



Apple Guide Complete

by Apple Computer, Inc.

Designing and
Developing
Onscreen
Assistance



Includes
Apple Guide
authoring
software

CD-ROM



Included



Apple Guide Complete: Designing and Developing Onscreen Assistance



Addison-Wesley Publishing Company

Reading, Massachusetts Menlo Park, California New York
Don Mills, Ontario Wokingham, England Amsterdam Bonn
Sydney Singapore Tokyo Madrid San Juan
Paris Seoul Milan Mexico City Taipei

🍏 Apple Computer, Inc.
© 1995 Apple Computer, Inc.
All rights reserved.

No part of this publication or the software described in it may be reproduced, stored in a retrieval system, or transmitted, in any form or by any means, mechanical, electronic, photocopying, recording, or otherwise, without prior written permission of Apple Computer, Inc. Printed in the United States of America.

The Apple logo is a trademark of Apple Computer, Inc. Use of the "keyboard" Apple logo (Option-Shift-K) for commercial purposes without the prior written consent of Apple may constitute trademark infringement and unfair competition in violation of federal and state laws.

No licenses, express or implied, are granted with respect to any of the technology described in this book. Apple retains all intellectual property rights associated with the technology described in this book. This book is intended to assist application developers to develop applications only for Apple Macintosh computers.

Apple Computer, Inc.
1 Infinite Loop
Cupertino, CA 95014
408-996-1010

Apple, the Apple logo, APDA, Ballon Help, Espy, ImageWriter, LaserWriter, Macintosh, PowerBook,

PowerTalk, and QuickTime are trademarks of Apple Computer, Inc., registered in the United States and other countries.

AppleGlot, AppleScript, Chicago, Finder, Geneva, Mac, ResEdit, and WorldScript are trademarks of Apple Computer, Inc.

Adobe Illustrator, Adobe Photoshop, and PostScript are trademarks of Adobe Systems Incorporated, which may be registered in certain jurisdictions.

America Online is a registered service mark of Quantum Computer Services, Inc.

CompuServe is a registered service mark of CompuServe, Inc.

FrameMaker is a registered trademark of Frame Technology Corporation.

Helvetica and Palatino are registered trademarks of Linotype Company.

ITC Zapf Dingbats is a registered trademark of International Typeface Corporation.

Optrotech is a trademark of Orbotech Corporation.

Windows is a trademark of Microsoft Corporation.

Simultaneously published in the United States and Canada.

ISBN 0-201-48334-3
1 2 3 4 5 6 7 8 9-MA-9998979695
First Printing, February 1995

Library of Congress Cataloging-in-Publication Data

Apple guide complete : designing and developing onscreen assistance /
[Apple Computer, Inc.].

p. cm.
Includes index.

ISBN 0-201-48334-3

1. Macintosh (Computer)—Programming. 2. Computer software—
Development. 3. Apple Guide. I. Apple Computer, Inc.

QA75.8.M3A669 1995
005.265—dc205

94-48781
CIP

LIMITED WARRANTY ON MEDIA AND REPLACEMENT

If you discover physical defects in the manuals distributed with an Apple product, Apple will replace the manuals at no charge to you, provided you return the item to be replaced with proof of purchase to Apple or an authorized Apple dealer during the 90-day period after you purchased the software. In addition, Apple will replace damaged manuals for as long as the software is included in Apple's Media Exchange program. See your authorized Apple dealer for program coverage and details. In some countries the replacement period may be different; check with your authorized Apple dealer.

ALL IMPLIED WARRANTIES ON THIS MANUAL, INCLUDING IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE, ARE LIMITED IN DURATION TO NINETY (90) DAYS FROM THE DATE OF THE ORIGINAL RETAIL PURCHASE OF THIS PRODUCT.

Even though Apple has reviewed this manual, APPLE MAKES NO WARRANTY OR REPRESENTATION, EITHER EXPRESS OR IMPLIED, WITH RESPECT TO THIS MANUAL, ITS QUALITY, ACCURACY, MERCHANTABILITY, OR FITNESS FOR A PARTICULAR PURPOSE. AS A RESULT, THIS MANUAL IS SOLD "AS IS," AND YOU, THE PURCHASER, ARE ASSUMING THE ENTIRE RISK AS TO ITS QUALITY AND ACCURACY.

IN NO EVENT WILL APPLE BE LIABLE FOR DIRECT, INDIRECT, SPECIAL, INCIDENTAL, OR CONSEQUENTIAL DAMAGES RESULTING FROM ANY DEFECT OR INACCURACY IN THIS MANUAL, even if advised of the possibility of such damages.

THE WARRANTY AND REMEDIES SET FORTH ABOVE ARE EXCLUSIVE AND IN LIEU OF ALL OTHERS, ORAL OR WRITTEN, EXPRESS OR IMPLIED. No Apple dealer, agent, or employee is authorized to make any modification, extension, or addition to this warranty.

Some states do not allow the exclusion or limitation of implied warranties or liability for incidental or consequential damages, so the above limitation or exclusion may not apply to you. This warranty gives you specific legal rights, and you may also have other rights which vary from state to state.

Contents

Figures, Tables, and Listings xiii

Preface About This Book xix

Who Should Use This Book xix
What's in This Book xx
 Designing Your Guide Files xx
 Building Your Guide Files xx
 Integrating Your Application xxi
 Using the Guide Script Commands xxi
 Reference Material xxi
Conventions Used in This Book xxi
 Special Fonts xxii
 Command Syntax xxii
 Types of Notes xxii
For More Information xxiii

Part One Designing Guide Files

Chapter 1 Introduction to Apple Guide 1-1

What Is Apple Guide? 1-3
 How the User Views Apple Guide 1-5
 The Many Uses of Apple Guide 1-8
A Typical Apple Guide Session 1-9

Designing Guide Files	2-5
Using Guide File Types	2-6
Showing Guide Files in the Help Menu	2-7
Designing About Guide Files	2-9
Designing Help Guide Files	2-10
Designing Tutorial Guide Files	2-11
Designing Shortcuts Guide Files	2-13
Designing Other Guide Files	2-14
Designing a Mixin Guide File	2-14
Designing Access Windows	2-15
Designing a Full Access Window	2-16
Designing the Application Logo or Title Area	2-18
Features for a Full Access Window With Topics Selected	2-19
Features for a Full Access Window With Index Selected	2-20
Features for a Full Access Window With Look For Selected	2-21
Designing a Single List Access Window	2-25
Designing a Simple Access Window	2-26
Designing Howdy Text on Access Windows	2-28
Designing Topic Areas and Topics	2-30
Designing Topic Areas and Topics for a Full Access Window	2-31
Designing Topics for a Single List Access Window	2-32
Designing Topics for a Simple Access Window	2-33
Designing Headings	2-33
Designing Panels	2-35
Panel Features	2-37
Designing Panel Prompts	2-39
Designing a Default Prompt Set	2-40
Overriding Default Prompts	2-42
Using the Recommended Panel Formats	2-43
Designing Your Own Panel Format	2-45
Using Graphics in Panels	2-46
Formatting Panel Text	2-47
Providing Navigation Methods on Panels	2-48
Designing Panel Types	2-50
Designing an Introductory Panel	2-51
Designing a Decision Panel	2-52

Designing an Action Panel	2-53
Designing an Information Panel	2-54
Designing a Tip Panel	2-55
Designing a Definition Panel	2-56
Designing a Related Topics Panel	2-58
Designing a Transition Panel	2-59
Designing a Closure Panel	2-60
Designing a Panel Associated With a Huh? Button	2-61
Designing an Oops Panel	2-63
Designing a Continue Panel	2-65
Designing a Sequence	2-66
Designing Branches	2-67
Designing Branches for Mutually Exclusive and Related Tasks	2-68
Designing Branches for a Specific Condition	2-69
Designing Buttons	2-70
Designing Navigation Buttons	2-71
Designing Content Area Buttons	2-72
Using Standard Buttons	2-74
Using Three-Dimensional Buttons	2-74
Using Radio Buttons and Checkboxes	2-76
Designing Hot Text, Objects, and Rectangles	2-77
Designing Coachmarks	2-79
Using Coachmark Types	2-80
Using Coachmark Styles	2-80
Using Context Checks	2-83
Comparison of Oops and Continue Panels	2-85
Analyzing a Sequence for Context Checks	2-85
Evaluation of Context Checks	2-89
Designing for Localization	2-91
Planning for Expanded Text	2-92
Translations for Apple Guide Phrases	2-92
Formats	2-93
Graphics and Buttons	2-94
Sequence Display Titles	2-94
Coachmarks	2-95
Context Checks	2-95
AppleScript	2-95

Chapter 3 Planning Your Help Content 3-1

- Determining and Creating Your Guide File Content 3-3
 - Determining Appropriate Content for Your Guide File 3-4
 - Creating Topic Areas and Topics 3-4
 - Using Flowcharts to Design Your Guide File Panels 3-7
- Helping the User Search 3-12
 - How Apple Guide Stems 3-13
 - How Apple Guide Matches Search Phrases With Topics 3-15
 - Creating a Guide File Index and Associated Lists 3-19
 - Creating a Guide File Index 3-20
 - Invisible Index Terms 3-22
 - Creating an Ignore List 3-23
 - Creating an Exception List 3-24
 - Creating a Synonym List 3-24

Part Two Building Guide Files

Chapter 4 Introduction to Guide Maker 4-1

- Guide Maker Overview 4-5
- Which Chapter Should I Read? 4-8

Chapter 5 Creating Your Guide File 5-1

- Preparing Your Source Files 5-3
- Building Your Guide File in Four Steps 5-6
- Setting Compile Options 5-8
- Checking the Syntax of Your Source Files 5-9
- Interpreting the Compile Messages 5-10
- Other Utilities 5-11
 - Importing and Exporting Resources 5-12
 - Specifying Guide File Information 5-12
 - Creating a Mixin for Your Guide File 5-13

Chapter 6 Testing Your Guide File 6-1

Testing Your Guide File's Interface	6-3
Obtaining Navigation Information	6-5
Getting Debugging Information	6-6
Testing Your Look For Content	6-8
Generating Reports	6-13
The Scopes and Keys Report	6-13
The Names to IDs Report	6-14
The Index Sort Strings Report	6-15
The Guide File Info Report	6-16
Verifying Coachmarks, Context Checks, and Event Functions	6-18
Testing Coachmarks	6-19
Testing Context Checks	6-19
Testing Event Functions	6-19
Planning for User Testing	6-20

Chapter 7 Localizing Your Guide File 7-1

The Localizing Process	7-3
Translating Text Strings	7-6

Chapter 8 Converting Windows Help Files 8-1

Preparing Your Windows Help Files	8-3
Converting Your Windows Help Files in Three Steps	8-4
Creating an Interface for Your Help Content	8-8

Part Three Integrating Guide Files

Chapter 9 Apple Guide API 9-1

Introduction	9-3
Determining Whether Apple Guide Is Available	9-4
Starting Up Apple Guide	9-5
Determining Which Guide Files Are Available	9-7
Opening and Closing Guide Files	9-11
Working With Open Guide Files	9-18
Getting Information About Guide Files	9-22
Installing and Removing Coachmark Handlers	9-33
Installing and Removing Context Check Handlers	9-36
Application-Defined Routines	9-38
Providing Object Locations for Coachmarks	9-38
Responding to Context Checks	9-40
Summary of Apple Guide API	9-45
Constants	9-45
Data Types	9-47
Functions	9-48
Result Codes	9-51

Part Four Scripting Guide Files

Chapter 10 Guide Script Command Reference 10-1

Guide Script Command Syntax	10-5
Guide Script Command Descriptions	10-8
Specifying Startup Information	10-8
Specifying the Startup Window	10-21
Specifying Default Settings	10-28
Creating Sequences	10-39
Creating Panels	10-52

Creating Buttons	10-57	
Defining and Using Text Blocks	10-82	
Formatting Text and Objects in a Panel	10-84	
Specifying Pictures and Movies	10-94	
Importing Resources	10-100	
Creating Coachmarks	10-105	
Creating Hot Items	10-119	
Defining Topic Areas	10-124	
Defining Index Terms	10-127	
Defining Topics for Topic Areas and Index Terms	10-133	
Specifying “Look For” Help	10-140	
Specifying Conditional Execution	10-152	
Defining and Using Context Checks	10-172	
Specifying Events	10-177	
Working With Mixin Guide Files	10-190	

Appendix A Guide Script Command Abbreviations A-1

Appendix B Guide Script Commands and Parameters Quick Reference B-1

Appendix C SurfWriter Guide and Its Source Files C-1

Getting Started	C-1
SurfWriter Guide Build File	C-3
Using Standard Files	C-4
The Standard Setup File	C-4
The Standard Resources File	C-5
Customizing the Setup Information	C-5
Navigation Information and Formats	C-5
Prompt Sets	C-7
Help Menu Information	C-8
Access Window Information	C-9
Finder Version Information	C-11

Help Content	C-11
Topic Areas and Topics	C-11
Sequences	C-15
The Placeholder Sequence	C-15
“How do I use the tools in the toolbar?” Sequence	C-15
“How do I add a word to the dictionary?” Sequence	C-16
“How do I create a custom dictionary?” Sequence	C-17
Panels	C-22
The Placeholder Panel	C-22
“How do I use the tools in the toolbar?” Panels	C-22
“How do I add a word to the dictionary?” Panels	C-24
“How do I create a custom dictionary?” Panels	C-28
Coachmarks	C-37
Context Checks	C-38
Event Functions	C-38
Index and Look For Content	C-39
Index Terms	C-39
The Ignore List	C-44
The Exception List	C-46
The Synonym List	C-47

Appendix D Checklist D-1

Designing Your Guide File Content	D-2
Scripting Your Source Files	D-3
Building Your Guide File	D-5
Testing Your Guide File	D-5
Additional Guide File Tasks	D-6

Glossary GL-1

Index IN-1

Figures, Tables, and Listings

Chapter 1	Introduction to Apple Guide	1-1
Figure 1-1	The Apple Guide components	1-4
Figure 1-2	A typical Apple Guide panel	1-6
Figure 1-3	A panel that is part of a sequence	1-6
Figure 1-4	The Macintosh Sound control panel circled with a coachmark	1-7
Figure 1-5	The typical steps a user takes to view an Apple Guide topic	1-10
Figure 1-6	A Help menu displaying Macintosh Guide and other guide files	1-11
Figure 1-7	A Full Access window	1-11
Figure 1-8	A Full Access Window with a topic area selected	1-12
Figure 1-9	A Full Access Window with a topic area and topic selected	1-13
Figure 1-10	The first help panel for the selected topic	1-13
Chapter 2	Authoring Tips and Suggestions	2-1
Figure 2-1	A Help menu showing the system software guide files and Balloon Help	2-8
Figure 2-2	The access window for About Apple Guide	2-9
Figure 2-3	A panel in About Apple Guide	2-10
Figure 2-4	The Full Access window for Macintosh Guide	2-11
Figure 2-5	The Macintosh Tutorial contents	2-12
Figure 2-6	The Macintosh Shortcuts contents	2-13
Figure 2-7	A Full Access window with default features	2-17
Figure 2-8	A Full Access window with an application logo and title	2-19
Figure 2-9	A Full Access window with the Topics features selected	2-20
Figure 2-10	A Full Access window with Index features selected	2-21
Figure 2-11	A Full Access window when Look For is selected	2-22
Figure 2-12	A Full Access window with Look For features and the cursor in the search phrase entry box	2-23
Figure 2-13	A Full Access window after a Look For search is completed	2-24
Figure 2-14	A Single List Access window	2-26
Figure 2-15	A Simple Access window with three-dimensional buttons	2-27
Figure 2-16	A Simple Access window that takes the user directly to the help information	2-28
Figure 2-17	A Full Access window with howdy text	2-29

Figure 2-18	A Single List Access window with howdy text	2-30
Figure 2-19	Topic headings on the Full Access window	2-34
Figure 2-20	A typical panel in Macintosh Guide	2-35
Figure 2-21	A pixel grid depiction of a panel	2-36
Figure 2-22	A minimized panel	2-37
Figure 2-23	A panel with default features	2-37
Figure 2-24	A panel design with text, graphic, and button	2-38
Figure 2-25	The default prompt for the first panel in a sequence	2-41
Figure 2-26	A decision panel	2-42
Figure 2-27	A panel with the Full format	2-44
Figure 2-28	A panel with the Tag and Body format	2-45
Figure 2-29	A panel containing a graphic centered with relative positioning	2-47
Figure 2-30	An introductory panel	2-52
Figure 2-31	A decision panel with radio buttons	2-53
Figure 2-32	A decision panel with checkboxes	2-53
Figure 2-33	An action panel	2-54
Figure 2-34	An information panel	2-55
Figure 2-35	A tip panel	2-56
Figure 2-36	A definition panel	2-57
Figure 2-37	Some definitions in the Full Access window	2-58
Figure 2-38	A related topics panel	2-59
Figure 2-39	A transition panel	2-60
Figure 2-40	A closure panel	2-61
Figure 2-41	A panel with an active Huh? button and an explanation of the button's associated panel	2-62
Figure 2-42	A typical panel associated with a Huh? button	2-62
Figure 2-43	An Oops panel	2-64
Figure 2-44	A Continue panel	2-65
Figure 2-45	A panel with GoStart (using the lightbulb icon) and Huh? buttons in the navigation bar	2-72
Figure 2-46	A panel with a button centered using relative positioning	2-73
Figure 2-47	A panel containing two buttons with absolute positioning	2-73
Figure 2-48	The up and down appearance (in grayscale) of a color Continue button	2-75
Figure 2-49	The up and down appearance of a black-and-white Continue button	2-75
Figure 2-50	A single word of hot text on a panel	2-78
Figure 2-51	A panel associated with a single hot-text word on a panel	2-78
Figure 2-52	A panel associated with a Huh? button	2-78
Figure 2-53	A menu coach	2-79

Figure 2-54	A red circle coachmark	2-81
Figure 2-55	A red underline coachmark	2-82
Figure 2-56	A green X coachmark	2-82
Figure 2-57	A red arrow coachmark	2-83
Figure 2-58	The SurfWriter sequence for adding a word to the dictionary	2-86
Figure 2-59	A Continue panel for a condition in a SurfWriter sequence	2-87
Figure 2-60	An Oops panel for a condition in a SurfWriter sequence	2-88
Figure 2-61	A sequence in Macintosh Guide for changing the beep sound	2-90
Figure 2-62	Avoiding embedding pictures inside of text	2-94
Table 2-1	The default prompt set recommended by Apple	2-41
Table 2-2	Override prompts by panel type	2-43
Table 2-3	Order in which Apple Guide evaluates context checks	2-89
Table 2-4	Common translations of Apple Guide terms	2-93

Chapter 3 Planning Your Help Content 3-1

Figure 3-1	A flowchart that breaks topic areas into topics	3-8
Figure 3-2	A typical flowchart	3-9
Figure 3-3	A flowchart for a SurfWriter sequence on creating a custom dictionary	3-11
Figure 3-4	An Apple Guide search in response to a search phrase	3-16
Figure 3-5	A typical Look For search	3-18
Table 3-1	Some words stemmed by Apple Guide	3-14
Table 3-2	Examples of index terms derived from sample topics	3-21
Table 3-3	Example of ignore words derived from sample topics	3-23
Table 3-4	Examples of synonyms derived from sample topics	3-26

Chapter 4 Introduction to Guide Maker 4-1

Figure 4-1	Building, testing, localizing, and converting your online assistance using Guide Maker	4-4
Figure 4-2	Accessing Guide Maker's utilities	4-5
Figure 4-3	Using Guide Maker's menus	4-7

Chapter 5 Creating Your Guide File 5-1

Figure 5-1	Creating a build file	5-5
Figure 5-2	Building your guide file using Guide Maker's Build utility	5-6
Figure 5-3	A successfully compiled guide file	5-7
Figure 5-4	Compile options dialog box showing default settings	5-8
Figure 5-5	Guide Maker displaying a compile error message	5-10
Figure 5-6	Guide Maker displaying a warning message	5-11
Figure 5-7	The Utilities menu	5-12
Figure 5-8	The Menu Appearance dialog box	5-13
Figure 5-9	A guide file and its guide file addition, or mixin	5-14

Chapter 6 Testing Your Guide File 6-1

Figure 6-1	The Diagnose window	6-4
Figure 6-2	The "All messages" debugging option	6-7
Figure 6-3	The Test Look For window	6-9
Figure 6-4	A parsed phrase in the Test Look For window	6-10
Figure 6-5	Results of a search	6-12
Figure 6-6	A Scopes and Keys report	6-14
Figure 6-7	A Names to IDs report	6-15
Figure 6-8	An Index Sort Strings report	6-16
Figure 6-9	The Options dialog box	6-17
Figure 6-10	A Guide file Info report	6-18

Chapter 7 Localizing Your Guide File 7-1

Figure 7-1	The Localize window	7-4
Figure 7-2	The Localize window with files and folders specified	7-4
Figure 7-3	Examples of text resources	7-7
Figure 7-4	Text resources for panels, index terms, and Look For content	7-8
Figure 7-5	Translating a text string	7-9
Table 7-1	The 'TEXT' resource names and the associated text strings	7-10

Chapter 8 Converting Windows Help Files 8-1

Figure 8-1	Converting your Windows Help files using Guide Maker's Convert utility	8-5
Figure 8-2	A successfully converted Windows Help file	8-6
Figure 8-3	A Windows Help file and its converted Guide Script source file	8-7
Figure 8-4	Creating an interface for your help content	8-8
Figure 8-5	Constructing an interface for a sample source file	8-10

Appendix A Guide Script Command Abbreviations A-1

Table A-1	Command abbreviations	A-1
------------------	-----------------------	-----

Appendix B Guide Script Commands and Parameters Quick Reference B-1

Table B-1	Commands quick reference	B-1
------------------	--------------------------	-----

Appendix C SurfWriter Guide and Its Source Files C-1

Figure C-1	The organization of the source files for SurfWriter Guide	C-2
Figure C-2	The access window with Topics selected	C-12
Figure C-3	A panel with radio buttons	C-18
Figure C-4	"How do I add a word to the dictionary?" panels	C-25
Figure C-5	"How do I create a custom dictionary?" panels (automatic branch)	C-29
Figure C-6	"How do I create a custom dictionary?" panels (manual branch)	C-30
Figure C-7	The Index window	C-39
Figure C-8	Matching a search phrase	C-44
Figure C-9	Results of an intersection between two index terms	C-48
Listing C-1	A build file ("Build file SURF.src" file)	C-3
Listing C-2	Events, navigation buttons, and formats (from the "Setup and Access Window.src" file)	C-6
Listing C-3	Prompt sets (from the "Setup and Access Window.src" file)	C-7
Listing C-4	Help menu information (from the "Setup and Access Window.src" file)	C-8
Listing C-5	Access window startup (from the "Setup and Access Window.src" file)	C-9

Listing C-6	Finder version information (from the "Setup and Access Window.src" file) C-11
Listing C-7	Topic areas and topics ("Topic Areas and Topics.src" file) C-12
Listing C-8	Placeholder sequence (from the "Sequence Definitions.src" file) C-15
Listing C-9	Sequence for "How do I use the tools in the toolbar?" (from the "Sequence Definitions.src" file) C-16
Listing C-10	Sequence for "How do I add a word to the dictionary?" (from the "Sequence Definitions.src" file) C-17
Listing C-11	Sequence for "How do I create a custom dictionary?" (from the "Sequence Definitions.src" file) C-19
Listing C-12	Sequence definitions for Huh?, Definition, and Related Topics (from the "Sequence Definitions.src" file) C-20
Listing C-13	Placeholder panel (from the "Panel Definitions.src" file) C-22
Listing C-14	Panels for "How do I use the tools in the toolbar?" (from the "Panel Definitions.src" file) C-22
Listing C-15	Panels for "How do I add a word to the dictionary?" (from the "Panel Definitions.src" file) C-26
Listing C-16	Panels for "How do I create a custom dictionary?" (from the "Panel Definitions.src" file) C-31
Listing C-17	Coachmarks ("CoachMarks SW. src" file) C-37
Listing C-18	Index terms ("Index Entries.src" file) C-40
Listing C-19	Ignore words ("Ignore List.src" file) C-45
Listing C-20	Words on the exception list ("Exception List.src" file) C-46
Listing C-21	Words on the synonym list ("Synonym List.src" file) C-48

Appendix D Checklist D-1

Figure D-1	Tasks required to create a guide file D-1
-------------------	---

About This Book

This book provides all the information you need to successfully produce an Apple Guide help system for your application. By reading it, you'll learn

- Guide Script, an authoring language for creating your help source files
- Guide Maker, the application for compiling (or *building*) your help source files into Apple Guide help files (known as *guide files*)
- authoring tips and suggestions for designing your guide files
- the application programming interface for Apple Guide, containing the functions used to integrate your guide files into your application

In the CD that accompanies this book, you'll find Guide Maker, a sample guide file (SurfWriter Guide), and a searchable command reference. It also contains Read Me files for all specific areas.

Who Should Use This Book

This book is written for people who design and develop guide files and integrate them into their applications. It applies to anyone who falls into one or more of the following categories:

- For instructional designers, it helps you plan and design guide files.
- For scriptors, it provides you with the necessary instruction to begin writing the Guide Script files that describe guide files. It shows you how to create and test guide files using Guide Maker.
- For developers, it shows you how to integrate guide files into general-purpose applications that run under System 7.5 and later.

This book is suitable for instructional designers who have some background in developing reference documentation but little if any programming experience, for scriptors who have some scripting experience but not Guide Script experience, and for developers who have some programming experience but not necessarily Macintosh programming experience.

This book is most suitable for those who are familiar with the concepts and terminology used with Macintosh computers, who have used a Macintosh computer and a few of its applications, and who are aware of the information described in the book *Macintosh Human Interface Guidelines*.

What's in This Book

This book contains four parts and four appendixes. It is designed to be read either from start to finish or in less structured ways. If you are responsible for all tasks associated with creating a guide file (that is, designing, scripting, and developing code), you may want to read the chapters in order. If you have fewer tasks, you may choose to read only the information that applies to them. You can also use this book as a reference, particularly Chapters 3 and 10 and the appendixes.

This book is filled with screen examples that illustrate Apple Guide's features and show how your interface should look. Some screen examples are from the Apple guide files that come with system software: About Help, Tutorial, Macintosh Guide, and Shortcuts. Other screen examples are from SurfWriter Guide, a sample guide file for SurfWriter, a pseudo word-processing application.

The next sections describe the information you'll find in each part of the book.

Designing Your Guide Files

The first part of the book introduces you to the features of Apple Guide. It shows you how to use interface elements in your guide files and describes strategies you can use to plan and design your guide file content before you even begin scripting. In addition, it shows you how to create two powerful search features for your guide files, *Index* and *Look For*.

