

MICRO FOCUS

COBOL™

**DIALOG SYSTEM
CHARACTER MODE USER'S GUIDE**



Copyright © 2001 Micro Focus International Limited.
All rights reserved.

Micro Focus International Limited has made every effort to ensure that this book is correct and accurate, but reserves the right to make changes without notice at its sole discretion at any time. The software described in this document is supplied under a license and may be used or copied only in accordance with the terms of such license, and in particular any warranty of fitness of Micro Focus software products for any particular purpose is expressly excluded and in no event will Micro Focus be liable for any consequential loss.

Animator®, COBOL Workbench®, EnterpriseLink®, Mainframe Express®, Micro Focus®, Net Express®, REQL® and Revolve® are registered trademarks, and AAI™, Analyzer™, Application to Application Interface™, AddPack™, AppTrack™, AssetMiner™, CCI™, Dialog System™, EuroSmart™, FixPack™, LEVEL II COBOL™, License Management Facility™, License Server™, Mainframe Access™, Mainframe Manager™, Micro Focus COBOL™, Object COBOL™, OpenESQL™, Personal COBOL™, Professional COBOL™, Server Express™, Session Recorder™, SmartFind™, SmartFind Plus™, SmartFix™, Toolbox™, VS COBOL™, WebSync™, and Xilerator™ are trademarks of Micro Focus International Limited. All other trademarks are the property of their respective owners.

No part of this publication, with the exception of the software product user documentation contained on a CD-ROM, may be copied, photocopied, reproduced, transmitted, transcribed, or reduced to any electronic medium or machine-readable form without prior written consent of Micro Focus International Limited.

Licensees may duplicate the software product user documentation contained on a CD-ROM, but only to the extent necessary to support the users authorized access to the software under the license agreement. Any reproduction of the documentation, regardless of whether the documentation is reproduced in whole or in part, must be accompanied by this copyright statement in its entirety, without modification.

U.S. GOVERNMENT RESTRICTED RIGHTS. It is acknowledged that the Software and the Documentation were developed at private expense, that no part is in the public domain, and that the Software and Documentation are Commercial Computer Software provided with RESTRICTED RIGHTS under Federal Acquisition Regulations and agency supplements to them. Use, duplication or disclosure by the U.S. Government is subject to restrictions as set forth in subparagraph (c)(1)(ii) of The Rights in Technical Data and Computer Software clause at DFAR 252.227-7013 et. seq. or subparagraphs (c)(1) and (2) of the Commercial Computer Software Restricted Rights at FAR 52.227-19, as applicable. Contractor is Micro Focus, 9420 Key West Avenue, Rockville, Maryland 20850. Rights are reserved under copyright laws of the United States with respect to unpublished portions of the Software.

Table of Contents

Preface	17
Audience	17
Notation in this Manual	17
UNIX Considerations	18
1 Overview	21
1.1 Dialog System	21
1.2 Environments	22
1.3 Program Development with Dialog System	22
2 Getting Started	25
2.1 Invoking Dialog System	25
2.1.1 Status Line	26
2.1.2 Menu System	27
2.1.3 Help Screens	28
2.1.4 UNIX Keyboard Mapping	28
2.2 Key Usage in Dialog System	29
2.2.1 Painting Panels	30
2.3 Using Dialog System from the Command Line	30
3 Sample Session in Character Mode	35
3.1 Creating a Screenset	36
3.1.1 Starting Dialog System	36
3.1.2 Defining A Data Block	37
3.1.3 Saving Your Screenset	39
3.1.4 Loading a Screenset	40
3.1.5 Painting a Panel	41
3.1.6 Moving Fields Around	46
3.1.7 Running Your Screenset	47

3.2 Writing a Related Program	50
3.2.1 The Program	50
3.2.2 Generating the Data Block	51
3.2.3 Running Your Program	52
3.3 Dialog	53
3.3.1 Defining Dialog	53
3.4 Attributes	57
4 Using Dialog System	59
4.1 Main Menu.	59
4.1.1 Help (F1)	59
4.1.2 Data Definition (F2)	59
4.1.3 Panels (F3)	59
4.1.4 Global Dialog (F4)	60
4.1.5 Generate (F5)	60
4.1.6 Run (F6)	60
4.1.7 Print (F7)	60
4.1.8 Tutorial (F8)	60
4.1.9 Alternate (Alt)	61
4.1.10 Escape (Esc)	61
4.2 Main Alt Menu.	61
4.2.1 Help (F1)	61
4.2.2 Initialize (F2)	62
4.2.3 Load Screenset (F3)	62
4.2.3.1 Select File (Enter)	63
4.2.3.2 List Files/List Library Catalogue (F2)	63
4.2.3.3 List Directories (F3)	63
4.2.3.4 Delete File (F4)	64
4.2.3.5 Sort Name (F5)	64
4.2.3.6 Sort Time (F6)	64
4.2.3.7 Unsort (F7)	64
4.2.3.8 List Ascending/Descending (F8)	64
4.2.3.9 Drive (F9)	64
4.2.3.10 Control Menu (Ctrl)	65
4.2.3.11 Escape (Esc)	65
4.2.4 Save Screenset (F4)	66

4.2.5 Screenset Switches (F5)	66
4.2.5.1 Normal / Micro Focus Panels (F2)	66
4.2.5.2 Global / Local Dialog First (F3)	67
4.2.5.3 Key Translation Off / On (F4)	67
4.2.6 Normal / Top / Bottom Coordinates (F6)	68
4.2.7 Set Details (F7).	68
4.2.8 Colorize (F8)	69
4.2.8.1 Off / On (F2).	69
4.2.8.2 Map Screenset Colors Map to COBOL System (F3-F9)	70
4.2.8.3 More (F10)	70
4.2.9 Import (F9).	70
4.2.9.1 Syntax Checking.	71
4.2.9.2 Semantic Checking	72
4.2.10 Export (F10)	72
4.2.10.1 Panel Components (F2)	74
4.2.10.2 Select All Entries (F4).	74
4.2.10.3 Unselect All Entries (F5).	74
4.2.10.4 Select / Unselect Entry (Space Bar)	74
4.2.10.5 Panel List (Enter)	75
4.2.10.6 Select All Panels (F4)	75
4.2.10.7 Unselect All Panels (F5)	76
4.2.10.8 Select / Unselect Panel (Space Bar)	76
4.2.10.9 Export Details (Enter)	76
4.2.10.10 Import Limitations.	76
5 Data Definition	79
5.1 Data Field Definition (F2).	80
5.1.1 Amend Group Size (F2).	82
5.1.2 Insert Field (F3)	83
5.1.3 Delete Field (F4)	83
5.1.4 Amend Group Repeats (F5)	84
5.1.5 Define Group (F6)	84
5.1.6 Undefine Group (F7).	85
5.1.7 Define Validation Details (F8)	85
5.1.7.1 Delete All Validations (F2)	87
5.1.7.2 Range / Table Validation (F3)	87
5.1.7.3 Check Digit Validation (F4).	91
5.1.7.4 Date Validation (F5)	94

5.1.7.5 Null Validation (F6)	96
5.1.7.6 User Validation (F7)	97
5.1.8 Error Messages Field Define / Undefine (F9)	99
5.2 Error Messages (F3)	99
5.2.1 Select Error Message Filename (F2)	100
5.2.2 Define Error Messages (F3)	100
5.2.2.1 Insert Line (F3)	101
5.2.2.2 Delete Line (F4)	101
5.2.2.3 Duplicate Line (F5)	102
5.2.2.4 Restore Line (F6)	102
6 Panels	103
6.1 Mark/Unmark (F2)	104
6.2 Panel List (F3)	104
6.2.1 Select (Enter)	105
6.2.2 Show Panel (F3)	105
6.2.3 Unshow Panel (F4)	106
6.2.4 Select First Panel (F5)	106
6.2.5 Copy Panel (F6)	106
6.2.6 Delete Panel (F8)	107
6.2.7 Relocate Panel (F9)	107
6.3 Action Bars (F4)	107
6.3.1 Action Bar (F2)	108
6.3.1.1 Accept Text (Enter)	111
6.3.1.2 Insert Before Entry (F2)	112
6.3.1.3 Insert After Entry (F3)	112
6.3.1.4 Delete Entry (F4)	112
6.3.1.5 Relocate Entry (F5)	112
6.3.1.6 Position Bar (F6)	113
6.3.1.7 Alter Size (F7)	113
6.3.1.8 Action Bar Name (F8)	114
6.3.1.9 Edit Action Bar Entry (Enter)	114
6.3.1.10 Complete Action Bar (Esc)	114
6.3.2 Save (F4)	114
6.3.3 Generate (F6)	114
6.3.4 Initialize (F9)	116

6.3.5 Pulldown (Enter)	116
6.3.5.1 Insert Before Entry (F2)	117
6.3.5.2 Insert After Entry (F3)	117
6.3.5.3 Delete Entry (F4)	117
6.3.5.4 Pulldown Name (F6)	117
6.3.5.5 Previous Pulldown Menu (<left-arrow>)	118
6.3.5.6 Next Pulldown Menu (<right-arrow>)	118
6.3.5.7 Edit Pulldown Entry (Enter)	118
6.3.5.8 Accept Text (Enter)	119
6.3.6 Key/Key+ Mouse (F10)	119
7 Panel Painting	121
7.1 Working with Blocks	122
7.1.1 Move Block (<down-arrow> <up-arrow> <right-arrow> <left-arrow>).	122
7.1.2 Paste Block (F2)	122
7.1.3 Copy from Block (F3).	123
7.1.4 Stack Block (F4)	123
7.1.5 Abandon (F10).	123
7.2 Panel Painting Main Menu	123
7.2.1 Mark/Unmark (F2).	123
7.2.2 Panel Fields (F3).	124
7.2.3 Panel Groups (F4)	124
7.2.4 Paint Attribute (F5).	124
7.2.5 Attribute Roll (F6).	125
7.2.6 Cut to Block (F7)	125
7.2.7 Copy to Block (F8).	126
7.2.8 Restore Block (F9)	126
7.2.9 Panel Maintenance (F10)	126
7.2.9.1 Move (F2)	127
7.2.9.2 Alter Panel Size (F3).	127
7.2.9.3 None/Single/Double Border (F4).	127
7.2.9.4 Top/Bottom Anchor (F5)	128
7.3 Panel Painting Alternate Menu (Alt)	128
7.3.1 Local Dialog (F2)	129
7.3.2 Attribute Palette (F3)	129
7.3.3 Panel Name (F4)	131

7.3.4 Clear (F6)	131
7.3.5 Protect Definition (F7)	132
7.4 Panel Painting Control Menu	133
7.4.1 Field Order (F2)	134
7.4.2 Previous Panel (F3)	134
7.4.3 Next Panel (F4)	135
7.4.4 Draw (F6)	135
7.4.4.1 Draw/Erase/Move (F2)	135
7.4.4.2 Single/Double (F3)	136
7.4.4.3 Attribute Roll (F6)	136
7.4.5 Character Palette (F7)	136
7.4.6 Delete Field Definition (F9)	137
7.4.7 Delete Group Definition (F10)	137
7.4.8 Menu On/Off (Ctrl+End)	137
8 Panel Fields	139
8.1 Defining Panel Fields (F3)	139
8.1.1 The Popup Panel	140
8.1.2 The Panel Field Menu	146
8.1.2.1 Select from List (Enter)	146
8.1.2.2 Panel Functions (F3)	146
8.1.2.3 Amend Field (F5)	147
8.1.2.4 Use Stacked Field (F6)	148
8.1.2.5 Sort List (F7)	149
8.1.2.6 Unsort List (F8)	149
9 Panel Groups	151
9.1 Group Types	151
9.2 Working with Groups	152
9.2.1 Selection Bar (F5)	152
9.2.1.1 Delete Selection Bar (F4)	153
9.2.1.2 Paint Attribute (F5)	153
9.2.1.3 Attribute Roll (F6)	153
9.2.1.4 Escape (Esc)	153
9.2.2 Group Name (F8)	154
9.2.2.1 Group Namelist (F2)	154
9.2.2.2 Escape (Esc)	154

9.2.3 Virtual Text/Attribute (F2)	154
9.2.3.1 Insert Line (F3)	156
9.2.3.2 Delete Line (F4)	156
9.2.3.3 Restore Line (F7)	156
9.2.3.4 Duplicate Line (F8)	156
9.2.3.5 Delete Virtual Text/Attribute (F9)	156
9.2.3.6 Abandon (F10)	157
9.2.3.7 Escape (Esc)	157
9.2.4 Group Size Maintenance (F9)	157
9.2.4.1 Expand/Contract (F2)	158
9.3 Data Groups	158
9.3.1 Group Accept Exit Bar/End (F2)	159
9.3.2 Selection Bar (F5)	159
9.3.3 Group Name (F8)	159
9.3.4 Group Occurrence Maintenance (F9)	160
9.3.4.1 Field (F3)	160
9.3.4.2 Group Size Maintenance (F9)	160
9.3.5 Add Occurrence (<down-arrow>)	161
9.3.6 Remove Occurrence (<up-arrow>)	161
9.4 Text Groups	161
9.4.1 Define Virtual Text (F2)	162
9.4.2 Selection Bar (F5)	162
9.4.3 Group Name (F8)	163
9.4.4 Group Size Maintenance (F9)	163
9.4.5 Escape (Esc)	163
9.5 Attribute Groups	163
9.5.1 Define Virtual Attributes (F2)	164
9.5.2 Group Name (F8)	164
9.5.3 Group Size Maintenance (F9)	164
9.5.4 Escape (Esc)	164
10 Dialog	165
10.1 Functions, Parameters and Procedures	165
10.1.1 Functions	166
10.1.1.1 Attribute Functions	166
10.1.1.2 Cursor Functions	166
10.1.1.3 Path Control Functions	167

10.1.1.4	Procedure Functions	167
10.1.1.5	Stack Functions	168
10.1.1.6	Calling Program Function	168
10.1.1.7	Callout Function	168
10.1.1.8	Refresh Functions	168
10.1.1.9	Keyboard Scan Function.	169
10.1.1.10	Mouse Function	169
10.1.1.11	Group Selection Bar Functions.	169
10.1.1.12	Screen Group Data Positioning Functions.	170
10.1.1.13	Screen Group Array Size and Register Functions	170
10.1.1.14	Screen Group Insertion and Deletion Functions.	171
10.1.1.15	Screen Group Procedure Functions	171
10.1.1.16	Conditional Functions	171
10.1.1.17	Data Manipulation Functions	172
10.1.1.18	Flag Functions.	172
10.1.1.19	Panel View Function	173
10.1.1.20	Move Panel Function	173
10.1.1.21	Clear Field Functions	173
10.1.1.22	Validate Function	173
10.1.1.23	Terminate Function.	174
10.1.1.24	Sound Function.	174
10.1.1.25	Timeout Function	174
10.1.2	Parameters	175
10.1.2.1	Panel Name Parameter.	175
10.1.2.2	Field Name Parameter	175
10.1.2.3	Group Name Parameter	175
10.1.2.4	Procedure Name Parameter	176
10.1.2.5	Numeric Value Parameter	176
10.1.2.6	Alphanumeric Value Parameter.	176
10.1.2.7	Attribute Parameter	176
10.1.2.8	Register Parameter	176
10.1.2.9	Null Parameter	177
10.1.3	Using Procedures	177
10.2	Dialog Definition Menu	178
10.2.1	List (F2)	180
10.2.2	Insert Line (F3)	181
10.2.3	Delete Line (F4)	181
10.2.4	Sort Dialog (F5).	181

12.2.5 Cancel Library (F6)	206
12.2.6 Browse File (F8)	206
12.2.7 Rename File (F9)	206
12.2.8 Delete File (F10)	206
12.3 Generate COBOL Menu (Enter)	207
12.3.1 Data Descriptions (F2)	208
12.3.2 Prefix On/Off (F3)	210
13 Programming	211
13.1 Calling Dialog System	211
13.2 Using Multiple Screensets from One Program	212
13.3 Using Error Files Directly from Your Program	213
13.3.1 The Dialog System Error File	214
13.3.2 Alternative Error Files	215
13.4 User Validation in Screensets	216
13.5 Control of User-defined Field Formats	217
13.5.1 Control Block for Dsusrfmt	218
13.6 User Control of Every Keystroke	227
13.7 Obtaining System Information from Dialog System	228
13.8 Run-time Path Name Support	229
13.9 Compiling Dialog System Applications	230
14 Linking	231
14.1 Dialog System Object Modules	231
14.1.1 Production Applications	232
14.1.2 Applications in Development	233
14.2 Linking Your Application	233
14.2.1 Using Dsclink for Linking	235
14.3 Using the Trace Facility in an Executable Object	236
15 Printing	237
15.1 Print Menu (F7)	237
15.1.1 Print Screenset (Enter)	238
15.1.2 Escape (Esc)	239

16 Character Set Support on UNIX	241
16.1 DOS and UNIX Screensets	241
16.1.1 DOS Screensets	241
16.1.2 UNIX Screensets	242
16.1.2.1 Screenset Naming	243
16.2 The Screenset Conversion Utility	243
16.2.1 Translation Files	244
16.2.2 Attribute Conversion	246
16.3 Transferring DOS Screensets to UNIX	247
16.3.1 Restrictions on Line-drawing Characters	247
16.3.2 Supported UNIX Characters	248
16.3.3 Modifying File Names in Your Application	248
17 Setting Up the Configuration File	251
17.1 The Configuration File dsdef.cfg	251
17.2 Copyfile Defaults	252
17.3 Defaults for Palettes and Attributes	255
17.4 Printing Defaults	257
17.5 General Run-time Behavior	259
18 Common Questions and Answers	269
18.1 Displaying a Validation Message	270
18.2 Defining a Validation Message Panel	271
18.3 Defining Scrolling Data Groups	272
18.4 Inserting and Deleting Fields in an Array	274
18.5 Checking the Size of an Array	274
18.6 Creating a Menu Entry Table	276
18.7 Creating a Highlighted Menu Entry Line	279
18.8 Extracting Virtual Text	282
18.9 Chaining Panels	283

19 Key Code List	285
19.1 ASCII Keys	285
19.2 Function Keys	285
19.3 Status Keys	287
19.4 Any Other Key	288
19.5 Error Key	288
19.6 Mouse Support	288
20 Function Code List	291
20.1 Function Parameters	291
20.2 Function Descriptions	292
21 Syntax of Import/Export Files	331
21.1 Data Types	331
21.2 Full Import/Export Syntax	332
21.2.1 Screenset Definitions	332
21.2.1.1 screenset-details	333
21.2.1.2 form-data-declaration	333
21.2.1.3 error-messages-declaration	333
21.2.1.4 data-validation-declaration	333
21.2.1.5 panel-declaration	334
21.2.1.6 global-dialog-declaration	334
21.2.1.7 panel-dialog-declaration	334
21.2.1.8 data-declaration	334
21.2.1.9 panel-details	335
21.2.1.10 attribute-palette-declaration	335
21.2.1.11 panel-text-declaration	335
21.2.1.12 panel-protect-declaration	336
21.2.1.13 panel-field-declaration	336
21.2.1.14 panel-group-declaration	337
21.2.1.15 dialog-declaration-1	337
21.2.1.16 dialog-declaration-2	338
21.2.1.17 attribute-details	338

21.2.1.18 panel-protect-field-declaration	338
21.2.1.19 Data Types	338
21.2.2 Sample Text File	339
21.2.3 Restrictions	341
22 Error Messages	343
22.1 Run-time System Error Messages.	343
22.2 Definition Time Error Messages.	346
22.2.1 Data Definition	346
22.2.2 Validation Definition	349
22.2.3 Error Message Definition	349
22.2.4 Panel Field Definition	350
22.2.5 Panel Group Definition.	352
22.2.6 Dialog Definition	353
22.3 Screen Painting Error Messages.	357
22.4 Import Utility Error Messages	364
22.4.1 Recoverable Errors	364
22.4.2 Fatal Errors	370
23 Tutorials	373
23.1 Tutorial 1 - Cutting and Pasting Experiment	373
23.2 Tutorial 2 - Groups and Scrolling	375
23.3 Tutorial 3 - Defining Validations	380
23.4 Tutorial 4 - Documenting the Screenset Definition	385
23.5 Tutorial 5 - Using the Trace and Trap	386
23.6 Tutorial 6 - Virtual Text	388
23.7 Tutorial 7 - Virtual Attributes	392
23.8 Tutorial 8 - Changing Fields in Groups	397
23.9 Tutorial 9 - Input Attributes and Procedures	400
23.10 Tutorial 10 - Using Different Group Types	401
23.11 Tutorial 11 - Example of Data Validation	402
23.12 Tutorial 12 - The Application Program Interface	403

23.13 Tutorial 13 - Running the Program with the Trap	413
23.14 Tutorial 14 - Case Sensitivity in Screensets	414
23.15 Tutorial 15 - Using Different Screensets	415
23.16 Tutorial 16 - Paging a Large File	416
23.17 Tutorial 17 - Using Micro Focus Panels	422
23.18 Tutorial 18 - Menu Selection - Example 1	423
23.19 Tutorial 19 - Menu Selection - Example 2	423
24 Dialog System Limits	425

Preface

This guide describes the features, applications and use of the Dialog System Character Mode Definition and Run-time software for DOS, OS/2 and UNIX environments.

Using this manual, the reader should be able to understand the purpose and capabilities of Dialog System, and use it to prototype, test, integrate and run a user interface.

Audience

Readers are expected to be familiar with the general concepts of business computing.

Notation in this Manual

- **Enter** refers to the carriage return or Enter key. Where commands to be typed are shown, the Enter key is not explicitly shown; it is treated as implicit that Enter must be pressed at the end of the line.
- Hexadecimal numbers are enclosed in quotation marks and preceded by a lower-case "x"; for example, x"9D".
- The term "window" means a delineated area of the screen normally smaller than the full screen.

The notation used to describe the format of command lines is as follows:

- Words printed in italics are generic terms representing names to be devised by you.

- Words printed in non-italics are the actual words you must enter. You must enter them in the exact case in which they appear.
- Material enclosed in square brackets [] is optional.
- When material is enclosed in braces { }, you must choose from the options within them. If there is only one option in the braces, the braces indicate repetition.
- the ellipses (...) follows { } or [] and means you can repeat the material in the { } or []. The number of repetitions allowed is unlimited unless otherwise stated. If the ellipsis is used with [], the material can be omitted altogether.
- If a command line will not fit across the page, it is continued on the next line; the continuation line is indented.

UNIX Considerations

- The term "UNIX" can be taken to mean all operating systems which are compatible with UNIX System V, Release 3 or later, complying with the System V Interface Definition (SVID).
- The commands given in this manual are specifically for the UNIX operating system. For all other similar operating system environments, you should refer to your **Release Notes** for details of the commands you need to use. You may also be referred to the **Release Notes** for other operating system specific details.
- All command line formats and examples are for the standard UNIX shell - the Bourne shell. If you are using another shell, see your UNIX documentation for the appropriate formats .

Where examples showing environment variables do not specifically show them being exported to the shell, it is treated as implicit that environment variables are exported.

- The keys described in this documentation are not available in all environments. References to pressing keys such as function or status keys imply the logical pressing and releasing of these keys rather than the physical keystrokes. A chart listing how your actual keystrokes map onto the keys shown in the documentation is provided as an appendix to this book.

- You may notice that what appears on your screen sometimes differs in minor ways from that illustrated in the manual. This will not affect the operation of your software.

1 Overview

This chapter introduces Dialog System and describes how programs are developed with Dialog System.

1.1 Dialog System

The Micro Focus Dialog System provides for independence between screen and keyboard input/output (I/O), and a COBOL application program. In much the same way that a database management system (DBMS) provides for independence of the data from the application, Dialog System separates out the human interface. This means that the application program needs only to be concerned with basic data processing tasks, and thus becomes smaller, simpler and much easier to code and maintain. Dialog System consists of a *definition* component and a *run-time* component.

You can use the definition component to define panels, which are rectangular working areas on the screen where you can specify data fields and keystrokes. The interaction between the screen and the user, based on this definition of keystrokes associated with a panel, is known as a *dialog*. Panels, which can be up to 80 characters wide and 25 lines long, can be used individually to create layouts for simple data entry, or they can be used collectively to make *screensets*.

A screenset is a set of panels with their associated data fields and dialog for one application. You can design complex screensets, incorporating pulldown menus, popup windows, selection bars, scrolling groups, help screens, function key support, and data validations.

You can use the run-time component to run the screenset and dialog interactions without an application program; this enables you to test a screenset and its dialog without the application program that uses the screenset. The run-time component also supports the screenset when it is being used by a program. This is done by using a simple call interface

between the program and the screenset through a control block and a Data Block.

Therefore, once a program is produced, you can make extensive changes to the screenset without actually changing the program.

1.2 Environments

This guide describes the Character Mode version of Dialog System. You can use this version to create character-based user interfaces for applications running in character environments. The alternative version, for Graphical Mode, enables you to create graphical user interfaces for applications to run in graphical environments such as Microsoft Windows.

Obviously, the differences between UNIX and Windows operating systems result in differences in the use of certain facilities, such as line drawing and character graphics. To avoid problems, use only standard ASCII characters (codes 032 through 127) in any text that you enter. See the chapter *Panel Painting* for more information. Other limitations are clearly identified in this guide.

1.3 Program Development with Dialog System

The approach to program development using Dialog System is different to previous methods of program development. How much you change your own approach depends on a number of factors, including your own individual preferences.

The fundamental advantage of using Dialog System over your previous method is that all keyboard and screen handling is removed from your program. Additionally, most of the data validation, menu control, scrolling lists and help screens are no longer handled by your program.

Instead, all of these functions are handled by the Dialog System, using a very simple call interface to the COBOL program through a control block and a Data Block.

The major steps in developing your application with Dialog System are:

- 1 Defining the Data Block. The Data Block is passed between the calling program and the Dialog System at run time, and contains the data fields and any data validations. See the chapter *Data Definition*.
- 2 Defining the panels. This step includes creating screen panels and establishing any desired colors and screen prompts. See the chapter *Panels*.
- 3 Positioning data fields on the panels. See chapters *Panel Fields* and *Panel Groups*.
- 4 Defining the dialog. The dialog is the control of the function of keystrokes, for example showing a "help" panel if F1 is pressed. See the chapter *Dialog*.
- 5 Testing the screenset. You can run the screenset to test the behavior of the screenset and its associated dialog. See the chapter *Running the Screenset*.
- 6 Modifying the screenset, as required.
- 7 Generating a copyfile. You generate a COBOL copyfile from the screenset for use in the COBOL application program. See the chapter *Generating the Copyfile*.
- 8 Writing a calling program. You write a COBOL application program with the necessary calls to the Dialog System. See the chapter *Programming*.
- 9 Testing the calling program.
- 10 Modifying the screenset, as required.
- 11 Linking Dialog System modules to your application. This is an optional step you can use if you want to create a single executable object by linking Dialog System modules to your application. See the chapter *Linking*.

2 Getting Started

This chapter describes:

- The procedures for invoking Dialog System
- The on-screen menu system that this system uses
- The key functions for moving around the panel area and for using lists
- Using Dialog System from the command line

2.1 Invoking Dialog System

Windows: If you are running under Windows, you invoke the Dialog System Definition software from the command line by typing:

```
dsch
```

and pressing **Enter**.

UNIX: If you are running under UNIX, enter:

```
ds
```

If you are running Dialog System on an operating system other than Windows or UNIX, see your **Release Notes** for details of the command to invoke the software on your particular operating system.

When you invoke Dialog System, an identification screen containing copyright notices, the software version number and your User Reference Number appears briefly and is then replaced by a screen containing a status line and the Main menu at the bottom.

The Main menu looks similar to Figure 2-1. Depending on the type of terminal you are using, you might see either a single or double line on the status line.

Figure 2-1. Main Menu

```

New set-----Row:01=Col:01=Ins=Cap=Num=Scroll
F1=help F2=data-definition F3=panels F4=global-dialog F5=generate F6=run
F7=print F8=tutorial                               Alt Escape

```

2.1.1 Status Line

The status line, located just above each menu, gives the current status of the active panel. As you work with the Dialog System, the status line is updated with the following information:

Name	This is the name of the current menu or selected screenset. If you are working on a new screenset, "New set" is displayed.
Function name	This is the name of the function currently in use in a submenu (if applicable).
Attribute	This reflects the character attribute selected (if applicable).
Row	This shows the number of the row in which the cursor is positioned. This row number might be relative to the current panel or to the top or bottom of the screen.
Col	This shows the number of the column in which the cursor is positioned. This row number might be relative to the current panel or to the top or bottom of the screen.
Ins, Caps, Scroll	These indicators show the status of the Insert, Capitals, and Scroll toggles.
Alt, Ctrl	These indicate that an Alternate menu or Control menu available with the current facility is active; they are in bold when the keys are pressed. There is a limitation, described in the next section, on the use of these keys in UNIX environments.

2.1.2 Menu System

Dialog System operates under a hierarchy of menus, with one menu leading into another. The current menu is always displayed at the bottom of the screen. However, it is possible to remove the menu from the screen, if desired (see the chapter *Panel Painting* for details).

From the Main menu, you can access the submenu, which provide main definition functions, testing functions, and utilities, by pressing the appropriate function key. You can also access the Main Alternate (Alt) menu by holding down the **Alt** key from the Main menu (see Figure 2-2). This menu enables you to access the screenset functions, access some additional utilities, and set some defaults.

Figure 2-2. Main Alt Menu

```
New set=_____Row:01=Col:01=Ins=Caps=Nur=Scrol
F1=help F2=initialize F3=load-screenset F4=save-screenset F5=screenset-switches
F6=normal/top/botton coords F7=set-dets F8=colorize F9=import F10=export Alt
```

Windows: To select a facility from the Main Alt menu on Windows, press the appropriate key while you hold down the **Alt** key. To return to the Main menu from the Main Alt menu, release the **Alt** key.

UNIX: To select a facility from the Main Alt menu on UNIX, toggle the **Alt** key on (so that Alt appears on the status line) and then press the appropriate function key. To return from the Main menu, toggle the **Alt** key off.

Many menus in the system contain Alternate or Control Menus, which are accessed in the same way as the Main Alt menu. The mnemonics "Alt" or "Ctrl" appear on the lower right side of the menu if these facilities are available.

UNIX: In UNIX environments, if you press the **Alt** or **Ctrl** keys when the Alternate or Control menus are not available, the current menu remains on the screen, but the system beeps when you press a function key. To return to the current menu so that you can select the desired facility, toggle the **Alt** or **Ctrl** key off (so the mnemonic disappears from the status line).

To return from a selected submenu to the previous menu without executing any of the available facilities, you simply press the **Escape** key. So, if you enter a submenu and subsequently decide that you do not wish to use any of the available options, you can exit it without affecting any of the work you have done on your screenset.

2.1.3 Help Screens

All menus in the Dialog System contain a help facility, which is accessed by pressing the **F1=help** key. This provides a help screen associated with the function or menu you last selected. For example, requesting help from the Main menu results in a help screen explaining the seven other facilities available from the Main menu. Requesting help from the Data Definition menu results in help screens explaining the two facilities available for data definition. The menu from which you requested help remains visible at the bottom of the screen. You can clear the help screen by pressing **F1** again or the **Space** bar. All Dialog System help screens behave in the same way, so they are not described with the other menu facilities detailed throughout this manual.

2.1.4 UNIX Keyboard Mapping

UNIX: Some UNIX terminals might not provide all the control keys referred to earlier in this chapter. If your terminal does not, you can use the following key sequences to achieve the same effect:

Control Key	Equivalent
Alt	/a or /A
Ctrl	/c or /C
/	//
Escape	/@
F1 to F9	/1 to /9
F10	/0
Help	/H
Maphelp	/m
Insert	/i
Delete	/d

Control Key	Equivalent
Backspace	/b
Home	/h
End	/e
Tab	/t
Backtab	/T
Caps Lock	/l or /L
Scroll	/S
PageUp	/p
PageDown	/n
Clear-to-End-of-Screen	/E

Remember that UNIX is case sensitive, so you must ensure that any of these key sequences, particularly those for Tab and Backtab, are entered in the appropriate case.

2.2 Key Usage in Dialog System

As well as the options available on each menu, there are some keys that have particular functions in Dialog System when you lay out the details of a panel or when you use lists. The following two sections show the available functions, the key used to invoke the function, and a description of the resulting action. If your keyboard does not have the control keys shown below, please consult your Release Notes to see which keys on the standard keyboard for your system provide the equivalent functions.

2.2.1 Painting Panels

Function	Documented Key	Description
Cursor left	←	Moves the cursor left one character position within the limits of the panel.
Cursor right	→	Moves the cursor right one character position within the limits of the panel.
Cursor up	↑	Moves the cursor up one line within the limits of the panel.
Cursor down	↓	Moves the cursor down one line within the limits of the panel.
Enter	Enter	Moves the cursor to the first character position on the next line.
Tab	→	Moves the cursor four positions to the right within the limits of the current panel.
Backtab	←	Moves the cursor four positions to the left within the limits of the current panel.
Left of Panel	Home	Moves the cursor to the left- hand side of the current panel.
Right of Panel	End	Moves the cursor to the right- hand side of the current panel.

2.3 Using Dialog System from the Command Line

You can use Dialog System from the command line in batch mode.

Windows: If you are running under Windows, use a command in the format:

```
dsch input-option output-option [update-option]
```

UNIX: If you are running under UNIX, use a command in the format:

```
ds input-option output-option [update-option]
```

where:

<i>input-option</i>	is mandatory and must be one of the following batch mode options:
<i>*is screenset-name</i>	Load a screenset from a screenset file.
<i>*ie export-file</i>	Load a screenset from an export file.
<i>output-option</i>	is mandatory and must be one of the following batch mode options:
<i>*os screenset-name</i>	Save resultant screenset to <i>screenset-name</i> file.
<i>*oe export-file</i>	Export resultant screenset to <i>export-file</i> .
<i>*op print-file</i>	Save printable description of resultant screenset to <i>print-file</i> .
<i>*oc copyfile</i>	Generate <i>copyfile</i> from the resultant screenset..
<i>*ob copybook</i>	Generate <i>copyfile</i> from the resultant screenset with data names not prefixed by screenset names.
<i>*oy export-file</i>	Export resultant screenset to <i>export-file</i> with 2-character year fields in dates converted to 4-character year fields, using the date field format <i>yyyymmdd</i>
<i>*om export-file</i>	Export resultant screenset to <i>export-file</i> with 2-character year fields in dates converted to 4-character year fields, using the date field format <i>mmddyyyy</i> .

	<code>*od export-file</code>	Export resultant screenset to <i>export-file</i> with 2-character year fields in dates converted to 4-character year fields, using the date field format <code>ddmmyyyy</code> .
<code>update-option</code>	is optional and can be the following batch mode option:	
	<code>*ue export-file</code>	Apply amendments from export file.

where:

- * should be replaced by the forward slash character (/) for non-UNIX applications, and by the exclamation mark (!) for UNIX applications.

Note: The options `*oy`, `*om` and `*od` convert 2-character year date fields in the screenset to 4-character year date fields. All affected dates are changed to the same date format, regardless of the original date format, for example, if you choose the `*oy` option for a screenset containing a mix of `ddmmyy` and `mmddy` date fields, all the date fields are converted to `yyymmdd` format. The data definitions of these fields in the Data Block are amended, related validations are changed as necessary, and the size and possibly position of any such date fields displayed on screen are adjusted. Each change in the export file is preceded by a comment line, helping you to locate such changes quickly. The export files produced have extensions of `d20`, `m20` or `y20` depending on the chosen date format.

The options are not case sensitive, and you can specify them in any order. You can specify the filename for an option immediately adjacent to the option, or separated from it by spaces.

For example, in Windows, to open a screenset `oldsset.s`, apply the amendments in the export file `modfile.txt` and save the resulting file as the screenset `newsset.s`, use the following command:

```
dsch /is oldsset.s /ue modfile.txt /os newsset.s
```


When you run Dialog System in batch mode, the command line is displayed on the screen. After run completion you receive either the message 'Completed OK' or an error message.

A log file (**ds.log**) is either created in the current directory, or, if it already exists, is appended with the information that is displayed on the screen. Every log file entry contains a date and time stamp.

This file is in text format, and can be listed or edited. You can delete the file, in which case a new log file will be created in the current directory during the next batch run.

3 Sample Session in Character Mode

This sample session will familiarize you with the Character Mode version of Dialog System.

Using the sample program **Test1** you will learn the basic steps to define the data items for a screenset, design its on-screen appearance, test it, write a program to use it, specify the effects of keys, and set screen attributes.

This sample session will produce a simple data entry screen for entering the following product information:

- Product code
- Product name
- Cost price
- Quantity in stock

All the features covered are described in detail later in this online book.

In Dialog System, data areas in memory (known in COBOL as data items) are referred to as fields. This same term is used for areas on the screen; for example data entry fields are areas where data can be entered.

A familiarity with the basic ideas of COBOL may be helpful in using this tutorial.

3.1 Creating a Screenset

In this section you will:

- 1 Invoke the Dialog System definition software.
- 2 Define a simple Data Block.
- 3 Save a screenset to disk, and then load it back.
- 4 Paint a panel.
- 5 Reposition fields in the panel.
- 6 Test the running of your screenset.

3.1.1 Starting Dialog System

Windows: To start the Dialog System definition software on Windows, from the Net Express IDE, click **UNIX, Dialog for UNIX**.

UNIX: On UNIX, the tutorial programs for this product are in subdirectory **demo** in the directory pointed to by the `$COBDIR` environment variable. Before you use any of them, we recommend you copy them to your own work directory to avoid the danger of other users working with them at the same time, as follows:

- 1 Copy the source file **TEST1.cbl** from **\$COBDIR/demo** into one of your work directories, and **cd** into that directory.
- 2 Enter:

```
tbox
```
- 3 Press **Ctrl-C** to get the Ctrl menu, then press **F6**.

The Dialog System banner appears, followed by the main menu of Dialog System.

You are now ready to define a screenset.

3.1.2 Defining A Data Block

Here you define the Data Block for your screenset. When your application calls the Dialog System run-time software, it passes the Data Block as a parameter. The Data Block contains all the fields that the application wishes to display on the various panels that make up a screenset.

- 1 Select **F2=data-definition**, then select **F2=data-fields** from the menu that appears.
- 2 The Data Definition menu appears, together with a list, as shown in Figure 3-1. The cursor is positioned in the first entry in this list, under the column heading **Field**. This is where you are going to enter the description of the first field, **P-CODE**.

Figure 3-1. The Data Definition Menu

Field	Fmt	Int.Dc	Comp	Rpts	Val
		0.0		0	

Data-Definition Ins-Caps-Ctrl-Scroll
 F1=help F2=amend-grp-size F3=ins-field F4=del-field F5=amend-grp-rpts F6=def-grp
 F7=undef-grp F8=validation F9=err-msg-fld def/undef ← ↑ ↓ PgUp PgDn Escape

Note: During the following steps, if you type anything invalid, an error message will be displayed. Use the **cursor-left**, **cursor-right**, **Tab** and **Back-tab** keys to return to your error and correct it.

- a Type P-CODE and press the **Tab** key.
- b This takes the cursor to the Fmt column, where you specify the field format. Dialog System supports four field formats:
 - A = Alphabetic
 - X = Alphanumeric
 - 9 = Numeric
 - S = Signed Numeric

Because the field you are defining is alphanumeric, type "X" under the Fmt heading.

- c The cursor automatically skips to the next column, which is headed Int.Dc. Here you enter the field size, as an integer part and a decimal part. P-CODE is going to be 6 characters long, with no decimal places, so enter "6".

There are three further columns in the list, called Comp, Rpts and Val. You do not need to use these in this session.

- d Now enter the details for the next field, the product name. Press **Enter** to get a new line. The second line is highlighted. Enter the values "P-NAME", "X" and "15" into the Field, Fmt, and Int.Dc columns, respectively.
- e Enter the remaining two fields as:
 - P-COST, numeric (9), three digits before the decimal point, two digits after the decimal point (3.2).
 - P-QUANT, signed numeric (S), four digits before the decimal point, no digits after the decimal point (4).

The list of fields should now be as shown in Figure 3-2.

Figure 3-2. The Completed List of Fields

Field	[Fmt]	Int.Dc	Comp	Rpts	Val
P-CODE	X	6.0		0	
P-NAME	X	15.0		0	
P-COST	9	3.2		0	
P-QUANT	S	4.0		0	

Data-Definition Ins-Caps-Ctrl-Scroll
 F1=help F2=amend-grp-size F3=ins-field F4=del-field F5=amend-grp-rpts F6=def-grp
 F7=undef-grp F8=validation F9=err-msg-flt def/undef ← ↑ ↓ PgUp PgDn Escape

- f Press **Esc** twice to exit from the Data Definition menu. This brings you back to the Dialog System main menu.

What you have just defined has disappeared from the screen, but it has not been lost. Generally, pressing **Esc** returns you to a higher level menu and retains what you have just defined. To confirm this, enter the Data Definition window again (select **F2=data definition** and then **F2=data fields**) to see it. Then press **Esc** twice to return to the main menu.

3.1.3 Saving Your Screenset

The next step is to save the screenset on disk.

- 1 Type **Alt+F4** (that is, hold down the **Alt** key and press **F4**). You are prompted to give the screenset a name. A default path, which you can alter, appears.

Windows: On Windows, we recommend you use the demo subdirectory of the Dialog System directory. i.e. `c:\netexpress\UNIX\dsschar\demo`.

- 2 Enter the name "TEST1" and press **Enter**. There will be some disk activity, then the main menu returns. However, there is a difference; the words "New set" have been replaced with the name of the screenset, "TEST1".

The screenset has now been saved. At this point you could if you wished exit Dialog System and come back to this screenset later.

Dialog System saves all character mode screensets with the extension `.S`, so your screenset TEST1 is stored on disk in a file named `TEST1.S`.

3.1.4 Loading a Screenset

Although you are not going to exit Dialog System at this point, why not clear the screenset from memory and then reload it from disk just to confirm that it has been saved successfully.

- 1 Press **Alt+F2=initialize**. You are prompted:

```
Initialize. Are you Sure? Y/N
```

- 2 Reply **Y**.

The word "TEST1" changes back to "New set". You are now ready to start another screenset.

- 3 Press **Alt+F3=load screenset**. The menu prompts for a screenset name. The menu also shows **F2=directory**.
- 4 You can either type the name of the screenset at the prompt, or press **F2=directory** and select it from the list of existing screensets that is then displayed. To use the directory function, after pressing **F2** select **TEST1.S** with the cursor keys. Then press **Enter**. This copies the name **TEST1.S** to the prompt line.
- 5 Whether you typed it yourself or used the directory function, you should now have the name **TEST1.S** at the prompt line. Press **Enter**.

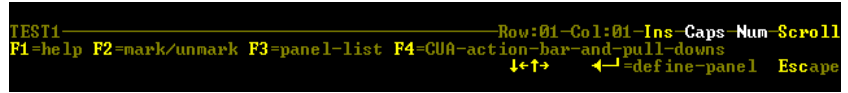
TEST1 is now loaded and you can begin further development of this screenset.

3.1.5 Painting a Panel

Now that you have defined the fields, you need to specify the visual aspects of the screen. You do this by painting a panel.

- 1 On the main menu, press **F3=panels**. The Panel menu appears, as shown in Figure 3-3 (the Figure shows just the menu itself, not the whole screen).

Figure 3-3. The Panel Menu



```

TEST1-----Row:01-Col:01-Ins-Caps-Nun-Scroll
F1=help F2=mark/unmark F3=panel-list F4=CUA-action-bar-and-pull-downs
↓←→ ←↓=define-panel Escape
  
```

- 2 You can move the cursor around using the cursor keys and the **Tab**, **Backtab**, **Home**, and **End** keys. Move the cursor to row 5, column 10. You can check this from the Row and Col positions displayed on the status line just above the menu.
- 3 Press **F2=mark/unmark**.

A highlighted character appears in the cursor position. Now move the cursor to row 15, column 70. The entire rectangle thus marked out becomes highlighted. This area is going to be your working panel.
- 4 Press **Enter** to define this area as a panel.
- 5 You are prompted to give the panel a name. Enter **STOCK** and press **Enter**.

The panel has now been defined and a border is drawn around it.

The menu has now changed to the Panel Painting menu shown in Figure 3-4. You are in panel painting mode and the cursor is now restricted to being inside the panel. The row and column numbers are now relative to the panel.

Figure 3-4. The Panel Painting Menu

```

STOCK-----Attribute-----Row:01-Col:01-Ins Caps Num Scroll
F1=help F2=field-order F3=previous-panel F4=next-panel F6=draw F7=char-palette
F9=delete-field-defn F10=delete-group-defn          Ctrl+End=menu-on/off

```

The word "STOCK" appears at the left-hand end of the status line. The menu has a single line drawn along the status line instead of the double line in the main menu.

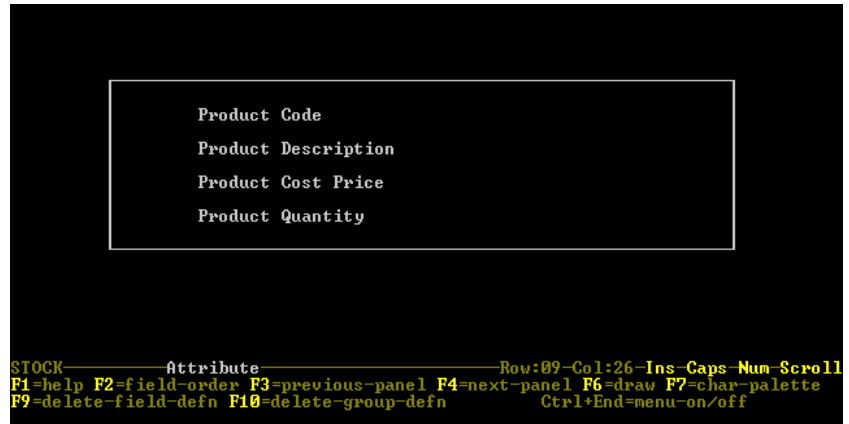
If you press **Esc**, you return to the Panels menu again. If you do this by accident and you want to return to painting the Stock panel again, select **F3=panel-list** from the Panels menu. This brings down a list of panels ? it will have just one entry, Stock ? with a highlighted *selection bar*. You select a panel by using the **cursor-up** and **cursor-down** keys to move the selection bar and pressing **Enter** at the required panel.

- 1 Try moving the cursor around with the cursor keys and the **Tab**, **Backtab**, **Home** and **End** keys. Watch the row and column counters.
- 2 Press the **Ins** key. Look at the Ins characters on the status line. When insert mode is on they are highlighted.
- 3 Press the **Ins** key again. Watch the status line.
- 4 Press the **Caps Lock** and **Scroll Lock** keys and watch the status line.
- 5 Turn all these keys off for the moment.
- 6 Now you add the data entry fields and their accompanying text to your panel. You put the cursor in the required position and then type the heading, or select and describe the field, that you want.

First add the accompanying text. Move the cursor to row 3, column 10 (remember these numbers are now relative to the panel) and type "Product Code".

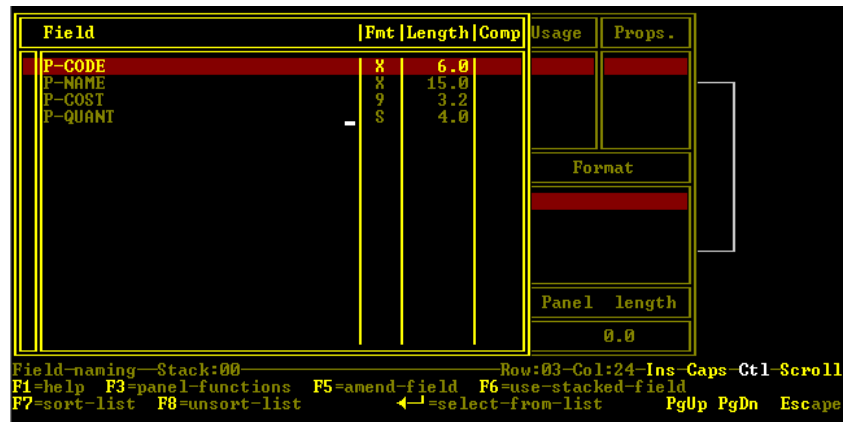
- 7 Similarly, type "Product Description" at row 5, column 10; "Product Cost Price" at row 7, column 10; and "Product Quantity" at row 9, column 10. Your panel should now be as shown in Figure 3-5.

Figure 3-5. The Panel With Text Added



- 8 The next step is to define the data entry fields. Move the cursor to row 3, col 24 and press **F3=field**. The menu changes and the Field Selection panel appears, as shown in Figure 3-6. This is a table showing fields and related information, allowing you to define the properties you want for the entry fields. The fields listed are the fields you defined earlier, and the part of the table containing their names is in a highlighted box.
-

Figure 3-6. The Field Selection Panel



- 9 You use the **cursor-up** and **cursor-down** keys to select a data item. A highlighted selection bar indicates the item selected. Select P-CODE and press **Enter**.

The highlighting now surrounds the Usage part of the Field Selection panel. This part of the table contains a selection bar too. You can now specify the usage of P-CODE, that is, the purpose for which P-CODE will be used. (Do not confuse this usage with the USAGE clause in COBOL.)

One of the entries in the Usage list has the symbols > and < beside it. This entry is the currently selected usage. The default usage is IN, which means this field is to be used for input.

To select a usage, move the selection bar to it and press the space bar. Only one usage can be active at a time. The usage you want for P-CODE is in fact the default, IN, so select this. Then press **Tab**.

- 10 The highlighted surround now moves to the Properties part of the Field Selection Panel, and a selection bar appears there. You can now specify the properties of P-CODE, that is, how the field will behave.

You can select a property by moving the selection bar to it and pressing the space bar. Several properties can be active at one time. You can de-select a property by pressing the space bar again.

Make sure that AUTO is the only property selected. Then press **Tab**.

- 11 The selection bar now moves to the Format part of the Field Selection panel.

You can have only one format active. The use of formats will be discussed later. Make sure that ALPHANUMERIC is selected. Then press **Tab**.

- 12 You can now enter the Panel Length. This is the length of the field, P-CODE, when it is displayed on your panel. The Panel Length defaults to the length of the field in the Data Block, which in this case is 6. You can make the field length displayed on the panel shorter than 6, but not longer.

Make sure that the entry in Panel Length is 6.

The Field Selection panel at this point should be as shown in Figure 3-7. If you have got any of it wrong use the **Tab** and **Backtab** keys to move around and correct it.

Figure 3-7. The Completed Field Selection Panel

Field	Fmt	Length	Comp	Usage	Props.
P-CODE	X	6.0		>IN <	>AUTO <
P-NAME	X	15.0		OUT	REQD
P-COST	9	3.2		EXIT	FULL
P-QUANT	S	4.0		XMOD	NOECHO
				XREG	UPPER
Format					
>ALPHANUMERIC <					
Panel length					
6.0					

Field-naming—Stack:00— Row:03—Col:24—Ins—Caps—Ct1—Scroll
 F1=help F2=reset F3=panel-functions ←=complete
 F4=horizontal-scrolling-on/off →|← Escape

- 13 Press **Enter**.
- 14 Press **Esc** to return to the Field Selection panel.
- 15 A row of six caret characters appears on your panel at the point where the cursor is. The Field Selection panel is now ready for you to position and describe another field.
 You now want to position P-NAME.
- 16 Move the cursor to row 5, col 33. Press **F3=field**.
- 17 Put the selection bar on P-NAME and press **Enter**. The highlighted surround moves to the Usage part.
- 18 This time you want all the defaults, so you need not go through the description process used above. Simply press **Enter** a second time. Press **Esc** to return to the Field Selection panel.
- 19 Similarly, insert P-COST at row 7, column 30 and P-QUANT at row 9, column 28. Don't worry if you get these positions wrong, there is a very easy way to correct them.
- 20 When you have finished positioning and describing the fields, your panel should look similar to Figure 3-8.

Figure 3-8. The Panel With Entry Fields Added

```

Product Code  ^^^^^^
Product Description  ^^^^^^^^^^^^^^^^^^
Product Cost Price  ^^^.^^
Product Quantity  _^^^

STOCK-----Attribute-P-QUANT-----Field-----Row:09-Col:28-Ins-Caps-Num-Scroll
F1=help F2=field-order F3=previous-panel F4=next-panel F6=draw F7=char-palette
F9=delete-field-defn F10=delete-group-defn          Ctrl+End=menu-on/off

```

- 21 Move the cursor around your panel. You can see that, as the cursor moves across a field, the word field appears on the status line together with the name of the field.

3.1.6 Moving Fields Around

To make the panel look neater, you can align the data entry fields.

- 1 Move the cursor to the first caret character of the field P-CODE.
- 2 Press **F2=mark** to turn on marking.
- 3 Move the cursor right so that the highlighting covers the entire field P-CODE.
- 4 Press **F7=cut-to-block**.
- 5 Press the cursor keys and observe what happens.

You can move the cut block to anywhere on the panel. You want to keep this field in line with its prompt "Product Code", but move it to the right by 6 character positions so you can line all the fields up underneath each other.

- 6 Move the cut block to its correct position and select **F2=paste-block**. The block, containing the field, is pasted back on the panel in the required position.
- 7 Do the same with the other fields so that they all line up correctly. The panel should finally look something like Figure 3-9.

Figure 3-9. The Panel After Tidying Up



- 8 You have now completed defining the panel for your screenset. Press **Esc** twice to get back to the main menu. Then save the screenset, using **Alt+F4**.

3.1.7 Running Your Screenset

You can run the screenset without writing a program. This will allow you to evaluate and test its behavior, and to alter it by simply returning to the screenset definition. This is useful for prototyping your interface.

- 1 From the main menu select **F6=run**. After some disk access the screen shown in Figure 3-10 appears. This is called the Trap window.

Figure 3-10. The Trap Screen

Control Block Values		panel name.....< >	
control.....[N]	<C,D,N,P>	error code.....<00>	
screen set name..[TEST1.S]	validation code...<000>	
clear dialog.....[0]	<0, 1>	exit field.....<0>	
procedure.....[000]	<P000-P255>	data block versn..<00>	
control paran....[000]		field number.<0000>.occurrence..<0000>	
		field change.<0>.....count..<0000>	
Data item	Data value		
Scroll=Position=000			
F1=help F2=trace-on/off F3=trap-on/off F4=initialize-data-fields PgUp PgDn Esc			

- 2 Do not try to understand this window fully at this point. Just be aware that there is a field on the window called control which has the value "N", meaning new screenset.
- 3 To run the screenset, press **Enter**.
- 4 What you see now is the panel you have just painted, with the cursor sitting in the first field on the screen. Try to:
 - Enter data into these fields.
 - Enter alphabetic information into one of the numeric fields.
 - Use the **cursor-right** and **cursor-left** keys.

These will allow you to move through and from field to field in either direction.
 - Use the **Tab** and **Backtab** keys.
 - Use the **cursor-up** and **cursor-down** keys.

Note the effects these keys have. Some keys do nothing but beep. The section *Dialog* later in this chapter will explain how to define effects for keys.
- 5 When you have finished moving around the screen, press **Esc**. You return to the Trap window.

Figure 3-11. The Trap After Running the Screenset

Control Block Values		panel name.....<STOCK >	
control.....[C]	<C,D,N,P>	error code.....<00>	
screen set name..[TEST1.S] <0, 1>	validation code...<000>	
clear dialog.....[0]	<0, 1>	exit field.....<0>	
procedure.....[000]	<P000-P255>	data block versn..<01>	
control param....[000]		field number.<0004>.occurrence..<0000>	
		field change.<0>.....count..<0004>	

Data item	Data value
P-CODE	QQQQQQ
P-NAME	logs
P-COST	124.34
P-QUANT	3333

Scroll=Position=001

F1=help F2=trace-on/off F3=trap-on/off F4=initialize-data-fields PgUp PgDn Esc

If you have not returned to the Trap window, you probably accidentally pressed **F3=trap on/off** when you were first in the Trap window. Simply press **Ctrl+Break** followed by **Esc** in this case.

This window has changed in several ways (see Figure 3-11):

- The value of the control field has changed to "C". This means continue with the current screenset.
- At the top right of the window the panel name is displayed. There are other values up here that we will use later.
- The list of data fields in the bottom half of the window now contains the values you entered.

6 Press **F3=trap on/off** and make sure that "off" is highlighted.

7 Press **Enter** again. The screenset resumes running.

8 If you press **Esc**, nothing happens because you have turned the trap off. To get back to the trap, press **Ctrl+Break** to enable the trap, and then press **Esc**.

This takes you back to the trap. A message to show that **Ctrl+Break** has been pressed is shown.

9 Press **Esc**. This returns you to the main menu.

- 10 Press **Esc**. You are prompted "Exit. Are you sure? Y/N".
- 11 Answer **Y** to exit from the Dialog System definition software.

3.2 Writing a Related Program

Dialog System handles the interface between a COBOL application program and a user using screen and keyboard. This means an application program only needs to be concerned with basic data processing tasks, and thus it becomes smaller, simpler and much easier to code and maintain.

In the first section *Creating a Screenset*, you developed a user interface consisting of a panel and four data entry fields. Now you will develop and animate a COBOL program to use this interface.

3.2.1 The Program

Use an editor to enter the program **test1.cbl** below.

```
$set ans85
working-storage section.
    copy "ds-cntrl.mf".
    copy "test1.cpb".

procedure division.

main-process.
    perform program-initialize
    perform call-screen-manager
    stop run.

program-initialize.
    initialize test1-data-block
    initialize ds-input-fields
    move ds-new-set to ds-control
    move test1-data-block-version-no to
        ds-data-block-version-no
    move "test1" to ds-set-name.
```

```
call-screen-manager .
  call "DSRUN" using ds-control-block,
                    test1-data-block.
```

This program refers to two COPY-files. They are called **ds-cntrl.mf** and **test1.cpb**.

ds-cntrl.mf is provided with the Dialog System software and defines the Control Block that your program passes to the Dialog System run-time software. The control block contains basic information about running the screenset. It is a fixed part of Dialog System and applies to all screensets.

test1.cpb defines the fields passed between your program and Dialog System. These items are called the Data Block. The Data Block is specific to each screenset. Most of the information in the Data Block is information that you entered when you defined your fields using the definition software. You generate **test1.cpb** from the definition software.

3.2.2 Generating the Data Block

- 1 Start up the Dialog System definition software as before, and then load your screenset using **Alt+F3**.
- 2 Press **F5=generate**.
- 3 You are prompted for a name for the copyfile. By default, the copyfile will have the name of your screenset with an extension **.cpb**. If you wish, you can change the name. The sample COBOL program expects **test1.cpb** as the name of the copyfile, so press **Enter** to accept this name.
- 4 Select **F2=data-descriptions**. This starts generation. A message saying generation is under way is displayed. Once generation is complete, this message disappears.
- 5 Exit from Dialog System.

3.2.3 Running Your Program

To run your screenset, you compile and run the program.

Windows: On Windows:

- 1 From the IDE, select **File, New**, and double-click **Project** from the list.
You want to create an empty project with a name of test1, and select the `\UNIX\dschar\demo` directory as the folder to contain the project.
- 2 Select **Project, Properties** and then click **IDE, Import, Import**. Click on COBCPY and then **OK**. COBCPY is then displayed. Double-click on this, so that it is shown in the Variable and Value fields at the bottom of the dialog box.
- 3 Insert `c:\netexpress\unix\dschar\source;` at the beginning of the entry in the Value field, and click **Set** and **OK, OK**.
- 4 Now right-click in the left-hand pane of your project window to add files to the project. Select `test1.cbl` and click **Add**.
- 5 Click **Rebuild**, and click **Yes** to rebuild the whole project.
- 6 Now click **Run** to run your program.

UNIX: On UNIX:

- 1 Enter the COBOL editor as before.
- 2 Use **Alt+F3** to load the sample program.
- 3 Press **F2=COBOL**.
- 4 Press **F2=cmpl/animate**
- 5 Ensure that 'Compile' is shown on the status line. If it isn't, press **F2=compl/anim** until it is.
- 6 Press **Ente**.
The program is compiled.
- 7 Press **Esc** to return to the main menu.
- 8 Press **F6=run**.

9 Press **Enter**.

The program runs and displays your panel. Try entering data. You will find it behaves just as it did when you tested it using the Trap.

10 When you've finished, Press **Esc**.

Your program exists to the main menu.

3.3 Dialog

This section shows how to improve the screenset's functionality by defining the effect of particular keys.

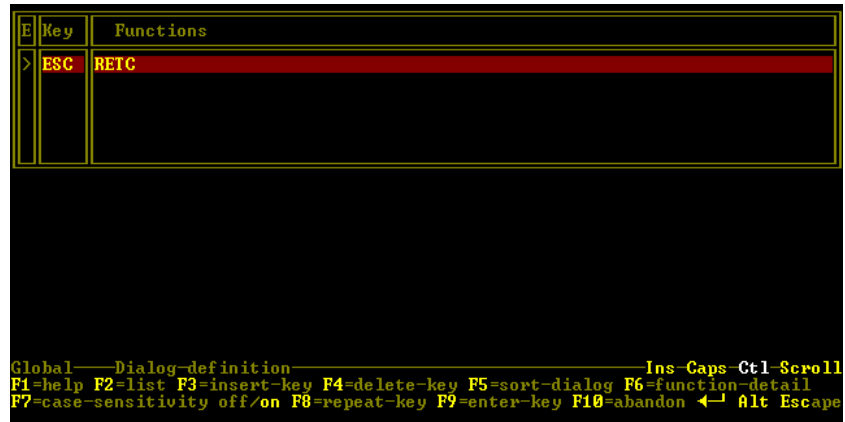
You do this by defining a dialog. The dialog defines procedures and/or functions to be executed when the user presses a key. Functions include functions to move between fields, cursor functions, and keyboard functions. Dialog can be local to a panel or global to the screenset. You can specify which of these types of dialog is to be resolved first.

For example, the next section describes the global dialog associated with the event generated when the **Esc** key is pressed. The global dialog defines the default behavior whenever **Esc** is pressed. However, if you define a different local dialog for **Esc** in one panel, this behavior normally overrides that of the global dialog. (You can change the order in which dialogs are resolved. You change the default setting using the screenset switches option reached from the main menu, **Alt+F5**.)

3.3.1 Defining Dialog

- 1 Go back into Dialog System and load the screenset file **test1**.
- 2 From the main menu select **F4=global-dialog**. The Dialog Definition menu appears, as shown in Figure 3-12.

Figure 3-12. The Dialog Definition Screen



There is one line already defined in the dialog. This defines that the event ESC (pressing the **Esc** key) will cause the action RETC (return to calling program). You have already seen the result of this definition, when you pressed **Esc** in the running screenset and it returned to the trap.

A dialog entry consists of an event (for example ESC) and one or more functions to be performed if the event occurs. A complete list of Dialog System functions can be found in the chapter *Function Code List*.

- 3 You will define some dialog so that the **Tab** key makes the cursor skip to the next field, and **Backtab** makes the cursor skip to the previous field.

The selection bar is currently in the top line of the dialog list. Press **Enter** to move it down one line.

- 4 The cursor is now on the second line. This is where you define the key that is pressed. The various keys on the keyboard have mnemonics associated with them. For example, **Esc** has the mnemonic ESC. To see the list of mnemonics, press **F2=list**.

A list appears, from which you can select the key that you want.

- 5 Move up and down this list using the arrow keys (the list scrolls when you reach the bottom) until the selection bar is on the mnemonic for **Tab**.
- 6 Press **Enter**. The mnemonic TAB appears in the key column and the list disappears.

Alternatively, you can simply type the mnemonic TAB yourself, or you can select **F9=enter-key** followed by the key you want (**Tab**).

- 7 Press **F6=function-detail**.

The Function Definition panel appears. This is where you define the function or functions to be executed when the **Tab** key is pressed. Each function consists of a function mnemonic and up to three parameters. The cursor is currently in the Function column.

- 8 With the cursor in the Function column, press **F2=list**.

This gives you a list of allowable functions. Move down this list until you find SKNF (the Skip to Next Field function). Select it by pressing **Enter**. The function name SKNF appears in the key column and the list disappears.

