASIC statement for 98
  - definition 98
  - ILE C/C++ function for 98
  - ILE COBOL statement for 98
  - ILE RPG operation for 98

write-read operation
- with no wait, CL command for 84
- with wait
  - CL command for 84
  - ILE C/C++ function for 84
  - ILE RPG operation for 84

write-read operation for subfiles
- definition 98
- ILE C/C++ function for 98

write-read operation for subfiles *(continued)*
- ILE RPG operation for 98

X

XH1 through XH4 (extended help headings) tag
- example, sample heading 639
- formatting rules 639
- overview 638
- use in help module 401

XMP (example) tag
- example, formatting an example 640
- help module 402
- overview 639

Z

Z-variable
- definition 636

Z36ENV dialog variable 636
ZCANCEL dialog variable 636
ZDBCS dialog variable 636
ZDSPSIZ dialog variable 636
ZEXIT dialog variable 636
ZJOB dialog variable 636
ZJOBNBR dialog variable 636
ZLMTCPB dialog variable 636
ZMENU dialog variable 636
ZMNULIB dialog variable 636
ZSYSNAM dialog variable 636
ZUSER dialog variable 636

Printed in USA

SC41-5715-01