Building Your Guide Files

The second part of the book introduces you to Guide Maker. It shows you how to build and test guide files using Guide Maker, how to localize them, and how to convert Windows help content into guide files.

Integrating Your Application

The third part of the book describes the Apple Guide application programming interface (API). You can use the functions described in this chapter to integrate your guide files into your application.

Using the Guide Script Commands

The fourth part of the book describes each Guide Script command. For each one, it provides correct syntax, a description of its parameters, a description of the command itself, and examples of its use.

Reference Material

The appendixes provide additional information about the topics described in this book. Appendix A contains a list of abbreviations for all Guide Script commands. Appendix B contains a summary list of all Guide Script commands and their parameters. Appendix C describes SurfWriter Guide, a sample guide file for a pseudo word-processing application, and includes many of its source files. Appendix D contains a checklist you use to help you check off tasks as you create, modify, and localize guide files.

The glossary, which follows the appendixes, defines all common terms that appear in boldface in the book. At the end of the book, you'll find an index that can help you easily locate information about particular topics of interest.

Conventions Used in This Book

This book uses various conventions to present information. Words that require special treatment appear in specific fonts or font styles. Certain information, such as command line options, uses special formats so that you can scan it quickly.

Special Fonts

This book uses several typographical conventions. All code listings, reserved words, and the names of actual data structures, constants, fields, parameters, and routines are shown in Courier (`this is Courier`).

Words that appear in **boldface** are key terms or concepts and are defined in the glossary.

Command Syntax

This book uses the following notation and font conventions to describe Guide Script commands:

[optional]	Brackets indicate that the enclosed language element or elements are optional.
[optional]. . .	Three points of ellipsis (. . .) after a group defined by brackets indicate that you can repeat the group of elements within the brackets 0 or more times.
language element	Plain computer font indicates an element that you must type exactly as shown. If there are special symbols (for example + or &), you must also type them exactly as shown.
<i>placeholder</i>	Italic text indicates a placeholder that you must replace with a real value that matches the definition of the element.

Types of Notes

This book uses two types of notes.

Note

A note like this contains information that is interesting but possibly not essential to an understanding of the main text. ♦

IMPORTANT

A note like this contains information that is essential for an understanding of the main text. ▲

For More Information

APDA is Apple's worldwide source for hundreds of development tools, technical resources, training products, and information for anyone interested in developing applications on Apple platforms. Customers receive the *APDA Tools Catalog* featuring all current versions of Apple development tools and the most popular third-party development tools. APDA offers convenient payment and shipping options, including site licensing.

To order products or to request a complimentary copy of the *APDA Tools Catalog*, contact

APDA
Apple Computer, Inc.
P.O. Box 319
Buffalo, NY 14207-0319

Telephone	1-800-282-2732 (United States) 1-800-637-0029 (Canada) 716-871-6555 (International)
Fax	716-871-6511
AppleLink	APDA
America Online	APDAorder
CompuServe	76666,2405
Internet	APDA@applelink.apple.com

Designing Guide Files

Introduction to Apple Guide

Contents

What Is Apple Guide?	1-3
How the User Views Apple Guide	1-5
The Many Uses of Apple Guide	1-8
A Typical Apple Guide Session	1-9

This chapter introduces you to Apple Guide, a help delivery system that supports the development of onscreen interactive instructions for general-use applications. You should read this chapter if you are designing or scripting Apple Guide guide files. If you are developing code for guide files, you might also find this information useful.

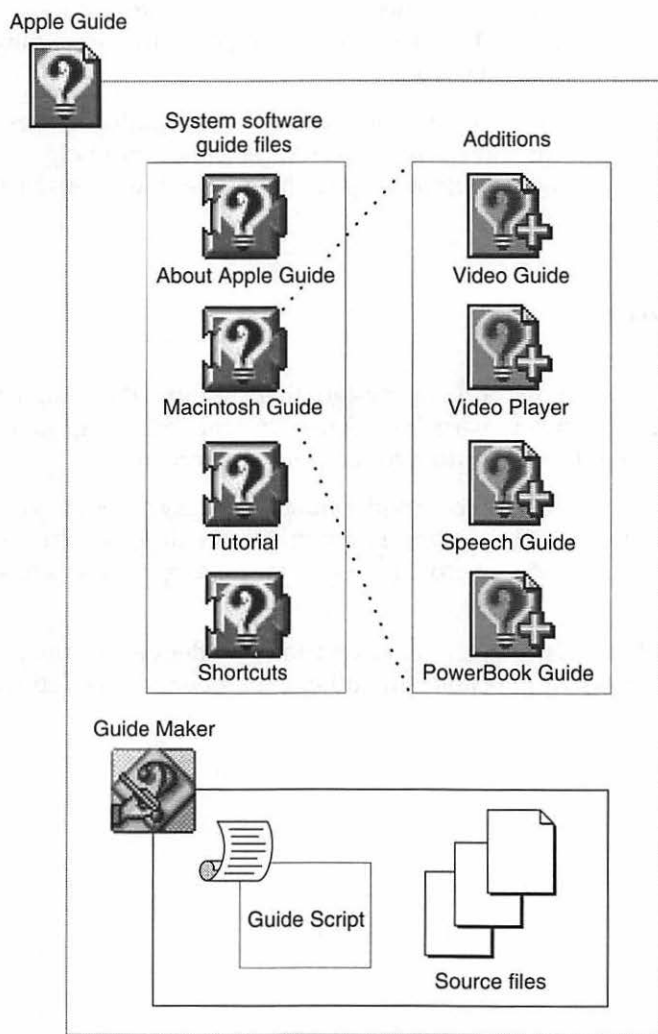
The chapter begins with an overview of the Apple Guide features. It then explains the different ways you can use Apple Guide as a help system. It concludes with a description of a typical help session with Apple Guide.

What Is Apple Guide?

Apple Guide is a powerful help system that supports the design and delivery of interactive onscreen instructions. These instructions are grouped in files (known as *guide files*) according to their particular focus.

For end users, Apple Guide provides quick and easy access to versatile help resources tailored to their needs. For instructional designers, scriptors, and developers, Guide Maker provides a tool for developing comprehensive onscreen help.

Apple Guide consists of a system extension that delivers the help system. Figure 1-1 shows Apple Guide and other components provided with it.

Figure 1-1 The Apple Guide components

These components include

- guide files that describe the Finder and other features of the Macintosh Operating System. Examples include the following:

- ❑ **About Apple Guide** describes the help system provided with the Macintosh computer.
- ❑ **Macintosh Guide** is the main guide file for system software's help system and provides step-by-step instructions for a variety of tasks.
- ❑ **Tutorial** provides training in basic Macintosh skills.
- ❑ **Shortcuts** provides keyboard commands and tips.
- four **Macintosh Guide additions** guide files—Video Guide, Video Player, Speech Guide, and PowerBook Guide—that add content to Macintosh Guide about a specific piece of hardware attached to the Macintosh computer or certain system software features.
- **Guide Maker**, a tool for building and testing your guide files.
- **Guide Script**, an authoring language for developing guide files for your application.

Apple Guide is one of two help systems currently available from Apple. It joins **Balloon Help**, a help system provided in System 7 and higher versions of system software. (If you are not familiar with Balloon Help, see *Inside Macintosh: Macintosh Toolbox Essentials*.) With Balloon Help, users can find out the function or significance of virtually any object on the Macintosh screen, such as icons, windows, and commands. Balloon Help answers the question "What is this item?" With Apple Guide, you can take help to the next level of user inquiry: "How do I accomplish this task?" As a result, Apple Guide does not replace or duplicate Balloon Help but instead expands the help available to users.

How the User Views Apple Guide

Apple Guide leads users to answers through interactive windows (known as *panels*) that explain a concept or task. Apple Guide panels are movable and float on top of other application windows. Users can therefore carry out the help instructions they read in panels, even from within an application, as they work on a given task. You link each of your help topics to a single panel or *panel sequence*, which is a set of related panels that the user can access linearly using left and right navigation arrows. A panel sequence can also contain subsequences (or *panel branches*).

For example, Figure 1-2 shows the first panel that the user views in Macintosh Guide after selecting the topic "How do I display colors or grays?"

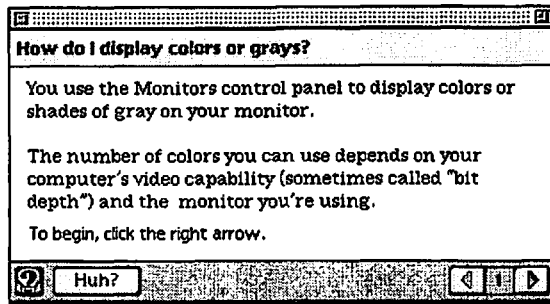
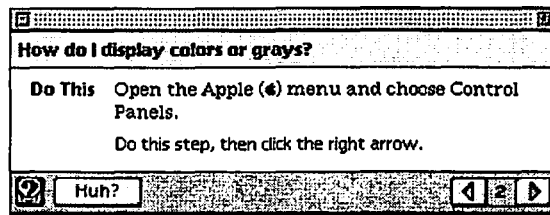
Figure 1-2 A typical Apple Guide panel

Figure 1-3 shows the next panel in that same sequence.

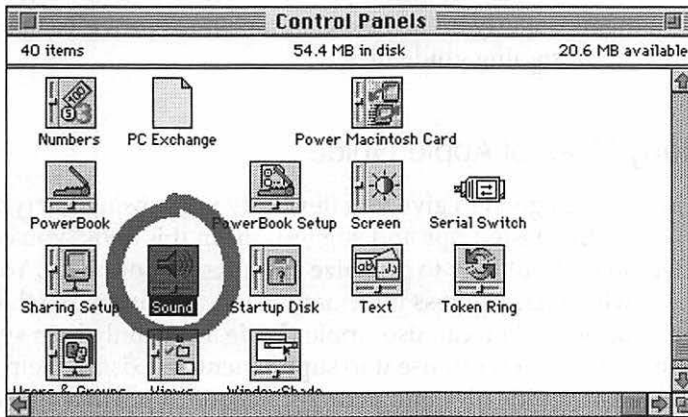
Figure 1-3 A panel that is part of a sequence

Apple Guide automatically provides left and right navigation arrows on panels so that the user can move linearly through a sequence. You can place additional buttons on panels to provide quick access to different parts of the guide file or your application. Along with buttons, you can also place styled text, graphics, and QuickTime movies. To help you create a clear and consistent interface, Apple suggests using standard panel types that apply to specific categories of information.

To further enhance your help instructions, you can identify interface elements to the user using special Apple Guide markers—called *coachmarks*—that circle or point to items in the screen. For example, Figure 1-4 shows a coachmark that appears in Macintosh Guide whenever a panel tells the user to open the Sound

control panel. This particular coachmark draws a red circle around the Sound control panel.

Figure 1-4 The Macintosh Sound control panel circled with a coachmark



In addition to these powerful design elements, you can have Apple Guide present context-sensitive help. If you do, Apple Guide displays help instructions that specifically apply to the user's needs. To provide context-sensitive help, you use commands where programmatically possible to verify whether the user's environment meets a specific requirement or has changed. The functions that check the user's environment are called *context checks*.

For example, you can use context checks with Guide Script commands to have Apple Guide

- skip a panel that tells the user to perform an action when the associated condition is already true (for example, skip a panel that tells the user to open a folder if that folder is already open)
- display a panel that explains how to remedy an error the user made on a previous panel (for example, if the user fails to open a folder, display a panel that tells the user how to open it)
- display a panel only if a certain condition is true (for example, display a panel only if the user has particular software installed)

Introduction to Apple Guide

If programmatically possible, you can also have Apple Guide perform certain actions for the user (for example, use AppleScript to open a control panel for the user). You can have Apple Guide perform a step on a panel only if the user fails to do so or you can create an entire sequence where Apple Guide demonstrates how to perform a task.

You can find out more about the Apple Guide features and learn how to implement them properly in the next chapter, which provides tips and suggestions for designing guide files.

The Many Uses of Apple Guide

Apple Guide is designed to give you flexibility in choosing the type of help you provide. Using the design tips and suggestions in this book, you can make certain decisions about how to organize and present your help. You can choose the order in which users access information and the format in which the information appears. You can use Apple Guide as the only help system for your application, or you can use it to supplement an existing help system. You can also integrate guide files into your application, a practice that Apple strongly recommends.

The Apple Guide guide files can accommodate a diverse range of help requirements. You can pick from several guide file types depending on the form of instruction that you desire. These types include

- orientation to your entire help system
- task-oriented procedures on your application's features and use
- tutorials that guide users through a focused learning path
- advanced or specialized features required by only certain users
- tips and reference material commonly found on quick reference cards

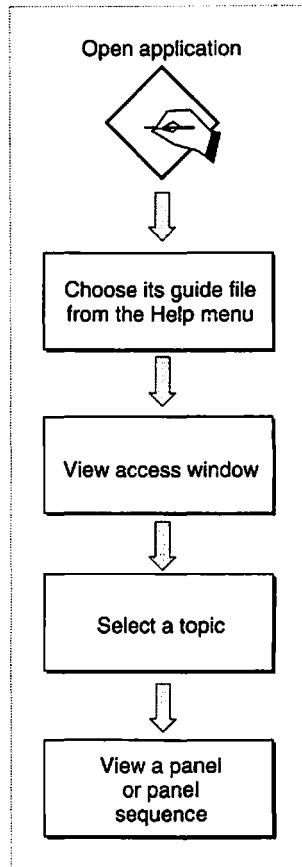
For more about the guide file types, see the section "Designing Guide Files" beginning on page 2-5.

In addition to creating guide files for your application, you can also use them for internal training or as a presentation tool. For example, you can create a guide file that shows employees how to do standard company procedures, such as filling out benefit forms or ordering supplies. Or you can create a guide file that describes a project or plan.

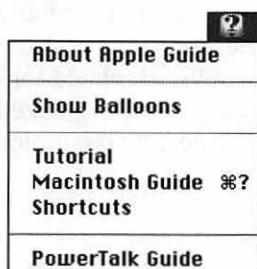
A Typical Apple Guide Session

Apple Guide is easy to use: in general, the user selects a guide file from the Help menu to invoke an *access window* that presents the guide file help topics. Typically, the user picks a topic from the access window and views the panels associated with it.

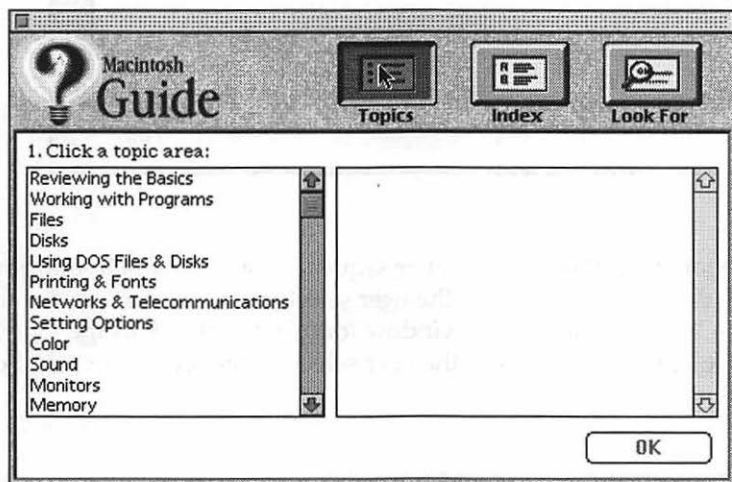
Figure 1-5 shows the typical steps required to view help in Apple Guide.

Figure 1-5 The typical steps a user takes to view an Apple Guide topic

Guide file names appear in the Help menu, along with Balloon Help and any other help systems provided by your application. For example, Figure 1-6 shows a Help menu displaying Macintosh Guide and other guide files. (PowerTalk Guide, a guide file for PowerTalk, appears at the bottom of this menu.)

Figure 1-6 A Help menu displaying Macintosh Guide and other guide files

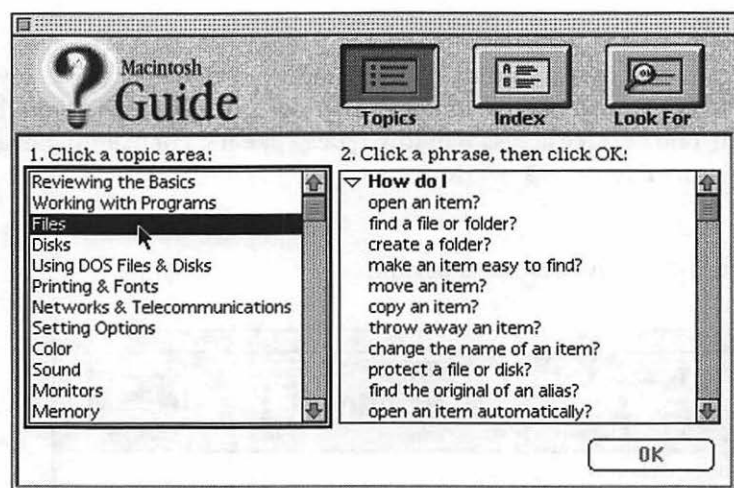
The user can invoke a guide file from the Help menu or from a keyboard shortcut you assign. Each time the user invokes a guide file from the Help menu, one of three access window types appears. For example, Figure 1-7 shows the Full Access window.

Figure 1-7 A Full Access window

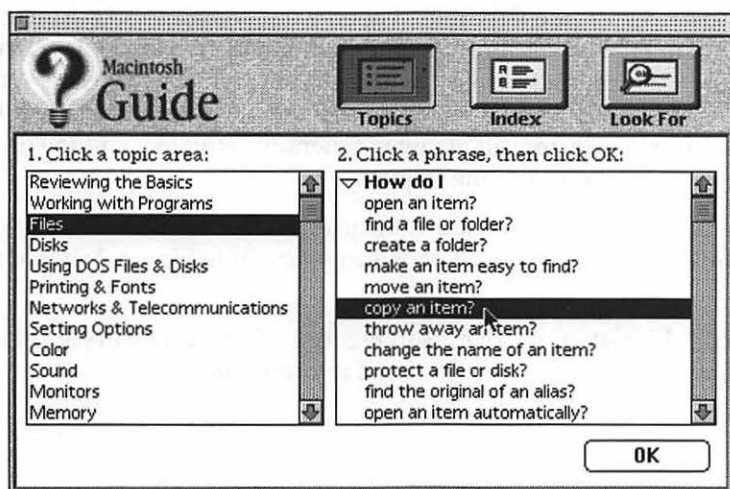
One of these types, the Full Access screen (Figure 1-7) presents topics to the user through three buttons in the upper portion of the window. Each button

provides different search features on the window to accommodate users' varying styles of conceptualizing and searching for information. In Figure 1-7, the Topics button is selected. Note that the left column of the Full Access window displays a list of broad topic categories (or *topic areas*) similar to a table of contents. When the user selects a topic area, a list of related topics appears in the right column of the screen. The associated topics are organized by headings such as "How do I". Figure 1-8 shows the user selecting the topic area "Files".

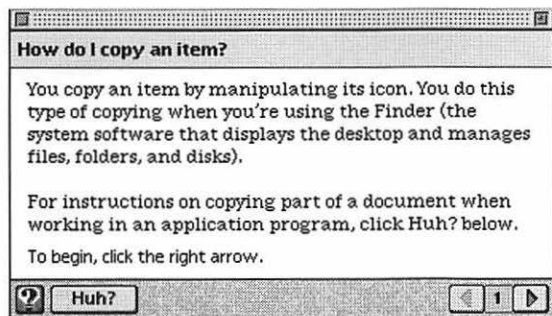
Figure 1-8 A Full Access Window with a topic area selected



Each topic is linked to a panel or sequence containing help information. To view the associated panels, the user selects the topic and presses the OK button in the lower portion of the window (or just double-clicks the topic). For example, Figure 1-9 shows the user selecting the topic "How do I copy an item?"

Figure 1-9 A Full Access Window with a topic area and topic selected

Once the user selects a topic, the related single panel (or first panel in a sequence) appears and the access window is hidden. Each panel typically displays one step in a procedure or one item of information. For instance, a panel can instruct the user to open a menu and select a command. Or it can describe a particular feature, such as an audio CD. Figure 1-10 shows the first panel of the topic “How do I copy an item?”

Figure 1-10 The first help panel for the selected topic

Introduction to Apple Guide

Notice that the lower bar of the panel (the *navigation bar*) contains navigation arrows on the lower-right side and that the lower-left side contains two buttons: GoStart (the lightbulb-shaped icon) and Huh? (the icon containing the word “Huh?”). In this example, the right navigation arrow is active to indicate that the user can click it to go to the next panel in the sequence. A dimmed right arrow indicates that the user has reached the end of the sequence. At the end of a sequence, the left arrow is generally activated, and the user can click it to go backward through the sequence.

To return to the access window, the user can click the GoStart button. This button should appear on all panels in your guide files and should always be active.

If the Huh? button is active, as in Figure 1-10, the user can click it to view another panel containing additional information.

Authoring Tips and Suggestions

Contents

Designing Guide Files	2-5
Using Guide File Types	2-6
Showing Guide Files in the Help Menu	2-7
Designing About Guide Files	2-9
Designing Help Guide Files	2-10
Designing Tutorial Guide Files	2-11
Designing Shortcuts Guide Files	2-13
Designing Other Guide Files	2-14
Designing a Mixin Guide File	2-14
Designing Access Windows	2-15
Designing a Full Access Window	2-16
Designing the Application Logo or Title Area	2-18
Features for a Full Access Window With Topics Selected	2-19
Features for a Full Access Window With Index Selected	2-20
Features for a Full Access Window With Look For Selected	2-21
Designing a Single List Access Window	2-25
Designing a Simple Access Window	2-26
Designing Howdy Text on Access Windows	2-28
Designing Topic Areas and Topics	2-30
Designing Topic Areas and Topics for a Full Access Window	2-31
Designing Topics for a Single List Access Window	2-32
Designing Topics for a Simple Access Window	2-33
Designing Headings	2-33
Designing Panels	2-35
Panel Features	2-37
Designing Panel Prompts	2-39

Designing a Default Prompt Set	2-40
Overriding Default Prompts	2-42
Using the Recommended Panel Formats	2-43
Designing Your Own Panel Format	2-45
Using Graphics in Panels	2-46
Formatting Panel Text	2-47
Providing Navigation Methods on Panels	2-48
Designing Panel Types	2-50
Designing an Introductory Panel	2-51
Designing a Decision Panel	2-52
Designing an Action Panel	2-53
Designing an Information Panel	2-54
Designing a Tip Panel	2-55
Designing a Definition Panel	2-56
Designing a Related Topics Panel	2-58
Designing a Transition Panel	2-59
Designing a Closure Panel	2-60
Designing a Panel Associated With a Huh? Button	2-61
Designing an Oops Panel	2-63
Designing a Continue Panel	2-65
Designing a Sequence	2-66
Designing Branches	2-67
Designing Branches for Mutually Exclusive and Related Tasks	2-68
Designing Branches for a Specific Condition	2-69
Designing Buttons	2-70
Designing Navigation Buttons	2-71
Designing Content Area Buttons	2-72
Using Standard Buttons	2-74
Using Three-Dimensional Buttons	2-74
Using Radio Buttons and Checkboxes	2-76
Designing Hot Text, Objects, and Rectangles	2-77
Designing Coachmarks	2-79
Using Coachmark Types	2-80
Using Coachmark Styles	2-80
Using Context Checks	2-83
Comparison of Oops and Continue Panels	2-85
Analyzing a Sequence for Context Checks	2-85

Evaluation of Context Checks	2-89
Designing for Localization	2-91
Planning for Expanded Text	2-92
Translations for Apple Guide Phrases	2-92
Formats	2-93
Graphics and Buttons	2-94
Sequence Display Titles	2-94
Coachmarks	2-95
Context Checks	2-95
AppleScript	2-95

The best guide files are those that provide the user with a consistent and clear interface. This chapter provides tips and suggestions on how to develop this consistent interface in your files. You should read this chapter if you need to plan, design, and write content for a guide file. If you are scripting a guide file or developing code for it, you should also be familiar with this chapter.

This chapter assumes that you are familiar with the Apple Guide features described in Chapter 1 of this book. You should also be familiar with the general guidelines for Macintosh products as described in the *Macintosh Human Interface Guidelines*.

The chapter describes the requirements for the different guide file types. It explains how to select an access window for your guide file and how to give it the proper look and content. It also shows how to design the guide file panels that contain your help instructions, including how to

- add required features to panels
- format text and graphics on panels
- use standard panel types for different categories of help instructions
- design panel sequences and branches
- design button for panels

In addition, this chapter describes how to use coachmarks to lead the user's attention to screen areas described in help instructions. It then explains how to use context checks to display help instructions more specific to the user's environment. Finally, it provides localization guidelines for translating your guide file into another language.

Designing Guide Files

A **guide file** is a single file containing help content that conforms to one of five guide file types supported by Apple Guide. Once you have developed a guide file, you can create a special guide file (known as a *mixin*) to add to or modify its contents.

Developing a guide file generally requires knowledge of instructional design, scripting with Guide Script, and building and testing guide files with Guide Maker. It also requires familiarity with the design information in this chapter.

Authoring Tips and Suggestions

To integrate a guide file into an application requires knowledge of the Apple Guide application programming interface (API). You might have all the skills and background to develop a guide file yourself, or you might work with a team that includes instructional designers, scriptors, and developers.

To develop a guide file, you first design its help content in hard-copy form using either a flowchart or a storyboard. This process includes planning the guide file topic areas, topics, and associated panels, and determining the required context checks, which are functions you use to have Apple Guide display help instructions specific to the user's environment. You should always verify as soon as possible that it is programmatically possible to create all context checks; if not, return to your guide file design and revise any affected panels.

Next, you describe the guide file's content in **help source files**—files that contain Guide Script commands that define the look, content, and navigation path of all panels in your guide file. You can create help source files in any word processor that stores text as 'TEXT' files (or for which you have an XTND translator). Then you use the Guide Maker application to **build**—that is, compile—the help source files into a guide file and perform testing. At this stage, you can also have users test your guide file to verify its clarity and ease of use. You also need to write the code for any context checks your guide file uses. And, if you choose to integrate the guide file into your application, use the Apple Guide API to do so.

Note

To have your guide file appear in the Help menu, place it in the same folder as its corresponding application (or place an alias to the guide file in the folder containing the application). ♦

This section describes the content, naming, and Help menu location of each type of guide file. It also discusses how to modify an existing guide file's contents using a Mixin guide file. For additional details on producing guide files, see Appendix D.

Using Guide File Types

You should create guide files that conform to Apple Guide's five guide file types: About, Tutorial, Help, Shortcuts, and Other. Each **guide file type** has a particular focus, content, naming convention, and Help menu location.