- 9 The cursor moves to the Parameter 1 column, where you are required to enter a parameter. The SKNF function requires a parameter saying how many fields to skip. In our case we only wish to skip one field, so type 0001 into the Parameter 1 column.

You could also use the name of one of your data fields for this parameter. The skip would then skip to next field as many times as the value in your specified data field at run time.

Many functions have parameters such as panel name or field name. When a parameter is needed, you are forced to enter it. Again, pressing the **F2** key provides you with a list to select from.

- 10 Press **Esc**. The Function Definition panel disappears, and you are ready to define the next key.

The SKNF mnemonic has been placed in the list beside TAB, but its parameters are not visible. This allows you to inspect the functions quickly without bothering about the details of the parameters.

- 11 Move down to the next line in the list by pressing **Enter**. Enter a dialog line to define that the **Backtab** key (BTAB) has the function Skip to previous field (SKPF). Enter a parameter of 0001 again.

- 12 Press **Esc** to exit the Function Definition Panel, then press **F5=sort dialog**.

Your dialog window should now be as shown in Figure 3-13. The order of the items in the dialog does not matter. You can re-order the items using **F3=insert-key** and **F4=delete-key** if you wish.

Figure 3-13. The Dialog You Defined

E	Key	Functions
	TAB	SKNF
	B TAB	S KPF
>	ESC	RETC

Global — Dialog-definition Ins-Caps-Ctrl-Scroll
 F1=help F2=list F3=insert-key F4=delete-key F5=sort-dialog F6=function-detail
 F7=case-sensitivity off/on F8=repeat-key F9=enter-key F10=abandon Alt Escape

(The > symbol shown against ESC means it is an event key.)

- 13 When your dialog is correct, press **Esc** to return to the main menu and then save the screenset using **Alt+F4**.
- 14 Now run the screenset again, as described in the section the section *Running Your Screenset* earlier. Try the effect of the **Tab** and **Backtab** keys.
- 15 When you have run the screenset, press **Esc** to return to the trap window and then again to get back to the main menu.

3.4 Attributes

The panel you have created is rather plain. If you are using a color screen, it would be good to improve the panel's visual effect. When you painted the panel earlier, you might have noticed that the status line on the Panel Painting menu contained the word Attribute. This refers to the color on a color screen, and the attributes of highlighting and underline on a monochrome screen.

In this section, we assume you are using a color screen. If you are using a monochrome screen, changes of color described in this section will appear as changes of highlighting and underlining.

- 1 From the main menu, go back to the Panel Painting Menu by pressing **F3=panels**, then **F3=panel-list**. Select the panel STOCK. The word Attribute on the status line is shown on the screen in the same color as the text that you typed onto the panel.
- 2 Press **F6=attribute-roll** several times and observe the effect on the word Attribute.

You can see that the color of the word changes. There are six different colors for the word. These colors make up your current attribute palette.

- 3 Roll to a color you have not yet used on the panel and then type some text on the panel. The text is typed in the current color.
- 4 Move the cursor to a text item and press **F5=paint-attribute** several times. The current color is painted onto the text.
- 5 Move the cursor to a data field and press **F5=paint-attribute** just once. The current color is painted onto the entire field.
- 6 Mark a block which includes both text and data, then press **F5**. This block must not partly cover any data fields. The entire block is painted with the selected attribute.

The current attribute palette contains six active attributes. One of the six is the default background attribute and the other five are called the roll attributes. These six attributes are preconfigured to behave reasonably on both a color and monochrome screen. You can change the attributes to any of the 256 found on your personal computer.

4 Using Dialog System

This chapter describes:

- The facilities available from the Main menu
- The facilities available from the Main Alt menu

4.1 Main Menu

The facilities described in the following sections are accessed from the Main menu (see the chapter *Getting Started* for the diagram of the menu tree).

4.1.1 Help (F1)

Pressing **F1=help** from the Main menu displays a help screen explaining the facilities available from the Main menu. To clear the help screen, press **F1=help** again or the **Space** bar.

4.1.2 Data Definition (F2)

Pressing **F2=data-definition** from the Main menu invokes the Data Definition menu, which you use to specify data fields (see the chapter *Data Definition* for further details).

4.1.3 Panels (F3)

Pressing **F3=panels** from the Main menu invokes the Panels menu, which you use to create and maintain panels (see the chapter *Panels* for further details).

4.1.4 Global Dialog (F4)

Pressing **F4=global-dialog** from the Main menu invokes the Dialog Definition menu, which you use to specify *global dialog*. Global dialog contains the procedures and functions that are active for the entire screenset (see the chapter *Dialog* for further details).

4.1.5 Generate (F5)

Pressing **F5=generate** from the Main menu invokes the Generate menu, which you use to generate the COBOL *copyfile*. The copyfile contains the Data Block, and field numbers of all the fields used in the Data Block, to be used by the calling program or a COBOL program. This copyfile, which is simply a file referenced in a COBOL COPY statement, enables the inclusion of COBOL source code from files other than the source code file currently being compiled (see the chapter *Generating the Copyfile* for further details).

4.1.6 Run (F6)

Pressing **F6=run** from the Main menu runs the current screenset (see the chapter *Running the Screenset* for further details).

4.1.7 Print (F7)

Pressing **F7=print** from the Main menu provides a list of options that you can use to print the current screenset to a text file (see the chapter *Printing* for further details).

4.1.8 Tutorial (F8)

Pressing **F8=tutorial** from the Main menu invokes an on-screen tutorial providing a brief overview of the features included in the Dialog System. You can exit from the tutorial at any time by pressing the Escape key.

This action results in the following prompt:

```
Exit from Tutorial. Are you Sure? Y/N
```

Press **Y** to exit from the tutorial and return to the Main menu or **N** to continue with the tutorial.

4.1.9 Alternate (Alt)

Pressing **Alt** from the Main menu invokes the Main Alt menu, the functions of which are described in the following section (see the chapter *Getting Started* for the diagram of this menu in the menu tree).

4.1.10 Escape (Esc)

Pressing **Esc** from the Main menu results in the following prompt:

```
Exit. Are you Sure? Y/N
```

Press **Y** to exit from the definition software to the operating system command line or **N** to continue your current session on Dialog System.

4.2 Main Alt Menu

Dialog System works on a single screenset at one time, so to save any data, panels, or dialog that you have defined, you must save them as a screenset. You can subsequently load that screenset to continue working with it. The facilities to do this, and the following other facilities, are accessed from the Main Alt menu:

4.2.1 Help (F1)

Pressing **F1=help** from the Main Alt menu provides a help screen explaining the facilities available from the Main Alt menu.

4.2.2 Initialize (F2)

Pressing **F2=initialize** from the Main Alt menu clears the current screenset, which contains definitions that you have created but no longer require. The system is then ready to start a new screenset.

When you press **F2** from the current screenset, the following prompt appears:

```
Are you Sure? Y/N
```

Press **Y** to clear the screenset, or **N** to cancel the initialization procedure.

4.2.3 Load Screenset (F3)

Pressing **F3=load-screenset** from the Main Alt menu invokes the Load Screenset menu to load a screenset file that you have previously created and saved. If you have worked on a screenset but not saved it, the following prompt appears when you select this option:

```
Load without saving? Y/N
```

Press **N** to return to the Main menu without loading another screenset. Press **Y** to invoke the Load Screenset menu (see Figure 4-1).

Figure 4-1. Load Screenset Menu



```
Load set _____Row: 01=Col: 01=Ins-Caps-Mem-Scroll
F1=help F2=directory
File E:\DS\DEMO\
Escape
Ctrl
```

At the prompt, type the name of the screenset to be loaded and press **Enter**. Alternatively, you can select the directory option, which enables you to select the screenset name from your current directory.

The **F2=directory** option displays the directory path, and extension of the listed files at the top of the screen. Directly beneath this line there is a list of existing screensets with file extension, date and time of creation, and size in bytes, together with the submenu shown in Figure 3-2.

Figure 4-2. Directory Menu

```

Default— E:\DS\DEMO\                               Asc-unsorted-Ins-Caps-Num-Scroll
F1=help      ←=select-file F2=list-files             F3=list-dirs F4=delete-file
F5=sort-name F6=sort-time F7=unsort F8=list-asc/desc F9=drive   Ctrl Escape
total size=210,522,112 available space= 6,873,088 listed files= 85,745

```

This Directory menu is available from several Dialog System menus.

If no drive or path is shown, the current default directory is listed. If no extension is specified, the directory lists only the files that can be used at the point from which you entered the directory. For example, if you were loading a source file into the editor, the directory lists source files only.

UNIX: Figure 4-2 shows a Windows-format path on the status line. On UNIX, the drive designation, F:, does not appear and backslashes (\) are replaced by slashes (/).

4.2.3.1 Select File (Enter)

Pressing **Enter** from the Directory menu enables you to select a file from the directory listing for loading. Use the <up-arrow> and <down-arrow> keys to highlight the required file, then press **Enter**. The filename is displayed at the prompt on the Load Screenset menu. Press **Enter** again to load the screenset.

4.2.3.2 List Files/List Library Catalogue (F2)

Pressing **F2=list-files** from the Directory menu redisplay the directory, or, if the highlighted file is a library, it lists the files contained in the library with an **.lbr** extension.

4.2.3.3 List Directories (F3)

Pressing **F3=list-dirs** from the Directory menu lists the subdirectories in the current directory.

4.2.3.4 Delete File (F4)

Pressing **F4=delete-file** from the Directory menu displays the following prompt:

```
Delete filename? Y/N/Esc
```

where:

filename is the name of the selected file.

Press **Y** to delete the file, and **N** or **Escape** to cancel the delete operation.

4.2.3.5 Sort Name (F5)

Pressing **F5=sort-name** from the Directory menu sorts the filenames of the files in the directory list into alphabetical order.

4.2.3.6 Sort Time (F6)

Pressing **F6=sort-time** from the Directory menu sorts the files in the directory list into chronological order.

4.2.3.7 Unsort (F7)

Pressing **F7=unsort** from the Directory menu shows the files in the directory list in the order in which they were created.

4.2.3.8 List Ascending/Descending (F8)

Pressing **F8=list-asc/desc** from the Directory menu shows the files in the directory in ascending or descending order, according to the sorting option chosen.

4.2.3.9 Drive (F9)

Pressing **F9=drive** from the Directory menu invokes a prompt for a drive letter, then lists files for the current directory in the drive you specify.

Ensure that there is a disk or tape in the drive you specify before you use this option, otherwise there is a flashing error message. If you get this error message, insert a disk or tape and select the **retry** option to continue.

4.2.3.10 Control Menu (Ctrl)

Pressing **Ctrl** from the Directory menu invokes the Directory Control menu as shown in Figure 4-3. This menu offers the sorting options and browse-file facility explained as follows:

Figure 4-3. Directory Control Menu

```

Default— E:\DS\DEMO\          Asc-unsorted—Ins—Caps—Num—Scroll
F1=help
F5=sort-size F6=sort-ext      F8=browse-file
total size=210,522,112 available space= 6,873,088 listed files= 85,745

```

4.2.3.10.1 Sort Size (F5)

Pressing **F5=sort-size** from the Directory Control menu sorts the files in the directory in size order.

4.2.3.10.2 Sort Extension (F6)

Pressing **F6=sort-ext** from the Directory Control menu sorts the files in the directory in alphabetic order of their file extension. Files with the same extension are further sorted in order of filename.

4.2.3.10.3 Browse File (F8)

Pressing **F8=browse-file** from the Directory Control menu displays the contents of the highlighted file (the system assumes that the highlighted files are in displayable text format).

4.2.3.11 Escape (Esc)

Pressing **Esc** from the Directory menu exits back to the file prompt on the Load Screenset menu.

4.2.4 Save Screenset (F4)

Pressing **F4=save-screenset** from the Main Alt menu invokes the Save screenset menu (see Figure 4-4) to save a screenset file that you create. This menu contains the standard help option and also the Directory menu described earlier in the section *Load Screenset (F3)*.

Figure 4-4. Save Screenset Menu

```
Save set-----Row:01=Col:01=Ins-Caps-Nun-Scroll
F1=help F2=directory      Escape
File E:\DS\DEMO\         ← Ctrl
```

You can use this menu to copy a screenset. Load the screenset and save it with a different name. This does not affect your original screenset.

4.2.5 Screenset Switches (F5)

Pressing **F5=screenset-switches** from the Main Alt menu enables you to alter the current screenset settings, such as the priority of global or local dialog. The submenu shown in Figure 4-5 is displayed when you select this option.

Figure 4-5. Screenset Switches Menu

```
New set-----Row:01=Col:01=Ins-Caps-Nun-Scroll
F1=help F2=normal/MF-panels F3=global/local dialog 1st F4=key-translation off/on
Escape
```

4.2.5.1 Normal / Micro Focus Panels (F2)

Pressing **F2=normal-MF-panels** from the Screenset Switches menu determines whether Dialog System uses its own internal coding (**normal**) or Micro Focus Panels to manipulate the panels in a screenset at run time. This is an attribute of the screenset and is saved when the

screenset is saved. You can change this option at any time during the definition process.

Whether or not you use a Panels screenset depends on your application. The advantage of using a Panels screenset is that these panels can be moved at run time using dialog (using the MOVEPNL function described in the chapter *Dialog*). The disadvantages are that these panels occupy more memory at run time than normal panels, and that you are limited to 50 panels per screenset compared with the 500 allowed for normal panels screensets. The size of the screenset itself is not affected.

Default: Normal panels.

4.2.5.2 Global / Local Dialog First (F3)

Pressing **F3=global/local-dialog-1st** from the Screenset Switches menu determines the default order in which the dialog for the screenset is obeyed at run time. If the order is set to local first, the dialog for a panel is processed at run time by searching the local dialog first, obeying any keystrokes or procedures defined there and ignoring any definition of those keystrokes or procedures in the global dialog. If a keystroke or procedure is not defined in the local dialog, its definition in the global dialog is obeyed. If the order is set to global first, the reverse is true. See the chapters *Panels* and *Dialog* for details of the local and global dialog facilities.

Default: Local dialog first.

4.2.5.3 Key Translation Off / On (F4)

Pressing **F4=key-translation-off/on** from the Screenset Switches menu specifies whether Dialog System's run-time component looks for a module that performs key translation (Dsusrtrn) when this screenset is run (see the section *User Control of Every Keystroke* in the chapter *Programming* for more details on this module).

4.2.6 Normal / Top / Bottom Coordinates (F6)

Pressing **F6=normal/top/bottom coords** from the Main Alt menu determines the type of coordinates that are displayed on the status line. Pressing **F6** toggles between the following settings:

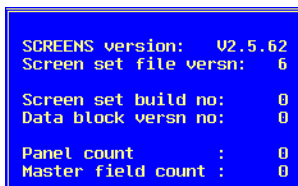
- | | |
|--------|--|
| Normal | When you are in a panel, the displayed coordinates have as their origin the top left-hand corner of the panel. When you are not in a panel, the displayed coordinates have as their origin the top left-hand corner of the screen. |
| Top | The displayed coordinates always have as their origin the top left-hand corner of the screen. |
| Bottom | The displayed coordinates always have as their origin the bottom left-hand corner of the screen. |

Default: Normal.

4.2.7 Set Details (F7)

Pressing **F7=set-dets** from the Main Alt menu displays a panel, similar to that shown in Figure 4-6, which contains the current screenset's details. This is a display feature only.

Figure 4-6. Set Details Screen



```

SCREENS version:  U2.5.62
Screen set file versn:  6

Screen set build no:  0
Data block versn no:  0

Panel count      :  0
Master field count :  0
  
```

4.2.8.2 Map Screenset Colors Map to COBOL System (F3-F9)

Pressing one of the keys **F3** to **F9** from the Colorize menu maps the screenset colors to the COBOL system colors SYS-1 to SYS-7. Colorization is only active if the F2 toggle is set to on.

UNIX: On UNIX, to assign an attribute from the palette of 255 attributes to one of these function keys, use the <right-arrow> and <left-arrow> keys to move the "X" located on the palette to the desired attribute and press the required function key.

As you scroll the "X" through the palette, the two numbers in parentheses above the palette line change. The first number is the attribute's palette number (1-255) and the second number is its hexadecimal value. When the "X" reaches the far right of the screen, the palette wraps, replacing the current row of attributes with the next selection.

4.2.8.3 More (F10)

Pressing **F10=more** from the Colorize menu invokes the second colorize menu (see Figure 4-8), which you can use to map the system colors SYS-8 to SYS-16.

Figure 4-8. Second Colorize Menu

```
Colorize-----Row: 01=Col: 01=Ins=Caps=Nur=Scroll
F1=help F2 (SYS-8)=X F3 (SYS-9)= F4 (SYS-10)=X F5 (SYS-11)=X F6 (SYS-12)=X
F7 (SYS-13)= F8 (SYS-14)= F9 (SYS-15)= ↓ (007/07h) F10 (SYS-16)=X Escape
XXXXXXXXXX XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
```

4.2.9 Import (F9)

Pressing **F9=import** from the Main Alt menu invokes the Import Files menu, shown in Figure 4-9, which enables you to import screenset text files.

Figure 4-9. Import Files Menu



```

Import file-----Row:01-Col:01=Ins-Caps-Nun-Scroll
F1=help F2=directory      Escape
File E:\DS\DEMO\         ← Ctrl
  
```

Dialog System can transfer the components of a screenset, for example the Data Block, panel text and panel attributes, between two different screensets using the import and export utilities. The definition of the screenset components is held in a text file. The import utility extracts the information from such a text file to insert new components into a screenset (see the chapter *Syntax of Import/Export Files* for details on the structure of the text file). It is possible to create your screens in this format (using a method other than export, for example where you are converting from some other application to Dialog System) and then import them into your screenset.

At the prompt, type the name of the screenset to be imported and press **Enter**. You can alternatively use the Directory menu described earlier in this chapter.

After you select a file, the import of the file begins. The import process consists of two stages, described in the following sections:

- 1 syntax checking
- 2 semantic checking

4.2.9.1 Syntax Checking

The import file is checked for valid syntax (see the chapter *Syntax of Import/Export Files* for further details). If any invalid syntax is found, the process is aborted with an error message similar to the following:

```
FATAL ERROR:LINE 0085 After Token: END Esc: To ABORT.
```

The error message gives the line number and the last token of syntax that was successfully parsed in the import file. To end the import process when invalid syntax has been found, press the **Escape** key.

4.2.9.2 Semantic Checking

After syntax checking, the imported details are inserted into the current screenset and the imported information is checked for conditions such as conflicts with existing components in the screenset, or reaching screenset limits. If any errors are found, either recoverable or fatal error messages, as appropriate, are displayed.

Recoverable error messages provide the option to continue with the process. For example, if a field length is too large, the following recoverable error message appears:

```
ERROR Maximum field-length is 9999 Continue Y/N?
```

If you select Y to continue, corrective action is automatically taken. In this example, a field length of 9999 would be used.

For a list of recoverable and fatal error messages that could occur when you import an external file, see the chapter *Error Messages*.

4.2.10 Export (F10)

Pressing **F10=export** from the Main Alt menu invokes the Export Files menu, shown in Figure 4-10, which enables you to export screenset text files.

Figure 4-10. Export Files Menu

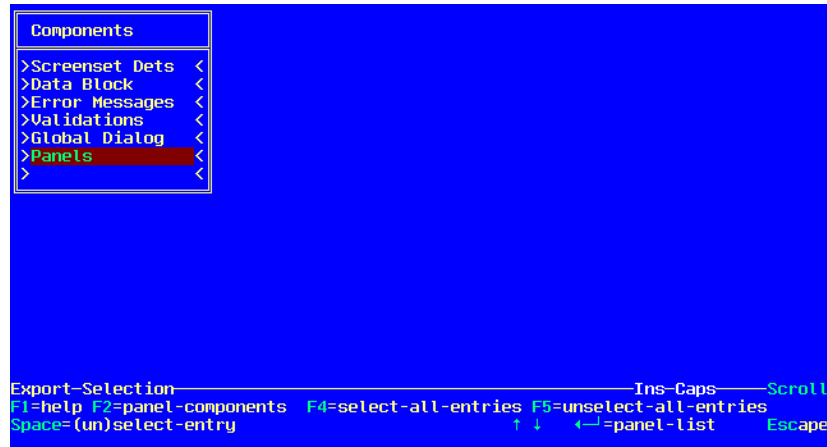


```
Export file                                     Row:01=Col:01=Ins-Caps-Num-Scroll
F1=help F2=directory
File E:\DS\DEMO\
Escape
← Ctrl
```

Dialog System can transfer the components of a screenset, for example the Data Block, panel text, and panel attributes, between two different screensets using the import and export utilities. The export utility creates a text file containing the definition of these screenset components (see the chapter *Syntax of Import/Export Files* for details on the structure of the text file). It is possible to create your screens in this format.

At the prompt, type the name of the screenset to be exported and press **Enter**. You can use the Directory menu described previously in this chapter. If you do not specify a name, the name **work.txt** is used. After you select a filename, a popup component list and the Export Component Selection menu appear (see Figure 4-11).

Figure 4-11. Export Component Selection Menu



The popup list contains the following components:

Screenset Details	Screenset switches and colorize palette
Data Block	Data Block master fields & groups
Error Messages	Error messages and error message filename
Validations	Validations per master field
Global Dialog	Dialog
Panels	Text, attributes, fields, groups, dialog

Use the functions in the Export Component Selection menu to select which components you want to export then create the export file or invoke the Export Panel Selection menu.

4.2.10.1 Panel Components (F2)

Pressing **F2=panel-components** displays the panels component list. This function is available only when the selection bar is positioned on the Panels component and the Panels component is selected. Select the panels components you want to export in the same way as the main components.

The panels component list contains the following components:

Text and Attributes
Fields and Groups
Dialog

4.2.10.2 Select All Entries (F4)

Pressing **F4=select-all-entries** from the Export Component Selection menu selects every component in the component list for output to the export file. All selected components are marked with the greater-than (>) and less-than (<) symbols.

4.2.10.3 Unselect All Entries (F5)

Pressing **F5=unselect-all-entries** from the Export Component Selection menu deselects all the components in the component list.

4.2.10.4 Select / Unselect Entry (Space Bar)

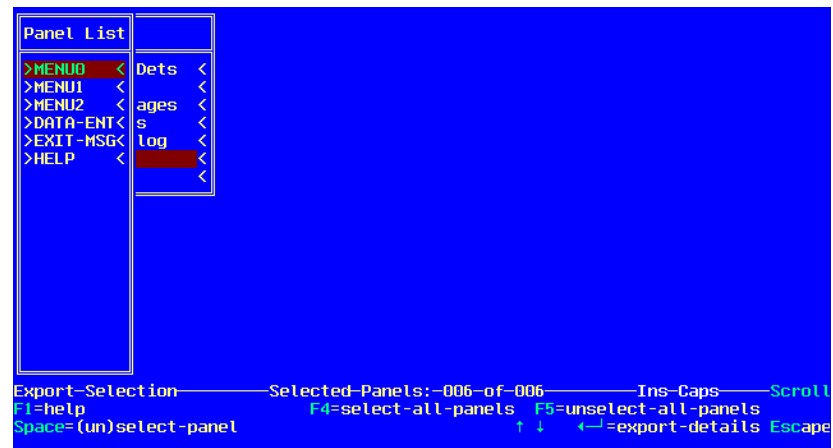
Pressing **Space=(un)select-entry** from the Export Component Selection menu enables you to select the components that you wish to export by selecting or deselecting individual components. Use the <up-arrow> and <down-arrow> keys to move up and down the list to the desired component name, and the **Space** bar to toggle the component between selected and deselected. Selected components are marked with the > and < symbols.

4.2.10.5 Panel List (Enter)

Pressing **Enter** from the Export Component Selection menu either creates the export file, or displays a further menu. Use this function after you have selected all the components you want to export. If you have not selected the Panels component, the export file is created. If you have selected the Panels component, the popup panel list and the Export Panel Selection menu, shown in Figure 4-12, appear.

You use the Export Panel Selection menu to select the panels that you want to export and create the export file.

Figure 4-12. Export Panel Selection Menu



4.2.10.6 Select All Panels (F4)

Pressing **F4=select-all-panels** from the Export Panel Selection menu selects every panel in the panel list for output to the export file. All selected panels are marked with the > and < symbols.

4.2.10.7 Unselect All Panels (F5)

Pressing **F5=unselect-all-panels** from the Export Panel Selection menu deselects all the panels in the panel list. This is useful if you want to export only the Data Block.

4.2.10.8 Select / Unselect Panel (Space Bar)

Pressing **Space=(un)select-panel** from the Export Panel Selection menu enables you to restrict what is exported by selecting or deselecting panels. Use the <up-arrow> and <down-arrow> keys to move up and down the list to the desired panel name, and the **Space** bar to toggle the panel between selected and deselected. Selected panels are marked with the > and less-than < symbols.

4.2.10.9 Export Details (Enter)

Pressing **Enter** from the Export Panel Selection menu creates the export file. Use this function after you have selected all the panels you want to export.

4.2.10.10 Import Limitations

Import is not an effective way to update a screenset, because if you import components into a screenset that already contains identically named components, those components are not updated. For example, you cannot change values such as the length, usage, or properties of a master field by importing an identically named field with new values. The best way to update a screenset is to use the following steps:

- 1 Load the screenset.
- 2 Export all components of the screenset to an export file.
- 3 Edit the file to make required changes.
- 4 Import the file into an empty screenset to obtain an updated version of the screenset.
- 5 Save the screenset.

The following table lists the limitations in this version of Dialog System:

Component/ sub-component	Insert	Update	Comments
Screenset Details			
First Panel	n/a	Yes	(exist by default
Screenset Type	n/a	Yes	(and thus can only
Key Translation Flag	n/a	Yes	(be updated
Colorization Palette	n/a	Yes	
Data Block			
Master Field	Yes	No	
Master Group	Yes	No	
Error Messages	Yes	Yes	
Validations			
Null	Yes	Yes	
Range-Table	Yes	Yes	
Check Digit	Yes	Yes	
Date	Yes	Yes	
Global Dialog			
Key Entry (i.e. Key+Ftns+Params)	Yes	Yes	see note below
Panel			
Attribute Palette	n/a	Yes	inserted with the panel
Border	n/a	Yes	inserted with the panel
Text/Attributes	Yes	Yes	
Field	Yes	No	no insert to existing group
Group	Yes	No	
Select Bar	n/a	No	bar is inserted with group

Component/ sub-component	Insert	Update	Comments
Dialog			
Key Entry	Yes	Yes	

Note: The following keys cause problems with the import parser, and cannot currently be imported:

[(left square bracket)
] (right square bracket)
" (double quote)
^ (caret)

Although you cannot import these keys, you can use them in a dialog. You must remove these keys from a dialog before you export it, and add them manually to the dialog after you have imported it.

5 Data Definition

The first step when you develop your Dialog System application is to define the Data Block for your screenset. The Data Block contains all the fields to be displayed in an application or to be entered on the various panels that make up a screenset, and is passed as a parameter from your application program. So, to place a field on a panel in the screenset, you must first define that field in the Data Block.

This chapter describes:

- How to define data fields for subsequent use on panels
- How to specify validation for data fields
- How to create an error message file to identify data validation errors occurring in screensets at run time

To access the data definition function, press **F2=data-definition** from the Main menu. When you select the data definition function, a further menu providing options for data field definition or error message definition is displayed (see Figure 5-1).

Figure 5-1. Data Definition Menu



```

New set-----Row: 01-Col: 01-Ins-Caps-Nur-Scroll
F1=help F2=data-fields F3=error-messages      Escape
  
```

You can designate one data field as the *error message field*; this is the field in which error messages appear if validation errors are found at run time. You can specify this field at any stage when you are developing your dialog; you do not have to specify it at the same time as you define your Data Block.

5.1 Data Field Definition (F2)

Press **F2=**data-fields from the Data Definition menu to select this option, and the menu shown in Figure 5-2 is displayed with a popup list of fields above it. When you access this function for the first time for a new screenset, the list of fields is empty. When you access this function and there are some fields already defined, you can scroll up and down the list, and insert, change, or delete fields in this list, using the appropriate function keys shown in the Data Field Definition menu. The maximum number of data fields in a screenset is 512.

Figure 5-2. Data Field Definition Menu

Field	Fmt	Int. Dc	Comp	Rpts	Val
		0.0		0	

Data-Definition — Ins-Caps — Scroll
 F1=help F2=amend-grp-size F3=ins-field F4=del-field F5=amend-grp-rpts F6=def-grp
 F7=undef-grp F8=validation F9=err-nsg-flt def/undef ← ↑ ↓ PgUp PgDn Escape

You use the data field list to define the fields that are used for data entry in your application. Also, you can arrange selected fields that repeat, for example an address field, into groups.

The following data details can be entered on this list:

- Field The name of the field. The name can be up to 30 characters long and must conform to COBOL rules on data names. The name must not be blank.
- Fmt The format of the named field. The following formats are available:

A	Alphabetic field . This format can be any length.										
X	Alphanumeric field. This format can be any length.										
9	Numeric unsigned field . This format must not exceed 18 digits for the integer and decimal parts in total, with the decimal part not exceeding 9 digits.										
S	Numeric signed field . This format must not exceed 18 digits for the integer and decimal parts in total, with the decimal part not exceeding 9 digits.										
Int.Dc	<p>The field length. You must make an entry in this field. Enter a value of the form n.m, omitting .m for character fields or numeric fields with no decimal part.</p> <p>If you define a field as format X, COMP-X, the value you enter here should be the number of bytes occupied by the field.</p>										
Comp	<p>The field is a numeric computational field. The following formats are supported:</p> <table border="0" style="margin-left: 20px;"> <tr> <td>spaces</td> <td>The field is not COMP.</td> </tr> <tr> <td>C</td> <td>COMP field</td> </tr> <tr> <td>C3</td> <td>COMP-3 field.</td> </tr> <tr> <td>C5</td> <td>COMP-5 field. A format X COMP-5 field is assumed to be unsigned.</td> </tr> <tr> <td>CX</td> <td>COMP-X field. Signed COMP-X fields are not allowed.</td> </tr> </table> <p>COMP-5 and COMP-X fields can be defined as format 9 or X. If you specify format X, this is automatically converted to 9 on leaving the definition line.</p> <p>For example, PIC X COMP-X is identical to PIC 9 COMP-X or PIC 99 COMP-X. In COBOL terms, all these definitions say that the data is binary, occupies one byte, and can hold a value up to 255.</p>	spaces	The field is not COMP.	C	COMP field	C3	COMP-3 field.	C5	COMP-5 field. A format X COMP-5 field is assumed to be unsigned.	CX	COMP-X field. Signed COMP-X fields are not allowed.
spaces	The field is not COMP.										
C	COMP field										
C3	COMP-3 field.										
C5	COMP-5 field. A format X COMP-5 field is assumed to be unsigned.										
CX	COMP-X field. Signed COMP-X fields are not allowed.										
Rpts	This field indicates the number of occurrences for the field(s) contained in a data group. The single-character field preceding the field name contains a "[", which encompasses those fields contained in the group. These are display fields only; there is a separate facility provided to define groups and the number of times they repeat.										

Val This field indicates either that validation has been set up for the named field ("V"), using the Define Validation Details facility, or that the named field has been defined as the error message field ("E"). This is a display field only; there is a separate facility provided to define these validations.

When you access the Data Field Definition function, the cursor appears on the first line of the list of data fields. For a new screenset, you can simply type in the appropriate data, using **Tab** to move from the **Field** column to the **Fmt** column. The cursor automatically skips from the **Fmt** column to the **Comp** column. The **Rpts** column is displayed, but you cannot access it. To access this column, you use the Define Group function described later in this chapter. When you have finished entering the details for a data field, press **Enter** to move down to the next line and repeat the process to specify the details for another data field. There is a maximum of 512 data fields per screenset.

To change an existing data field, overtype the entries for that field. You cannot change certain field attributes if that data field is already being used on a panel, or if validation rules have been assigned to it. If you try to change an attribute and cannot, check that the data field is deleted from all panels and that the validation rules have been deleted.

Once you have defined fields for a screenset, you can scroll to the desired field and use the facilities on the Data Fields Definition menu described in the following section. When you have finished defining your data fields and any validations desired, press the **Escape** key to return to the Data Definition menu.

5.1.1 Amend Group Size (F2)

Pressing **F2=amend-group-size** from the Data Fields Definition menu displays a submenu (see Figure 5-3) where you can alter the number of data fields that are contained in a previously defined group (details on defining data groups are provided later in this chapter). The submenu also offers the standard help facility.

Figure 5-3. Amend Group Size Menu

```

Data-Definition-----Ins-Caps-----Scroll
F1=help F2=expand/contract group
                                     ↑ ↓
                                     ←=complete  Escape
  
```

You can expand or contract the group. First select the appropriate mode by pressing **F2=expand/contract group** to toggle between expand and contract. To expand the group, press <down-arrow> for the required number of lines; to contract the group, press <up-arrow> to move the cursor to the desired entry. The "[" expands and contracts to encompass those fields contained in the group. Press **Enter** to complete the process. To abandon any changes made so far, press the **Escape** key.

5.1.2 Insert Field (F3)

Pressing **F3=ins-field** from the Data Fields Definition menu enables you to insert a new field at the current selection bar position. If the selection bar is currently in a group, the new field is also included in that group.

To insert a new field at the end of a group, insert a normal non-group item outside the group and then extend the group using the Amend Group Size function.

5.1.3 Delete Field (F4)

Pressing **F4=del-field** from the Data Fields Definition menu deletes the field at the selection bar's current position. You cannot delete a field if the field is in use in any of the panels or in the dialog (unless you have turned on the auto-comment feature; see the chapter *Panel Painting* for details).

5.1.4 Amend Group Repeats (F5)

Pressing **F5=amend-grp-rpts** from the Data Fields Definition menu displays the menu shown in Figure 5-4, which enables you to change the number of repeats of a previously defined group.

Figure 5-4. Amend Group Repeats Menu

```

Data-Definition-----Ins-Caps-----Scroll
FI=help      Enter number of repeats      ↑ ↓      ←=complete  Escape
  
```

You can always increase the number of repeats in a group, up to a maximum of 999. However, if a group is being used in a panel, you can only decrease the number of repeats to the number defined when the group was added to the panel. If the group is not in use in any of the panels, you can decrease the number of repeats to any number.

To change the number of repeats in a group, from the Data Field Definition menu, position the cursor on any field in that group and press **F5**. At the prompt, enter the number of repeats you want.

5.1.5 Define Group (F6)

Pressing **F6=def-grp** from the Data Fields Definition menu displays the same menu as shown in Figure 5-4, which enables you to define selected fields as repeating groups once a number of fields have been entered. You are prompted to enter the number of repetitions, up to a maximum of 999, for the items in the highlighted group.

Once you have specified the number of repetitions, you can include extra fields in the group by pressing **<up-arrow>** or **<down-arrow>** to highlight the desired fields. When you have finished defining your group, press **Enter** to execute the changes. To abandon any changes made so far, press the **Escape** key.

5.1.6 Undefine Group (F7)

Pressing **F7=undef-grp** from the Data Fields Definition menu enables you to delete a group definition, making any fields in this group non-group items again. However, you cannot undefine a group if the selected group has been used in any panels.

5.1.7 Define Validation Details (F8)

Pressing **F8=validation** from the Data Fields Definition menu displays the Validation menu, an example of which is shown in Figure 5-5. This menu enables you to enter validation details against the current field.

Figure 5-5. Validation Menu

Field	Fmt	Int. Dc	Comp	Rpts	Val
C-CODE	X	5.0		0	
C-NAME	X	15.0		0	
C-ADDR1	X	15.0		0	
C-ADDR2	X	15.0		0	
C-ADDR3	X	15.0		0	
C-ADDR4	X	15.0		0	
C-LIMIT	S	4.0		0	
C-AREA	X	1.0		0	
ORD-NO	S	6.0		10	
ORD-DATE	9	6.0		10	
ORD-VAL	9	4.2		10	
PAY-VAL	S	4.2		10	
ORD-BAL	S	4.2		10	
C-BAL	S	5.2		0	
DEL-FLG	9	1.0		1	

Validation
 F1=help F2=delete-all-validations F3=range/table
 F7=user-validation

Ins-Caps Scroll
 F6=null
 Escape

To define data validation, there are four stages:

- 1 Define validation rules for each of the data fields to be validated.
- 2 Define suitable error messages.
- 3 Associate the appropriate error messages with each validation rule.
- 4 Define a data field to receive the error messages when validation is performed.

The functions displayed on this menu vary depending on the current field type; the menu enables you to define only details appropriate to validate that field type. For example, if the current field is a single-digit field, the menu does not show the date validation key, because a date cannot be defined in a single-digit field.

The validations that are currently active for the chosen field are displayed on the status line. When you have defined validation against a particular field, the letter "V" appears in the **Val** column.

The validations you specify are applied at run time whenever that field is used for input. That means that the validation is performed when the cursor enters a field and a subsequent attempt is made to move the cursor out of the field. To validate all input fields on a panel at run time, whether or not the cursor has entered a field, there are two methods:

- Use the Dialog function VAL (see the chapter *Dialog* for further details of the VAL facility).
- Use an option in the Panel Painting menu that delays validation until just before control is returned to the user (see the chapter *Panel Painting* for details).

At run time, certain validation checks are activated automatically upon data entry because of either the data definition format or the screen format. For example:

- Numeric fields can contain only numeric data.
- Alphabetic fields can contain only alphabetic data or spaces.
- Required fields cannot be left empty if the cursor goes to that field.

These automatic validations do not result in any error messages. If you require error message handling, you must specifically define validation using the facilities provided in this menu.

5.1.7.1 Delete All Validations (F2)

Pressing **F2=delete-all-validations** from the Validation menu enables you to delete all validations associated with the current field.

This function is useful when you want to change the format of a field, because you cannot change the format of fields that are validated. This feature of the data definition facility ensures that the validations specified remain appropriate to the field. For example, if you assign validations to an alphanumeric field, then later want to change that field to a numeric field, the defined validations would not make sense. Therefore, you must delete all of the validation details before the format of a field can be changed.

5.1.7.2 Range / Table Validation (F3)

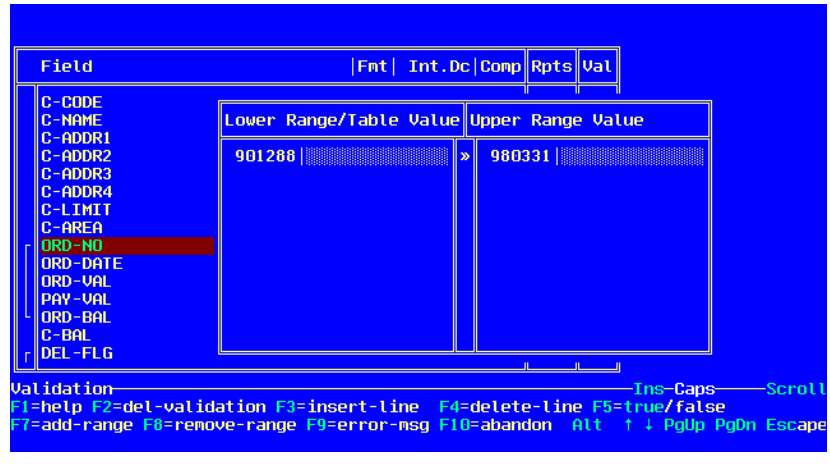
Pressing **F3=range/table** from the Validation menu enables you to enter the upper and lower values for multiple ranges for numeric, alphabetic, and alphanumeric fields, and a table of entries for all field types. You can enter these values and entries in any order.

You can specify range validation for date fields of all formats except DDMMYY. When you enter a date value in the range table, you must enter it in the order year, month, day regardless of the actual date format of the field. For example, to define range validation for a date field defined as DDMMYY and verify that the user enters a date between 10 01 91 (10 Jan 1991) and 06 04 93 (6 Apr 1993), you need the following values in the range table:

```
910110
930406
```

When you choose the **F3=range/table** option, the screen shown in Figure 5-6 is displayed. You set any validation ranges for a field and add any table entries for a field using the appropriate functions from the menu, as described in the following section. By default, you enter validation values in the left-hand column of the table and they are treated as table values, unless you enter a value in the right-hand column, when the pair of values are treated as upper and lower values of a range. You can list a maximum of 30 range/table validations for each field.

Figure 5-6. Range/Table Validation Menu



5.1.7.2.1 Delete Validation (F2)

Pressing **F2=del-validation** from the Range/Table Validation menu deletes all the range/table validations for the current field.

5.1.7.2.2 Insert Line (F3)

Pressing **F3=insert-line** from the Range/Table Validation menu inserts a blank line at the selection bar's current position.

5.1.7.2.3 Delete Line (F4)

Pressing **F4=delete-line** from the Range/Table Validation menu deletes the line at the selection bar's current position.

5.1.7.2.4 True/False (F5)

Pressing **F5=true/false** from the Range/Table Validation menu enables you to choose whether the entry in a field is validated for being within the defined ranges, including the upper and lower values, or being one of the table entries (true), or for not being within the ranges or not being one of the table entries (false).

Default: True

5.1.7.2.6 Hexadecimal Input (F6)

Pressing **F6=hexadecimal-input** from the Range/Table Validation menu enables you to enter the range and table values in hexadecimal form. This option applies only to alphanumeric fields.

5.1.7.2.7 Add Range (F7)

Pressing **F7=add-range** from the Range/Table Validation menu moves the cursor into the right-hand column to enter an upper range value.

5.1.7.2.8 Remove Range (F8)

Pressing **F8=remove-range** from the Range/Table Validation menu deletes the right-hand upper range value and returns the cursor to the left-hand column of the table.

5.1.7.2.9 Error Message (F9)

Pressing **F9=error-msg** from the Range/Table Validation menu enables you to assign an error message number to the range/table validation. If the error message file has been defined for this screenset and is present in the current disk directory, you can select from the list of currently defined error messages. You can also use a number that is not in the list and define the error message later if required. To do this, enter the error message number you want at the following prompt:

```
Enter error number [000] then Enter
```

If you specify error number 000 for a validation, no error message is displayed, but the system beeps.

At run time, if any range or table validation error occurs on a field where you have assigned an error message, the appropriate error

message is placed into the field defined as the error field, and the error message number is placed in the control block. If the error field is present on the current panel, the error message is displayed. When the error is corrected, the message field is automatically cleared and the error number in the control block is changed to zero.

There is a further facility, the ERR key code, to specify in your dialog that an error procedure is performed, for example a procedure to display an error panel. If you use this ERR key code, the functions defined with the ERR are performed whenever a validation error occurs (see the chapter *Dialog* for further details of this facility).

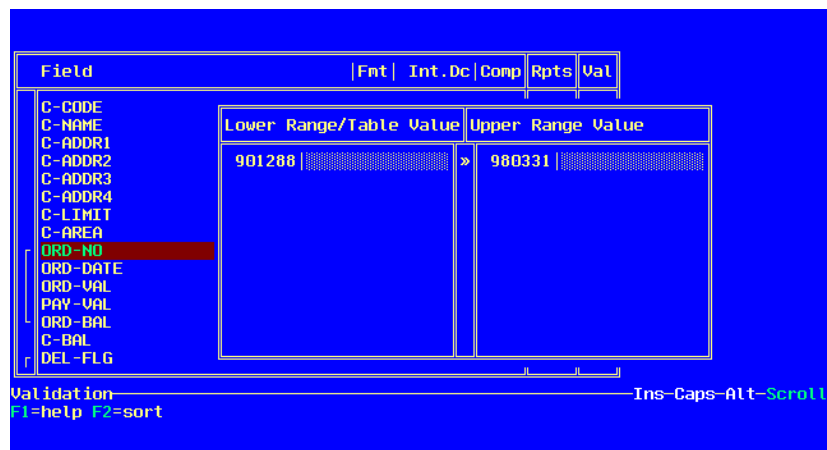
5.1.7.2.10 Abandon (F10)

Pressing **F10=abandon** from the Range/Table Validation menu cancels the definition or range/table validations and returns to the Validation menu.

5.1.7.2.11 Alternate Menu (Alt)

Pressing **Alt** from the Range/Table Validation menu invokes the Range/Table Validation Alternate menu, shown in Figure 5-7.

Figure 5-7. Range/Table Validation Alt Menu

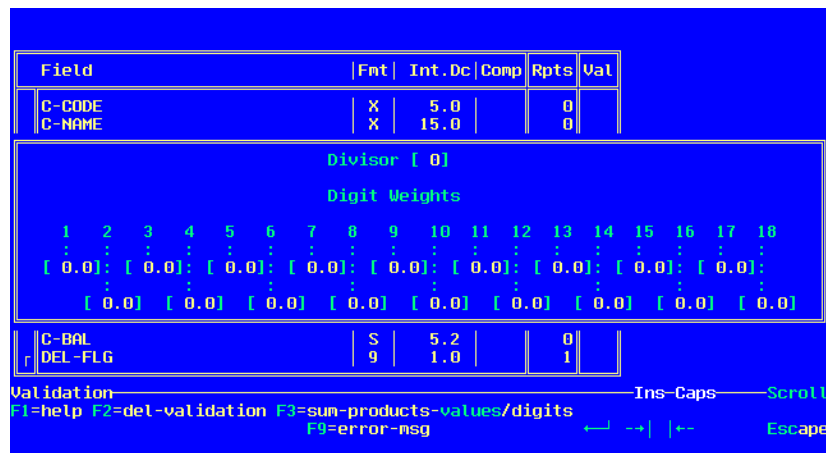


This menu provides the standard Help facility and a Sort facility. The list is sorted automatically when you exit from range/table validation, but this option enables you to sort the range/table validations into ascending sequence before you exit. It also tidies up and merges any overlapping validations. Press **F2=sort** from this menu to use this facility.

5.1.7.3 Check Digit Validation (F4)

Pressing **F4=check-digit** from the Validation menu displays the Check Digit Validation panel and menu as shown in Figure 5-8. You can specify check digit validation only for numeric fields that have no decimal places.

Figure 5-8. Check Digit Validation Menu



The panel contains the following fields:

- Divisor** This is the divisor to be used by the check digit system. It can be any non-zero, two-digit number.
- Digit Weights** These are the weights to be used, starting with the most significant digit as digit one.

From this menu you can enter the divisor and the weights associated with the check digit routine. Enter the required divisor and press

<right-arrow>| to move the cursor to the Digit Weights fields for entry. Press |<left-arrow> to move back to the Divisor field if required. Press **Enter** to complete your selection. Remember that the divisor must not be zero.

The functions available from the Check Digit Validation menu are described in the sections following this section.

The method of calculating whether a check digit is valid or not is illustrated by the following example.

Example 1

There is a 4-digit number 6843. The defined weights are 2, 3, 4, and 5. The divisor is 11.

The following calculation is performed:

- $(6*2) = 12$
- $(8*3) = 24$
- $(4*4) = 16$
- $(3*5) = 15$

The default method is to add the results of the calculation to give:

- $12 + 24 + 16 + 15 = 67$

An alternative method is to add the individual digits in the results of the calculation to give:

- $(1 + 2) + (2 + 4) + (1 + 6) + (1 + 5) = 22$

The result of the addition is divided by 11. If the remainder is zero, the number is considered valid.

Therefore, in this example, the first method gives an invalid check digit, the second method gives a valid check digit.

Example 2

This is a more detailed example of a check digit validation, taken from a real check digit system used by one of the major credit card companies.

The rules of this system are as follows:

- 1 Numbers are 13 digits
- 2 Weights are 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1
- 3 Multiply digits by weights
- 4 If any resulting value is greater than 9, then add 1 to it
- 5 Sum these values
- 6 The divisor is 10
- 7 If the remainder when dividing by 10 is zero, then the number is valid.

None of these rules presents any problem to our system with the exception of Rule 4. However, this is easily handled by allowing non-integer weights with rounding on the product calculation.

Take the example of the number 5929734106706:

5 9 2 9 7 3 4 1 0 6 7 0 6

with weights

1 2 1 2 1 2 1 2 1 2 1 2 1

The values resulting from the products of the weights are:

5 18 2 18 7 6 4 2 0 12 7 0 6

Following Rule 4, these values then should be:

5 19 2 19 7 6 4 2 0 13 7 0 6

The sum is 90. This is divisible by 10 and, therefore, is a valid number.

The same result is very easily achieved with our definition system by using weights:

1 2.1 1 2.1 1 2.1 1 2.1 and so on.

When the values of the products are calculated, the run-time validation routine rounds any value up to the nearest integer. The following result, therefore, is given in our example:

5 19 2 19 7 6 4 2 0 13 7 0 6

with a sum of 90 as required.

5.1.7.3.1 Delete Validation (F2)

Pressing **F2=del-validation** from the Check Digit Validation menu deletes the check digit validation for the current field.

5.1.7.3.2 Sum Products Value / Digits (F3)

Pressing **F3=sum-products-values/digits** from the Check Digit Validation menu enables you to toggle between having the values or the digits of each product summed.

Default: Sum products values.

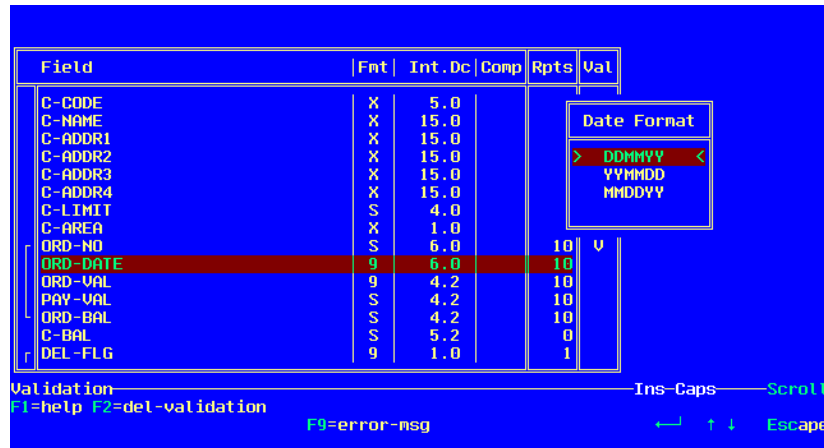
5.1.7.3.4 Error Message (F9)

Pressing **F9=error-msg** from the Check Digit Validation menu has the same effect as error message definition for range/table validation. You can assign an error message to the check digit validation so that if any check digit validation error occurs at run time, the error text is placed into the error field and the error message number is placed into the control block. You can, of course, use the same or different error messages to those used for range/table validation.

5.1.7.4 Date Validation (F5)

Pressing **F5=date** from the Validation menu displays the panel and menu shown in Figure 5-9.

Figure 5-9. Date Validation Menu



The Date Validation facility enables you to define date validation for a field if the field defined is appropriate for dates. If you have not defined a field appropriate for dates, this option is not displayed, because only functions available for the current field type appear on the Validation menu. Date validation is independent of whether you display the field on the panel in date edited form, for example 99/99/99. Full leap year checking is available.

Date validation checks that the components of a single date field are in the correct range, and therefore are not impossible dates. Do not confuse this with range validation for date fields, which checks that the value of a date field lies within a range that you specify (see the section *Range/Table Validation (F3)* earlier in this chapter).

You can define date validation for the following types of field:

- numeric of size 4, 5, 6 or 8 with no decimal places
- alphanumeric of size 7

Depending upon the size and type of the field, you can select any one of the following date types from the Date Format panel:

4-character dates	DDMM MMDD MMYY YYMM
5-character dates	YYDDD
6-character dates	DDMMYY YYMMDD MMDDYY
7-character dates	DDMMMYY where MMM is one of JAN, FEB, MAR, APR, MAY, JUN, JUL, AUG, SEP, OCT, NOV or DEC.
8-character dates	DDMMYYYY

Use the <up-arrow> or <down-arrow> keys to highlight the desired date type and press **Enter** to select it.

5.1.7.4.1 Delete Validation (F2)

Pressing **F2=del-validation** from the Date Validation menu deletes the date validation for the current field.

5.1.7.4.2 Error Message (F9)

Pressing **F9=error-msg** from the Date Validation menu has the same effect as error message definition for range/table validation. You can assign an error message to the date validation so that if any date validation error occurs at run time, the error text is placed into the error field and the error message number is placed into the control block. You can, of course, use the same or different error messages to those used for range/table validation.

5.1.7.5 Null Validation (F6)

Pressing **F6=null-validation** from the Validation menu enables you to determine whether a null entry in the field is valid or invalid. Null means zero in numeric fields or spaces in alphabetic or alphanumeric fields. The menu shown in Figure 5-10 is displayed.

Figure 5-10. Null Validation Menu



5.1.7.5.1 Allow / Disallow Null (F2)

Pressing **F2=allow/disallow-null** from the Null Validation menu enables you to toggle between the allow and disallow null options. If you choose the "disallow null" option, you can assign an error message number to this validation.

Default: Allow null values.

5.1.7.5.3 Error Message (F9)

Pressing **F9=error-msg** from the Null Validation menu enables you to assign an error message if the disallow null option has been selected. As with error message definition for range/table validation, you can assign an error message, the text of which will be placed into the error field and the number of which will be placed into the control block if any null validation error occurs at run time. You can, of course, use the same or different error messages.

5.1.7.6 User Validation (F7)

Pressing **F7=user-validation** from the Validation menu enables you to apply your own validation type to any field, cross-validate with other fields, set flags or change data in the data block, and dynamically change the error message for display.

The user validation menu is displayed.

5.1.7.6.1 Allow/ Disallow User-validation (F2)

Pressing **F2=allow/disallow-user-validation** from the User-validation menu enables you to toggle between the allow and disallow user validation options. If you choose the "allow user-validation" option, you can assign an error message number to this validation.

Default: Disallow user-validation.

At run time, when you are about to exit a field which has user validation enabled, the module DSUSRVAL is called. This module is passed the Control Block and full Data Block, as well as a small Status Block. DSUSRVAL contains validation code written by you. Your Micro Focus COBOL system includes a skeleton source of this module, containing comments to help you add your own code.

You use the Control Block to establish the screenset, panel, field-number and occurrence of the current field. Because of this level of control, the single module DSUSRVAL can be used by many different screensets, panels and fields.

To simplify the setting up of the current field in the user code, Dialog System can generate a copybook for each screenset you need to include in the source of **dsusrval.cbl**. These copybooks are similar to the standard screenset copybooks (**.cpb**), but are tailored for use specifically for inclusion in the user validation source file. You generate these copybooks using **F5=Generate** from the main menu.

Note: The copybook generated for screenset use is not interchangeable with the copybook generated for user validation. Therefore, ensure you use two different extensions for them. For example, **.cpb** and **.cpu**.

You use the copybook code to reference fields in the Data Block by name, and to identify the current field from the Control Block field number by using the associated constant (level-78) names in the copybook.

After validating the field, you can flag it as valid or invalid. If invalid, any error message you associated with this validation is, by default, displayed in your message data item. You can override this by specifying an error message number which will cause the corresponding message to be displayed.

See the chapter *Programming* for more details of the DSUSRVAL module.

5.1.8 Error Messages Field Define / Undefine (F9)

Pressing **F9=err-msg-fld def/undef** from the Data Fields Definition menu enables you to designate a single alphanumeric field on the Data Definition List to be the "error message field". This is the field in which error messages are placed if validation errors are found at run time. This field is automatically cleared if no errors are found, or when those found are corrected.

Press **F9** to designate a field as the error message field, and an "E" is displayed in the **Val** column of the Data Definition List. You are prevented from validating this field itself.

5.2 Error Messages (F3)

Pressing **F3=error-messages** from the Data Definition menu enables you to build up a file of error messages to use when validating the various data fields at run time. These error messages are stored in a relative file and are automatically accessed by their relative key.

You can assign these messages to any of the field validations in the validation definition function (described earlier in this chapter). This file can be common to many applications and does not need to relate to just one screenset, so as noted previously, you can use the same error message file in any number of screensets. Error messages will appear in the designated error message field.

When you select this facility, the Error Messages menu is displayed. If there is no error message file selected for the current screenset, the menu shown in Figure 5-11 is displayed. If there is already an error message file selected for the current screenset, the menu shown in Figure 5-12 is displayed.

Figure 5-11. Error Message Menu



A screenshot of a terminal window with a blue background. The text 'Error-message-definition' is at the top. Below it, 'F1=help' is highlighted in red. To the right, 'Ins-Caps' and 'Scroll' are listed, and 'Escape' is listed below them.

5.2.1 Select Error Message Filename (F2)

Pressing the **F2=select-error-message-filename** option from the Error Messages menu displays the following prompt:

Enter error message file name [] then **Enter**

The error message filename must not exceed eight characters, plus a three-character extension. If no extension is supplied, **.err** is the default.

After you name an error message file, the option **F3=define-error-messages** appears on the Error Messages menu. You can use this option to enter or modify the error message as shown in Figure 5-12.

Figure 5-12. Error Message Menu



A screenshot of a terminal window with a blue background. The text 'Error-message-definition' is at the top. Below it, 'F1=help' is highlighted in red, 'F2=select-error-message-filename' is highlighted in green, and 'F3=define-error-messages' is highlighted in blue. To the right, 'Ins-Caps' and 'Scroll' are listed, and 'Escape' is listed below them.

5.2.2 Define Error Messages (F3)

To define your error message, press **F3=define-error-messages** from the Error Messages menu and the screen shown in Figure 5-13 is displayed.

In the **No.** field, type the error code. This is a number in the range 001 to 998. The cursor automatically skips to the next field where you can type the error message text, which can be up to 76 characters long. To enter this text and move to the next line ready to enter another error

5.2.2.3 Duplicate Line (F5)

Pressing **F5=duplicate-line** from the Error Message Definition menu duplicates the current error message text on the next line, if that line is empty. If the next line is not free, the following prompt appears:

No place to duplicate

and the operation is canceled.

When you duplicate error message text, you must assign a new number to the text before you can move off the line.

5.2.2.4 Restore Line (F6)

Pressing **F6=restore-line** from the Error Message Definition menu enables you to undelete up to 16 deleted entries, starting with the most recently deleted one. The cursor must be on or directly beneath an empty line when you use this function. If the deleted entry has been restored out of sequence, you must alter the error number to put it in sequence.

6 Panels

The second step in developing your Dialog System application is defining panels for your screenset. Panels are rectangular working areas on the screen in which data fields and keystrokes are displayed. The size of a panel can be up to 80 characters wide and 25 lines long. You can include text anywhere on the panel; however, data fields will overwrite panel text and if you choose to have a border, it will overwrite any text on the border.

You use these panels to display the data fields you previously defined to be used in an application. The following chapters describe how to paint panels and place data fields and groups onto them.

This chapter describes:

- How to create new panels
- How to modify existing panels
- How to define action bars to produce your own menus

To enter the Panels menu, press **F3=panels** from the Main menu. The Panels menu is displayed, which provides options for defining a new panel or selecting a previously defined panel (see Figure 6-1).

Figure 6-1. Panels Menu

```

New set                                     Row: 01-Col: 01-Ins-Caps-Num-Scroll
F1=help F2=mark/unmark F3=panel-list F4=CUA-action-bar-and-pull-downs
                                         ↑+↑+ ←=define-panel  Escape
  
```

6.1 Mark/Unmark (F2)

You use this function to mark the area that a panel will occupy when you create a new panel. To mark an area, move the cursor to the position on the screen you wish to place your panel, then press the **F2=mark/unmark** toggle from the Panels menu. The current cursor position is then marked.

Use the <left-arrow>, <right-arrow>, <up-arrow> and <down-arrow> keys to extend the marked area from the original marked position in a rectangular fashion to the desired size. Panels can be any size from one character by one line up to 80 characters by 25 lines. When you have completed marking the area, press **Enter=define-panel**. You are then prompted to name the panel. Type in a name of up to eight characters, then press **Enter**. The Panel Painting menu is displayed, where you can enter text and data (see the chapter *Panel Painting* for details).

You can alter the size and position of the panel later, so you don't have to get the dimensions correct the first time. To cancel the mark function, press **F2** again before completing the panel definition.

6.2 Panel List (F3)

Pressing **F3=panel-list** from the Panels menu displays a popup list containing the names of panels previously defined for the current screenset and the Panel List menu as shown in Figure 6-2.

Figure 6-2. Panel List Menu



6.2.1 Select (Enter)

Pressing **Enter** from the Panel List menu enables you to work on a selected panel. To select a particular panel to work on, use the <up-arrow> and <down-arrow> keys to position the selection bar on the name of the panel, then press **Enter**. You can then edit the panel using the other facilities available on the Panel List menu.

6.2.2 Show Panel (F3)

Pressing **F3=show-panel** from the Panel List menu displays the chosen panel on the screen so that you can view it and consider its relationship to other panels.

This function is only for viewing purposes; to edit a panel, you must select as described in the previous section.

Up to 50 panels can be visible on the screen, but only one of these can be active at one time. Because Dialog System itself uses panels, for

example to display Dialog System menus, you might find that your practical limit to the number of visible panels is less than 50.

6.2.3 Unshow Panel (F4)

Pressing **F4=unshow-panel** from the Panel List menu removes the chosen panel from the visible screen. Remember the panel is still in the screenset, just no longer visible on the screen.

6.2.4 Select First Panel (F5)

This identifies which panel is invoked first at run time. You can select any of the panels to be this first panel. Position the selection bar on the desired panel name and press **F5=select-first-panel** from the Panel List menu. The selected panel name is marked with > and < symbols.

6.2.5 Copy Panel (F6)

Pressing **F6=copy-panel** from the Panel List menu creates a new panel that is a copy of the panel identified by the selection bar, including all of its text, attributes and dialog. When you use this facility, you are prompted for a name for the new panel. Type in the desired name and press **Enter**. The new panel is automatically copied and its name included on the panel list.

The new panel becomes the current panel and is listed on the status line. The new panel occupies the same space on the screen as the original. If required, you can move it elsewhere, resize it, or change it using the facilities provided on the Panel Maintenance menu described in the chapter *Panel Painting*.

6.2.6 Delete Panel (F8)

Pressing **F8=delete-panel** from the Panel List menu enables you to delete a panel from the screenset. Position the selection bar on the name of the desired panel, and press **F8**. The following prompt appears:

```
Delete panel. Are you sure? Y/N
```

Press "Y" to delete the selected panel or "N" to cancel the delete operation. If the selected panel is referenced in any dialog, you are prevented from deleting it.

6.2.7 Relocate Panel (F9)

The Relocate Panel facility enables you to adjust the order of the panel list by relocating the panels individually in the list. Position the selection bar on the desired panel name, and press **F9=relocate-panel** from the Panel List menu. The following prompt appears:

```
Relocating: panel-name
```

Use the <up-arrow> and <down-arrow> keys to position the selection bar where you want to insert the relocated panel, and press **Enter**. Once you have relocated a panel, the Panel Painting menu is displayed (see the chapter *Panel Painting* for details on the use of this menu).

6.3 Action Bars (F4)

Pressing **F4=CUA-action-bar-and-pull-downs** from the Panels menu enables you to create, generate, and maintain an action bar with associated pulldown menus. An action bar is a single line of menu text where each word defines a subsidiary menu name. A pulldown menu is the subsidiary menu that is displayed when an item on the action bar menu is selected.

Defining an action bar is a quick way to produce a menu that you can use to invoke other panels. To use an action bar menu, you simply place the cursor on the menu name and press **Enter**, or type the letter

of the associated mnemonic. It is easy to create an action bar in Dialog System. You only need the following steps:

- Supply the text for the action bar and pulldown menu entries (selection and format of mnemonics, and the definition of the help and exit items, is handled automatically).
- Generate the panels and associated dialog that comprise the action bar into the current screenset.
- Add dialog to the panel associated with the action bar to push the current screen and go to the action bar panel.
- Show the action bar panel as part of the associated panel's initialization procedure.
- Configure the colors to be used in action bars with an entry in the defaults file, if desired (see the chapter *Setting Up the Configuration File* for information on setting up the defaults file).

The panel associated with the action bar must be selected as the "first" panel (see the section *Select First Panel (F5)* earlier in this chapter).

To define an action bar, press **F4**, and the Action Bar Definition menu shown in Figure 6-3 is displayed. Only a limited number of the menu options are initially available; others appear after certain stages of action bar definition.