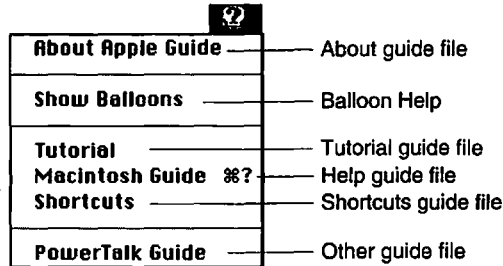
You should follow these suggestions for each guide file type's content and name. Note that the guide file name is the item name in the Help menu.

- **About.** Use an About guide file to introduce users to all available help systems in your application. Its name should include the application name preceded by the word About and followed by the word Guide; for example, About SurfWriter Guide.
- **Tutorial.** Use a Tutorial guide file to lead users through basic features of your application. You should name it Tutorial.
- **Help.** Use a Help guide file to provide users main information in your help system through a range of task-oriented information about your application. Its name should include the application name followed by the word Guide; for example, SurfWriter Guide.
- **Shortcuts.** Use a Shortcuts guide file to provide condensed reference material similar to that found on a quick reference card. You should name it Shortcuts.
- **Other.** Use an Other guide file for highly advanced or specialized information that does not conform to the content conventions of the other four guide types. Or, if you're not the developer of an application, use the Other guide file to ensure that your guide file appears in the Help menu (see the next section for details). The name you choose should indicate the type of help your application provides; for example, SurfWriter Quick Reference.

You can use all five guide file types or only a few of them. You can also use more than one Other guide file. For example, you might have several Other guide files but no Shortcuts guide file. Note, however, that you should always include an About guide file, even if you use only one other guide file type.

Showing Guide Files in the Help Menu

Each guide file type that is available to an application can appear as an item in its Help menu, along with Show/Hide Balloons and any other application help systems. (For information on making a guide file available to an application, see the note on page 2-6.) For example, Figure 2-1 shows the Help menu of the Finder displaying the system software guide files as well as the Show Balloons menu item. Note that PowerTalk Guide is of the guide file type Other.

Figure 2-1 A Help menu showing the system software guide files and Balloon Help

The guide file type determines where it appears in the Help Menu, with certain guide file types appearing closer to the top of the Help menu than others. Specifically, the Other guide file type appears at the bottom of the Help menu and the About guide file type appears at the top of the Help menu, followed by Show/Hide Balloons and the Tutorial, Help, and Shortcuts guide files.

The Help menu displays only one guide file for each of the About, Tutorial, Help, and Shortcuts types. However, there may be multiple guide files for any of these types. When there are, the Help menu displays the guide file that matches all conditions specified by <App Creator>, <Gestalt>, and <Mixin> commands and that comes first alphabetically. These guide files are generally reserved for the developer of the application.

In contrast, the Help menu alphabetically displays *all* guide files of type Other that match all conditions specified by <App Creator>, <Gestalt>, and <Mixin> commands. To ensure that the guide file appears in the Help menu, use an Other guide file to provide help for an application that you did not develop.

Note

You can assign a keyboard shortcut to a guide file. If you do, Apple Guide displays it next to the guide file name in the Help menu. Guide files of type Help automatically open if the user presses the Command-Shift-/combination (which maps to Command-Shift-? key combination on U.S. keyboards) or the Help key on the Apple Extended keyboard, even if you assign no keyboard shortcuts. ♦

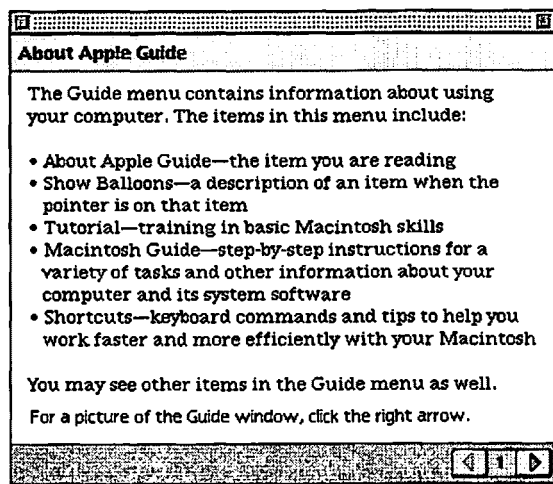
Designing About Guide Files

You should use an **About guide file** to describe the purpose and contents of each help item that appears in the Help Menu—including those that are not Apple Guide guide files—and provide general guidelines for using the help system. The About guide file is typically the first exposure that users have to your help system and the place they are likely to return to if they cannot find certain help information. You can also use this guide file to acknowledge individuals who designed or contributed to the help system. The About guide file should not include information that pertains directly to the application itself.

You should always include an About guide file with your guide files, even if you create only one other guide file. If you don't include one, a default dialog box appears instead.

An About guide file should be brief and can generally use a Simple Access window. For more information, see “Designing a Simple Access Window” on page 2-26. The first panel of an About guide file should identify each guide file in the menu as well as Balloon Help. Figure 2-2 shows the access window for About Apple Guide.

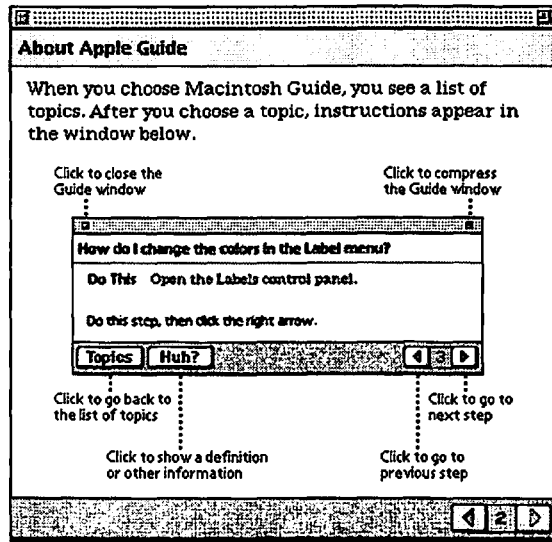
Figure 2-2 The access window for About Apple Guide



Authoring Tips and Suggestions

The subsequent panels in the guide file can provide detailed explanation of each guide file and the rest of the help system. Figure 2-3 shows a panel in About Apple Guide that describes how to use Macintosh Guide.

Figure 2-3 A panel in About Apple Guide



Designing Help Guide Files

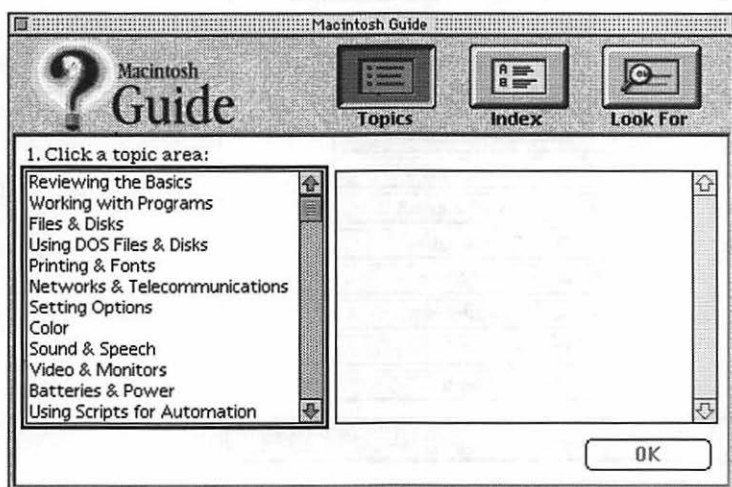
You should use a **Help guide file** to provide the main information for your help system. It should consist primarily of step-by-step instructions that guide users through a range of tasks in your application. (Where programmatically possible, you can also have Apple Guide perform certain tasks for the user.) You can also use this guide file to explain key concepts, define terminology, and address problems that users can encounter using your application. Your Help guide file should generally answer three categories of user inquiry:

- How do I do this task? (For example, how do I save my file?)
- Why can't I do this action? (For example, why can't I print my file?)
- Define this object or concept. (For example, define dithering).

For many users, the Help guide file is the one they turn to first when they encounter a problem during their work. If an application provides a Help guide file, it is the one that automatically appears when the user invokes help. For example, if you use the Help key in the Finder, Macintosh Guide automatically opens.

You should always use the Full Access window for a Help guide file. It provides random access of information, which is more appropriate for the extent and diversity of its topics. It also provides three powerful and varied searching methods that meet different user needs. Figure 2-4 shows the Full Access window for Macintosh Guide.

Figure 2-4 The Full Access window for Macintosh Guide



For more information about this window and the random access method, see "Designing Access Windows" beginning on page 2-15.

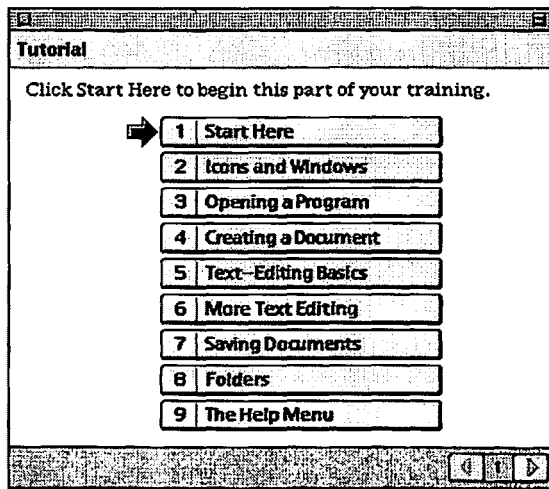
Designing Tutorial Guide Files

You should use a **Tutorial guide file** to lead users through basic use of your application. It is similar to the "quick start sections" commonly found in user

documentation. Unlike the Help guide file, which addresses a diverse range of learning goals, a Tutorial guide file should focus on bringing users to a base level of proficiency in your application. The Tutorial guide file should therefore not include advanced information but instead simply familiarize users with fundamental application features. If your application is a sophisticated publishing tool, for example, your tutorial can guide users through the process of writing and formatting a letter.

Figure 2-5 shows the contents of the Macintosh Tutorial, a guide file that was developed at Apple to describe basic use of a Macintosh computer.

Figure 2-5 The Macintosh Tutorial contents



Once users are more familiar with your application, they can use your Help guide file for more complex procedures or use a more advanced tutorial provided in an Other guide file.

Because a Tutorial guide file should lead users through a particular learning path, you should not use the Full Access window, which allows random access of information. You should instead use either the Simple or Single List access windows to provide sequential access of information. For more information, see "Designing Access Windows" beginning on page 2-15.

You can enhance a Tutorial with buttons that help the user navigate through its different parts. For example, you can provide buttons that

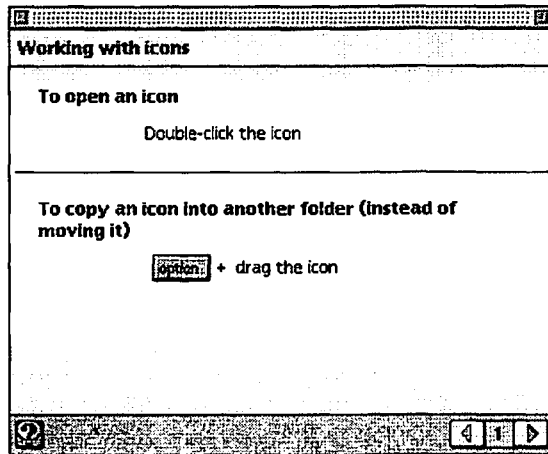
- lead users to panels that contain advanced information or hands-on tasks that are not required to complete the tutorial
- take the user to the beginning of the next procedure
- launch a demonstration of the application from within the tutorial

For more information on creating buttons, see “Designing Buttons” on page 2-70.

Designing Shortcuts Guide Files

You should use a **Shortcuts guide file** to provide users with quick access to condensed information, such as command lists or syntax rules. It is equivalent to the quick reference cards that are often included with applications. Figure 2-6 shows a panel in Macintosh Shortcuts on how to use icons.

Figure 2-6 The Macintosh Shortcuts contents



If the guide file is brief, for example, if it gives shortcuts for a particular task, you can use a Simple Access window. For more information, see “Designing a Simple Access Window” on page 2-26. For larger guide files, for example, one that gives shortcuts for a variety of features, you can use a Single List Access window. For more information, see “Designing a Single List Access Window” on page 2-25.

Designing Other Guide Files

You should use the **Other guide file** to create a guide file that does not conform to the content guidelines for the other four Apple Guide guide files (About, Tutorial, Help, and Shortcuts). You might also use an Other guide file to create a version of the other four guide file types that is particularly advanced or specialized, for example, a Tutorial that teaches high-level features of your application. Or, if you’re not the developer of an application, use the Other guide file to ensure that your guide file appears in the Help menu (see “Showing Guide Files in the Help Menu” on page 2-7).

Note

If you provide help for an application that you did not develop, Apple recommends that you use an Other guide file so that it appears in the Help menu. For more information, see “Showing Guide Files in the Help Menu” on page 2-7. ♦

If you are using the Other guide file to create a more advanced version of one of the other four guide file types, use as similar an interface as possible.

Designing a Mixin Guide File

You should use a **Mixin guide file** (also called a *mixin*) to revise the contents of your main guide file. A **main guide file** is any guide file containing help content that you can modify using a mixin. A Mixin guide file can either add content to a main guide file or change its content. In either case, you can use a mixin to insert topic areas, topics, and index entries in the guide file, and to create sequences and panels for topics that you add or modify.

Typically, you should use a Mixin guide file to describe software and hardware features that are specific to a certain condition or version. For example, you should use a Mixin guide file to describe

- features that are particular to a specific model of Macintosh computer (for example, to document features that apply only to a portable Macintosh computer)
- different software versions of the same application (for example, to expand a guide file for an earlier version of an application)
- features that are particular to or dependent on a specific piece of hardware attached to the Macintosh computer
- features specific to or dependent on certain system software features (for example, QuickTime)

IMPORTANT

You should never use a mixin to modify a guide file that you did not develop. This includes Macintosh Guide and any of the other guide files provided with system software. ▲

Designing Access Windows

Every guide file must provide an **access window**, which appears whenever the user selects a guide file from the Help menu. From this access window, the user selects (or goes directly to view) help topics.

You can choose from three types of access windows: Full Access, Single List Access, and Simple Access. The features of the windows range in complexity; the Full Access window provides several built-in access methods, the Single List access window provides one built-in access method, and the Simple Access window provides only standard navigation arrows unless you add your own access method. In general, follow these suggestions for selecting the appropriate access window for your guide file.

- Use the Full Access window if your guide file has more than 20 topics and must contain subtopics. You should also use it if you plan to incorporate the Apple Guide Look For and Index features described later in this section. The **Full Access window** provides three built-in buttons—Topics, Index, and Look For—from which the user makes selections or enters a search phrase.

Authoring Tips and Suggestions

- Use the Single List Access window if your guide file has more than 7 but fewer than 20 topics. The **Single List Access** window provides a single scrollable list of topics.
- Use the Simple Access window if your guide file has less than seven single-level topics. The **Simple Access** window takes the user directly to the help information via standard navigation arrows. Alternatively, you can provide your own access route (for example, a set of buttons).

Note that your ability to control how the user views the guide file content differs among the three access window types. The Full Access window provides only **random access** to information—that is, users can pick help topics in any order and skip topics of no interest. Random access is appropriate for guide files that address diverse goals and levels of expertise (for example, the Help guide file). With the Single List and Simple Access window types, you can also provide **sequential access** to information; in other words, they present topics to users in a structured order. Sequential access is appropriate when you want to direct the user through a specific learning exercise, as in a Tutorial guide file. You can enforce sequential access in the Simple Access window but not in the Single List access window. Apple recommends that you simply encourage sequential access rather than enforce it strictly. For example, you can let users access the topics in any order but provide a message indicating that they have accessed information out of order.

Each access window has certain default features, provided by the Guide Script command that creates the access window. For example, the Full Access window always includes the Topics, Index, and Look For buttons. You also need to provide certain features using Guide Script commands. Optionally, you can also include instructions about the guide file, known as *howdy text*. Apple recommends that you use howdy text only under certain conditions, which are described later in this section.

This section provides suggestions for using each of the three access window types and for using howdy text.

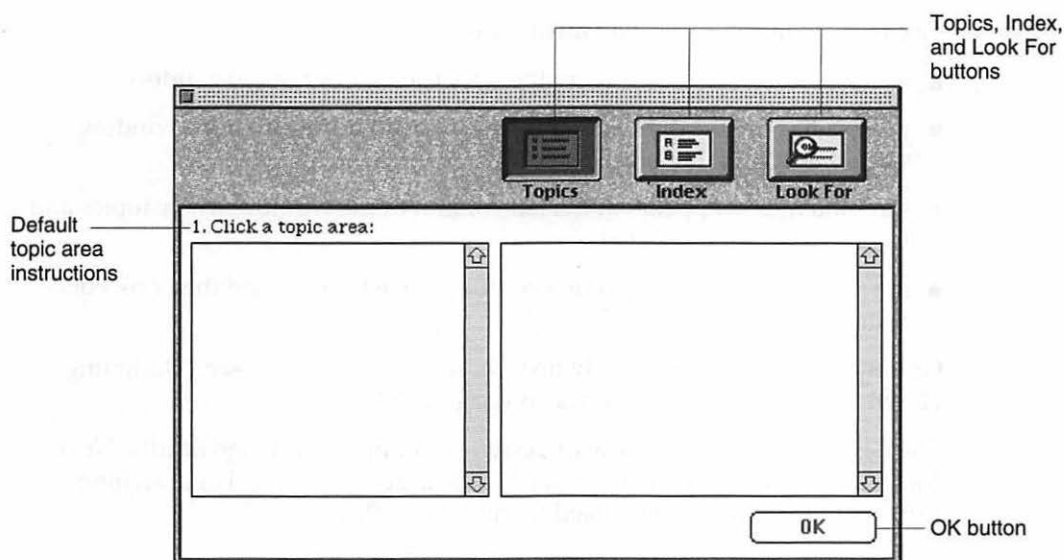
Designing a Full Access Window

Of the three access window types, the Full Access window is the only one that provides cross-referencing features as well as three search methods to accommodate the different ways users can conceptualize help information. You should always use the Full Access window for Help guide files and for Other guide files resembling Help guide files in content, for example, an Other guide

file containing advanced help. You should also use it for any guide file with a large number of topics (generally over 20) that require extensive searching by the user.

Figure 2-7 shows the default features of the Full Access window.

Figure 2-7 A Full Access window with default features



Note that this window automatically provides a two-column format. The left column varies according to which three search methods, Topics, Index, or Look For features, is selected, and the right column displays the topics derived by that method. By default, the window also provides

- the Topics, Index, and Look For buttons in the upper-right portion of the window (for example, Figure 2-7 shows the screen with Topics selected)
- Topic, Index, or Look For instructions, above the left and right columns of the window, that tell the user how to use respective features

Authoring Tips and Suggestions

- a standard OK button, in the lower-right corner, that the user clicks to view a selected topic (alternatively, the user can view the topic by double-clicking it)

You cannot modify any of these default features except the Topics, Index, and Look For instructions. In general, Apple recommends that you use the default instructions; if you replace them with your own versions, keep the text as short as possible. Apple Guide automatically wraps the instruction text to a second line, if necessary.

To complete the window, you must create

- an application logo (or title) in the upper-left corner of the window
- the headings and topics that appear in the right column of the window when the Topics, Index, and Look For features are selected
- the content that appears in the left column of the window when Topics and Index features are selected
- the strategy that Apple Guide uses to retrieve topics when the Look For feature is selected

Optionally, you can add howdy text (for more information, see “Designing Howdy Text on Access Windows” on page 2-28).

The next section explains how to design your application logo or title. Next come sections that explain how the Topics, Index, and Look For searching features work and how you need to implement them.

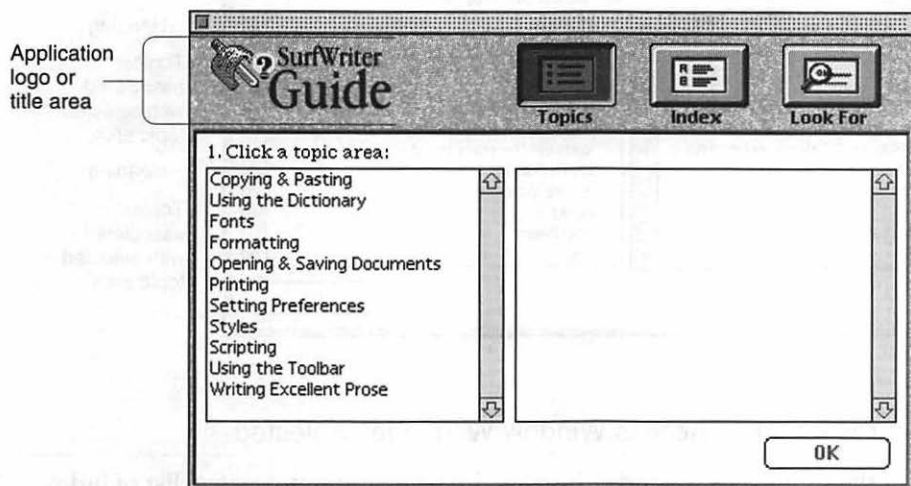
Designing the Application Logo or Title Area

Your application logo or title must fit in the available space of 59 by 185 pixels. You should

- provide a leatherette background in the application logo area.
- include with your logo a small version of the Apple Guide icon (a yellow lightbulb).
- display your application’s name and the word “Guide” in any font. You might prefer a font that is identified with your company or application. For example, in Macintosh Guide, Apple displays the guide file title in Apple Garamond.

Figure 2-8 shows the Full Access screen for SurfWriter Guide with an application logo. (SurfWriter Guide is a sample guide file that is used as an example throughout this book.)

Figure 2-8 A Full Access window with an application logo and title

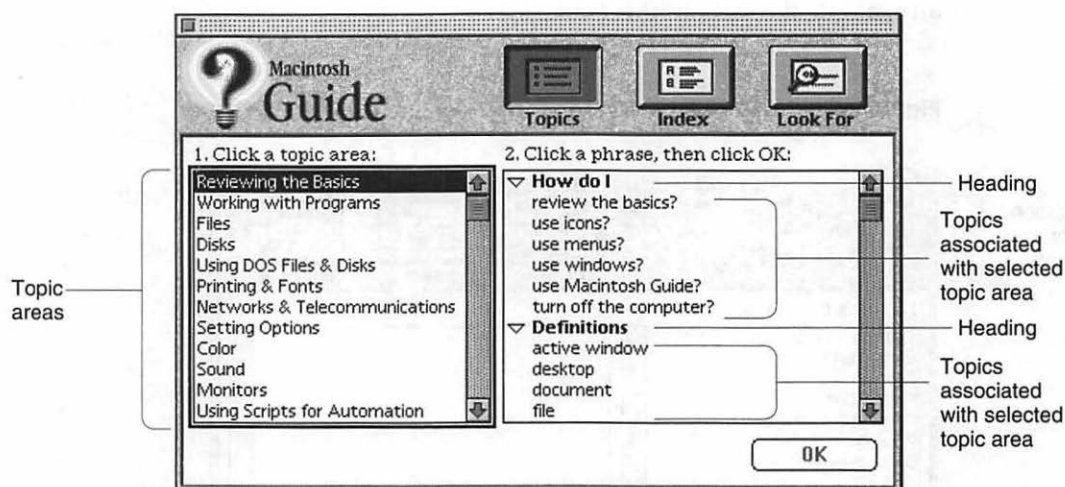


Note

Guide Maker provides a file, Standard Resources, that contains templates (two 'PICT' resources with IDs 501 and 502). You can use these templates to create your application logo picture. ♦

Features for a Full Access Window With Topics Selected

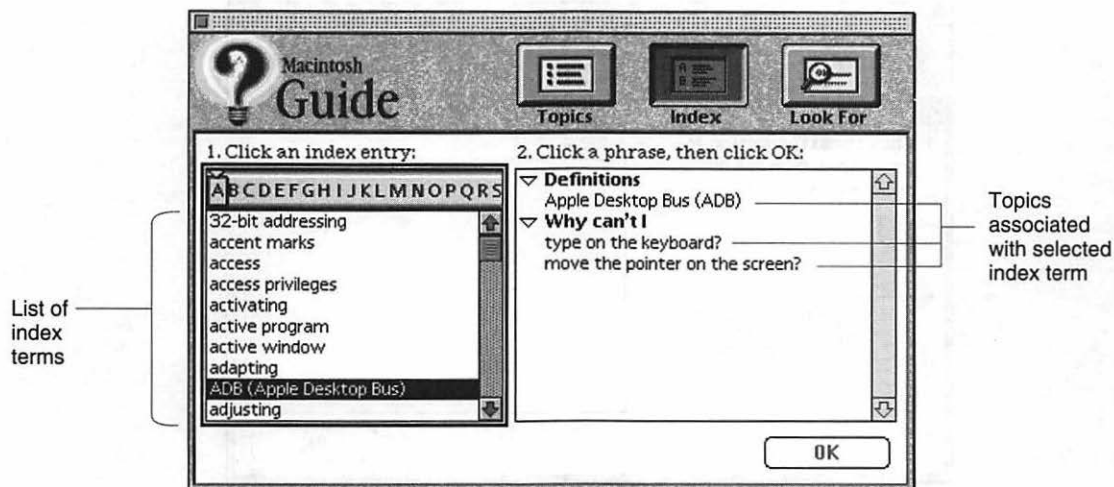
When the user clicks the **Topics button**, the left column of the window displays the guide file's topic areas. These are broad categories of help that subsume one or more topics. When the user selects one, a list of related topics, organized under headings, appears in the right column of the window. You need to provide a list of topic areas in the left column and, for each topic area, a list of topics in the right column, as shown in Figure 2-9. See "Designing Topic Areas and Topics" beginning on page 2-30 for more information. You also need to provide the headings for the topics. See "Designing Headings" on page 2-33.

Figure 2-9 A Full Access window with the Topics features selected

Features for a Full Access Window With Index Selected

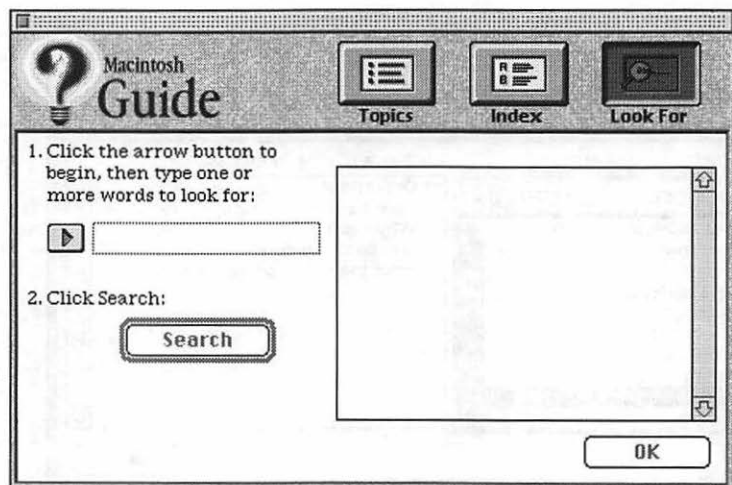
When the user clicks the **Index** button, the left column presents a list of index terms for the guide file. The user can select a term by scrolling through the entire index or by using the Alpha slider above the column to view the terms beginning with the selected letter. When the user selects a term, a list of topics related to that particular term appears in the right column.

You need to provide a list of index terms in the left column and the headings and topics associated with the index terms in the right column, as shown in Figure 2-10. For more information, see “Creating a Guide File Index” beginning on page 3-20.

Figure 2-10 A Full Access window with Index features selected

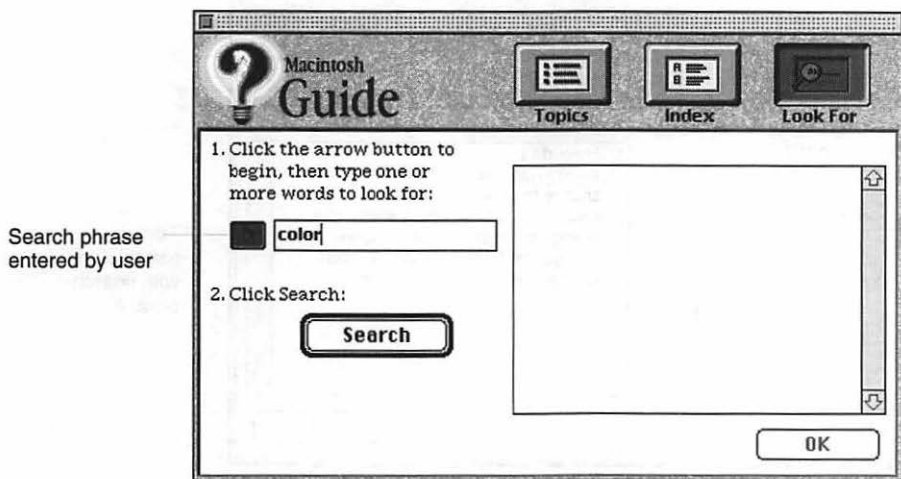
Features for a Full Access Window With Look For Selected

When the user clicks the **Look For** button on the Full Access window, the left column of the Full Access window provides a search phrase entry box where the user can enter a search phrase. Note that the search phrase entry box first appears in dotted lines with the Search button dimmed, as shown in Figure 2-11.

Figure 2-11 A Full Access window when Look For is selected

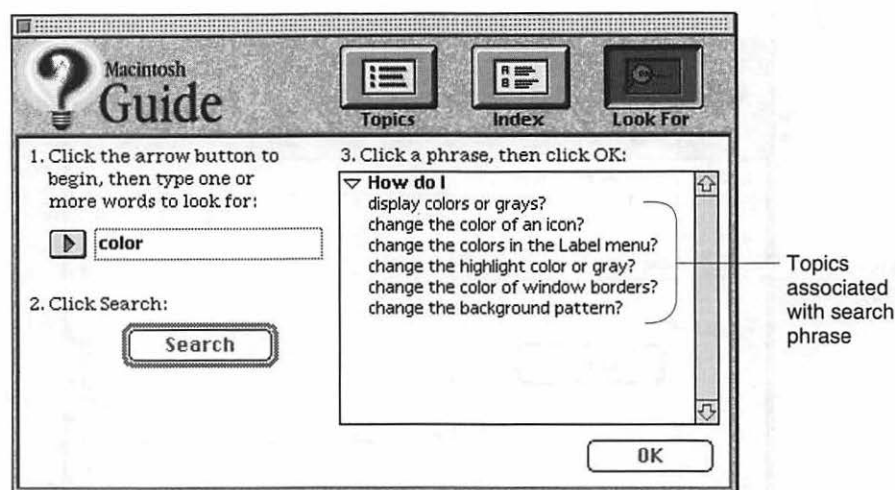
When the user clicks in the search phrase entry box, its dotted lines become solid and the Search button becomes active. The user can now enter a search phrase, as shown in Figure 2-12.

Figure 2-12 A Full Access window with Look For features and the cursor in the search phrase entry box



(Alternatively, the user can first click the arrow to the left of the search phrase entry box and then enter the search phrase; the cursor appears in the box.)

After the user enters a search phrase and clicks the Search button, the right column of the window lists all help topics associated with the particular phrase, as shown in Figure 2-13.

Figure 2-13 A Full Access window after a Look For search is completed

You need to determine the list of headings and topics that appear in the right column when the user enters a search phrase in the left column. See “Helping the User Search” on page 3-12 for complete instructions. In general, you should design the Look For feature so that users can receive a list of topics if they enter any of the following as a search phrase:

- multiple words
- an exact topic name
- slang, acronyms, abbreviations, or synonymous terms for a topic

Note

If you design the Look For feature so that it successfully retrieves topics only if the user enters a single search word, be sure to change the default Look For instruction to say “Click the arrow button to begin, then type one word to look for:” ♦

Designing a Single List Access Window

The Single List Access window displays a single scrollable list of topics. Unlike the Full Access window, it does not provide topic areas or index and search features. You should therefore use the Single List Access window for a guide file that provides a small number of focused topics (typically between 7 and 20). Good examples are Tutorial or Shortcuts guide files.

By default, the window provides

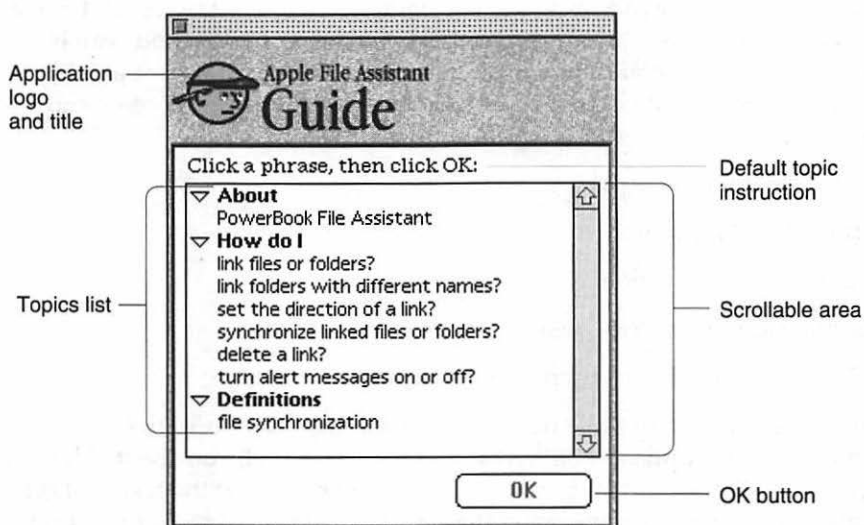
- a scrollable list for topics
- a default topic instruction
- an application title or logo area
- an OK button that the user can click to view a selected topic

You cannot modify any of these default features except for the Topics instruction and the application title or logo area. In general, you use the default instruction; and if you replace it with your own version, keep the text as short as possible. Apple Guide automatically wraps the instruction text to a second line, if necessary. For suggestions on creating an application logo or title, see “Designing the Application Logo or Title Area” on page 2-18.

To complete the window, you must provide the topics that appear in the scrollable column. If your topics fall into logical groups, organize them under headings. For more information, see “Designing Topic Areas and Topics” beginning on page 2-30.

Optionally, you can add howdy text. For information on adding howdy text, see “Designing Howdy Text on Access Windows” on page 2-28.

Figure 2-14 shows an example of a Single List Access window for a guide file. In this example, the window includes an application logo.

Figure 2-14 A Single List Access window

Designing a Simple Access Window

The Simple Access window is actually a *panel*, which is the window type used for all help instructions. Unlike the Full and Single List access windows, it does not provide a built-in access route. Use it to take users directly to help information via the standard navigation arrows or to provide an access method yourself. Because of its simplicity, use the window for particularly brief guide files (fewer than seven topics), such as an About or Shortcuts guide file.

By default, the window provides a title area, content area, and navigation bar. (For more information, see “Designing Panels” beginning on page 2-35.) There are three ways to fill in a Simple Access window. You can put in

- **your own access route (for example, buttons users can press to access topics in your guide file)**

You should use three-dimensional buttons instead of radio buttons because they are easier to read (see Figure 2-15 for an example). For more information, see “Designing Buttons” beginning on page 2-70.

- **the first topic of your guide file**

By placing the first help topic of your guide file in this window, you provide help instantly, but the user must view the entire guide file to discover all its topics (see Figure 2-16 for an example).

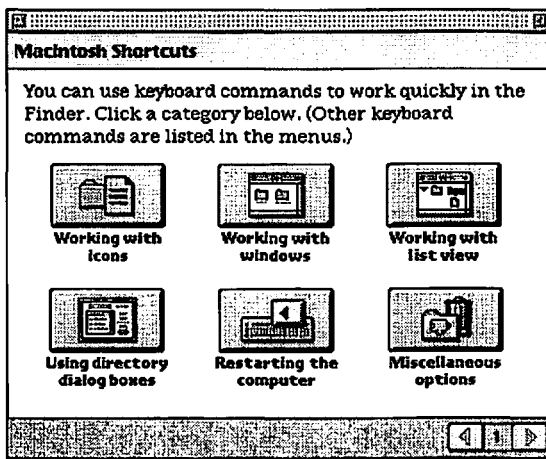
- **a description of the topics in your guide file**

By introducing your topics first, the user can instantly know the guide file contents (for example, you can briefly describe each topic in a list).

For more information on creating topics, see “Designing Topic Areas and Topics” beginning on page 2-30.

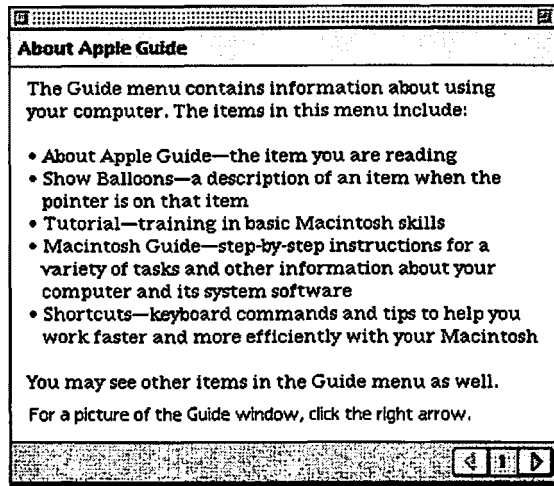
The Simple Access window for Macintosh Shortcuts, shown in Figure 2-15, uses three-dimensional buttons. The user selects any of these to access a help topic.

Figure 2-15 A Simple Access window with three-dimensional buttons



In contrast, the Simple Access window for About Apple Guide, shown in Figure 2-16, presents the user directly with help information.

Figure 2-16 A Simple Access window that takes the user directly to the help information



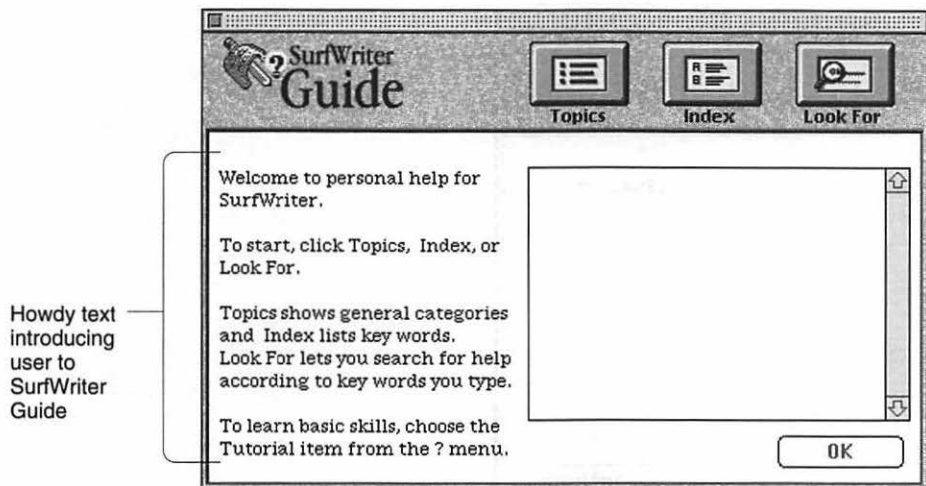
Designing Howdy Text on Access Windows

Howdy text is text that you can place on the access window of a guide file to describe its contents. It appears the first time the user invokes the guide file and does not appear again unless the user restarts the computer.

Apple recommends that you use howdy text to convey only crucial information to the user. For example, if you include both a general and an advanced help guide with your application, you can use howdy text to indicate the expertise level for each of them. You can also use howdy text to welcome the user to the help guide file and give initial instructions on its use.

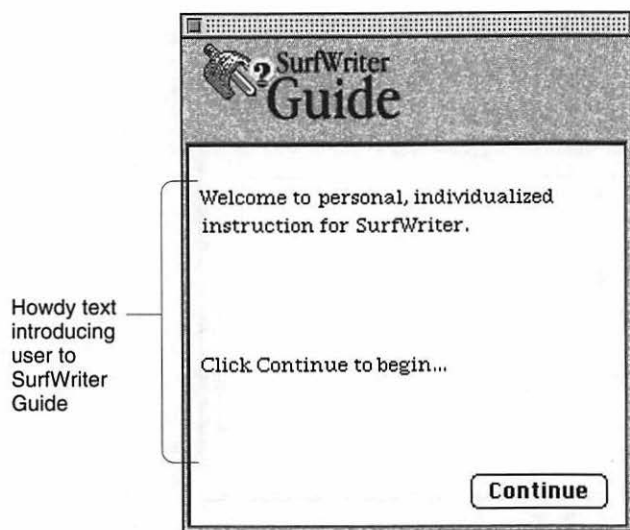
You use a Guide Script command to place howdy text on the Full and Single List access windows. Although you cannot use this command to place howdy text on a Simple Access window, you can use panel text for the same purpose.

Figure 2-17 shows howdy text that appears in the Full Access window of SurfWriter Guide.

Figure 2-17 A Full Access window with howdy text

You should place howdy text in the left column of the window. If the user clicks the Topics, Index, or Look For button, the howdy text disappears and the window shows the features associated with the button.

Figure 2-18 shows a Single List Access window that includes howdy text.

Figure 2-18 A Single List Access window with howdy text

Notice that the window includes a logo in the title area section, howdy text instead of a single list of topics, and a Continue button on the lower right portion of the window. The user must click Continue to view the single list of topics.

Designing Topic Areas and Topics

A **topic area** is a broad category of help that subsumes one or more topics and that appears in the left column of the Full Access window only when Topics features are selected. A **topic** is a category of help information that the user selects from an access window. Here is the way that topic areas and topics appear in Full Access, Single List Access, and Simple Access windows:

- In the Full Access window with Topics selected, topic areas appear in the left column and individual topics appear in the right column (see Figure 2-9). With Index or Look For selected, no topic areas appear in the left column but topics appear in the right column.

- In the Single Access window, topics appear in a single scrollable column.
- In the Simple Access window, topics can appear in an access route that you provide, or the user can go directly to topics.

When creating topics for the Full Access window, you should organize your topics under headings. You can also use headings in the Single List Access window if it contains multilevel topics. For the Full Access and Single List access windows, try to minimize the number of topics (and topic areas) so that the user doesn't have to scroll to view an entire column.

The sections that follow explain how to create topics for each access window type and how to use headers.

Note

If you add a Mixin guide file to your guide file, its additional topics (and topic areas) appear among the existing topics (and topic areas) in the guide file. You can integrate topic areas and topics of the mixin into those of your guide file using Guide Script commands. For more information on the <Mixin> command, see the chapter "Guide Script Command Reference." ♦

Designing Topic Areas and Topics for a Full Access Window

In the Full Access window, you must provide both topic areas (for when Topics features are selected) and topics.

Your list of topic areas should be a logical outline of the guide file contents, similar to the table of contents for a book. Keep topic area names short—no more than 31 characters—and use initial capitalization for each main phrase. For example, for a word-processing application, you could use such topic areas as "Files" and "Fonts and Formatting". Apple Guide lists the topic areas in the order in which they appear in your source files. You should order the topic areas by importance, so that the more important ones appear at the top of the column.

For your main help instructions, topic names should form a complete question or statement from the user's point of view, for example, "How do I open my folder?" You should therefore use the first person (for example, "I" and "my") for topic names using this form. Use a heading to provide the standard text of the question or statement, for example, "Why can't I" (See "Designing

Authoring Tips and Suggestions

Headings” on page 2-33 for headings recommended by Apple.) Beneath the heading, complete the statement or question with a phrase that describes the specific topic (for example, under the heading “How do I”, place the phrase “change the color of my icon?”). The phrases that appear beneath the heading should be short and begin with a lowercase letter. Note that the entire topic name (heading and phrase) should appear in the sequence display title areas of the associated panels, not just the topic phrase.

For less complex topics, you can use headings that do not form a complete question or sentence. For example, use the heading “Definitions” followed by a list of terms that begin in lowercase (such as “font”).

The topic name should always focus only on the main goal that the user wants to achieve and not on any choices associated with that goal. Choices should instead appear on decision panels that precede branches in the sequence. For example, with the Macintosh computer you can create your own desktop pattern or use an existing one. To avoid a long and confusing topic name—for example, “How do I change the desktop pattern by using an existing one or creating one myself?”—Macintosh Guide instead uses the topic name “How do I change the desktop pattern?” After selecting the topic, the user can choose one of two branches from a decision panel: one branch for creating a unique desktop pattern and the other branch for choosing an existing pattern. For more information, see “Designing Branches” beginning on page 2-67.

Apple Guide lists the topics in the order in which they appear in your guide files. In contrast to topic areas, you should order topics by frequency of use rather than importance, so that the more frequently used topics appear at the top of the column.

Designing Topics for a Single List Access Window

You should keep topic names in a Single List Access window short and direct, just as you do in a Full Access window. If your topics are multilevel, you can organize them as questions or statements under headings so that they resemble the right column of the Full Access window. For guidelines on forming topic names with headings, see the previous section. If you list topics with no headings, begin each topic with an uppercase letter (for example, “Formatting”). You can also add a number listing as part of the topic name, for example, for a Tutorial guide file, you can number each tutorial procedure to encourage sequential access).

Apple Guide lists the topics in the order in which they appear in your guide files. If you present your topics with headings, put the most frequently used topics at the top of the column. If you are not presenting topics this way, order them alphabetically. If you use numbers, order the topics sequentially.

Designing Topics for a Simple Access Window

If you create your own access route for a Simple Access window using buttons, you should use topic names that fit in the space allocated for a three-dimensional button, which is the button type that Apple recommends for this purpose. The graphics you place in the buttons should clearly convey the topic content and should be easily distinguished from one another. (For an example, see Figure 2-15.) If your topics require longer names for clarity, you can instead use radio buttons, whose titles accommodate more characters. For guidelines on both button types, see “Designing Content Area Buttons” on page 2-72.

If you take the user directly to the help instructions by using the navigation arrows, you can describe the guide file topics by placing panel text on the Simple Access window. This way, the user does not have to navigate through the entire guide file to determine its content. Alternatively, you can design the window as a standard help panel, in which case it contains the first topic in the guide file.

Designing Headings

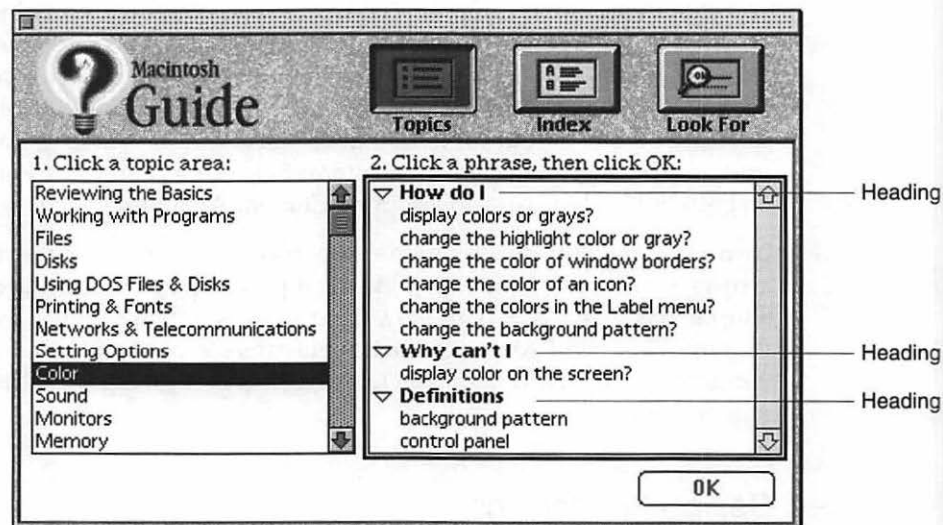
For the Full Access window, organize your topics under headings. You can also use headings for a Single List Access window if it contains multilevel topics. For the Full Access window, Apple recommends these headings: How do I, Why can't I, and Definitions.

- Use the **How do I** heading for topics that show the user how to accomplish a task (for example, “How do I create a custom dictionary?”).
- Use the **Why can't I** heading for topics that explain why the user cannot perform a certain action (for example, “Why can't I print a file?”).
- Use the **Definitions** heading to define terms that relate directly to the selected topic area (for example, if the user selects the topic area “Using the Dictionary”, the term “custom dictionary” could appear under the “Definitions” heading). You can also provide definitions in your guide file that do not directly relate to a particular topic area; in this case, they should

not appear under the “Definitions” heading. For more information, see “Designing a Definition Panel” on page 2-56.

For example, Figure 2-19 shows the Full Access window for Macintosh Guide, with some of these headings appearing in the right column.

Figure 2-19 Topic headings on the Full Access window



Except for the heading “Definitions”, these headings provide the standard text for a question or statement from the user’s point of view. You can complete the statement or question with a phrase that describes the specific topic. (For more information, see “Designing Topic Areas and Topics for a Full Access Window” on page 2-31.) If you create your own headings, you should follow this same convention for your topic names.

You should place all topics under headings, even if some headings contain only a single topic. For example, do not use the “How do I” heading for your task-oriented instructions and then omit the “Definitions” heading for your definitions; otherwise, the user might not easily identify the category of help provided by each topic. If you create a heading but place no topics under it, Guide Maker automatically eliminates it when you build a file.

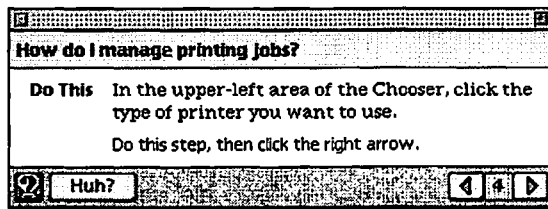
If possible, at least one heading should be visible in the topics column at all times.

Designing Panels

Whenever the user selects a topic from an access window, a presentation panel appears containing a help instruction. A **presentation panel** is an Apple Guide help window you use to describe a single concept or step. It is referred to here as a *panel*.

Figure 2-20 shows a typical panel in Macintosh Guide.

Figure 2-20 A typical panel in Macintosh Guide



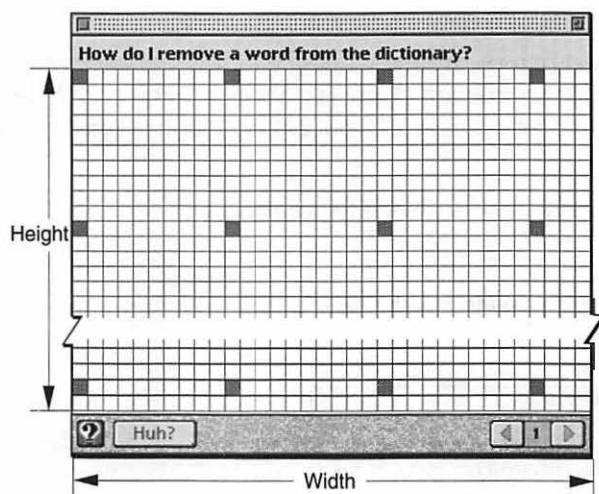
Notice that the panel focuses on only one issue and includes no actions or additional details about a related topic.

When you create a panel, it automatically comes with several features, including navigation arrows; a panel number; and reserved areas for the panel title, help instructions, prompt, and additional controls. You complete the reserved areas and add your own features to the panel following the guidelines in this chapter. The panel has a default layout and text format; you can, however, override these default settings, if necessary.

When you design a panel, it should conform to one of the standard panel types recommended by Apple. Each of these types applies to a different category of help instruction. For example, Figure 2-20 shows an action panel, which is the panel type you use to show a step in a procedure.

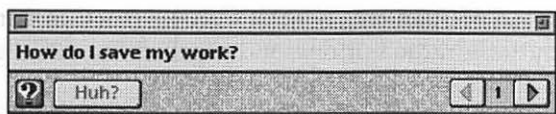
Figure 2-21 shows a panel with a pixel grid that is 341 pixels wide. Each box represents a 10-by-10 pixel square, and a filled box appears every 100 pixels.

Figure 2-21 A pixel grid depiction of a panel



The total width of the panel, including the window frame for a System 7.5 window, is 344 pixels, and the panel width within the window frame is 341 pixels. The **panel width** is a fixed measurement of 344 pixels, which you cannot change using Guide Script commands. The **panel height** is the distance between the title area (the top bar on the panel) and the navigation bar (the lower bar on the panel). Apple Guide automatically resizes the panel height to accommodate the information that you place in it according to limits that you set with Guide Script commands. You should try to keep the height of each panel as short as possible so that the user can easily keep it onscreen while performing a task.

By default, a panel always first appears in the lower-right portion of the screen. Because panels are movable and float on top of other application windows, users can easily view them while they carry out help instructions. A user can also minimize the panel to make it take less room on the screen. If minimized, the panel height becomes 0, as shown in Figure 2-22.

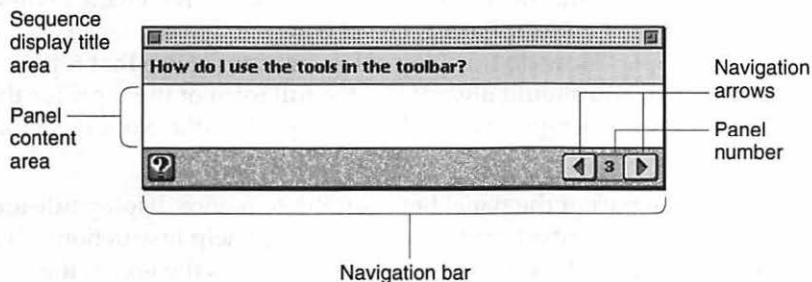
Figure 2-22 A minimized panel**Note**

If a topic consists of more than one action or concept, you should generally create additional panels; do not place excessive information on a single panel. For more information, see “Designing a Sequence” beginning on page 2-66. You should place more than one concept or action on a single panel under only two circumstances: to tell the user to perform several actions in the same place or to present an alternative way of doing the same task. ♦

The next section describes the panel features that Apple Guide provides. It shows how you can create your own designs and also explains different navigation methods you can provide on your panels.