Figure 6-3. Action Bar Definition Menu



```

CUR definition v1      Ins-Caps      Scroll
F1=help F2=action-bar F4=save      Escape
F10=key/key+mouse

```

6.3.1 Action Bar (F2)

Pressing **F2=action-bar** from the Action Bar Definition menu either enables you to define a new action bar, or amend an existing action bar. When you define a new action bar, a series of menus are displayed that you can use to specify the position, size and name of the new action bar. When you amend an existing action bar, the Amend Action Bar menu, shown in Figure 6-8 is displayed. For information about

amending a action bar, see the section *Accept Text (Enter)* later in this chapter.

When you define an action bar, it is often useful to have the panel associated with the action bar visible on the screen. This makes it easier to position and size the action bar. Typically, the action bar occupies the entire top line of the screen, and the associated panel occupies the remainder of the screen. You must position the action bar more than eight rows from the bottom of the screen, and define it as at least 14 characters wide to allow space for the automatically generated exit and help entries. Don't worry about getting the positioning perfect initially, because you can reposition the bar later on.

The first menu, shown in Figure 6-4, enables you to specify the position of your action bar.

Figure 6-4. Action Bar Definition - Position

```
Define action bar position—————Row:01-Col:01-Ins-Caps——Scroll
F1=help                               Escape
Select action bar panel start position using +↑↓, then press ←↓
```

Use the <down-arrow>, <left-arrow>, <up-arrow> and <right-arrow> keys to position the cursor at the desired location on the screen and press **Enter**. The second menu then appears (Figure 6-5), enabling you to define the width of your action bar.

Figure 6-5. Action Bar Definition - Width

```
Define action bar width—————Row:03-Col:06-Ins-Caps——Scroll
F1=help                               Escape
Use +, -, |←, and →| to mark the width of the action bar, then press ←↓
```

With the cursor position now marked, use the <right-arrow> and <left-arrow> or the <right-arrow>| and |<left-arrow> keys to mark the width of the action bar and press **Enter**, which displays a third menu as shown in Figure 6-6.

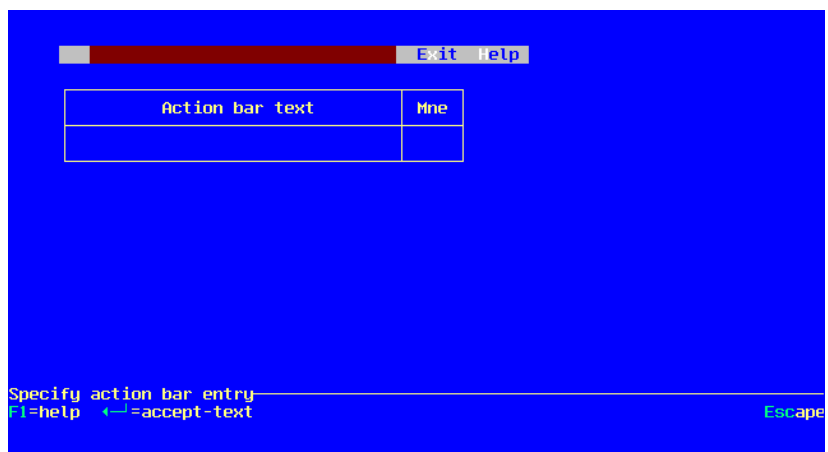
Figure 6-6. Action Bar Definition - Name

```
Specify action bar name _____ Ins-Caps _____ Scroll
F1=help                               Escape
Enter action bar name [      ] then ↵
```

At the prompt, type in a name of up to eight letters for the action bar and press **Enter**. The first four characters of the action bar name are used when the pulldown windows are generated. The generated name is in the format *AAAA-nnn* where *AAAA* are the first four characters of the action bar name, and *nnn* is the pulldown window's generated number.

After you enter a valid name, the action bar is displayed, with the automatically generated exit and help entries. The Action Bar Entry menu and popup panel then appear, as shown in Figure 6-7.

Figure 6-7. Action Bar Entry Menu



The screenshot shows a blue background with a white table and text. At the top, there is a red bar with the text "Exit Help" in white. Below this is a table with two columns: "Action bar text" and "Mne". The table has two rows, with the second row being empty. At the bottom, there is a prompt "Specify action bar entry" followed by "F1=help" and "↵=accept-text" on the left, and "Escape" on the right.

```
Specify action bar entry _____
F1=help ↵=accept-text                               Escape
```

This panel contains the following fields:

Action Bar Text	Where you specify the text of your action bar entry. This text must not contain spaces. You can specify entries until you have filled all available space in the action bar. If there is no free space in the action bar, you can only change or delete existing entries.
Mne	Where the mnemonic assigned to the action bar text appears. This identification letter is automatically assigned but you can change it if desired. You must choose a unique mnemonic. "X" and "H" are reserved mnemonic characters.

6.3.1.1 Accept Text (Enter)

This confirms the action bar text. Enter the text in the Action Bar Entry panel and press the **Enter** key from the Action Bar Entry menu. The menu name appears on the action bar with its mnemonic displayed in reverse video attribute. The action bar text list remains on the screen so you can specify subsequent entries until either there is no more free space, or you insert a blank entry, or you press the **Escape** key.

To change the mnemonic, press the <right-arrow>| key to move the cursor into the Mne field. Overtyping the existing entry and press **Enter**.

When you have completed all of your text entries, press **Enter** and the Amend Action Bar menu appears, as shown in Figure 6-8.

Figure 6-8. Amend Action Bar Menu

```
Amend action bar                               Ins-Caps   Scroll
F1=help F2=insert-before-entry F3=insert-after-entry F4=delete-entry ←=edit
F5=relocate-entry F6=position-bar F7=alter-size F8=action-bar-name   Escape
Use + & + to move selection cursor over the entry you wish to edit
```

When you invoke this menu, the action bar is reformatted to position any free space in the area before the help and exit entries. The selection cursor highlights the current entry, or the free space if there are no action bar entries. The exit and help entries are mandatory and you cannot change or reposition them; therefore, you cannot move the

selection cursor onto these entries. If there are no entries and no free space exists in the action bar, the selection cursor does not appear.

When you leave this menu, the exit and help entries are reformatted to appear immediately after the last user-defined entry. Any free space is placed at the end of the action bar following the help entry.

To use the facilities on the Amend Action Bar menu, position the cursor over the required action bar entry and press **Enter** before you select the desired function.

6.3.1.2 Insert Before Entry (F2)

Pressing **F2=insert-before-entry** from the Amend Action Bar menu prompts you to enter a new action bar entry before the highlighted entry.

6.3.1.3 Insert After Entry (F3)

Pressing **F3=insert-after-entry** from the Amend Action Bar menu prompts you to enter a new action bar entry after the highlighted entry. If this is the last entry, you can insert additional entries until all of the free space is filled.

6.3.1.4 Delete Entry (F4)

Pressing **F4=delete-entry** from the Amend Action Bar menu enables you to delete the current entry from the action bar together with its specified pulldown menu. Press "Y" to delete the entry at the following prompt:

```
Delete entry? Y/N?
```

After you delete the entry, the action bar is reformatted.

6.3.1.5 Relocate Entry (F5)

Pressing **F5=relocate-entry** from the Amend Action Bar menu displays a menu to enable you to relocate the highlighted entry elsewhere in the action bar. Move the selection cursor to the desired location on the

action bar and at the prompt select **F2=insert-before** or **F3=insert-after** to relocate the entry (see Figure 6-9).

Figure 6-9. Relocate Entry Menu

```
Relocate action bar entry _____ Ins-Caps _____ Scroll
F1=help F2=insert-before F3=insert-after _____ Escape
Relocating: File
Move selection cursor to required relocation point, then select before or after
```

6.3.1.5.1 Insert Before (F2)

Pressing **F2=insert-before** from the Relocate Action Bar Entry menu relocates the selected entry before the highlighted entry.

6.3.1.5.2 Insert After (F3)

Pressing **F3=insert-after** from the Relocate Action Bar Entry menu relocates the selected entry after the highlighted entry.

6.3.1.6 Position Bar (F6)

Pressing **F6=position-bar** from the Amend Action Bar menu enables you to reposition the action bar elsewhere on the screen. You are prevented from moving the bar to a point where any of its pulldown windows are not completely visible.

Use the <up-arrow>, <down-arrow>, <right-arrow> and <left-arrow> keys to position the action bar in the desired location, and press **Enter**.

6.3.1.7 Alter Size (F7)

Pressing **F7=alter-size** from the Amend Action Bar menu enables you to expand or contract the width of the action bar. You cannot contract the bar to a point where text is lost. As you alter the size, the bar is reformatted.

Use the <right-arrow> and <left-arrow> keys to resize the bar, and press **Enter**.

6.3.1.8 Action Bar Name (F8)

Pressing **F8=action-bar-name** from the Amend Action Bar menu enables you to rename the action bar. At the prompt, type over the existing action bar name with the new name and press **Enter**.

6.3.1.9 Edit Action Bar Entry (Enter)

Pressing **Enter** from the Amend Action Bar menu returns you to the Action Bar Definition menu, where you can amend the text of the current action bar.

6.3.1.10 Complete Action Bar (Esc)

Pressing **Escape** from the Amend Action Bar menu finalizes the definition of the action bar and reformats it to place any free space at the end. Then the action bar pulldown windows for any user-defined entry are generated. Remember that the exit and help pulldown menus are automatically generated because they require specific formats; however, you can amend the entries in these menus. All pulldown menus automatically have the text "**Esc=cancel**" at the bottom; this is a system requirement that you cannot alter. You return to the Action Bars menu.

6.3.2 Save (F4)

Pressing **F4=save** from the Action Bars menu saves the action bar into the current screenset. Remember you must still save the current screenset.

6.3.3 Generate (F6)

Pressing **F6=generate** from the Action Bars menu generates Dialog System panels and associated dialog into the current screenset. Once they are generated, these are just normal panels and you can edit them using the normal panel editing functions.

The generated dialog handles all movement along the action bar and selection from the pulldown menus when that dialog is invoked at run time. However, to use the action bar, you must add dialog to the application panel associated with the action bar, to push the current screen and go to the action bar panel. The action bar must also be shown as part of the associated panel's initialization procedure (P000).

Add the following dialog to the associated application panel (see the chapter *Dialog* for details on defining dialog):

```
key:    PUSH Pnnn
        GOP  action-bar-name
```

where:

key is the key that switches from your application panel to the action bar.

The parameter associated with the PUSH (*Pnnn*) specifies the procedure to be executed on return from the action bar. The next line, GOP (Go To Panel), goes to the action bar panel. The key used to switch from the application panel to the action bar should be the function key associated with the *resume* entry in the *exit* pulldown.

```
P000:   SHP action-bar-name
```

This SHP (Show Panel) function is contained in the application panel's initialization procedure. This makes the action bar visible when the application panel is visible.

```
Pnnn:   BPR start-procedure
```

This function causes the system to Branch to the Procedure (BPR) that is to be executed on return from the action bar. Depending on the value returned from the internal register, it should branch to the appropriate procedure, that is the start procedure determining the start of the procedure mapping (see the chapter *Dialog* for details of this register). For example, if the start procedure is P010 and a value of two is returned, then P011 is executed. If the register contains zero, then no branch takes place. The value returned in the register depends on the value associated with the selected pulldown entry at definition time. If nothing is selected, zero is returned in the register.

The action bar should be painted on the line *above* the application panel, because when control is passed back to a panel via the POP

function, Dialog System refreshes the panel. If the action bar panel were painted on the top line of the application panel, the action bar panel would be hidden.

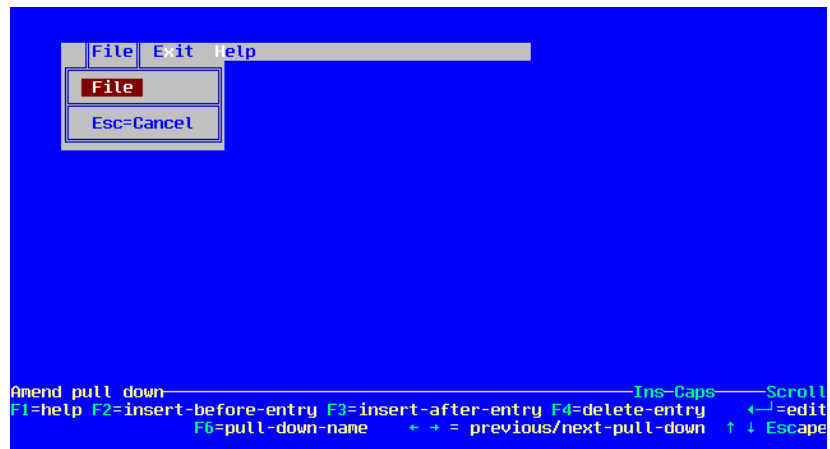
6.3.4 Initialize (F9)

Pressing **F9=initialize** from the Action Bars menu enables you to clear the action bar definitions. To delete this action bar, type in "Y" at the prompt. However, to remove the definition from the current screenset, this action bar must be saved.

6.3.5 Pulldown (Enter)

Pressing **Enter** from the Action Bars menu displays the Amend Pulldown menu (see Figure 6-10), which enables you to specify or change the contents of a pulldown window. A pulldown window must consist of at least one entry (when generated, the entry corresponds to the associated action bar text).

Figure 6-10. Amend Pulldown Menu



6.3.5.1 Insert Before Entry (F2)

Pressing **F2=insert-before-entry** from the Amend Pulldown menu prompts you to enter a new pulldown entry before the current entry. The pulldown list remains on the screen so you can specify subsequent entries until either the pulldown reaches the bottom of the screen, or you insert a blank entry, or you press the **Escape** key.

6.3.5.2 Insert After Entry (F3)

Pressing **F3=insert-after-entry** from the Amend Pulldown menu prompts you to enter a new pulldown entry after the current entry. The pulldown list remains on the screen so you can specify subsequent entries until either the pulldown reaches the bottom of the screen, or you insert a blank entry, or you press the **Escape** key.

6.3.5.3 Delete Entry (F4)

Pressing **F4=delete-entry** from the Amend Pulldown menu deletes the current entry from the pulldown, then reformats the menu. The pulldown must contain at least one entry in addition to the "**Esc=Cancel**" entry.

6.3.5.4 Pulldown Name (F6)

Pressing **F6=pull-down-name** from the Amend Pulldown menu enables you to specify the name of the current pulldown. The name entered is used in the generating phase as the generated panel name. Remember that the generated name is in the format *AAAA-nnn*, where *AAAA* are the first four characters of the action bar name, and *nnn* is the pulldown window's generated number. When you amend this name, it is advisable to select a name that does not exist elsewhere in this screenset and avoid any possible conflict during the generation phase. If any conflict does occur, you are prompted to rename the pulldown.

6.3.5.5 Previous Pulldown Menu (<left-arrow>)

Pressing <left-arrow> from the Amend Pulldown menu moves the cursor to the previous pulldown window or, if the current pulldown is the first on the action bar, to the help pulldown window.

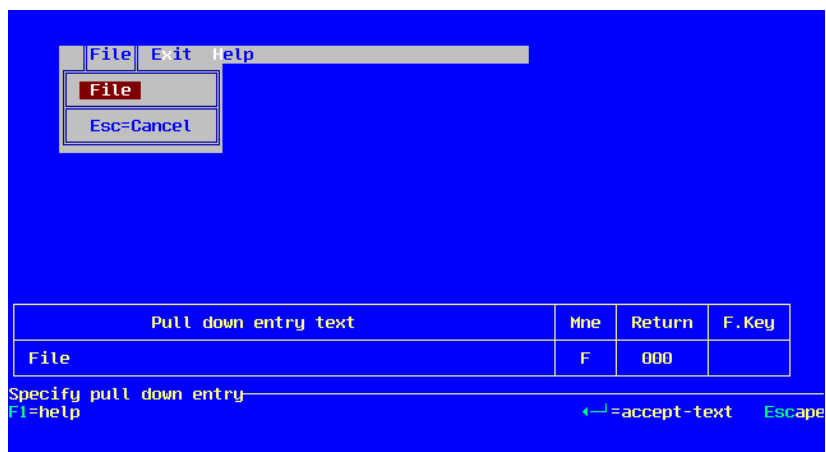
6.3.5.6 Next Pulldown Menu (<right-arrow>)

Pressing <right-arrow> from the Amend Pulldown menu moves the cursor to the next pulldown window or, if the current pulldown is the last one on the action bar, to the first pulldown window.

6.3.5.7 Edit Pulldown Entry (Enter)

Pressing **Enter** from the Amend Pulldown menu displays the Pulldown Entry Text menu and panel, which enable you to specify or amend a pulldown menu entry (Figure 6-11).

Figure 6-11. Pulldown Entry Text Menu



This panel contains the following fields:

Pull down entry text	The text for your pulldown menu. This text can contain spaces.
Mne	The mnemonic associated with this pulldown menu. This is generated automatically but you can override it by typing over the existing character.
Return	This is the value returned from the internal register to the application panel when this pulldown entry is selected. The return value can be in the range 1 to 999.
F.Key	This enables a "quick key" to be associated with the pulldown entry by assigning a function key to it. This function key is active in the action bar as well as the pulldown and is a quick way of selecting the entry, instead of entering the pulldown and selecting the entry. The function key must be unique within the action bar and pulldowns.

6.3.5.8 Accept Text (Enter)

Pressing **Enter** from the Pulldown Entry Text menu confirms the text of the pulldown entry. When you have entered the text, the mnemonic, and the return value as desired, press **Enter**. You are returned to the Amend Pulldown menu.

6.3.6 Key/Key+ Mouse (F10)

Pressing **F10=Key/Key+Mouse** from the Action Bars menu toggles to generate additional dialog that allows the mouse to be used to control the menu bar and the drop down menus when the application is run.

You need to use a CALLOUT program, **dsactbar.cbl**, which is supplied with Dialog System. Amend it to include a special copyfile, created by Dialog System at CUA generation time, called **screenset-basename.cua**. The source file, **dsactbar.cbl**, is clearly documented to show where the copyfile is to be inserted. This file can hold as many copyfiles as you

want, so that this single program can cater for many screensets with mouse controlled CUA menus.

Generating CUA menus for mouse control does not alter the keyboard control that is normally supported; the two interfaces coexist.

Although the mouse support code and dialog is automatically created for supporting the mouse once the top level CUA menu has been selected, you must add a small piece of code to allow the top level menu itself to be activated by the mouse.

7 Panel Painting

Painting panels is part of the defining panels step, which is the second step in developing your Dialog System application. The chapter *Panels* described how to create a panel and define its size. This chapter describes how to use the facilities available from the Panel Painting menu to manipulate the layout and attributes of the panel you have defined. From the Panel Painting Alternate menu, you can define keystroke functions in a panel. The Panel Painting Control menu provides additional facilities for designing the layout of your panels.

This chapter describes:

- How to work with blocks of data
- How to design the appearance of your panel
- How to define local dialog
- How to maintain panels

The Panel Painting menu (see Figure 7-1) is displayed whenever you define a new panel (see the chapter *Panels* for details on defining panels), or select a panel from the panel list.

Figure 7-1. Panel Painting Menu

```
PANEL - 1 Attribute Row:01-Col:01-Ins-Caps-Num-Scroll
F1=help F2=mark/unmark F3=field F4=group F5=paint-attribute F6=attribute-roll
F7=cut-to-block F8=copy-to-block F9=restore-block F10=panel-maint Alt Ctl Escape
```

Use the cursor keys to position the cursor in the panel, and enter any text that you need. If you position things incorrectly, you can cut and paste blocks, as described in the next section.

If you require your screenset to be portable between DOS and UNIX systems, do not use characters outside the standard ASCII character set (codes 032 through 127). For more information, see the section *None/Single/Double Border (F4)* later in this chapter.

7.1 Working with Blocks

In Dialog System, you can manipulate blocks to change your panel design. This means that you do not need to get your panel design perfect the first time. You can cut or copy blocks, which are rectangular areas in a panel, and copy or paste them onto the same panel or onto other panels. The Block menu provides the facilities that enable you to manipulate blocks (see Figure 7-2). You can access this menu using **F7=cut-to-block**, **F8=copy-to-block**, or **F9=restore-block** in the Panel Painting menu. If you cut a block, you cannot split data fields or groups.

Figure 7-2. Block Menu

```

Block      Attribute      Row:02-Col:02-Ins-Caps-Nun-Scroll
F1=help  F2=paste-block  F3=copy-from-block  F4=stack-block
Cursor keys=move block      F10=abandon      Escape
  
```

The block manipulation facilities are described together in this section. The menu options to open the Block menu are described in the relevant menu sections later in this chapter.

Before you open the Block menu, you must specify the desired block using the mark/unmark option described later in this chapter. The Block menu offers the following facilities:

7.1.1 Move Block (<down-arrow> <up-arrow> <right-arrow> <left-arrow>)

Pressing the <down-arrow>, <up-arrow>, <right-arrow>, and <left-arrow> keys from the Block menu moves the block to the desired position so that you can paste, copy or stack that block.

7.1.2 Paste Block (F2)

Pressing **F2=paste-block** from the Block menu relocates the block in a new position and returns to the previous menu.

7.1.3 Copy from Block (F3)

Pressing **F3=copy-from-block** from the Block menu copies the block to a new position. Block highlighting remains on, with control inside the Block menu so that you can select another function. You can copy the marked block multiple times to various places on the panel.

The Copy From Block function differs from the Paste function because after you place the new block in the position you specify, Copy From Block returns you to the Block menu, where you can copy the block again, or select another function. However, Paste returns you to the previous Panel Painting menu.

7.1.4 Stack Block (F4)

Pressing **F4=stack-block** from the Block menu places the block into a stack on a first-in-last-out basis and returns to the previous menu. You can restore the block from the stack later by using the Restore Block function. You can stack a maximum of 16 blocks in this way.

7.1.5 Abandon (F10)

Pressing **F10=abandon** from the Block menu returns a cut block to its original location and cancels the block manipulation function that you selected.

7.2 Panel Painting Main Menu

7.2.1 Mark/Unmark (F2)

Use the mark/unmark facility to specify the area to use in a panel painting function. Position the cursor at one corner of the area you want, and press **F2=mark/unmark** from the Panel Painting menu. Use the <down-arrow>, <up-arrow>, <right-arrow>, and <left-arrow> keys

to extend the rectangular area from the original marked position. To cancel the mark function, press the **F2=mark/unmark** toggle again. You need to mark an area before you use several functions in the Panel Painting menu, for example to mark a block before you use the block manipulation facilities described in the previous section.

7.2.2 Panel Fields (F3)

Pressing **F3=field** from the Panel Painting menu invokes the Panel Field menu, which enables you to place fields on the panel and to change existing fields. See the chapter *Panel Fields* for more information.

7.2.3 Panel Groups (F4)

Pressing **F4=group** from the Panel Painting menu defines a marked area as a group and then invokes the Panel Group menu. See the chapter *Panel Groups* for more information.

7.2.4 Paint Attribute (F5)

Pressing **F5=paint-attribute** from the Panel Painting menu enables you to use an attribute to paint a cursor position, block or field. You must select your attributes first (see the section *Attribute Palette (F3)* later in this chapter). To paint the cursor position, press **F5**. To paint a block, define the area you want to be painted by using the **F2=mark/unmark** toggle, then press **F5**. To paint a field, place the cursor anywhere on the field, then press **F5**.

The Paint Attribute function works in the following way:

- If the cursor is not on an existing field or group and you have not marked a block, the current character is painted with the chosen attribute and the cursor moves right one position.
- If you have marked a block that does not intersect any groups or data fields, the entire block is painted with the chosen attribute.

- If the cursor is on a field, the entire field is painted with the chosen attribute. If that field is part of a data group, all occurrences of that field in the group are painted with the attribute.
- If the cursor is on a virtual attribute group, you cannot use the paint attribute function. You can paint virtual attributes only through the group function (see the chapter *Panel Groups* for details).

7.2.5 Attribute Roll (F6)

Pressing **F6=attribute-roll** from the Panel Painting menu enables you select from the one default and five preset attributes (see the section *Attribute Palette (F3)* later in this chapter for details about setting these). To roll through the six available attributes, press **F6** until the attribute you require is displayed on the status line. This becomes the current active attribute that can be used to paint a character, field, or block.

You can use any combination of attributes on your screenset to achieve your desired effect. To use a color that is not one of the six presets, you can change the values at any time using the Attribute Panel option, described later in this chapter.

7.2.6 Cut to Block (F7)

Pressing **F7=cut-to-block** from the Panel Painting menu cuts an area you have marked, using the **F2=mark/unmark** key, to a block and invokes the Block menu. This menu is described in the section *Working with Blocks* earlier in this chapter.

If you cut a block that contains a group, there are some restrictions. You must either paste the block, copy from the block or abandon the cut. You cannot store the cut block on the stack until you have first copied it back onto the panel. If you then copy from this cut block, all the group properties are carried with the copied block. However, you will be prompted at the foot of the screen to provide different names for the newly copied groups. Once you have copied the block, you cannot abandon the cut.

7.2.7 Copy to Block (F8)

Pressing **F8=copy-to-block** from the Panel Painting menu copies an area that you marked, using the **F2=mark/unmark** key, and invokes the Block menu. The original area remains in place. The Block menu is described in the section *Working with Blocks* earlier in this chapter.

If you copy to a block that contains a group, you will be prompted for a new name for the copied group. There is no restriction on copying a block that contains a group, so you can place this copied group, along with all its properties, into the stack.

7.2.8 Restore Block (F9)

Pressing **F9=restore-block** from the Panel Painting menu restores the last block placed in the stack at the cursor position. Block highlighting remains on, and control is inside the Block menu so that you can select another function. For example, you can copy a restored block multiple times to various places on the panel, paste it into another location, or put it back onto the stack.

You can to use the copy and restore block functions between two different screensets. To do this, copy a block in one screenset and place the block on the stack. Then load the second screenset and restore the block. However, if you use this procedure, group and field definitions are not copied.

7.2.9 Panel Maintenance (F10)

Pressing **F10=panel-maint** from the Panel Painting menu invokes the Panel Maintenance menu (see Figure 7-3), which provides facilities to change the selected panel as described in the following section.

Figure 7-3. Panel Maintenance Menu

```
Panel maint—Attribute—Row:02-Col:02-Ins-Caps-Num-Scroll
F1=help F2=move F3=alter-panel-size F4=none/single/double border Escape
F5=top/bottom anchor
```

7.2.9.1 Move (F2)

Pressing **F2=move** from the Panel Maintenance menu enables you to move the panel around the screen using the <down-arrow>, <up-arrow>, <right-arrow>, and <left-arrow> keys, the <right-arrow>| and |<left-arrow> keys, and the **Home** and **End** keys. Press the **Escape** key or **F2=move** again to confirm the new panel location.

7.2.9.2 Alter Panel Size (F3)

Pressing **F3=alter-panel-size** from the Panel Maintenance menu displays a further menu (see Figure 7-4), enabling you to change the size of a panel.

Figure 7-4. Alter Panel Size Menu



```

Panel maint Attribute Row: 02-Col: 02-Ins-Caps-Nun-Scrol
F1=help F2=expand/contract panel ↑+↑ ←|=select Escape
  
```

From this menu, press the **F2=expand/contract** toggle to select the required mode. Use the <left-arrow> and <right-arrow> keys to expand or contract the panel in the direction of the relevant cursor key. You are prevented from contracting the panel in a way that would remove a field or group. When the panel is the required size, press **Enter** to confirm the change.

Be careful when you contract a panel because any text contained in the part of the panel that is contracted is lost.

7.2.9.3 None/Single/Double Border (F4)

Pressing the **F4=none/single/double-border** toggle from the Panel Maintenance menu enables you to place a border around the current panel. Press **F4** to select the desired mode. The default for a new panel is to have a single line border, provided there is space to draw a border (that is, the panel is at least one line high and one character wide).

If you place a border around the panel, ensure that you do not put any text on the edge of the panel, because the border will overwrite the

text. To put text on the edge of the panel and have a border around the panel except for the text area, you can define the panel to have no border, then draw the border yourself using the draw function.

UNIX: If you are transferring your screensets between DOS and UNIX environments, there are some limitations that UNIX imposes on line drawing. See the chapter *Character Set Support on UNIX* for more information. However, these limitations are on the display only; the data actually stored is not affected, so you can design your screensets to your requirements in either environment.

7.2.9.4 Top/Bottom Anchor (F5)

Pressing **F5=top/bottom anchor** from the Panel Maintenance menu enables you to specify whether the position of a panel on the screen is treated as relative to the top or bottom of the screen. This feature means that a screenset designed on a display with a certain number of lines can be run on a display with a different number of lines.

Pressing **F5** toggles between the following settings:

- | | |
|--------|--|
| Top | The position of the panel on the screen is relative to the top of the screen. For example, if you create a panel that starts three rows from the top of the screen, the panel will always appear three rows from the top of the screen at run time. This is the default. |
| Bottom | The position of the panel on the screen is relative to the bottom of the screen. For example, if you create a panel that starts 15 rows from the bottom of the screen, the panel will always appear 15 rows from the bottom of the screen at run time. |

7.3 Panel Painting Alternate Menu (Alt)

You can access the Alternate menu, shown in Figure 7-5, from the Panel Painting menu. The Panel Painting Alt menu provides additional facilities for designing the function and appearance of your panel.

Figure 7-5. Panel Painting Alternate Menu



A screenshot of a terminal window showing the Panel Painting Alternate Menu. The text is as follows:
 PANEL - 1 Attribute Row: 02 Col: 02 Ins-Caps-Num-Scroll
 F1=help F2=local-dialog F3=attribute-palette F4=panel-name F6=clear
 F7=protect

7.3.1 Local Dialog (F2)

Pressing **F2=local-dialog** from the Panel Painting Alt menu invokes the Dialog Definition menu, where you define the dialog for functions such as procedures, path control functions (for example, moving between fields and panels), cursor functions, keyboard functions, and group data positioning functions. When you invoke the Dialog Definition menu from the Panel Painting Alt menu, you can define local dialog. (You can also invoke the Dialog Definition menu from the Main menu, which enables you to define global dialog.)

You define local dialog when you want it to be specific to a panel. For example, you could define that **F1** displays a Help panel appropriate to the panel you are working with.

You define global dialog when you want it to be active for the entire screenset. For example, you could define that the <right-arrow>| key always skips to the next field in any panel.

Where a key or procedure is defined both in the local and the global dialog, the definition in the local dialog is acted upon first unless you change this precedence using the Global/Local Dialog First function described in the chapter *Using Dialog System*.

The functions available on the Dialog Definition menu are identical for both local and global dialog. For further information, see the section *Dialog Definition Menu* in the chapter *Dialog*.

7.3.2 Attribute Palette (F3)

Pressing **F3=attribute-palette** from the Panel Painting Alt menu invokes the Attribute Palette menu (see Figure 7-6), where you can paint text

or data with attributes. These attributes are color, brightness, and blinking. You can define two sets of attributes from this menu; one set for the static panel colors and the other for the active run-time colors.

Figure 7-6. Attribute Palette Menu - Roll List

You can use this menu to set up the default background attribute and five other attributes that make up the static panel attributes. If you reset the default attribute, any characters already painted within the current default attribute assume the new attribute. The default attribute requires less storage in the screenset than the other attributes, so you should set it to the attribute that is used most in the panel.

You set the static panel attributes in the same way that you set the colorization details. See the section *Colorize (F8)* in the chapter *Using Dialog System*. You can then use these six attributes in the Attribute Roll function during the panel editing process. See the section *Attribute Roll (F6)* earlier in this chapter.

To set the six active run-time attributes, press **F8=run-time-list** to display the menu shown in Figure 7-7.

Figure 7-7. Attribute Palette Menu - Run-Time

You can use this Attribute Palette Run-time menu to define the following three types of run-time attributes:

- | | |
|----------------------------|--|
| Auxiliary Field Attributes | Four attributes that you can define for use at run time. Assign the desired attributes to the keys F2 through F5 . To use these attributes, you must refer to them explicitly in the dialog. |
|----------------------------|--|

Error Field Attribute	The attribute that the run-time system uses for any input field that fails validation. If you have defined validation, the error attribute is always used when there is an error in a field. If you do not want to use the error attribute, you must select the same attribute that is used for the underlying field attribute. Assign the desired attribute to the F6 key.
Field Entry Attribute	The attribute that the run-time system uses for an input field when the cursor is positioned on it. Assign the desired attribute to F7(input)=X , then press the F8=input-attribute off/on toggle to activate it.

The run-time attributes are not used during panel editing, but are used at run time under the control of the dialog and the error procedures. These six attributes are local to a panel and are saved with that panel when you save the screenset.

7.3.3 Panel Name (F4)

Pressing **F4=panel-name** from the Panel Painting Alt menu enables you to change the name of a panel. Enter the new name for your panel at the prompt and press **Enter**. The new name is displayed on the status line. Remember that panels must have unique names.

7.3.4 Clear (F6)

Pressing **F6=clear** from the Panel Painting Alt menu enables you to clear the current panel together with all its data details, attribute details and dialog details. When you select this function, the following prompt is displayed:

```
Clear panel? Are you sure? Y/N
```

Press **"Y"** to clear the panel or **"N"** to cancel the operation.

The panel remains the same size and keeps its current name. Use this function with caution, because although the panel itself remains, the function deletes all of the previously defined details.

To actually delete the panel itself, you must use the **F8=delete-panel** option in the Panel List menu (see the section *Panel List (F3)* in the chapter *Panels*).

7.3.5 Protect Definition (F7)

Pressing **F7=protect definition** from the Panel Painting Alt menu displays the Protect Definition Screen, shown in Figure 7-8, where you can specify data flag variables. You can associate one or more panel fields with a data flag variable. The data flag variable can take the values zero and one. You use this option to switch a field between input and output at run time.

When you set a data flag variable to one (either from the application program or from dialog), all the fields associated with that variable become protected; that is, these fields cannot accept user input. When you set a data flag variable to zero, all the fields associated with that variable become unprotected; that is, these fields can accept user input. If you use any value other than zero or one, there will be undefined results, which will differ between releases.

Figure 7-8. Protection Definition Screen

Protect Flag	Fields											
	0	0	0	0	0	0	0	0	0	0	0	0
	0	0	0	0	0	0	0	0	0	0	0	0
	0	0	0	0	0	0	0	0	0	0	0	0
	0	0	0	0	0	0	0	0	0	0	0	0
	0	0	0	0	0	0	0	0	0	0	0	0
	0	0	0	0	0	0	0	0	0	0	0	0
	0	0	0	0	0	0	0	0	0	0	0	0
	0	0	0	0	0	0	0	0	0	0	0	0
	0	0	0	0	0	0	0	0	0	0	0	0
	0	0	0	0	0	0	0	0	0	0	0	0

PANEL-1—Protect-definition—Ins-Caps—Scroll
 F1=help F2=list F10=abandon Escape

To define a data flag variable and its associated fields:

- 1 Enter the name of the data flag variable in the Protect Flag column, or use the **F2=list** option to select a name from the list. The variable name can be up to 30 characters long.
- 2 Press **Tab** to move the cursor to the Fields column.
- 3 Enter the field number of a field to be associated with this flag variable. This number (starting at 1) is the number of the input field in the Data Block. You can use the **F2=list** option to set this value.

Enter as many field numbers as required, using **Tab** or cursor control keys to move from field to field. To specify more field numbers than can fit on one line, specify the same flag variable name in the Protect Flag column of the next line, and continue entering field numbers for that flag.

The fields must exist in the first 255 fields in the user data block.

- 4 Press **Enter**. The cursor returns to the Protect Flag column and you can enter details for another data flag variable.
- 5 When you have entered all the data flag variable definitions, press **Escape** to save them. When you generate the copyfile for the screenset, definitions of these flag variables are included in the Data Block.

Alternatively, press **F10=abandon** to abandon the definitions.

You can associate the same field number with more than one flag variable. You can use the same flag variable name for different screens in the screenset.

7.4 Panel Painting Control Menu

Pressing **Ctrl** from the Panel Painting menu invokes a Control menu (see Figure 7-9) that provides further facilities for designing the layout of your panel.

Figure 7-9. Panel Painting Control Menu

```
PANEL - 1      Attribute      Row:02-Col:02-Ins-Caps-Num-Scroll
F1=help F2=field-order F3=previous-panel F4=next-panel F6=draw F7=char-palette
F9=delete-field-defn F10=delete-group-defn      Ctrl+End=menu-on/off
```

7.4.1 Field Order (F2)

Pressing **F2=field-order** from the Panel Painting Ctrl menu enables you to define the order that input fields are accepted in at run time. By default, fields are accepted from top left to bottom right.

When you select this function, the first caret symbol of each field is replaced with the number 5. To define that a field is accepted first at run time, position the cursor at that field and overtype the number 5 with a number lower than five. Overtyping with a number greater than five has the opposite effect. If some fields have the same number, they are accepted in the default order from top left to bottom right.

The order that fields in a group are accepted is relative to that group, so you cannot define an acceptance order that goes in and out of a group. The characters you can use in the field order definition are 0-9, A-Z, and a-z, in ascending order.

7.4.2 Previous Panel (F3)

Pressing **F3=previous-panel** from the Panel Painting Ctrl menu enables you to change control from the current active panel to the previous panel in the screenset so that you can move rapidly between the panels in the screenset.

When you select this function, the panel that you are currently working on is removed from the screen and replaced by the one that you were working on previously.

To see more than one panel at once (so that you can evaluate the relative positioning on the screen), you must return to the Panel Definition menu and select **F3=show-panel** for the panels you require.

7.4.3 Next Panel (F4)

Pressing **F4=next-panel** from the Panel Painting Ctrl menu enables you to change control from the current active panel to the next panel in the screenset.

When you select this function, the panel that you are currently working on is removed from the screen and replaced by the next panel in the screenset. To see more than one panel at once (so that you can evaluate the relative positioning on the screen), you must return to the Panel Definition menu and select **F3=show-panel** for the panels you require.

7.4.4 Draw (F6)

Pressing **F6=draw** from the Panel Painting Ctrl menu invokes the Draw menu (see Figure 7-10), which enables you to draw lines under the cursor as it moves. The Draw menu functions are described in the following sections.

Figure 7-10. Draw Menu



The screenshot shows a blue menu bar with white text. The top line reads 'Drawing Attribute Row: 02-Col: 02-Ins-Caps-Nun-Scroll'. The bottom line lists function keys: 'F1=help F2=draw/erase/move F3=|/† F6=attribute-roll Escape'.

7.4.4.1 Draw/Erase/Move (F2)

Pressing the **F2=draw/erase/move** toggle from the Draw menu enables you to draw a selected line character, erase a previously drawn line character, or move the cursor without drawing or erasing a line character. Select draw mode to leave the selected line character behind as you move the cursor. Select erase mode to erase any line characters previously drawn as you move the cursor. Select move mode to move the cursor without drawing or erasing the line character.

UNIX: If you are working on UNIX, there is a defined set of supported UNIX characters (see the chapter *Character Set Support on UNIX*).

7.4.6 Delete Field Definition (F9)

Pressing **F9=delete-field-defn** from the Panel Painting Ctrl menu deletes the selected field from the panel. Position the cursor on the desired field and press **F9**. You cannot delete a field that is part of a panel group unless the group definition is deleted. Also, you cannot delete a field that is referenced in dialog. To remove a field from a group, you must use the Group Occurrence Maintenance menu, described in the section *Data Groups* in the chapter *Panel Groups*.

7.4.7 Delete Group Definition (F10)

Pressing **F10=delete-group-defn** from the Panel Painting Ctrl menu removes the group identification previously specified for a number of fields. Position the cursor in the desired group and press **F10**. You cannot delete a group definition if the group is referenced in dialog. The fields that previously comprised the group are not deleted.

For more information about the effect of this function on different types of groups, see the chapter *Panel Groups*. To remove a field from a group, you can use the Group Occurrence Maintenance menu, where you can insert, delete or amend fields in a group (see the section *Data Groups* in the chapter *Panel Groups*).

7.4.8 Menu On/Off (Ctrl+End)

This toggle function enables you to remove the Panel Painting menu from the physical screen to give you an unobstructed view while painting your panel. On DOS, press the **Ctrl** and **End** keys simultaneously to toggle the menu on and off. On UNIX, enter control mode and then the key sequence for end.

This removes the menu from the screen at any point in Dialog System.

8 Panel Fields

The third step in developing your Dialog System application is to position data fields on panels. This is the step where you specify where your data fields will appear on your panel for subsequent input or output. There is a limit of 512 data fields in a screenset, and 200 panel fields in any panel.

This chapter explains:

- How to position the data fields you previously defined onto your panels
- How to define screen characteristics to be associated with a field

8.1 Defining Panel Fields (F3)

Pressing **F3=field** from the Panel Painting menu accesses the Panel Field menu, shown in Figure 8-1, and a popup panel containing the list of defined fields. Position the cursor where you would like your field to begin, and press **F3**. You can use this menu to modify the field characteristics of a selected field, and place, remove, modify, and relocate fields on the current panel.

Figure 8-1. Panel Field Menu

Field	Fmt	Length	Comp	Usage	Props.
SUPPLIER	X	20.0			
DATE	9	6.0			
PRODUCT	X	10.0			
QUANTITY	9	3.0			
COST	9	4.2			
ERR-FLD	X	35.0			
				Format	
				Panel length	
				0.0	

Field-naming—Stack:00—Row:01—Col:01—Ins—Caps—Scroll
 F1=help F3=panel-functions F5=anend-field F6=use-stacked-field
 F7=sort-list F8=unsort-list ←=select-from-list PgUp PgDn Escape

8.1.1 The Popup Panel

You can use the popup panel to view and change the relevant characteristics of the required fields. The left half of the panel lists the data fields, their format, and their length, which you specified during data definition (see the chapter *Data Definition*). To select a field to work on, position the selection bar on the required field and press **Enter**. The cursor moves to the right side of the panel.

The right half of the panel contains field detail panels that enable you to change the characteristics of that field on the panel. The field detail panels are:

- Usage
- Properties
- Format
- Panel Length
- Delimiter

If the format of the field is a date format, the Delimiter panel is displayed; otherwise, the Panel Length panel is displayed.

In the field detail panels, you can choose from the existing default values, or select a different characteristic. Use the <right-arrow>| and |<left-arrow> keys to move between the characteristics and press the space bar to select one. The characteristics selected are highlighted with > and < symbols.

To return a field characteristic to its default value after you change it, press the **F2=reset** function. This function is available from each of the field detail panels and also some of the Panel Field submenus.

When you have completed your selection, press **Enter** . Your field is placed at the cursor position on the panel as a series of caret characters. The following screen field characteristics are available:

Usage	Specifies how a panel field is used at run time. You can select one of the following mutually exclusive options:
IN	Input field. This prefills the field with the current data field value and enables the entry of data at run time. This is the default type of usage.
OUT	Output field. This prefills the field with the current data field value and protects it from input at run time.
EXIT	Exit field. This is an input field that forces return to the calling program when the field is exited by any key defined in the dialog, provided that the field has not failed any validations.
XMOD	Exit modified field. This is an exit field that forces return to the calling program if the field contents have changed, provided that the field has not failed any validations.

	XREG	Exit regardless field. This is an exit field that forces return to the calling program when the field is exited, regardless of whether the field has failed validation. However, the status of the validation is given in the control block.
	XENT	Exit on entry. This exits to the calling program immediately when the field is entered. So, for example, if you skip from the previous field to this field, return to the calling program is forced before any values are entered.
Properties		Enables you to define properties for the selected field to adopt at run time. These properties are not valid if the field is defined as an output field.
	AUTO	Autoskip. Forces the cursor to move automatically to the next field when the field is filled at run time. This is the default property.
	REQD	Required. Prevents the cursor from leaving the field if it is empty at run time, that is, if the field contains spaces in an alphabetic or alphanumeric field, or contains zero in a numeric field. For a date field, prevents the cursor from leaving the field if the day or the month are empty. Although the cursor cannot leave the field, an error message is not produced at run time unless you have defined a data validation. Validations are defined at the data level (see the chapter <i>Data Definition</i>) and are applied whenever that data item is used, whereas the REQD property is applied only at local screen level.

FULL	<p>Prevents the cursor from leaving the field if an alphabetic or alphanumeric field has any leading or trailing spaces, or a numeric field has any leading zeros at run time. You cannot use this property for date fields.</p> <p>Although the cursor cannot leave the field, an error message is not produced. The FULL property is applied at local screen level; to force a field to be full whenever this field is used, you must define validation rules for it at the data level.</p>
UPPER	<p>Changes any lower-case characters in alphabetic or alphanumeric fields to upper case when they are entered.</p> <p>NLS is supported, for the folding of alphabetic characters. This feature is optional, and is controlled by either setting NLS in the sidefile dsdef.cfg, or by linking dsnlsrtn.obj as well as the COBOL system file cobnlsmsg.obj when creating the executable application.</p>
LOWER	<p>Changes any upper-case characters in alphabetic or alphanumeric fields to lower case when they are entered.</p> <p>NLS is supported, for the folding of alphabetic characters. This feature is optional, and is controlled by either setting NLS in the sidefile dsdef.cfg, or by linking dsnlsrtn.obj as well as the COBOL system file cobnlsmsg.obj when creating the executable application.</p>

NOECHO Prevents keystrokes entered from being displayed on the screen at run time. Any characters entered cannot be captured by any facilities that look at the screen buffer, such as a Print Screen key on the keyboard. This provides a secure way of entering passwords and related functions.

By default, a non-echoed field displays spaces when data is entered. You can configure Dialog System to use some other character; see the chapter *Setting Up the Configuration File*.

Format Specifies how the field is displayed at run time. The formats you can choose from depend on the format of the field defined during data definition (A, X, 9, or S). For example, if you defined a numeric field, you can choose zero suppression in the display, or if you defined a date field, you can choose from several date display formats. Again, the choice depends on the length of the date field in the data definition.

Formats are mutually exclusive and only formats that are valid for the selected field are displayed on the popup panel. Possible formats are:

ALPHABETIC All upper-case and lower-case alphabetic characters, plus the space character, can be entered.

ALPHANUMERIC All upper-case and lower-case alphabetic characters, the space character, and all the other ASCII keys, including the extended characters entered using the Alt key, can be entered.

NUMERIC All numeric characters and the decimal point can be entered.

SIGNED All numeric characters, the decimal point, and the "+" and "-" signs, for positive or negative values, can be entered.

SUPPRESS	Provides zero suppression for input and output, and right justifies data entry and display. The last zero in the number is always displayed.
99/99	Places four-character dates in DD/MM, MM/DD, MM/YY, or YY/MM format.
99/999	Places five-character dates in YY/DDD format, where DDD is the day number from 1-365.
99/99/99	Places six-character dates in DD/MM/YY, YY/MM/DD, or MM/DD/YY format.
99/AAA/99	Places seven-character dates in DD/MON/YY format, where MON is a three-letter mnemonic for the month (see the chapter <i>Data Definition</i> for valid month mnemonics).
99/99/9999	Places eight-character dates in DD/MM/YYYY format. If your applications need to work with the year 2000 and beyond, you should use this format. You might need to revise your existing screensets.

When the cursor enters the Format panel, the additional menu option **F4=specify-user-format** is displayed, which enables you to select a user-defined format rather than the fixed formats described previously. Pressing **F4** displays a popup panel where you can enter a format number in the range 1 through 4, where the format number corresponds to one of up to 4 user-defined formats. If you select a user-defined format, you must write a program, named **dsusrfmt.cbl**, that handles the format, before you can use the resulting screenset. See the chapter *Programming* for more information about the **dsusrfmt.cbl** program.

Panel Length	<p>Enables you to choose a panel length for the field that is less than the true data field length. You cannot choose a length greater than the true data field length. You can use this characteristic to set a field up as a scrolling field.</p> <p>When you enter the Panel Length panel, the additional menu option F4=horizontal-scrolling-on/off is displayed, which enables you define the field as a scrolling field. Specify the panel length for the display field, then press F4 to toggle the scrolling on. A scrolling field uses two extra character positions on the screen for indicators at either end of the field to show whether if scrolling is possible in either direction.</p>
Delimiter	<p>Enables you to specify a date delimiter when the field is a date format. You can use any character, such as "/", "-", or ".", as a delimiter or separator. The default is "/".</p>

8.1.2 The Panel Field Menu

The facilities provided in the Panel Field menu are described in the following sections.

8.1.2.1 *Select from List (Enter)*

Enables you to select the field whose characteristics you wish to set. Use the <up-arrow> and <down-arrow> keys to position the selection bar on the required field and press **Enter** from the Panel Field menu. The cursor moves to the right-hand side of the popup panel, where you can specify usage, properties, format, and panel length or delimiter, as described in the previous section.

8.1.2.2 *Panel Functions (F3)*

Pressing **F3=panel-functions** from the Panel Field menu returns control to the cursor in the panel and invokes the Panel Functions menu, shown in Figure 8-2. This menu enables you to use the facilities for storing field definitions when you copy or move existing fields. This option is also available from the Amend Field and Use Stacked Field menus, described later in this chapter.

Figure 8-2. Panel Functions Menu

```

Field-naming Stack:00 Row:01-Col:01-Ins-Caps
F1=help F4=cut-to-stack F5=copy-to-stack
↑ ↓ + + ← Home End → | ← Escape

```

The Cut To Stack and Copy To Stack options in this menu work in a similar way to the block manipulation facilities described in the chapter *Panel Painting*. If you use these options to place a field in a stack, you can restore that field to the screen using the **F6=use-stacked-field** function described later in this chapter.

When you select **F3**, the cursor moves from the popup panel to the panel screen. This can be useful if you wish to define field characteristics but you have not positioned the cursor where you require it in the panel screen before you accessed the popup panel. You can select **F3** to return to the panel screen and reposition the cursor, then press **Escape**, which exits the Panel Functions menu and returns control to the popup panel.

8.1.2.2.1 Cut to Stack (F4)

Pressing **F4=cut-to-stack** from the Panel Functions menu deletes the field at the cursor position and places it in the field stack. The status line indicates the number of fields in the stack.

8.1.2.2.2 Copy to Stack (F5)

Pressing **F5=copy-to-stack** from the Panel Functions menu copies the field at the cursor position into the field stack, leaving the original field in its place on the panel. The status line then indicates the number of fields in the stack.

8.1.2.3 Amend Field (F5)

Pressing **F5=amend-field** from the Panel Field menu displays the Amend Field menu, shown in Figure 8-3, which enables you to change the characteristics of an existing field. Position the cursor on the field you want to amend and press **F5**. The selection bar is positioned on the appropriate entry in the data list and the characteristics currently set

for the field are displayed. You can change the usage, properties, format, and panel length or delimiter as described in the section *The Popup Panel* earlier in this chapter.

Figure 8-3. Amend Field Menu

```

Field-naming—Stack: 00—          Row: 01—Col: 02—  Ins—Caps—Scroll
F1=help  F2=reset  F3=panel-functions
Use the space bar to mark an option      ←=complete
                                           →|←  Escape
  
```

To delete a field rather than amend it, you must use the **F9=delete-field-defn** option in the Panel Painting Ctrl menu (see the chapter *Panel Painting*).

8.1.2.3.1 Reset (F2)

Pressing **F2=reset** from the Amend Field menu resets the field characteristics in the highlighted field details panel to their default values.

8.1.2.3.2 Panel Functions (F3)

Pressing **F3=panel-functions** from the Amend Field menu invokes the Panel Functions menu described earlier in this chapter.

If you select **F3=panel-functions** from the Amend field menu to reposition the cursor before you finalize the field characteristics, a copy of the field is placed on the panel in the new position, and the original is left unchanged. You cannot position a field on top of an existing field or a group.

8.1.2.4 Use Stacked Field (F6)

Pressing **F6=use-stacked-field** from the Panel Field menu places the details of the field currently on top of the stack onto the popup panel and displays the Use Stacked Field menu, shown in Figure 8-4. You can use this menu in the same way as the Amend Field menu to modify the characteristics of the field selected from the stack before you place it on the panel.

Figure 8-4. Use Stacked Field Menu



```
Field-naming-Stack:00 Row:01-Col:02-Ins-Caps Scroll
F1=help F2=reset F3=panel-functions ←=complete
Use the space bar to mark an option →|←=Escape
```

8.1.2.4.1 Reset (F2)

Pressing **F2=reset** from the Use Stacked Field menu resets the field characteristics in the highlighted field details panel to their default values.

8.1.2.4.2 Panel Functions (F3)

Pressing **F3=panel-functions** from the Use Stacked Field menu invokes the Panel Functions menu described earlier in this chapter.

8.1.2.5 Sort List (F7)

Pressing **F7=sort-list** from the Panel Field menu sorts the field name list into alphabetical order, which might make it easier to locate the field you want. The groups in the data list remain together as a body, but the fields in each group are sorted alphabetically.

8.1.2.6 Unsort List (F8)

Pressing **F8=unsort-list** from the Panel Field menu reverts the field name list back to the original order, that is, the order in which you defined the fields in data definition (see the chapter *Data Definition* for details). The fields will occur in this original order in the generated Data Block.

9 Panel Groups

Defining groups in panels is part of the positioning panel fields step, which is the third step in developing your Dialog System application.

A group defines a rectangular area on the panel where you can move up or down, or vertically scroll data or text. You can use groups to create a selection menu, or to display data lines in a visible area that is smaller than the number of lines available.

This chapter explains:

- How to use groups to design menus that scroll and use selection bars
- How to use groups to contain more text than is visible on the screen
- How to use groups to highlight separate parts of a line

9.1 Group Types

There are three types of groups: *data groups*, *text groups*, and *attribute groups*.

Data groups contain at least one data field defined as a group during data definition (see the chapter *Data Definition*). These groups might contain repeated data fields, all of which must be from the same underlying Data Block group. These group repetitions equate to the COBOL OCCURS clause. You can define a data group to scroll at run time and assign a selection bar to it. You would typically use a data group where you require a table for data, for example on an order entry form.

Text groups contain text rather than data fields. You can also assign selection bars to text groups. You can define fixed text groups or virtual text groups. Virtual text groups can contain more text than is visible on the group area, so the user can scroll through the data at run

time. It can be useful to think of text groups as data groups that are used only for output. Fixed text groups are often used for menus, while virtual text groups are often used for extended help screens or pick and point lists.

Attribute groups contain a selection of attributes that you can use to modify text by moving them over a line on your panel. Attribute groups are only one line high, so you cannot assign a selection bar to them. Attribute groups are often used for highlighting selected areas, for example the scroll and caps indicators on the Dialog System status line.

To invoke the Group menus from the Panel Painting menu, either move the cursor to a previously defined group and select **F4=group**, or mark an area for a new group and select **F4=group**. Groups cannot overlap each other or intersect fields and groups cannot be nested. Depending on the size of the marked area, you can invoke menus to define data groups, text groups, or attributes, as described in the following sections of this chapter.

9.2 Working with Groups

When you define groups, some functions are common to the different types of groups. These common functions are described in this section, but you can access them from several different group menus.

9.2.1 Selection Bar (F5)

Pressing **F5=select-bar** enables you to assign a selection bar to the group. You can use this option from the Data Group menu or the Text Group menu. When you select this option, the cursor is positioned on the top line of the selected group. If you have previously defined a selection bar for a group, pressing **F5** enables you to inspect the current selection bar attributes (see Figure 9-1).

Figure 9-1. Selection Bar Menu

The screenshot shows a blue menu bar with white text. The menu items are: Attribute, ORD-LINE, Group, Row: 12-Col: 08-Ins-Caps, Scroll, F1=help, F4=delete-select-bar, F5=paint-attribute, F6=attribute-roll, ←=complete, and Escape.

The facilities provided by the Selection Bar menu are described in the following sections.

9.2.1.1 Delete Selection Bar (F4)

Pressing **F4=delete-select-bar** from the Selection Bar menu deletes a previously saved selection bar.

9.2.1.2 Paint Attribute (F5)

Pressing **F5=paint-attribute** from the Selection Bar menu paints the current attribute on the cursor position. To paint the entire selection bar with this attribute, press **F5** the number of times required to reach the defined length , then press **Enter**.

9.2.1.3 Attribute Roll (F6)

Pressing **F6=attribute-roll** from the Selection Bar menu enables you to select from the one default and five preset attributes you previously defined (see the chapter *Panels* for details). To select one of the six available attributes, press **F6** until the required attribute is displayed on the status line. This becomes the current active attribute that you use with the Paint Attribute function described previously.

9.2.1.4 Escape (Esc)

Pressing **Escape** from the Selection Bar menu cancels the selection bar definition and returns to the previous menu without saving any of the changes made. For example, if you change the attribute you previously defined for a selection bar then press **Escape**, the new definition is canceled, and the selection bar is restored to its original form.

9.2.2 Group Name (F8)

Pressing **F8=group-name** from the Data Group, Text Group or Attribute Group menu invokes the Group Name menu, shown in Figure 9-2, which enables you to change the group name. Each group must have a name that is unique in the current screen.

Figure 9-2. Group Name Menu



```
Data-Group          ORD-LINE  Group      Row: 12-Col: 07-Ins-Caps  Scroll
F1=help F2=group-namelist
Enter Group Name [ORD-LINE] and press ↵
```

To change the group name, type a new name for the group at the prompt and press **Enter**.

9.2.2.1 Group Namelist (F2)

Pressing **F2=group-namelist** from the Group Name menu displays a list of current group names to which you can refer when you choose a new name for your group.

9.2.2.2 Escape (Esc)

Pressing **Escape** from the Group Name menu cancels the renaming operation and returns you to the relevant Group menu.

9.2.3 Virtual Text/Attribute (F2)

Pressing **F2=define-virtual-text** from the Text Group menu invokes the Virtual Text Group Definition menu shown in Figure 9-3.

Figure 9-3. Virtual Text Group Definition Menu

```
Virtual-Text-Definition          Position-001-of-002  Ins-Caps  Scroll
F1=help F3=insert-line F4=delete-line F7=restore-line F8=duplicate-line
F9=delete-virtual-text F10=abandon          PgUp PgDn Escape
```

Pressing **F2=define-virtual-attribute** from the Attribute Group menu invokes the Virtual Attribute Group Definition menu shown in Figure 9-4.

Figure 9-4. Virtual Attribute Group Definition

```
Attribute-Defn-Attribute        Position-001-of-002  Ins-Caps  Scroll
F1=help F3=insert-line F4=delete-line F5=paint-attribute F6=attribute-roll
F7=restore-line F8=duplicate-line F9=delete-virtual-attrs F10=abandon Escape
```

The facilities in both of these group menus are nearly identical, and are described together in the following sections. The Virtual Attribute Group Definition menu contains the same function as the Virtual Text Group Definition menu, plus the Attribute Roll and Paint Attribute functions described earlier in this chapter.

These two menus enable you to define or change virtual text or virtual attribute groups, as appropriate. You can enter text as required, using the <up-arrow> and <down-arrow> keys to scroll the virtual text. To add a line at the end of the text, press **Enter** on the last entry.

A virtual text group can be up to 255 lines, or up to 4000 characters (that is, the number of lines multiplied by the width of the virtual text panel), whichever limit is reached first. Therefore, if a virtual text panel is 25 characters wide, the group can be up to 160 lines long (4000 divided by 25).

9.2.3.1 Insert Line (F3)

Pressing **F3=insert-line** from the Virtual Text/Attribute Group Definition menu inserts a new line at the current position and increases the total size of the group by one. You can insert lines into a group until you reach the maximum number of lines or characters, as described in the preceding section.

9.2.3.2 Delete Line (F4)

Pressing **F4=delete-line** from the Virtual Text/Attribute Group Definition menu deletes the current line and reduces the total size of the group by one. The deleted line is saved, so you can subsequently restore it if required. You can store up to 16 deleted lines.

9.2.3.3 Restore Line (F7)

Pressing **F7=restore-line** from the Virtual Text/Attribute Group Definition menu restores the last deleted line, inserting it at the current position. You can restore up to 16 deleted entries.

9.2.3.4 Duplicate Line (F8)

Pressing **F8=duplicate-line** from the Virtual Text/Attribute Group Definition menu duplicates the current line onto the next line if it is empty. The total size of the group is incremented by one.

9.2.3.5 Delete Virtual Text/Attribute (F9)

Pressing **F9=delete-virtual-text** from the Virtual Text Group Definition menu deletes the hidden entries and redefines the group as a fixed text group. The visible occurrences at the top of the virtual text are retained in the fixed text group. If you define virtual text and there is no hidden text, the virtual text group automatically reverts to a fixed text group.

Pressing **F9=delete-virtual-attribute** from the Virtual Attribute Group Definition menu deletes the virtual attributes previously defined, retaining on screen the visible occurrences at the top of the group.

9.2.3.6 Abandon (F10)

Pressing **F10=abandon** from the Virtual Text Group Definition menu cancels the virtual text definition and exits to the Fixed Text Group menu.

Pressing **F10=abandon** from the Virtual Attribute Group Definition menu restores the group to its original form and exits to the Virtual Attribute Group menu.

9.2.3.7 Escape (Esc)

Pressing **Escape** from the Virtual Text Group Definition menu saves the virtual text definition and returns to the Virtual Text Group menu. The first occurrences in the virtual text group are shown in the visible screen group area.

Pressing **Escape** from the Virtual Attribute Group Definition menu saves the virtual attribute definition and returns to the Virtual Attribute Group menu.

9.2.4 Group Size Maintenance (F9)

Pressing **F9=group-size-maintenance** from the Text Group or Attribute Group menu invokes the Group Size Maintenance menu, as shown in Figure 9-5. This menu enables you to expand or contract a virtual text or virtual attribute group horizontally in either direction. You can also expand or contract virtual text groups vertically from the bottom.

Figure 9-5. Group Size Maintenance Menu

```
Virtual-attr-group  ATTR  Group  Row:08-Col:34-Ins-Caps  Scroll
F1=help F2=expand/contract  ←=complete  Escape
```

9.2.4.1 Expand/Contract (F2)

Pressing the **F2=expand-contract** toggle from the Group Size Maintenance menu enables you to expand or contract the group. Press **F2** to set the desired mode, then use the <left-arrow>, <right-arrow> or <down-arrow> keys to adjust the width as required. The new width is applied to all group occurrences.

Be careful when you contract a group horizontally, because virtual text groups will be adjusted to fit into the revised space, and you might unintentionally lose some text.

9.3 Data Groups

A data group is a group that contains at least one data field defined as a group during data definition (see the chapter *Data Definition*). It can be useful to think of a data group as a single line of data fields (one occurrence) that you can repeat as required.

You can expand a data group vertically, up to the limits of the panel, or the number of repeats specified for the field during data definition, as long as the group does not overlap any other groups or data fields. However, all the fields in the data group must be on the same line of the panel. Also, all the fields in the group must belong to the same underlying Data Block group. You can assign a selection bar to this group. The data group can contain more lines than can fit in the visible panel area, because the hidden lines can be scrolled into the visible area at run time.

To access the Data Group menu shown in Figure 9-6, use the **F2=mark/unmark** function from the Panel Painting menu to mark an area that contains at least one data field defined as a group and press **F4=group**. The Group Name menu and prompt are displayed. Type a name for the group, which must be unique in the current panel. You can use the **F2=group-namelist** function from the Group Name menu to view a list of the group names already in use on the current panel. Press **Enter**. This invokes the Data Group menu.

Figure 9-6. Data Group Menu

```

Data-Group          ORD-LINE      Group          Row:12-Col:08-Ins-Caps      Scroll
F1=help F2=group-accept-exit bar/end F5=select-bar F8=group-name
F9=group-occurrence-maint ↓=add-occurrence ↑=remove-occurrence ←=complete

```

9.3.1 Group Accept Exit Bar/End (F2)

Pressing the **F2=group-accept-exit bar/end** function enables you to set the behavior of the selection bar in a data group at run time. You can specify where the selection bar moves when it reaches the last field on a line with multiple fields. When a data group contains multiple fields in each group line, you must define the skip to next field (SKNF) function in the dialog (see the chapter *Dialog* for details of defining dialog) so that the selection bar can move between these fields.