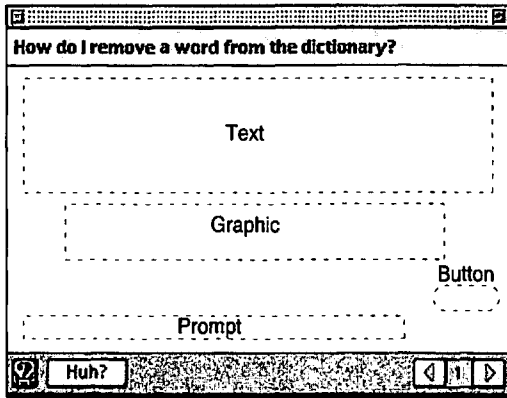
Panel Features

By default, panels have the features shown in Figure 2-23.

Figure 2-23 A panel with default features

These features include a title area, a content area, and a navigation bar, all of which you need to complete. In addition, you can place on a panel text and objects, including styled or plain text, control features (such as standard and three-dimensional buttons, radio buttons, and checkboxes), PICT graphics, and QuickTime movies, as shown in Figure 2-24.

Figure 2-24 A panel design with text, graphic, and button



You should design your panels to adhere to certain types designed by Apple. For more information, see “Designing Panel Types” on page 2-50.

The **sequence display title area** is the bar, in the upper portion of the panel, that contains the panel title. The title should match the corresponding topic that appears in the access window. The title of panels should not change within a sequence. In Macintosh Guide, for example, all panels in the sequence associated with the topic “How do I display colors or grays?” use that topic name in their title area. You should always use the full form of the topic for the panel title, regardless of its length. Apple Guide wraps the title text if it is more than a single line.

The **content area** is the part of the panel between the sequence display title area and the navigation bar (described next) where you place help instructions. This can include text and objects. By default, Apple Guide places the text in the content area in one column. For certain panel types, you override the default format with Guide Script commands and create a two column format; for more

information, see “Formatting Panel Text” on page 2-47. To place buttons on the content area, see “Designing Content Area Buttons” on page 2-72. To place graphics on the content area, see “Using Graphics in Panels” on page 2-46.

You should use active voice for all help instructions and keep them as concise as possible. Where possible, you should also keep panels in the same sequence close to the same size by using a similar amount of information on each one; in this way, the window size does not excessively shrink or expand as the user moves through the sequence.

By default, Guide Maker also allocates space in the content area for you to provide a *prompt*, which provides navigation and other instructions for the user. For more information, see “Designing Panel Prompts” on page 2-39.

The **navigation bar** is the bar, on the lower portion of the panel, that always displays the left and right navigation arrows the user clicks to move between panels. For each panel, Apple Guide makes the right navigation arrow active or dimmed according to whether the user can navigate to the following panel. Apple Guide makes the left navigation arrow active or dimmed according to whether the user can navigate to the previous panel. Guide Maker automatically assigns a number to each panel (the **panel number**) in a dynamic sequence. This number appears between the left and right navigation arrows.

You can add up to three additional navigation buttons to the navigation bar. You should always place the GoStart button on the navigation bar of all panels in your guide files. For more information on adding navigation buttons, see “Using Context Checks” on page 2-83.

Designing Panel Prompts

A **prompt** is the panel text that tells the user what to do (for example, pick an option) and where to go (for example, click a navigation arrow to continue).

You should assign a prompt to virtually all panels in your guide file, including the first, middle, and last panels in a sequence, as well as the panels that contain control features (for example, a panel with radio buttons, checkboxes, or standard buttons). These prompts should tell the user what to do and where to go (for example, “Do this step, then you’re done”). When you need the full content area of a panel (for example, to include a full-size graphic that takes up the entire content area), you should use a Guide Script command to tell Guide Maker not to allocate prompt space for that panel. If you deallocate the prompt space, you should not place a prompt on the panel.

You specify prompts by defining them in a **prompt set**, a collection of four prompts that Apple Guide chooses to display based on whether a panel in the sequence is the first, middle, or last, or whether it contains controls (radio buttons, checkboxes, or standard buttons). You can associate a prompt set with

- all panels in all sequences
- all panels in a specific sequence
- one specific panel in a sequence

You can designate a particular prompt set as the **default prompt set**. If you do, Guide Maker uses this prompt set for all panels in all sequences by default. You can override the default on a sequence or panel-by-panel basis.

To provide the user with clear and consistent instructions throughout your guide file, you should create one or more prompt sets as needed for each type of panel or sequence. Typically, you create a prompt set that works for most of your sequences, and then create other prompt sets to work for panels or sequences with special requirements. You usually specify the most-used prompts as the default prompt set and then override it as needed.

You can use up to 255 characters for a single prompt. Apple Guide places the prompt in attributes 10-point Espy Sans plain and automatically wraps to a second line, generally aligning it with the left margin. If your panel uses the Tag and Body format, described in “Using the Recommended Panel Formats” on page 2-43, Apple Guide aligns the prompt with the Body text.

Designing a Default Prompt Set

If possible, use one prompt set throughout your guide file and make it the default prompt set. Sometimes, however, you need to override the default prompts for an entire sequence or for a particular panel. See the next section for details.

Table 2-1 shows a prompt set that Apple has defined for the first, middle, and last panels. This definition assumes that panels might contain standard controls

such as checkboxes or radio or standard buttons. Where applicable, you should use this prompt set in your guide file.

Table 2-1 The default prompt set recommended by Apple

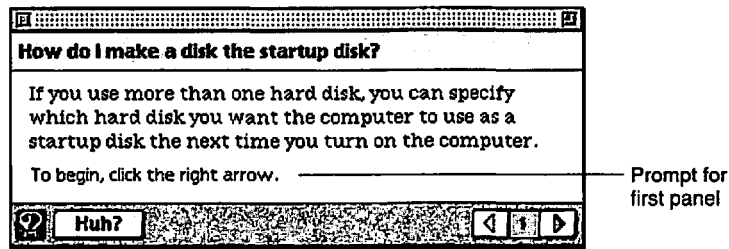
Panel	Prompt
First	To begin, click the right arrow.
Middle	Do this step, then click the right arrow.
Last	Do this step, then you're done.
Controls	Make your choice, then click the right arrow.

Note

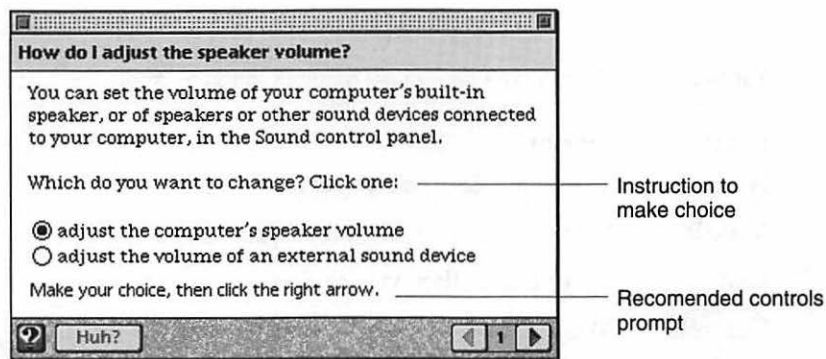
In certain cases, you need to override the default panels. See the next section for details. ♦

Figure 2-25 shows the prompt for the first panel in a sequence that uses the prompt set in Table 2-1. Here, the user clicks the right arrow to begin.

Figure 2-25 The default prompt for the first panel in a sequence



In contrast, Figure 2-26 shows the recommended prompt for a panel with controls. Note the instructions that precede the radio buttons. You should always include these additional instructions for any panel that contains radio buttons or checkboxes.

Figure 2-26 A decision panel

Overriding Default Prompts

If, for a particular sequence, the control features or content varies significantly, you should override the default prompt set for that sequence and use a prompt set that is more applicable. For example, if your guide file contains some sequences that describe only tasks and other sequences that describe only concepts, you need two prompt sets: one that tells the user to do actions and the other that tells the user to read information. Similarly, if your guide file contains a sequence that uses special navigation buttons rather than the standard navigation arrows, you need to use a prompt set that tells the user to click buttons rather than arrows.

You should also override a default prompt that does not apply to the control feature of a particular panel in a sequence; for example, if a panel in a sequence uses special navigation buttons that do not appear on other panels in that sequence, it requires a unique prompt.

Certain panel types recommended by Apple require that you override the recommended prompt set and use a more applicable prompt, as shown in Table 2-2.

Table 2-2 Override prompts by panel type

Panel type	Override prompt
Continue	After the <i>action occurs</i> , click Continue.*
Panel associated with a Huh? button†	Read this information, then you're done.
Definition	Read this information, then you're done.
Tip	Read this information, then you're done.
Related topics	Read this information, then you're done.
Closure panel that does not contain a final step	That's all, you're done.

* Replace the italic text in the prompt with text specific to the action being performed.

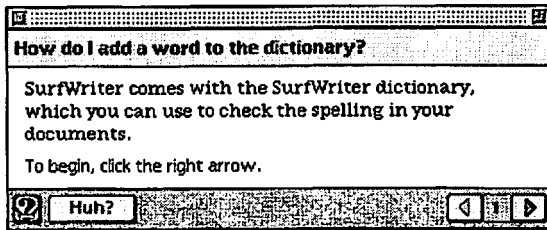
† Use only for panels that you specifically create for and associate with a Huh? button. Do not use for existing panels in the guide file that you associated with a Huh? button and that already have an assigned prompt.

For more information on these panel types, see “Designing Panel Types” beginning on page 2-50.

Using the Recommended Panel Formats

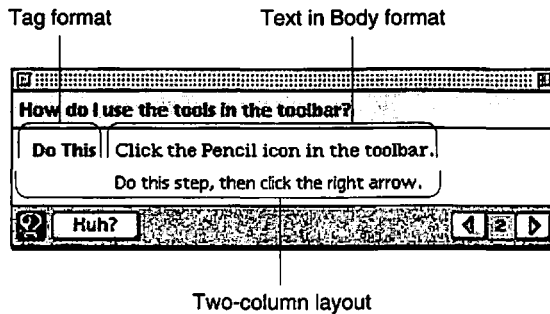
To format your panels, Apple recommends that you use either the *Full format* or the *Tag and Body format*. In general, you should use the Full format for panels unless they describe an action or contain lengthy introduction text, in which case, you should override it and use the Tag and Body format. (For more information, see “Designing an Action Panel” on page 2-53 and “Designing an Introductory Panel” on page 2-51.) If necessary, you can also devise and apply your own format using Guide Script commands. See the next section for details.

Figure 2-27 shows a panel with the Full format.

Figure 2-27 A panel with the Full format

The **Full format** provides the full-panel width—less an 11-pixel margin at each side—as a single column. In addition, it places the prompt just above the navigation bar and aligns it with the left edge of the leftmost format; that is, the text-column or full-panel width (less 11 pixels). It also applies the **Apple Guide font**—10-point Espy Serif Plain Black—to any text you place on the content area of the panel except text for radio buttons, checkboxes, and standard buttons, which by default appear in the system font. You should use Guide Script commands to specify that these button types instead use the Apple Guide font.

The **Tag and Body format** refers to using two formats—the Tag format and Body format—together. The **Tag format** provides a left column that you can use to format tags. A tag is a bold phrase (for example, “Do This” or “Oops”) that describes text in the Body format. The **Body format** provides a right column for text. The Tag and Body format thus divides the panel into two columns, in which the text in the left column uses the Tag format and the text in the right column uses the Body format. Figure 2-28 shows the Tag and Body format for an action panel.

Figure 2-28 A panel with the Tag and Body format

Designing Your Own Panel Format

You can apply your own panel layout and formatting using Guide Script commands. Apple recommends that you use the Full format and Tag and Body format wherever possible in your guide files. In general, you should create your own panel designs only for categories of help instructions that do not use one of the standard panel types recommended by Apple (see “Designing Panel Types” on page 2-50).

To provide the user with a consistent and clear panel design, you should use

- simple designs, so that the user can focus on the panel’s information rather than on its appearance
- the same design for all panels of the same type
- only a small number of designs, so that the interface appears simple and consistent
- a design that will not significantly increase the panel height, to avoid excessive shrinking or expansion of the window size as the user moves through sequences

You can create a column with certain attributes and then use them to simplify the layout of your panel objects. You can then use formats to control the placement of most panel objects, including text paragraphs, checkboxes, and radio buttons; PICT graphics and QuickTime movies; and labels for graphics. Guide Maker places panel text formatted this way inline in the boundaries specified by the format’s column and according to the format’s attributes.

For example, you can define a format whose bounds are defined by a column beginning with a top coordinate of 50 and a left coordinate of 75 that extends horizontally for 100 pixels. If you then specify centering a radio button in this column, it appears in the center of the column and not in the center of the panel.

You can also apply a format to Guide Maker commands that place text and objects in that panel. When you do, the format overrides all other previously specified formats, including the Full format. Guide Maker places inline the panel objects that follow the specified format in the boundaries specified by the format's column coordinates. Guide Maker also refers to this format to assign panel text attributes and to align prompts.

Using Graphics in Panels

Graphics increase the size of your guide files and should therefore be used sparingly. In general, you should use them only to describe an application feature that you cannot access using a coachmark. For more information on using coachmarks, see "Designing Coachmarks" beginning on page 2-79. You can also occasionally use a graphic to illustrate a concept (for example, to show relationships between several application features).

If you use a graphic to depict an onscreen item that the user can normally click or move (for example, an icon), you should indicate that the item is a graphic and not the actual item. For example, enclose the item in a box or place shading behind it (for an example, see Figure 2-34 on page 2-55).

To specify a picture in a panel, you must place it in a resource file or PICT format file and identify it to Guide Maker with its resource ID, resource name, or filename. You must also specify a replacement picture that is used only if the bit depth of the user's monitor is set to 4 bits or fewer. You should use the same size for both pictures. You can specify the picture's general location to be left, right, or center. When you do, Guide Maker places the picture justified within the current format.

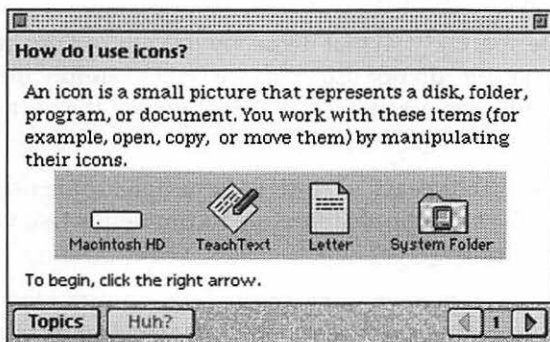
You can have Apple Guide place the graphic inline with the surrounding text and justify it within the current format (*relative positioning*) or you can specify the button's location relative to the current pen position by specifying a specific point (*absolute positioning*).

IMPORTANT

The *pen location* is the place where Apple Guide returns to place an object. The current pen location is not reset after you place an object with absolute positioning. ▲

For example, Figure 2-29 shows a Macintosh Guide panel containing a graphic centered with relative positioning.

Figure 2-29 A panel containing a graphic centered with relative positioning



For optimal localization, Apple recommends that you use relative rather than absolute positioning of objects on the panel. (When translated into another language, panel text can increase and possibly overflow onto a panel object.) To create a graphic that contains hot rectangles, however, use absolute positioning. For more information about hot rectangles, see “Designing Hot Text, Objects, and Rectangles” on page 2-77.

Formatting Panel Text

All panel text follows certain attributes. In general, almost all text that you place in the content area of a panel appears in the Apple Guide font—10-point Espy Serif Plain Black—by default. Apple Guide also automatically applies the text font, type size, and attributes to the panel title and prompt; the title bar is 10-point Espy Sans Bold and the prompt is Espy Sans 10 Plain. In addition, Apple Guide places all tags in attributes 10-point Espy Sans Bold.

Note

By default, text associated with radio buttons, checkboxes, and standard buttons uses the system font. You should use Guide Script commands to specify that these button types instead use the Apple Guide font, so that they match the other text on the panel. ♦

You can apply any formatting attributes to the Apple Guide font. When you format text yourself, use consistent styling conventions that enhance the clarity of your help instructions. Here are some suggestions:

- Use an underscore to indicate hot text.
- Use boldface type sparingly so that it effectively calls out crucial information to the user (for example, do not use boldface type to simply introduce a section of information on a panel but do use it to lead the user's attention to a warning or important note).
- Use text attributes that are easy to read. For example, some users find it difficult to read italic type. Apple therefore recommends that you instead use quotes for emphasis.
- Use the same formatting conventions throughout your guide files. For example, if you use quotes to enclose book title names on a panel, follow that same convention throughout.

If you plan to localize your guide files, you need to be aware of possible side effects of using styled text. For more information, see "Designing for Localization" on page 2-91.

Providing Navigation Methods on Panels

The user can navigate through panels using the left and right arrow buttons, other navigation buttons, content area buttons, and hot areas (including hot text, hot objects, and hot rectangles).

- With the navigation arrows, the user can move backwards and forwards through a sequence, panel by panel.
- With navigation and content area buttons, the user can go to a panel in the same sequence or in another, or can go to the application itself.
- With all three hot types, the user can go to a panel that contains useful but optional information for understanding the original panel.

Apple Guide always displays left and right navigation arrows in the navigation bar of each panel. These arrows are described in “Panel Features” beginning on page 2-37. For most panels, you must specify a navigation prompt that tells the user where to go, for example, “Click the right arrow to continue”. (Certain panels instead require a prompt that tells the user what to do, for example, “Read this panel, then you’re done.”) For guidelines on creating prompts, see “Designing Panel Prompts” beginning on page 2-39.

If a required condition is not true and the user clicks the right arrow button on a panel, you should force the user to go to an Oops or Continue panel instead of the next panel. For more information, see “Using Context Checks” beginning on page 2-83.

You can add up to three buttons in the navigation bar of a panel to supplement the default navigation arrows. (The default navigation arrows still appear on the navigation bar, even if a panel does not use them.) You should use the same navigation bar buttons consistently throughout a sequence and, if possible, throughout your entire guide file. You should always add to the navigation bar of each panel a GoStart button, which takes the user back to the access screen. You can also add a Huh? button, which, if active, will take the user to another panel containing crucial information. The panel associated with the Huh? panel opens on top of the original panel, allowing the user to read its information and return easily to the original panel.

You can also add buttons to the content area of a panel if you want to provide a navigation route specific to a particular panel. For example, you can add a button that takes the user to a QuickTime movie applying only to that panel.

For complete guidelines on creating navigation and content area buttons, see “Designing Buttons” beginning on page 2-70.

You should use hot text, objects, and rectangles to provide the user with information that is useful but not crucial. You can take the user to a panel you created specifically for the hot type or to a panel that exists in another sequence. Like the Huh? panel, the panel associated with a hot area opens up on top of the original panel. For more information, see “Designing Hot Text, Objects, and Rectangles” on page 2-77.

Designing Panel Types

The panels in your guide file present different types of help instructions. For example, a panel can present a step for the user to do, define terminology, or explain a concept. To ensure that your users can identify information in your guide files rapidly and easily, you should use **standard panel types**, which are panel designs that apply to specific categories of help information. For example, you should use the same panel type for all panels that give a tip.

Apple has designed a set of twelve standard panel types that you should use consistently throughout your guide files. Nine of these panels apply to basic help instructions that you typically use in your guide files: introductory, decision, action, information, tip, definition, related topics, transition, and closure. The three additional panels apply to Apple Guide features that can increase the scope and sophistication of your guide files: a panel associated with a Huh? button and Oops and Continue panels. Note that your guide files might not necessarily require all the Apple standard panel types.

Note

You should create your own panel design only if no Apple standard panel type is appropriate for a particular category of information in your guide file. If you do, be sure to follow the guidelines in this chapter for presenting help information. ♦

This section describes each panel type in detail. Most panels use the same title as the one for the associated topic. Therefore, the following paragraphs discuss panel titles only when they require a different convention. This section does not explain how to design a sequence using standard panel types. For this information, see “Designing a Sequence” beginning on page 2-66.

Note that all but two of these panel types use the Full format, which Apple Guide applies by default. Two panels, the action panel and lengthy introductory panel, use the Tag and Body format. For more information on both formats, see “Using the Recommended Panel Formats” beginning on page 2-43.

Designing an Introductory Panel

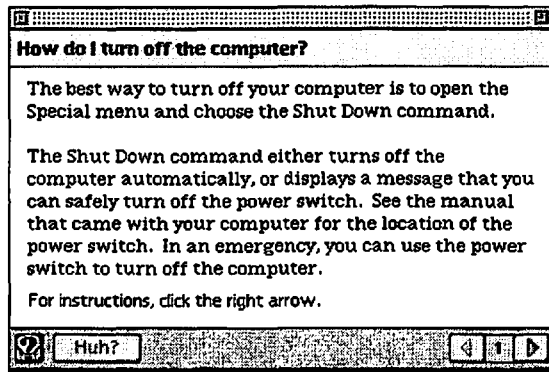
You should use an **introductory panel** at the beginning of each sequence for topics under the “How do I” or “Why can’t I” headings to describe its contents. If a sequence describes a task, the introductory panel should describe the required action. For example, if the sequence describes choosing a paintbrush from a toolbar, the introductory panel should briefly describe how to find the paintbrush and how to select it. This description prepares new users for the task and may even give experienced users enough information so that they can skip the rest of the sequence.

If the first panel of a sequence precedes a branch, you should use a decision panel with introductory text. Alternatively, you can use both an introductory panel and a decision panel. For more information, see “Designing a Decision Panel” on page 2-52.

Because the introductory panel is the first panel the user views after selecting a topic, the user can assume it contains instructions to perform an action. To avoid confusion, use the Full format for introductory panels and the Tag and Body format for panels containing an action. (For more information, see “Designing an Action Panel” on page 2-53.) For particularly long introduction panels, however, you can use the Tag and Body format to distinguish key points.

Use the prompt “To begin, click the right arrow.”

Figure 2-30 shows the introductory panel for a Macintosh Guide sequence on how to turn off the computer. Notice that the panel summarizes the steps that make up the task.

Figure 2-30 An introductory panel

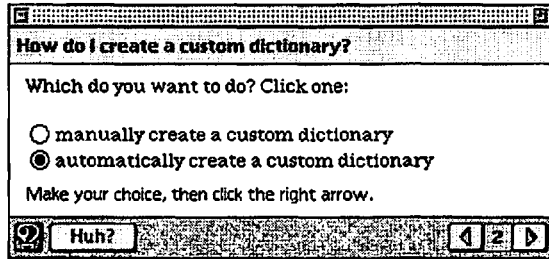
Designing a Decision Panel

You should use a **decision panel** to let the user view one or more branches. Create a decision panel using either radio buttons or checkboxes. To give the user only one option, use radio buttons; to give the user one or more options, use checkboxes.

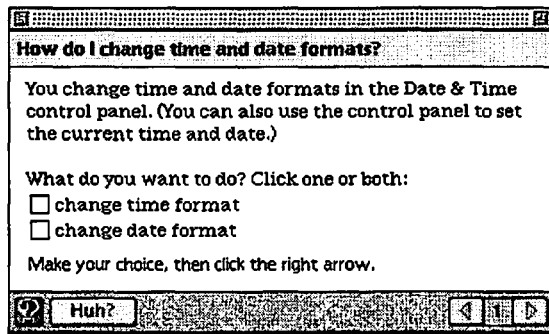
To create a decision panel, do all of the following:

- Use the Full format.
- Follow the Apple guidelines for using radio buttons and checkboxes (see “Using Radio Buttons and Checkboxes” on page 2-76).
- Use the prompt “Make your choice, then click the right arrow.”
- Precede the radio buttons or checkboxes with an instruction that tells the user what to do. For radio buttons, use the instruction “Which do you want to do? Click one.” For checkboxes, use the instruction “Which do you want to do? Click one or both.” (This instruction is in addition to the prompt.)

For example, Figure 2-31 shows a panel that asks the user to choose, using radio buttons, one of two methods for doing the same task.

Figure 2-31 A decision panel with radio buttons

In contrast, Figure 2-32 shows a panel that asks the user to choose, using checkboxes, one or *more* tasks.

Figure 2-32 A decision panel with checkboxes

Designing an Action Panel

You should use an **action panel** to present a single step in a procedure (for example, to tell the user to open a menu). You should present more than one step on an action panel only if there are several ways to perform the same action or several actions that occur in the same place.

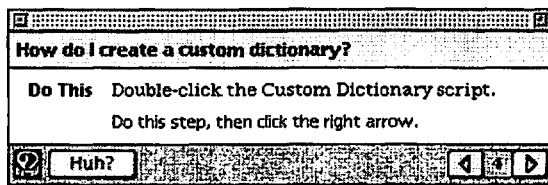
Authoring Tips and Suggestions

To create an action panel, do all of the following:

- Use the Tag and Body format to effectively lead the user's attention to the required action. The Tag should tell the user to do the action described on the panel (for example, Macintosh Guide uses the Tag "Do This") and the Body text should describe the actual action. For instructions that require lengthier text (for example, panels that contain multiple steps), place the Tag on the line above the text describing the action.
- Use a consistent Tag phrase for all action panels in your guide files.
- Use terse instructions to keep the panel size small. (The user is likely to keep the panel onscreen while performing the associated action).
- Use the prompt "Do this step, then click the right arrow."

Figure 2-33 shows a SurfWriter Guide panel that tells the user to open a script.

Figure 2-33 An action panel



Designing an Information Panel

You should use an **information panel** to provide brief conceptual explanations. You typically use an information panel in a sequence or branch that explains a concept.

Note

You can use an information panel in a sequence that explains a task, but it might be more appropriate to place the information in a Huh? or Tip panel. For more information, see "Designing a Tip Panel" beginning on page 2-55 and "Designing a Panel Associated With a Huh? Button" beginning on page 2-61. ♦

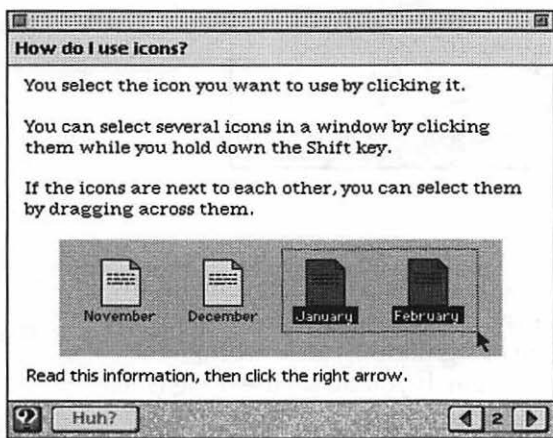
To create an information panel

- use the Full format
- use the prompt “Read this information, then click the right arrow.”