You use the **F2** toggle to specify the selection bar behavior at the end of the group line. If you select the group-accept-exit bar mode, the data in the group is accepted and the selection bar skips to the next field outside the group. If you select the group-accept-exit end mode, the data in the group is accepted and the selection bar skips to the next field inside the group.

Default: Group-accept-exit bar.

9.3.2 Selection Bar (F5)

Pressing **F5=select-bar** from the Data Group menu invokes the Selection Bar menu described earlier in this chapter.

9.3.3 Group Name (F8)

Pressing **F8=group-name** from the Data Group menu invokes the Group Name menu described earlier in this chapter.

9.3.4 Group Occurrence Maintenance (F9)

Pressing **F9=group-occurrence-maintenance** from the Data Group menu enables you to amend fields contained in the group or to expand or contract the group's size. When you select the **F9** function, the group is visibly reduced to its one occurrence on the panel, and the menu shown in Figure 9-7 is displayed.

Figure 9-7. Group Occurrence Maintenance Menu

```

Data-Group-----ORD-VAL-----Group-----Row:12-Col:34-Ins-Caps-----Scroll
F1=help F3=field F9=group-size-maintenance-----Escape
  
```

9.3.4.1 Field (F3)

Pressing **F3=field** from the Group Occurrence Maintenance menu invokes the Panel Field menu and popup list, which enables you to add, change or delete fields in the group (see the chapter *Panel Fields* for more information about this menu). You can alter only the fields in the current group, so these are the only fields displayed in the popup list.

If you use the Panel Field menu option to cut all the fields from a group, you will exit from the group maintenance function. A warning message is displayed and you can either paste in a field to remain in the group maintenance function, or continue to exit. If you exit, the group is no longer treated as a data group, but as an empty text field.

9.3.4.2 Group Size Maintenance (F9)

Pressing **F9=group-size-maintenance** from the Group Occurrence Maintenance menu invokes the Group Size Maintenance menu described earlier in this chapter.

9.3.5 Add Occurrence (<down-arrow>)

Pressing the <down-arrow> key from the Data Group menu expands the group vertically downward by one occurrence at a time, up to the limits of the panel, or the number of repeats specified for the field during data definition.

9.3.6 Remove Occurrence (<up-arrow>)

Pressing the <up-arrow> key from the Data Group menu contracts the group vertically upward by one occurrence at a time. Remember that a data group must contain at least one data field.

9.4 Text Groups

A text group is a group that contains no data fields. You can expand this group vertically up to the limit of the panel size, as long as that it does not overlap any other groups or data fields. You can assign a selection bar to this group.

You can make a text group into a virtual text group, that is, a group that contains more text than can fit in the visible area. At run time, these hidden lines of text can be scrolled in and out of the visible area. A text group that is not defined as a virtual group is referred to as a fixed text group.

To access the Text Group menu shown in Figure 9-9, use the **F2=mark/unmark** function from the Panel Painting menu to mark an area that contains no fields and press **F4=group**. If the area marked covers more than one line, it is assumed to be a text group, and the Group Name menu is displayed immediately. If the marked area is only one line high, the Group Selection menu shown in Figure 9-8 is displayed, where you must select **F3=text-group** before the Group Name menu is displayed.

Figure 9-8. Group Selection Menu

```

Group-selection          Group          Row:01-Col:08-Ins-Caps  Scroll
F1=help F2=attribute-group F3=text-group          Escape

```

At the prompt on the Group Name menu, type a name for this group, which must be unique within the current panel. You can use the **F2=group-namelist** function on the Group Name menu to view a list of the group names already in use on the current panel. Press **Enter** to invoke the Text Group menu.

Figure 9-9. Text Group Menu

```

Fixed-Text-Group        GROUP-1          Group          Row:01-Col:08-Ins-Caps  Scroll
F1=help F2=define-virtual-text F5=select-bar F8=group-name  Escape
F9=group-size-maintenance

```

9.4.1 Define Virtual Text (F2)

Pressing **F2=define-virtual-text** from the Text Group menu invokes the Virtual Text Group Definition menu described earlier in this chapter. This menu enables you to redefine a fixed text group as a virtual text group. The status line in the Virtual Text Definition menu shows two values: one is the number of the current line in the virtual text and the other is the current total number of lines of virtual text. The default value for the total is one greater than the number of visible lines in the marked area.

9.4.2 Selection Bar (F5)

Pressing **F5=select-bar** from the Text Group menu invokes the Selection Bar menu described earlier in this chapter.

9.4.3 Group Name (F8)

Pressing **F8=group-name** from the Text Group menu invokes the Group Name menu described earlier in this chapter.

9.4.4 Group Size Maintenance (F9)

Pressing **F9=group-size-maintenance** from the Text Group menu invokes the Group Size Maintenance menu described earlier in this chapter.

9.4.5 Escape (Esc)

Pressing **Escape** from the Text Group menu exits the Text Group menu and returns to the Panel Painting menu.

9.5 Attribute Groups

An attribute group contains a single line of attributes that can be used to modify text by moving them over a line on your panel. These groups, which can be only one line high, are always virtual groups. You cannot assign a selection bar to an attribute group.

To access the Attribute Group menu shown in Figure 9-10, use the **F2=mark/unmark** function from the Panel Painting menu to mark an area that is one line high and contains no fields and press **F4=group**. The Group Selection menu is displayed. Select **F2=attribute-group**, and the Group Name menu and prompt is displayed. Type a name for this group, which must be unique in the current panel. You can use the **F2=group-namelist** function on the Group Name menu to view a list of the group names already in use on the current panel. Press **Enter** to invoke the Attribute Group menu.

Figure 9-10. Attribute Group Menu

Virtual-attr-group GROUP-2 Group Row:01-Col:03-Ins-Caps Scroll
 F1=help F2=define-virtual-attributes F8=group-name F9=group-size-maintenance
 Escape

9.5.1 Define Virtual Attributes (F2)

Pressing **F2=define-virtual-attributes** from the Attribute Group menu invokes the Virtual Attribute Group Definition menu described earlier in this chapter. This menu enables you to specify a set of attributes that can be moved over a line on your panel to modify the line attribute.

9.5.2 Group Name (F8)

Pressing **F8=group-name** from the Attribute Group menu invokes the Group Name menu described earlier in this chapter.

9.5.3 Group Size Maintenance (F9)

Pressing **F9=group-size-maintenance** from the Attribute Group menu invokes the Group Size Maintenance menu described earlier in this chapter. Remember that attribute groups can only be expanded left or right; they cannot be expanded down.

9.5.4 Escape (Esc)

Pressing **Escape** from the Attribute Group menu exits the Attribute Group menu and returns to the Panel Painting menu.

10 Dialog

The fourth step in developing your Dialog System application is to define dialog. Dialog is the set of functions you define to control the behavior of your screenset, such as procedures, path control functions (for example, moving between fields and panels), cursor functions, keyboard functions, and group data positioning functions.

This chapter explains:

- What functions and parameters are available
- How to create procedures
- How to define global and local dialog
- How to define event keys

Dialog can be local, which is specific to a panel, or global, which is active for the whole screenset. For example, to specify that the F1 key displays a Help panel appropriate to that panel, you would use local dialog, whereas to specify that the <right-arrow>| key always skips to the next field, you would use global dialog.

Where a key or procedure is defined both in the local and the global dialog, the definition in the local dialog is acted upon first unless you change this precedence using the Global/Local Dialog First function described in the chapter *Using Dialog System*.

10.1 Functions, Parameters and Procedures

The following sections describe all the available functions, the parameters you can use, and how you can combine functions in procedures. A full description of each function and its parameters is in the chapter *Function Code List*.

To use any of these available functions and their parameters, you can either type in the required function key codes, or use the facilities available in the Dialog Definition menu described later in this chapter.

10.1.1 Functions

The following sections describe all the available functions.

10.1.1.1 Attribute Functions

You use attribute functions to set attributes on fields. You might do this to highlight certain fields on a panel, or all the fields on a panel. For example, you would use the error attribute to highlight an error found during a validation performed in the calling program (for example, file record not found). You can use any of the six run-time attributes described in the chapter *Panel Painting*. The following attribute functions are available:

SFAT	Set Field Attribute
SGAT	Set Global Attribute

10.1.1.2 Cursor Functions

You use the cursor functions to turn the cursor state on and off. This might be useful, for example, if you do not require the cursor for a panel that contains no input fields but is already on the screen because of some screen activity outside the control of the Dialog System. You could set the cursor state to off to remove the cursor from the screen.

If you turn the cursor on and off, it has no relevance to panels that do contain input fields, because the cursor is automatically switched on to mark the field entry position. The following cursor functions are available (the default is cursor off):

CON	Cursor On
COFF	Cursor Off
SETCUR	Set Cursor

10.1.1.3 Path Control Functions

You use the path control functions to control the flow through the screenset. You can set movement to selected input fields and transfer control to selected panels. Also, you can redirect a keystroke to another panel and then repeat that keystroke. This might be useful, for example, if there is a pulldown window visible and you want to redirect the <down-arrow> key on the current panel to the pulldown window so that a selection bar moves down a list of menu items.

Similarly, repeating the current keystroke is very useful to POP out of a panel after an ANYO keystroke (see the chapter *Key Code List*), then to use that keystroke again in the panel that you POP to. The following path control functions are available:

GOF	Go To Field
GOP	Go To Panel
RKEY	Redirect Key
RPKY	Repeat Last Keystroke
SKNF	Skip To Next Field
SKPF	Skip To Previous Field

10.1.1.4 Procedure Functions

You use procedure functions to move to and execute procedures. You can use procedures to extend a dialog line beyond the limit of eight entries; to simplify the dialog; and to be used by the calling program. You simplify the dialog by placing common sets of functions into one place to permit use from several dialog entries. Procedure names take the format *Pnnn*, where *nnn* is the number of the procedure (000-255). The following procedure functions are available:

BP	Branch To Procedure
BPE	Branch To Procedure On Exception
BPR	Branch To Procedure Depending On Register
XP	Execute Procedure
XPE	Execute Procedure on Exception
XPR	Execute Procedure Depending On Register

10.1.1.5 Stack Functions

You use stack functions to PUSH and POP panels, so that control is passed to a particular panel from a number of other panels and then is easily returned to the original panel.

The following stack functions are available:

PUSH Push To Stack
POP Pop From Stack

10.1.1.6 Calling Program Function

RETC forces control to return to the calling program. When control is returned to Dialog System from the calling program, the rest of the dialog following the RETC is obeyed. If the calling program requires a different action from this, the program must request a procedure (see the chapter *Running the Screenset*). The calling program function is:

RETC Return To Calling Program

10.1.1.7 Callout Function

You use the callout function to call a user program directly from dialog. You might use this function, for example, when disk access or complex validation is required.

When a call is made to an external program, Dialog System passes the DS-Control-Block and the *screenset-Data-Block* to the called program. The callout function is:

CALLOUT Call External Program

10.1.1.8 Refresh Functions

You use the refresh functions to force Dialog System to refresh the panel. You can refresh both the panel text and data or simply the text. You can also delay the panel refresh until after a number of screen activities are complete.

Dialog System attempts to optimize how much screen refreshing it performs by calculating the smallest rectangle it must display to show all changes made to the current panel since its last display. Therefore, usually only part of the panel is redrawn during these functions. The following refresh functions are available:

RFTD Refresh Text And Data
 RFT Refresh Text
 SUSP Suspend Panel Refresh

10.1.1.9 Keyboard Scan Function

You use the keyboard scan function to determine whether any of the status keys, such as **Ins** or **Caps** have been depressed while control was outside Dialog System. You might use this function to update a status line on the screen. The keyboard scan function is:

KS Keyboard Scan

10.1.1.10 Mouse Function

You use the mouse function to allow useful mouse behavior without the need to return to the application or execute a CALLOUT. The mouse function is:

GOMOUSE Control level of functionality of mouse

10.1.1.11 Group Selection Bar Functions

You use the group selection bar functions to position the selection bar on a group in a panel. These functions can be applied to a data group, a fixed text group, or a virtual text group. Remember that you cannot use selection bars in attribute groups, therefore you cannot use these functions with attribute groups.

You can use these functions to move a selection bar up and down the underlying group. The following group selection functions are available:

PBDN	Position Bar Down
PBUP	Position Bar Up
SB	Set Bar

10.1.1.12 Screen Group Data Positioning Functions

You use the screen group data positioning functions to scroll underlying data or text through a visible screen group, without moving the selection bar. These functions apply to data groups, virtual text groups or attribute groups. They do not apply to fixed text groups because their size is the same as the screen group's size.

PDDN	Position Data Down
PDUP	Position Data Up
SBOD	Set Bar On Data Item
SD	Set Data

10.1.1.13 Screen Group Array Size and Register Functions

You use the screen group array size and register functions to modify the contents of the array size in relation to a screen group. The array size limits the amount of movement permitted in the data group. Like the internal register, the array size is an internal variable that you can manipulate in various ways. If you do not set the array size yourself, it defaults to the size of the underlying data group. The functions available are:

MAS	Move Array Size
MB	Move Bar
MBD	Move Bar's Data Position
MTD	Move Top Data Position
SAS	Set Array Size

10.1.1.14 Screen Group Insertion and Deletion Functions

You use the screen group insertion and deletion functions to insert and delete lines in a data group without returning to the calling program. These functions automatically adjust the array size. The functions available are:

DBP Delete At Bar Position

IBP Insert At Bar Position

10.1.1.15 Screen Group Procedure Functions

You use the screen group procedure functions to branch to or execute a procedure, depending on the positioning in a group. You can use these functions for all group types: data, fixed text, virtual text, and attribute groups. You might use these functions when a text or attribute group is being used as a menu.

BPD Branch To Procedure Depending On Data

XPD Execute Procedure Depending On Data

10.1.1.16 Conditional Functions

You use the conditional functions to test the values in fields or in the register so there can be either conditional branching in the dialog, or a conditional procedure performed with subsequent return to the calling dialog.

You do this by specifying two items to test and a procedure to branch to or perform if that test is true. The functions available are:

IF If condition is true, branch to procedure

XIF If condition is true, execute procedure then return

The following conditions can be tested for both functions:

IF=	IF equal to
IF<	IF less than
IF>	IF greater than
IF<=	IF less than or equal to
IF>=	IF greater than or equal to
IFNOT=	IF NOT equal to

10.1.1.17 Data Manipulation Functions

You use the data manipulation functions to initialize or update the value of fields or of the register. The functions available are:

DECVAl	Decrement The Value By 1
INCVAl	Increment The Value By 1
MOVE	Move Data
MOVEVtXt	Move Virtual Text Data
MPID	Move Panel ID

10.1.1.18 Flag Functions

You use the flag functions to communicate with the calling program. A flag is a single-digit numeric data item in the Data Block that can have the value 0 (false) or 1 (true). You normally use a flag in the dialog to signify that an event has occurred that you want to communicate to the calling program. You can set, unset, or change the value of the flag.

The functions available are:

CLRF	Clear Flag
SETF	Set Flag
TOGF	Toggle Flag

10.1.1.19 Panel View Function

You use the panel view function to show another panel without actually passing control to it. You might use this function when there is a data entry screen and you wish to show a different menu at certain points during data entry. Another example is when the user enters a new screenset and you wish to display several panels but to pass control to just one of them. The function is:

SHP Show Panel

10.1.1.20 Move Panel Function

You use the move panel function to move a Micro Focus Panels screenset panel around the screen at run time. (See the chapter *Using Dialog System* for details of Micro Focus Panels.) The function is:

MOVEPNL Move Panel

10.1.1.21 Clear Field Functions

You use the clear field functions to clear all input fields, clear an individual input field, or clear from the current cursor position to the end of the field . Clearing a field consists of moving zeros to a numeric field, or spaces to alphabetic or alphanumeric fields. The functions available are:

CLEAR Clear All The Fields
 CFLD Clear Current Field
 CEOF Clear To The End Of The Field

10.1.1.22 Validate Function

You use the validate function to force all the validations on the current screen to be applied, rather than waiting for the automatic validation of an input field that is performed when the cursor is moved off that field.

Only use this function if an exception condition is being monitored, for example with a BPE or XPE function (see the section *Procedure Functions* earlier in this chapter).

If a validation failure is encountered, the validate function makes the field in error the current field and raises an exception condition. If there is no exception condition monitoring, the behavior of this function is unpredictable. The function is:

VAL Perform Validations

10.1.1.23 Terminate Function

You use the terminate function to terminate the current screenset when a program completes. You must do this so that the next program (or the original one) can start another screenset. The termination ensures that all

the necessary internal storage used by the run-time system is properly reallocated. The function is:

TERM Terminate Screenset

10.1.1.24 Sound Function

You use the sound function to specify when the bell should be sounded.

BEEP Sound The Beeper

10.1.1.25 Timeout Function

You use the timeout function to specify a procedure to follow after a specified period of time has elapsed during any future input.

TIMEOUT Timeout

10.1.2 Parameters

The following sections describe the types of parameter that can appear in a function.

10.1.2.1 Panel Name Parameter

The panel name parameter is the name of the panel to be used in a particular function. For example, when you use the GOP (GO to Panel) function, you must specify the panel to pass control to.

10.1.2.2 Field Name Parameter

The field name parameter is either the name of the field, or a numeric value, to be used in a particular function. For example, when you use the GOF (GO to Field) function, the field name identifies the field to go to. When you use the SKNF (SKip to Next Field) function, the numeric value sets the number of fields to skip.

The field name can have a subscript . When the parameter is a field name, the subscript identifies a particular occurrence of the field. For example, GOF CUSTOMER(4) means go to the fourth occurrence of field CUSTOMER. When the parameter is a numeric value, the subscript identifies the occurrence of a field from which to retrieve a value. For example, SKNF DISTANCE(6) means skip over the number fields given by the value in the sixth occurrence of field DISTANCE. A subscript can be a numeric literal, the internal register, or another field that is not part of a group.

10.1.2.3 Group Name Parameter

The group name parameter is the name of the panel, field, or group to be used in a particular function. For example, when you use the PBDN (Position Bar Down) function, you can specify the group through which the selection bar scrolls.

10.1.2.4 Procedure Name Parameter

The procedure name parameter is the name of the procedure to be used in a particular function. For example, when you use the XP (eXecute Procedure) function, you must specify which procedure is to follow.

10.1.2.5 Numeric Value Parameter

The numeric value parameter is the numeric value to be used in a particular function.

10.1.2.6 Alphanumeric Value Parameter

The alphanumeric value parameter is the alphanumeric character to be used in a particular function. For example, when you use the MOVE (MOVE value) function, you can use textual or numeric data.

If you enter a quote character (") in the first position of a parameter, an alphanumeric literal is assumed. This enables the entry of a text literal up to 80 characters in length by scrolling the field horizontally.

10.1.2.7 Attribute Parameter

The attribute parameter is the attribute to be used in a particular function. For example, when you use the SFAT (Set Field Attribute), you must specify the attribute to be used.

10.1.2.8 Register Parameter

The register parameter is Dialog System's internal two-byte computational field register, which some functions use to provide a parameter value. The following register is provided:

\$REG You can use this register to store values temporarily rather than using a field in the Data Block .

10.1.2.9 Null Parameter

The null parameter is used when a particular function provides the option of not specifying a parameter value. For example, the PUSH (PUSH panel) function requires no parameters. The null parameter is signified by the mnemonic "\$NULL".

10.1.3 Using Procedures

A procedure is a sequence of dialog functions that you can use to:

- Place commonly occurring sequences of entries together.
- Contain sequences that can be initiated by the calling program.
- Provide a screen or screenset initialization sequence.

Like dialog, you can define procedures at global or local level by entering the Dialog Definition menu from the appropriate place (see the chapter *Panel Painting* for details on accessing local dialog). Procedures are identified by their number and type. Procedures must be numbered in the range 000 through 255; however, there is a maximum of 100 global dialog entries and 100 local dialog entries (all of which could be procedures).

The procedure number P000 in global dialog is an initialization procedure for the screenset, which is executed whenever this screenset is first entered. You must ensure that the clear dialog flag is set to 0 on the initialization call (see the chapter *Running the Screenset* for details of this flag). Similarly, procedure number P000 in local dialog is an initialization procedure for the specific panel for which it is defined. It is executed whenever control is passed to that panel. Typically, you can use it to show a menu panel or a help panel.

Procedure numbering is an important consideration when you wish to execute different procedures depending on a register value. The procedure named in the function parameter is considered as relating to a register value of one. For example, if the internal register contains a value of two, the procedure number is incremented by one; if the register contains a value of five, the procedure number is incremented by four. The relevance of this correlation between procedure numbers and the register is related to functions like Branch to Procedure

Depending On Register (BPR) and Execute Procedure Depending on Register (XPR), described later in this chapter.

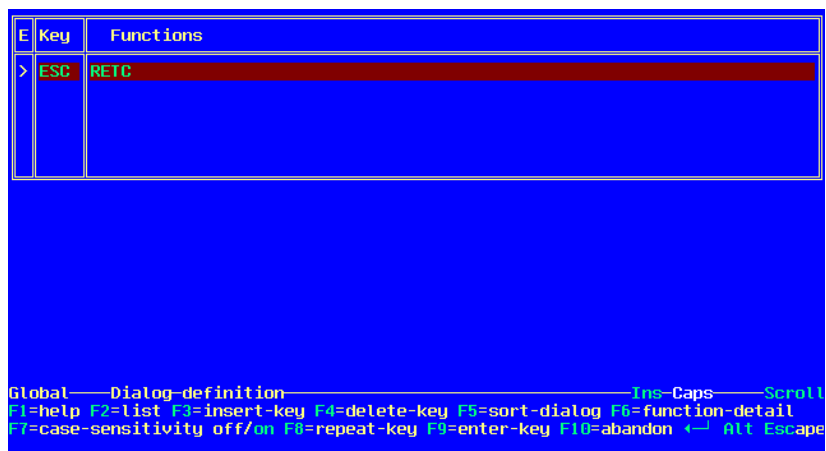
A procedure is just like any other dialog line except that it cannot be defined as an event, because it is not actually a key. Of course, procedures can use procedures, including the initialization procedures.

You specify the procedure number in the Key field in the popup list. The sequence you must follow is defined in the Function field, which is described later in this chapter.

10.2 Dialog Definition Menu

The Dialog Definition menu is accessed from the Main menu for global dialog and from the Panel Painting Alt menu for local dialog. In either case, its functions are identical. To define global dialog, press **F4=global dialog** from the Main menu. The popup panel and menu shown in Figure 10-1 are displayed.

Figure 10-1. Dialog Definition Menu



When you first access this menu, one dialog definition already exists. It specifies that the **Escape** key returns control to the calling program. This

exists so that if you run a screenset, there is always a way out of it. You can, of course, remove this entry. When you access this function after you have already defined some dialog entries, you can scroll up and down the list and insert, change, or delete them, using the appropriate function keys.

You can enter the following details on the popup panel:

Key	The mnemonic identifying the key that you want to define dialog for. You can enter up to 100 key mnemonics per dialog table (see the chapter <i>Key Code List</i> for a full list of key codes). Also, you can enter a procedure name, or the special mnemonic for error procedures, instead of a key value.
Functions	The mnemonic specifying the function, with up to three associated parameters, that the specified key performs. You can assign up to eight to any key (see the chapter <i>Function Code List</i> for a full list of these function codes). These functions determine the behavior of a particular key when it is pressed at run time. You cannot edit the display of functions on this popup panel; it is shown so that you have a high-level view of the dialog. There is a separate menu facility for defining these functions.

It can be useful to visualize your dialog in the form of a table. The following table is an example for the dialog to scroll an array of data (see the chapter *Common Questions and Answers* for details of how these dialog functions perform).

Key	Funct	Param 1	Param 2	Param 3
CURU	PBUP	DATA-GRP	0001	
CURD	PBDN	DATA-GRP	0001	
TAB	SKNF	0001		
BTAB	SKPF	0001		

You can either type your dialog entries directly onto this list or use the facilities available from the Dialog Definition menu.

10.2.1 List (F2)

Pressing **F2=list** from the Dialog Definition menu displays a popup list of available key values. Position the selection cursor on an empty line and press **F2**. Use the <up-arrow> and <down-arrow> keys to scroll up and down the list to the desired value and press **Enter** to select the mnemonic and place it on the list.

A sample of the key mnemonics available in the list facility is shown below:

- ALT1 = Alt key on
- ALT0 = Alt key off
- CTL1 = Ctrl key on
- CTL0 = Ctrl key off
- F1 = Function key 1
- ESC = Escape
- a = character "a"
- Z = character "Z"
- ATA = Alt+character "a"
- AT1 = Alt+character "1"
- ANYO = any keyboard character other than those already defined.

You can also define procedures in the key field. For example, the mnemonic "P003" refers to procedure number 003. If you define this procedure in global dialog, it can be executed from any of the other dialog lines in the current panel.

Similarly, you can enter the mnemonic "ERR", which behaves like a procedure. You use this to define the set of functions that are executed when a validation error occurs.

You can enter the value "*" to define a line of comment text. Enter the comment text in the Parameters column.

To define a procedure in the key field, you must type the procedure number or "ERR", because they are not contained in the list facility.

10.2.2 Insert Line (F3)

Pressing **F3=insert-line** from the Dialog Definition menu enables you to insert a new dialog line above the current line in the list. To add a new line at the end of the list, position the selection bar on the last entry in the list and press **Enter**.

10.2.3 Delete Line (F4)

Pressing **F4=delete-line** from the Dialog Definition menu deletes the current dialog line from the list. If the dialog line defines a procedure that is referenced in the dialog, you are prevented from deleting the line.

10.2.4 Sort Dialog (F5)

Pressing **F5=sort-dialog** from the Dialog Definition menu sorts the dialog lines into the same sequence in which they are automatically sorted when the dialog is exited and saved.

The sort order is:

- 1 Function keys such as F1, Escape, Ctrl+F5, Tab
- 2 Data keys such as "A", "*"
- 3 Shift keys such as Ctrl on and off
- 4 Lock keys such as Caps Lock
- 5 Any other key
- 6 Error key
- 7 Procedures

10.2.5 Function Detail (F6)

Pressing **F6=function-detail** from the Dialog Definition menu invokes the Function Detail menu (see Figure 10-2), which enables you to add or change function entries in a dialog line. A popup panel is also displayed, listing each function with up to three associated parameters. You can define up to eight functions for the current key.

Figure 10-2. Function Detail Menu

Function	Parameter 1	Parameter 2	Parameter 3
RETC	>	>	>

Global—Dialog-definition—Ins-Caps—Scroll
 F1=help F2=list F3=insert-function F4=delete-function F7=clear-parameter
 F0=repeat-function F10=abandon ← Escape

You can use the scroll delimiters in the dialog fields to look at the data names, which can be up to 30 characters. If a dialog field contains text literals, the scroll delimiters are visible over the text, unless that field is the current field. If a dialog field containing text literals is the current field, the scroll delimiters are suppressed and you can edit the field in the normal way.

Use the <right-arrow>| and |<left-arrow> keys to move between the parameter fields in this list. The parameters differ depending on the function with which they are associated or on the type of the parameter entered. You can use fields that are part of a data group as parameters. In this case you must provide a subscript, which can be a numeric literal, a non-group field, or the internal register.

To use this menu option, you need to know about the functions that can be defined and the parameters associated with each function. The functions are described briefly earlier in this chapter. See the chapter *Function Code List* for a full list of function codes and parameter types.

10.2.6 Case Sensitivity Off/On (F7)

Pressing **F7=case-sensitivity off/on** from the Dialog Definition menu enables you to define whether or not the run-time software allows upper- and lower-case keys. If you set this toggle on, you can define both lower- and upper-case versions of a single character key (for example, "a" and "A") and these can be used at run time. If you set this toggle to off, you can set only upper-case characters in the dialog, and all character entries at run time are treated as upper case. Case sensitivity off is useful if a panel does not need to differentiate the case of letters, because you only need to define one dialog entry for each key, rather than two entries; one for each case.

If you attempt to switch case sensitivity off when entries exist in dialog for both the upper-case and lower-case of any key, an error is displayed. If an entry exists for the lower-case of a key, the entry is converted to upper case.

Default: Case-sensitivity on.

10.2.7 Repeat Key (F8)

Pressing **F8=repeat-key** from the Dialog Definition menu duplicates the current dialog line, except for the key value. The cursor remains in the key field so you can enter the key value, which must be unique in the current dialog entries. You cannot leave the line until you enter a key mnemonic. When deliberately defining recursive dialog, you can press **Enter** to override the warning message produced.

10.2.8 Enter Key (F9)

Pressing **F9=enter-key** from the Dialog Definition menu enables you to select a key value by pressing the desired key on the keyboard. Position the selection bar on an empty line, press **F9**, then press the key you require. The key mnemonic is displayed in the popup list. You can use this function for all the keys except the status keys, such as **Ctrl** or **Scroll Lock**.

To select a status key, you must use one of the two alternative methods described earlier to select keys. You can type in the mnemonic for the key you require, or you can select the **F2=list** function and choose the desired mnemonic from the popup list.

The run-time behavior of certain keys, described in the following list, is preconfigured. You can override these configurations by defining other functions for these keystrokes in the list of dialog entries. However, if you do redefine a key, the preconfigured function of that key is no longer executed on the panel where you have redefined the key.

Your keyboard might not have exactly the same control keys as the following list. Consult your Release Notes to see which keys on the standard keyboard for your system provide the equivalent functions.

<right-arrow>	Moves the cursor to the right in the current input field. If the field is an autoskip field, makes the cursor skip from the rightmost character of a field to the start of the next field.
<left-arrow>	Moves the cursor to the left in the current input field. If the field is an autoskip field, makes the cursor skip from the first character of a field to the previous field.
Home Key	Moves the cursor to the first character in the current input field.
End Key	Moves the cursor to the last character in the current input field.
Del Key	Deletes the character at the cursor position in the current input field.
Backspace	Deletes the character to the left of the cursor position in the current input field.

10.2.9 Abandon (F10)

Pressing **F10=abandon** from the Dialog Definition menu cancels the dialog definition without saving any of the current changes. You return to the Main menu if you were defining global dialog, or the Panel Painting menu if you were defining local dialog.

10.2.10 Alternate (Alt)

Pressing **Alt** from the Dialog Definition menu invokes the Dialog Definition Alt menu described later in this chapter.

10.2.11 Escape (Esc)

Pressing **Escape** from the Dialog Definition menu saves the functions and parameters defined during the current session and exits from the menu. You return to the Main menu if you were defining global dialog, or the Panel Painting menu if you were defining local dialog.

10.3 Dialog Definition Alternate Menu

The Dialog Definition Alt menu provides the following facility for dialog definition.

10.3.1 Event Key Define/Undefined (F2)

Pressing **F2=event-key** from the Dialog Definition Alt menu enables you to define any key as an event key. An event key takes priority over any validation of the current field or any field exits that you might define, so the dialog you define for an event key is obeyed at run time, regardless of any validations or field exits.

This might be useful if you want to display a help panel to explain validation rules if a field fails validation. You can specify that the key to

invoke the help panel is an event key, so the user can view help, even though normally they cannot leave a field if it has failed a validation rule.

To define an event key, position the selection bar on the desired key and press **F2**. A greater-than (>) character is displayed by the key mnemonic in the popup list, which identifies the key as an event key. You can define any number of event keys.

After you define the dialog for a screenset, you are ready to test your screenset and make any required modifications. See the chapter *Running the Screenset*.

11 Running the Screenset

The fifth step in developing your Dialog System application is to run your screenset. Dialog System can simulate most of the behavior of the screenset that occurs when you actually produce and run the application program. This means that you can run a screenset before you write or use a program. You can evaluate and test the behavior of the screenset, then modify it if you need to. This is useful to prototype your user interface.

When you have run your screenset, the sixth step in developing your Dialog System application is to modify the screenset by returning to the appropriate step in defining the screenset, described in the earlier chapters in this book. Repeat these two steps until you have the screenset you want.

This chapter explains:

- How to run a screenset without a program
- How to preview the execution of a line of dialog
- How you can use a mouse when you run a screenset

11.1 How to Run the Screenset

To run the screenset, you can use any of the following methods:

- Select **F6=run** from the Main menu.
- Invoke the definition software with the `/t` option.
- Invoke the definition software with the `/r` option.

If you use either of the first two methods, the *trap screen* is displayed. If you use the third method, you can run the screenset without displaying the trap screen first.

11.1.1 The Trap Screen

The trap screen, shown in Figure 11-1, displays the set of information that would normally pass between a calling program and the run-time system.

Figure 11-1. Trap Screen

Control Block Values		panel name.....< >	
control.....[N]	{C,D,N,P}	error code.....<00>	
screen set name..[CUSTOMER.S]		validation code...<000>	
clear dialog....[0]	{0, 1}	exit field.....<0>	
procedure.....[000]	{P000-P255}	data block versn.<00>	
control param....[000]		field number.<0000>.occurrence..<0000>	
		field change.<0>.....count..<0000>	
Data item	Data value		
Scroll-Position-000			
F1=help F2=trace-on/off F3=trap-on/off F4=initialize-data-fields PgUp PgDn Esc			

The information displayed at the top of the trap screen reflects the information contained in the control block. The left-hand side displays input fields, and the right-hand side displays output fields. The lower part of the trap screen displays the data fields contained in the Data Block described in the chapter *Generating the Copyfile*. You can alter some of this information, but some is for display purposes only. You can alter displayed data values so that you can test the screenset with values that you choose. The following sections describe the trap screen fields and their relation to the control and data blocks used in the call interface. See the section *Trap Screen Menu* later in this chapter for information about using the trap screen.

11.1.1.1 Trap Input Fields

The trap input fields contain the parameter values sent to the Dialog System control block. They are displayed on the trap screen so that you can view and change the control block fields passing across the program interface. Where appropriate, the following descriptions include the control block name and level number for these values in parentheses.

Control	A single-character field in which you can enter values indicating the procedure to execute when you press Enter (05 DS-CONTROL). Possible values are:
N	Start to run a new screenset (78 DS-NEW-SET). This runs the screenset specified in the Screenset Name field, starting with the panel defined as the first panel in the screenset. Control remains inside the screenset and returns to the trap again only when a dialog function such as RETC or an EXIT field is encountered.
C	Continue to run a screenset (78 DS-CONTINUE). This runs the screenset from the point where control returned to the trap until the next exit from the run-time system. When control returns to the trap from a running screenset, the run-time system automatically sets the control value to "C".
P	Run a new screenset, but display the data items of the screenset first (predisplay). At this point, the screenset has not been started, and you can amend any of the data values if you wish. When you press Enter, the control parameter is automatically set back to "C" (78 DS-CONTINUE) and the screenset runs using the values set into the data items. If the procedure P000 exists, it is executed.
D	Enter the definition process. This terminates the run and returns to the screenset definition process. To run the screenset with the trap again, select F6=Run from the main menu.

	L	Load run-time software (78 DS-LOAD-SYSTEM). This preloads the run-time software. You do not have to use this call, but you can use it as part of a normal program initialization routine if required.
Screenset Name		The name of the screenset you wish to run (05 DS-SET-NAME).
Clear Dialog		A field you can use to stop any outstanding dialog from being executed (05 DS-CLEAR-DIALOG). You can enter the following values:
	1	Clear outstanding current dialog entries before continuing. This runs the screenset from the point where it stopped; however, any outstanding entries in the current dialog line are cleared. The screenset then awaits further input.
	0	Continue running the screenset normally. This is the default.
Procedure		The procedure that is invoked before the screenset continues to run with the outstanding dialog (05 DS-PROC-NO). This normally has the value "000", which is the initialization procedure, but you cannot perform this procedure from the trap. You can change this to the number of the procedure you wish to use (see the chapter <i>Dialog</i> for information about defining procedures). The procedure you specify is executed when the screenset continues to run, and, as long as the clear dialog field is not set to "1", the rest of the current dialog is executed. Remember that local dialog is searched before global dialog unless you have changed the Global/Local Dialog First flag (see the chapter <i>Using Dialog System</i> for details about this flag). Be careful to use the clear dialog flag correctly when you run procedures from the trap or program. If you do not set the clear dialog to "1", the procedure is executed and any outstanding procedures are added to the procedure stack. There is a limit of 16 stacked procedures; if this limit is exceeded, an error is returned.

Control Param A two-byte field whose bit settings determine whether to apply version number checking, whether to clear the current screen before starting the new screenset, or whether to activate **Ctrl+Break** (DOS)/**INTERRUPT** (UNIX) detection. This parameter is used with the dialog functions to CALL a new screenset or PUSH a screenset, so it is only applicable when the control variable is set to "N".

The bit settings are as follows:

- bit 0 Do not clear the screen before a new screenset starts (78 DS-CONTROL-PARAM-DEFAULT). This is the default.
- bit 1 Do not check the Data Block version number for compatibility with the version of the Data Block in the screenset (78 DS-IGNORE-DB-VER-NO).
- bit 2 Cause **Ctrl+Break** (Windows)/**INTERRUPT** (UNIX) to return an error code to the user program (78 DS-CHECK-CTRL-BREAK).

You can specify the following values with these bit settings:

- 0 Clear the screen before the new screenset starts and check the version number.
- 1 Do not clear the screen before the new screenset starts, but do check the version number.
- 2 Clear the screen before the new screenset starts, but do not check the version number. You must be careful if you use this setting.
- 3 Do not clear the screen before the new screenset starts, and do not check the version number. You must be careful if you use this setting.
- 4 Clear the screen before the new screenset starts, check the version number, and detect **Ctrl+Break/INTR**.

11.1.1.2 Trap Output Fields

The trap output fields contain the values returned from the Dialog System control block. These values provide the application program with information about what happened while control was in your screenset. They are displayed on the trap screen so that you can view the control block fields passing across the program interface. These values are output only; the run-time system ignores any changes you make to them.

Panel Name	The name of the panel that is currently active in the running screenset (05 DS-PANEL-NAME).
Error Code	The code for any run-time error message that Dialog System returns (05 DS-SYSTEM-ERROR-NO). If the code displayed is zero, no error was detected. See the chapter <i>Error Messages</i> for descriptions of the Dialog System run-time error messages.
Validation Code	The code that you defined for field validation errors (05 DS-VALIDATION-ERROR-NO). If the code displayed is zero, either no error was detected or you have not defined any error code (see the chapter <i>Data Definition</i> for details on defining validation errors).
Exit Field	A field that identifies whether return to the trap is forced by a Return to Calling (RETC) program function or caused by an exit field (see the chapter <i>Panel Fields</i> for details on defining exit fields), according to the following values (05 DS-EXIT-FIELD): <ul style="list-style-type: none"> 0 Return forced by a RETC function. 1 Return caused by an exit field.
Data Block Versn	The version number of the current Data Block (05 DS-DATA-BLOCK-VERSION-NO). This number is contained inside the screenset. You can view the version number by pressing F7=Set Details from the Main Alt menu. The version number is significant when you run the screenset with a program, because the generated Data Block must match the Data Block definition in the screenset.
Field Number	The number of the input field on which the cursor is currently positioned (05 DS-FIELD-NO). Fields are numbered sequentially according to their order in data definition.

Occurrence	The occurrence number of the field on which the cursor is currently positioned (05 DS-FIELD-OCCURRENCE). This value applies only if the field is in a group; for a non-repeating field, the value is zero.
Field Change	This field identifies whether or not the current input field was changed during the last input (05 DS-FIELD-CHANGE). The field can take the following values: 0 Not changed during last input. 1 Changed during last input.
Field Count	The total number of fields defined for the specified screenset, that is, contained in the Data Block (05 DS-FIELD-COUNT).

11.1.1.3 Trap Data Fields

The trap data fields are the fields displayed in the list in the lower portion of the trap screen. You can amend these fields. The fields display the following information contained in the Data Block:

Data Item	Lists the names of all the data fields in the current panel.
Data Values	Lists the current values of the listed data fields. You can alter the values displayed here.

11.1.1.4 Control Blocks

If you are a COBOL programmer, it might be useful to see where these trap screen input and output fields appear in the control block used in the call interface. Two separate control blocks are provided with the Dialog System software: **ds-cntrl.mf** (Micro Focus constants) and **ds-cntrl.ans** (ANSI constants). You can use only one of these control blocks in your COBOL program.

11.1.1.4.1 Micro Focus Constants

ds-cntrl.mf (Micro Focus constants) contains the following information:

```

01 DS-CONTROL-BLOCK.
03 DS-VERSION-NUMBERS.
    05 DS-DATA-BLOCK-VERSION-NO PIC 9(4).
    05 DS-VERSION-NO PIC 9.
03 DS-INPUT-FIELDS.
    05 DS-CONTROL PIC X.
        78 DS-CONTINUE VALUE "C".
        78 DS-NEW-SET VALUE "N".
        78 DS-LOAD-SYSTEM VALUE "L".
        78 DS-QUIT-SET VALUE "Q".
        78 DS-PUSH-SET VALUE "S".
        78 DS-PATHNAME VALUE "P".
        78 DS-ERR-FILE-OPEN VALUE "E".
        78 DS-MULTI-POP VALUE "M".
        . 78 DS-TRACE-OFF VALUE "O"
        . 78 DS-TRACE-ON VALUE "T"
        . 78 DS-USE-SET VALUE "U"
        . 78 DS-WHAT-SET VALUE "W"
        .
    05 DS-CONTROL-PARAM PIC 9(4) COMP.
        78 DS-CONTROL-PARAM-DEFAULT VALUE 0.
        78 DS-SCREEN-NOCLEAR VALUE 1.
        78 DS-IGNORE-DB-VER-NO VALUE 2.
        78 DS-CHECK-CTRL-BREAK VALUE 4.
    05 DS-SET-NAME PIC X(16).
    05 DS-CLEAR-DIALOG PIC 9.
    05 DS-PROC-TYPE PIC X.
    05 DS-PROC-NO PIC 9(4) COMP.
    05 DS-INPUT-RESERVED PIC X(12).
03 DS-OUTPUT-FIELDS.
    05 DS-SYSTEM-ERROR-NO PIC 9(4) COMP.
        88 DS-NO-ERROR VALUE 0.
        88 DS-NOT-INITIALIZED VALUE 1.
        88 DS-CANNOT-OPEN-SET VALUE 2.
        88 DS-ERROR-READING-FILE VALUE 3.
        88 DS-INVALID-SET VALUE 4.
        88 DS-CANNOT-CREATE-PANEL VALUE 5.
        88 DS-DYNAMIC-ERROR VALUE 6.
        88 DS-INVALID-FUNCTION VALUE 7.
        88 DS-INVALID-PROC VALUE 8.
        88 DS-VALIDATION-PROG-ERROR VALUE 9.
        88 DS-DATA-BLOCK-VERNO-ERROR VALUE 10.
        88 DS-PUSH-LIMIT-REACHED VALUE 11.
        88 DS-ERROR-FILE-MISSING VALUE 12.
        88 DS-SUBSCRIPT-ERROR VALUE 13.

```

```

      88 DS-PROC-LIMIT-REACHED      VALUE 14.
      88 DS-CTRL-BREAK-PRESSED     VALUE 15.
      88 DS-ERROR-ON-TRACE-FILE    VALUE 16.
05 DS-VALIDATION-ERROR-NO         PIC 9(4) COMP.
05 DS-PANEL-NAME                  PIC X(8).
05 DS-RESERVED-1                  PIC X(2).
05 DS-FIELD-COUNT                 PIC 9(4) COMP.
05 DS-FIELD-OCCURRENCE           PIC 9(4) COMP.
05DS-MOUSE-DETAILS.              PIC X(6).
    07 DS-OUTPUT-MOUSE-X          PIC 9(2) COMP.
    07 DS-OUTPUT-MOUSE-Y          PIC 9(2) COMP.
    07 DS-OUTPUT-MOUSE-FLD        PIC 9(2) COMP.
    07 DS-OUTPUT-MOUSE-OCCURRENCE PIC 9(4) COMP.
05 DS-FIELD-NO                    PIC 9(4) COMP.
05 DS-FIELD-CHANGE                PIC 9.
    88 DS-FIELD-CHANGE-TRUE        VALUE 1.
05 DS-EXIT-FIELD                  PIC 9.
    88 DS-EXIT-FIELD-TRUE          VALUE 1.
05 DS-OUTPUT-RESERVED             PIC X(16).

```

11.1.1.4.2 ANSI Constants

ds-ctrl.ans (ANSI constants) contains the following information:

```

01 DS-CONTROL-BLOCK.
    03 DS-VERSION-NUMBERS.
        05 DS-DATA-BLOCK-VERSION-NO PIC 9(4).
        05 DS-VERSION-NO           PIC 9.
    03 DS-INPUT-FIELDS.
        05 DS-CONTROL                PIC X.
        05 DS-CONTROL-PARAM          PIC 9(4) COMP.
        05 DS-SET-NAME               PIC X(16).
        05 DS-CLEAR-DIALOG           PIC 9.
        05 DS-PROC-TYPE              PIC X.
        05 DS-PROC-NO                PIC 9(4) COMP.
        05 DS-INPUT-RESERVED         PIC X(12).
    03 DS-OUTPUT-FIELDS.
        05 DS-SYSTEM-ERROR-NO        PIC 9(4) COMP.
            88 DS-NO-ERROR            VALUE 0.
            88 DS-NOT-INITIALIZED     VALUE 1.
            88 DS-CANNOT-OPEN-SET     VALUE 2.
            88 DS-ERROR-READING-FILE  VALUE 3.
            88 DS-INVALID-SET        VALUE 4.
            88 DS-CANNOT-CREATE-PANEL VALUE 5.
            88 DS-DYNAMIC-ERROR       VALUE 6.
            88 DS-INVALID-FUNCTION    VALUE 7.
            88 DS-INVALID-PROC        VALUE 8.
            88 DS-VALIDATION-PROG-ERROR VALUE 9.

```

```

      88 DS-DATA-BLOCK-VERNO-ERROR VALUE 10.
      88 DS-PUSH-LIMIT-REACHED     VALUE 11.
      88 DS-ERROR-FILE-MISSING     VALUE 12.
      88 DS-SUBSCRIPT-ERROR        VALUE 13.
      88 DS-PROC-LIMIT-REACHED     VALUE 14.
      88 DS-CTRL-BREAK-PRESSED     VALUE 15.
      88 DS-ERROR-ON-TRACE-FILE    VALUE 16.
05 DS-VALIDATION-ERROR-NO        PIC 9(4) COMP.
05 DS-PANEL-NAME                 PIC X(8).
05 DS-RESERVED-1                 PIC X(2).
05 DS-FIELD-COUNT                PIC 9(4) COMP.
05 DS-FIELD-OCCURRENCE          PIC 9(4) COMP.
05 DS-RESERVED-2                 PIC X(6).
05 DS-FIELD-NO                   PIC 9(4) COMP.

      05 DS-FIELD-CHANGE           PIC 9.
          88 DS-FIELD-CHANGE-TRUE VALUE 1.
05 DS-EXIT-FIELD                 PIC 9.
          88 DS-EXIT-FIELD-TRUE  VALUE 1.
05 DS-OUTPUT-RESERVED           PIC X(16).
01 DS-CONSTANTS.
    03 DS-CONTROL-CONSTANTS.
      05 DS-CONTINUE              PIC X VALUE "C".
      05 DS-NEW-SET               PIC X VALUE "N".
      05 DS-LOAD-SYSTEM           PIC X VALUE "L".
      05 DS-QUIT-SET              PIC X VALUE "Q".
      05 DS-PUSH-SET              PIC X VALUE "S".
      05 DS-PATHNAME              PIC X VALUE "P".
      05 DS-ERR-FILE-OPEN         PIC X VALUE "E".
    03 DS-CONTROL-PARAM-CONSTANTS.
      05 DS-CONTROL-PARAM-DEFAULT PIC 9(4) COMP VALUE 0.
      05 DS-SCREEN-NOCLEAR        PIC 9(4) COMP VALUE 1.
      05 DS-IGNORE-DB-VER-NO     PIC 9(4) COMP VALUE 2.
      05 DS-CHECK-CTRL-BREAK     PIC 9(4) COMP VALUE 4.

```

These control blocks contain a number of reserved fields that you must not use. Also, there are some control constants in the control block that are not described in this chapter. These provide advanced programming facilities, which you should not use until you are thoroughly familiar with Dialog System (see the chapter *Programming* for details of these advanced features).

11.2 Trap Screen Menu

The previous section described the information that is passed between the control block and the COBOL program when a program is run. The following sections describe the menu facilities provided with the trap screen. To use these facilities, you set the options you want, then press **Enter** to start to run your screenset.

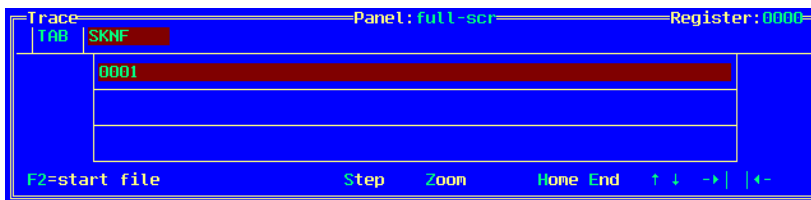
To run a screenset from a previous release of Dialog System through the Trap, you must ensure that the screenset is resaved with the current version number, or there will be an error. You must either run the screenset in a development environment, where the screenset is loaded and automatically resaved with the correct version number before it enters the Trap, or you must run the screenset through the development system to load and save it before you invoke it using a call from an application to DS.

If you try to run a screenset from a previous release of Dialog System by using a call from an application to DS (rather than Dsrun), the screenset will load, but it is not resaved, and an error occurs because the version number is wrong.

11.2.1 Trace On/Off (F2)

Pressing **F2=trace on/off** from the Trap Screen menu toggles the trace facility on and off. The default is trace off.

You can use the trace facility to view the execution of any local or global dialog defined for the panel when the screenset runs. When a keystroke that you defined in your dialog is pressed, the trace screen shown in Figure 11-2 is displayed. For more information about dialog, see the chapter *Dialog*.

Figure 11-2. Trace Panel

The display fields in the trace panel contain the following information:

Trace	Lists the current dialog line, that is, the specified function and associated parameters. If the current function is a procedure, this field lists the contents of the procedure; if the procedure contains more functions than the screen can display, the list scrolls horizontally as each function begins to execute.
Panel	The name of the panel currently running.
Register	The value contained in the internal register (see the chapter <i>Dialog</i> for details on the internal register).

To use this trace facility from a COBOL program without running the trap, you can use an additional function call to Dsrun. To make this call, enter a value of "T" in the Ds-Control field of the control block. To turn the trace off, enter a value of "O" in the Ds-Control field. When you no longer require use of the trace field, you must remember to place a value of "C" back in Ds-Control.

11.2.1.1 Start/Close File (F2)

Pressing **F2=start-file** from the Trace menu starts to write trace information to a text output file named *screenset-name.trc*. The **F2** key becomes **F2=close-file**. To stop writing trace information to the text output file, press **F2** again. You can use this function to control what is written to the file by starting and stopping the file writes when you require. You can then examine the file in an editor and look at the path taken through the dialog.

Dialog can be written to the file either in step or zoom mode, described in the next sections. The output file is always extended, so you must erase it from the current directory as necessary.

11.2.1.2 Step (S)

Pressing **S=step** from the Trace menu executes the dialog displayed in the trace screen one function at a time, performing the current function and then halting. When a function has been completed, press **S** again to execute the next dialog function.

11.2.1.3 Zoom (Z)

Pressing **Z=zoom** from the Trace menu executes the entire dialog line, displaying the user screen instead of the individual steps. When you use this facility, the trace screen is disabled until the next return to the trap, where you can turn the trace on again.

11.2.1.4 Move Panel (Home, End, <up-arrow> or <down-arrow>)

This facility on the Trace menu enables you to reposition the trace screen so that you can have an unobstructed view of your user screen. Use the **Home** key to move the trace screen to the top of the screen, the **End** key to move it to the bottom of the screen and the <up-arrow> and <down-arrow> keys to move the screen up or down line by line to the desired position.

11.2.1.5 View Parameters (<right-arrow>|, |<left-arrow>)

This facility on the Trace menu enables you to view the parameters defined for the current dialog function without actually executing the function. Press <right-arrow>| to look at the next parameter or |<left-arrow> to look at the previous one.

11.2.2 Trap On/Off (F3)

Pressing **F3=trap-on/off** from the Trap Screen menu toggles the trap facility on and off. The default is trap on.

When you run the screenset with the trap on, all exits from the run-time system return to the trap screen. Control does not return to the trap if Dialog System is waiting for input; it only returns when dialog functions are being executed. Therefore, to force a return to the trap, you might have to press a key that has dialog associated with it.

When you run the screenset with the trap off, the return to the trap is disabled, even when an RETC function is invoked. To return to the trap screen before the execution of a function that would naturally return to the trap, you must press **Ctrl+Break** while the screenset is running (Windows only). The trap is then be switched on, and you can press **Enter** to continue running the screenset.

It is also possible to make the call to the Dialog System Run-time System go via the trap. This can be useful when you debug your application program. To do this, change the CALL statement in your COBOL program from:

```
call "DSRUN" using DS-CONTROL-BLOCK, screenset-DATA-BLOCK.
```

to:

Windows: `call "DSC" using DS-CONTROL-BLOCK, screenset-DATA-BLOCK.`

UNIX: `call "DSC.gnt" using DS-CONTROL-BLOCK, screenset-DATA-BLOCK.`

where *screenset* is the name of your screenset.

See the chapter *Programming* for more details of calling Dialog System.

This call displays the trap. However, remember the following points when you call the trap this way:

- In Windows environments, the file **ds.gnt** is in the library **ds.lbr**, so you must ensure that the file **ds.lbr** is in your current directory before the call is made. This restriction does not apply to UNIX.
- You cannot change the screenset name or the control value in the trap.

- The trap defaults to trace mode. However, you can turn the trace off from the trap screen if required.

11.2.3 Initialize Data Fields (F4)

Pressing **F4=initialize-data-fields** from the Trap Screen menu initializes all numeric fields in the Data Block to zero and all other fields to spaces. You can view this in the trap data field display in the lower part of the trap screen.

11.2.4 Page Data Fields (PgUp, PgDn)

Pressing **PgUp** and **PgDn** keys from the Trap Screen menu enables you to page up and down the data field values displayed on the trap screen as required. You cannot change these fields from the trap.

11.2.5 Escape (Esc)

Pressing **Escape** from the Trap Screen menu returns you to the point of entry to the trap. If you started the trap by selecting **F6=run** from the main menu, you return to the main menu. If you invoked the trap from the command line using the `/t` option, you return to the operating system.

The trap enables you to evaluate and change the run-time behavior of the screenset without needing to produce a program. You can use the information you viewed using the trap to write a COBOL application program. You need an application program to control any changes in data values as a result of actions such as file access and calculations. A sample COBOL program called **customer.cbl** is provided with the Dialog System software.

12 Generating the Copyfile

The seventh step in developing your Dialog System application is to generate a COBOL copyfile. This copyfile, which is simply a file referenced in a COBOL COPY statement, is based on the Data Block you created for your screenset during data definition (see the chapter *Data Definition* for details).

This chapter explains how to generate the COBOL copyfile that your application program can use.

To use the Dialog System Run-time System from an application program, you need a Data Block and a control block. There are two versions of the control block provided with the Dialog System Run-time System software; see the section *Control Blocks* in the chapter *Running the Screenset*. There is a facility to generate the Data Block provided with the definition software.

The program that uses the screenset must have a copy of the Data Block information that matches the screenset. For example, the field order and field sizes created in data definition must correspond exactly with those in the Data Block used in the call interface. Therefore, if you make any changes to your screenset data, you must generate another copyfile. The names of the fields in the Data Block and the copyfile do not have to match, although it can be useful to use the same names.

12.1 Generate Prompt Menu (F5)

Pressing **F5=generate** from the Main menu invokes the Generate Prompt menu shown in Figure 12-1. This prompt defaults to the name of the current screenset, with a file extension of **.cpb**. This menu provides the standard help option, the directory option, and a Control menu.

Figure 12-1. Generate Prompt Menu



```
Generate                                     Row:01=Col:01=Ins-Caps-Num-Scroll
F1=help F2=directory                          Escape
File E:\DS\DEMO\CUSTOMER.CPB                  ← Ctrl
```

To generate a copyfile with a different name than the one displayed at the prompt, type the name you want, with a **.cpb** extension.

You can use the **F2=directory** option to view the list of existing screenset names and ensure that you do not replicate a name. For more information about using the directory option, see the section *Load Screenset (F3)* in the chapter *Using Dialog System*. When you use this option, a message asks whether you wish to overwrite the existing Data Block copyfile.

To select a screenset from a different drive (Windows only) or directory, you can use the facilities available from the Control menu, described in the next section.

12.2 Generate Prompt Control Menu (Ctrl)

Pressing **Ctrl** from the Generate Prompt menu invokes the Control menu shown in Figure 12-2, which enables you to view the current default drive or directory and either to select the default to use in the file prompt, or change the drive or directory.

Note: References to the drive in this, and the following, sections are relevant only in the Windows environment and do not apply if you are using UNIX.

Figure 12-2. Generate Prompt Control Menu

```
Generate Row:01-Col:01=Ins-Caps-Nun-Scroll
F1=help F2/F3/F4=show/use/set-def-dir F5/F6=call/canct-lbr F8=brs F9=ren F10=del
File E:\DS\DEMO\CUSTOMER.CPB
```

12.2.1 Show Default Directory (F2)

Pressing **F2=show-def-dir** from the Generate Prompt Ctrl menu displays the current default drive or directory on the prompt line at the bottom of the menu.

12.2.2 Use Default Directory (F3)

Pressing **F3=use-def-dir** from the Generate Prompt Ctrl menu replaces the drive or directory in the file string with the current default.

12.2.3 Set Default Directory (F4)

Pressing **F4=set-def-dir** from the Generate Prompt Ctrl menu changes the current default to the drive or directory displayed in the file prompt. To change the default, overwrite the current entry with the desired drive or directory and press **F4**. If concurrency is in operation, a message asks you to confirm before the default is changed, because the new default takes effect for all run units.

12.2.4 Call Library (F5)

Pressing **F5=call-lbr** from the Generate Prompt Ctrl menu enables you to call a specified library and make the files in that library temporarily available to the system. To load a library (**.lbr**) catalog, type in the name of the library catalog file on the prompt line.

12.2.5 Cancel Library (F6)

Pressing **F6=cancel-lbr** from the Generate Prompt Ctrl menu cancels the specified library catalog file and makes the files in that library unavailable to the system.

12.2.6 Browse File (F8)

Pressing **F8=brs** from the Generate Prompt Ctrl menu enables you to view the specified file's copyfile, as long as copyfile is in readable text format. To browse a file in a library, specify *path\library-name.lbr\filename*. Use the **PgDn**, <down-arrow>, or **Enter** keys to move forward through the file. Press **Escape** to return to the Generate Prompt menu.

12.2.7 Rename File (F9)

Pressing **F9=ren** from the Generate Prompt Ctrl menu invokes the Rename File menu, shown in Figure 12-3 (this screen is a Windows example).

Figure 12-3. Rename File Menu (Windows)

```

Rename-E:\DS\DEMO\CUSTOMER.CPB          Ins-Caps-Num-Scroll
F1=help F2=directory Escape
File E:\DS\DEMO\CUSTOMER.CPB           ← Ctrl
Do not change drive
  
```

To rename a file in a library, overwrite the current file name on the prompt line with the desired name and press **Enter**. You must not change the drive from this menu (Windows only) and you cannot use global characters (for example *).

12.2.8 Delete File (F10)

Pressing **F10=del** from the Generate Prompt Ctrl menu deletes the file specified in the file prompt. You can use global characters; however, a

message asks for confirmation before individual files are deleted. If you specify only the drive and path, the filename *.* is assumed.

When you delete files from a library, the files still occupy space in the library catalog file. If you have already loaded the library catalog when you delete a file from a library, the deleted file remains accessible until the catalog is reloaded, for example, if you cancel and then call the library.

12.3 Generate COBOL Menu (Enter)

You use this menu to generate a copyfile for your screenset. Use the menu options described earlier in this chapter to place the filename you require at the prompt on the Generate Prompt menu. Press **Enter**, which invokes the Generate COBOL menu. If you have previously generated a copybook with this file name, the following prompt first appears:

```
File already exists. Overwrite? Y/N
```

Your program requires the most up-to-date version of your data definition, so press "Y" to generate a new copyfile for this screenset. The Generate COBOL menu shown in Figure 12-4 is displayed.

Figure 12-4. Generate COBOL Menu

```
Generate_____Row:01=Col:01=Ins-Caps-Nun-Scroll
F1=help F2=data-descriptions F3=prefix on/off F4=screenset/user-validations
Escape
```

12.3.1 Data Descriptions (F2)

Pressing **F2=****data-descriptions** from the Generate COBOL menu generates the Data Block. A message is displayed at the bottom of the screen:

```
Generating COPY-file - Please wait.....
```

When this message is removed from the screen, the Data Block has been generated. Press **Escape** to return to the Main menu. You can use an editor to view the generated copyfile, which is called **screenset.cbp**. The copyfile contains two sections; *screenset-DATA-BLOCK* and *screenset-FIELD-NUMBERS*.

Data Block	<p>The first field in this section contains the Data Block version number. When you write your application program, you must move this number to Ds-Data-Block-Version-No in the control block. At run time, the value in Ds-Data-Block-Version-No is matched against a value stored in the screenset.</p> <p>The second field is the <i>screenset-VERSION-NO</i> and is provided for information.</p> <p>The third field is the <i>screenset-BUILD-NO</i> and is provided for information.</p> <p>The remainder of the Data Block contains the data fields you defined.</p> <p>The Data Block also contains level 88 constants for any single-digit numeric data items. These data items are often used as program flags, so these level 88 constants can be useful in EVALUATE statements. You do not have to include these 88s if they are not required and can be configured out of the defaults file dsdef.cfg (see the chapter <i>Setting Up the Configuration File</i> for details).</p>
Field Numbers	<p>This section is a set of constants that contain the internal field numbers of all of the fields in the Data Block. Fields are numbered sequentially according to their order in data definition.</p> <p>You might need to refer to these constants, for example, if you use a function that requires a field number as a parameter.</p>

Your application program uses this generated copyfile in conjunction with one of the control blocks provided with Dialog System. The following example shows the Data Block and field numbers generated for a screenset named CUSTOMER using the default options in the configuration file:

```
*****
* Data Block for Set CUSTOMER
*****

01 CUSTOMER-DATA-BLOCK-VERSION-NO  PIC 9(4) VALUE 36.
01 CUSTOMER-VERSION-NO              PIC 9      VALUE 1.
01 CUSTOMER-SET-BUILD-NO            PIC 9(4) VALUE 50.

01 CUSTOMER-DATA-BLOCK.
  03 CUSTOMER-C-CODE                 PIC X(5).
  03 CUSTOMER-C-NAME                 PIC X(15).
  03 CUSTOMER-C-ADDR1                PIC X(15).
  03 CUSTOMER-C-ADDR2                PIC X(15).
  03 CUSTOMER-C-ADDR3                PIC X(15).
  03 CUSTOMER-C-ADDR4                PIC X(15).
  03 CUSTOMER-C-LIMIT                PIC S9(4).
  03 CUSTOMER-C-AREA                 PIC X.
  03 CUSTOMER-GROUP-001.
    04 CUSTOMER-GROUP-ITEM-001 OCCURS 10.
      05 CUSTOMER-ORD-NO              PIC S9(6).
      05 CUSTOMER-ORD-DATE            PIC 9(6).
      05 CUSTOMER-ORD-VAL             PIC 9(4)V9(2).
      05 CUSTOMER-PAY-VAL             PIC S9(4)V9(2).
      05 CUSTOMER-ORD-BAL             PIC S9(4)V9(2).
  03 CUSTOMER-C-BAL                  PIC S9(5)V9(2).
  03 CUSTOMER-GROUP-002.
    05 CUSTOMER-DEL-FLG              PIC 9.
      88 CUSTOMER-DEL-FLG-TRUE      VALUE 1.
    05 CUSTOMER-LOAD-FLG            PIC 9.
      88 CUSTOMER-LOAD-FLG-TRUE     VALUE 1.

    05 CUSTOMER-SAVE-FLG             PIC 9.
      88 CUSTOMER-SAVE-FLG-TRUE     VALUE 1.
    05 CUSTOMER-CLR-FLG             PIC 9.
      88 CUSTOMER-CLR-FLG-TRUE     VALUE 1.
    05 CUSTOMER-EXIT-FLG           PIC 9.
      88 CUSTOMER-EXIT-FLG-TRUE    VALUE 1.
  03 CUSTOMER-ERR-MSG                PIC X(20).

*****
* End of Data Block for Set CUSTOMER
*****
```

```

*****
* Field Numbers for Set CUSTOMER
*****

01 CUSTOMER-FIELD-NUMBERS.
   03 CUSTOMER-FLD-NO-C-CODE      PIC 9(4) COMP VALUE 1.
   03 CUSTOMER-FLD-NO-C-NAME      PIC 9(4) COMP VALUE 2.
   03 CUSTOMER-FLD-NO-C-ADDR1     PIC 9(4) COMP VALUE 3.
   03 CUSTOMER-FLD-NO-C-ADDR2     PIC 9(4) COMP VALUE 4.
   03 CUSTOMER-FLD-NO-C-ADDR3     PIC 9(4) COMP VALUE 5.
   03 CUSTOMER-FLD-NO-C-ADDR4     PIC 9(4) COMP VALUE 6.
   03 CUSTOMER-FLD-NO-C-LIMIT     PIC 9(4) COMP VALUE 7.
   03 CUSTOMER-FLD-NO-C-AREA      PIC 9(4) COMP VALUE 8.
   03 CUSTOMER-FLD-NO-ORD-NO      PIC 9(4) COMP VALUE 9.
   03 CUSTOMER-FLD-NO-ORD-DATE    PIC 9(4) COMP VALUE 10.
   03 CUSTOMER-FLD-NO-ORD-VAL     PIC 9(4) COMP VALUE 11.
   03 CUSTOMER-FLD-NO-PAY-VAL     PIC 9(4) COMP VALUE 12.
   03 CUSTOMER-FLD-NO-ORD-BAL     PIC 9(4) COMP VALUE 13.
   03 CUSTOMER-FLD-NO-C-BAL       PIC 9(4) COMP VALUE 14.
   03 CUSTOMER-FLD-NO-DEL-FLG     PIC 9(4) COMP VALUE 15.
   03 CUSTOMER-FLD-NO-LOAD-FLG    PIC 9(4) COMP VALUE 16.
   03 CUSTOMER-FLD-NO-SAVE-FLG    PIC 9(4) COMP VALUE 17.
   03 CUSTOMER-FLD-NO-CLR-FLG     PIC 9(4) COMP VALUE 18.
   03 CUSTOMER-FLD-NO-EXIT-FLG    PIC 9(4) COMP VALUE 19.
   03 CUSTOMER-FLD-NO-ERR-MSG     PIC 9(4) COMP VALUE 20.

*****
* End of Field Numbers for Set CUSTOMER
*****

```

The chapter *Running the Screenshot* shows how the information contained in the Data Block is used in the call interface to the COBOL application program.