Consider including a content area button that takes the user to the part of the application that the panel describes. For more information, see “Designing Content Area Buttons” beginning on page 2-72.

For example, Figure 2-34 shows an information panel (from Macintosh Guide) that uses both text and graphics to explain how to use an icon.

Figure 2-34 An information panel



Designing a Tip Panel

You should use a **tip panel** to give the user a hint about how to perform an action or use an application feature (for example, to tell the user that a script is available for a particular task). The user should access the panel through a button or hot text. See “Designing Buttons” beginning on page 2-70 and “Designing Hot Text, Objects, and Rectangles” on page 2-77 for more information.

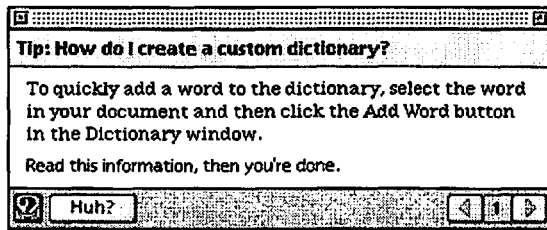
Authoring Tips and Suggestions

To create a Tip panel, do all of the following:

- Use the Full format.
- Place the word “Tip:” followed by the panel title in the panel tile area.
- Use the prompt “Read this information, then you’re done.”

For example, Figure 2-35 shows a SurfWriter Guide panel that gives a shortcut for adding a word to the dictionary.

Figure 2-35 A tip panel



Note

You can instead place a tip on the panel that contains the associated action or concept, as long as it does not greatly increase the panel size or significantly distract from the focus of the help instruction. If you do so, use a Tag (such as “Tip”) to indicate it to the user. Be sure to use the same Tag phrase for tips throughout your guide files. ♦

Designing a Definition Panel

You should use a **definition panel** to define key terminology that appears in a panel. Your guide file definitions should use different wording from the definitions provided by help balloons, which define such objects on the Macintosh screen as icons, windows, and commands.

Always use a separate panel for a definition rather than placing it on the panel that contains the associated term. The user should access the panel through hot text. See “Designing Hot Text, Objects, and Rectangles” on page 2-77 for more information.

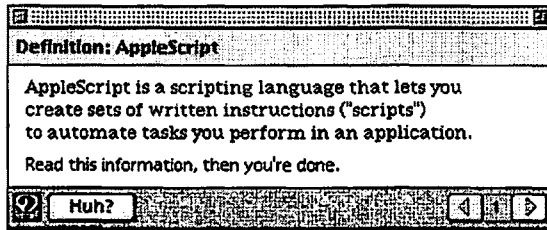
Authoring Tips and Suggestions

To create a definition panel

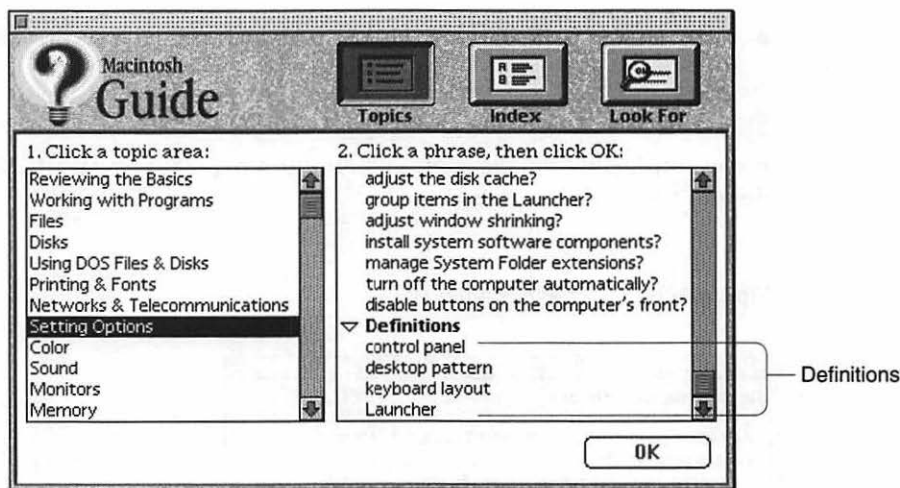
- use the Full format
- place the word “Definition:” followed by the definition term in the panel title area
- use the prompt “Read this information, then you’re done.”

For example, Figure 2-36 shows a definition panel (in SurfWriter Guide) that defines the term “AppleScript.”

Figure 2-36 A definition panel



If a definition applies directly to a topic area, you should make it a topic in the Full Access window under the heading “Definitions.” For example, Figure 2-37 shows the terms that appear under the “Definitions” heading when you select the Setting Options topic area in Macintosh Guide.

Figure 2-37 Some definitions in the Full Access window

Designing a Related Topics Panel

You should use a **related topics panel** to refer the user to other guide file topics that pertain to a specific panel or sequence. You can have the user access a related topics panel using a button in the content area or navigation bar, or using hot text. Where you place the button or hot text depends on the content of the related panel that it calls.

- To call a panel that lists topics pertaining to an entire sequence, place the Related Topics button or hot text in the content area of the closure panel of that sequence. Alternatively, you can place the Related Topics button in the navigation bar of all panels in that sequence.
- To call a panel that lists topics pertaining to a specific panel in a sequence (except the closure panel), place the Related Topics button or hot text in the content area of that panel.

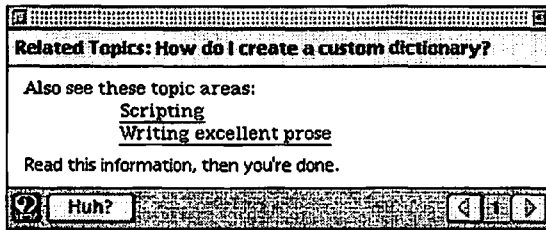
To create a related topics panel

- use the Full format
- place the words "Related Topics:" in the panel title area, followed by the user sequence title

- distinguish the related topics (and topic areas) from other text on the panel; for example, place them in a list or use styled text
- list the exact topic or topic area name that appears on the access window
- use the prompt “Read this information, then you’re done.”

For example, Figure 2-38 shows a related topic panel that appears in SurfWriter Guide when the user clicks a Related Topics button on the closure panel of a sequence.

Figure 2-38 A related topics panel



Designing a Transition Panel

You should use a **transition panel** to connect parts of multipart sequences. That is, use it if the user can consecutively view several branches in a sequence (for example, if the user can select checkboxes on a decision panel). The transition panel must always let the user know that one branch has ended and another is beginning.

You should also use a transition panel between the panel that ends a procedure and subsequent panels that provide optional information pertaining to that procedure. Here, the transition panel should let the user know that the required actions are complete and that the remaining panels contain only helpful information.

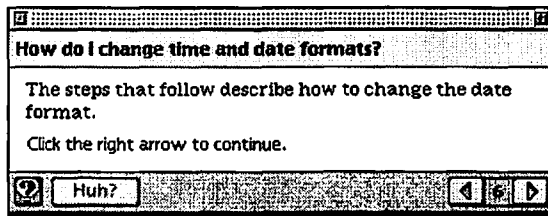
Authoring Tips and Suggestions

To create a transition panel

- use the Full format
- use the prompt “Click the right arrow to continue.”

For example, Macintosh Guide contains the topic “How do I change time and date formats?” This topic contains two branches: one for changing the time format and the other for changing the date format. Figure 2-39 shows the transition panel that takes the user from the time format branch to the date format branch.

Figure 2-39 A transition panel



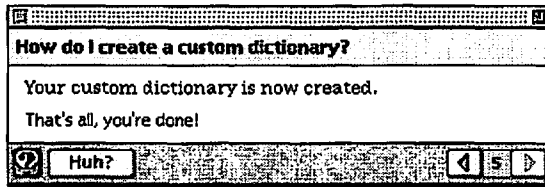
Designing a Closure Panel

You should use a **closure panel** at the end of a sequence to summarize the information covered by that sequence. The text on the panel should focus only on the help topic rather than repeating navigation information covered by the prompt text.

To create the panel

- use the Full format
- use the prompt “That’s all, you’re done.”

For example, Figure 2-40 shows the closure panel (in SurfWriter Guide) for the sequence “How do I create a custom dictionary?”

Figure 2-40 A closure panel

Designing a Panel Associated With a Huh? Button

A panel associated with a **Huh?** button appears when the user clicks an active Huh? button. The associated panel it calls up provides information crucial to understanding the original panel. The Huh? button must be active in the navigation bar of all panels in a sequence that has an associated panel. Its associated panel can already exist in the guide file, or it can be one created specifically for this button. When the user clicks the active button, the associated panel opens up on top of the original one. If you haven't associated a panel with a Huh? button, it is dimmed. For information on creating a Huh? button, see "Designing Navigation Buttons" on page 2-71.

If a panel contains an active Huh? button, always explain the contents of the button's associated panel, so that the user can decide whether to view it. And if you specifically create an associated panel, use the prompt "Read this information, then you're done."

Figure 2-41 shows a panel from SurfWriter Guide that contains an active Huh? button. Notice the text explaining the contents of the button's associated panel.

Figure 2-41 A panel with an active Huh? button and an explanation of the button's associated panel

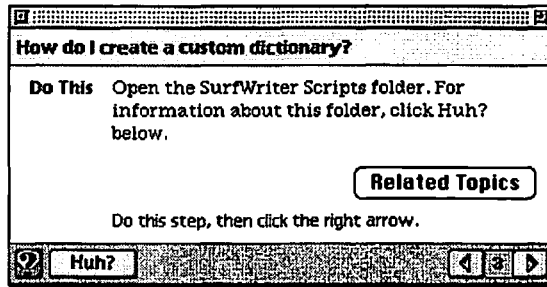
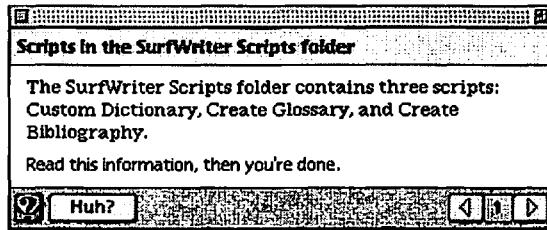


Figure 2-42 shows the panel associated with the Huh? button on the previous panel (Figure 2-41).

Figure 2-42 A typical panel associated with a Huh? button



In general, you should not associate more than one panel with a Huh? button. If the information exceeds one panel, consider using a related topics panel or creating a topic or branch.

- For information that appears in another sequence in the guide file, use a related topics panel to point the user to the sequence. For more information, see "Designing a Related Topics Panel" on page 2-58.

- For information that you create specifically for the panel, make it a new topic on the access window or in a branch in the sequence. For more information, see “Designing Topic Areas and Topics” on page 2-30 and “Designing Branches” on page 2-67.

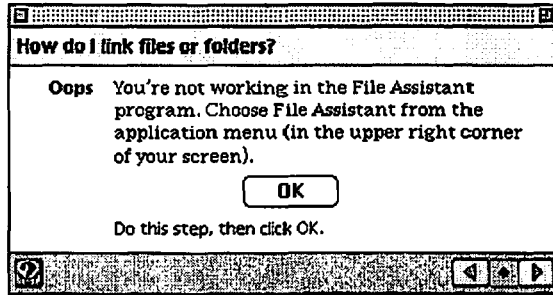
You can occasionally associate a sequence of panels with a Huh? button for information that is not suitable as either a topic or a branch, for example, information that the user typically reads only once or that applies to only a limited number of users. As with a single panel associated with a Huh? button, the sequence can be another sequence in the guide file or one that you create specifically for the panel.

Designing an Oops Panel

You should use an **Oops panel** to tell the user that a condition specified on a previous panel (for example, opening a control panel) was not met and that it must be met to continue to the next panel. The Oops panel should appear only when the user does not complete an instruction and then clicks the right arrow to continue to the next panel in the sequence; for example, the user does not follow a panel’s instructions to open a control panel and then tries to move to the next panel. Use the <Make Sure> command to include an Oops panel in a sequence. For more information, see “Using Context Checks” beginning on page 2-83.

An Oops panel should contain an OK button, a description of how to correct the condition, and instructions to correct the condition or press the OK button. If the user makes the condition true and clicks the OK button, Apple Guide takes the user to the next panel in the sequence. If the user does not make the condition true and clicks the OK button, Apple Guide closes the Oops panel and returns the user to the first previous panel that either does not have a <Make Sure> command specified for it or that has a <Make Sure> command whose condition evaluates true.

For example, Figure 2-43 shows an Oops panel (from Macintosh Guide) that appears if the user failed to follow instructions to open the File Assistant application.

Figure 2-43 An Oops panel

To create an Oops panel, do all of the following:

- In its sequence display title area, place the same title used by the sequence.
- Use the Tag and Body format so that the user can easily discern the required action.
- Use the tag "Oops" (or its localized equivalent).
- Place the OK button in the center of the panel. For more information, see "Using Standard Buttons" on page 2-74.
- Use the standard text to describe the Oops condition and tell the user what to do: *"The condition is not true. Click OK for instructions (or make condition true, then click OK)."* Replace the italic text with text specific to the condition that is not true.
- Avoid using a prompt in this panel.

If possible, use only one panel to describe the Oops information. If you use multiple panels in an Oops sequence, the right arrow button on the first panel and the left arrow button on the last panel must be dimmed. For panels in an Oops sequence, you can specify the <Skip If> and <If> commands but not the <Make Sure> command. For more information, see "Using Context Checks" beginning on page 2-83.

If you specify that the Oops panel appear if a certain condition is not true on a panel—for example, the condition is not true if a particular control panel is not open—you should specify it for all other panels in that sequence that contain the same condition.

You should also verify whether it would be better to use a Continue panel, which is described in the next section. You should not mix use of the Continue and Oops panels for the same condition.

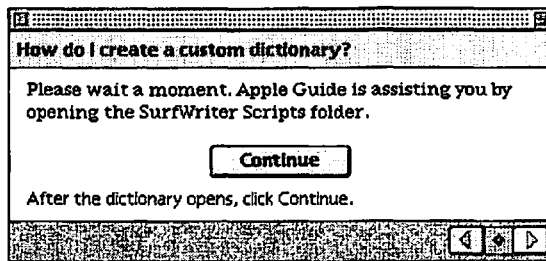
Designing a Continue Panel

You should use a **Continue panel** to have Apple Guide offer to complete for the user the condition that hasn't been met (for example, opening a control panel). The panel appears only if the condition on a panel is not true when the user clicks the right arrow to continue to that panel in the sequence. For example, if the user does not follow a panel's instructions to open a control panel and then tries to move to the next panel, a Continue panel appears. You include a Continue panel in a sequence using the <Make Sure> command. For more information, see "Using Context Checks" beginning on page 2-83.

A Continue panel should tell the user to wait for Apple Guide to perform the condition and then click the Continue button on the panel.

Figure 2-44 shows a Continue panel that appears in SurfWriter Guide.

Figure 2-44 A Continue panel



To create a Continue panel, do all of the following:

- Place in the sequence display title area the same title used by the sequence.
- Use the standard text to tell the user what Apple Guide is doing: "Please wait a moment. Apple Guide is assisting you by *condition being performed*." Replace the italic text with text specific to the condition being performed.

Authoring Tips and Suggestions

- Place the Continue button in the center of the panel. For more information, see “Using Three-Dimensional Buttons” on page 2-74.
- Use the prompt “After the *action occurs*, click Continue.” (Replace the italic text with text specific to the action being performed.)

Before specifying a Continue panel, you must first see if it is programmatically possible to perform the given condition. For example, you can use AppleScript to perform the task for the user. If it is not programmatically possible to use a Continue panel, use an Oops panel instead (described in the previous section).

If you specify that the Continue panel appear if a certain condition is not true on a panel—for example, the condition is not true if a particular control panel is not open—you should specify it for all other panels in that sequence that contain the same condition. You should not mix use of the Continue and Oops panels for the same condition.

Designing a Sequence

When you create a guide file—particularly of the types Help and Tutorial—you typically encounter topics that require a panel sequence rather than just one panel. A **panel sequence** (referred to here as a **sequence**) is a set of related panels that the user can access linearly using left and right navigation arrows. A sequence can also contain subsequences (or branches). To create a sequence, you first break the associated topic into a series of user tasks or concepts and then create a panel for each task or concept. Plan your sequences in hard copy form before creating your help source files (see Chapter 3). Where possible, you should use context checks so that your sequences provide information specific to the user’s needs. See “Using Context Checks” on page 2-83 for more information.

To design a sequence, you should be familiar with the standard panel types, described in the previous section. Then design the sequence with two levels.

The **first-level panels** should directly lead the user through the task or concept of the associated topic. If the sequence describes a task, these panels typically consist of an introductory panel that describes the sequence, one or more action panels that lead the user through the task, and a closure panel that sums up the task. If the sequence describes a concept, these panels typically consist of an introductory panel and one or more information panels containing explanatory

text. If the sequence includes branches, you design each branch following these same guidelines. If you want the user to view one or more branches, precede the branches by a decision panel. See the next section for information on when to create a branch.

The **second-level panels** should consist of *supplemental help panels*, which the user chooses to view, and Oops and Continue panels, which Apple Guide automatically presents to the user under certain conditions. Supplemental help panels should contain additional information that the user can access by clicking a button or hot area on a first-level panel. These panels can include a panel associated with a Huh? button or hot text, or definition, tip, and related topics panels. You specify Oops and Continue panels with the <Make Sure> command, which specifies a condition that must be true before Apple Guide shows a panel to the user. For more information, see “Comparison of Oops and Continue Panels” on page 2-85.

You should not exceed certain quantities of panels and branches in a single sequence. Do not exceed

- 15 panels in a branch
- 32 panels in a single sequence, including all branch panels
- 10 branches in a single sequence

Designing Branches

A **panel branch** (known here as a **branch**) is a sequence within another sequence. With branches, you can effectively present choices or context-specific instructions to the user under a single topic with a short direct name. You should always use branches to let the user

- choose one of several help instruction options that are mutually exclusive
- choose one or more help instruction options for information that is parallel or highly related
- automatically receive help instructions about a specific condition that you can verify with context checking
- choose to view help instructions about a specific condition that you cannot verify

Authoring Tips and Suggestions

When designing your help, reduce the number of topics in the access window, keep topic names short and direct, and provide help that is specific to the user's context. Avoid creating separate topics for tasks that are highly related to the same goal or feature. And don't create a sequence that describes several separate tasks sequentially. If you do, you force the user to navigate through panels to reach panels of interest. If the length and style of your topic name don't conform to the suggested style in this chapter, your sequence might require a branch or you might be describing the branch in the topic name rather than only the topic goal. For more information, see "Designing Topic Areas and Topics" beginning on page 2-30.

You can have Apple Guide display a branch to the user automatically, or you can have the user enter the branch from a button linked to a built-in function or from a decision panel using checkboxes or radio buttons. (For more information on decision panels, see "Designing a Decision Panel" on page 2-52.)

IMPORTANT

The Guide Script commands used to create a branch determine whether the user can return to the original sequence or to the access window. Wherever possible, create a branch from which the user can return to the original sequence. See the chapter "Guide Script Command Reference" for more information. ▲

The rest of this section provides details on creating branches.

Designing Branches for Mutually Exclusive and Related Tasks

A guide file topic can contain instructions that are mutually exclusive (for example, several tasks that accomplish the same goal) or that are highly related to each other (for example, parallel tasks or tasks that occur in the same place). For both kinds of tasks, you should create a branch that the user accesses through a decision panel. Use radio buttons to give the user only one choice among several options and use checkboxes to give the user the choice of one or more options.

The following example calls for radio buttons. The SurfWriter application lets the user enter a word in the dictionary using either menu commands or a script. Rather than presenting a separate topic for each method (for example, "How do I enter a word in the dictionary using menu commands?" and

“How do I enter a word in the dictionary using a script?”), the guide file instead uses a topic name that describes the main goal (“How do I enter a word in the dictionary?”) and creates a branch for each method. Because both branches accomplish the same goal, the user selects only one of the branches from a decision panel with radio buttons.

This next example calls for checkboxes. Macintosh Guide uses a single topic for the question “How do I change the time and date formats?” because both of these tasks are accomplished in the same place, the Date & Time control panel. The sequence contains a separate branch for each task. Unlike the previous example, these tasks are not mutually exclusive; the user might want to change only the date format, only the time format, or both. Therefore, the user can select one or both branches from a decision panel with checkboxes. For more information, see “Using Radio Buttons and Checkboxes” on page 2-76.

Designing Branches for a Specific Condition

A guide file topic can include information that specifically applies to a condition of the user’s environment or context (for example, information that applies only if particular software is installed on the user’s computer).

For this specific information, you can create a branch that the user selects or one that is provided automatically. If you can check the condition programmatically, you should use the <If> and <Else> commands or the <Skip If> command to have Apple Guide automatically show the appropriate branch to the user. Otherwise, you should have the user choose the appropriate branch from a decision panel.

For example, assume that your application provides two methods for checking the spelling of a document: one uses the standard dictionary and special thesaurus, and the other uses only the standard dictionary. Because you can verify whether the user has installed the thesaurus, you can create a branch for each method using the <If> and <Else> commands. In this way, Apple Guide verifies whether the thesaurus is installed and automatically presents the appropriate branch to the user.

In contrast, assume that your guide file contains the sequence “How do I print a document?” Your users need to view only those panels that apply to their particular printer type—in this case, a LaserWriter or ImageWriter. You create a branch for each printer type. And because you cannot use context checking to

verify the printer type, you provide a decision panel from which the user can choose the appropriate branch.

Designing Buttons

This section contains guidelines for creating buttons for your panels. For more information on button alignment, see “Designing Your Own Panel Format” on page 2-45. Also see the guidelines on standard toolbox controls in the *Macintosh Human Interface Guidelines*.

You can add two types of buttons to your panels: navigation and content area.

- A **navigation button** always appears in the navigation bar of a panel and takes the user to different parts of the guide file (for example, a navigation button can take the user back to the access window).
- A **content area button** appears in the content area of a panel and is generally associated with an event or navigation route specific to that panel. You can add these types of buttons to a content area: radio buttons, checkboxes, standard, and three-dimensional. They are described here.
 - Radio buttons and checkboxes both take the user to a single branch in the sequence; the user can select only one radio button at a time and select one or more checkboxes at time.
 - A **standard button** is a two-dimensional button, drawn by the Macintosh toolbox, that has an event associated with it (for example, a Cancel button, which dismisses the operation the user started).
 - A **three-dimensional button** can perform as either a navigation button or a standard button (for example, a three-dimensional button that takes the user to a related sequence). It has two states to indicate its current behavior—up and down—and contains a graphic inside it.

Note

You cannot use more than 50 user control features—that is, buttons and checkboxes—for any single sequence. ♦

Designing Navigation Buttons

In the navigation bar, you can add from one to three buttons to supplement the default navigation arrows. (If a panel does not use the default navigation arrows, they appear dimmed on the navigation bar.) For any navigation bar button, you should use a height of 18 pixels and include a gray pixel (level 1 or 2) in its corners so that it blends into the panel background.

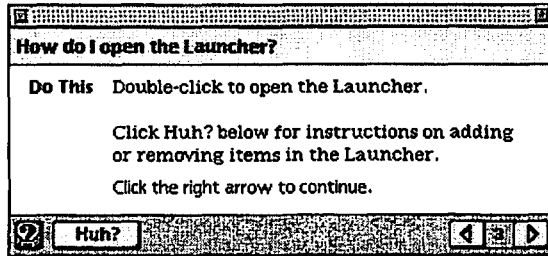
If possible, use the same set of navigation bar buttons throughout your guide file. If a panel in a sequence requires a button that is unique to it (for example, a button that launches a demonstration that is specific to that panel), do not use a navigation button. Instead, use a content area button. See the next section for details.

On the navigation bar of each panel, provide a **GoStart button** to take the user back to the access screen. The GoStart button contains the Apple Guide icon, or lightbulb, and should appear in the lower-left corner of each panel. (In some system software versions, the GoStart button instead contains the word “Topics”.)

Another useful button to place on the navigation bar is the **Huh? button**, because you can use it to take the user to a place that expands the information on a panel. The Huh? button contains the word “Huh?” (or its localized equivalent); it should appear on all panels in the sequence. The Huh? button should appear active on the panel only if you associate it with another panel. Then if the user clicks the Huh? button, the panel associated with it opens on top of the original panel. If you do not associate a panel with this button, it will appear dimmed. For more information, see “Designing a Panel Associated With a Huh? Button” on page 2-61.

Figure 2-45 shows a panel that contains GoStart and Huh? buttons in the navigation bar.

Figure 2-45 A panel with GoStart (using the lightbulb icon) and Huh? buttons in the navigation bar



Note

Guide Maker provides the files needed to create the GoStart and Huh? buttons. ♦

A guide file can contain only one dimmed navigation button at a time (that is, a navigation button that is inactive). Apple recommends that you provide dimmable functionality to the Huh? button.

Designing Content Area Buttons

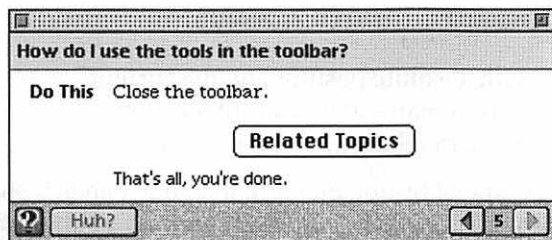
Apple Guide imposes no limit on the number of buttons that you can add to the content area of a panel. Apple, however, sets a specific limit for radio buttons and checkboxes and recommends that you avoid placing an excessive number of standard and three-dimensional buttons; otherwise, your panel design can become confusing and cluttered. Also remember that Apple Guide imposes a maximum number of 50 control features per sequence, as described earlier in this section.

By default, radio buttons, checkboxes, and standard buttons use the system font. You should use Guide Script commands to specify that these button types instead use the Apple Guide font.

You can have Apple Guide place the button inline with the surrounding text and justify it within the current format, or you can specify the button's location relative to the current pen position by specifying a specific point. To make localization easier, Apple recommends that you use relative rather than absolute positioning of buttons on the panel. (When translated into another language, panel text can increase and possibly overflow onto a panel button.)

Figure 2-29 shows a panel with a button centered with relative positioning.