12.3.2 Prefix On/Off (F3)

Pressing **F3=prefix on/off** from the Generate menu enables you to specify whether or not data item names in the generated copyfile are prefixed with the screenshot name.

Pressing **F3** toggles the setting between the following values:

- ON Data item names have the screenshot name as prefix.
- OFF Data item names do not have the screenshot name as prefix.

13 Programming

The eighth step in developing your Dialog System application is to write a calling program. Dialog System has several advanced features that enable you to create more sophisticated programs. It is recommended that you do not try these advanced features until you are thoroughly familiar with the basic features of Dialog System.

This chapter explains:

- How to call Dialog System from your application program
- How to use multiple screensets from one program
- How to use one or more error files directly from the program
- How to control user-defined field formats
- How to control every keystroke
- How to obtain system information from Dialog System
- How to obtain files from other directories
- Restrictions on compiling Dialog System applications

13.1 Calling Dialog System

The screenset runs using a simple call interface between your program and the Dialog System Run-time System and uses the Data Block you created and a control block provided with the system. The call statement for this interface passes two parameters:

```
call "DSRUN" using DS-CONTROL-BLOCK,  
                  screenset-DATA-BLOCK.
```

where:

screenset is the name of the screenset to run. The run-time system searches for the screenset either in the current directory, or in the directory specified by the P option (see the section Run time Path Name Support later in this chapter), then finally in the directories specified by the environment variable COBDIR (or COBPATh under UNIX), in order.

The contents and values of the control block and Data Block are described in the chapters *Running the Screenset* and *Generating the Copyfile*.

13.2 Using Multiple Screensets from One Program

Occasionally, you might wish one program to use two or more screensets. If these screensets occur successively in the program, each one can be called by setting the Ds-Control field in the control block to a value of "N" (start running new screenset). The previous screenset is canceled.

However, to start a new screenset and, on completion, return to the point from which you left the original screenset, you must use the facilities to POP and PUSH screensets. To do this, you set the following values in the Ds-Control field:

- Q quit screenset (78 Ds-Quit-Set). This quits running the current screenset and pops to the last screenset pushed. If there is no screenset pushed, the current screenset simply unloads itself.
- S push screenset (78 Ds-Push-Set). This pushes the current screenset onto a stack and starts the new one.
- U use screenset (78 Ds-Use-Set). This causes the current screenset to be swapped with the named screenset on the push stack. Execution then continues as if 'C' had been issued. This feature is supported for Micro Focus Panels screensets only. If there is no named screenset on the stack, an error 4 (invalid screenset) is returned in DS-SYSTEM-ERROR-NO.
- W what screenset (78 Ds-What-Set). This tells the application the name of the current screenset. This is returned in DS-SET-NAME.

To start a new screenset (screenset number two), and push the current screenset (screenset number one), you must place a value of "S" rather than "N" in the Ds-Control field for the new screenset.

To keep the current screen visible (screenset number one), you must also set the Ds-Control-Param field to a value of one to override the default. (The default setting clears the current screen before a new screenset is started.)

To terminate the second screenset, there must be a call to Dsrun placing a value of "Q" in the Ds-Control field. This call quits screenset number two and pops back to screenset number one.

When you use these facilities, remember the following points:

- The trap does not support the screenset PUSH and POP facilities, so you cannot fully test the way the screensets are pushed and popped using the trap.
- If you push and pop screensets that are a mixture of Micro Focus Panels screensets and non-Panels screensets, there can be unpredictable results.
- Screensets that have been pushed are stored in memory, which is automatically paged to disk if memory usage is high. This could be a problem on diskless workstations in Windows environments.
- When you push screensets, their associated error files are closed. Ensure that you open the error file again when you pop the screenset, as described in the next section.

13.3 Using Error Files Directly from Your Program

There might be situations where you wish to use the Dialog System error file directly from your program (see the chapter *Data Definition* for details on setting up the error file). For example, if you have a complex validation that Dialog System cannot perform, the input must be validated by your program. Normally, your program must also deal with the use of any associated error messages. However, Dialog System enables you to hold these error messages in its own error message file.

Also, Dialog System enables you to access multiple error files from other sources via the calling program. You can access up to 15 files not defined in Dialog System without using the run-time system. Because this facility is used from the calling program, it does not affect the normal error message handling inside the Dialog System Run-time System, which continues to use the error message file created during dialog definition.

When you specify an error message file, the run-time system searches for the file in the current directory, then in any directory specified by the P option (see the section Run time Path Name Support later in this chapter), then finally in the directories specified by the environment variable COBDIR (or COBPATH under UNIX), in order.

13.3.1 The Dialog System Error File

To access this error file from the program, ensure that the error file is open, then read the required error message. You do not need a call to close this error file because the run-time system closes all files when the screenset is terminated.

The error file is often already be open because of earlier errors in user input. However, to ensure that it is open, you can set the value in the Ds-Control field of the control block immediately following the call to start a new screenset ("N").

Use the following value:

E open error file (78 Ds-Err-File-Open). This opens the error file for use by the run-time system.

This "E" call does no harm if the file is already open.

If you have pushed a screenset (Ds-Control set to "S"), its associated error file is closed. When you pop the screenset (Ds-Control set to "Q"), make a new call to Dsrun with Ds-Control set to "E" to reopen the error file.

To read the error file, use the following COBOL call:

```
call "DSERRHAN" using ERROR-FILE-LINKAGE
                        ERROR-FILE-DATA
```

where ERROR-FILE-LINKAGE and ERROR-FILE-DATA are:

```

01 ERROR-FILE-LINKAGE .
   03 SHORT-FILE-NAME          PIC X(8) .
   03 FILE-ACCESS              PIC XX .
   03 FILLER                   PIC X(6) .
   03 E-RETURN-CODE           PIC XX .

01 ERROR-FILE-DATA .
   03 ERROR-RECORD-NUMBER      PIC 9(4) COMP .
   03 ERROR-RECORD-CONTENTS    PIC X(76) .

```

For example, to find the contents of error message number 135 in an error file called **customer.err**, place the following values in ERROR-FILE-LINKAGE and ERROR-FILE-DATA.

```

SHORT-FILE-NAME "CUSTOMER"
FILE-ACCESS     "R" {read}
ERROR-RECORD-NUMBER "135"

```

If record 135 is not found, the return value is:

```
E-RETURN-CODE "NF"
```

If record 135 is found, the return value is:

```
E-RETURN-CODE "OK"
```

and the text of the error record is contained in:

```
ERROR-RECORD-CONTENTS
```

13.3.2 Alternative Error Files

To access other error files from your application program, you can make direct calls to the Dserrhan file provided with the Dialog System software. This facility enables you to open, read and close other files.

The following example shows how you can use the Dserrhan file to open an alternative error file.

For an error file called **cust1.err**, in a directory **c:\files\errors**, you must set up the following values:

```
SHORT-FILE-NAME "CUST1 "
FILE-ACCESS     "NI "
ERROR-FILE-DATA "C:\FILES\ERRORS\CUST1.ERR"
```

The call to Dserrhan is:

```
call "DSERRHAN" using ERROR-FILE-LINKAGE
                        ERROR-FILE-DATA
```

To read alternative error files, use exactly the same COBOL call as you do for the Dialog System error file:

```
call "DSERRHAN" using ERROR-FILE-LINKAGE
                        ERROR-FILE-DATA
```

However, unlike the Dialog System error file, you must close the alternative error file. To close the file, you must make a call to Dserrhan using the following values:

```
SHORT-FILE-NAME "CUST1 "
FILE-ACCESS     "NC "
```

13.4 User Validation in Screensets

If you want to use user-validation on your screensets, you must write a module called DSUSRVAL. Your Micro Focus COBOL system includes a skeleton source of this module, containing comments to help you add your own code.

The Dialog System Run-time System calls Dsusrval when you are about to exit a field.

The parameters passed to the routine are:

Dialog Control Block	input	Use this to determine the current screenset, panel and field. Any changes you make to this Control Block are ignored by Dialog System.
Screenset Data Block	input/output	You can access and change any fields for the current screenset in this Data Block.
DS-User-Val-Status	output	If you want to flag a field validation failure, set this flag to 1. If validation is OK, set this field to 0.
DS-User-Error-Number	output	This field is ignored unless DS-User-Val-Status is non-zero. When a field validation error has occurred, if this field is set to 0, the error message associated with this validation will be displayed. You can force a different error message to be displayed by placing its number in this field.

13.5 Control of User-defined Field Formats

Dialog System provides a number of screen field formats, which depend on the original definition of the field. For example, input to a numeric field can be zero suppressed or signed. The formats provided are sufficient for most applications. However, you can configure the field formats for any input or output fields. To do this, you must write your own program, and call it DSUSRFMT.

The Dialog System Run-time System calls Dsusrfmt after every keystroke in the field. The field Ds-User-Format-Number is set to the value (in the range 1 to 4) that you associate with the field in the Panel Field menu. You must provide the correct behavior for the field for any access; however, you do not always need to define the behavior

completely because you can instruct Dialog System to format the field according to its underlying main format. Your `Dsusrfmt` program must never make any screen input or output.

The Dialog System software includes an example user format program called `dsusrfmt.cbl`, together with a screenset called `dsusrfmt.s` that uses it. To use this program, compile it and place it in the same directory as the screenset.

You must write `Dsusrfmt` with a Linkage Section containing the following items:

- 1 A control block. See the section *Control Block for dsusrfmt* for a description of its contents.
- 2 A field buffer containing the current contents of the field, with a decimal point if necessary. The size of the buffer, which varies depending on the size of the field, is specified in one of the control block fields (03 `Ds-Buffer-Length`). The buffer size can be larger than the size of the field on the panel; however, the user format routine must not access any data item larger than the current size of the field because if it does, undefined errors can occur. When you access this buffer, it is advisable to use reference modification.
- 3 The Data Block generated for the screenset that `Dsusrfmt` uses.

13.5.1 Control Block for `Dsusrfmt`

The contents of the control block are:

```

01 Ds-User-Linkage.
   03 Ds-Access-Flag                pic x.
   03 Ds-Screenset-Name             pic x(8).
   03 Ds-Panel-Name                 pic x(8).
   03 Ds-Field-Name                 pic x(8).
   03 Ds-Field-Offset               pic 9(4) comp-x.
   03 Ds-Field-Screen-Length        pic 9(4) comp-x.
   03 Ds-Master-Forma               pic 9(2) comp-x.
   03 Ds-Screen-Forma               pic 9(2) comp-x.
   03 Ds-Screen-Subforma            pic 9(2) comp-x.
   03 Ds-User-Format-Number         pic 9(2) comp-x.
   03 Ds-Length                     pic 9(4) comp-x.
   03 Ds-Decimal-Length             pic 9(2) comp-x.
   03 Ds-Sign                       pic 9(2) comp-x.
   03 Ds-Buffer-Length              pic 9(4) comp-x.
```

```

03 Ds-Shift-State                pic 9(2) comp.
03 Ds-Lock-State                pic 9(2) comp-x.
03 Ds-Cursor-Position          pic 9(2) comp-x.
03 Ds-Sign-Value                pic 9(2) comp-x.
03 Ds-Buffer-Screen-Offset     pic 9(4) comp-x.
03 Ds-Flags.
    05 Ds-Display-Field-Flag    pic 9(2) comp-x.
    05 Ds-Display-Cursor-Flag  pic 9(2) comp-x.
    05 Ds-Autoskip-Left        pic 9(2) comp-x.
    05 Ds-Autoskip-Right      pic 9(2) comp-x.
    05 Ds-Do-Internal-Formatting pic 9(2) comp-x.
03 Ds-Input-Key.
    05 Ds-Key-Type              pic 9(2) comp-x.
    05 Ds-Key-Description       pic 9(2) comp-x.
    05 Ds-Key-Description-X redefines
        Ds-Key-Description      pic X.
    05 filler                    pic 9(2) comp-x.
03 filler                        pic x(10).
03 Ds-Field-Long-Name          pic x(30).

```

The field buffer is declared as follows:

```

01 Ds-Field-Buffer.
    03 Field-Buffer-Comp        pic 9(2) comp occurs 9999.

```

The Dialog System software also includes a copyfile, **ds-user.cpy**, which contains this control block. This copyfile contains the most recent format and naming of the Dsusrfmt control block, with appropriate descriptions. Another copyfile, **ds-key.cpy**, contains the constants that you require.

If the user presses a function key combination that generates a multi-byte scan code that Dialog System does not recognize, for example **Ctrl+Del** or **Ctrl+Ins**, an error code is returned. This is a general error code and is not specific to any key.

The control block is split into two parts; one contains fields that provide information to the user format module, and the other contains fields that provide information and control the subsequent behavior of the Dialog System Run-time System.

The following are the parameter values that Dialog System passes to the Dsusrfmt control block to provide information to the user format module:

Ds-Access-Flag	<p>A one-character flag that denotes the current stage of processing of a field so that Dsusrfmt can take the appropriate action. This flag is passed at four points during processing according to the following settings:</p> <ul style="list-style-type: none"> L when the screenset is initialized (DS-LOAD-ACCESS). I when a field is set to its default state, for example, when the panel fields are refreshed during an RFTD or a POP function, or before the first keystroke is pressed when moving onto a field for input (DS-INITIALIZE-FIELD). C after each keystroke during input on the field (DS-NORMAL-ACCESS). T when the field is completed, for example, if a key with dialog has been pressed or the user format routine has requested a move to another field (DS-TERMINATE-FIELD).
Ds-Screenset-Name	The name of the screenset currently in use.
Ds-Panel-Name	The name of the currently active panel. This enables you to specify different input behavior for different panels.
Ds-Field-Name	The field name if the name does not exceed eight characters. If the name exceeds eight characters, the field contains an indicator '@' in the first character position and binary data in the rest of the field. The field name is also contained in the field Ds-Field-Long-Name.

Ds-Field-Offset	A number in the range 0 to 1999 specifying the offset on the panel on which the current field is positioned. This might be useful if the same field occurs more than once on the current panel and you wish the different versions to behave differently.
Ds-Field-Screen-Length	The length of the field on the panel, not including the sign if present. This can be shorter than the buffer length.
Ds-Master-Format	The underlying master field format for the current field. This can have the following values: <ul style="list-style-type: none"> 1 alphanumeric 2 alphabetic 3 numeric 4 signed-numeric 19 COMP 20 signed COMP 35 COMP-X 51 COMP-3 52 signed COMP-3 67 COMP-5 68 signed COMP-5
Ds-Screen-Format	The format of the current field on the panel. This can have the following values: <ul style="list-style-type: none"> 0 date 50 numeric 100 alphabetic 150 alphanumeric
Ds-Screen-Subformat	Specifies any subformats applicable to some screen formats, such as date and numeric fields. For a date field, the following formats are valid: <ul style="list-style-type: none"> 1 XX-XX-XX 2 XX-XXX-XX 3 XX-XXX 4 XX-XX 5 XX-XX-XXXX

	For a numeric field, the following formats are valid:
	0 not zero suppressed (the default)
	2 zero suppressed
Ds-User-Format-Number	A number in the range 1 to 4 specifying the user format number of the current field.
Ds-Length	The length of alphabetic and alphanumeric fields or the length of the integer part of a numeric field.
Ds-Decimal-Length	The length of the decimal part of a numeric field, not including the decimal point. For non-numeric fields, this is zero.
Ds-Sign	Indicates whether or not there is a sign with this field on the panel, according to the following settings:
	1 sign present
	2 no sign present
Ds-Buffer-Length	The length of the formatted buffer passed from Dialog System to the user format routine.
Ds-Shift-State	The state of the shift keys (Alt , Ctrl , Left Shift and Right Shift) after they have been enabled. The state is given by a byte according to the following bit settings:
	bit 0 Right Shift
	bit 1 Left Shift
	bit 2 Control
	bit 3 Alternate
	bits 4-7 Not used
Ds-Lock-State	The state of the lock keys (Insert , Caps , Number (Windows only) and Scroll Locks) after they have been pressed. The state is given by a byte according to the following bit settings:
	bit 0 Scroll Lock
	bit 1 Num Lock (Windows only)
	bit 2 Caps lock
	bit 3 Insert lock
	bits 4-7 Not used

Ds-Input-Key

Two fields that identify the key that has just been pressed. The **Ds-Key-Type** field identifies the type of key that has been pressed, and the **Ds-Key-Description** field identifies the exact key pressed, according to the values given in the following tables.

For data keys, the key description is the ASCII code of the key pressed, except for those codes given in the second table (type 51) and **Ctrl+h**, **Ctrl+i**, and **Ctrl+m**. These three keys return function codes for <right-arrow>|, **Backspace**, and **Enter**, respectively.

These key values, which are used by the run-time system, are in the file **ds-key.cpy**, provided with the Dialog System software. If your keyboard does not have the keys shown below, please consult your Release Notes to see which keys on the standard keyboard for your system provide the equivalent functions.

Type 49	Function-Key
1-9	F1-F9
0	F10
91	F11
92	F12
61	Shift+F1
62	Shift+F2
13-19	Shift+F3-Shift+F9
20	Shift+F10
93	Shift+F11
94	Shift+F12
21-29	Ctrl+F1-Ctrl+F9
30	Ctrl+F10
95	Ctrl+F11
96	Ctrl+F12
31-39	Alt+F1-Alt+F9
40	Alt+F10
97	Alt+F11
98	Alt+F12

Type 49	Function-Key
41	Enter
42	<left-arrow>
43	<right-arrow>
44	<up-arrow>
45	<down-arrow>
46	Home
47	<right-arrow>
48	<left-arrow>
49	Backspace
50	End
51	Page Up
52	Page Down
53	Ctrl+Home
54	Ctrl+End
55	Ctrl+Page Up
56	Ctrl+Page Down
57	Ctrl+ <left-arrow>
58	Ctrl+<right-arrow>
60	Delete
65	Escape
71-79	Alt+1-Alt+9
80	Alt+0
81	Alt+Minus
82	Alt+Equals
101-102	Alt+a-Alt+z
Type 51	Data Key
1-7	Ctrl+a-Ctrl+g
10	Ctrl+j
11	Ctrl+k
12	Ctrl+l
14-26	Ctrl+n-Ctrl+z
Type 53	Shift Key
0	Alt Key On
1	Alt Key Off
2	Ctrl Key On
3	Ctrl Key Off
4	Left Shift Key On
5	Left Shift Key Off
6	Right Shift Key On
7	Right Shift Key Off

Type	Lock Key
16	Insert On
17	Caps Lock On
18	Num Lock On
19	Scroll Lock On
32	Ins Off
33	Caps Lock Off
34	Num Lock Off
35	Scroll Lock Off

The following are the parameter values that Dialog System passes to the Dsusfmt control block to provide information, and control the subsequent behavior of the Dialog System Run-time System.

Ds-Cursor-Position	The current position of the cursor in the field on the panel starting from position one. It should not exceed the value of Ds-Field-Screen-Length; otherwise, the cursor can appear outside the current field.
Ds-Sign-Value	Indicates the current value of the sign of numeric fields, according to the following settings: 0 negative 1 positive
Ds-Buffer-Screen-Offset	The offset of the first character in the field buffer that is shown in position one of the current field on the panel. This offset starts from position one.
Ds-Display-Field-Flag	A flag that determines whether the field is redisplayed on the panel. For example, key "A" is given as a value 51 in Ds-Key-Type and 65 in Ds-Key-Description, but it is not in the buffer. If the user presses key "A", it does not appear on the panel unless you place it in the appropriate place in Ds-Field-Buffer and set the Ds-Display-Field-Flag to true.
Ds-Display-Cursor-Flag	A flag that determines whether the cursor is displayed when control returns to the run-time software. This is usually required if the user format routine has moved the cursor to a new position.

Ds-Autoskip-Left	A flag that makes Dialog System return control to the previous input field. Only effective if DS-Do-Internal Formatting is set to 0. This parameter setting may be used when DS-Access-Flag is "I" or "C".
Ds-Autoskip-Right	A flag that makes Dialog System move control to the next input field. Only effective if DS-Do-Internal Formatting is set to 0. This parameter setting may be used when DS-Access-Flag is "I" or "C".
Ds-Do-Internal-Formatting	<p>A flag that determines whether Dialog System performs its own formatting routines according to the screen format of the current field, according to the following settings. This enables the user format routine to slightly modify the input behavior of a field without providing all of the field handling code.</p> <p>0 do nothing</p> <p>1 perform the internal formatting. This is the default Dialog System sets after each keystroke.</p>
Ds-Input-Key	Two fields that identify the key that has just been pressed. The Ds-Key-Type field identifies the type of key that has been pressed, and the Ds-Key-Description field identifies the exact key pressed, according to the values given in the tables in the description of the Ds-Input-key in the section <i>Control of User-defined Field Formats</i> . This field can be changed to return to Dialog System a different key value than normally given by the key actually pressed.
Ds-Field-Long-Name	A 30-character field name, containing the name of the current field.

13.6 User Control of Every Keystroke

Dialog System enables you to write a program that can intercept every keystroke and modify its value to any other key value you wish. If you write such a program, you must call it `Dsusrtrn`.

To use this program, you must set the **F4=key-translation-off/on** switch from the Screenset Switches menu in the Main Alt menu. When you set this switch on, the Dialog System Run-time System calls the `Dsusrtrn` program for every keystroke. You can then use this program to change values of keystrokes. Your `Dsusrtrn` program must never make any screen input or output. See the section *Screenset Switches (F5)* in the chapter *Using Dialog System*.

If you use both this program and the `Dsusrfmt` program described in the previous section, the order you call them is `Dsusrtrn` followed by `Dsusrfmt`.

You must write `Dsusrtrn` with the Linkage Section, **ds-tran.cpy**, given below. The **ds-key.cpy** file provided with the Dialog System software contains the constants that you require for this program. The Dialog System Run-time System sends certain values in the Linkage Section, while you can change certain others of these values to affect the keystroke.

The contents of the **ds-tran.cpy** file are:

```
*****
* Interface to called user-defined key translation module
*****

01 ds-tran-linkage.
   03 t-access-flag          pic x.
       78 t-load-access      value "L".
       78 t-normal-access    value "C".
   03 t-input-key.
       05 t-key-type         pic 9(2) comp.
       05 t-key-description  pic 9(2) comp.
       05 filler             pic 9(2) comp.
   03 filler                 pic x(10).
```

The Dialog System Run-time System sends the following parameter values to Dsusrtrn:

T-Access-Flag	<p>A single-character flag that denotes the current stage of processing of a field so that the Dsusrtrn program can take the appropriate action. This flag has the following settings:</p> <ul style="list-style-type: none"> L when the screenset is initialized. This flag is automatically set to "L" on the "N" call to Dsrun, so the first call to Dsusrtrn has a value "L" in this field. C while the screenset is running. Any calls after the screenset is initialized have a value "C" in this field.
T-Input-Key	<p>Two fields that identify the key that has just been pressed. The Ds-Key-Type field identifies the type of key that has been pressed, and the Ds-Key-Description field identifies the exact key pressed, according to the values given in the tables in the description of the Ds-Input-Key in the section <i>Control of User-Defined Field Formats</i>. You can alter these keys in your program to change the value of any key that is pressed.</p>

13.7 Obtaining System Information from Dialog System

You can determine what was the last key pressed and where the cursor is currently positioned in the application program by specifying a third parameter on the call to Dsrun. The format of this parameter is:

```

01 Ds-Third-Param.
  03 Last-Key-Pressed
    05 Key-Type-Hex      pic x.
    05 Key-Type-Decimal redefines Key-Type-Hex
                          pic 9(2) comp
    05 Actual-Key-Hex   pic x.
    05 Actual-Key-Decimal redefines Actual-Key-Hex
                          pic 9(2) comp.

```

```

03 Cursor-Position.
   05 Cursor-Row      pic 9(2) comp.
   05 Cursor-Column  pic 9(2) comp.

```

The following parameter values are returned by this call:

Last-Key-Pressed	The key code for the last key pressed (see the description of the Ds-Input-Key in the section <i>Control of User-defined Field Formats</i> for key code tables).
Cursor-Position	The position of the cursor relative to the screen. This is only set if there is currently field input on the active panel.

13.8 Run-time Path Name Support

You can instruct the Dialog System Run-time System to look outside the current directory for screensets and error files by prepending the desired path to a filename. You specify this current path with a second parameter in the call to Dsrun:

```
01 Current-Path PIC X(52).
```

You must then call Dsrun with the following value in the Ds-Control field of the control block:

- P Indicate path name. This specifies the name of the path to be followed (78 Ds-Pathname). A pathname of spaces causes the search to reset to the current directory.

This call to set the current path to a named location instead of the current directory in the run-time software can be made at any time. If a screenset or error message file is not found in the current directory or the directory specified by the P option, Dialog System searches the directories specified by the environment variable \$COBDIR (or \$COBPATh under UNIX), in order.

13.9 Compiling Dialog System Applications

When you are compiling a Dialog System application, there is a restriction:

- Do not compile the application with the directive `MS"2"`.

14 Linking

The eleventh, optional, step in creating your Dialog System application, is to link Dialog System object modules to your application to create a single executable object.

This chapter explains:

- The Dialog System object modules that are provided
- How to link Dialog System object modules to your application
- How to use the trace facility when you test the executable object

14.1 Dialog System Object Modules

The following Dialog System object modules are provided:

Module (Windows)	Module (UNIX)	Contents
dstrun.obj	DSRUN.o	Standard entry module to Dialog System (character and graphical)
dscrun	DSCRUN.o	Interface to character DSCHAR
dsfld.obj	DSFLD.o	Screen field handler
dsadskey.obj		Keyboard handler
dsrtns.obj		Service routines for shifting and filling data
dscvalrn.obj	DSCVALRN.o	Field validation routine
dserrhan.obj	DSERRHAN.o	Error file handler
dstracer.obj	DSTRACER.o	Dialog trace routine
dsdlgini.obj	DSGLINI.o	Initialization for trace routine
dsc.obj	DSC.o	Replacement for Dstrun if the application calls Dialog System via the Trap

Module (Windows)	Module (UNIX)	Contents
dsgetss.obj	DSGETSS.o	Loads screensets
	DSTERM.o	Terminal type dependencies
	DSLCONV.o	Native line drawing routine
	DSUSRCAL.o	User routine interface
	DSUXSYSP.o	System file locator
	DSVRSCRN.o	Handles variable screen size for menu placement
dscomp.obj	DSCOMP.o	Comp data type handler
dspanels.obj	PANELS.o	Micro Focus panels
dsnlrtn.obj	DSNLSRTN.o	NLS support routine - needed only if NLS case-folding is required
	DSUSRVAL.o	User validation
	DSUSRFMT.o	User field formatting

Depending on the features your application uses, you might not require all of these object modules. Under UNIX, use `-X module-name` to exclude any such modules. The following sections list the object modules that production applications and applications in development require.

14.1.1 Production Applications

The following table lists the object modules required by production applications:

Application Type	Windows	UNIX
All Dialog System applications	dscrn.obj dsfld.obj dsadskey.obj dsrtns.obj	DSCRUN.o DSFLD.o DSTERM.o DSLCONV.o DSUSRCAL.o DSUXSYSP.o
Applications using field validation	dscvalrn.obj dserrhan.obj	DSCVALRN.o DSERRHAN.o
Applications using user validation	dsusrval.obj	DSUSRVAL.o

Application Type	Windows	UNIX
Applications using comp, comp-x, comp-5, comp-3 data types	dscomp.obj	DSCOMP.o
Applications using field formatting	dsusrfmt.obj	DSUSRFMT.o
Applications using Micro Focus PANELS	dspanels.obj	PANELS.o

14.1.2 Applications in Development

The following table lists the object modules required by applications in development:

Application Type	Windows	UNIX
Applications that use the trace facility	dstracer.obj dsdlgini.obj	DSTRACER.o DSTRACER.o
Applications that call Dialog System via the Trap	dsc.obj dsgetss.obj	DSC.o DSGETSS.o

For more information about using trace, see the section *Using the Trace Facility in an Executable Object*.

When you have fully tested an application, you should re-link the application without any of these modules to create a smaller production object.

14.2 Linking Your Application

Windows: To link your program on Windows systems, use the following command line:

```
link prog+@ds-link-file,prog,,coblib;
```

where:

prog is the name of your application.

ds-link-file is the name of a linker response file. This is a text file that contains a list of all the Dialog System object modules to be linked to your application.

If you are using file handling, you must link in ExtFH. If you are using user formatting, you must link in Dsusrfmt. If you are using user key translation, you must link in Dsusrtrn.

The following linker response files are provided with Dialog System:

dsc.lnk	Contains all the object modules listed in the section <i>Dialog System Object Modules</i> except those that are required for debugging purposes only. If you use this file, your application definitely has all the dialog objects that it needs. However, if your application does not use all the Dialog System features (such as field validation), you create a larger object than is necessary.
dsbasic.lnk	Contains only those modules that are required for any Dialog System application to run, which are: dsrun.obj dscrn.obj dsfld.obj dsadskey.obj dsrtns.obj
dscdebug.lnk	Contains all the object modules required for debugging a Dialog System application.

Examples

Windows: `link customer+extfh+dsusrfmt+dsc.lnk,customer,,coblib;`

Links ExtFH and Dsusrfmt for file handling and user formatting.

You can, of course, create your own custom linker response file.

UNIX: To link your program on UNIX, use a command line of the form:

```
cob -xv prog module-name...
```

where:

prog is the name to be given to the executable object

module-name is the name of an object module to be linked into the object.

14.2.1 Using Dsclink for Linking

Dsclink is a utility that generates the appropriate linker files and calls the linker to create an executable file. You can use it directly or use it as a template for your own linking procedure to suit your application. It is supplied as a Windows batch file.

```
dsclink object-name[+object-name...] /d /t /nc /nv /p /nls
/st
```

where the parameters are:

<i>object-name</i>	An object name. The executable (.exe file) is given the same name as the first object filename. You can specify multiple object names by concatenating them with the plus sign. You cannot specify a linker response file in this list because Dsclink creates one and you can use only one as input to the linker.
/d	Debug version. Specify this parameter to use the Trap option, which also enables you to use Trace.
/t	Trace version. Specify this parameter to use the Trace option without the Trap. If you also specify the /d parameter, it overrides the /t parameter.
/nc	No COMP support.
/nv	No field validation support.
/p	Panels support.
/nls	National Language (case folding) support. If you use the supplied utility dsclink.bat (Windows), a switch allows you to control this simply. The linker file DSC.lnk has also been amended to include reference to these NLS supporting .obj files.
/st	Static linked application (Icobol). Default is shared runtime linked application (coblib).

Use **dsclink.bat** to create Windows Dschar executables.

Before linking, ensure your LIB environment variable contains the paths to **...cobol\lib**, **...ds\charmode**, and any other libraries that your application requires.

Examples

Windows: `dsclink customer+extfh+dsusrfmt`

Creates **customer.exe** for Windows that links ExtFH and Dsusrfmt for file handling and user formatting.

14.3 Using the Trace Facility in an Executable Object

You might wish to use the trace facility while you test an executable object. To do this, add code to your application to turn the trace facility on or off, using the following steps:

- Create the executable with the following modules linked in:
 - dstracer.obj**
 - dsdlgini.obj**
- Call Dsrun with the value "T" in the DS-Control field to turn tracing on.
- Call Dsrun with the value "O" in the DS-Control field to turn tracing off.

15 Printing

Dialog System enables you to obtain a listing of the entire screenset for documentation purposes. The listing is written to a text file called **screenset.lis**, where *screenset* is the name of the current screenset. If you have not yet saved the screenset, the listing is written to the file **ds.lis**.

This chapter explains:

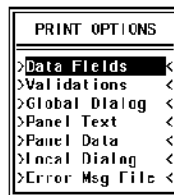
- How to write a listing of a screenset to a text file

There are some configuration defaults associated with printing screensets. You can change these, according to your requirements. To do this, edit the line sequential configuration file **dsdef.cfg**. For details about this file, see the chapter *Setting Up the Configuration File*.

15.1 Print Menu (F7)

Pressing **F7=print** from the Main menu invokes the Print menu and the Print Options popup panel shown in Figure 15-1.

Figure 15-1. Print Screen Menu



CUSTOMER Row: 02=Col: 01=Ins-Caps Scroll
 F1=help ←=print-screen-set Escape

You use the Print Options list to select the portions of the screenset you wish to print. The following options are available:

Data Fields	Details of all the data fields used in the screenset (see the chapter <i>Data Definition</i> for details). This includes field sizes, field types and field occurrences.
Validations	Details of all validations defined for all data fields used in the screenset (see the chapter <i>Data Definition</i> for details). This includes range/table validation, date validation and check digit validation.
Global Dialog	Details of dialog associated with the entire screenset (see the chapter <i>Dialog</i> for details). This includes the definition of dialog keys with their functions and associated parameters.
Panel Text	Details of all panels used in the screen set (see the chapter <i>Panels</i> for details). This includes panel size and position and text definition.
Panel Data	Details of field positioning, group properties, field usage and attributes in the panels.
Local Dialog	Details of dialog associated with individual panels in the screenset (see the chapter <i>Panel Painting</i> for details). This includes the definition of dialog keys with their functions and associated parameters.
Error Msg File	Details of error messages associated with the screenset.

The default is to print all of the screenset details. The print options that are selected for printing are enclosed by > and < symbols.

You can select or deselect any or all of the print options. To do this, use the <up-arrow> and <down-arrow> keys to position the selection bar on the desired option, then press the **Space** bar.

15.1.1 Print Screenset (Enter)

Pressing **Enter** from the Print menu prints the selected screenset items to a text file. The options you selected are displayed at the bottom of the screen while they are printed. When these messages clear, printing is complete. Press **Escape** to return to the Main menu. You can use an editor to view the printed file.

There is a demonstration screenset named CUSTOMER provided with your Dialog System software. You might wish to print this screenset and view the resultant text file in your editor. The following listing is a sample section of this file:

Micro Focus Dialog System Screenset Listing

Screenset: screenset.LIS

Screenset Creation Time: 16:42 1991/01/01

Screenset DETAILS

Number of Panels: 1
 Number of Data Fields: 20
 Error File Name:
 Screenset Version Number: 48
 Data Block Version Number: 36
 Screenset Type: Normal
 First Panel Name: full-scr
 Dialog Search: Global first
 Screenset Colorization: OFF
 Screenset Key Translation: OFF

PANEL DETAILS

Name of Panel: full-scr Panel 1 of 1
 Start-row: 01 Start-col: 01 Width: 80 Height: 25 Border: DOUBLE

15.1.2 Escape (Esc)

Pressing **Escape** from the Print menu cancels the printing process and returns you to the Main menu.

16 Character Set Support on UNIX

Dialog System on UNIX supports the use of character sets other than the standard ASCII character set. This chapter describes:

- The capabilities of DOS and UNIX screensets
- How to convert screensets to use different character sets
- How to transfer screensets from DOS to UNIX

16.1 DOS and UNIX Screensets

Dialog System supports two types of screenset; DOS screensets and UNIX screensets.

16.1.1 DOS Screensets

A DOS screenset is one that contains only data in the extended DOS character set.

All screensets you create on a DOS system are DOS screensets. Screensets you create on a UNIX system are DOS screensets unless you specify a configuration parameter in the configuration file (see the section *UNIX Screensets* later in this chapter). DOS screensets can run on UNIX systems, with the following restrictions (by default):

- Text characters with values greater than hex 7F are ignored, except for the line-drawing characters, which are converted to the nearest UNIX equivalent.
- The following ASCII control characters are not displayed:

hex 00 - hex 1F
hex 7F

If your UNIX terminal supports the extended DOS character set, you can specify the configuration parameter `TERMINAL DOS8` (see the chapter *Setting Up the Configuration File* for details). This parameter removes the above restrictions on running DOS screensets on UNIX systems.

16.1.2 UNIX Screensets

By default, screensets you create on a UNIX system are DOS screensets. However, you can set the configuration parameter `TERMINAL GENERIC8` in your configuration file, so that all screensets you create are UNIX screensets.

You can create UNIX screensets only on a UNIX system, and UNIX screensets are not portable to DOS systems.

See the chapter *Setting Up the Configuration File* for more details of these configuration parameters.

UNIX screensets support character sets containing non-English characters, such as ISO-8859 and HPRoman8 (HP LaserJet ROMAN-8).

UNIX screensets also support the use of line-drawing characters, even where the character set itself does not include line-drawing characters. All line-drawing characters in a UNIX screenset are stored internally as characters in the range hex 00 to hex 1F (the unused range of the character set). The internal representation of line-drawing characters is as follows:

01	up arrow
02	down arrow
03	right arrow
04	left arrow
05	top left corner
06	top right corner
07	bottom right corner
08	bottom left corner
09	cross lines
10	vertical line
11	horizontal line

- 12 left tee
- 13 right tee
- 14 bottom tee
- 15 top tee

When the screenset is loaded, these internal characters are converted to the line-drawing characters expected by the terminal in use, as defined in terminfo. If no line-drawing characters are defined, characters in the internal format are converted to simple character equivalents.

If a screenset has been designed and saved as a UNIX screenset, users must also have the `TERMINAL GENERIC8` configuration parameter set when they run the screenset so that non-English and line-drawing characters are handled correctly.

16.1.2.1 Screenset Naming

If you intend to transfer a screenset from UNIX to DOS, remember to name it in upper case, because DOS supports upper case only.

16.2 The Screenset Conversion Utility

You can use the utility, `SSTRAN`, to either convert between screensets using the extended DOS character set, and screensets using DOS, Windows or OS/2 character sets, or convert attributes in a screenset. You can also use the utility to convert between different DOS character sets (codepages).

To use the conversion utility, enter a command line of the form:

UNIX: `sstran -a [-y] input-screenset output-screenset
translation-file`

non-UNIX: `run sstran -a [-y] input-screenset output-screenset
translation-file`

where the parameters are:

<code>-a</code>	A flag indicating that this is an attribute conversion.
<code>-y</code>	An optional parameter that is equivalent to the user responding "yes" to each prompt produced by the utility. The only prompt currently produced asks the user to confirm that they wish to overwrite the input screenset with the output screenset; this occurs when you specify the same filename for <i>input-screenset</i> and <i>output-screenset</i> .
<i>input-screenset</i>	The filename (including path) of the screenset to be converted. This can be up to 65 characters long.
<i>output-screenset</i>	The filename (including path) of the converted screenset. This can be up to 65 characters long. The output filename before the file extension must be identical to the input filename. You can use the same, or a different file extension.
<i>translation-file</i>	The filename (including path) of the translation file that controls the conversion. See the next section for details.

Text and attributes cannot be converted at the same time: a separate run is needed for each.

16.2.1 Translation Files

A translation file is an ASCII text file that controls the operation of the SSTRAN screenset conversion utility.

A translation file can contain the following types of entry:

- Comment lines
- Directive lines
- Translation table lines

Any line beginning with "*" is treated as a comment. You can also include blank lines.

Directives indicate to SSTRAN whether the input and output screensets are DOS or UNIX. The following directives are available:

```
SOURCE DOS
SOURCE UNIX
TARGET DOS
TARGET UNIX
```

A translation file must contain one SOURCE directive and one TARGET directive.

A translation table is a series of lines that describe the character substitutions to be performed. Each line controls the setting of 16 bytes, and has the form:

```
hex-digit X"hex-string"
```

where:

<i>hex-digit</i>	specifies which 16 bytes in the character set are controlled by this line. For example, the digit B indicates that this line controls the translation of characters in the range hex B0 to hex BF.
<i>hex-string</i>	specifies the characters to which the input characters are to be converted.

For example, to convert input characters in the range hex B0 to hex BF into the uppercase alphabetic characters ABCDEFGHIJKLMNOP, you would specify the following translation line:

```
B X"4142434445464748494A4B4C4D4E4F50"
```

To leave some characters unconverted, specify hex 00 in the appropriate position in the conversion string. Using the previous example, to leave the characters B5, B6, and BF unconverted, you would specify the following translation line:

```
B X"4142434445000048494A4B4C4D4E4F00"
```

If none of the corresponding characters are to be converted, so the entire translation line would contain 00, you do not have to specify a line in the translation table for the range.

A number of text translation files are available:

Translation File	Translates...
437-unix.TRN	DOS codepage 437 to UNIX
437-8859.TRN	DOS codepage 437 to ISO 8859
850-8859.TRN	DOS codepage 850 to ISO 8859
850-HPR8.TRN	DOS codepage 850 to HPRoman8
850-unix.TRN	DOS codepage 850 to UNIX

These files contain detailed comments about translation file format.

16.2.2 Attribute Conversion

When converting attributes, you do not need to define the screenset type, so the SOURCE and TARGET directives are not required. To ensure that a text translation file is not used instead of an attribute translation file, and vice versa, we recommend that you do not include SOURCE and TARGET directives in the attribute translation file.

To leave some characters unconverted, specify hex FF in the appropriate position in the conversion string.

If none of the corresponding characters are to be converted, so the entire translation line would contain FF, you do not have to specify a line in the translation table for the range.

Two attribute translation files are available:

Translation File	Converts...
DOSC2M.TRN	Colors in a screenset to monochrome attributes supported by DOS
OLD2NEWM.TRN	Old style UNIX monochrome attributes (DS1.1) to new fully portable style

Note: Attribute conversion is not limited to UNIX. It is available under all platforms mentioned in this manual.

16.3 Transferring DOS Screensets to UNIX

Dialog System enables you to execute screensets that have been defined on DOS, on a UNIX system.

You can transfer screensets from the DOS creation environment into the appropriate UNIX directory using a binary transfer method.

16.3.1 Restrictions on Line-drawing Characters

When you transfer screensets from a DOS to a UNIX environment, UNIX imposes the following limitations on line-drawing. The limitations apply only to the display; the data actually stored is not affected.

- Double lines appear as single lines on UNIX. So, any double lines you have defined on DOS appear as single lines when you transfer the screenset to UNIX.
- Each terminfo file can include a list of line drawing characters. If this file does not contain any line drawing information, ASCII characters are displayed where lines are defined.
- There is a defined set of supported UNIX characters (see the section *Supported UNIX Characters* later in this chapter). Any DOS characters not included in this set are not supported in the UNIX environment and are converted to spaces on the screen. However, if such DOS characters are loaded onto UNIX then reloaded onto DOS, they are not affected as long as they were not manipulated while on UNIX.

Note: These restrictions do not apply if you use the `TERMINAL DOS8` configuration parameter.

16.3.2 Supported UNIX Characters

If you do not specify the `TERMINAL DOS8` configuration parameter described earlier in this chapter, the set of UNIX characters supported contains the ASCII characters in the range space to tilde (~), that is hexadecimal values 20 to 7e, and the following single-line drawing characters:

- divider
- up arrow
- down arrow
- right arrow
- left arrow
- top left of box
- top right of box
- bottom right of box
- bottom left of box
- cross lines
- vertical line
- horizontal line
- left tee
- right tee
- bottom tee
- top tee

The actual graphic characters are machine dependent. If your system does not support these characters, the equivalent ASCII characters are displayed.

16.3.3 Modifying File Names in Your Application

When you move the screensets from DOS to UNIX and create your COBOL application, you might need to modify some of the filenames in your application. This is because the filename of a screenset must match exactly the filename for that screenset as it appears in your COBOL source code. If the filenames do not match, you must modify either the file name of the screenset or your application program so that the names correspond exactly.

This is more important when you transfer screensets from DOS to UNIX, because UNIX leaves the case and capitalization of filenames exactly as they were originally entered, whereas DOS systems convert all filenames entered to upper case. When your application program calls Dsrun to request a screenset, Dialog System searches for a screenset with the exact same filename as specified in your program. In a UNIX system, if the capitalization of the screenset filename does not match exactly, Dsrun cannot find the requested screenset and an error message is returned. For example, if your application program contains the following:

```
move "DStutor" to DS-set-name
```

Dsrun searches for **DStutor.s**. It does not search for any variations in capitalization such as "DSTUTOR" or "dstutor". So, if this screenset appears as "DStutor" in your application program but has a filename of "DSTUTOR", Dsrun cannot find it.

Dialog System automatically adds an .s extension to your screenset filename, so you do not have to specify the extension yourself. If you do specify the extension, Dialog System searches for the extension in the case that you specify.

These rules on case sensitivity also apply to any error message files. These files must be in the working directory at run time.

17 Setting Up the Configuration File

There are various configuration defaults associated with the operation of Dialog System. You can change these defaults by editing the configuration file **dsdef.cfg** provided with your Dialog System software. The configuration file contains certain characters that might interfere with some editors; however, you can use the Micro Focus COBOL Editor to edit the configuration file.

This chapter explains how to configure the defaults associated with:

- Copyfile generation
- Screenset printing
- Palettes and attributes
- General run-time behavior

The first three categories of configuration parameters are relevant only to panel designers rather than end users.

17.1 The Configuration File dsdef.cfg

The configuration file, **dsdef.cfg**, is a standard ASCII text (line sequential) file that must exist in either the current directory or the directory indicated by the environment variable DSDIR . The current directory is searched first.

The configuration file contains the configuration parameters described in this chapter. A configuration parameter entry must be in upper case, and must start in column one of a line. Any other type of entry is treated as a comment.

Configuration parameters can be specified in any order.

17.2 Copyfile Defaults

The configuration file can contain parameters that perform the following options when the copyfile is generated:

- Generate control block level-88 items
- Generate Data Block level-88 items
- Generate Micro Focus or ANSI style level-78 items
- Generate DATA BLOCK INDEXED BY clause
- Maintain control block compatibility

Control Block Level-88s

The configuration parameter CONTROL BLOCK EIGHTY EIGHTS enables you to place level-88 constants in the following control block fields:

- Ds-System-Error
- Ds-Field-Change
- Ds-Exit-Field

Syntax: CONTROL BLOCK EIGHTY EIGHTS [YES|NO]

Default: NO

Remarks: This configuration parameter is provided only as a compatibility support for previous releases of Dialog System on DOS. Current releases, including the UNIX version, automatically generate these constants in the COBOL copyfile.

Data Block Level-88s

The configuration parameter DATA BLOCK FLAGS EIGHTY-EIGHTS causes level-88 items to be placed in all single-digit numeric fields in the generated copyfile. These level-88 items enable you to replace expressions by condition names that can be tested for TRUE, which improves the readability of source code.

Syntax: DATA BLOCK FLAGS EIGHTY-EIGHTS [YES|NO]

Default: YES

ANSI or Micro Focus Constants

The configuration parameter CONTROL BLOCK MF-CONSTANTS causes Micro Focus style level-78 items to be placed in the following control block fields:

- Ds-Control
- Ds-Control-Param
- Ds-Proc-Type

These level-78 items provide the definitions of constants that you can use in your program.

Syntax: CONTROL BLOCK MF-CONSTANTS [YES|NO]

Default: NO

Remarks: Level-78 items are not ANSI standard, so to use these preset values and remain in the ANSI standard, you must set the configuration parameter CONTROL BLOCK ANSI-CONSTANTS to YES. This creates a group item that contains data items matching the level-78 constant definitions.

Syntax: CONTROL BLOCK ANSI-CONSTANTS [YES|NO]

Default: YES

Remarks: If you set both of these configuration parameters to YES, the ANSI-CONSTANTS overrides the MF-CONSTANTS.

These configuration parameters are provided only as a compatibility support for previous releases of Dialog System on DOS. Current releases, including the UNIX version, provide both Micro Focus and ANSI style control blocks with the system software.

Data Block Indexed by Clause

The configuration parameter DATA BLOCK INDEXED BY CLAUSE causes an INDEXED BY clause to be appended to all the fields in a group. This enables group fields to be indexed as well as subscripted.

Syntax: DATA BLOCK INDEXED BY CLAUSE [YES|NO]

Default: NO

Dialog System Beta Version Compatibility

The configuration parameter CONTROL BLOCK COMPATIBILITY causes Ds-Global-Proc and Ds-Local-Proc fields to be included in your generated COBOL copyfile.

Syntax: CONTROL BLOCK COMPATIBILITY [YES|NO]

Default: NO

Remarks: This configuration parameter is provided only as a compatibility support for previous releases of Dialog System on DOS.

17.3 Defaults for Palettes and Attributes

The configuration file can contain parameters that control the appearance of palettes and attributes for the following palettes:

- Screenset palette
- Action bar high level definition palette

Dialog System Screenset Palette

The configuration parameter PALETTE defines the default screenset palette used in the Colorize menu to map 16 system attributes onto the attributes used in the screenset. (See the chapter *Using Dialog System* for details.) The default panel attribute roll list is derived from this screenset palette. (See the chapter *Panel Painting* for details on setting the default attributes.)

Syntax: `PALETTE = X" h0h1h2h3h4h5h6h7h8h9hahbhchdhehf"`

Parameters: `h0` thru `hf` The hexadecimal attribute values corresponding to each of the 16 screenset palette attribute entries. Whenever a new panel is defined, attributes `h1` through `h5` are taken as the default attribute roll list, and `hf` is taken as the default panel attribute. You can view the list of colors and their associated hexadecimal values in the Colorize menu.

Default: The Dialog System palette defaults are:

gray
 yellow on a red background
 brown
 yellow
 blue
 white

The configuration file contains the following sample palette setting:

```
PALETTE = X"004E070E0F060B070300000000000007".
```

This sets the following attributes:

```
gray
yellow on a red background
gray
yellow
white
brown
```

You can either use this setting as it stands, or change it to the attributes you want. This default is commented out in the configuration file. To use it, simply delete the leading asterisk from this entry.

Dialog System Action Bar High Level Definition Palette

The configuration parameter CUA-PALETTE alters the attributes used by the Action Bar high level definition facility.

Syntax: `CUA-PALETTE = X"h0h1h2h3h4h5h6"`

Parameters: `h0` through `h6` Have the following values:

<code>h0</code>	Action Bar text attribute
<code>h1</code>	Action Bar selection cursor attribute
<code>h2</code>	Action Bar entry mnemonic attribute
<code>h3</code>	Pulldown window text attribute
<code>h4</code>	Pulldown window selection bar attribute
<code>h5</code>	Pulldown window border attribute
<code>h6</code>	Action Bar pulldown delimiter character attribute.

Default: `X"304F3F304F3130"`

This sets the following attributes:

Foreground	Background
black	light blue
white	red
white	light blue
black	light blue
white	red
blue	light blue
black	light blue

You can either use this setting as it stands or change it to the attributes you want. This default is commented out in the configuration file. To use it, simply delete the leading asterisk from this entry.

Remarks:

You can view the list of colors and their associated hexadecimal values in the Colorize menu described in the chapter *Using Dialog System*.

17.4 Printing Defaults

The configuration file contains parameters that control the way in which the following features of screensets appear when you print them:

- Background character
- Page size
- ASCII values

Background Character

The configuration parameter `DEFAULT BACKGROUND CHARACTER` specifies the default background character used to fill any space left around a panel screen.

Syntax:

`-DEFAULT BACKGROUND CHARACTER char`

Parameters: *char* Any printable character.
Default: " : "

Page Size

The configuration parameter NO OF PRINT LINES ON A PAGE specifies the number of lines per page available on your printer.

Syntax: +NO OF PRINT LINES ON A PAGE *number*

Parameters: *number* Number of print lines - must be at least 34.

Default: 60

ASCII Values

The configuration file contains a series of entries, each entry describing the character to print for each ASCII code.

Syntax: ASCII VALUE: *nnn* - PRINT CHARACTER: *char1 char2*

Parameters: *nnn* ASCII code for the character.

char1 Default character corresponding to that ASCII code.

char2 Character to actually print for that ASCII code.

The configuration file contains entries for all ASCII codes (001 to 255). To specify what to print for a particular ASCII code, alter the appropriate ASCII VALUE entry in the configuration file. Simply change *char2* to the character you want to print for that code.

You must not alter the column position of either the ASCII number, which is in columns 14-16, or the new character you define, which is in column 48.

17.5 General Run-time Behavior

The configuration parameters described in this section control various aspects of run-time behavior:

- Beep when field is full
- Beep when screen is full
- Beep on invalid keystroke
- Use of large cursor
- Date format
- Decimal point character
- Appearance of empty date fields
- Cursor position in numeric fields
- Enable mouse support
- NLS folding
- Path not cleared
- Appearance of non-echoed fields
- Shadow on displayed panel
- Sign position in signed numeric fields
- Cursor position in suppressed numeric fields
- Appearance of suppressed numeric fields
- UNIX character set support

Beep When Field is Full

The configuration parameter BEEP-EOF causes the beep to be sounded when a field that does not have the autoskip property (see the chapter *Panel Fields*) is filled.

Syntax: [NO]BEEP-EOF

Default: BEEP-EOF

Beep When Screen is Full

The configuration parameter BEEP-EOS causes the beep to be sounded when the last field on the screen is completed and no automatic exit (see the chapter *Panel Fields*) is in effect.

Syntax: [NO]BEEP-EOS

Default: BEEP-EOS

Beep on Invalid Keystroke

The configuration parameter BEEP-INVALID causes the beep to be sounded when an invalid keystroke is detected.

Syntax: [NO]BEEP-INVALID

Default: BEEP-INVALID

Large Cursor

The configuration parameter CURSOR-LARGE causes a larger than normal cursor to be used at run time. This improves screen readability.

Syntax: CURSOR-LARGE

Default: A normal size cursor is used.

Date Format

The configuration parameter DATEFORM enables you to specify a date format that overrides the screenset settings. You can use this parameter to change the date format for internationalization. You can use any of the four separate dateform types.

Syntax: DATEFORM DDMM | MMDD
DATEFORM MMY Y | YYMM

DATEFORM DDMMYY | MMDDYY | YYMMDD
 DATEFORM DDMMYYYY | MMDDYYYY

Default: The format defined by the screenset.

Decimal Point

The configuration parameter DECIMALPOINT enables you to specify which character is used as the decimal point in numeric fields.

Syntax: DECIMAL-POINT " *char* "

Parameters: *char* The character to be used.

Default: "."

Cursor Position in Suppressed Decimal Fields

The configuration parameter DEC-SUPPRESS-CURSOR-RIGHT causes suppressed decimal fields to have the cursor located on the rightmost digit on entry to the field.

Syntax: [NO]DEC-SUPPRESS-CURSOR-RIGHT

Default: On entry to the field, the cursor is positioned on the decimal point character.

Empty Date Fields

The configuration parameter EMPTY-DATE-BLANK specifies that a date field with no value is displayed as:

/ /

rather than:

00/00/00

Syntax: EMPTY-DATE-BLANK

Default: Empty date fields are displayed as 00/00/00.

Cursor Position in Numeric Fields

The configuration parameter IGNORE-NUMERIC-SETCUR causes the runtime component of Dialog System to ignore the effect of SETCUR functions when the cursor moves into a numeric field. This means that the cursor always moves to the first character position.

Syntax: [NO] IGNORE-NUMERIC-SETCUR

Default: SETCUR functions operate normally.

Enable Mouse Support

The configuration parameter MOUSE-ENABLE adds mouse support. Clicking the left mouse button anywhere within the current input field causes the cursor to relocate to that position. You do not need to add new dialog code: the behavior is instantly available on existing screensets when this parameter is set.

Syntax: [NO] MOUSE-ENABLE

Default: No mouse support.

NLS Folding

The configuration parameter NLS causes alphabetic characters entered into a panel field to be folded (see the chapter *Panel Fields*). If the required NLS routines are not linked into the application, the application will abort on failing to invoke the NLS folding routines.

Syntax: [NO] NLS

Default: If no setting is included, the application will operate in NLS folding mode if the NLS support routine are linked in; otherwise, it will default to non-NLS behavior.

Path Not Cleared

The configuration parameter NOCLEAR-PATH specifies that when a QUIT is issued and there are no screensets remaining on the PUSH stack, any path previously set up in dialog is not cleared.

Syntax: [NO]CLEAR-PATH

Default: The path is cleared.

Non-Echoed Fields

The configuration parameter NOECHO-CHARACTER specifies the character to be displayed when data is entered into a field with the no echo property (see the chapter *Panel Fields*).

Syntax: NOECHO-CHARACTER "char"

Parameters: char The character to be used.

Default: Space.

Shadow on Displayed Panel

The configuration parameters SHADOWRB1 and SHADOWRB2 enable you to specify that a panel is displayed with a shadow appearance at run time. The panel must be in a screenset that uses Micro Focus Panels, rather than normal panels. You can define the character to display the shadow, and that character's attribute value, for a list of panels by defining each individual panel.

However, each shadow uses a Micro Focus Panel itself, so it reduces the number of panels available to the screenset. You must ensure that you do not exceed the maximum panel limit (see the section *Normal/Micro Focus Panels (F2)* in the chapter *Using Dialog System*). For efficiency, you should also remove any shadow definitions from **dsdef.cfg** that do not apply to the application you are running.

Syntax: SHADOWRB1 screenset panel X"xx" [character]

SHADOWRB2 screenset panel X"xx" [character]

Parameters:	<i>screenset</i>	the screenset name.
	<i>panel</i>	the panel name in the screenset.
	<i>xx</i>	the hexadecimal value specifying the attribute of the shadow character. If you specify X"00", the shadow on Windows is black.
	<i>character</i>	the character to use to display the shadow. If you omit this parameter, the space character is used.
Remarks:	SHADOWRB1 results in a shadow to the right and bottom of the panel using a character width of one. SHADOWRB2 results in a shadow to the right and bottom of the panel using a character width of two.	

Sign Position in Signed Numeric Fields

The configuration parameter SIGN-TRAILING enables you to specify that all numeric signed fields are displayed with a trailing sign.

Syntax: [NO]SIGN-TRAILING

Default: Signed numeric fields are displayed with a leading sign.

Cursor Position in Suppressed Numeric Fields

The configuration parameter SUPPRESS-CURSOR-RIGHT causes suppressed numeric fields to have the cursor located on the rightmost digit on entry to the field.

Syntax: [NO]SUPPRESS-CURSOR-RIGHT

Default: SUPPRESS-CURSOR-RIGHT

Appearance of Suppressed Numeric Fields

The configuration parameter SUPPRESS-TO-BWZ enables you to specify that the integer part of all numeric suppressed fields are blank when they are zero.

Syntax: [NO]SUPPRESS-TO-BWZ

Default: Numeric suppressed fields that are zero are displayed as a single 0 character.

UNIX Character Set Support

The configuration parameter `TERMINAL` controls character set support for screensets developed or run on UNIX systems. This configuration parameter is ignored if you are running in a non-UNIX environment.

Syntax: `TERMINAL` `DOS8` | `GENERIC8` | `GENERIC`

Parameters:

`DOS8` Makes use of the extended DOS character set if your terminal supports this character set.

`GENERIC8` Makes use of non-DOS character sets, such as ISO-8859 or HP LaserJet ROMAN-8.

`GENERIC` Specifies standard UNIX behavior; that is, characters greater than hex 7F are ignored, except for line-drawing characters, which are converted to the nearest UNIX equivalent.

Default: `TERMINAL` `GENERIC`

When a screenset is loaded at run time, the actions taken depend on the type of screenset (DOS or UNIX) and the setting of the `TERMINAL` parameter. The following table summarizes the effects of the possible combinations of the screenset type and the `TERMINAL` setting:

Screenset Type	TERMINAL Setting	Actions on Loading Screenset
DOS	GENERIC	Translate DOS line-drawing chars to UNIX equivalents. Translate characters <32 to spaces. Translate characters >127 to spaces.
DOS	GENERIC8	Translate DOS line-drawing chars to UNIX equivalents. Translate characters <32 to spaces. Translate characters >127 to spaces.
DOS	DOS8	No translation required.

Screenset Type	TERMINAL Setting	Actions on Loading Screenset
UNIX	GENERIC	Translate internal UNIX line-drawing chars to UNIX terminal characters. Translate characters <32 to spaces. Translate characters >127 to spaces.
UNIX	GENERIC8	Translate internal UNIX line-drawing characters to UNIX terminal characters.
UNIX	DOS8	Translate internal UNIX line-drawing characters to DOS.

Remarks:

The setting of the TERMINAL parameter also affects the actions taken when you create or edit a screenset, as shown by the following table:

TERMINAL Setting	Action on Creating/Editing Screenset
GENERIC	Paint using UNIX terminal draw characters.
GENERIC8	Paint using UNIX terminal draw characters.
DOS8	Paint using DOS draw characters.

The setting of the TERMINAL parameter also affects the way in which a screenset is saved, as shown in the following table:

TERMINAL Setting	Action on Saving Screenset
GENERIC	Save as a DOS screenset. Translate UNIX terminal draw characters to DOS.
GENERIC8	Save as a UNIX screenset. Translate UNIX terminal draw characters to UNIX internal draw characters.
DOS8	Save as a DOS screenset. No translation required.

See the chapter *Character Set Support on UNIX* for more details.

Keyboard Timeout

The configuration parameter `TIMEOUT` causes a regular invocation of a user-supplied program. This provides a degree of system control to carry out various functions, for example, to initiate regular backups or cause an emergency shutdown.

This timeout is active only when the cursor is in an input field; it is a keyboard timeout, and is reset whenever the user hits a key.

Syntax: `TIMEOUT user-written-program-name n...nn`

Parameters: `user-written-program-name`

Is the name of a program written by you, to be invoked at a given interval.

`n...nn` Is the time, in seconds, between each automatic invocation of the user-written program (theoretically up to 9999999 seconds). 0 is treated as invalid, and no calls are made to the user-written-program.