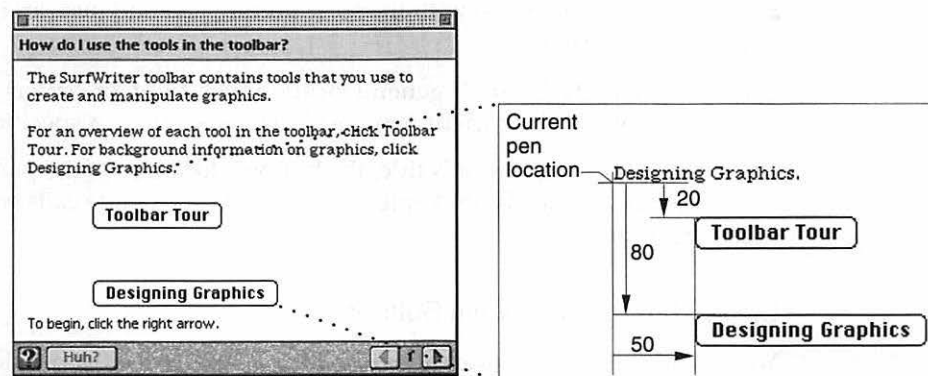
Figure 2-46 A panel with a button centered using relative positioning



This particular panel uses the Tag and Body format. Notice that Guide Maker centers the button on the Body format. By default, Apple Guide places the button just under the panel text. You specify left, center, or right justification within the current format. If you prefer more white space between the panel text and button, add a carriage return after the button, as shown in Figure 2-29.

In contrast, Figure 2-47 shows a panel containing two buttons, each with absolute positioning.

Figure 2-47 A panel containing two buttons with absolute positioning



This panel uses the Full format. Notice that the location of the first button is specified as `Point(50,20)`, and the location of the second button as `Point(50,80)`. Guide Maker places the first button 50 pixels to the right of and 20 pixels down from the current pen location and places the second button 50 pixels to the right of and 80 pixels down from the current pen location.

IMPORTANT

If you place a button with absolute positioning, the current pen location is not reset (it remains at its current location until you place an object with relative positioning). ▲

If you use a particular type of button on more than one panel, it should appear in the same location for all panels that use it throughout your guide files. For example, you might place all Tip buttons on the lower-right side of the panel content area.

The rest of this section contains guidelines specific to creating standard and three-dimensional buttons, as well as radio buttons and checkboxes.

Using Standard Buttons

Standard buttons are a set height of 20 pixels with a minimum width of 59 pixels (the width of a standard OK button). Apple Guide automatically

- adjusts the button width to accommodate the button title (the width of the text plus 10 pixels on each side)
- highlights the button in response to the user's actions
- places the button inline with the surrounding text and justifies it within the current format

You can specify the button's general location (left, right, or center) or its location relative to the current pen position by specifying a specific point.

You must specify the button's title, the button's location on the panel, and the event function (typically an Apple event) that Apple Guide calls when the user clicks the button.

Using Three-Dimensional Buttons

You create a three-dimensional button of any size. You must provide the color pictures that describe the button's appearance when it's selected (up) and not selected (down). You should also provide an additional set of black-and-white

pictures, which Apple Guide uses according to the bit depth of the user's monitor. Be sure to use the same size for both pictures.

You must also provide the button's location on the panel and the event function that Apple Guide calls when the user clicks the button.

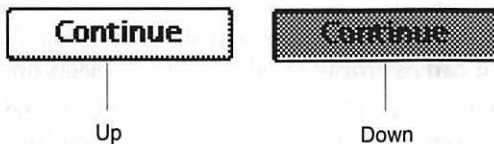
One of the recommended panel types, Continue, uses a three-dimensional button. Figure 2-48 shows the up and down appearance, in grayscale, of a color Continue button.

Figure 2-48 The up and down appearance (in grayscale) of a color Continue button



In contrast, Figure 2-49 shows the up and down appearance of a black-and-white Continue button.

Figure 2-49 The up and down appearance of a black-and-white Continue button



For more information on the Continue panel, see "Designing a Continue Panel" on page 2-65.

Apple Guide places the button inline with the surrounding text and justifies it within the current format. You can specify the button's general location (left, right, or center) or its location relative to the current pen position by specifying a specific point. For example, if you specify the button location as `Point (50, 100)`, Guide Maker places the button 50 pixels to the right and 100 pixels down from the current pen location. For more information on placing buttons

at specific coordinates, see the previous section. If you create a three-dimensional button that resembles an icon, you should also provide a label that tells the user what to do.

Using Radio Buttons and Checkboxes

Use radio buttons or checkboxes to have the user select branches in a sequence, following these guidelines:

- Use radio buttons to let the user choose only one of several branches that are mutually exclusive (for example, for branches that describe different methods to achieve the same goal).
- Use checkboxes to let the user choose one or more branches of related information (for example, for branches that describe parallel tasks or tasks that occur in the same place).
- Do not exceed four radio buttons or checkboxes on the content area of a single panel.
- Do not use radio buttons and checkboxes on the same panel.
- Specify the default state (on or off) of each radio button and checkbox.
- Provide each radio button or checkbox with a title that clearly identifies the choice it offers.
 - Place the title in the Apple Guide font without boldface.
 - Keep the title as short as possible. Brevity is particularly crucial for successful localization; if a title expands in size when translated into another language, it can overflow to other text or objects on the panel.
 - Use a unique name for each title in a single sequence. Apple Guide cannot distinguish between two radio buttons or checkboxes with the same title on different panels in the same sequence.
 - Make sure that checkbox titles reflect clearly different states.
- Use the default formatting that Apple Guide provides for radio buttons and checkboxes, where the radio button or checkbox appears inline with the surrounding text and is justified within the current panel format. Be sure, however, to change the default font to the Apple Guide font.

You should use a prompt that tells the user how to select the radio button or checkboxes: for example, “Make your choice, then click the right arrow.” Above the radio buttons or checkboxes, you should include an instruction that tells

the user what to do. For example, the instructions for radio buttons can say “Which do you want to do? Click one.” In contrast, the instruction for checkboxes can say, “What do you want to do? Click one or both.” For examples of such instructions, see “Designing Panel Prompts” on page 2-39.

Designing Hot Text, Objects, and Rectangles

You can use hot text, objects, and rectangles to provide information that is useful but not crucial to completing the particular task on a panel. Each of these types creates a hot (or active) area on the panel. When the user clicks a **hot area**, Apple Guide performs the action associated with the area, for example, it opens another panel that contains related information. This panel can be one that you specifically create for the hot area or another existing panel in the guide file. Do not confuse the information provided by hot types with information that belongs on a panel associated with a Huh? button. For more information, see “Designing a Panel Associated With a Huh? Button” on page 2-61.

You should follow these guidelines for using each of the hot types:

- Use **hot text** to specify certain text on the panel as a hot area. For example, you can use hot text to make the word “dictionary” a hot area on a panel.
- Use a **hot object** to specify a hot area using the rectangle of the next object (either text or a graphic) specified in the panel definition. For example, you can create hot objects out of 'PICT' graphics (for example, an icon).
- Use a **hot rectangle** to make a specific rectangle on the panel a hot area. Creating a hot rectangle provides more precise control over the placement of the hot rectangle than creating a hot object, which assumes only the rectangle of the following object. Be aware, however, that this positioning might create unwanted side effects during localization. See “Designing for Localization” on page 2-91.

You should use an underscore to identify hot text to the user, for example, dictionary.

The appearance of a graphic or hot rectangle designated as a hot type does not change, so you should use text to indicate that it's a hot area. For example, for a hot graphic, provide the instruction “Click this graphic for more details.”

Authoring Tips and Suggestions

Figure 2-50 shows a panel that contains a single word of hot text, indicated with an underscore. This panel also contains an active Huh? button. Therefore two panels are associated with it, one (Figure 2-51) for the hot text and one (Figure 2-52) for the Huh? button.

Figure 2-50 A single word of hot text on a panel

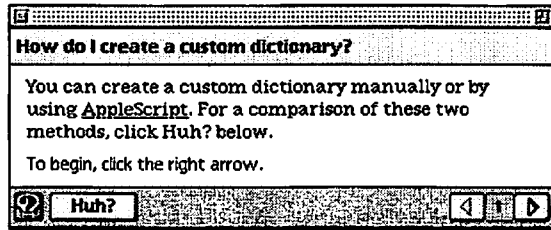


Figure 2-51 A panel associated with a single hot-text word on a panel

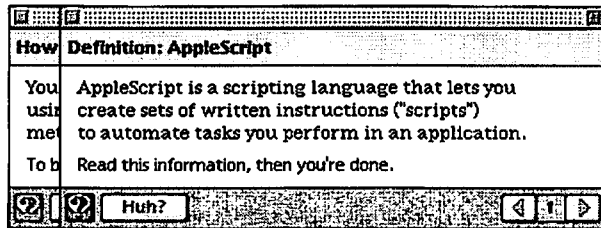


Figure 2-52 A panel associated with a Huh? button

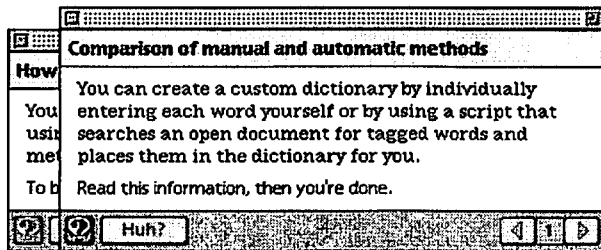
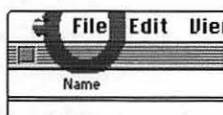


Figure 2-52 compares two methods for creating a custom dictionary that are referred to by the original panel (Figure 2-50).

Designing Coachmarks

A **coachmark** is an onscreen graphic that circles or points to an item on the screen. Use coachmarks to guide the user's attention to screen areas described in a help instruction, typically if the user needs to perform an action (for example, to open a menu or type information). Figure 2-53 shows a coachmark circling a menu that the user needs to open.

Figure 2-53 A menu coach



The coachmark remains visible until system software processes the next system event or user event in the application or panel.

You can assign only one coachmark per panel, and one is sufficient if you present only one action or concept per panel. Once you assign a coachmark to a particular panel, it appears each time the user opens that panel.

IMPORTANT

You should use coachmarks only for those elements of the user interface whose location or software condition you can verify; otherwise, you can inadvertently create a coachmark that appears at a random location on the screen. ▲

Apple provides five types of coachmarks and several built-in coachmark styles. These are described next.

Using Coachmark Types

There are five **coachmark types**: menu, item, object, window, and AppleScript.

A **menu coach** is a coachmark for a specific menu or menu item. When the user opens a panel with a menu coach, Apple Guide uses that coach style and coach color to draw a coachmark for the specified menu and menu item. When the user pulls down that menu, Apple Guide uses the specified color and text style for the specified menu item.

An **item coach** is a coachmark for an item in a dialog box or other interface element in a window (or dialog box). When the user opens a panel that includes a command with an item coach, Apple Guide uses the specified coach style to draw a coachmark for the specified item.

An **object coach** is a coachmark for an object based on a rectangle that your application returns for the named object. When the user opens a panel that includes a command that names a defined object coach, it sends an Apple event to your application that requests it to return a rectangle for the named object. When Apple Guide receives the rectangle for the object, Apple Guide draws the coachmark.

A **window coach** is a coachmark for a specific area of a window. When the user opens a panel that includes a command that names a defined window coach, Apple Guide uses the specified coach style to draw a coachmark based on the location of the coachmark, as specified by the command.

An **AppleScript coach** is a coachmark that uses AppleScript to determine the object to mark. When the user opens a panel that includes a command that names a defined AppleScript coach, Apple Guide executes the specified script. Once the script returns a rectangle for the object, Apple Guide draws the coachmark.

For more information on creating these coachmark types, see the chapter “Guide Script Command Reference.”

Using Coachmark Styles

Apple Guide provides four built-in coachmark styles: a red circle, a red underline, a red arrow, and a green “X” character.

- Use the red circle coachmark for a menu coach or to tell the user where to click in a limited or enclosed area.

- Use the red underline coachmark to indicate an item in a menu or to tell the user where to type information.
- Use the red arrow to point to an area on the screen. It is particularly effective for pointing to the target of a drag.
- Use the green “X” character to mark where the user needs to type information or to mark a large region.

For example, Figure 2-54 shows a red circle coachmark around the Macintosh Chooser icons.

Figure 2-54 A red circle coachmark

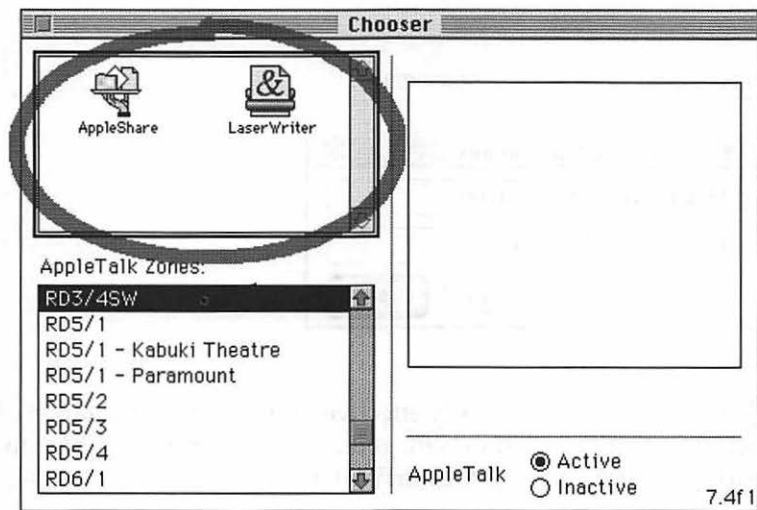
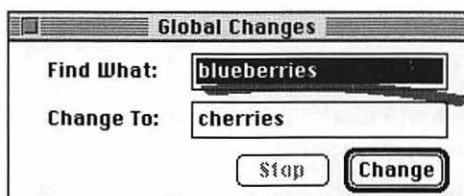
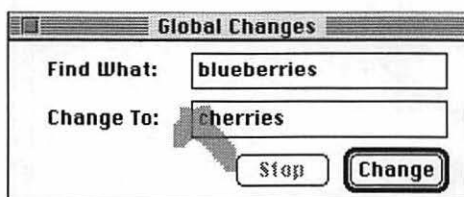


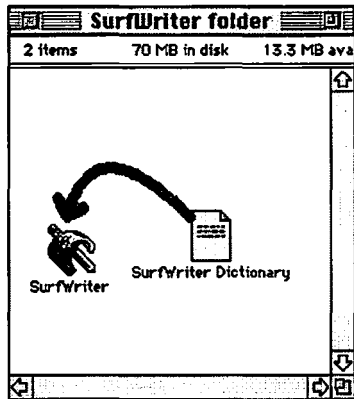
Figure 2-55 shows a red underline coachmark indicating the Find What field where the user needs to type text.

Figure 2-55 A red underline coachmark

Similarly, you can use the green X to point to an area on the screen. For example, Figure 2-56 shows the green X pointing to the Change To field, where the user needs to type text.

Figure 2-56 A green X coachmark

The red arrow is particularly effective for pointing to the target of a drag. For example, Figure 2-57 shows the red arrow demonstrating how to drag the SurfWriter dictionary to the SurfWriter application.

Figure 2-57 A red arrow coachmark

Using Context Checks

Context checks are functions that verify certain conditions of the user's environment so that Apple Guide can dynamically skip or show certain panels to the user based on their appropriateness. These conditions can be tasks that the user needs to perform or conditions based on the user's environment, for example, whether a folder is open or a file is present. In either case, if a panel specifies that a certain condition must be true, you need to determine whether you can perform context checks on that condition.

For example, your sequence can contain panels that instruct the user to open a window or click an item. If the user has already performed the task specified by a panel, there is no need for the user to view that panel. For each panel, therefore, you should use the <Skip If> command to verify whether the specified condition is true and if so, to tell Apple Guide to skip the panel. Similarly, your sequence can contain a branch that applies only to a specific condition that you can programmatically verify. When it does, you can use the <If> command to show the user that branch only if the specific condition is true. For more information, see "Designing Branches for a Specific Condition" on page 2-69.

Your sequence can also contain panels that you do not want the user to view unless a certain condition is true (for example, certain panels can require that a specific folder be open). For these panels, you should use the <Make Sure> command to verify that the specified condition is true before Apple Guide shows the associated panel to the user. If the condition is not true, you can specify that one of two panel types appear—an Oops or Continue panel—that work in different ways to make the condition true. These panels are described later in this section. For additional guidelines on creating these panels, see “Designing an Oops Panel” on page 2-63 and “Designing a Continue Panel” on page 2-65.

Apple recommends that you use context checks in your guide files wherever possible. Specifying context checks in your guide file generally occurs in four stages. You might have the skills and background to do this work yourself, or you might work with a team that includes instructional designers, scriptors, and developers. Here are the tasks you or your team need to accomplish:

1. Determine those panels that require context checks in the early stages of designing your guide file.
2. Determine whether panels with a context check are associated with an Oops or Continue panel. For Continue panels, you must also determine whether it is programmatically possible to perform the given task for the user.
3. Create the code that checks the condition specified by the context check.
4. Test whether the context checks for any user action (for example, selecting a menu command) still work properly if the user performs the action in the wrong order in the sequence.

Note

Guide Maker provides a file, Standard Setup, that contains built-in context checks. ♦

This section explains how to analyze your panels to determine whether they require <Make Sure> and <Skip If> commands. It also describes how to design Oops and Continue panels. It then describes the order Apple Guides uses to evaluate context checks. For complete descriptions of all context checking commands, see the chapter “Guide Script Command Reference.”

Comparison of Oops and Continue Panels

When the user clicks the right arrow to move to the next panel and the next panel is preceded by a <Make Sure> command, Apple Guide checks the condition associated with the <Make Sure> command before displaying the next panel. If the condition is false, you can specify that an Oops or Continue panel appear.

An Oops panel tells the user that the condition (typically an action described on a previous panel) must be true for the user to continue to the next panel. A Continue panel offers to have Apple Guide do the condition that must be true before the user can go to the next panel; that is, do the action for the user. You can therefore use a Continue panel only if it is programmatically possible to perform the required action that makes the condition true. For guidelines, see “Designing an Oops Panel” on page 2-63 and “Designing a Continue Panel” on page 2-65.

Note

If you specify that a Continue or Oops panel appear for a certain condition—for example, the condition is false if a particular control panel is closed—you should specify the same result for all other panels in that sequence that contain the same condition. Also, you should not mix use of the Continue or Oops panels for the same condition. ♦

Analyzing a Sequence for Context Checks

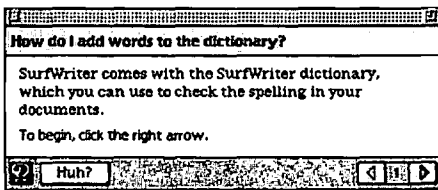
To determine the context checks for your guide file, go through each sequence, panel by panel, and consider these issues:

- Does the panel tell the user to perform an action? Can you programmatically check whether the condition that results from this action is true? (For example, can you check whether a control panel is open?) If so, specify a <Skip If> command for that panel.
- Does that panel set up a condition that must be true for the user to use subsequent panels in the sequence? Can you programmatically check whether this condition is true? If so, specify a <Make Sure> command for that condition.

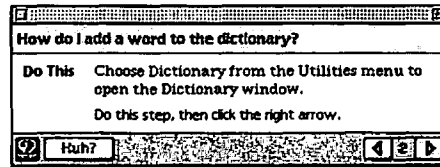
For example, assume you want to determine the context checks sequence for the SurfWriter topic “How do I add words to the dictionary?” shown in Figure 2-58.

Figure 2-58 The SurfWriter sequence for adding a word to the dictionary

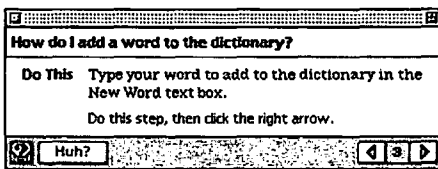
Panel 1



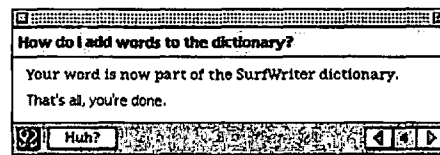
Panel 2



Panel 3



Panel 4



This sequence contains the typical panels for a topic that describes a task.

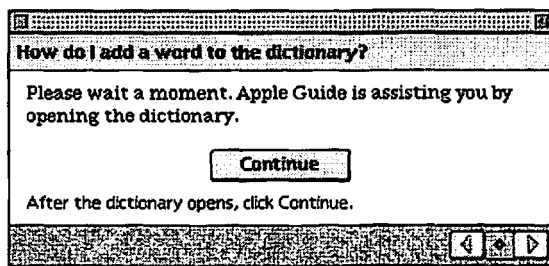
- Panel 1 is an introductory panel that describes the purpose of the sequence—adding a word to the dictionary—to the user.
- Panels 2 and 3 are action panels that each contain a step that the user should perform to accomplish the task:
 - Panel 2 instructs the user to open the SurfWriter dictionary.
 - Panel 3 asks the user to type a word.
- Panel 4 is a closure panel that tells the user the task is done.

First, consider panel 1. Apple Guide should not skip an introductory panel in a sequence, because it prepares the user for the task ahead. This panel does not require a <Skip If> statement. Furthermore, you should not specify a <Make Sure> for the first panel of a sequence. Panel 1, therefore, requires no context checks.

In contrast, panel 2 sets up a condition by telling the user to do an action (open the SurfWriter dictionary). You can verify whether this condition is true. You should therefore specify a <Skip If> command so that Apple Guide skips this panel if the SurfWriter dictionary is already open and then displays panel 3.

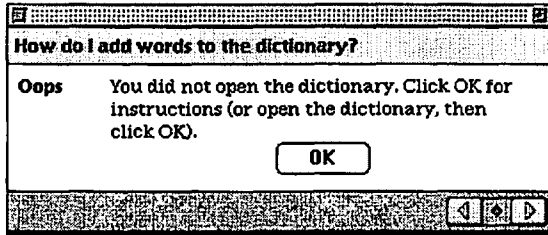
Panel 3 also tells the user to perform an action (type a word in the dictionary); here, a specific condition must be true (the SurfWriter dictionary must be open) for the user to perform this action. You should therefore specify a <Make Sure> command to verify that the SurfWriter dictionary is open before Apple Guide displays this panel. You can also specify a Continue panel if the <Make Sure> command results in a false condition. For example, Figure 2-59 shows the Continue panel that would appear if the user pressed the right navigation arrow on panel 2 without opening the SurfWriter dictionary.

Figure 2-59 A Continue panel for a condition in a SurfWriter sequence



(Notice that the Continue panel tells the user to wait until Apple Guide opens the SurfWriter dictionary.) This panel is associated with code (for example, an AppleScript command) that automatically opens the dictionary. After the user clicks the Continue button, Apple Guide then advances to panel 3 because the condition associated with the <Make Sure> command is true.

Alternatively, you can specify an Oops panel with the <Make Sure> command. For example, Figure 2-60 shows the Oops panel that would appear for the same condition.

Figure 2-60 An Oops panel for a condition in a SurfWriter sequence

If the user clicks OK without performing any other action, Apple Guide returns to panel 2, the first previous panel shown before the condition specified by the <Make Sure> command failed. If the user opens the SurfWriter dictionary and clicks OK, Apple Guide displays the next panel in the sequence.

Panel 4 does not depend on a condition being true and therefore requires no <Make Sure> command. In addition, Apple Guide should not skip a closure panel in a sequence, because it summarizes the task accomplished by the user. You should therefore not specify a <Skip If> statement for this panel. This panel therefore requires no context checks.

Evaluation of Context Checks

Apple Guide always evaluates context checks in a certain order, as shown in Table 2-3.

Table 2-3 Order in which Apple Guide evaluates context checks

Button pressed	Apple Guide movement through panels	<Skip If>	<Make Sure>
Right arrow	Normal forward	Evaluated	Evaluated after <Skip If>
Left arrow	Normal backward	Not evaluated	Not evaluated
Right arrow	Forward on Oops panel	Evaluated	Not allowed
Left arrow	Backward on Oops panel	Not evaluated	Not allowed
OK on Oops panel without user making condition true	Backward search	Not evaluated	Evaluated

If Apple Guide is moving forward through a sequence because the user clicks the right arrow, it evaluates <Skip If> commands before <Make Sure> commands. If Apple Guide is moving backward through a sequence (because the user is using the left arrow), Apple Guide ignores the <Skip If> and <Make Sure> commands and moves backward through panels regardless of their context checks. If the user clicks the OK button on an Oops panel without correcting the specified condition, Apple Guide returns the user to the **first previous panel** in the sequence. This panel is the first panel that Apple Guide finds, searching backward through the sequence, that meets one of two criteria: it does not have a <Make Sure> command specified for it, or it has a <Make Sure> command whose condition evaluates to true.

To find the first previous panel, Apple Guide searches backward through the sequence and checks each panel to see if it has <Make Sure> commands associated with it. If a panel does not have any <Make Sure> commands specified, Apple Guide displays the panel. If the panel does have <Make Sure>

Authoring Tips and Suggestions

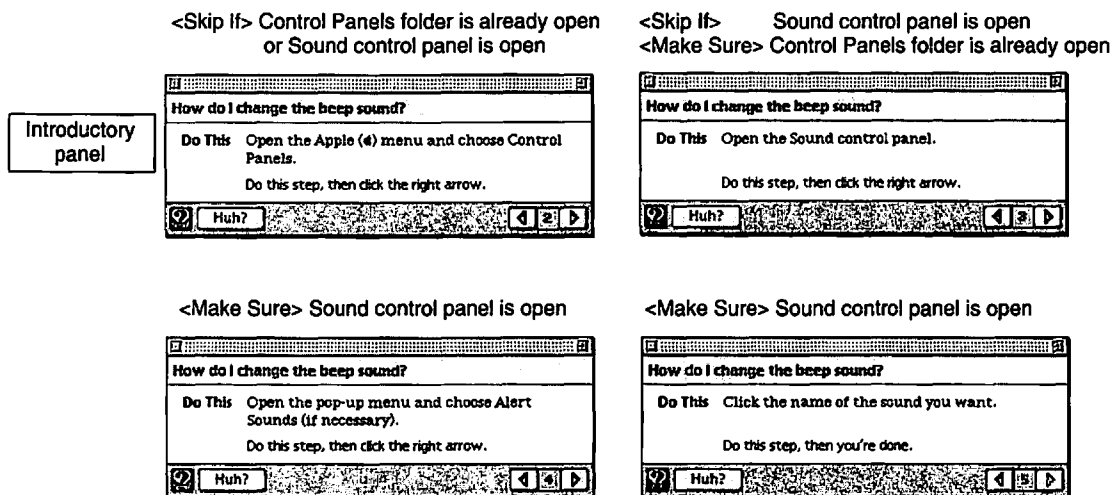
commands specified, Apple Guide evaluates whether each condition verified by the <Make Sure> command is true or false. If the conditions are all true, Apple Guide displays the panel. If the conditions are false, Apple Guide moves backward to the next panel. Apple Guide repeats this process until it finds a panel where one of the following conditions is true:

- The panel has no <Make Sure> commands specified.
- The conditions specified by the <Make Sure> command are all true for the panel.
- Apple Guide has reached the first panel of the sequence.