Remarks: The Dialog Screenset itself might use timeout in its own logic - for example, displaying a help panel if the user has not entered any data for 1 minute. The new timeout mechanism described here is designed to coexist with and not affect any other screenset timeouts.

In cases where the screenset timeout period is shorter than the period set for configuration timeout in `dsdef.cfg`, then configuration timeout may never occur. It is thus important that the user-written program is activated for all timeouts. A flag in the parameters passed to the user-written program indicates which sort of timeout triggered the entry.

Where the configuration timeout period is shorter than the screenset timeout period, then an entry will be made to the user-written program for each elapsed configuration timeout period up until the total elapsed time matches the screenset timeout period, when an entry will be made for this too.

The single parameter passed to the user written program is in the form of a parameter block:

```
01 param-block.
   03 screenset-name      pic x(8).
   03 screen-name        pic x(8).
   03 field-name         pic x(30).
```

```

03  timeout-type           pic x.
    88  screenset-timeout value "S".
    88  config-timeout   value "C".
03  action                 pic x comp-x.
    88  retc-timeout     value 1.

```

where the first three fields identify the current screenset, screen and field in the system, the next identifies whether the timeout is a screenset timeout (that is, the conventional timeout), or a configuration file timeout (that is, the new type), and the last is an action field which the user written program may set to value 1 to cause an immediate return to the application, with an error status of 17 in DS-SYSTEM-ERROR-NO.

This error status is really a notification signal to the application rather than a true error, and DSRUN may be subsequently called again. When the user-written program exits back to Dialog System without requesting an immediate return to the application, then Dialog System will take different courses of action depending on the type of timeout that occurred:

- For screenset timeout, processing will continue by executing the user defined timeout procedure, as usual.
- For configuration timeout, Dialog System will return to waiting for a keystroke, with the screen unchanged. The cursor will be positioned as before timeout, and the contents of the current field (if one exists) will also be the same as before timeout. This means that the terminal user should be unaware that any configuration timeout has occurred at all.

Dialog System determines which sort of timeout has occurred by inspecting the timeout-type field on return from the user-written program.

Thus, in general your own program should not amend this field.

18 Common Questions and Answers

This chapter provides answers to some questions that people unfamiliar with Dialog System typically ask. Each answer includes a detailed description with step-by-step instructions for solving the problem.

Also, your Dialog System software includes some screensets with sample solutions to each question. The names of these screensets are **question-number.s**, for example, **q1.s**, **q2.s**, and **q3.s**.

The subjects covered by the questions are:

- 1 Displaying a validation message
- 2 Defining a validation message panel
- 3 Defining scrolling data groups
- 4 Inserting and deleting fields in an array
- 5 Checking the size of an array
- 6 Creating a menu entry table
- 7 Creating a highlighted menu entry line
- 8 Extracting virtual text
- 9 Chaining panels

18.1 Displaying a Validation Message

Q1: How do I define how validation messages are displayed when an error is found and removed when the error is corrected?

A: The display and removal of validation messages is completely automatic once you make the relevant definitions. Use the following steps:

- 1 Define the data fields that you require for your screenset (see the chapter *Data Definition* for details).
- 2 Define an additional data field called ERR-MSG as alphanumeric and length 40.
- 3 Define this field to be the error message field (see the chapter *Data Definition* for details of error message field definition).
- 4 Define the error messages that you are going to use.
- 5 Define validations for the required fields (see the chapter *Data Definition* for details of validation definition).
- 6 Associate an error message with each of these validations (see the chapter *Data Definition* for validation error message definition).
- 7 Define a suitable panel (see the chapter *Panels* for details).
- 8 Place the data fields for which you have defined validations on this panel and ensure that their usage is set to INPUT (see the chapter *Panel Painting* for details on positioning data fields on panels).
- 9 Place the ERR-MSG field on this panel and ensure that its usage is set to OUTPUT.
- 10 Define the following global dialog: the <right-arrow>| key causes a skip to next field; and the |<left-arrow> key causes a skip to the previous field (see the chapter *Dialog* for details). You might find this dialog table helpful:

Key	Function	Parameters
TAB	SKNF	0001
BTAB	SKPF	0001

This screenset is now ready to run (see the chapter *Running the Screenset* for details on running screensets). You can compare your screenset with the sample provided in `q1.s`.

18.2 Defining a Validation Message Panel

Q2: How do I define just one panel in which all of my error messages can appear?

A: This is a common requirement. The previous example explained how to place the error message field on the relevant panel, which you could do for every panel. However, it is also quite straightforward to create a separate panel. This example continues on from the example in Question One. Use the following steps:

- 1 Define a separate panel to display error messages, and place the ERR-MSG output field, created in the last example, on it.
- 2 Define the following local dialog for this error message panel: Any keystroke pops the panel; the keystroke is repeated on the panel popped to; but the Escape key is not repeated after the POP (otherwise it causes a return to the calling program). See the chapter *Dialog* for details on defining dialog. You might find the following dialog table helpful:

Key	Function	Parameters
ESC	POP	
ANYO	RPKY	
	POP	

- 3 Define the following local dialog for the panel containing the data fields for which you have defined validations: a validation error pushes the current panel; the cursor is turned on (remember, the cursor is automatically turned off in panels that contain no input fields); and control goes to the error panel.

The dialog table for this "error procedure" looks like this:

Key	Function	Parameters
ERR	PUSH	NULL
	CON	
	GOP	ERROR

This screenset is now ready to run (see the chapter *Running the Screenset* for details of running screensets). You can compare your screenset with the sample in **q2.s**.

18.3 Defining Scrolling Data Groups

Q3: How do I define scrolling for an array of data fields?

A: This is also a common requirement. Dialog System can have up to 30 separate scrolling groups on each panel. For simplicity, the example shows how to define just one.

- 1 Define five data fields as follows (see the chapter *Data Definition* for details on defining data):
 - Name FIELD1, Format X, Size 20
 - Name FIELD2, Format X, Size 10
 - Name FIELD3, Format X, Size 3
 - Name FIELD4, Format 9, Size 4.2
 - Name FIELD5, Format 9, Size 4.2
- 2 Define FIELD1 as a data group with 50 repeats (see the chapter *Data Definition* for details on defining data groups).
- 3 Include the remaining four fields in the group to create a group of data items that repeat 50 times in the Data Block. Now you must define how this group is used on the panel.
- 4 Define a panel wide enough to contain all the fields side by side, that is, a panel of approximately 60 characters (see the chapter *Panels* for details).
- 5 Position the fields side by side along one row of the panel (see the chapter *Panel Fields* for details).

The fields are now on the panel, but there is only one occurrence of each. If you run the screenset at this stage, you can enter data only on the first of each of the 50 occurrences in each field.

- 6 Define your row of fields as a panel group and call it "DATA-GRP" (see the chapter *Panel Groups* for details).
- 7 Extend the size of the group to the number of occurrences you wish to be visible on the screen at one time. Remember that the final array will scroll, so you do not need to have all 50 occurrences visible at once.
- 8 Select the group-accept-exit-bar option from the Data Group menu. This forces the cursor to move off the group after the last field in the last occurrence of the group, rather than after the last field in the current occurrence. It also automatically moves the selection bar.
- 9 Define a selection bar on this group (see the chapter *Panel Groups* for details on defining selection bars).
- 10 Define the following local dialog for this panel: <up-arrow> moves the selection bar up by one line; <down-arrow> moves the selection bar down by one line; <right-arrow>| causes a skip to next field; and |<left-arrow> causes a skip to the previous field (see the chapter *Dialog* for details of defining dialog). You might find this dialog table helpful:

Key	Function	Parameters
CURU	PBUP	DATA-GRP 0001
CURD	PBDN	DATA-GRP 0001
TAB	SKNF	0001
BTAB	SKPF	0001

This screenset is now ready to run (see the chapter *Running the Screenset* for details on running screensets). Scrolling at the top and bottom of the list is completely automatic. You can compare your screenset with the sample in **q3.s**.

18.4 Inserting and Deleting Fields in an Array

Q4: How do I define inserting and deleting in an array of data fields?

A: You can do this easily using the following steps, which continue on from the previous example.

- 1 Follow the steps in Question three.
- 2 Define the following additional local dialog for your panel group named DATA-GRP: F3 enables a line to be inserted at the current bar position; and F4 enables a line to be deleted at the current bar position (see the chapter *Dialog* for details on defining dialog). Your new dialog table looks like this:

Key	Function	Parameters
F3	IBP	DATA-GRP
F4	DBP	DATA-GRP
CURU	PBUP	DATA-GRP 0001
CURD	PBDN	DATA-GRP 0001
TAB	SKNF	0001
BTAB	SKPF	0001

This screenset is ready to run (see the chapter *Running the Screenset* for details on running screensets). You can compare your screenset with the sample in **q4.s**.

18.5 Checking the Size of an Array

Q5: How can I keep track of how many items there are in an array after inserting and deleting?

A: You can both control and monitor this array size. Once you have set the array size, the Dialog System Run-time System prevents you from moving above the first occurrence or below the final occurrence. For

example, the previous example uses an underlying array size of 50. Every use of the DBP function reduces the internal array size by one, down to a limit of one. Every use of the IBP function increases the array size by one, up to a limit of 50. However, if you set the array size to five, the IBP function applies only up to a limit of five.

The following steps explain how you can monitor the array size, using the previous example screenset as a starting point.

- 1 Define a new data field called ARR-SIZE, which is format 9 and size 2 (see the chapter *Data Definition* for details on defining data fields). This field will contain the current array limit.
- 2 Place the array size field on the panel on which you have placed your panel group, as an OUTPUT field, so that you can see what is happening to it (see the chapter *Panel Fields* for details).
- 3 Define the following additional global dialog for your panel group named DATA-GRP: initialize the screenset, placing a value of two in the data item ARR-SIZE (to ensure that there are only two entries accessible in the panel group at the start). The dialog table looks like this:

Key	Function	Parameters
P000	MOVE	0002 ARR-SIZE

- 4 Define the following additional local dialog for this panel: set the internal array size to the value in the ARR-SIZE field; move the current internal array size of DATA-GRP to the ARR-SIZE field when an item is inserted; refresh the panel so that the revised value of ARR-SIZE can be viewed; move the current internal array size of DATA-GRP to the ARR-SIZE field when an item is deleted; reset the value of ARR-SIZE to two if its value becomes less than two; and refresh the panel. The amended dialog table looks like this:

Key	Function	Parameters
F3	IBP	DATA-GRP
	MAS	DATA-GRP ARR-SIZE
	RFTD	
F4	DBP	DATA-GRP
	MAS	DATA-GRP

Key	Function	Parameters
		ARR-SIZE
	IF	ARR-SIZE
		0002
		P001
	RFTD	
CURU	PBUP	DATA-GRP
		0001
CURD	PBDN	DATA-GRP
		0001
TAB	SKNF	0001
BTAB	SKPF	0001
P000	SAS	DATA-GRP
		ARR-SIZE
P001	MOVE	0002
		ARR-SIZE
	XP	P000

This screenset is ready to run (see the chapter *Running the Screenset* for details on running screensets). Watch the behavior of the group and the value in ARR-SIZE as it runs. You can compare your screenset with the sample in **q5.s**.

18.6 Creating a Menu Entry Table

Q6: How do I define a list of menu options in the middle of the screen, from which I can select an option either by moving up and down the list and pressing **Enter** or by typing a key letter?

A: This is a common requirement that makes your application much more useful. Use the following steps:

- 1 Define a data field called "OPT-NO" with format 9 and size one (see the chapter *Data Definition* for details on defining data fields).
- 2 Define a panel and place five menu text items, such as Load, Save, Clear, Delete and Exit, in the middle of it (see the chapter *Panel*

Fields for details). These are your menu entries, which can be selected either by using the selection bar or typing in the appropriate key letter such as L, S, C, D, X.

- 3 Define four additional panels to go to when a menu entry is selected. Name them LOAD, SAVE, CLEAR, and DELETE, respectively (you do not need a separate panel for Exit, because you will use that to terminate the current screenset rather than to go to another panel). Don't place anything in these panels at the moment.
- 4 Define the following local dialog for each of these panels: any keystroke pops the panel (see the chapter *Dialog* for details on defining dialog). The dialog table looks like this:

Key	Function	Parameters
ANYO	POP	

- 5 Define your five menu entries as a fixed text group named MENU (see the chapter *Panel Groups* for details).
- 6 Define a suitable selection bar on this group (see the chapter *Panel Groups* for details on defining selection bars).
- 7 Define the following local dialog for this panel: **Enter** branches to the procedure to load a menu panel; <up-arrow> and <down-arrow> move the selection bar one line up and down, respectively, and set it on one of the menu options; "C" sets the selection bar on menu option three and goes to the CLEAR menu panel; "D" sets the selection bar on menu option four and goes to the DELETE menu panel; "L" sets the selection bar on menu option one and goes to the LOAD menu panel; "S" sets the selection bar on menu option two and goes to the SAVE menu panel; and "X" sets the selection bar on menu option five and returns to the calling program.

The dialog table looks like this:

Key	Function	Parameters
CR	BPD	MENU P011
CURU	PBUP	MENU 0001
CURD	PBDN	MENU 0001

Key	Function	Parameters
C	MOVE	0003
		OPT-NO
	XP	P001
D	XP	P013
	MOVE	0004
		OPT-NO
L	XP	P001
	XP	P014
	MOVE	0001
S		OPT-NO
	XP	P001
	XP	P011
X	MOVE	0002
		OPT-NO
	XP	P001
P000	XP	P012
	MOVE	0005
		OPT-NO
P001	XP	P001
	XP	P015
	MOVE	0000
P011		OPT-NO
	SB	MENU
		OPT-NO
P012	PUSH	\$NULL
	GOP	LOAD
	PUSH	\$NULL
P013	GOP	SAVE
	PUSH	\$NULL
	GOP	CLEAR
P.014	PUSH	\$NULL
	GOP	DELETE
	RETC	

You have defined the characters C, D, L, S and X in upper case but not in lower case. This is because by default Dialog System recognizes these as the same character. However, if you require case sensitivity, you can turn it on. See the section *Dialog Definition Menu* in the chapter *Dialog* for details.

The procedures defined for the lines in a text group must occur in the same order as the lines in the group.

This screenset is ready to run (see the chapter *Running the Screenset* for details on running screensets). You can compare your screenset with the sample in **q6.s**.

18.7 Creating a Highlighted Menu Entry Line

- Q7:** How do I define a menu line at the top of the screen from which I can select an option either by moving along the list and pressing, or by typing a key letter?
- A:** This uses the same principles as the previous question. The differences are that instead of using a fixed text group for the menu and moving a selection bar up and down it, you use a one-line virtual attribute group and press the <left-arrow> and <right-arrow> keys to position the virtual attribute data up and down. Use the following steps:
- 1 Define a data field called "OPT-NO" with format 9 and size one, the same as in the previous example (see the chapter *Data Definition* for details on defining data fields).
 - 2 Define a one-line panel and place the five menu text items, Load, Save, Clear, Delete and Exit, across it (see the chapter *Panel Fields* for details).
 - 3 Define the LOAD, SAVE, CLEAR and DELETE panels to pass control to when the relevant menu entry is selected. Do not place anything in these panels at the moment.

- 4 Define the following local dialog for each of these panels: any keystroke POPs the panel (see the chapter *Dialog* for details on defining dialog). The dialog table looks like this:

Key	Function	Parameters
ANYO	POP	

- 5 Define your five menu entries as a virtual attribute group named MENU and define the set of attributes that will move over the text items on your panel (see the chapter *Panel Groups* for details).
- 6 Define the following local dialog for this panel: **Enter** branches to the procedure to load a menu panel; <left-arrow> and <right-arrow> position the data one menu item up and down, respectively; "C" sets the data on menu option three and goes to the CLEAR menu panel; "D" sets the data on menu option four and goes to the DELETE menu panel; "L" sets the data on menu option one and goes to the LOAD menu panel; "S" sets the data on menu option two and goes to the SAVE menu panel; and "X" sets the data on menu option five and returns to the calling program.

The dialog table for this menu panel looks like this:

Key	Function	Parameters
CR	BPD	MENU P011
CURL	PDUP	MENU 0001
CURR	PDDN	MENU 0001
C	MOVE	0003 OPT-NO XP P001 XP P013
D	MOVE	0004 OPT-NO XP P001 XP P014
L	MOVE	0001 OPT-NO XP P001

Key	Function	Parameters
	XP	P011
S	MOVE	0002
		OPT-NO
	XP	P001
	XP	P012
X	MOVE	0005
		OPT-NO
	XP	P001
	XP	P015
P000	MOVE	0001
		OPT-NO
P001	SD	MENU
		OPT-NO
P011	PUSH	\$NULL
	GOP	LOAD
P012	PUSH	\$NULL
	GOP	SAVE
P013	PUSH	\$NULL
	GOP	CLEAR
P014	PUSH	\$NULL
	GOP	DELETE
P015	RETC	

This screenset is ready to run (see the chapter *Running the Screenset* for details on running screensets). You can compare your screenset with the sample in **q7.s**.

18.8 Extracting Virtual Text

Q8: Can I extract the contents of a virtual text item into a data field?

A: Yes. Use the following steps:

1 Define two data fields:

Name POSITION, Format 9, Size 2

Name TEXT, Format X, Size 30

The field "TEXT" is the field to move the required virtual text to, and the field "POSITION" is the field to use to monitor this move on the current screen.

2 Define a panel that contains these two fields as output fields.

3 Define a virtual text group called "TEXT-GRP" and define about two to three times as much virtual text as can fit on the visible group area.

4 Define a selection bar on this virtual text.

5 Define the following local dialog for this panel: carriage return moves the position of the selection bar on the TEXT-GRP group to the POSITION field, transfers the virtual text to the TEXT field at the offset defined by POSITION, then refreshes the information on the panel; <up-arrow> and <down-arrow> position the selection bar one line up and down, respectively, on the virtual text. The dialog table looks like this:

Key	Function	Parameters
CR	MBD	TEXT-GRP POSITION
	MOVEVTEXT	TEXT-GRP POSITION TEXT
	RFTD	
CURU	PBUP	TEXT-GRP 0001
CURD	PBDN	TEXT-GRP 0001

This screenset is ready to run (see the chapter *Running the Screenset* for details on running screensets). You can compare your screenset with the sample in **q8.s**.

18.9 Chaining Panels

Q9: How can I "chain" many panels so that <right-arrow>| in the last field in a panel moves to next panel and |<left-arrow> in the first field in a panel moves to the previous panel?

A: This is another common requirement. The solution uses the ability to detect an exception condition when a SKNF function attempts to execute but there are no more fields on the panel, and the similar but opposite situation for SKPF. Use the following steps:

- 1 Define a number of data fields.
- 2 Paint a number of panels, for instance AAAAAAAAA, BBBB BBBB, CCCCCCCC, DDDDDDDD and EEEEEEEE.
- 3 Place the data fields you defined onto these panels.
- 4 Define the following local dialog for the first panel, AAAAAAAAA: when SKNF causes an exception, it performs a PUSH and Goes to Panel BBBB BBBB. The dialog table looks like this:

Key	Function	Parameters
TAB	BPE	P001
	SKNF	0001
BTAB	SKPF	0001
P001	PUSH	\$NULL
	GOP	BBBBBBB

- 5 Define the following local dialog for the second panel, BBBB BBBB: when SKNF causes an exception, it performs a PUSH and Goes to Panel CCCCCCCC, and when SKPF causes an exception, it performs a POP. The dialog table looks like this:

Key	Function	Parameters
TAB	BPE	P001
	SKNF	0001
BTAB	BPE	P002
	SKPF	0001
P001	PUSH	\$NULL
	GOP	CCCCCCCC
P002	POP	

- 6 Define the local dialog for the remaining three panels in the same way, with the additional dialog for the final panel to skip to the first panel on exception. Thus, the dialog table for the last panel looks like this:

Key	Function	Parameters
TAB	SKNF	0001
BTAB	BPE	P002
	SKPF	0001
P002	POP	

You can chain a number of panels in this way. However, the maximum number of panels you can PUSH is 16.

This screenset is ready to run (see the chapter *Running the Screenset* for details on running screensets). You can compare your screenset with the sample in **q9.s**.

19 Key Code List

This chapter provides a list of the key codes used in the dialog definition process (see the chapter *Dialog* for details). There are five kinds of keys, each of which has associated mnemonics.

19.1 ASCII Keys

The ASCII keys all use a mnemonic that is the key value itself. For example:

A is the character "A"
5 is the character "5"
z is the character "z"
/ is the character "/"

The space character is the only exception. This character is represented by the mnemonic:

SPC

19.2 Function Keys

These mnemonics represent the function of the key; however, for the specific keystrokes to achieve each key's function, please refer to your Release Notes.

UNIX: For Ctrl or Alt keystroke combinations on UNIX, first enable Ctrl or Alt (press "/c" or "/a", respectively), then press the key or sequence

required for the second part of the combination. Remember to disable Ctrl or Alt afterward.

Mnemonic	Function
ESC	ESCape
F1 - F12	Function keys 1 - 12
CR	Carriage Return
DEL	DElete
CURL	CURsor Left
CURR	CURsor Right
CURU	CURsor Up
CURD	CURsor Down
HOME	HOME key
TAB	TAB key
BTAB	BackTAB key
BS	Back Space key
END	End key
PGUP	PaGe UP
PGDN	PaGe DowN
SF1 - SF12	Shift+Function keys 1 - 12
CF1 - CF12	Ctrl+Function keys 1 - 12
AF1 - AF12	Alt+Function keys 1 - 12
CTHM	CTrl+HoMe key
CTEN	CTrl+ENd key
CTPU	CTrl+Page Up
CTPD	CTrl+Page Down
CTLF	CTrl+cursor LeFt
CTRT	CTrl+cursor RighT
AT0 - AT9	Alt keys 0 - 9
AT-	Alt+minus key
AT=	Alt+equals key
ATA - ATZ	Alt+A - Alt+Z keys
CTA - CTG	Ctrl+A - Ctrl+G keys
CTJ - CTL	Ctrl+J - Ctrl+L keys

Mnemonic	Function
CTN - CTZ	Ctrl+N - Ctrl+Z keys
MSL	Left mouse button
MSR	Right mouse button

19.3 Status Keys

The mnemonics reflect the fact that status keys are recognized only at the moment they are turned on and off. On UNIX, they reflect the logical pressing and releasing of these keys rather than the physical keystrokes.

Mnemonic	Status
ALT1	ALT on
ALT0	ALT off
CTL1	CTrL on
CTL0	CTrL off
LSH1	Left SHift key on
LSH0	Left SHift key off
RSH1	Right SHift key on
RSH0	Right SHift key off
INS1	INSert key on
INS0	INSert key off
CAP1	CAPs lock on
CAP0	CAPs lock off
NUM1	NUM lock on (Windows only)
NUM0	NUM lock off (Windows only)
SCL1	SCroLI lock on
SCL0	SCroLI lock off

19.4 Any Other Key

The mnemonic for any key other than those previously listed in the dialog is:

ANYO ANY Other key

19.5 Error Key

This is not actually a key, but a type of procedure. The mnemonic:

ERR

indicates that the specified actions are invoked whenever an error occurs in the validation routines. This mnemonic, which appears in the key position of the dialog, is typically used to go to an error message panel. (See the chapter *Dialog* for details about this key.)

19.6 Mouse Support

Mouse support is available via two dialog key mnemonics, MSL and MSR.

The mouse is visible only when Dialog System is waiting for user keyboard input.

The left and right mouse buttons have dialog key mnemonics MSL and MSR respectively. These represent a single click. They can be used in the same way as other keys to drive dialog functions.

Since in general it is required that the action triggered by a mouse click be dependent on the mouse position, the position of the mouse at the time of the click is recorded in the Dialog System control block.

The X and Y coordinates are character positions, with the position 1,1 being top left of the screen. Note that the coordinates are relative to the screen and not to the panel.

The last two fields give the field number and its occurrence if the mouse was positioned within an input field when the mouse was clicked, otherwise zero valued.

The MSL procedure is not triggered when the cursor is already located in the same field as the mouse when the left mouse button is clicked. The cursor simply relocates itself to the mouse location as described above.

There are thus two methods of taking action on a mouse click. These are either by returning to the application (RETC) so the mouse position can be established and the appropriate action taken, or by using CALLOUT to a user program, which is passed the control block and hence the mouse information, and also the data block which it can modify (or in addition/alternatively the user program can directly set \REG or the dialog exception condition.

20 Function Code List

This chapter provides detailed descriptions of the functions that you can include in a global or local dialog (see the chapter *Dialog* for details).

20.1 Function Parameters

Each function can have up to three parameters, which can be one of the following types:

attr	<p>One of the following run-time attribute names:</p> <p>AUX1 AUX2 AUX3 AUX4 ERROR INPUT NATIVE</p> <p>These enable you to associate foreground/background colors with panel fields when the screenset runs.</p>
group	The name of a group field.
proc	The name of a dialog procedure, <i>Pnnn</i> , where <i>nnn</i> is a procedure number in the range 000 to 255. In functions that require a procedure parameter, you can supply the procedure number in a field or register.
numeric	A numeric literal.
string	An alphanumeric literal surrounded by quotes ("). If the first character of a parameter is a quote ("), it is assumed to be an alphanumeric literal. You can enter a literal of up to 80 characters by scrolling the field horizontally.

field	The name of a data field. If the field is a group field, the field name can have a subscript that identifies a particular occurrence of the group: <i>field(subscript)</i> <i>subscript</i> can be a numeric literal, a register, or the name of another field (which must not be group field).
panel	The name of a panel in a screenset.
register	Dialog System maintains a number of internal registers. All registers are two-byte computational fields. The following register is provided: \$REG A general purpose register. Functions can read from \$REG or write to it.
null	You use the value \$NULL to indicate that you are not supplying a parameter value for an optional parameter (for example, the optional procedure name parameter of the PUSH function).

20.2 Function Descriptions

This section contains a description of each function and its parameters.

Where a parameter is described as *field(sub)*, the parameter can be a subscripted or non-subscripted field name.

BEEP

BEEP sounds the beep on the machine.

Syntax: BEEP

Parameters: None

BP

BP branches to a procedure without executing the remaining functions in the dialog line.

Syntax: `BP procedure`

Parameters: `procedure` `proc, field(sub), or register.`
The procedure to be branched to.

Examples: `BP P063`

Branches to procedure 063.

```
MOVE 75 $REG
BP $REG
```

Branches to procedure 075.

BPD

BPD branches to a procedure without executing the remaining functions in the dialog line.

The destination procedure number is $n+m-1$, where:

- n is the value of parameter 2.
- m is the position of the selection bar in the group specified by parameter 1. If the group does not have a selection bar (as is the case with virtual attribute groups), m is the position of the first visible line in the data.

Syntax: `BPD group procedure`

Parameters: `group` The name of the group on which to act.
`procedure` `proc, field(sub), or register.`
The procedure to branch to if the selection bar in `group` is at position one.

Example: `BPD MYGROUP 65`

If the selection bar in MYGROUP is in position 1, control branches to procedure 065; if it is in position 4, control branches to procedure 068.

BPE

BPE branches to a procedure only if the next function in the dialog line causes an exception condition. The remaining functions in the dialog line are not executed.

Syntax: `BPE procedure`

Parameters: `procedure` proc or field(*sub*).

The exception procedure to be branched to.

Remarks: A function that can cause an exception has an entry in its description describing the conditions under which it can cause an exception.

Example: `BPE P045`
`POP`

Branches to procedure 045 only if the POP function causes an exception by attempting to pop an empty stack.

BPR

BPR branches to a procedure without executing the remaining functions in the dialog line.

The destination procedure number is $n+m-1$, where:

- n is the value of parameter 1.
- m is the value in the specified register.

Syntax: `BPR procedure`

Parameters: `procedure` proc, field(*sub*), or register.

The procedure to which control branches if the value in the register is 1.

Example: `MOVE 4 $REG`
`BPR 65`

Branches to procedure 068.

CALLOUT

Use CALLOUT to make a synchronous call to an external program, so that you can call a user program directly from dialog.

Syntax: `CALLOUT program-name callout-type callout-parameter`

Parameters:	<i>program-name</i>	string The name of the program being called.
	<i>callout-type</i>	numeric Must be 0000 or 0006. When the value 0006 is specified, the content of RETURN-CODE on exiting the callout program is moved into \$REG, unless the value is -9999, when \$REG is unaffected, but the dialog exception condition is set. RETURN-CODE is always cleared to zero.
	<i>callout-parameter</i>	Must be \$NULL. Specifies that the whole of the Data Block is passed.

Remarks: The normal search paths for COBOL programs are used.

The first parameter passed by Dialog System to the called program is the Ds-Control-Block, described in either **ds-ctrl.mf** or **ds-ctrl.ans** in the chapter *Running the Screenset*. You cannot make any changes to this control block; any changes you attempt will not be reflected in Dialog System's copy of the control block.

The second parameter passed by Dialog System to the called program is the *screenset-Data-Block*. The callout program can modify the Data Block as desired. You can use the screenset copybook that is generated by Dialog System to define the Data Block in the main application, as the data definition of the Data Block in the callout program.

The CALLOUT function has the following restrictions:

- Do not call any Dialog System run-time modules from the CALLOUT program or make recursive calls. Unpredictable results can occur.
- Because of system limits, the maximum length of the name of the called module, including its path, is 82 bytes.

Exception: The function causes an exception condition (see BPE) if the program specified is not found.

Example: `CALLOUT "\usr\abc\user-program" 0000 $NULL`

Executes the program user-program.

CEOF

CEOF clears a field from the current cursor position to the end of the field and fills it with spaces. The field must be alphanumeric or alphabetic, otherwise this function has no effect.

Syntax: `CEOF`

Parameters: None

CFLD

CFLD clears all of an input field. The field is filled with spaces if it is an alphanumeric or alphabetic field, or zeros if it is a numeric field.

Syntax: `CFLD field`

Parameters: *field* `field(sub)`.
The name of the field. If this parameter is NULL, the current field is cleared.

CLEAR

CLEAR clears all of the fields on the current panel.

Syntax: `CLEAR`

Parameters: None

CLRF

CLRF sets the value of a field to false (0).

Syntax: `CLRF field`

Parameters: `field` `field(sub)`.
The name of the field.

Example: `CLRF MYGROUP(10)`
Sets the tenth field in group MYGROUP to zero.

COFF

COFF turns the cursor state to off. The cursor is invisible if the current panel has no input fields.

Syntax: `COFF`

Parameters: None

Examples: This function is system dependent, so it might not work on your system.

CON

CON turns the cursor state to on. The cursor is present even if there are no input fields on the current panel.

Syntax: `CON`

Parameters: None

Examples: This function is system dependent, so it might not work on your system.

DBP

DBP deletes the line at the current selection bar position.

Syntax: `DBP group`

- Parameters:** *group* The name of the group on which to act.
- Remarks:** The size of the group's internal array is decremented by one. Data in lines beyond the deleted line is shifted down one position.
- When the internal array size is one, the DBP function does not work. The first occurrence is cleared, but the array size remains at one, because you cannot have a group that contains zero occurrences.
- Example:** The group MYGROUP contains the following values (the values between the lines represent the fields visible on the panel, and the asterisks represent the position of the selection bar):

```

12
20
-----
35
4*****
65
-----
19
25

```

The function:

```
DBP MYGROUP
```

alters MYGROUP as follows:

```

12
20
-----
35
65*****
19
-----
25

```

DECVAL

DECVAL decrements a numeric field or register by one.

Syntax: DECVAL *item*

Parameters: *item* field(*sub*) or register (\$REG only).
The name of the item to be decremented.

Examples:
DECVAL \$REG
DECVAL QUANTITY(3)

The first example decrements the value of \$REG; the second example decrements the third occurrence in group field QUANTITY.

GOF

GOF transfers control to a specified field in the current panel. If the field is not on the current panel, each panel in the screenset is searched until a panel containing the specified field is found.

Syntax: GOF *field*

Parameters: *field* field(*sub*) or register.
The name of the field to which control is transferred.

Remarks: You can pass a field number in a register. Remember that this is the number of the field in the Data Block, rather than the number of the field in the panel.

Exception: The function causes an exception when the specified field cannot be found in any panel in the screenset.

Examples:
GOF DESCR
GOF PRICE(\$REG)
GOF \$REG

The first example transfers control to the field named DESCR. The second example transfers control to an occurrence in the group field PRICE; the occurrence is determined by the current value of \$REG. The final example transfers control to field number *n*, where *n* is the value of \$REG; note that this is the *n*th field in the Data Block, not the *n*th field in the panel.

GOMOUSE

GOMOUSE enables mouse support, without you having to return to the application or execute a CALLOUT.

Syntax: `GOMOUSE number`

Parameters: `number` numeric.

A number that represents the following settings:

- +1 Will support relocation of the cursor to the start of a different input field (which is not in a group) after validating the current input field. If validation fails, no cursor movement occurs.
- +2 Will support relocation of the cursor to the start of a different input field (which is in a group and at the current action bar position) after validating the current input field. If validation fails, no cursor movement occurs.
- +4 Will support relocation of the cursor to the start of a different input field (which is in a group), relocating the action bar if necessary, after validating the cursor input field. If validation fails, no cursor movement occurs.

Remarks: These settings exhibit cumulative behavior. All relocations are to the position of the mouse cursor when a mouse button was last clicked.

Example: `MSL GOMOUSE 0007`

Enables all of these actions for a panel.

GOP

GOP passes control to a specified panel.

Syntax: `GOP panel`

Parameters: `panel` The name of the panel to which control passes.

Remarks: Because this function passes control to another panel, you cannot define any subsequent entries in that dialog line.

The target panel must exist before you can use this function in dialog.

Examples: GOP NEXTPNL

Passes control to the panel named NEXTPNL.

IBP

IBP inserts an empty line into a group at the position of the selection bar.

Syntax: IBP *group*

Parameters: *group* The name of the group on which to act.

Remarks: The group's internal array size is incremented by one, subject to the maximum size of the Data Block. Data in lines after the inserted line is moved up by one position. The empty line contains zeros in numeric fields, and spaces elsewhere.

Exception: The function causes an exception when it attempts to insert a line into a group that is already full, because this would cause loss of data from the last item in the data group.

Example: The field MYGROUP contains the following values (values between the lines are those values that are visible in the panel, and the asterisks show the position of the selection bar):

```

24
-----
28
36*****
23
-----
2
30

```

The function:

```
IBP MYGROUP
```

alters MYGROUP as follows:

```

24
-----
28
*****
36
-----
23
2
30

```

IF

IF tests an expression for validity, and if true, branches to a specified procedure where processing continues. Control does not return to the calling procedure. If the expression is false, execution continues at the line below the IF function.

The following IF functions are available:

IF <, IF <=, IF =, IF >, IF >=, IFNOT=

Syntax:

IF-statement expression1 expression2 procedure

Parameters:

<i>IF-statement</i>	One of the above IF statements.
<i>expression1</i>	numeric, string, field(<i>sub</i>), or register. The first operand of the test.
<i>expression2</i>	numeric, string, field(<i>sub</i>), or register. The second operand of the test.
<i>procedure</i>	proc, field(<i>sub</i>), or register. The procedure to branch to if the test is true.

Remarks:

Expression1 and *expression2* must be compatible types. For example, an expression cannot test a numeric literal against an alphanumeric literal. A similar set of conditional functions are provided that perform the target procedure and return control to the calling procedure. See the XIF function later in this chapter.

Examples:

```
IF= QUANTITY 3 P045
```

Branches to procedure 045 if the field QUANTITY has the value 3.

```
IF> MYGROUP(2) $REG TARGET
```

Branches to the procedure whose number is stored in field TARGET if the second occurrence in group field MYGROUP is greater than the value in the register.

INCVAl

INCVAl increments a numeric field or register by one.

Syntax: `INCVAl item`

Parameters: `item` field(*sub*) or register (\$REG only).
The item to be incremented.

Examples: `INCVAl $REG`
`INCVAl STEP`

The first example increments the register; the second example increments the field STEP.

KS

KS scans the keyboard, checks the status of all the status keys, and forces any status key dialog to execute.

Syntax: `KS number`

Parameters: `number` numeric, field(*sub*), or register.
A number that represents the following bit settings:

- bit 7 set - check Ins status
- bit 6 set - check Caps Lock status
- bit 5 set - check Num Lock status (DOS)
- bit 4 set - check Scroll Lock status
- bit 3 set - check Alt key pressed status
- bit 2 set - check Ctrl key pressed status
- bit 1 set - check Left Shift pressed status
- bit 0 set - check Right Shift pressed status

Remarks: The Ins, Caps, Num (DOS only), and Scroll Lock status keys are tested for their toggle status. For example, the status of the Ins key after it is pressed five times is the same as after it is pressed just once.

However, the Alt, Ctrl, Left Shift and Right Shift keys are tested to determine whether or not they are currently being held down.

If your keyboard does not have the status keys described in this section, please consult your Release Notes to see which keys on the standard keyboard for your system provide the equivalent functions.

Example: KS 55H

Checks the status of the Caps, Scroll Lock, Ctrl and Right Shift keys and activates any dialog associated with them.

MAS

MAS moves the defined internal array size of a group to a specified field or to the register.

Syntax: MAS *group destination*

Parameters:

<i>group</i>	The name of the group on which to act.
<i>destination</i>	field(<i>sub</i>) or register.
	The location where the array size is stored.

Example: MAS MYGROUP \$REG

Moves the internal array size of group MYGROUP to the register.

MB

MB returns the value of the item at the selection bar in a group to a specified field or register.

Syntax: MB *group destination*

Parameters:

<i>group</i>	The name of the group on which to act.
--------------	--

destination field(*sub*) or register (\$REG only).
The location where the value of the item at the selection bar position is stored.

Example: The group MYGROUP contains the following values (values between the lines are visible in the panel, and asterisks mark the position of the selection bar):

```
SMITH
JONES
-----
ADAMS
KNIGHT*****

SNOW
STEWART
-----
MADISON
```

The function:

```
MB MYGROUP NAME
```

stores the value "KNIGHT" in field NAME.

MBD

MBD returns the data position where the selection bar is currently located to a specified field or register.

Syntax: MBD *group destination*

Parameters: *group* The name of the group on which to act.

destination field(*sub*) or register (\$REG only).

The location where the bar position is stored.

Example: The group MYGROUP contains the following values and the selection bar is at the fourth data item in the group (values between the lines are visible in the panel, and asterisks mark the position of the selection bar):

```
SMITH
JONES
-----
ADAMS
KNIGHT*****
```

```
SNOW
STEWART
-----
MADISON
```

The function:

```
MBD MYGROUP INDEX
```

stores the value 4 in the field INDEX.

MOVE

MOVE places a value in a field or register.

Syntax: `MOVE source destination`

Parameters: *source* numeric, string, field(*sub*), or register.

The source of the move.

destination field(*sub*) or register (\$REG only).

The target of the move.

Remarks: *Source* and *Destination* must be compatible types. For example, you cannot move an alphanumeric source to a numeric destination. If you move values between fields of different lengths, the value is truncated, as necessary.

Examples:

```
MOVE 3 $REG
MOVE "INVALID PARAMETER" MSGFLD
MOVE MYGROUP(INDEX) TEMP
```

The first example moves the value 3 into the register. The second example moves the string "INVALID PARAMETER" into the field MSGFLD. The third example moves the occurrence in MYGROUP given by INDEX into the field TEMP.

MOVEPNL

MOVEPNL moves a panel around the screen in any direction for any distance.

Syntax: `MOVEPNL panel direction distance`

Parameters:	<i>panel</i>	The name of the panel to move. If you specify a null parameter, the current active panel is moved.
	<i>direction</i>	numeric, field(<i>sub</i>), or register. The direction in which to move the panel, indicated by the following numbers:
		1 left
		2 right
		3 up
		4 down
	<i>distance</i>	numeric, field(<i>sub</i>), or register. The distance, in characters, to move the panel.

Example:

```
MOVEPNL  PANEL1  2  20
MOVEPNL  PANEL1  4  10
```

Moves PANEL1 20 characters right and 10 characters down.

MOVEVTEXT

MOVEVTEXT transfers an occurrence of a virtual text group to an alphanumeric or alphabetic field.

Syntax: MOVEVTEXT *group occurrence destination*

Parameters:	<i>group</i>	The name of the virtual text group.
	<i>occurrence</i>	numeric, field(<i>sub</i>), or register. The group occurrence number.
	<i>destination</i>	field(<i>sub</i>) (must be an alphanumeric field). The target of the move.

Example:

```
MOVEVTEXT  VGROUP  4  TXTFLD
```

Moves the contents of the fourth occurrence in virtual text group VGROUP to field TXTFLD.

MPID

MPID obtains the identification number of a panel.

Syntax: `MPID panel destination`

Parameters: `panel` The panel name.
`destination` field(*sub*) or register (\$REG only).
The location where the panel identification number is stored.

Remarks: The calling program uses the panel id for a screenset that uses Micro Focus Panels (see the chapter *Using Dialog System* for details on using Micro Focus Panels) in DOS environments.

This function is not supported in the UNIX environment.

Example: `MPID HELPPNL PANEL-ID`
Stores the identification number of panel HELPPNL in the field PANEL-ID.

MTD

MTD returns the number of the data item at the top of a visible data list to a specified field or register.

Syntax: `MTD group destination`

Parameters: `group` The name of the group on which to act.
`destination` field(*sub*) or register (\$REG only).
The location where the data occurrence is stored.

Example: The group MYGROUP contains the following values and the third data item is at the top of the visible screen (the values between lines are visible in the screen panel, and asterisks mark the position of the selection bar):

```
25
34
20
-----
0
```

```

10
28*****
12
-----
46
45

```

The function:

```
MTD MYGROUP $REG
```

stores the value 3 in the register.

PBDN

PBDN moves the selection bar down a specified number of positions in a group.

Syntax: `PBDN group number`

Parameters:

<i>group</i>	The name of the group on which to act.
<i>number</i>	numeric, field(<i>sub</i>), or register. The number of occurrences by which to move the bar down.

Remarks: If you move the selection bar down by more lines than are visible in the group, the group automatically scrolls down the data, as long as the move does not exceed the number of data items.

Exception: The function causes an exception when the bar is positioned on the last occurrence in the group.

Example: The group MYGROUP contains the following values (the values between lines are visible in the screen panel, and asterisks mark the position of the selection bar):

```

10
-----
23
80*****
17
42
-----
0
4

```

The function:

PBDN 2

has the following effect:

```

10
-----
23
80
17
42*****

-----
0
4

```

If this is followed by:

PBDN 1

the display becomes:

```

10
23
-----
80
17
42
0*****

-----
4

```

PBUP

PBUP moves the selection bar up a specified number of positions in a group.

Syntax: PBUP *group number*

Parameters:

<i>group</i>	The name of the group on which to act.
<i>number</i>	numeric, field(<i>sub</i>), or register. The number of occurrences by which to move the bar up.

Remarks: If you move the selection bar up more lines than are visible in the group, the group automatically scrolls up the data, as long as the move does not exceed the number of data items.

Exception: The function causes an exception when the bar is positioned on the first occurrence in the panel group.

Example: The group MYGROUP contains the following values (the values between lines are visible in the screen panel, and asterisks mark the position of the selection bar):

```

25
10
-----
23
80*****
17
42
-----
4

```

The function:

```
PBUP 2
```

alters the display as follows:

```

25
-----
10*****
23
80
17
-----
42
4

```

PDDN

PDDN scrolls down a data array by the amount specified, without moving the selection bar.

Syntax: `PDDN group number`

Parameters: *group* The name of the group on which to act.

number numeric, field(*sub*), or register.

The amount by which to scroll down the data.

Exception: The function causes an exception when it is not possible to position the data any further down on the panel.

Example: The group MYGROUP contains the following values (the values between lines are visible in the screen panel, and asterisks show the position of the selection bar):

```
FIRST
SECOND
THIRD
-----
FOURTH
FIFTH
SIXTH*****
SEVENTH
EIGHTH
-----
NINTH
TENTH
ELVENTH
TWELFTH
```

The function:

```
PDDN MYGROUP 2
```

alters the display as follows:

```
FIRST
SECOND
THIRD
FOURTH
FIFTH
-----
SIXTH
SEVENTH
EIGHTH*****
NINTH
TENTH
-----
ELVENTH
TWELFTH
```

PDUP

PDUP scrolls up a data array by the amount specified, without moving the selection bar.

Syntax: `PDUP group number`

Parameters: `group` The name of the group on which to act.
`number` numeric, field(*sub*), or register.
The amount by which to scroll the data up.

Exception: The function causes an exception when it is not possible to position the data any further up on the panel.

Example: The group MYGROUP contains the following values (the values between lines are visible in the screen panel, and asterisks show the position of the selection bar):

```
FIRST
SECOND
THIRD
-----
FOURTH
FIFTH
SIXTH*****
SEVENTH
EIGHTH
-----
NINTH
```

The function:

```
PDUP MYGROUP 2
```

alters the display as follows:

```
FIRST
-----
SECOND
THIRD
FOURTH*****
FIFTH
SIXTH
-----
SEVENTH
EIGHTH
NINTH
```

POP

POP pops the most recent physical screen image from the stack and return control to that panel. The panel is the one that was active at the time the PUSH function was executed.

Syntax: POP

Parameters: None

Remarks: Because this function passes control to another panel, you cannot define any subsequent entries in the dialog line containing the POP.

You can initiate dialog in the panel that is popped from the stack by specifying a procedure in the PUSH function that pushed the panel onto the stack.

Exception: The function causes an exception when the stack is empty.

PUSH

PUSH pushes the current physical screen image onto a stack and retains control in the current panel.

Syntax: PUSH *procedure*

Parameters: *procedure* proc or null.

The name of the procedure to execute after the POP associated with this PUSH is performed. A null parameter means that no procedure is executed after the panel is popped from the stack.

Exception: The function causes an exception when the push panel stack is full (that is, when it contains the maximum 16 entries).

Example: PUSH P053

Pushes the current screen image onto the stack, and specifies that after it is popped from the stack, procedure 053 is executed.

RETC

RETC forces control to return to the calling program, where the program can perform some action such as accessing the data base or deriving some new data values.

Syntax: RETC

Parameters: None

Remarks: The only other way to return to the calling program is to define an input field as an exit field (see the chapter *Panel Fields*).

RFT

RFT refreshes the text in the current panel.

Syntax: RFT

Parameters: None

Example: This function is especially useful following the SUSP function, and ensures that all updates made during the suspend are actually displayed.

RFTD

RFTD refreshes all fields and scroll bars from the Data Block. In other words, it updates the optimization rectangle that incorporates all fields and scroll bars in the current panel.

Syntax: RFTD

Parameters: None

RKEY

RKEY redirects the keystroke to another panel.

Syntax: `RKEY panel`

Parameters: `panel` The name of the panel to which the keystroke should be redirected.

Exception: The function causes an exception when the specified key has already been redirected, or the redirected key does not exist on the target screen.

Example: The key in the original dialog line in the current panel is F10. The line:

```
RKEY PANEL2
```

in the dialog line passes control to the F10 dialog entry in PANEL2. After completion of the F10 entry in PANEL2, control returns to the current panel.

RPKY

RPKY repeats the last keystroke upon completion of the current dialog line.

Syntax: `RPKY`

Parameters: None

Remarks: You normally use this function when the key that is pressed requires some action in the current panel that needs a different panel, and the keystroke also needs to be carried forward to this new panel.

You should use this function with caution, because an infinite loop is created at run time if control remains in the current panel.

SAS

SAS sets the internal array size of a group, limiting the amount of scrolling that can take place to the value specified.

Syntax: `SAS group number`

Parameters: `group` The name of the group on which to act.
`number` numeric, field(*sub*), or register.
The new value of the array size.

Exception: The function causes an exception when it attempts to set an array size of zero.

Example: `SAS MYGROUP 20`

You can only display the first 20 occurrences in group MYGROUP.

SB

SB places the selection bar on a specified occurrence in the visible data list of a group.

Syntax: `SB group number`

Parameters: `group` The name of the group on which to act.
`number` numeric, field(*sub*), or register.
The occurrence to position the selection bar on.

Remarks: If you specify a value larger than the number of occurrences on the panel, the bar is positioned on the last panel occurrence.

Exception: The function causes an exception when it attempts to set the bar on a line that is beyond the limit of the panel group.

Example: The visible data list of group MYGROUP looks like this (asterisks show the position of the selection bar):

```
10*****
12
0
25
```

The function:

```
SB MYGROUP 3
```

alters the display as follows:

```
10
12
0*****
25
```

The function:

```
SB MYGROUP 7
```

alters the display as follows:

```
10
12
0
25*****
```

SBOD

SBOD positions the selection bar on the data item specified, and if necessary, scrolls the data to make the relevant lines visible. This function potentially can affect both the selection bar and data positions.

Syntax: `SBOD group occurrence`

Parameters: *group* The name of the group on which to act.
occurrence numeric, field(*sub*), or register.
The data occurrence on which to set the selection bar.

Remarks: The specified data item and selection bar do not necessarily appear at line one. For example, if the required data item is near the end of the list, it is positioned at the end of the visible list rather than on line one.

If, however, the required data item is already visible, the selection bar simply moves to the item without scrolling the group.

Exception: The function causes an exception when it attempts to set the bar either on data item zero, or outside the limit of the Data Block.

Example:

The group MYGROUP contains the following values (values between lines are visible on the screen panel, and asterisks show the position of the selection bar):

```
BRIDGE
CANASTA
-----
POKER
WHIST*****
BLACKJACK
SOLITAIRE
-----
RUMMY
CRIBBAGE
```

The function:

```
SBOD MYGROUP 2
```

alters the display as follows:

```
BRIDGE
-----
CANASTA*****
POKER
WHIST
BLACKJACK
-----
SOLITAIRE
RUMMY
CRIBBAGE
```

SD

SD scrolls the panel area to position the specified data occurrence in a group at the top of the associated panel, leaving the position of the selection bar unaltered.

Syntax:

```
SD group occurrence
```

Parameters:

<i>group</i>	The name of the group on which to act.
<i>occurrence</i>	numeric, field(<i>sub</i>), or register. The data occurrence to be positioned at line one of the screen panel.

Exception: The function causes an exception when it attempts to set the panel area either on data item zero, or outside the limit of the Data Block.

Example: The group MYGROUP contains the following values (values between lines are visible in the screen panel, and asterisks show the position of the selection):

```

1
2
-----
3
4
5*****
-----
6
7
8

```

The function:

```
SD MYGROUP 5
```

alters the display as follows:

```

1
2
3
4
-----
5
6
7*****
-----
8

```

SETCUR

SETCUR positions the cursor in the next input field.

Syntax: SETCUR *position*

Parameters: *position* numeric, field(*sub*), or register.
The position of the cursor in the field.

Remarks: The cursor is moved only when the field is used for input.

Example: SETCUR 5

Positions the cursor at character position 5 in the next input field.

SETF

SETF sets the value of a field to true (1).

Syntax: SETF *item*

Parameters: *item* field(*sub*).
The name of the field.

Example: SETF FLAGS(3)

Sets the third occurrence of group field FLAGS to 1.

SFAT

SFAT sets a run-time display attribute on a particular field on the current panel.

Syntax: SFAT *field attribute*

Parameters: *field* field(*sub*) or register.
The field on which to set the attribute.

attribute attr.
The name of the run-time attribute to be used. You can use any one of the six active run-time attributes (AUX1 through AUX4, INPUT, and ERROR), or NATIVE, which resets the attribute to the colour defined by the screenset. See the section *Attribute Palette* in the chapter *Panel Painting*.

Exception: The function causes an exception when it attempts to set the attribute either on a field that does not exist or on a group field.

Example: SFAT FIELD2 AUX1

Sets the field FIELD2 to the run-time display attribute AUX1.

SGAT

SGAT sets a run-time display attribute on all fields on the current panel.

Syntax: `SGAT attribute`

Parameters: `attribute` attr.

The run-time attribute to be used. You can use any one of the six active run-time attributes (AUX1 through AUX4, INPUT, and ERROR), or NATIVE, which resets the attribute to the colour defined by the screenset. See the section *Attribute Palette* in the chapter *Panel Painting*.

Example: `SGAT AUX2`

Sets all fields of the current panel to the run-time display attribute AUX2.

SHP

SHP displays a panel while retaining control in the current panel.

Syntax: `SHP panel`

Parameters: `panel` The name of the panel to show.

Example: `SHP ERRORPNL`

Displays the panel ERRORPNL, but retains control in the current panel.

SKNF

SKNF moves the cursor to a subsequent input field on the current panel.

Syntax: `SKNF number`

Parameters: `number` numeric, field(*sub*), or register.
The number of fields to skip over.

Remarks: If you specify a value larger than the remaining number of input fields in the panel, the cursor skips to the last input field on the panel; SKNF 9999 is guaranteed to do this.

Exception: The function causes an exception when there is no subsequent input field.

Example: `SKNF 1`

Skips to the next input field in the current panel.

SKPF

SKPF moves the cursor to a previous input field on the current panel.

Syntax: `SKPF number`

Parameters: *number* numeric, field(*sub*), or register.
The number of fields to skip over.

Remarks: If you specify a value larger than the number of input fields before the current field, the cursor skips to the first input field on the panel; SKPF 9999 is guaranteed to do this.

Exception: The function causes an exception when there is no previous input field.

Example: `SKPF 999`

Skips to the first input field of the current panel (there is a maximum of 512 fields per screenset, so this is bound to be greater than the number of previous input fields in the current panel).

SUSP

SUSP delays the panel refresh until either the end of the current dialog, or until the panel is explicitly refreshed with the RFTD or RFT functions.

Syntax: `SUSP`

Parameters: None

Remarks: The function is also terminated if the POP, SHP or GOP functions are invoked.

TERM

TERM terminates a screenset, shutting down all the internal storage used by that screenset.

Syntax:

TERM

Parameters:

None

Remarks:

If you use this function with a screenset based on Micro Focus Panels, the function deletes all of the panels used by the screenset. (See the chapter *Using Dialog System* for details of Micro Focus Panels.)

TIMEOUT

TIMEOUT specifies a procedure to execute after a specified period of time if there is no input in the current panel.

Syntax:

TIMEOUT *timeout procedure*

Parameters:

timeout numeric, field(*sub*), or register.
 The number of seconds to elapse before a timeout occurs during any future input. A value of zero means wait indefinitely for input.

procedure proc, field(*sub*), register, or null.
 The procedure to execute when the timeout occurs. This can be \$NULL.
 You are advised to define this procedure in the default procedure table, because timeout can occur on any panel. If the procedure cannot be found in the panel dialog, an error occurs.

Remarks:

A typical screenset timeout procedure displays a timeout panel, containing a message, and waits for the user to type something. If the user does not respond in the timeout period, a further timeout occurs, and the same timeout procedure is entered.

To return the user to the original panel when he or she acknowledges the timeout, your timeout procedure must PUSH the current environment before doing a GOP to the timeout panel. The timeout

procedure must then POP the stacked panel before it exits from the procedure.

A program loop or stack overflow can occur if the timeout panel itself times out, because the timeout panel is pushed before it is reinvoked. If this happens repeatedly, the stack will eventually overflow. If the user presses a key to make the timeout panel exit before overflow occurs, the user has to repeat the keystroke until all the stacked timeout panels have been popped, before getting back to the original panel. To prevent this, reset the timeout to zero on entry to the timeout procedure, and reset the timeout to the required value before exiting the timeout procedure.

Example: TIMEOUT 120 P032

Specifies that procedure 032 is to be executed if no input takes place in the current panel for two minutes.

TOGF

TOGF toggles the value of a field between true (1) and false (0).

Syntax: TOGF *item*

Parameters: *item* field(*sub*).
 The name of the field.

Example: TOGF FLAG

Toggles the value of field FLAG.

VAL

VAL forces validation of all the input fields on the current screen when an exception condition is being monitored. VAL sets an exception condition when an invalid field is encountered and that invalid field becomes the current field.

Syntax: VAL

Parameters: None

Example: This function must be used in conjunction with a function that monitors an exception condition, for example BPE, XPE, or the error key ERR. If there is no exception condition monitoring, the behavior of this function is unpredictable. For more information about the error key, see the section *Error Key* in the chapter *Key Code List*.

If there are any groups on the screen, validations on that group are performed on all fields in that group up to the current array size.

XIF

XIF tests an expression for validity, and if true, performs the procedure specified, where processing continues. When execution of the procedure is complete, control returns to dialog statement following the XIF. If the expression is false, execution continues at the dialog statement following the XIF function. The following XIF functions are available:

XIF<, XIF<=, XIF=, XIF>, XIF>=, XIFNOT=

Syntax: *XIF-statement expression1 expression2 procedure-name*

Parameters:

<i>XIF-statement</i>	One of the XIF statements specified above.
<i>expression1</i>	numeric, string, field(<i>sub</i>), or register. The first operand of the test.
<i>expression2</i>	numeric, string, field(<i>sub</i>), or register. The second operand of the test.
<i>procedure</i>	proc, field(<i>sub</i>), or register. The procedure to perform if the test is true, or an alphabetic field that contains the name of the procedure to perform. Fields and procedures might have the same name, in which case the contents of the field are used.

Remarks: *Expression1* and *expression2* must be compatible types. For example, an expression cannot test a numeric literal against an alphanumeric literal. A similar set of conditional functions are provided that execute the target procedure and do not return control to the calling procedure. See the IF function earlier in this chapter.

Example: XIF> GOODS-COST 1500 P100

If the field GOODS-COST has a value greater than 1500, executes the procedure P100, then returns to the next function.

XP

XP executes a procedure and returns control to the next function in the dialog line.

Syntax: *XP procedure*

Parameters: *procedure* proc, field(*sub*), or register.
The procedure to be executed.

Example:
XP PROCNO
BEEP

Executes the procedure whose number is given by field PROCNO. When the procedure has executed, control returns to the dialog and executes the BEEP function.

XPD

XPD branches to a procedure, specified by the number in the parameter and the position of the selection bar in a group, then returns control to the next function after the procedure has been performed.

Syntax: *XPD group procedure*

Parameters: *group* The name of the group on which to act.
procedure proc, field(*sub*), or register.
The procedure to execute if the selection bar in *group* is at position one, or the value to use in the destination procedure calculation.

Remarks: The destination procedure number is $n+m-1$, where:

- n is the value of *procedure*.
- m is the position of the selection bar in the group specified by *group*. If the group does not have a selection bar (as is the case

with virtual attribute groups), *m* is the position of the first visible line in the data.

Example: XPD MYGROUP 65

If the selection bar in MYGROUP is in position one, procedure 065 is executed; if the selection bar is in position four, procedure 068 is executed.

XPE

XPE executes a procedure and return control to the next function in the dialog line upon completion. The procedure is executed only if the next function in the dialog entry causes an exception condition (for example, attempting to scroll beyond the limit of a scrolling group).

Syntax: XPE *procedure*

Parameters: *procedure* proc or field(*sub*).

The exception procedure to be executed.

Remarks: A function that can cause an exception has an entry in its description specifying the conditions under which this happens.

Example: XPE P045
POP

Procedure 045 is executed only if the POP function causes an exception by attempting to pop an empty stack.

XPR

XPR executes a procedure, specified by the number in the parameter and the value in the register, then returns control to the next function in the dialog line upon completion.

Syntax: XPR *procedure*

Parameters: *procedure* proc, field(*sub*), or register.
The procedure to execute if the value in the register is one, or the value used in the destination procedure calculation.

Remarks: The destination procedure number is $n+m-1$, where:

- n is the value of *procedure*.
- m is the value in the register.

Example:
MOVE 4 \$REG
XPR 65

This executes procedure 068.

21 Syntax of Import/Export Files

This chapter describes the syntax of the files created by the export utility from screenset definitions (see the sections *Import (F9)* and *Export (F10)* in the chapter *Using Dialog System* for information about how to use the import and export facilities).

The DOS, OS/2, and UNIX versions all support the full Import/Export syntax.

The following notations are used in this section:

<i>upper-case word</i>	a required key word
<i>lower-case word</i>	a user-defined word, literal, or syntactic entry
[]	material in brackets is an option or a portion of syntax that you can include or omit, as required.
{ }	one of the options in the braces must be specified
[]	any number or none of the options between the parallel lines can be specified, but each option can only be specified once.
{ }	one or more of the options between the parallel lines can be specified, but each option can only be specified once.
...	the position at which the syntax can be repeated.

21.1 Data Types

The table below shows the mappings of types used in data declaration to the types used in Dialog System:

Data-Type	Dialog System Representation
Character	X
Alphabetic	A

Data-Type	Dialog System Representation
integer, n	9 S
Decimal	9 S
data-group-num	An integer, denoting group number.
text-value	Number of characters to be specified for one line. Maximum of 80 characters. A single quotation mark (") must be represented by two quotation marks (") in the string.
start-row, start-col	An integer giving coordinates of top left corner of the rectangular area.
end-row, end-col	Integer giving coordinates of bottom right corner of the rectangular area.
color	One of the following colors: BLACK, BLUE, GREEN, CYAN, RED, MAGENTA, BROWN, WHITE.

21.2 Full Import/Export Syntax

The following syntax is the full screenset definition syntax.

21.2.1 Screenset Definitions

```

FORM screenset-name
  [ screenset-details ]
  [ form-data-declaration ]
  [ error-messages-declaration ]
  [ data-validation-declaration ]...
  [ global-dialog-declaration ]
  [ panel-declaration ]...
END FORM

```

21.2.1.1 screenset-details

```

SCREENSET DETAILS
  { | FIRST panel-name                               | }
  { | SCREENSET { NORMAL|MF }                       | }
  { | SEARCH { GLOBAL|LOCAL }                      | }
  { | KEYTRANS { ON|OFF }                          | }
  { | COLORIZE { ON|OFF }                          | }
  { | [system-color-name attribute-details]...     | }
  { | END COLORIZE                                  | }
END DETAILS

```

21.2.1.2 form-data-declaration

```

FORM DATA
[[ GROUP data-group-num VERTICAL OCCURS n
  [ data-declaration ] ...           ] |
 [data-declaration]...               ]
END DATA

```

21.2.1.3 error-messages-declaration

```

ERROR MESSAGES msg-filename
  [ msg-no "text-value" ]...
END MESSAGES

```

21.2.1.4 data-validation-declaration

```

DATA VALIDATION data-name
  { | NULL DISALLOWED [ ERRMSGNO msg-no ]         | }
  { | RANGE-TABLE { TRUE|FALSE }                  | }
  { | [ ERRMSGNO msg-no ]                          | }
  { | { | { X-low } [ THRU { X-up } ] | | }         | }
  { | { | { 9-low } [ THRU { 9-up } ] | | }...     | }
  { | END RANGE-TABLE                             | }
  { | CHECKDIGIT { VALUES|DIGITS }                | }
  { | [ ERRMSGNO msg-no ]                          | }
  { | [ DIVISOR n ]                                | }
  { | [ WEIGHT (n.m) ]...                          | }
  { | END CHECKDIGIT                               | }
  { | DATE date-format [ ERRMSGNO msg-no ]         | }
END VALIDATION

```

21.2.1.5 *panel-declaration*

```
PANEL panel-name
  [ panel-details ]
  [ attribute-palette-declaration ]
  [ BORDER { NONE|SINGLE|DOUBLE } ]
  [ panel-text-declaration ]...
  [ panel-protect-declaration ]...
  [ panel-field-declaration ]...
  [ panel-group-declaration ]...
  [ panel-dialog-declaration ]
END PANEL
```

21.2.1.6 *global-dialog-declaration*

```
GLOBAL DIALOG
  CASE-SENSITIVITY { ON|OFF }
  [ dialog-declaration-1 ]...
  [ dialog-declaration-2 ]...
END DIALOG
```

21.2.1.7 *panel-dialog-declaration*

```
PANEL DIALOG
  CASE-SENSITIVITY { ON OFF }
  [ dialog-declaration-1 ]...
  [ dialog-declaration-2 ]...
END DIALOG
```

21.2.1.8 *data-declaration*

```

      { CHARACTER(n) [ ERROR-FIELD ] }
      { ALPHABETIC(n) }
data-name { INTEGER(n) [ IMPLICIT SIGN
              [ COMPUTATIONAL]] |
          [ COMPUTATIONAL ] |
          [ COMP [SIGNED]] |
          [ COMP-3[SIGNED]] |
          [ COMP-5[SIGNED]] |
          [ COMP-X ] }
```

```

{ DECIMAL(n.m) [ IMPLICIT SIGN]      |
                                     |
                                     | [COMP [SIGNED]]
                                     | [COMP-3[SIGNED]]
                                     | [COMP-5[SIGNED]]
                                     | [COMP-X          ] }

```

21.2.1.9 panel-details

```

ANCHOR TOP|BOTTOM
SIZE LINE start-row END-LINE end-row COLUMN
      start-col END-COLUMN end-col

```

21.2.1.10 attribute-palette-declaration

```

ATTRIBUTE PALETTE
  [ DEFAULT attribute-details ]
  [ LIST-1 attribute-details ]
  [ LIST-2 attribute-details ]
  [ LIST-3 attribute-details ]
  [ LIST-4 attribute-details ]
  [ LIST-5 attribute-details ]
  [ AUX1 attribute-details ]
  [ AUX2 attribute-details ]
  [ AUX3 attribute-details ]
  [ AUX4 attribute-details ]
  [ ERROR attribute-details ]
  [ INPUT attribute-details ]
  INP-ATTR { ON OFF }
END PALETTE

```

21.2.1.11 panel-text-declaration

```

LITERAL TEXT
  LINE start-row COLUMN start-col
  VALUE "text-value"
  [ attribute-details ]
END LITERAL

```

21.2.1.12 panel-protect-declaration

```
PANEL PROTECT
  [ panel-protect-field-declaration ]...
END PROTECT
```

21.2.1.13 panel-field-declaration

```
PANEL FIELD data-name [ SCROLLING ]

  LINE start-row COLUMN start-col

  LENGTH {n } [ ORDER order-ind ]
          {n.m}

          { IN   }
          { OUT  }
  USAGE { EXIT }
          { XMOD }
          { XREG }
          { XENT }

  [           [ | AUTO   | ] ]
  [           [ | REQD   | ] ]
  [ PROPERTIES [ | FULL   | ] ]
  [           [ | NOECHO | ] ]
  [           [ | UPPER  | ] ]
  [           [ | LOWER  | ] ]

          { ALPHABETIC   }
          { ALPHANUMERIC }
          { NUMERIC      }
          { SIGN         }
  FORMAT { SUPPRESS     }
          { SIGN-SUPPRESS }
          { 99/99       }
          { 99/999     }
          { 99/99/99   }
          { 99/AAA/99  }
          { 99/99/9999 }

  [ DELIMITER "date-delim" ]

  [           { USERFMT1 } ]
  [ USERFMT { USERFMT2 } ]
```



```

[          { USERFMT3 } ]
[          { USERFMT4 } ]

END FIELD

```

21.2.1.14 panel-group-declaration

```
PANEL GROUP panel-group-name VERTICAL
```

```

LINE start-row COLUMN start-col
                                HEIGHT n WIDTH n

```

```

                                { BAR } }
{ DATA ACC-EXIT {           } }
TYPE {              { END } }
{ ATTRIBUTE      }
{ FIXED-TEXT     }
{ VIRTUAL-TEXT  }

```

```

[ SELECT BAR ]
[ [ LENGTH integer ] ]
[ [ attribute-details ]... ]
[ END BAR ]

```

```

{ [ panel-field-declaration ]... }
{ }
{ [ { OCCURRENCE group-occ } ] }
{ [ [ LENGTH n attr-dets]... ]... ] }
{ [ { END OCCURRENCE } ] }
{ [ ]... }
{ [ { OCCURRENCE group-occ } ] }
{ [ { VALUE "text-value" }... ] }
{ [ { END OCCURRENCE } ] }

```

```
END GROUP
```

21.2.1.15 dialog-declaration-1

```

[ EK ] proc-num [ | fn p1 (subs-1) | ]
                [ | p2 (subs-2) | ]
                [ | p3 (subs-3) | ]...

```

EK signifies an event key. You must place each function (with any parameters) in square brackets.

21.2.1.16 dialog-declaration-2

```
[ EK ] key-value [ | fn p1 (subs-1) | ]
                [ | p2 (subs-2) | ]
                [ | p3 (subs-3) | ]...
```

EK signifies an event key. You must place each function (with any parameters) in square brackets.

21.2.1.17 attribute-details

```
BACKGROUND bground-color
           FOREGROUND fground-color
           [ | INTENSITY BLINK | ]
```

21.2.1.18 panel-protect-field-declaration

```
data-name      n n n n n n n n n n n
```

21.2.1.19 Data Types

n	integer
date-format	{ DDMM } { MMDD } { MMY Y } { YYMM } { YYDDD } { DDMMYY } { YYMMDD } { MMDDYY } { DDMMYY }
date-delim	One character denoting the delimiter used for a date field (for example "/").
data-name, panel-name, panel-group-name	Maximum of 8 characters. Names longer than 8 characters are truncated.
system-color-name	Dialog System system color name, values SYS-1 through SYS-16.
data-group-num	An integer denoting group number.

<code>msg-filename</code>	Error messages filename. Maximum of 12 characters.
<code>msg-no</code>	Error message number. Integer.
<code>text-value</code>	A line of text characters. Maximum of 80 characters. To represent a quote character ("), use two quotes (" ") in the string.
<code>X-low,</code> <code>X-up</code>	Lower and upper alphanumeric range value, specified as an alphanumeric literal, in quotes. Maximum of 20 characters.
<code>9-low,</code> <code>9-up</code>	Lower and upper numeric range value, specified as a numeric literal. Maximum of 18 characters.
<code>start-row,</code> <code>start-col</code>	Integer giving coordinates of top left corner of the rectangular area.
<code>end-row,</code> <code>end-col</code>	Integer giving coordinates of bottom right corner of the rectangular area.
<code>group-occ</code>	Integer denoting occurrence of panel group.
<code>order-ind</code>	One character denoting field order. Possible values are 0-9, A-Z, a-z.
<code>background-color,</code> <code>fground-color</code>	Can be one of the following colors: BLACK, BLUE, GREEN, CYAN, RED, MAGENTA, BROWN, WHITE.
<code>proc-num,</code> <code>key-value</code>	Valid Dialog System procedure number or key mnemonic. Maximum of 4 characters.
<code>fn</code>	Valid Dialog System function mnemonic. Maximum of 8 characters.
<code>p1, p2, p3, subs-1,</code> <code>subs-2, subs-3</code>	Parameter and subscript values. Maximum of 30 characters.

21.2.2 Sample Text File

The following sample external text file was created by the export facility from components of the screenset **customer.s** and can be imported into a Dialog System screenset.

FORM CUSTOMER

SCREENSET DETAILS

```

FIRST full-scr
SCREENSET NORMAL
SEARCH GLOBAL
KEYTRANS OFF
COLORIZE OFF
END COLORIZE
END DETAILS

```

FORM DATA

```

C-CODE CHARACTER(5)
C-NAME CHARACTER(15)
C-ADDR1 CHARACTER(15)
C-ADDR2 CHARACTER(15)
C-ADDR3 CHARACTER(15)
C-ADDR4 CHARACTER(15)
C-LIMIT INTEGER(4) IMPLICIT SIGN
C-AREA CHARACTER(1)

```

GROUP 1 VERTICAL OCCURS 10

```

ORD-NO INTEGER(6) IMPLICIT SIGN
ORD-DATE INTEGER(6)
ORD-VAL DECIMAL(4.2)
PAY-VAL DECIMAL(4.2) IMPLICIT SIGN
ORD-BAL DECIMAL(4.2) IMPLICIT SIGN
END GROUP

```

```

C-BAL DECIMAL(5.2) IMPLICIT SIGN

```

GROUP 2 VERTICAL OCCURS 1

```

DEL-FLG INTEGER(1)
LOAD-FLG INTEGER(1)
SAVE-FLG INTEGER(1)
CLR-FLG INTEGER(1)
EXIT-FLG INTEGER(1)
END GROUP

```

```

ERR-MSG CHARACTER(20)
C-ORD-NO INTEGER(6)
C-ORD-DT INTEGER(6)
C-ORD-VL DECIMAL(4.2) IMPLICIT SIGN
C-PAY-VL DECIMAL(4.2) IMPLICIT SIGN

```

END DATA

```

GLOBAL DIALOG
  CASE-SENSITIVITY ON
    P255 [RFTD  ]
END DIALOG

END FORM

```

21.2.3 Restrictions

Since the import process is driven by DS keywords (PANEL, GROUP), it is important that user-defined words (data-names, panel-names) are not also DS keywords or the import will stop. Note that Dialog itself does not restrict the use of these words, only the import module. Thus it is possible to create a screenset, export it successfully and for the import module to be unable to import it again.

As well as the keywords listed in the syntax diagrams, there are some additional internal keywords that the import module disallows as user-defined words. These are:

```

ACC-EXIT
ALPHABET
ALPHANUM
ATTRIB
BGROUND
CASESEN
CHAR
CHECKDIG
OLDCOMP
DELIMIT
DEPTH
DISALLOW
ERRORFLD
FIX-TXT
FGROUND
INTENS
OCCURR
PROPS
RECT
RNGE-TAB
SCROLL
SSET
SIGN-SUP

```

VALID
VIRT-TXT
WIDTH

You must not export a screen containing the circumflex (^) character, as it is reserved for internal use in export/import.

22 Error Messages

This chapter lists the error messages that might be returned while you are developing your Dialog System program.

22.1 Run-time System Error Messages

The following error messages are returned by the Dialog System Run-time System software. If errors are detected while you are running your program through the trap screen, the error message number appears in the error code field and the message is displayed at the bottom of the screen.

There are two types of run-time system error messages: recoverable and fatal. Recoverable errors can be trapped in your program, so you can take corrective action and continue to run the program. Fatal errors cause a message to be sent to the screen then terminate your program immediately. Although you cannot recover from a fatal error during the run of your program, once it has terminated, you might be able to rectify the conditions that caused the error.

Screenset not initialized (fatal)

- You have probably set the Ds-Control field of the control block to a value of "C" before using a value of "N" for a new screenset.

Cannot open screenset (fatal)

- The system has been unable to find a screenset you have attempted to access in your program.

Error reading file (fatal)

- A disk corruption has occurred.

Invalid screenset (fatal)

- The screenset you are attempting to open is not recognized as a valid screenset.
- You have used an invalid filename.
- If the screenset is from a previous release of Dialog System, ensure it is resaved with the current version number.

Cannot create panel (fatal)

- There is a maximum of 50 Micro Focus Panels allowed per screen set (Windows only).

Dynamic internal error (fatal)

- This should never occur. If it does, it might be because a memory limit has been reached. Please contact Product Support.

Invalid function code (fatal)

- An invalid function was found in the screenset, but Dialog System cannot identify at what stage the error occurred. Check the functions in your screenset. If you cannot find an invalid function, contact Product Support.

Invalid procedure number (recoverable)

- You have specified a procedure that does not exist. Check that the procedure number you have specified is one that you have already defined.

Validation program error (fatal)

- The validation program has detected one of errors 010 - 016.

Invalid data block version number (fatal)

- The Data Block version number in the copyfile does not match the version number in the screenset.

Panel limit reached (fatal)

- There is a limit of 50 active Micro Focus Panels per screen set (Windows only).

Error file missing (fatal)

- You have tried to access the error file yourself (see the chapter *Programming*), but it cannot be found. This error should not occur during normal operation of the Dialog System Run-time System.

Subscript error (fatal)

- You have used a subscript in dialog that is greater than the size of the associated group.

Procedure stack overflow (fatal)

- The maximum level of nesting of procedures (that is, XP nesting) is 16.

Ctrl-break pressed (recoverable)

- Ctrl+Break was pressed while in the Dialog System Run-time software (Windows only). Ctrl+Break detection is only active if the appropriate control parameter is used on the screenset initialization call. Control is passed back to the user program, where you can perform appropriate processing or reinvoke the Dialog System Run-time software.

Error on trace file (fatal)

- The system has encountered an error, such as a full disk, while trying to access the trace file.

Screenset not found (fatal)

- This error is returned by the common GUI/Character dsrun module for the first screenset load of the run. It indicates that the named screenset has been searched for in both character and GUI form, but has not been located.

22.2 Definition Time Error Messages

The following error messages are produced by the Dialog System definition software.

22.2.1 Data Definition

Cannot change format - field in group

- You cannot change the format of a field contained in a group unless you delete the group definition.

Cannot change format - field in use

- You cannot change the format of a field used in a panel unless you delete the field from the panel(s) that use it.

Cannot change format of validated field

- You cannot change the format of a field for which validation has been defined unless you delete the validation.

Cannot change length - field in group

- You cannot change the length of a field contained in a group unless you delete the group definition.

Cannot change length - field in use

- You cannot change the length of a field used in a panel unless you delete it from the panel(s) that use it.

Cannot decrease repeats - field(s) in use

- You cannot amend field(s) that are used in a panel unless you delete them from those panels.

Decimal length cannot be specified for this type

- You can specify decimal length only for numeric and signed numeric fields.

Error field already defined

- You can define only one error field for any screenset.

Error field cannot repeat

- You can define only one error field for any screenset.

Error field must be alphanumeric

- You can use only an alphanumeric field as an error field.

Field(s) in copied or cut block

- This field has an entry in the stack.

Field length cannot be zero

- The minimum length of a data field is one.

Field used in dialog

- This field has been referenced in at least one dialog table.

Field used on at least one panel

- You cannot delete fields that have been used in a screenset.

Group already defined

- Data group names must be unique.

Group empty. Exiting now will convert group to text type. Are you sure? Y/N

- You have cut all the fields in a data group to a stack. If you exit from the group, the group is no longer treated as a data group, but becomes an empty text group.

Group must repeat at least once

- The minimum number of occurrences in a group is one.

Integer length cannot be greater than 18

- The maximum length of an integer is 18.

Integer length cannot be zero

- The minimum length of an integer is one.

Numeric length cannot be greater than 18

- The maximum length of a numeric field, including any decimal part, is 18 bytes.

Only A, S, X, 9 allowed for field format

- Valid field formats can be alphabetic, signed numeric, alphanumeric or numeric.

Only space valid for field comp with A format

- You cannot specify COMP format for an alphabetic field.

Only space, CX, or C5 valid for field comp with X format

- You can specify only space, CX, or C5 for an alphanumeric field.

Only space, C, CX, C3, C5 valid for field comp

- You have specified an invalid COMP format.

Only space, C, C3, C5 valid for field comp with S format

- The only COMP formats you can specify for a signed numeric field are COMP, COMP-3, or COMP-5.

Only 255 groups may be defined

- There is a maximum of 255 groups in a screenset.

Only 512 fields may be defined

- There is a maximum of 512 data fields in a screenset.

Spaces not allowed in field name

- Field names must start with a alphanumeric letter and must not include any spaces.

This field has already been defined

- A field cannot be defined more than once.

This is not a group

- There is a minimum of one occurrence in a data group.

Validation not possible for error field

- An error field can only be used as an output field. Validation is only possible for input fields.

22.2.2 Validation Definition

Divisor must be greater than zero

- The check digit system can use any non-zero, two-digit divisor.

Internal memory error occurred

- There is insufficient memory to save the validation.

No more room in the list

- The maximum number of validations that can be held in the list is 30.

Upper limit must be greater than the lower limit

- The upper limit of the validation must be greater than the lower limit.

22.2.3 Error Message Definition

Corrupted File

- This file has been edited outside of Dialog System error definition since it was last saved.

Incorrect File Type

- This file has not been created by Dialog System error definition.

Invalid error msg number

- You must enter a valid error message number in the range of 1 - 998.

Invalid File Name

- A filename must start with an alphanumeric letter and must not include any spaces.

No place to duplicate

- Error definition cannot allocate a new number to this error message.

No place to insert

- Error definition cannot allocate a new number to this error message.

Nothing to restore

- The error definition stack is empty.

22.2.4 Panel Field Definition

Attempt to overwrite an existing field or group

- A field cannot overwrite an existing field or group.

Cannot insert on an existing field

- You cannot place a panel field on top of an existing field.
- You have exceeded the maximum of 200 panel fields for one panel.

Cannot insert on an existing group

- You cannot place a field on top of an existing group.

Ctrl+End not allowed from this screen

- Ctrl+End keystroke is not active from this panel.

Decimal length not allowed with integer length zero

- A field with a decimal part must have an integer part.

Error deleting field

- An error has been detected while deleting a field.

Field cannot be positioned off the screen

- A field must be positioned in the current panel.

Field length must be shorter than in the data block

- The length of a field on a panel cannot be greater than its length in the Data Block.

Field stack is empty

- There are no fields stored in the stack to use.

Field stack is full

- The field stack can hold a maximum of 16 fields.

Groups must be modified in group maintenance

- Fields contained in groups can only be modified from the group maintenance menu.

Internal system error

- An internal memory allocation error has occurred. Run Dialog System with more memory, if possible.

Invalid decimal length

- The decimal panel length selected is not valid for this field.

Invalid length

- The panel length selected is not valid for this field.

Length cannot be zero

- The length of a field on a panel must be greater than zero.

Not positioned on a field

- The operation requires you to position the cursor on an existing field.

Scrolling not allowed on an output field

- You cannot make an output field scroll horizontally.

User formats cannot be specified for output fields

- You cannot specify that an output field is a user format field.

22.2.5 Panel Group Definition

Group cannot be expanded any more

- You cannot expand a group beyond the limits of the panel coordinates defined.

Group cannot be reduced any more

- You have reached the minimum group size. Data groups must contain at least one data field. Attribute groups must be at least one line high.

Group cannot overlap a field

- You cannot define a group on top of an existing panel field.

Group cannot overlap another group

- You cannot define a group on top of an existing panel group.

Group max size cannot be exceeded

- The maximum size of a group is 4000 bytes.

Group used in dialog

- You cannot delete a group that is referenced in a dialog table.

Invalid group name

- Group names must start with an alphanumeric letter and must not include any spaces.

Name already used in screenset

- Group names must be unique.

22.2.6 Dialog Definition

Alphanumeric literal invalid here

- You cannot use an alphanumeric literal for this parameter.

A select bar has not been defined for this group

- The parameter for this function requires a group with a selection bar.

Dialog not allowed after GOP, POP, BP, or POPGOP

- No further functions are allowed following the GOP, POP or BP functions. Function entry should not be blank.
- You must enter a valid function.

Invalid combination of operands for MOVE

- The parameters of the MOVE statement are not compatible types.

Invalid comparison of operands

- The parameters of the IF statement are not compatible types.

Invalid field subscript

- You must use a subscript that is a numeric literal, a numeric field or the register.

Invalid Function

- You must enter one of the valid functions listed in the chapter *Function Code List*.

Invalid key entry

- You must enter one of the valid keys listed in the chapter *Programming*.

Invalid parameter

- You have specified an invalid parameter for this function.

Key entry cannot be blank

- You must enter a correct key value.

Key has already been defined

- This key has already been defined in this dialog table.

No global procedures have been defined

- You have not defined any global procedures for this screenset.

No groups have been defined

- You have not defined any groups have been defined for the screenset.

No local procedures have been defined

- You have not defined any local procedures for this screenset.

No master fields have been defined

- You have not defined any fields for the screenset.

No more dialog entries can be made

- There is a maximum of 100 keys in a dialog table.

No panels have been defined

- You have not defined any panels for the screenset.

Numeric literal invalid here

- You cannot use a numeric literal for this parameter.

Parameter cannot be blank

- You must enter one or more valid parameter(s) for this function.

PDUP, PDDN not valid for fixed text group

- The PDUP and PDDN functions are valid for virtual text, virtual attribute or data groups.

Procedure has already been defined

- You have already defined this procedure in this dialog table.

Procedure has not been defined

- You have specified as a parameter a procedure that you have not defined in either the local or global dialog tables.

Procedure used in dialog

- This procedure has been used at least once in the local or global dialog tables.

Procedures are nested too deeply

- There is a maximum of 16 nested procedures.

Procedures have been defined

- You have already defined this procedure name.

Some keys have both upper and lower case entries

- You must leave case sensitivity on.

Subscript cannot be blank

- The field specified is a group item that requires a subscript.

This field is not a flag

- You can use only a single-digit numeric field with a value of zero or one as a flag.

This field is not alphabetic or alphanumeric

- You can use only an alphabetic or alphanumeric field for a parameter.

This field is not numeric

- You can use only a numeric field for this parameter.

This system name is invalid here

- You cannot use \$NULL or \$REG for this parameter.

Valid procedure numbers are P000 - P255

- You must enter a valid procedure name, consisting of the letter "P" followed by a number ranging from 0 to 255.

Warning: Recursive procedure definition

- Definition of this procedure will cause recursion, which might cause a loop.

22.3 Screen Painting Error Messages

Dialog System might produce the following error messages when you paint your screensets.

Block stack full

- You can stack a maximum of 16 blocks.

Block still in existence

- Only one block can be active at any one time.

Can't abandon after copy/restore

- The abandon facility is available only after you use a cut block facility.

Clear panel? Are you sure? Y/N

- The text, data and attributes will be cleared from the current panel if you proceed.

Config file tag not found

- The configuration file (**dsdef.cfg**) does not contain the required information.

Currently standalone mode

- Micro Focus Screens is being run in standalone mode, rather than as part of Dialog System. You cannot run the screenset.

Cut/Copy without paste? Y/N

- The current block will be discarded unless you paste it.

Data block too large (64K)

- The maximum size of the Data Block is 65535 bytes.

Delete panel. Are you sure? Y/N

- The specified panel will be deleted if you proceed.

Error closing file

- File I/O error. Check your available disk space, ensure that the requested file is not a read only, check that you have used a valid filename. If you still receive this error, contact Product Support.

Error creating file

- File I/O error. Check your available disk space, ensure that the requested file is not a read only, check that you have used a valid filename. If you still receive this error, contact Product Support.

Error reading file

- File I/O error. Check your available disk space, ensure that the requested file is not a read only, check that you have used a valid filename. If you still receive this error, contact Product Support.

Error writing file

- File I/O error. Check your available disk space, ensure that the requested file is not a read only, check that you have used a valid filename. If you still receive this error, contact Product Support.

Exit. Are you sure? Y/N

- You will exit the definition software if you proceed.

Exit without saving? Y/N

- You have made changes to the screenset that will be lost if you exit without first saving the screenset.

Field definition overlap

- The marked area overlaps or splits a field definition.

Field not in master group

- A field in this panel group is not contained in a data group.

Fields in different groups

- The group specified contains data fields that belong to different data groups.

File already exists. Overwrite? Y/N

- The filename already exists in the specified directory.

File not found

- The system was unable to find the requested file in the directory specified.

FSI - Data truncated

- Internal error. Try to run Dialog System with more memory, if possible. If you still receive this error, contact Product Support for assistance.

FSI - D0 allocation fail

- Internal error. Try to run Dialog System with more memory, if possible. If you still receive this error, contact Product Support for assistance.

FSI - Heap full

- Internal error. Try to run Dialog System with additional disk space or more memory, if possible. If you still receive this error, contact Product Support for assistance.

FSI - No data

- Internal error. Contact Product Support for assistance.

Generate without saving? Y/N

- You have made changes to the screenset that will not be included if you generate the data definitions/skeleton program without first saving the screenset.

Group definition overlap

- The marked area overlaps or splits a group definition.

Group nesting not allowed

- You cannot nest group definitions.

Illegal order character

- You have specified an invalid field order character. Valid characters are 0 - 9, A - Z, and a - z.

Incompatible version

- The screenset specified is not compatible with this version of Dialog System.

Initialize. Are you sure? Y/N

- All panels, data, and dialog in the current screenset will be deleted if you proceed.

Internal error on heap open

- Internal error. Contact Product Support for assistance.

Internal error on heap close

- Internal error. Contact Product Support for assistance.

Internal error on heap read

- Internal error. Contact Product Support for assistance.

Internal error on heap write

- Internal error. Contact Product Support for assistance.

Internal heap error

- Internal error. Contact Product Support for assistance.

Internal overflow

- Internal error. Contact Product Support for assistance.

Invalid delete

- You cannot delete a field that is part of a group. You must use the group maintenance menu to alter the group.

Invalid FEM function

- Internal error. Contact Product Support for assistance.

Invalid function

- Internal error. Contact Product Support for assistance.

Invalid line in config file

- The configuration file (**dsdef.cfg**) contains an invalid line.

Invalid mark

- The marked area is not valid for the selected function.

Invalid - mark is active

- This function is not available while an area is being marked.

Load without saving? Y/N

- The current screenset will not be saved before the next screenset requested is loaded.

Name already exists

- The name specified already exists in the current screenset.

No block defined

- The selected function requires you to define a block. For example, to specify a new group there must be a block specifying its dimensions.

No block to restore

- The block stack is empty.

No border possible

- There is a minimum panel height of three lines and width of three characters required for a border.

No field in group definition

- The marked area does not contain a field definition.

No screenset loaded

- A screenset must be loaded before it can be run.

No occurs

- The group only has one occurrence.

Non-existent field

- The cursor is not positioned over a panel field.

Not on a group

- There is no group in which to insert the field.

Not on Input or I-O field

- You cannot specify field order for an output field.

Not possible; block has group(s)

- Once a block containing groups has been cut from the panel, you must paste or copy it back onto the current panel.

Panel field table full

- There is a maximum of 100 data fields allowed per panel.

Panel group limit reached

- There is a maximum of 30 panel groups allowed.

Panel limit reached

- There is a maximum of 50 panels that can be visible on the screen. Use the unshow-panel function to remove some of the panels from the screen.

Protected area - field

- You cannot overtype a data area.

Referenced in dialog

- This item is a referenced in a dialog table.

Screenset copied (name changed)

- The screenset specified cannot be loaded because it has been copied to a new name. Save it under its original filename before you attempt to reload it.

Screenset empty

- No panels in the screenset.

Source data repeat limit

- You have specified more repetitions for your panel group than you defined in your data group.

Too many panel attributes

- The attribute buffer is full; no more attributes are allowed. Try to group attributes together, if possible.

Unknown file format

- The file specified cannot be loaded.

Validation space full

- The validation buffer is full. Save and reload your screenset or delete any unnecessary validations.

Virtual group area

- You can modify an area defined as a virtual group only using the group function.

Virtual space full

- The virtual space buffer is full. Save and reload your screen set, or delete unnecessary virtual text, attributes, or selection bars.

22.4 Import Utility Error Messages

The following error messages might be produced by Dialog System when you import a screenset.

22.4.1 Recoverable Errors

A Check Digit validation is allowed for numeric field only

- You can specify check digit validation only for numeric fields that have no decimal places.

A maximum of 30 Range/Table validations are allowed

- The maximum number of range/table validations that can be held in the list is 30 for each field.

A maximum of 18 weights are allowed

- The maximum number of weights you can enter to use for check digit validation is 18.

Attempted to overlap existing group: *group-name*

- An imported panel group cannot overlap an existing group.

Attributes exceed panel height

- The attribute rectangle being imported exceeds the defined height of the panel.

Attributes exceed panel width

- The attribute rectangle being imported exceeds the defined width of the panel.

Case sensitivity must be ON

- Some keys have upper- and lower-case entries in the dialog, so case-sensitivity must remain on.

Date format does not match field size: *field-name*

- You can define date format for numeric fields of size 4, 5, 6, or 8 with no decimal places, or alphanumeric of size 7, but the date format must be the same size as the field.

Decimal length must be zero: *field-name*

- A field is an integer so must not have a decimal part.

Dialog not allowed after GOP,POP,BP,BPD or BPR

- No further functions are allowed following the GOP, POP or BP functions. Function entry should not be blank.

Dialog table is full

- There is a maximum of 100 keys in a dialog table.

Duplicate Dialog key entry: *key-value*

- This key is already defined in this dialog table.

Error message number already exists: *message-no*

- A file already exists with a message with this number.

Error message number cannot exceed 998

- You must enter a valid error message number in the range 1 through 998.

Field does not exist: *field-name*

- Validation has been specified for a non-existent field.

Field is not a Dialog flag: *field-name*

- Only a single-digit numeric field can be used as a flag.

Field is not alphanumeric: *field-name*

- Only an alphanumeric field can be used for this parameter.

Field is not numeric: *field-name*

- Only a numeric field can be used for this parameter.

Field name already exists

- An existing field in the Data Block cannot be overwritten.

Field not found: *field-name*

- There is a reference to a field that does not exist in the datablock.

Function has incorrect number of params: *function-value*

- You must specify the correct number of parameters for a function in dialog, as listed in the chapter *Function Code List*.

Group not found: *group-name*

- There is a reference to a group that does not exist.

Group will not fit on panel: *group-name*

- The size of an imported group is too large for the panel.

Invalid Dialog function: *function-value*

- The dialog must contain only valid functions listed in the chapter *Function Code List*.

Invalid Dialog IF statement

- The parameters of the IF statement in dialog are not compatible types.

Invalid Dialog key: *key-value*

- The dialog must contain only valid keys listed in the chapter *Programming*.

Invalid Dialog parameter subscript: *param-value*

- The subscript must be a numeric literal, a numeric field, or the register.
- The field is not a group field so cannot have a subscript.

Invalid Dialog parameter type: *param-value*

- The dialog parameter must be a valid type for this function, as listed in the chapter *Function Code List*.

Invalid Dialog procedure: *proc-value*

- The procedure specified has not been defined in either the local or global dialog tables.

Invalid Event Key

- Event keys in the dialog must be only valid keys as listed in the chapter *Programming*.

Invalid MOVE operands

- The parameters for the MOVE function in dialog are not compatible types.

Invalid numeric literal

- A numeric literal cannot be used for this parameter.

Maximum decimal length is 9

- The decimal part of a numeric field cannot exceed nine digits.

Maximum field name length is 30

- A field name cannot be longer than thirty characters.

Maximum group occurrence is 9999

- A group cannot repeat more than 9999 times.

No border allowed for this panel

- A minimum panel height of three lines and width of three characters is required for a border.

No group has been defined

- Dialog or panel definition references a group item that has not been defined.

No select bar for group: *group-name*

- The parameter for this function in dialog requires a group with a selection bar.

Numeric field must be 18

- The maximum size of a numeric field, including its decimal part, is 18 digits.

Panel coordinates do not fit on screen

- The panel does not fit on the screen. Change its coordinates.

Panel dimensions invalid: *panel-name*

- This panel will not fit on the screen.

Panel name already exists

- An existing panel in the Data Block cannot be overwritten.

Panel not found: *panel-name*

- The dialog refernces a panel that does not exist.

Procedure not declared - Dialog flushed: *proc-name*

- If a procedure is referenced that does not exist, the whole dialog table is invalid.

Select bar not found: *group-name*

- The parameter for this function in dialog requires a group with a selection bar.

Text will not fit on panel

- The text size is too large for the panel.

The previous dialog line was empty

- A field with a decimal part must have an integer part.

This field has invalid format: *field-name*

- A field must be a valid format as listed in the section *Defining Panel Fields (F3)* in the chapter *Panel Fields*.

This field has invalid position: *field-name*

- A field must be positioned inside a panel.

This field has invalid properties: *field-name*

- This field has been defined with properties that are valid only for an alphabetic or alphanumeric field.

This field has invalid size: *field-name*

- The length of a field on a panel must be greater than zero.
- The length of a field on a panel cannot be greater than its length in the Data Block.

This field has invalid usage: *field-name*

- A field must be set to a usage listed in the section *Defining Panel Fields (F3)* in the chapter *Panel Fields*.

This validation is not possible for this field type

- A particular field type allows only certain validation types. See the section *Define Validation Details (F8)* in the chapter *Data Definition*.

This group has invalid position: *group-name*

- An imported panel group cannot overlap an existing group or field.

22.4.2 Fatal Errors

ACC-EXIT only applicable to Data groups

- ACC-EXIT BAR (or END) can only be specified for data groups.

Error in allocating dynamic memory

- Import was unable to open a heap. Try running Dialog System with more disk space or memory, if possible.

Error in opening file

- The import file could not be accessed.

Field table is full

- There is a maximum of 512 fields allowed in the Data Block.

INTENSITY/BLINK must be off/on

- The INTENSITY and BLINK attributes can be set only to on or off.

Invalid border number

- Valid borders are single, double or none.

Maximum literal value is 80

- There is a maximum of 80 characters in a text literal.

The color is not supported

- Valid colors are BLACK, BLUE, GREEN, CYAN, RED, MAGENTA, BROWN and WHITE.

Unexpected end of import file

- The file contains incomplete information.

23 Tutorials

This chapter contains a set of tutorials, each of which addresses a particular component or function of Dialog System. The tutorials illustrate many of the features of Dialog System, and were, of course, produced by Dialog System.

Before you start these tutorials, you might find it useful to select **F8=tutorial** from the top menu. This gives you an on-line overview of the facilities offered by Dialog System. You can step through the tutorial by pressing **Enter**. (**Escape** exits the tutorial).

The tutorials in this chapter follow on from the Character Mode sample session in the chapter *Sample Session in Character Mode*. You should work through that sample session before you start on the tutorials presented here. You need to work through these tutorials in sequence, because some of them require screensets created in earlier ones.

23.1 Tutorial 1 - Cutting and Pasting Experiment

You can cut and paste across multiple lines containing mixtures of text and data.

Experiment with cutting and pasting using the following steps:

- 1 Start the definition process and load Test1.
- 2 Press **F3=panels** followed by **F3=panel-list**.
A list of panels appears (at present you only have one).
- 3 Select the panel STOCK by pressing **Enter**.

4 Try a variety of ways to cut and paste text and data.

To cut a block:

position the cursor at the top left corner of the block

press **F2=mark/unmark**

move the cursor to the bottom right corner of the block

press **F7=cut-to-block**

To paste a block:

move the block to the required position using cursor keys

press **F2=paste**

You are prevented from doing two things:

- You cannot cut a block that only contains part of a field.
- You cannot paste a block on top of another field.

When you are moving a block around you can see that the menu enables you to use the copy-from-block function.

5 Try selecting **F3=copy-from-block**.

A copy of the block you cut is placed at the desired position and you still have the cut block to move elsewhere. This means that you can place multiple copies at various parts of the panel. You can see that you can copy not only text, but also fields, together with all the field details, in this way.

6 After you mark and cut or copy a block, select **F4=stack-block**.

You return to the panel painting menu without pasting or copying the block. Instead, the block you cut has been placed in a stack (maximum of 16). You can pop the block from the stack by selecting **F9=restore-block**.

When you pop a block, it acts as if it had just been cut or copied, and you can move it to the required location and paste it. If necessary, you can push it back onto the stack instead.

7 Try stacking several blocks, then try restoring them.

The only block function you have not yet used is the **F8=copy-to-block** function on the panel menu.

8 Try marking an area, then selecting **F8=copy-to-block**.

You can see that this function behaves in the same way as the cut except that a copy of the block is left in the original position.

- 9 Make yourself familiar with all these block functions because you will use them a lot.

It should be obvious by now that you do not need to make the original panel a perfect design right from the start. You only need to get the text and the data fields onto the panel, then you can cut, copy, and paste blocks later to improve the layout.

This is the end of Tutorial 1.

23.2 Tutorial 2 - Groups and Scrolling

In this tutorial you will perform the following actions:

- Define a scrolling group.
 - Display a new panel.
- 1 Start the definition process again and load in your saved screenset Test1.
 - 2 Save this screenset immediately as Test2. Press **Alt+F4**, change **test1.s** to **test2.s**, then press **Enter**.

The next steps describe how to modify Test2.

The first thing to do is to modify the Data Block. For this example, we have decided that our product has a number of different selling prices. In fact, there are a maximum of 50 different selling prices for each product. For each of these prices, there must be a description.

- 3 Select **F2=data-definition** from the main menu, then **F2=data-fields**. Enter two new data items:
 - S-PRICE**, numeric (9), 3 Digits + 2 decimal places (3.2);
 - S-DESC**, alphanumeric (X), 20 characters (20).
- 4 Move the selection bar to S-PRICE and press **F6=define group**.
- 5 Enter the number of repeats as **50** and press <down-arrow>.
- 6 Watch what happens to the number of repeats in S-DESC and the brackets to the left of the two data names.

- 7 Press **Enter** to finish the definition of the group (see Figure 23-1).

Figure 23-1. Defining the Group

Field	Fnt	Int. Dc	Comp	Rpts	Val
P-CODE	X	6.0		0	
P-NAME	X	15.0		0	
P-COST	9	3.2		0	
P-QUANT	S	4.0		0	
S-PRICE	9	3.2		50	
S-DESC	X	20.0		50	

Data-Definition
 F1=help F2=amend-grp-size F3=ins-field F4=del-field F5=amend-grp-rpts F6=def-grp
 F7=undef-grp F8=validation F9=err-msg-flt def/undef ← ↑ PgUp PgDn Escape

- 8 Press **Escape** to exit data fields and **Escape** again to get back to the main menu.
- 9 Select **F3=panels** and **F3=panel-list**.
- 10 Select **F3=show-panel**.
- 11 Press **Escape** to get back to the panels menu again.
- You should now have the cursor in the top left hand corner of the screen and the STOCK panel visible on the screen.
- 12 Create a new panel with top left hand corner at row 8, col 11 and bottom right hand corner at row 14, col 69.
- 13 Name this panel **SELLING**.
- 14 Paint this panel with a suitable background color.
- 15 Inside this panel, move the cursor to row 2, col 15 and type **Selling Price**.
- 16 Type **Description** at row 2, col 32.
- 17 Select **F3=field** and place the S-PRICE field at row 3, col 17 and the S-DESC field at row 3, col 32.

18 Escape from field definition.

You have now placed the group fields on the screen. However, you have only placed one occurrence of them on the screen and there are 50 in the Data Block. There is, in fact, only room for four occurrences on this screen.

19 Place the cursor on the first character of the first of these fields and select **F2=mark**.**20** Move the cursor to the last character of the second field and press **F4=group**.

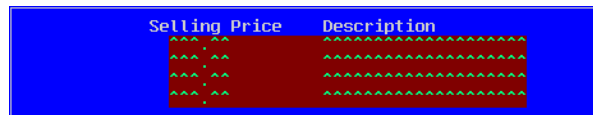
This defines the marked area, including the fields, as a group. You are prompted for a name for the group.

21 Call this group S-DETS.

The menu now enables you to add occurrences to the group by pressing the down cursor.

22 Press <down-arrow> three times so that you have four occurrences visible on the panel, as shown in Figure 23-2.

Figure 23-2. Four Occurrences on the Panel

**23** Press **Enter** to exit the group menu.

If you try to run the screenset, it would run in the same way as the original screenset (TEST1) because we have not yet defined any dialog that will enable us to use this second panel.

24 Return to the panels menu and select the panel painting menu for the original panel (STOCK) by selecting **F3=panel-list** and pressing **Enter**.

We need to choose a keystroke to use on this panel that will cause the second panel (SELLING) to be displayed at run time. Let's use the **F10** key. You can put some text on the panel as a reminder for this key.

- 25 Type the text **F10=selling details** at the foot of the panel STOCK, just above the border.

Note: If you paint the text on the border, the text will not appear at run time, because the border takes precedence. (You can remove the border if you wish - see later).

Now you can set up the dialog that enables the **F10** key to display the second panel. Previously you have defined global dialog that operated across the entire screenset. Now you just wish to define dialog that is local to the current panel.

- 1 Select **Alt+F2=local-dialog**.

This invokes the dialog menu that you have seen before. The keystroke we wish to define is **F10**.

- 2 Enter **F10** in the **Key** column.

The action we want is that when **F10** is pressed, the second panel is activated. We can decide how to return from the second panel later. However, when we return from the second panel, we want to return to the STOCK panel in the place we left off. To do this, associate the following dialog functions with the **F10** key:

Function PUSH with parameter \$NULL
 Function GOP with parameter SELLING

The two functions (PUSH and GOP) each require one parameter. Most functions you can use require between zero and three parameters.

For this simple example, the single parameter for the PUSH function is a null parameter, which is named \$NULL. Later examples show the PUSH function with a parameter that is not null.

For the GOP function, the single parameter is the name of a panel.

The first function in the dialog pushes the current state of the screen onto a stack, which you can pop from the stack later. The second function means Go To the Panel named SELLING.

- 3 Enter these values and exit dialog definition.

- 4 Select the SELLING panel and select local dialog definition in that panel.

Let's decide to use **Escape** to go back to the previous panel.

- 5 Define ESC to have the function POP.

There is already a function defined for the **Escape** key in the global dialog. This local version, however, overrides that definition for this panel only. (You can select to have global overriding local, or the opposite, from the main ALT menu using the **F5=screenset-switches** option).

- 6 Save this screenset and select **F6=run** from the main menu.

- 7 Press **Enter** in the trap.

- 8 Select **F10** and try to enter data into the group.

You will find that you can enter data only into the first occurrence, so we need to rectify this.

- 9 Go back into definition and select the panel SELLING.

- 10 Move the cursor over the group and press **F4=group**.

To accept data into more than just the first occurrence in a group, you need to define a selection bar for this group.

- 11 Select **F5=select-bar**.

This enables you to paint the attributes for a selection bar on the top of the first occurrence of the group. You can use **F6** to roll around the current attributes and **F5** to paint the selected attribute.

- 12 Paint a suitable selection bar.

If you do not like any of the attributes, you need to exit group definition to set up some alternative attributes.

- 13 Press **Enter** to exit from select bar definition.

- 14 Press **Enter** to exit from group definition.

When you exit select bar definition and group definition, the select bar attributes apparently disappear. They have not actually disappeared, which you can confirm by going back into select bar definition again.

At run time, all data accepted into this group is accepted at the position of the selection bar. Therefore we need to define in the local dialog how to move the selection bar. An obvious choice is to allow the cursor down key to move the selection bar down and the cursor up key to move the selection bar up.

- 15 Go into local dialog definition and define key CURU (CURsor Up).
- 16 Apply function PBUP (Position Bar UP), with the value S-DETS for parameter one (group name - there could be more than one group on the panel) and the value 0001 for parameter two.
- 17 This means that, at run time, the cursor up key will cause the selection bar to be positioned up by one line within the S-DETS group.
- 18 Define key CURD (CURsor Down).
- 19 Press **Escape**.
- 20 Apply function PBDN (Position Bar Down), with the value S-DETS for parameter one (group name), and value 0001 for parameter two.
- 21 Escape from local dialog and run the screenset selecting **F6=run**. This saves the screenset first.

You will see that you can enter data into all four entries in the screen group. Better than that, you can enter data into all 50 entries in the data group. The group automatically scrolls when you reach the top and bottom of the list.

This is the end of Tutorial 2.

23.3 Tutorial 3 - Defining Validations

In this tutorial you will perform the following actions:

- Apply validation to data fields
 - Define validation messages
- 1 Start the definition process again and load in your saved screenset Test2.

- Save this screenset immediately as Test3 so that we can modify Test3.

The following table shows the fields in the Data Block for Test3 and the validation criteria we wish to apply to each field:

Data Block Field	Validation Criteria
P-CODE	must not be empty must be either "AAAAAA", "BBBBBB", "CCCCCC", or greater than or equal to "ZAAAAA"
P-NAME	must not be empty
P-COST	must be greater than 1.50
P-QUANT	must be in the range 10 to 400 inclusive
S-PRICE	no validation
S-DESC	no validation

- Enter data definition and place the selection bar on P-CODE and select **F8=validation**.

You can then select the function to define range/table and null validation. The status line (see Figure 23-3) shows which validation types have already been defined.

Figure 23-3. Validation Selected

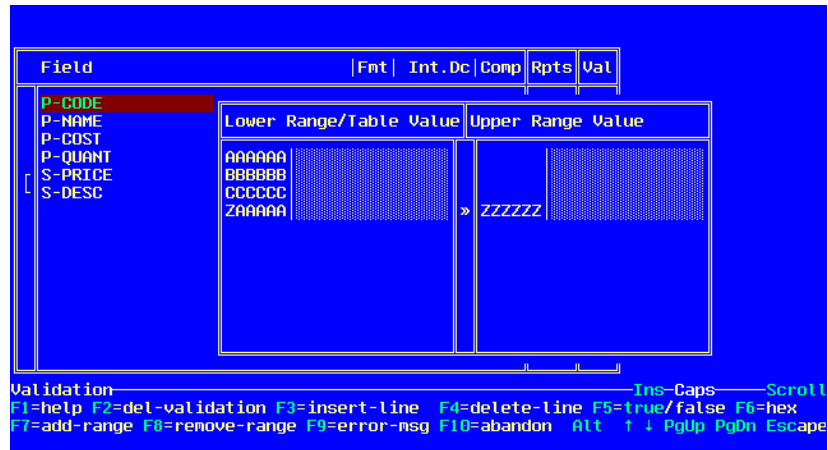
Field	Fmt	Int. Dc	Comp	Rpts	Val
P-CODE	X	6.0		0	
P-NAME	X	15.0		0	
P-COST	9	3.2		0	
P-QUANT	S	4.0		0	
S-PRICE	9	3.2		50	
S-DESC	X	20.0		50	

Validation
 F1=help F2=delete-all-validations F3=range/table
 F7=user-validation
 Ins-Caps Scroll
 F6=null
 Escape

4 Select range/table validation by pressing **F3**.

The panel shown in Figure 23-4 appears. You can see in this panel that a number of range and table validation values have been entered.

Figure 23-4. Range/Table Validation Screen



You can enter a mixture of range and table validations on this panel.

5 Enter the valid values for P-CODE as defined above.

Enter the upper range value by selecting **F7=add-range**.

You will see that the toggle **F5=true/false** is set to true. This is what we want. You can test for the values being false if you wish.

This particular field is obviously required to be in upper case only. The next three steps show how to do this.

6 Escape from validation, escape from data definition and select the panel STOCK.

7 Position the cursor on the field P-CODE and select **F3=field**.

8 Select **F5=amend-field** and select *UPPER* in the properties entry.

This forces upper case to be entered at run time.

- 9 Press **Enter** to complete the field definition.
- 10 Press **Escape** to go back to the top menu. Go back into data definition and select validation for the field P-CODE.
- 11 Select **F6=null** and select the **F2=disallow-null** toggle.
- 12 Escape from validations and enter validations for P-NAME.
- 13 Select the required validations, then do the same for P-COST and P-QUANT.

When you return to the data definition menu, you will see that each of these fields have now got a V beside them.

- 14 Save the screenset and run it. Try entering values in the validated fields.

You will see that two things happen when you put an invalid value in any of the validated fields.

- The attribute of the field changes
- The cursor remains on the offending field

We can control the error attribute. We can also ensure that a suitable error message appears on the panel.

You will also see that if you press **F10** when the current field is invalid, you cannot go to the other panel. There is a way to change this, which is described in a later tutorial.

- 15 Go back into definition and select the STOCK panel.
- 16 Select **Alt+F3=attribute-palette** and select **F8=run-time-list**.

One of the entries on this menu is **F6(error)**. This is the attribute that is used on the current panel when a validation error occurs.

- 17 Change the error attribute to one that you want, by moving along the attribute list and pressing **F6** on the required attribute.

You have set up the error attribute for the STOCK panel. This attribute can be different for every panel.

We now need to define some error messages to appear whenever a validation error occurs. Error messages are held in a relative file at run time and are accessed by their relative key automatically. This file can be common to many applications and does not need to relate to just one screenset.

To define the use of such a file in relation to a particular screenset we need to define:

- The name of the file.
 - The contents of the file (the error message numbers and text).
 - The error message numbers to be used by each validation.
- 18** Go back to the main menu and select **F2=data-definition** and **F3=error-messages**. This invokes a menu where you can select **F2=select-error-message-file-name**.
- 19** Press **F2** and enter the name **test.err** as the name of the file. Press **Enter**.

You have defined the name of the file.

- 20** Select **F3=define-error-messages**, then define:

001 Field must not be empty
 002 Only A..A, B..B, C..C, and Z.. allowed
 003 Cost must be more than 1.50
 004 Quantity in range 10 - 400

- 21** Press **Escape** twice.

We must now allocate each message to the particular validation(s) to which it refers.

- 22** Enter data fields again and go into validation for each of the validated fields.

When you select a validation type such as range/table or null, there is a menu entry **F9=error-message**. Select this option. You can either type in the error message number, or pick the number from the error message file by moving the selection bar to the appropriate message and pressing the space key.

There are two final things to do before you can use any of the above effectively; create a field in the Data Block that the error message can be moved to, and position the field on the appropriate panel.

The error message field appears to be like any other field in the Data Block. However, when we define this field as the error message field, the run-time system treats it differently. If the run-time system finds an error, it fills this field with the correct value

from the error message file, and when the error is resolved, it replaces the value with spaces.

- 23** Select data definition and define a field at the end of the data fields. Call this field P-ERROR and make it alphanumeric of length 45.

- 24** Define this field as the error field by pressing **F9=err-msg-flid-def/undef**.

The symbol E appears beside the field. You can define only one field in this way.

We now wish to ensure this field is visible on the STOCK panel.

- 25** Select the stock panel and select **F3=field**.

- 26** Place the field P-ERROR at an appropriate point on the panel and make sure its usage is OUTPUT.

You should also ensure that this field has the same background attribute as the rest of the panel, so that it is not visible when it is empty.

You might wish to have the error field in a panel of its own that appears only when a validation error occurs. This is quite straightforward and is described in a later example.

- 27** Run the screenset.

You will see that all the validations you defined are handled and all the messages are correctly set up and removed.

This is the end of Tutorial 3.

23.4 Tutorial 4 - Documenting the Screenset Definition

In this tutorial you will produce a text listing of the screenset.

- 1** Load the Test3 screenset.

- 2 From the main menu select **F7=print**.

This displays a list of print options that you can select and de-select using the space bar. We want all the options, so leave this list with every item marked.

- 3 Press **Enter=print-screen-set**.

This produces a detailed listing of your screenset into a file called **test3.lis**. If you print a screenset that does not have a name because you have not yet saved it, the list file is written to **ds.lis**.

- 4 Load in the previous two screensets Test1 and Test2 and produce listings of them.
- 5 Print off the listings on your printer and review them.

Whenever you encounter a new screenset from now on, you might find it useful to print off the details of the screenset.

If your printer cannot handle some of the graphics characters you have used, you can configure these to your printer by altering the file **dsdef.cfg**. This is described in detail in the chapter *Setting Up the Configuration File*.

This is the end of Tutorial 4.

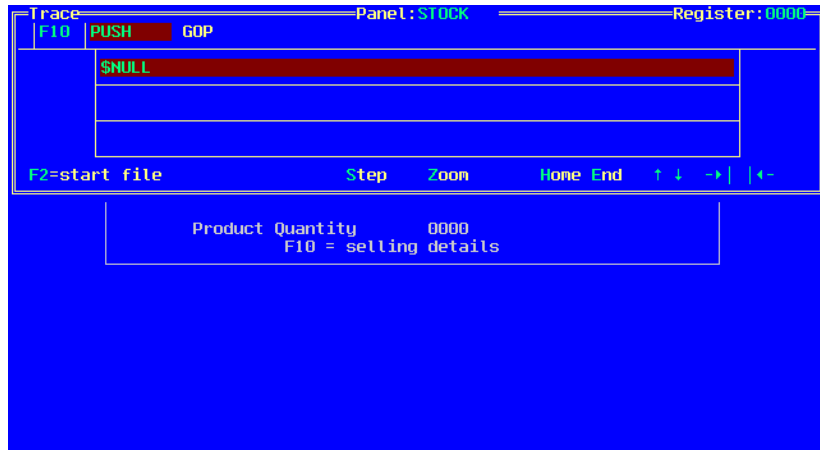
23.5 Tutorial 5 - Using the Trace and Trap

In this tutorial you will learn how to trace the dialog using the trap.

- 1 Load your latest screenset (Test3), select **F6=run** and stop at the trap screen.
There is a menu entry **F2=trace on/off**.
- 2 Select trace on, then press **Enter**.
- 3 Type "**AAAAAA**" into the first field, type "**xxx**" , into the second field, and press **BackTab**.

A separate panel appears, shown in Figure 23-5, which contains the following information:

Figure 23-5. Trace Panel



- Panel: The name of the current active panel (in our example, STOCK).
- Register: An internal register that we will use later for some dialog functions. Its current value is 0.
- Dialog line: The current dialog line that is active. It displays the name of the key that activated the dialog, and the entries in the dialog line with the parameters for the current function.

You can move the trace panel anywhere on the screen to a more suitable position if required, using the cursor up and down keys and the home and end keys.

The following additional keys are now valid:

- S single step through each dialog function.
- Z zoom through all dialog functions until the next return to the trap.

F2=start file causes the trace to be written to a file (until **F2** is pressed again). The trace file is the screenset name with extension **.trc**. This file is always extended so you can write several traces successively.

- 4 Press **S=Step**, then **F10** and observe the dialog being traced.
- 5 Experiment with using trace.
- 6 Try using trace on your earlier screensets.

One other useful function on the trap screen is **F4=initialize-data-fields**.

This enables you to clear all the data fields in the way that a calling program might.

This is the end of Tutorial 5.

23.6 Tutorial 6 - Virtual Text

In this tutorial you will perform the following action:

- Define a virtual text group.
- 1 Start the definition process again and load in your saved screenset Test3.
 - 2 Save this screenset immediately as Test4.

Now we will define a help panel that will appear whenever **F1** is pressed. We will make this help panel quite small and fit more text into it than it is apparently capable of holding (virtual text).

- 3 From panel definition, start by marking an area on the screen with top left corner at row 1, col 57 and bottom right corner at row 13, col 80.
- 4 Define this area as a panel called HELP.
- 5 Choose a suitable set of colors for this panel.
- 6 Place a double line border around this panel by selecting **F10=panel-maint** then **F4=double border**, and pressing **Escape**.

- 7 From the panel main menu, type the letters **HELP** in the middle of the second line of the help panel.
- 8 Move the cursor to row 3, col 2 of the panel and mark an area down to row 12, col 23 (that is, everything inside the panel except the borders and heading).
- 9 Press **F4=group** and name the group HELP-TXT.

The following menu appears (see Figure 23-6):

Figure 23-6. Fixed Text Group Menu

```
Fixed-Text-Group-----HELP-TXT-----Group-----Row:12-Col:23-Ins-Caps-----Scroll
F1=help F2=define-virtual-text F5=select-bar F8=group-name Escape
F9=group-size-maintenance
```

- 10 Press **F2=define-virtual-text**.

The group is now surrounded with a highlighted border and the following menu appears (see Figure 23-7):

Figure 23-7. Virtual Text Definition Menu

```
Virtual-Text-Definition-----Position:001-of-011-----Ins-Caps-----Scroll
F1=help F3=insert-line F4=delete-line F7=restore-line F8=duplicate-line
F9=delete-virtual-text F10=abandon PgUp PgDn Escape
```

You can now enter the text you would like on the help screen. There can, of course, be more text than can fit on the visible panel.

Note: There is a counter on the status line, which tells you how many lines there are and which line you are currently on.

- 11 Enter more help lines than can fit on the screen.
- 12 Try the various insert, delete, duplicate and restore functions on the menu and observe the line counters on the status line.

- 13 When you are happy with the virtual text, press **Escape** until you get back to the main menu.

We now need to define how to activate this help panel.

- 14 Go into global dialog definition and define the F1 key to push the screen, then go to the help panel (PUSH function, then GOP HELP).

We now need to define the dialog for the panel HELP.

- 15 Select the HELP panel and go into local dialog definition.

We need to be able to move up and down the virtual text. We have seen so far how to move a selection bar up and down a data group. You can also do this with a virtual text group.

The functions we are about to use are concerned with moving around the underlying group. This applies to data groups, virtual text groups and virtual attribute groups (as we shall see later).

We can think of the virtual text as hidden data that we cannot alter at run time.

- 16 Define the following entries in the dialog table:

Key	Function	Parameters	Action
CURU	PDUP	HELP-TXT 0001	Positions the data up by one line
CURD	PDDN	HELP-TXT 0001	Positions the data down by one line
PGUP	PDUP	HELP-TXT 0009	Positions the data up by nine lines
PGDN	PDDN	HELP-TXT 0009	Positions the data down by nine lines

We also need to define how to exit from help.

We could choose that any key other than the five already listed causes a pop back to the screen that invoked the help panel. To do this, you define this as key ANYO (ANY Other) with function POP.

- 17 Run this screenset. You might also try to trace the screenset while it runs.

You will see that this is not behaving quite as you would like. If you press **F1**, you get an error message on the field you are positioned

on. This is because the field is being validated and the **F1** keystroke produces a validation error.

You must suppress validation for each field when the **F1** key is pressed, by defining the **F1** key as an event key. The **Escape** key in the global dialog was defined as an event key. That is why validation is not carried out when you press **Escape**.

18 Go back into definition and go into global dialog definition.

19 Move the selection bar to the definition of the **F1** key dialog entry and press **Alt+F2=event-key-definition**.

This places a > symbol beside the **F1**, indicating that it is an event key.

You have just defined the **F1** key in global dialog. More often, you might wish to define the help key locally for each panel, because you might require a different help panel for each of your data entry panels.

20 Save the screenset and run it again.

There are two further problems you might notice.

When you press the **Escape** key on the help screen, the global definition of **Escape** key is used so there is a return to the calling program. You almost certainly do not want this.

Also, if you press **F1** inside the help panel, nothing appears to happen. Think about this, and if you are still not sure what is happening, trace the screenset.

21 Go back into definition and define the **Escape** key inside the help screen to do a POP.

22 Fix the problem you found when you used the **F1** key from the help panel by doing a POP when **F1** is pressed.

23 Run the screenset again.

Observe how the double push and pop going to the screen **SELLING**, then to the screen **HELP**, then back again, is working. There is a maximum of 16 pushes.

This is the end of tutorial 6.

23.7 Tutorial 7 - Virtual Attributes

In this tutorial you will perform the following actions:

- Define a virtual attribute group.
 - Use the virtual attribute group as a menu.
- 1 Start the definition process again and load in your saved screenset Test4.
 - 2 Save this screenset immediately as Test5.

Virtual attribute groups behave in a similar way to virtual text groups. Instead of entering text against each occurrence in the group, you enter attributes. The two major differences are that a virtual attribute group cannot have a selection bar and can only be one line high.

- 3 Select data definition and define the new data items shown in Figure 23-8:

Figure 23-8. Define New Data Items

Field	Fnt	Int. Dc	Comp	Rpts	Val
P-CODE	X	6.0		0	V
P-NAME	X	15.0		0	V
P-COST	9	3.2		0	V
P-QUANT	S	4.0		0	V
S-PRICE	9	3.2		50	
S-DESC	X	20.0		50	
P-ERROR	X	45.0		0	E
CUSTOMER	X	10.0		0	
CREDIT	9	5.0		0	
SALESMAN	X	10.0		0	
COMMS	9	4.2		0	

Data-Definition -Ins-Caps- Scroll
 F1=help F2=amend-grp-size F3=ins-field F4=del-field F5=amend-grp-rpts F6=def-grp
 F7=undef-grp F8=validation F9=err-nsg-flt def/undef ← ↑ ↓ PgUp PgDn Escape

- 4 Paint two new panels called CUSTOMER and SALESMAN. Place the appropriate fields on these panels as input fields.

- 5 Define the dialog on both these new panels so that **Escape** pops the panel (POP function).
- 6 Alter the dialog on the STOCK panel so that **Escape** pops the panel (POP function).

Now we will produce a menu so that we can select from one of the three panels STOCK, CUSTOMER and SALESMAN.

- 7 Create a new panel called MENU that is just one line deep and at the top of the screen from column 1 to column 80.
A border is not drawn because the panel is only one line high. It might be useful to change the default attribute of the panel so you can see it more clearly.
- 8 Type the text shown in Figure 23-9 into this panel.

Figure 23-9. Column Headings

```
Stock Details Customer Details Salesman Details Exit
```

- 9 Move the cursor to the first character of the word Stock and mark an area up to the last character of the words Exit.
- 10 Select **F4=group**.

A prompt appears asking you to decide whether the group is an attribute group or a text group (see Figure 23-10).

Figure 23-10. Group Selection Menu

```
Group-selection Group Row:01-Col:58-Ins-Caps Scroll
F1=help F2=attribute-group F3=text-group Escape
```

- 11 Select **F2=attribute-group**, then name the group MENU.

The menu changes to the Virtual Attribute Group menu (see Figure 23-11).

Figure 23-11. Virtual Attribute Group Menu

```
Virtual-attr-group      MENU      Group      Row:01-Col:58-Ins-Caps-Scroll
F1=help F2=define-virtual-attributes F8=group-name F9=group-size-maintenance
                                         Escape
```

12 Select F2=define-virtual-attributes.

The menu changes to the Attribute Definition menu (see Figure 22-12).

Figure 23-12. Dsdemo1; Data Block

```
Attribute-Defn-Attribute      Position-001-of-002-Ins-Caps-Scroll
F1=help F3=insert-line F4=delete-line F5=paint-attribute F6=attribute-roll
F7=restore-line F8=duplicate-line F9=delete-virtual-attrs F10=abandon Escape
```

You are now ready to paint the virtual attributes in a similar way as you did inside selection bar definition.

You want to produce four attribute entries where:

- entry 1 highlights "Stock Details"
- entry 2 highlights "Customer Details"
- entry 3 highlights "Salesman Details"
- entry 4 highlights "Exit"

13 Experiment with the virtual attribute features until you have just four virtual attribute entries which highlight the four pieces of text as you require.

Remember that to get to the third entry from the second, you need to press **Enter** rather than cursor down, because the default is to begin with just two entries in the list.

14 Press Escape twice to get back to the panel menu.

15 Enter the following dialog for the menu panel:

Key	Function	Parameters
CURR	PDDN	MENU 0001
CURL	PDUP	MENU 0001
P021	PUSH	\$NULL
	GOP	STOCK
P022	PUSH	\$NULL
	GOP	CUSTOMER
P023	PUSH	\$NULL
	GOP	SALESMAN
P024	RETC	
CR	BPD	MENU P021

The CURR dialog line refers to the cursor right key. The number 0001 is the number of lines to position the data down.

The CURL dialog is similar, but for the cursor left key.

The next four entries are "procedure" lines. The key entry in these lines are not keystrokes but procedure entries. The procedure numbers are P021, P022, P023 and P024. You can see that these procedures use the PUSH and GOP functions (except P024, which is a return to the calling program). They are invoked from the last dialog entry line.

The last line of dialog refers to the **Enter** key. When **Enter** is pressed, the function is BPD (Branch to procedure depending on data position). This causes procedure P021 to be executed if the attribute data of the MENU group is in position 1, P022 if the attribute in position 2, and so on.

Procedures are named P000 through P255. You can also define procedures in global dialog. If you use the same numbering, the local procedures take default over global procedures unless you have set the reverse as default on the **Alt+F5=screenset-switches** menu from the main menu.

If you use procedure P000, it has a special significance. This is the panel initialization procedure, which is executed when that panel is activated, before any keystrokes. P000 in global dialog is the screenset initialization procedure.

You need one more change before you can run this screenset. You must define that the first panel that is entered is the menu panel.

- 16 Escape back to the panels menu and select **F3=panel-list**.
- 17 Move the selection bar to highlight the "MENU" panel name.
- 18 Press **F5=select-first-panel**.

This selects the MENU panel as the first panel. You can reorder this list, if you wish, using the **F9=relocate-panel** function.

- 19 Save the screenset and run it, tracing if necessary.

You will see that cursor right and cursor left from the menu apparently move the highlighted bar left and right along the line. One thing you might wish to alter is that if you attempt to move beyond the fourth entry, the first entry is highlighted, and vice versa. This gives the effect of rolling round the top of the list.

- 20 Go back into definition and dialog entry for the MENU panel.
- 21 Define two more procedures.

Key	Function	Parameters
P001	SD	MENU 0001
P002	SD	MENU 0004

The first of these procedures, if activated, causes the data to be set "SD"=Set Data) at position 1 in the MENU group, and has the effect of displaying the first virtual attribute entry (that is, position 1).

The second procedure sets the data at position 4.

To activate either of these procedures, we are using the concept of an internal exception condition. For example, if you try to move down the virtual attribute group beyond the last entry, an internal exception condition is created. You might have already observed that this normally causes a beep, but you can also intercept this exception.

The dialog entry for CURR consists of moving one line down the data. Place the following before the PDDN, using **F3=insert-function**:

Key	Function	Parameters
CURR	BPE	P001

This causes any exception on the PDDN function to branch to procedure P001.

- 22 Try the above and see if you can work out what is necessary for the CURL key.
- 23 Save the screenset and run it. Tracing will help to clarify what is happening.

This is the end of Tutorial 7.

The remainder of the tutorials are based upon sample screensets and programs that are provided on your issue disks.

23.8 Tutorial 8 - Changing Fields in Groups

In this tutorial you will perform the following actions:

- Insert into a screen group.
- Delete from a screen group.
- Specify a specific field for cursor movement.

The file you need for this tutorial is **dsdemo1.s**

- 1 Go into definition mode and load the screenset Dsdemo1.
- 2 Look at the Data Block (see Figure 23-13).

Figure 23-13. Dsdemo1; Panel

Field	Fnt	Int. Dc	Comp	Rpts	Val
SUPPLIER	X	20.0		0	
DATE	9	6.0		0	U
PRODUCT	X	10.0		25	
QUANTITY	9	3.0		25	
COST	9	4.2		25	
ERR-FLD	X	35.0		0	E

Data-Definition Ins-Caps Scroll
 F1=help F2=amend-grp-size F3=ins-field F4=del-field F5=amend-grp-rpts F6=def-grp
 F7=undef-grp F8=validation F9=err-msg-fld def/undef ← ↑ ↓ PgUp PgDn Escape

You can see that there is a group item with three fields. Also the date field is validated. The validation message is in the file **dsdemo.err**.

- 3 Look at the panel in this screenshot (see Figure 23-14). You can see that there is just one panel.

Figure 23-14. Dsdemo1: Local Dialog

STOCK ENTRY SCREEN

Supplier Name [^^^^^^^^^^^^^^^^^^^^] Date of delivery [^^/^^/^^]

Product Name	Quantity	Cost Price
^^^^^^^^	^^^	^^^.
^^^^^^^^	^^^	^^^.
^^^^^^^^	^^^	^^^.
^^^^^^^^	^^^	^^^.
^^^^^^^^	^^^	^^^.
^^^^^^^^	^^^	^^^.
^^^^^^^^	^^^	^^^.
^^^^^^^^	^^^	^^^.
^^^^^^^^	^^^	^^^.
^^^^^^^^	^^^	^^^.

F3=insert line F4=delete line Escape=exit

- 4 Look at the local dialog for this panel (see Figure 23-15), and the parameters for each function (see Figure 23-16).

Figure 23-15. Dsdemo1: Local Dialog Function Parameters

E Key	Functions
F3	IBP GOF
F4	DBP SAS GOF
CR	PBDN GOF
CURU	PBUP
CURD	PBDN
TAB	SKNF

Figure 23-16. Dsdemo2; Data Block

Function	Parameter 1	Parameter 2	Parameter 3
DBP	P-GROUP >	>	>
SAS	P-GROUP >	0025 >	>
GOF	PRODUCT > (0000 >)	>	>

There are three new functions you have not yet used:

- IBP Insert at bar position. This inserts a blank line in the group at the selection bar position.
- DBP Delete at bar position. This deletes the line at the selection bar position.
- GOF Go to field. This forces the cursor to go to the specified field. The 0000 in the PRODUCT entry indicates the current bar line. A non zero value in here would indicate a specific array element.

- 5 Run this screenset and observe its behavior, using trace if necessary.
This is the end of Tutorial 8.

23.9 Tutorial 9 - Input Attributes and Procedures

In this tutorial you will perform the following actions:

- Use input attributes.
- Use a procedure in the dialog.

The files you need for this tutorial are **dsdemo2.s** and **dsdemo.err**.

- 1 Go into definition mode and load the screenset Dsdemo2 (see Figure 23-17).

Figure 23-17. Dsdemo2: definition mode

Field	Fnt	Int . Dc	Comp	Rpts	Val
P-NAME	X	10.0		20	
P-FREE-Q	9	4.0		20	
P-ORD-Q	9	4.0		20	
P-S-PRCE	9	4.2		20	
P-C-PRCE	9	4.2		20	
S-NAME	X	20.0		0	
S-TELE	9	12.0		0	
S-ADDR	X	20.0		5	
C-NAME	X	20.0		0	
C-ADDR	X	20.0		5	
C-TELE	9	12.0		0	
C-CREDIT	9	5.0		0	U
O-NO	9	5.0		0	
O-DATE	X	7.0		0	U
I-NO	9	6.0		0	

Data-Definition Ins-Caps Scroll
 F1=help F2=amend-grp-size F3=ins-field F4=del-field F5=amend-grp-rpts F6=def-grp
 F7=undef-grp F8=validation F9=err-msg-fld def/undef ← ↑ ↓ PgUp PgDn Escape

- 2 Look at the Data Block.

You can see that there are some groups in this Data Block and some validations. The validations have been assigned error message numbers, but these are not shown on the data field panel. To see them, you can select **F8=validation** on the appropriate fields.

- 3 Look at the panels in this screenset. There are a number of panels in use.

- 4 Look at the dialogs associated with each of these panels. You will see a new function:

XP execute procedure - forces a procedure to execute.

- 5 Press **F8** from the attribute palette menu to look at the attribute run-time list for one of the panels.

The *input-attribute* is turned *on* for some of the panels but not for others. If you turn the input attribute on, it forces the chosen input attribute to be used when there is input to a field. You will see the behavior when you run the screenset.

- 6 Look at one of the screen groups. The group-accept-exit flag is set to "end" rather than "bar".

This affects how a user can move from field to field in a group accepting input at run time. If this flag is set to "bar", any attempt to move off the last field of the current group occurrence moves on to the first field outside the group. If the flag is set to "end", a move off the last field of an occurrence moves on to the first field of the next occurrence of the group. This continues up to the last field of the last occurrence of the group, that is the end of the group. The cursor then moves to the first field outside the group.

- 7 Run this screenset and observe its behavior, using the trace if necessary.

This is the end of Tutorial 9.

23.10 Tutorial 10 - Using Different Group Types

In this tutorial you will see some examples of:

- Fixed text groups
- Virtual text groups
- Attribute groups

The file you need for this tutorial is **dsdemo3.s**.

There is no Data Block in this screenset. It is purely a demonstration of fixed text groups, virtual text groups and attribute groups. The example it uses is the list of mnemonics in Dialog System and their structure. Because it is an example, the lists are not strictly up to date. You might want to make them up to date as an exercise.

Study the dialog for some of these panels, then run the screenset using trace.

This is the end of Tutorial 10.

23.11 Tutorial 11 - Example of Data Validation

In this tutorial you will see some examples of data validation.

The files you need for this tutorial are **dsdemo4.s** and **dsdemo.err**.

- 1 Go into definition mode and load the screenset Dsdemo4.
- 2 Look at the panel in this screenset (see Figure 23-18). This example is used purely to illustrate various validated fields.

Figure 23-18. Customer Data Block; Part 1

```

This screen illustrates error handling using a separate panel for
error messages and repeating a keystroke back to the original panel

Alphabetic Field [^]      This field must be A, B or C.

Numeric Field [^^^]      This field must be 21, 22 or in the range
                          1000 to 5000.

Numeric Field [^^^]      This field must be non zero and check digitd

Alphabetic Field [^]      This field must be non blank and NOT A, B or C

Escape=Finish  Ctrl/Z=abort

MAIN-SCR      Attribute      Row:01-Col:01-Ins-Caps-Num-Scroll
F1=help F2=mark/unmark F3=field F4=group F5=paint-attribute F6=attribute-roll
F7=cut-to-block F8=copy-to-block F9=restore-block F10=panel-maint Alt Ctl Escape

```

- 3 Go into data validation and study the validations.

Look at the dialog for the main panel. You can see that there is an entry for a "key" called ERR. This is not a key at all, but the name for a reserved procedure that is executed when a validation error occurs. You can see that this procedure pushes the current panel and goes to the error panel. If you look at the dialog for the error panel, you can see that every keystroke causes a POP, but the RPKY causes that keystroke to be repeated when the POP is complete.

- 4 Observe the effect of this by running the screenset through the trap.

This is the end of Tutorial 11.

23.12 Tutorial 12 - The Application Program Interface

In this tutorial you will see a simple update program interfacing to a screenset. This and the next three tutorials use the same program with four different screensets. The screensets will behave in very different ways, but the underlying program in each case is identical.

The files you need for this tutorial are:

customer.s1
customer.cbl
ds-cntrl.mf
customer.cpb (you must generate this file)
customer.err

- 1 Generate the file **customer.cpb**. Go into the definition software and load the screenset **customer.s1**.
- 2 Select **F5=generate**, then press **Enter**.
- 3 Press **F2** to generate the data descriptions.

This generates the **customer.cpb** file for you. You can set various defaults for this generation, which are described in the chapter *Generating the Copyfile*. Do not worry about them here.

The program **customer.cbp** updates an indexed customer file.

- 4 The following listing is the listing for **customer.cbp**, shown so you can study it:

```

$SET ANS85
***** IDENTIFICATION DIVISION *****
  identification division.
  program-id. customer.
***** ENVIRONMENT DIVISION *****
  environment division.

*-----*
  input-output section.
  file-control.
    select customer-file assign "cust.ism"
      organization is indexed
      record key is file-c-code
      access is dynamic.

***** DATA DIVISION *****
  data division.

*-----*
  file section.
  fd customer-file.
  01 customer-record.
    03 file-c-code          pic x(5).
    03 file-c-name         pic x(15).
    03 file-c-addr1        pic x(15).
    03 file-c-addr2        pic x(15).
    03 file-c-addr3        pic x(15).
    03 file-c-addr4        pic x(15).
    03 file-c-limit        pic 9(4) comp.
    03 file-c-area         pic x.
    03 file-c-order.
      78 no-of-orders      value 10.
      05 file-c-order-entry occurs no-of-orders.
        07 file-ord-no     pic 9(6).
        07 file-ord-date   pic 9(6).
        07 file-ord-val    pic 9(4)v99 comp.
        07 file-pay-val    pic 9(4)v99 comp.
*-----*
  working-storage section.
*   copy "ds-cntrl.mf".

```

```

*****
*   Control Block (Micro Focus Constants)
*****

01 DS-CONTROL-BLOCK.
  03 DS-VERSION-NUMBERS.
    05 DS-DATA-BLOCK-VERSION-NO   PIC 9(4).
    05 DS-VERSION-NO              PIC 9.
  03 DS-INPUT-FIELDS.
    05 DS-CONTROL                 PIC X.
      78 DS-CONTINUE              VALUE "C".
      78 DS-NEW-SET               VALUE "N".
      78 DS-LOAD-SYSTEM          VALUE "L".
      78 DS-QUIT-SET             VALUE "Q".
      78 DS-PUSH-SET             VALUE "S".
      78 DS-PATHNAME             VALUE "P".
      78 DS-ERR-FILE-OPEN        VALUE "E".
    05 DS-CONTROL-PARAM          PIC 9(4) COMP.
      78 DS-CONTROL-PARAM-DEFAULT VALUE 0.
      78 DS-SCREEN-NOCLEAR       VALUE 1.
      78 DS-IGNORE-DB-VER-NO    VALUE 2.
      78 DS-CHECK-CTRL-BREAK     VALUE 4.
    05 DS-SET-NAME               PIC X(16).
    05 DS-CLEAR-DIALOG          PIC 9.
    05 DS-PROC-TYPE             PIC X.
    05 DS-PROC-NO               PIC 9(4) COMP.
    05 DS-INPUT-RESERVED        PIC X(12).
  03 DS-OUTPUT-FIELDS.
    05 DS-SYSTEM-ERROR-NO       PIC 9(4) COMP.
      88 DS-NO-ERROR             VALUE 0.
      88 DS-NOT-INITIALISED      VALUE 1.
      88 DS-CANNOT-OPEN-SET      VALUE 2.
      88 DS-ERROR-READING-FILE   VALUE 3.
      88 DS-INVALID-SET         VALUE 4.
      88 DS-CANNOT-CREATE-PANEL  VALUE 5.
      88 DS-DYNAMIC-ERROR        VALUE 6.
      88 DS-INVALID-FUNCTION     VALUE 7.
      88 DS-INVALID-PROC         VALUE 8.
      88 DS-VALIDATION-PROG-ERROR VALUE 9.
      88 DS-DATA-BLOCK-VERNO-ERROR VALUE 10.
      88 DS-PUSH-LIMIT-REACHED   VALUE 11.
      88 DS-ERROR-FILE-MISSING   VALUE 12.
      88 DS-SUBSCRIPT-ERROR      VALUE 13.
      88 DS-PROC-LIMIT-REACHED   VALUE 14.
      88 DS-CTRL-BREAK-PRESSED   VALUE 15.
      88 DS-ERROR-ON-TRACE-FILE  VALUE 16.
    05 DS-VALIDATION-ERROR-NO    PIC 9(4) COMP.
    05 DS-PANEL-NAME            PIC X(8).

```

```

05 DS-RESERVED-1 PIC X(2).
05 DS-FIELD-COUNT PIC 9(4) COMP.
05 DS-FIELD-OCCURRENCE PIC 9(4) COMP.
05 DS-RESERVED-2 PIC X(6).
05 DS-FIELD-NO PIC 9(4) COMP.
05 DS-FIELD-CHANGE PIC 9.
    88 DS-FIELD-CHANGE-TRUE VALUE 1.
05 DS-EXIT-FIELD PIC 9.
    88 DS-EXIT-FIELD-TRUE VALUE 1.
05 DS-FIELD-NAME PIC X(8)
05 DS-OUTPUT-RESERVED PIC X(8).
*****
* End of Control Block
*****

*copy "customer.cpb".

*****
* Data Block for Set CUSTOMER
*****

01 CUSTOMER-DATA-BLOCK-VERSION-NO PIC 9(4) VALUE 36.
01 CUSTOMER-VERSION-NO PIC 9 VALUE 1.

01 CUSTOMER-SET-BUILD-NO PIC 9(4) VALUE 50.

01 CUSTOMER-DATA-BLOCK.
03 CUSTOMER-C-CODE PIC X(5).
03 CUSTOMER-C-NAME PIC X(15).
03 CUSTOMER-C-ADDR1 PIC X(15).
03 CUSTOMER-C-ADDR2 PIC X(15).
03 CUSTOMER-C-ADDR3 PIC X(15).
03 CUSTOMER-C-ADDR4 PIC X(15).
03 CUSTOMER-C-LIMIT PIC S9(4).
03 CUSTOMER-C-AREA PIC X.
03 CUSTOMER-GROUP-001.
    04 CUSTOMER-GROUP-ITEM-001 OCCURS 10.
        05 CUSTOMER-ORD-NO PIC S9(6).
        05 CUSTOMER-ORD-DATE PIC 9(6).
        05 CUSTOMER-ORD-VAL PIC 9(4)V9(2).
        05 CUSTOMER-PAY-VAL PIC S9(4)V9(2).
        05 CUSTOMER-ORD-BAL PIC S9(4)V9(2).
    03 CUSTOMER-C-BAL PIC S9(5)V9(2).
03 CUSTOMER-GROUP-002.
    05 CUSTOMER-DEL-FLG PIC 9.
        88 CUSTOMER-DEL-FLG-TRUE VALUE 1.
    05 CUSTOMER-LOAD-FLG PIC 9.
        88 CUSTOMER-LOAD-FLG-TRUE VALUE 1.

```

```

05 CUSTOMER-SAVE-FLG          PIC 9.
    88 CUSTOMER-SAVE-FLG-TRUE VALUE 1.
05 CUSTOMER-CLR-FLG          PIC 9.
    88 CUSTOMER-CLR-FLG-TRUE VALUE 1.
05 CUSTOMER-EXIT-FLG         PIC 9.
    88 CUSTOMER-EXIT-FLG-TRUE VALUE 1.
03 CUSTOMER-ERR-MSG          PIC X(20).

*****
*   End of Data Block for Set CUSTOMER
*****

*****
*   Field Numbers for Set CUSTOMER
*****

01 CUSTOMER-FIELD-NUMBERS.
    03 CUSTOMER-FLD-NO-C-CODE      PIC 9(4) COMP VALUE 1.
    03 CUSTOMER-FLD-NO-C-NAME      PIC 9(4) COMP VALUE 2.
    03 CUSTOMER-FLD-NO-C-ADDR1     PIC 9(4) COMP VALUE 3.
    03 CUSTOMER-FLD-NO-C-ADDR2     PIC 9(4) COMP VALUE 4.
    03 CUSTOMER-FLD-NO-C-ADDR3     PIC 9(4) COMP VALUE 5.
    03 CUSTOMER-FLD-NO-C-ADDR4     PIC 9(4) COMP VALUE 6.
    03 CUSTOMER-FLD-NO-C-LIMIT     PIC 9(4) COMP VALUE 7.
    03 CUSTOMER-FLD-NO-C-AREA      PIC 9(4) COMP VALUE 8.
    03 CUSTOMER-FLD-NO-ORD-NO      PIC 9(4) COMP VALUE 9.
    03 CUSTOMER-FLD-NO-ORD-DATE    PIC 9(4) COMP VALUE 10.
    03 CUSTOMER-FLD-NO-ORD-VAL     PIC 9(4) COMP VALUE 11.
    03 CUSTOMER-FLD-NO-PAY-VAL     PIC 9(4) COMP VALUE 12.
    03 CUSTOMER-FLD-NO-ORD-BAL     PIC 9(4) COMP VALUE 13.
    03 CUSTOMER-FLD-NO-C-BAL       PIC 9(4) COMP VALUE 14.
    03 CUSTOMER-FLD-NO-DEL-FLG     PIC 9(4) COMP VALUE 15.
    03 CUSTOMER-FLD-NO-LOAD-FLG    PIC 9(4) COMP VALUE 16.
    03 CUSTOMER-FLD-NO-SAVE-FLG    PIC 9(4) COMP VALUE 17.
    03 CUSTOMER-FLD-NO-CLR-FLG     PIC 9(4) COMP VALUE 18.
    03 CUSTOMER-FLD-NO-EXIT-FLG    PIC 9(4) COMP VALUE 19.
    03 CUSTOMER-FLD-NO-ERR-MSG     PIC 9(4) COMP VALUE 20.

*****
*   End of Field Numbers for Set CUSTOMER
*****

78 refresh-text-and-data-proc      value 255.
78 dialog-system                    value "DSRUN".

01 array-ind                         pic 9(4) comp.
01 display-error-no                 pic 9(4).

```

```

procedure division.
*-----*
controlling section.
    perform program-initialize
    perform program-body
        until CUSTOMER-EXIT-FLG-TRUE
    perform program-terminate.
*-----*
program-initialize section.
    initialize DS-CONTROL-BLOCK
    initialize CUSTOMER-DATA-BLOCK
    move CUSTOMER-DATA-BLOCK-VERSION-NO
        to DS-DATA-BLOCK-VERSION-NO
    move CUSTOMER-VERSION-NO to DS-VERSION-NO
    open i-o customer-file
    perform load-screenset.
*-----*
program-body section.
    if DS-EXIT-FIELD-TRUE
        perform derivations
        perform set-up-for-refresh-screen
    end-if
    evaluate true
        when CUSTOMER-DEL-FLG-TRUE
            perform delete-record
        when CUSTOMER-LOAD-FLG-TRUE
            perform load-record
        when CUSTOMER-SAVE-FLG-TRUE
            perform save-record
        when CUSTOMER-CLR-FLG-TRUE
            perform clear-record
    end-evaluate
    perform clear-flags
    perform call-dialog-system.
*-----*
program-terminate section.
    close customer-file
    stop run.
*-----*
delete-record section.
    move CUSTOMER-C-CODE to file-c-code
    delete customer-file
    perform clear-record.
*-----*
load-record section.
    initialize customer-record
    move CUSTOMER-C-CODE to file-c-code

```



```

if file-c-code not = spaces
  read customer-file
  invalid key
  initialize CUSTOMER-DATA-BLOCK
  move file-c-code to CUSTOMER-C-CODE
  not invalid key
  perform fill-screen-from-record
  perform derivations
end-read
else
  initialize CUSTOMER-DATA-BLOCK
end-if
perform set-up-for-refresh-screen.
*-----*
save-record section.
perform fill-record-from-screen
rewrite customer-record
invalid key
write customer-record
end-write
end-rewrite.
*-----*
clear-flags section.
initialize CUSTOMER-GROUP-002.
*-----*
clear-record section.
initialize customer-record
initialize CUSTOMER-DATA-BLOCK
perform set-up-for-refresh-screen.
*-----*
fill-record-from-screen section.
move CUSTOMER-C-CODE to file-c-code
move CUSTOMER-C-NAME to file-c-name
move CUSTOMER-C-ADDR1 to file-c-addr1
move CUSTOMER-C-ADDR2 to file-c-addr2
move CUSTOMER-C-ADDR3 to file-c-addr3
move CUSTOMER-C-ADDR4 to file-c-addr4
move CUSTOMER-C-LIMIT to file-c-limit
move CUSTOMER-C-AREA to file-c-area
perform varying array-ind from 1 by 1
  move CUSTOMER-ORD-NO(array-ind)
    to file-ord-no(array-ind)
  move CUSTOMER-ORD-DATE(array-ind)
    to file-ord-date(array-ind)
  move CUSTOMER-ORD-VAL(array-ind)
    to file-ord-val(array-ind)

```

```

        move CUSTOMER-PAY-VAL(array-ind)
          to file-pay-val(array-ind)
      end-perform.
*-----*
fill-screen-from-record section.
  move file-c-code to CUSTOMER-C-CODE
  move file-c-name to CUSTOMER-C-NAME
  move file-c-addr1 to CUSTOMER-C-ADDR1
  move file-c-addr2 to CUSTOMER-C-ADDR2
  move file-c-addr3 to CUSTOMER-C-ADDR3
  move file-c-addr4 to CUSTOMER-C-ADDR4
  move file-c-limit to CUSTOMER-C-LIMIT
  move file-c-area to CUSTOMER-C-AREA
  perform varying array-ind from 1 by 1
    until array-ind > no-of-orders
    move file-ord-no(array-ind)
      to CUSTOMER-ORD-NO(array-ind)
    move file-ord-date(array-ind)
      to CUSTOMER-ORD-DATE(array-ind)
    move file-ord-val(array-ind)
      to CUSTOMER-ORD-VAL(array-ind)
    move file-pay-val(array-ind) to
      to CUSTOMER-PAY-VAL(array-ind)
  end-perform.
*-----*
set-up-for-refresh-screen section.
  move refresh-text-and-data-proc
    to DS-PROC-NO.
*-----*
derivations section.
  move 0 to CUSTOMER-C-BAL
  perform varying array-ind from 1 by 1
    until array-ind > no-of-orders
    compute CUSTOMER-ORD-BAL(array-ind) =
      CUSTOMER-ORD-VAL(array-ind) -
      CUSTOMER-PAY-VAL(array-ind)
    add CUSTOMER-ORD-BAL(array-ind)
      to CUSTOMER-C-BAL
  end-perform.
*-----*
load-screenset section.
  move DS-NEW-SET to DS-CONTROL
  move "CUSTOMER" to DS-SET-NAME
  perform call-dialog-system.
*-----*
```

```

call-dialog-system section.
  call dialog-system using DS-CONTROL-BLOCK,
                        CUSTOMER-DATA-BLOCK
  if not DS-NO-ERROR
    move DS-SYSTEM-ERROR-NO
      to display-error-no
    display "DS ERROR NO:  "
      display-error-no
    perform program-terminate
  end-if.
***** END OF PROGRAM *****

```

The CUSTOMER-DATA-BLOCK and CUSTOMER-FIELD-NUMBERS are generated by the Dialog System definition software in the file **customer.cpb**. The file **ds-cntrl.mf** is provided with Dialog System.

- 5 Look at the Data Block in the screenset, shown in two parts in the Figures 22-19 and 22-20). You can see that, apart from the obvious fields, there are some fields which appear to be used as flags.

Figure 23-19. Customer Data Block; Part 2

Field	Fmt	Int. Dc	Comp	Rpts	Val
C-CODE	X	5.0		0	
C-NAME	X	15.0		0	
C-ADDR1	X	15.0		0	
C-ADDR2	X	15.0		0	
C-ADDR3	X	15.0		0	
C-ADDR4	X	15.0		0	
C-LIMIT	S	4.0		0	
C-AREA	X	1.0		0	
ORD-NO	S	6.0		10	
ORD-DATE	9	6.0		10	
ORD-VAL	9	4.2		10	
PAY-VAL	S	4.2		10	
ORD-BAL	S	4.2		10	
C-BAL	S	5.2		0	
DEL-FLG	9	1.0		1	

Data-Definition Ins-Caps Scroll
F1=help F2=anend-grp-size F3=ins-field F4=del-field F5=anend-grp-rpts F6=def-grp
F7=undef-grp F8=validation F9=err-msg-fld def/undef ← ↑ ↓ PgUp PgDn Escape

This is exactly what they are. If you look at the PROGRAM-BODY SECTION in the program listing, you can see how the program acts on these flags.

Figure 23-20. Customer Data Block; Part 2 Program Body Section

Field	Fnt	Int .Dc	Comp	Rpts	Val
C-ADDR4	X	15.0		0	
C-LIMIT	S	4.0		0	
C-AREA	X	1.0		0	
ORD-NO	S	6.0		10	
ORD-DATE	9	6.0		10	
ORD-VAL	9	4.2		10	
PAY-VAL	S	4.2		10	
ORD-BAL	S	4.2		10	
C-BAL	S	5.2		0	
DEL-FLG	9	1.0		1	
LOAD-FLG	9	1.0		1	
SAVE-FLG	9	1.0		1	
CLR-FLG	9	1.0		1	
EXIT-FLG	9	1.0		1	
ERR-MSG	X	20.0		0	

Data-Definition Ins-Caps Scroll
 F1=help F2=amend-grp-size F3=ins-field F4=del-field F5=amend-grp-rpts F6=def-grp
 F7=undef-grp F8=validation F9=err-msg-fld def/undef ← ↑ ↓ PgUp PgDn Escape

You can see a SETF function (SET Flag) in the dialog. Its effect is to put the value 1 into the field defined in the parameter.

Apart from the SETF function, you can see the RETC function (RETurn to Calling program). If you look at the first line in the dialog you can see that the F2 key causes the LOAD-FLG to be set.

- 6 Save this screensent as **customer.s** and run the screensent through the trap. Observe its behavior.

The **customer.cpb** program requires a screensent called **customer.s**, which is why you saved the screensent with that name.

You now need to compile your program to test it running. It is assumed that you are using Micro Focus Net Express.

- 7 Go into Net Express and "Check" **customer.cbl**.
- 8 Animate this program and spend some time observing its behavior until you are fully familiar with all the features of the program.

This is the end of Tutorial 12.

23.13 Tutorial 13 - Running the Program with the Trap

In this tutorial you will learn more about the behavior of a screenset and its relation to the calling program. You will also learn how to run a program through the trap.

The files you need for this tutorial are:

customer.s2
customer.cbl
ds-cntrl.mf
customer.cpb
customer.err

The program is the same program that you have just used, but the screenset **customer.s2** is different. Before the program can use this new screenset, you must load it and save it as **customer.s**.

- 1 Look at the local dialog in each of the panels.

Look especially at the dialog in the panel "MENU-SCR". This panel contains a fixed text group that is used as a menu. The menu items can be selected either by typing a key letter or using the selection bar and pressing **Enter**.

- 2 Run this screenset with trace until you fully understand its behavior, and then run the program.

It might be useful to run the program and the trap at the same time. You can do this using the following steps.

- 3 Edit the program **customer.cbl** to change the line:

```
78  DIALOG-SYSTEM      VALUE "DSRUN" .
```

to:

```
78  DIALOG-SYSTEM      VALUE "DS" .
```

- 4 You need to ensure that there is a copy of the file **ds.lbr** in your current directory in addition to the **dsrun.lbr** which is currently there.
- 5 Go into Net Express and "check" **customer.cbl**.

6 Run the program **customer.cbl**.

The program is now calling the trap. This enables you to have all the trap functions except for two; you cannot change the screenset name and you cannot change the control field. It is probably important here for you to consider how this facility will help you in developing your applications.

7 Edit the program **customer.cbl** to change the line back to its original form.

8 Go into Net Express and "check" **customer.cbl** again, ready for tutorial 14.

23.14 Tutorial 14 - Case Sensitivity in Screensets

In this tutorial you will learn more about the behavior of a screenset and its relation with the calling program.

The files you need for this tutorial are:

customer.s3
customer.cbl
ds-cntrl.mf
customer.cpb
customer.err

The program is the same program that you have just used, but the screenset **customer.s3** is different. Before the program can use this new screenset, you must load it and save it as **customer.s**.

- 1 Look at the local dialog in each of the panels.
- 2 Run this screenset with trace until you fully understand its behavior, and then run the program.

You can see that you are now forced to enter upper case values into the customer code. This is because case sensitivity off was selected when the field was placed on the panel. Also you cannot access any customer records whose key contains lower case letters. This

illustrates why you must decide in advance whether you wish to force upper case on keys.

This is the end of Tutorial 14.

23.15 Tutorial 15 - Using Different Screensets

In this tutorial you will learn more about the behavior of a screenset and its relation with the calling program.

The files you need for this tutorial are:

customer.s4
customer.s5
customer.cbl
ds-cntrl.mf
customer.cpb
customer.err

Just like the other examples, load the screensets and run them through the trap before you run the program. Save the screenset to use as **customer.s** so you can use it with the program **customer.cpb**.

In **customer.s4**, the screenset shows the menu permanently and switches between menu and input screen using the **F10** key.

The screenset **customer.s5** is included as an example of a user interface still using the same **CUSTOMER** program and created by the action bar and pulldown generation facility included with Dialog System.

This is the end of Tutorial 15.

23.16 Tutorial 16 - Paging a Large File

In this tutorial you will see how a program is used to control the paging of data in and out of a file, and how this relates to the panel and Data Block paging in the screenset.

The files you need for this tutorial are:

dsdemo5.s
dsdemo5.cbl
ds-cntrl.mf
dsdemo5.cpb
dsdemo5.file

Generate **dsdemo5.cpb** in Dialog System using the generate feature.

The application illustrated in this tutorial is typical of the sort that developers commonly wish to create.

There is a relative file containing many records and you wish to display a number of records in a panel. (This particular panel displays 18 records.) To minimize disk activity, you can create a Data Block that has more records than the panel. (This Data Block has 52 records, which allows for page down on 18 twice without accessing the disk.)

The COBOL program is:

```
$set ans85 comp noqual noalter align(1) noosvs
$set vsc2
```

```
identification division.
```

```
* This program handles the viewing of the contents of a
* relative file. The screen has a visible size of
* "visible-size". The size of the file array in the data block
* is "file-array-size".
* For this program to work correctly the file-array-size must
* be greater than the visible-size.
```

```
* You are provided with test file "dsdemo5.file"which
* illustrates the use of this program. It may be of interest
* to note that the record size of this test file is exactly
* the same as the record size of any of the error message
* files you create. As a tutorial you may wish to use this
* program to look at one of your own error message files.
* (Beware however that the error message files have a dummy
```



```
* system record at the start, so that your error numbers are
* always one greater than the relative record key.
```

```
environment division.
```

```
input-output section.
```

```
file-control.
```

```
    select test-file assign "dsdemo5.file"
           organization is relative
           access mode is dynamic
           relative key is rel-key
           file status is file-status.
```

```
data division.
```

```
file section.
```

```
fd test-file.
```

```
01 test-file-record          pic x(76).
```

```
working-storage section.
```

```
* local proc numbers
```

```
78 data-up-proc              value 144.
```

```
78 data-down-proc           value 145.
```

```
* run time and screenset name constants
```

```
78 dialog-system            value "DSRUN".
```

```
78 screen-set-name          value "DSDEMO5".
```

```
* visible-size is one less than the size of the visible
* portion of the array on the screen. This is to allow paging
* of data to still leave one of the previous records visible
* on the screen
```

```
78 file-array-size          value 52.
```

```
78 visible-size             value 17.
```

```
* relative record variables
```

```
01 rel-key                   pic 9(4) comp.
```

```
01 rec-limit                 pic 9(4) comp.
```

```
01 actual-amount-read        pic 9(4) comp.
```

```
* temporary indexes used in array manipulations
```

```

01 temp-ind-1          pic s9(4) comp.
01 temp-ind-2          pic s9(4) comp.
01 file-status        pic x(2).
    88 successful-io   value "00".
    88 end-of-file     value "10" "46".

* buffer used to store extra records when paging
* beyond the current limits of the array.

01 paging-buffer.
    03 paging-buffer-item occurs visible-size.
        05 buffer-rnumber    pic 9(3).
        05 buffer-rcontents  pic x(76).

copy "ds-cntrl.mf".

copy "dsdemo5.cpb".

*****
* procedure division          *
*****
procedure division.
main-process section.
    perform initialise-demo-handler
    perform process-file-list
        until dsdemo5-esc-flag-true
    perform terminate-process.

*****
initialise-demo-handler section.
    initialize dsdemo5-data-block
    move dsdemo5-data-block-version-no
        to ds-data-block-version-no
    move dsdemo5-version-no to ds-version-no
    open input test-file
    perform load-array-from-file
    if rec-limit < file-array-size
        move rec-limit to dsdemo5-line-lim
    else
        move file-array-size to dsdemo5-line-lim
    end-if
    initialize ds-input-fields
    move ds-new-set to ds-control
    move screen-set-name to ds-set-name
    perform call-dialog-system
    move ds-continue to ds-control.

```

```

*****
process-file-list section.
  if dsdemo5-up-flag-true
    perform adjust-array-up
  else
    if dsdemo5-dn-flag-true
      perform adjust-array-down
    end-if
  end-if
  perform clear-flags
  perform call-dialog-system.
*****
terminate-process section.
  close test-file
  stop run.
*****
adjust-array-up section.
  move dsdemo5-rnumber(1) to rel-key
  perform read-previous-page
  if actual-amount-read > zero
    perform shuffle-array-down
    move actual-amount-read
      to dsdemo5-adj-cnt
    move data-down-proc to ds-proc-no
  end-if.
*****
adjust-array-down section.
  move dsdemo5-rnumber(file-array-size)
    to rel-key
  perform read-next-page
  if actual-amount-read > zero
    perform shuffle-array-up
    move actual-amount-read
      to dsdemo5-adj-cnt
    move data-up-proc to ds-proc-no
  end-if.
*****
read-previous-page section.
* this section attempts to read backwards through the file up
* to visible-size records
  move zero to actual-amount-read
  subtract 1 from rel-key
  if rel-key not = 0
    move visible-size to temp-ind-1
    perform read-test-file

```

```

perform until rel-key = 0
  or actual-amount-read = visible-size
  if successful-io
    add 1 to actual-amount-read
    move rel-key
      to buffer-rnumber(temp-ind-1)
    move test-file-record
      to buffer-rcontents(temp-ind-1)
    subtract 1 from temp-ind-1
  end-if
  subtract 1 from rel-key
  perform read-test-file
end-perform
end-if.
*****
read-next-page section.
* this section attempts to read forwards through
* the file up to visible-size records
move zero to actual-amount-read
perform read-test-file
perform read-next-record
perform until end-of-file
  or actual-amount-read = visible-size
  if successful-io
    add 1 to actual-amount-read
    move rel-key
      to buffer-rnumber(actual-amount-read)
    move test-file-record
      to buffer-rcontents(actual-amount-read)
  end-if
  perform read-next-record
end-perform.
*****
load-array-from-file section.
* this section does the initial load of as much of
* the file as it can get into the array
move 1 to rec-limit
move 1 to rel-key
perform read-test-file
perform until not successful-io
  or rec-limit > file-array-size
  if successful-io
    move rel-key
      to dsdemo5-rnumber(rec-limit)
    move test-file-record
      to dsdemo5-rcontent(rec-limit)
    add 1 to rec-limit
  end-if

```

```

        perform read-next-record
    end-perform
    subtract 1 from rec-limit.
*****
shuffle-array-up section.
    move actual-amount-read to temp-ind-1
    add 1 to temp-ind-1
    move 1 to temp-ind-2
    perform until temp-ind-1 > file-array-size
        move dsdemo5-group-item-001(temp-ind-1)
        to dsdemo5-group-item-001(temp-ind-2)
        add 1 TO temp-ind-1
        add 1 TO temp-ind-2
    end-perform
    move 1 to temp-ind-1
    move file-array-size to temp-ind-2
    subtract actual-amount-read from temp-ind-2
    add 1 to temp-ind-2
    perform until temp-ind-2 > file-array-size
        move paging-buffer-item(temp-ind-1)
        to dsdemo5-group-item-001(temp-ind-2)
        add 1 TO temp-ind-1
        add 1 TO temp-ind-2
    end-perform.
*****
shuffle-array-down section.
    move file-array-size to temp-ind-1
    subtract actual-amount-read from temp-ind-1
    move file-array-size to temp-ind-2
    perform until temp-ind-2 = actual-amount-read
        move dsdemo5-group-item-001(temp-ind-1)
        to dsdemo5-group-item-001(temp-ind-2)
        subtract 1 from temp-ind-1
        subtract 1 from temp-ind-2
    end-perform
    move visible-size to temp-ind-1
    move actual-amount-read to temp-ind-2
    perform until temp-ind-2 < 1
        move paging-buffer-item(temp-ind-1)
        to dsdemo5-group-item-001(temp-ind-2)
        subtract 1 from temp-ind-1
        subtract 1 from temp-ind-2
    end-perform.
*****
clear-flags section.
    move 0 to dsdemo5-esc-flag
    move 0 to dsdemo5-up-flag
    move 0 to dsdemo5-dn-flag.

```

```

*****
read-test-file section.
  read test-file.
*****
read-next-record section.
  read test-file next.
*****
call-dialog-system section.
  call dialog-system using ds-control-block
                        dsdemo5-data-block
  if ds-system-error-no > zero
    perform terminate-process
  end-if.
*****

```

The best way to understand the combination of screenset, program and file is to animate the program while you run the screenset.

A number of new functions are used, which monitor the size of the panel array. For further information about these, see the chapters *Function Code List* and *Dialog*.

This is the end of Tutorial 16.

23.17 Tutorial 17 - Using Micro Focus Panels

Windows: In this tutorial, which is not available on UNIX, you will learn how a Micro Focus Panels screenset can be manipulated. Full details of the use of Micro Focus Panels can be found in your Net Express documentation.

The files you need for this tutorial are:

```

dsdemo6.s
dsdemo6.cbl
ds-cntrl.mf
dsdemo6.cpb
panlink.cpy

```

Generate **dsdemo6.cpb** using the generate option in Dialog System.

This is a demonstration of the use of Micro Focus Panels based screensets. When you load the screenset, you can see that it is a Micro

Focus Panels screenset by looking at the **F2** key on the **Alt+F5** menu from the main menu. The main additional function used here is the extraction of the panel-id (MPID). The program the uses this panel-id to call Micro Focus Panels directly. The Panels program provided with Dialog System is called Dspanels.

Animate the program to examine the behavior of the screenset and the program.

This is the end of Tutorial 17.

23.18 Tutorial 18 - Menu Selection - Example 1

In this tutorial, you will see a menu that uses a virtual text group with multiple pulldown sub-menus for each main menu item. The pulldown menus are all examples of fixed text groups. Each pulldown option passes control to a panel.

The file you need for this tutorial is **dsdemo7.s**.

Load this screenset and examine the dialog in each of the panels. Run the screenset through the trap and look at its behavior with the trace facility.

This is the end of Tutorial 18.

23.19 Tutorial 19 - Menu Selection - Example 2

In this tutorial, you will see examples of two pick and point menu types. Also, you will see an example of a data group with automatic scrolling backward and forward using both the Tab and cursor keys.

The file you need for this tutorial is **dsdemo8.s**.

Load this screenset and examine the dialog in each of the panels. Run the screenset through the trap and look at its behavior with the trace facility.

This is the end of Tutorial 19.

24 Dialog System Limits

This chapter lists the system limits of Dialog System.

- The maximum length of a numeric field, including any decimal part, is 18 digits.
- The maximum decimal part of a numeric field is 9 digits.
- The maximum number of characters in a text literal is 80.
- The maximum number of data fields in a screenset is 512.
- The maximum number of fields allowed in the Data Block is 512.
- The maximum size of the Data Block is 65535 bytes.
- The maximum number of data fields allowed per panel is 100.
- The maximum number of panels that can be visible on the screen is 50.
- The maximum number of active Micro Focus Panels per screenset is 50.
- The maximum number of panel fields for one panel is 200.
- The maximum number of groups in a screenset is 255.
- The maximum size of a group is 4000 bytes.
- The maximum number of panel groups allowed is 30.
- Maximum group occurrence is 9999
- A group cannot repeat more than 9999 times.
- Maximum field name length is 30 characters.
- The maximum number of fields in a field stack is 16.
- The maximum number of blocks in a block stack is 16.
- The maximum number of validations in a list is 30.
- A maximum of 30 Range/Table validations are allowed for each field.

- The maximum number of weights you can enter to use for check digit validation is 18.
- Error message numbers must be in the range of 1 - 998.
- The maximum number of keys in a dialog table is 100.
- The maximum number of nested procedures is 16.
- Valid procedure numbers are P000 - P255
- A minimum panel height of three lines and width of three characters is required for a border.
- The maximum level of nesting of pushed panels is 15.
- The maximum level of nesting of screensets is 16.

Index

A

- Abandon
 - block 123
 - range/table validation 90
 - virtual group 157
- Action bar 107, 108
 - definition menu 108
 - dialog 115
 - entry 110
 - generate 114
 - initialization procedure 115
 - initialize 116
 - mnemonic 111
 - name 109, 114
 - palette 256
 - position 109, 113
 - pull-down 116
 - save 114
 - text 111
 - width 109, 113
- Add occurrence
 - group 161
- Add range 89
- Allow/disallow null 97
- Alphabetic
 - field 81, 86
 - field format 144
- Alphanumeric
 - field 81
 - field format 144
- Alphanumeric value parameters 176
- Alt key 286, 303
- Alter panel size 127
- Alter size
 - action bar 113
- Amend
 - field 147
 - field menu 148
 - group repeats 84
 - group size 82
 - pull-down menu 116
- ANSI constants 253
- ANYO 288
- Array
 - checking size 274
 - deleting 274
 - inserting 274
- ASCII keys 285
- ASCII VALUE configuration parameter 258
- Attribute
 - default background 130
 - functions 166
 - group menu 163
 - groups 152, 163
 - palette, panel painting 129
 - parameters 176
 - static panel 130
- Attribute Roll
 - panel painting 125
- Attribute roll 130
 - draw 136
 - selection bar 153
- Attributes 69, 392, 400
 - auxiliary field 130
 - error field 131
 - field entry 131
 - of panel 57
 - palette 255

- Autoskip
 - field property 142
- Auxiliary field attributes 130

B

- BEEP 174, 292
- BEEP-EOF configuration parameter 259
- BEEP-EOS configuration parameter 260
- BEEP-INVALID configuration parameter 260
- Beta version compatibility 254
- Block 46
 - copy from 123
 - copy to 126
 - cut to 125
 - move 122
 - paste 122
 - restore 126
 - stack 123
- Block menu 122
- Blocks
 - working with 122
- Border
 - panel 127
- BP 167, 293
- BPD 171, 293
- BPE 167, 294
- BPR 167, 294
- Browse file 65
 - generate 206

C

- Call interface 200, 211
- Call library
 - generate 205
- Calling program
 - writing 211
- Calling program function 168
- CALLOUT 168

- Callout function 168
- Cancel library
 - generate 206
- Case sensitivity off/on
 - dialog 183
- CEOF 173
- .CFG 69
- CFLD 173
- Chaining
 - panels 283
- Character palette
 - panel painting 136
- Character set supported on UNIX 248
- Check digit
 - example 93
 - validation 91
 - validation menu 91
- CLEAR 173
- Clear
 - panel 131
- Clear dialog
 - trap field 190
- Clear field functions 173
- Clearing a screenset 40
- CLRF 172
- COBOL program 50
- COFF 166, 297
- Color 57
- Colorize 255
- Colorize menu 69
- Command line 30
- Complete action bar 114
- Computational numeric field 81
- CON 166, 297
- Conditional functions 171, 302
- Configuration 237
- Configuration file 251
- Configuration parameters
 - ASCII VALUE 258
 - BEEP-EOF 259
 - BEEP-EOS 260
 - BEEP-INVALID 260
 - CONTROL BLOCK ANSI-CONSTANTS 253
 - CONTROL BLOCK COMPATIBILITY 254

- CONTROL BLOCK MF-CONSTANTS 253
- CUA-PALETTE 256
- CURSOR-LARGE 260
- DATA BLOCK FLAGS EIGHTY-EIGHTS 253
- DATA BLOCK INDEXED BY CLAUSE 254
- DATEFORM 260
- DECIMAL-POINT 261
- DEFAULT BACKGROUND CHARACTER 257
- EMPTY-DATE-BLANK 261
- IGNORE-NUMERIC-SETCUR 262
- NO OF PRINT LINES ON A PAGE 258
- NOCLEAR-PATH 263
- NOECHO-CHARACTER 263
- PALETTE 255
- SHADOW 263
- SIGN-TRAILING 264
- SUPPRESS-CURSOR-RIGHT 261, 264
- SUPPRESS-TO-BWZ 264
- TERMINAL DOS8 242
- TERMINAL GENERIC8 242
- Constants
 - MF or ANSI 253
- Control
 - trap field 189
- Control block 51
 - ANSI copyfile 195
 - level-88s 252
 - MF copyfile 194
- CONTROL BLOCK ANSI-CONSTANTS
 - configuration parameter 253
- CONTROL BLOCK COMPATIBILITY
 - configuration parameter 254
- Control block fields 189
- CONTROL BLOCK MF-CONSTANTS
 - configuration parameter 253
- Control blocks 193
- Control param
 - trap field 191
- Conversion utility
 - translation file 244
- Copy
 - panel 106
- Copy from block 123, 126, 374

- Copy to block 374
- Copy to stack 147
- Copyfile 203
 - defaults 252
 - generating 203
- COPY-files 51
- .CPB 203, 204
- Creating a screenset 36
- Ctrl key 286, 303
- CUA-PALETTE 256
- CUA-PALETTE configuration parameter 256
- Cursor functions 166
- Cursor keys 286
- CURSOR-LARGE configuration parameter 260
- Cut to block 125
- Cut to stack 147
- Cutting 373

D

- Data Block 37, 51
- Data block 79, 203
 - copyfile 208
 - example 209
 - INDEXED BY clause 254
 - level-88s 253
- DATA BLOCK FLAGS EIGHTY-EIGHTS
 - configuration parameter 253
- DATA BLOCK INDEXED BY CLAUSE
 - configuration parameter 254
- Data block version number
 - trap field 192
- Data definition 59, 79
- Data descriptions
 - generate 208
- Data entry fields 42
- Data field 80
 - array 274
 - scrolling 272
- Data group menu 158
- Data groups 151, 158

- Data item
 - trap field 193
- Data items 35
- Data manipulation functions 172
- Data values
 - trap field 193
- Date delimiter
 - panel field 146
- Date validation 94
 - menu 94
- DATEFORM configuration parameter 260
- DBP 171, 297
- DECIMAL-POINT configuration parameter 261
- DECVAl 172, 298
- DEFAULT BACKGROUND CHARACTER configuration parameter 257
- Defaults
 - copyfile 252
 - printing 237, 257
- Define
 - error messages 100
 - group 84
 - panel fields 139
 - validation details 85
 - virtual attributes 164
 - virtual text 162
- Definition component 21
- Delete
 - field 83
 - panel 107
 - validation 88
- Delete all validations 87
- Delete entry
 - action bar 112
 - pull-down 117
- Delete field definition 137
- Delete file 64
 - generate 206
- Delete group definition 137
- Delete line
 - dialog 181
 - error message definition 101
 - range/table validation 88
 - virtual group 156
- Delete selection bar 153
- Delete validation 94, 96
- Delete virtual text/attribute 156
- Dialog 21, 53, 165
 - global 165
 - global/local first 67
 - local 129, 165
- Dialog definition alternate menu 185
- Dialog definition menu 178
- Directory control menu 65
- Directory menu 62
- Divisor
 - check digit validation 91
- Documenting 385
- DOS 22
- DOS screensets 241
 - restrictions on UNIX 247
 - transferring to UNIX 247
- Draw 135
- Draw/erase/move 135
- Drive 64
- ds.lis 237
- Dsclink 235
- ds-cntrl.ans 195
- ds-cntrl.mf 51, 194
- Dscomp 233
- Dsdakey 232
- dsdef.cfg 237, 251
- DSDIR 251
- Dsdlgini 233
- DSERRHAN 216
- Dserrhan 214, 215, 232
- Dsfld 232
- Dsgetss 233
- ds-key.cpy 227
- Dslconv 232
- Dspanels 233
- Dsrtno 232
- Dsrun 211, 232
- Dsterm 232
- Dstracer 233
- Dsusrcal 232

- Dsusrfmt 145, 217, 227, 233, 234
 - control block 218
 - parameters 220
- Dsusrtrn 227, 234
 - parameters 228
- Dsusrval 232
- Dsuxpath 232
- Dsuxsysp 232
- Dsvalrun 232
- Duplicate line
 - error message definition 102
 - virtual group 156

E

- Edit action bar entry 114
- EMPTY-DATE-BLANK configuration
 - parameter 261
- Enter key 184
- Entry fields 42
- Environment variable
 - DSDIR 251
- ERR 288
- Error code
 - trap field 192
- Error field attributes 131
- Error key 288
- Error Messages
 - check digit validation 94
- Error messages 99, 343, 425
 - date validation 96
 - definition time 346
 - dialog definition 353
 - error message definition 349
 - field 79, 270
 - file 213, 214
 - import 364
 - null validation 97
 - panel 271
 - panel field definition 350
 - panel group definition 352
 - range/table validation 89

- run-time 343
- screen painting 357
- validation definition 349
- Escape key 286
- Event 185
- Event key define/undefine 185
- Events 53
- Exit
 - field usage 141
- Exit field
 - trap field 192
- Exit modified
 - field usage 141
- Exit on entry
 - field usage 142
- Exit regardless
 - field usage 142
- Expand/contract
 - group size 158
- Export 72
- Export files
 - syntax 331
- Extension
 - of screenset file 40
- ExtFH 234

F

- Field change
 - trap field 193
- Field count
 - trap field 193
- Field entry attributes 131
- Field format 80
 - user-defined 217
- Field name 80
- Field name parameter 175
- Field naming menu 139
- Field number
 - trap field 192
- Field numbers
 - copyfile 208

- Field Selection panel 43
 - Field size 38
 - Fields 35
 - alphabetic 81, 86
 - alphanumeric 81
 - data 80
 - delete 83
 - format 144
 - group occurrence 160
 - insert 83
 - length 81
 - numeric 86
 - numeric signed 81
 - numeric unsigned 81
 - order 134
 - panel 139
 - properties 142
 - repeats 81
 - required 86
 - usage 141
 - validation 82
 - Flag functions 172
 - Format
 - field 80
 - of fields 38
 - panel field 144
 - Full
 - field property 143
 - Function
 - dialog 179
 - Function code list 291
 - Function definition 55
 - Function detail menu 182
 - Function key
 - pulldown 119
 - Function keys 285
 - Functions 166
 - attribute 166
 - calling program 168
 - callout 168
 - clear field 173
 - conditional 171
 - cursor 166
 - data manipulation 172
 - flag 172
 - keyboard scan 169
 - move panel 173
 - panel view 173
 - path control 167
 - procedure 167
 - refresh 168
 - screen group array size 170
 - screen group data positioning 170
 - screen group insertion and deletion 171
 - selection bar 169
 - sound 174
 - stack 168
 - terminate 174
 - timeout 174
 - validate 173
- ## G
- Generate 60
 - action bar 114
 - Generate COBOL menu 207
 - Generate copyfile 203
 - Generate prompt control menu 204
 - Generate prompt menu 203
 - Generating 51
 - Global Dialog 60
 - Global dialog 53, 165
 - Global/local dialog first 67
 - GOF 167
 - GOP 167
 - Group accept exit bar/end 159
 - Group name
 - attribute group 164
 - data group 159
 - menu 154
 - parameters 175
 - text group 163
 - Group namelist 154
 - Group occurrence maintenance 160
 - Group occurrence maintenance menu 160
 - Group selection bar functions 169

- Group selection menu 161
- Group size maintenance 157
 - attribute group 164
 - group occurrence 160
 - menu 157
 - text group 163
- Group types 151
- Groups 397
 - amend repeats 84
 - amend size 82
 - attribute 152, 163
 - data 151, 158
 - define 84
 - panel 151
 - scrolling 375
 - text 152, 161
 - types 401
 - undefine 85
 - working with 152

H

- Hexadecimal input 89
- High level definition
 - palette 256

I

- IBP 171, 301
- IF 171
- IF functions 171, 302
- IGNORE-NUMERIC-SETCUR configuration
 - parameter 262
- Import 70
 - limitations 76
 - semantic checking 72
 - syntax checking 71
- Import files
 - syntax 331
- INCVAl 172, 303

- Information
 - system 228
- Initialization procedure 177
 - action bar 115
- Initialize 62
 - action bar 116
 - data fields 201
- Input
 - field usage 141
- Input attributes 400
- Ins key 303
- Insert after
 - action bar 113
- Insert after entry
 - action bar 112
 - pull-down 117
- Insert before
 - action bar 113
- Insert before entry
 - action bar 112
 - pull-down 117
- Insert field 83
- Insert line
 - dialog 181
 - error message definition 101
 - range/table validation 88
 - virtual group 156
- Interface 403
- Internal register 178
- Invoking Dialog System 25

K

- Key
 - dialog 179
- Key code list 285
- Key translation off/on 67
- Keyboard scan function 169
- Keys 29
 - Alt 286
 - ASCII 285
 - Ctrl 286

- cursor 286
- error 288
- escape 286
- function 285
- lock 287
- other 288
- run-time behavior 184
- shift 287
- status 287
- Keystroke control 227
- KS 169, 303

L

- Line
 - delete, range/table validation 88
 - insert, range/table validation 88
- Linking 231
 - dsclink 235
 - response files 234
- .LIS 237
- List
 - dialog 180
- List ascending 64
- List descending 64
- List directories 63
- List files/list library catalogue 63
- Listing
 - screenset 237
- Load screenset 62
- Loading a screenset 40
- Local dialog 53, 129, 165
- Lock keys 287, 303
- Lower case
 - field property 143

M

- Main alternate menu 61
- Main menu 25, 59

- Map screenset colors map to COBOL system
 - 70
- Mark/Unmark 104
- Mark/unmark
 - panel painting 123
- Marking an area 41
- MAS 170, 304
- MB 170, 304
- MBD 170, 305
- Menu
 - action bar definition 108
 - amend field 148
 - amend pulldown 116
 - attribute group 163
 - block 122
 - check digit validation 91
 - colorize 69
 - data group 158
 - date validation 94
 - dialog definition 178
 - dialog definition alternate 185
 - directory 62
 - directory control 65
 - error message definition 99
 - field naming 139
 - function detail 182
 - generate COBOL 207
 - generate prompt 203
 - generate prompt control 204
 - group name 154
 - group occurrence maintenance 160
 - group selection 161
 - group size maintenance 157
 - main 25, 59
 - main alternate 61
 - null validation 96
 - panel functions 146
 - panel maintenance 126
 - panel painting 121, 123
 - panel painting alternate 128
 - panel painting control 133
 - panels 103
 - print 237
 - range/table validation 87

- range/table validation alternate 90
- selection 423
- selection bar 152
- trace 197
- trap screen 197
- validation 85
- virtual attribute group definition 155
- virtual text group definition 155

Menu entry

- table 276

Menu on/off 137

Menu system 27

Micro Focus constants 253

Micro Focus Panels 66, 422

Mnemonic

- action bar 111
- pulldown 119

MOUSE-ENABLE 262

MOVE 172, 306

Move

- block 122
- panel 127

Move panel

- trace menu 199

Move panel function 173

MOVEPNL 173, 306

MOVETXT 307

MOVEVTXT 172

Moving fields 46

MPID 172, 308

MTD 170, 308

Multiple screensets 212

N

Next panel 135

NLS folding 143, 262

NO OF PRINT LINES ON A PAGE configuration parameter 258

NOCLEAR-PATH configuration parameter 263

Noecho

- field property 144

NOECHO-CHARACTER configuration parameter 263

None/single/double border 127

Normal/Micro Focus Panels 66

\$NULL 177

Null parameter 177

Null validation 96

Null validation menu 96

Numeric

- field format 144

Numeric fields 86

Numeric signed field 81

Numeric value parameters 176

O

Object modules

- applications in development 233
- linking 231
- production applications 232

Occurrence

- trap field 193

Off/on

- colorize 69

Order of dialog 56

Output

- field usage 141

P

Page data fields 201

Paint attributes

- panel painting 124
- selection bar 153

Painting a panel 41

Painting panels 121

PALETTE 255

- Palette
 - screensset 255
- PALETTE configuration parameter 255
- Panel
 - chaining 283
 - copy 106
 - delete 107
 - fields 139
 - relocate 107
 - select first 106
 - show 105
 - trap screen 198
 - unshow 106
- Panel components
 - export 74
- Panel fields 124, 139
 - amend 147
 - defining 139
 - format 144
 - properties 142
 - scrolling 146
 - usage 141
 - user-defined format 145
- Panel functions menu 146
- Panel groups 124, 151
- Panel length 44, 146
- Panel list 104
 - export 75
- Panel maintenance 126
- Panel maintenance menu 126
- Panel name 131
 - trap field 192
- Panel name parameter 175
- Panel painting 121
- Panel painting alternate menu 128
- Panel painting control menu 133
- Panel painting menu 121, 123
- Panel view function 173
- Panels 59, 103
 - menu 103
 - Micro Focus 66, 422
- Parameters 175
 - alphanumeric value 176
 - attribute 176
 - field anem 175
 - group name 175
 - null 177
 - numeric value 176
 - panel name 175
 - procedure name 176
 - register 176
- Paste block 122
- Pasting 373
- Path control functions 167
- Path name support 229
- PBDN 170, 309
- PBUP 170, 310
- PDDN 170, 311
- PDUP 170, 313
- POP 168, 314
 - screensset 212
- POPGOP 168
- Position bar 113
- Prefix on/off 210
- Previous panel 134
- Print 60, 237, 385
 - screensset 237
- Print menu 237
- Printing
 - defaults 237
- Printing defaults 257
- Procedure
 - functions 167
 - numbering 178
 - trap field 190
- Procedure name parameters 176
- Procedures 177, 400
- Program
 - example 404
 - writing 50
- Program development 22
- Programming 211
- Properties
 - panel field 142
- Protect definition 132
- Prototyping 47
- Pulldown 116
 - edit entry 118

- entry text 119
- function key 119
- mnemonic 119
- name 117
- next menu 118
- previous menu 118
- return value 119

PUSH 168, 314

- screensset 212

R

Range

- add 89
- remove 89

Range/table validation 87

Range/table validation alternate menu 90

Range/table validation menu 87

Refresh functions 168

\$REG 176

Register

- trap screen 198

Register parameters 176

Relocate entry

- action bar 112

Relocate panel 107

Remove occurrence

- group 161

Remove range 89

Rename file

- generate 206

Repeat key 183

Repeats

- amend group 84
- fields 81

Required

- field property 142

Required fields 86

Reset

- panel field 148, 149

Restore block 126

Restore line

- error message definition 102
- virtual group 156

Restrictions

- on compiling applications 230

RETC 168, 192, 315

Return value

- pulldown 119

RFT 169, 315

RFTD 169, 315

RKEY 167, 316

RPKY 167, 316

Run 60

Running a screensset 47

Running screensset 187

Run-time component 21

S

Sample program 50

Sample Session

- Character Mode 35

SAS 170, 317

Save

- action bar 114

Save screensset 66

Saving a screensset 39

SB 170, 317

SBOD 170, 318

Screen group functions

- array size 170
- data positioning 170
- insertion and deletion 171
- procedure 171

Screensset 21, 36

- listing 237
- multiple 212
- palette 255
- POP 212
- print 237
- PUSH 212

Screensset conversion utility 243

- Screenset name
 - trap field 190
 - Screenset switches 66
 - Screensets
 - conversion 243
 - DOS 241
 - UNIX 242
 - Scrolling
 - data 272
 - groups 375
 - Scrolling field 146
 - SD 170, 319
 - Select all entries
 - export 74
 - Select all panels
 - export 75
 - Select error message filename 100
 - Select first panel 106
 - Selection bar 42, 152
 - data group 159
 - functions 169
 - menu 152
 - text group 162
 - Semantic checking
 - import 72
 - Set default directory
 - generate 205
 - Set details 68
 - SETCUR 167, 320
 - SETF 172, 321
 - SFAT 166, 321
 - SGAT 166, 322
 - SHADOW configuration parameter 263
 - Shift keys 287, 303
 - Show default directory
 - generate 205
 - Show Panel 105
 - SHP 173, 322
 - Signed
 - field format 144
 - SIGN-TRAILING configuration parameter 264
 - Single/double
 - draw 136
 - SKNF 167, 322
 - SKPF 167, 323
 - Sort dialog 181
 - Sort extension 65
 - Sort list
 - panel field 149
 - Sort name 64
 - Sort size 65
 - Sort time 64
 - Sound function 174
 - SPC 285
 - SSTRAN conversion utility 243
 - Stack block 123, 374
 - Stack functions 168
 - Start/close file
 - trace menu 198
 - Status keys 287
 - Status line 26
 - Step
 - trace menu 199
 - Subscript 175
 - Sum products value/digits 94
 - Suppress
 - field format 145
 - SUPPRESS-CURSOR-RIGHT configuration
 - parameter 261, 264
 - SUPPRESS-TO-BWZ configuration parameter
 - 264
 - SUSP 169, 323
 - Syntax checking
 - import 71
 - System information 228
- ## T
- Tab
 - between panels 283
 - TERM 174, 324
 - TERMINAL DOS8 configuration parameter
 - 242
 - TERMINAL GENERIC8 configuration
 - parameter 242
 - Terminate function 174

- test1.cpb 51
- Testing a screenset 47
- Text
 - virtual 388
- Text groups 152, 161
- TIMEOUT 324
- Timeout function 174
- TOGF 172, 325
- Trace 386
 - object modules 233
 - trap screen 198
- Trace facility 197
 - in executables 236
- Trace menu 197
- Trace on/off
 - trap 197
- Translation file 244
- Trap 187, 386, 413
 - calling from program 200
 - display fields 193
 - input fields 189
 - object modules 233
 - output fields 192
 - screen 188
 - using 188
- Trap on/off 200
- Trap screen menu 197
- Trap window 47
- True/false 89
- Tutorial
 - Character Mode 35
- Tutorials 60, 373

U

- Undefine group 85
- UNIX 22
 - supported character set 248
- UNIX considerations 18
- UNIX screensets 242
- UNIX shell 18

- Unselect all entries
 - export 74
- Unselect all panels
 - export 76
- Unshow panel 106
- Unsort 64
- Unsort list
 - panel field 149
- Upper case
 - field property 143
- Usage
 - panel field 141
- Use default directory
 - generate 205
- Use stacked field 148
- User-defined field format 217
- User-defined format
 - panel field 145

V

- VAL 174, 325
- Validate function 173
- Validation 82, 85, 402
 - check digit 91
 - date 94
 - defining 380
 - delete 88
 - delete all 87
 - menu 85
 - message panel 271
 - messages 270, 380
 - null 96
 - range/table 87
 - true/false 89
- Validation code
 - trap field 192
- Validation details
 - define 85
- View parameters
 - trace menu 199
- Virtual attribute group definition menu 155

- Virtual attributes 392
- Virtual text 388
 - extraction 282
- Virtual text group definition menu 155
- Virtual text/attribute 154

W

- Weights
 - check digit validation 91
- Working with groups 152

X

- XIF 171, 326
- XP 167, 327
- XPD 171, 327
- XPE 167, 328
- XPR 167, 328

Y

- Year field conversion 30

Z

- Zoom
 - trace menu 199