Apple Guide displays the first panel that meets one of these conditions to the user.

Figure 2-61 indicates the context checks for a Macintosh Guide sequence that describes how to change the sound on a Macintosh computer.

Figure 2-61 A sequence in Macintosh Guide for changing the beep sound



For this sequence, a Continue panel rather than an Oops panel appears whenever the user clicks the right arrow button on a panel where the specified condition is not true.

The introduction panel is the first panel in the sequence. Note that the third, fourth, and fifth panels all have a <Make Sure> command: the <Make Sure> command for the third panel specifies that the Control Panels folder be open and the <Make Sure> command for the fourth and fifth panels specifies that the Sound control panel be open.

Suppose that the user moves successfully through the first four panels without receiving an Oops panel; that is, the user opens the Control Panels folder and Sound control panel so that the conditions on the third and fourth panels are true. Now assume that, while viewing the fourth panel, the user closes the Control Panels folder and Sound control panel and then clicks the right navigation arrow. Because the next panel contains a <Make Sure> command that specifies that the Sound control panel should be open, Apple Guide displays an Oops panel. Assume that the user still does not open the Sound control panel but clicks the OK button on the Oops panel. Apple Guide now searches backward for the first previous panel in the sequence that either does not have a <Make Sure> command specified for it or that has a <Make Sure> command whose condition evaluates to true.

Apple Guide first goes back to the fourth panel, which contains a <Make Sure> command to specify that the Sound control panel be open. Because this condition is false, Apple Guide goes back to the third panel, which contains a <Make Sure> command to specify that the Control Panels folder be open. Because this condition is also false, Apple Guide goes back to the second panel. This panel does not contain a <Make Sure> command and is therefore the first previous panel that Apple Guide returns to.

Designing for Localization

By planning for localization as you design your guide file, you can more readily prepare your software for worldwide markets. In general, localization issues revolve around planning for

- cultural differences (such as the cultural significance of the images that you use for graphics or three-dimensional buttons)
- text that expands when translated to other languages
- specific elements (such as dialog items or folders) whose location or name changes based on the script in use

Authoring Tips and Suggestions

For specific information on various cultural values and localization principles, see the *Macintosh Human Interface Guidelines*. This section discusses issues you should take into account as you design your panels and implement features of your guide file.

Planning for Expanded Text

When writing your text, consider that U.S. English text can become up to 50 percent longer when translated into another language. Because you want your panels to remain small and easy to read, keep your panel text as terse as possible. To avoid expanded text overflowing onto graphics, use relative rather than absolute positioning when placing objects or graphics on a panel. (Remember that, by default, Apple Guide wraps the text around graphics and objects for you.)

Translations for Apple Guide Phrases

Once you design a guide file, part of the localization process involves translating all the text in your help content. This includes text in panels, text in radio buttons and checkboxes, topic areas and topics, and index terms. Guide Maker provides a Localize utility to help you with this process. For information on using the Localize utility, see the chapter “Localizing Your Guide File.” For information on specific Guide Script commands, see the chapter “Guide Script Command Reference.”

Part of localizing your content for a specific region includes localizing tags and button text. Table 2-4 shows common translations for Oops, Huh?, Tip, and Do This.

Table 2-4 Common translations of Apple Guide terms

Language	Oops	Huh?	Tip	Do This
Arabic	انتبه	ماذا؟	نصيحة	نفذ الآن
Brazilian Portuguese	Oops	Hein?	Dica	Faça isto
Bulgarian	One	Кво?	Съвет	Направете следното
Croatian	Jaol	Hm...	Trik	Postupak
Czech	No tohle	Cože	Tip	Provedte
Danish	Hov	Mere	Tip	Sådan
Dutch	Let op	Hè?	Tip	Doe dit
Finnish	Oho	Mitä?	Vinkki	Tee näin
French	Attention	Infos	Conseil	Action
German	Hinweis	Noch Fragen?	Tip	Aktion
Hungarian	Hopp	Tessék	Ötlet	Tegye ezt
Icelandic	Úps	Ha	Ábending	Gerið þetta
Italian	Ops	Come?	Consigli	Fai così
Japanese	しまった!	えっ?	ヒント	操作
Korean	아차	그런가?	비밀	이렇게
Norwegian	Å neil	Hva?	Tips	Gjør dette
Polish	Błąd	Co?	Wskazówka	Zrób tak
Romanian	Opa	Hă	Sfat	Fă Asta
Russian	Внимание	Как	Совет	выполните
Spanish	¡Ojo!	¿Qué?	Consejo	Haga esto
Swedish	Ojdå!	Mer	Tips:	Gör så här:
Turkish	Dur	Ne	İpucu	Şunu Yapın

Formats

Specify font information using <Define Format> commands rather than embedding font and style information within your source files. When you export and merge text strings (using Guide Maker's Localize utility) all style information is lost and any style information directly specified in the source file must be reapplied.

In general, try to use a small number of formats, such as Tag, Body, and Full. Using a large number of formats in a panel typically requires that the localizer readjust the formats to fit expanded text.

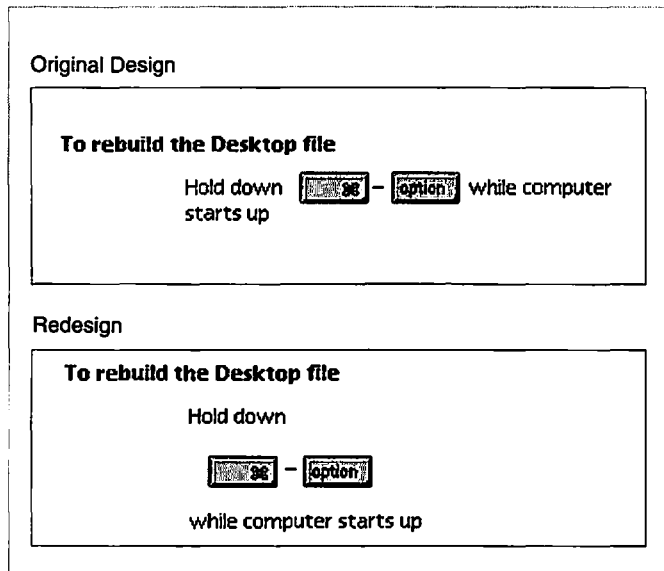
Avoid using multiple tags (for example, two Do This tags) on a panel. Using multiple tags often requires that the localizer line up the translated body text with the tags.

Graphics and Buttons

Always use relative positioning of graphics, checkboxes, and buttons on panels; otherwise, translated text can expand and overflow the object.

Do not embed pictures inside a line of text, because if the text grows during localization, you will have to adjust the placement of the pictures. Figure 2-62 shows an example of a graphic with embedded text that was redesigned for better localization.

Figure 2-62 Avoiding embedding pictures inside of text



Sequence Display Titles

When defining a sequence with the <Define Sequence> command, you must specify the optional second parameter. If you don't, Guide Maker's Localize utility won't extract the text string of the sequence display title (and your localized panels will have the same sequence display title as your original panels).

Coachmarks

Give your coachmarks meaningful names and include comments in the source file describing the object being coached. When defining item coachmarks in dialog boxes, use the `DialogID` function rather than specifying a fixed rectangle coordinate because the item's location might change if the dialog box is localized.

Context Checks

When using the `<Define Context Check>` command, always define strings that need to be translated as having the data type `LPSTRING`, not `PSTRING`. Guide Maker's Localize utility extracts strings that are specified by the type `LPSTRING` but not strings specified by the type `PSTRING`. Don't hard code any strings in external code modules. Instead, specify the string (with data type `LPSTRING`) in the `<Define Context Check>` command line.

AppleScript

When writing scripts (for use with AppleScript or other scripting languages) that reference special folders (such as the System Folder, Extensions, or Control Panels), you don't need to use a path reference to the folder name. Instead, use the built-in folder constants (`extensions folder`, `control panels folder`, or `system folder`). See the *AppleScript Finder Guide* for more information on these and other folder names.

Planning Your Help Content

Contents

Determining and Creating Your Guide File Content	3-3
Determining Appropriate Content for Your Guide File	3-4
Creating Topic Areas and Topics	3-4
Using Flowcharts to Design Your Guide File Panels	3-7
Helping the User Search	3-12
How Apple Guide Stems	3-13
How Apple Guide Matches Search Phrases With Topics	3-15
Creating a Guide File Index and Associated Lists	3-19
Creating a Guide File Index	3-20
Invisible Index Terms	3-22
Creating an Ignore List	3-23
Creating an Exception List	3-24
Creating a Synonym List	3-24

You'll find developing your help system much easier if you plan the content and organization of each guide file before you prepare it with Guide Script. First, include time in your online development schedule to determine your guide file topic areas and topics, to break them into panels and sequences, and to design your help instructions. Then develop your guide file in a hard copy format—for example, a series of flow charts or story boards—that can serve as your road map during the scripting phase. If you create a guide file using the Full Access window, you also need to plan and create Index and Look For content.

This chapter helps you with all of these tasks. It describes the type of content that is best suited for your guide files. It then explains how to derive topic areas and topics from existing reference documentation. Next, it shows how to break these topic areas and topics into panels and sequences using flowcharts. It concludes with information on creating a guide file index and Look For features. This information includes a description of how Apple Guide matches user search phrases to topics using the guide file index, three additional lists of terms, and a process called stemming.

You should read this chapter if you are designing or scripting a guide file.

Determining and Creating Your Guide File Content

The first step in developing a guide file is to determine its content. This section suggests a two-part approach to that task. First, determine the content that your guide files should contain. Next, if possible, analyze a list of headings derived from existing reference documentation to determine topic areas and topics.

Note

The help focus and content differ for each of the five guide file types—About, Tutorial, Help, Shortcuts, and Other—supported by Apple Guide. For example, the About, Shortcuts, and Tutorial guide files generally contain a small number of one-level topics, whereas the Help guide file contains an extensive number of multilevel topics. If you are not already familiar with the content requirements for each guide file, see Chapter 2. ♦

Determining Appropriate Content for Your Guide File

Remember that not all material is appropriate for your guide files. In general, place in your guide files any information that answers these questions:

How do I accomplish this task?

Why can't I accomplish this task?

What is the meaning of this term?

Other information is usually better presented in reference documentation. This information includes instructions that the user needs when the computer is not turned on or working properly, or overly complex material. For example, do not include in your guide files

- installation instructions (for example, how to install your application or any other application used with it)
- extensive reference material (for example, a complete programming language reference)
- safety information (for example, how to avoid injury)

Creating Topic Areas and Topics

If you have reference documentation for your application, you can use its headings to create your topic areas and topics. Here is an example of the table of contents from the documentation describing the SurfWriter application:

Copying Text

Pasting Text

Opening a File

Saving a File

Selecting Different Styles for Your Text

Changing Your Default Preferences

Using Dictionaries

Getting Started With SurfWriter

Introducing the Menus

Planning Your Help Content

Introducing the Toolbar

Introducing Icons

Introducing the Windows

Creating Your First Document

Command Keyboard Equivalents

The first seven headings describe tasks that many users might want to know about and that would be appropriate content for a guide file of type Help, which uses the Full Access window. Remember that with Topics selected, the left column of the window shows topic areas. You therefore need to break the headings into topic areas, and then topics. For example, you might derive from those headings the following five topic areas:

Copying & Pasting

Opening & Saving Documents

Styles

Setting Preferences

Using the Dictionary

Notice that in the first two, two headings were combined to make a single topic area.

You can now break the topic areas into one topic or several topics. For example, you might break the topic area "Using the Dictionary" into the following topics:

How do I add a word to the dictionary?

How do I look up a word?

How do I create a custom dictionary?

How do I add or remove a dictionary?

In contrast, the eighth heading of the SurfWriter manual—"Getting Started With SurfWriter"—and its five subheadings—"Introducing the Menu," "Introducing the Toolbar," "Introducing Icons," "Introducing the Windows," and "Creating Your First Document"—contain information that introduces users to basic features of the application. It is therefore more appropriate information for a Tutorial guide file than for a Help guide file. Here, you typically use a Single List Access window, which presents a small number of

Planning Your Help Content

focused topics. For example, from the five subheadings you can create a Tutorial with these topics:

- Using Menus

- Using the Toolbar

- Using Icons and Windows

- Creating a Document

Finally, the ninth heading—“Command Keyboard Equivalents”—contains a list of keyboard commands for various application features. Such quick reference information belongs in a Shortcuts guide file, using either a Single List or Simple Access window. You can create, for example, a Simple Access window with three-dimensional buttons that lead to the following topics:

- Working with Menus

- Working with the Toolbar

- Working with Icons

- Working with Windows

- Editing Your Document

Note

If possible, minimize the number of topic areas and topics in the access window so that the user does not have to scroll the columns to view them. ♦

Using Flowcharts to Design Your Guide File Panels

One way to plan your guide file content is to create a flowchart, in which guide file components are represented by graphic elements (for example, a topic is a circle shape, a related panel is a rectangle shape, and a branch point is a diamond). With a flowchart, you can easily view the organization of your guide file, including the topics under each topic area, the tasks under each topic, the branches in a single sequence of panels, and the panels shared among different sequences.

To create a flowchart, you should know how to

- **design a sequence**

For example, you should know the difference between the first and second levels of panels in a sequence. For more information, see “Designing a Sequence” beginning on page 2-66.

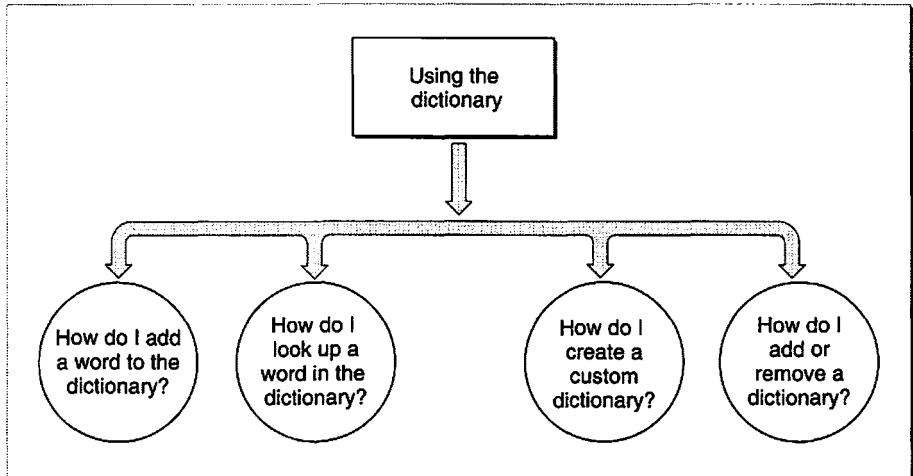
- **use the panel types recommended by Apple for different categories of help instructions**

For example, you should know when to use an action panel. For more information, see “Designing Panel Types” on page 2-50.

- **determine context checks**

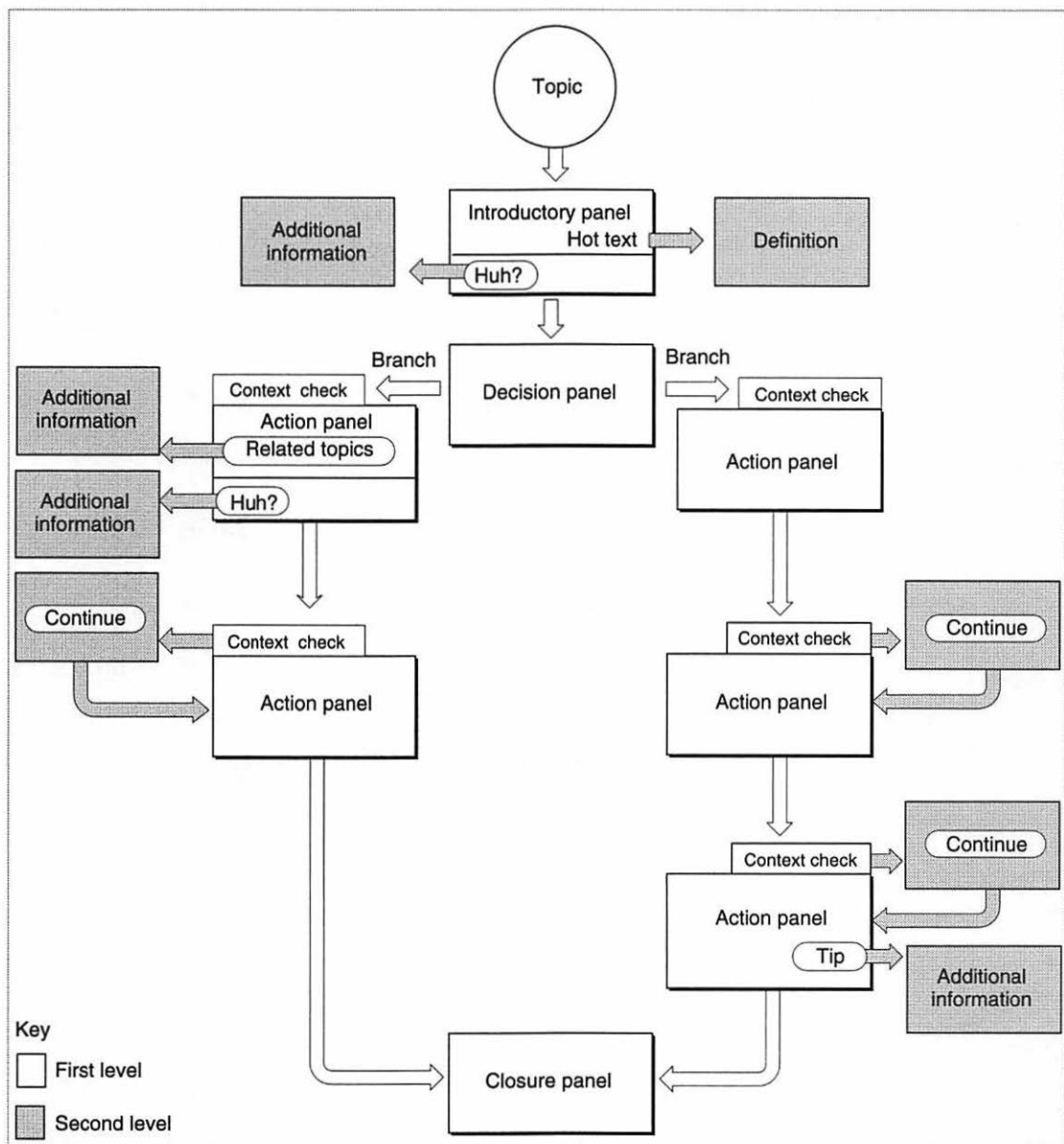
For example, you should be able to determine whether Apple Guide should skip a panel if the condition on it is true. For more information, see “Using Context Checks” on page 2-83.

If your guide file uses a Single List or Simple Access window, you need to create flowcharts that show each topic and its related panels. If your guide file uses a Full Access window, you need to create flowcharts that also break topic areas into topics. Figure 3-1 contains a flowchart for the topic area “Setting Preferences,” which appears in the SurfWriter Full Access window.

Figure 3-1 A flowchart that breaks topic areas into topics

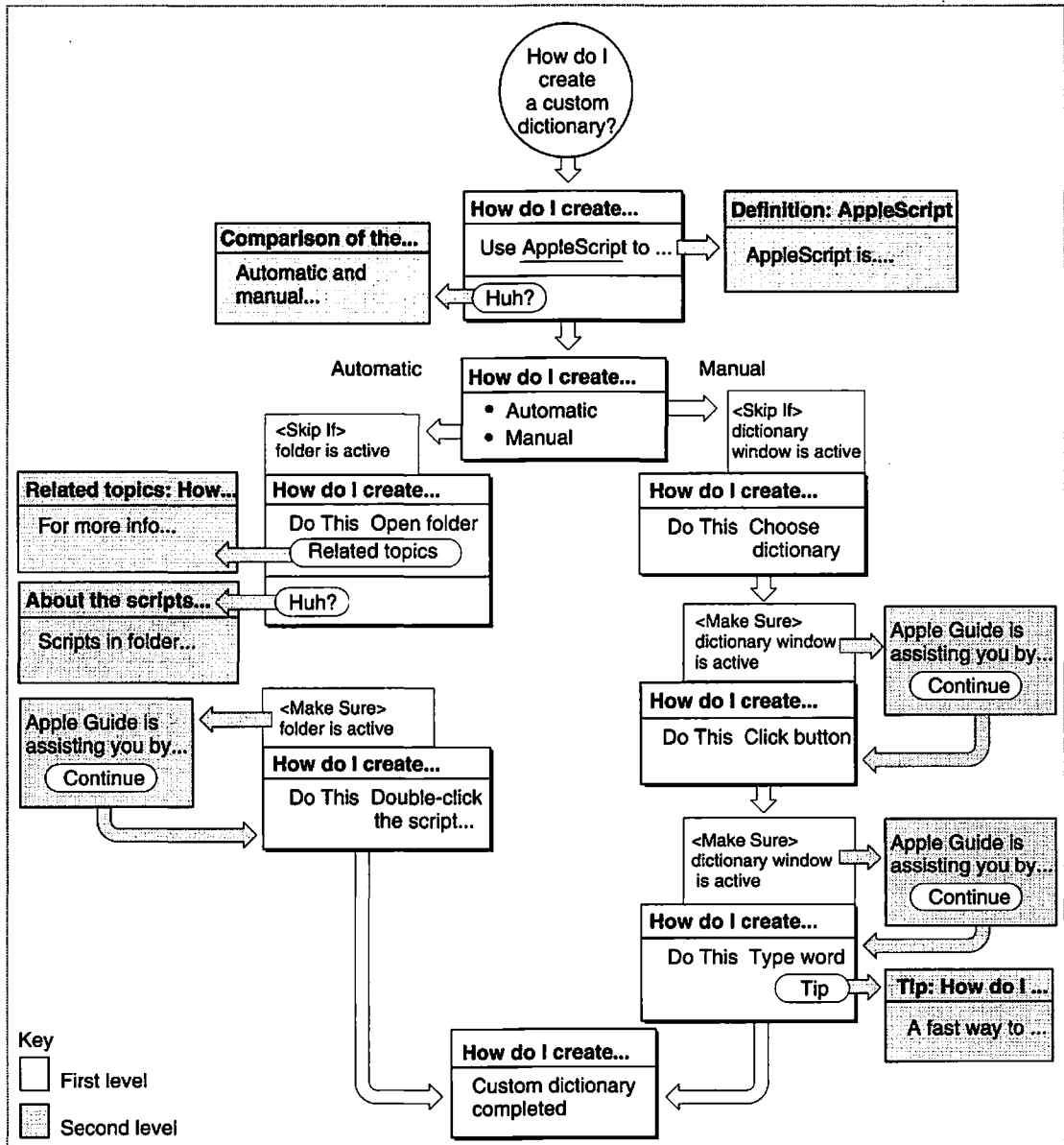
The flowchart breaks the topic into four separate topics, with a circle representing each one. This flowchart is simple and therefore requires few graphical elements. To break each of these topics into a sequence requires more complex elements, as shown in Figure 3-2.

Figure 3-2 A typical flowchart



Planning Your Help Content

One of the topics shown in Figure 3-1— “How do I create a custom dictionary”—is broken into a sequence that contains two branches, shown in Figure 3-3.

Figure 3-3 A flowchart for a SurfWriter sequence on creating a custom dictionary

Planning Your Help Content

In this example, one branch shows the user how to create a custom dictionary manually, and the other branch shows the user how to create a custom dictionary using a script. Notice that the first two panels in the sequence—the introductory and decision panels—are shared because they apply to both branches in the sequence. The introductory panel introduces the user to the topic, and the decision panel presents the user the choice of creating the dictionary using either the menu command or the script method. Until the last panel is reached, the next panels for each branch contain information specific to each branch and are therefore not shared. The last panel tells the user information that applies to both branches—specifically, that the dictionary contains the new word—and is therefore shared.

The flowchart also shows the required context checks. Notice that the two Continue panels on the right are for the same condition and could therefore be a shared panel. For more information on specifying context checks, see Chapter 2.

Helping the User Search

The Look For feature provides a powerful search facility that, in combination with information you provide, allows the user to enter one or more words—even a complete sentence—and obtain a list of topics related to a given search phrase. For example, when a guide file contains an index and Look For content, the user can give Apple Guide a phrase such as “print”, “printing documents”, or “How do I print my documents”, and receive a list of topics related to that task.

With Look For features selected in the Full Access window, the user can enter a search phrase in a search phrase entry box located in the left column of the window. After receiving the search phrase, Apple Guide searches the guide file index for one or more matching entries. (The guide file index contains entries that point to each guide file topic and is the same one you must create to appear in the left column of the Full Access screen when the user selects Index features.) If found, Apple Guide displays the associated topics in the right column of the window.

Apple Guide automatically reduces common word variations in search phrases to their root words. For example, Apple Guide reduces the word “copying” to “copy”. This process is known as **stemming**. To further enhance the

effectiveness of the Look For feature, you should provide Apple Guide with three lists: ignore, exception, and synonym. With these lists, you can control whether Apple Guide stems certain user search phrases and ensure that Apple Guide finds matches for search phrases containing terms not found in the index.

- The **ignore list** contains words that Apple Guide should remove if they appear in the user search phrase. For example, you can tell Apple Guide to remove from the search phrase the words “How”, “do”, and “I”, which appear in the topic headings recommended by Apple.
- The **exception list** contains words that you do not want stemmed and words whose stemmed version matches another index term.
- The **synonym list** contains words that have identical meaning to index terms but that do not appear in the index. For example, you can make the word “clone” a synonym for the index term “copying”.

You need to create your guide file index in a way that best accommodates the use of these lists. In addition, you can also create *invisible index terms*. Apple Guide uses these terms to match search phrases with topics that do not appear in Apple Guide’s *visible* index.

If you do not create your index properly or create the associated lists, your users are likely to encounter significantly more failed searches.

How Apple Guide Stems

Apple Guide stems the user’s search phrases by removing suffixes from words. These suffixes include “ion”, “al”, “s”, “ies”, “ing”, “ed”, “ize”, and double occurrences of the same letter, for example, “ll”. In some cases, Apple Guide automatically adds an “e” to the end of the stemmed word. For example, Apple Guide stems “filing” by removing the “ing” but then adds an “e” to the result (“fil”) to create the word “file”.

Apple Guide also automatically stems punctuation for the user’s search phrase. For example, “random-access memory” becomes “random access memory”.

Planning Your Help Content

Table 3-1 shows some words that Apple Guide stemmed. You can also use Guide Maker's Test Look For utility to see how Apple Guide stems words and parses phrases.

Table 3-1 Some words stemmed by Apple Guide

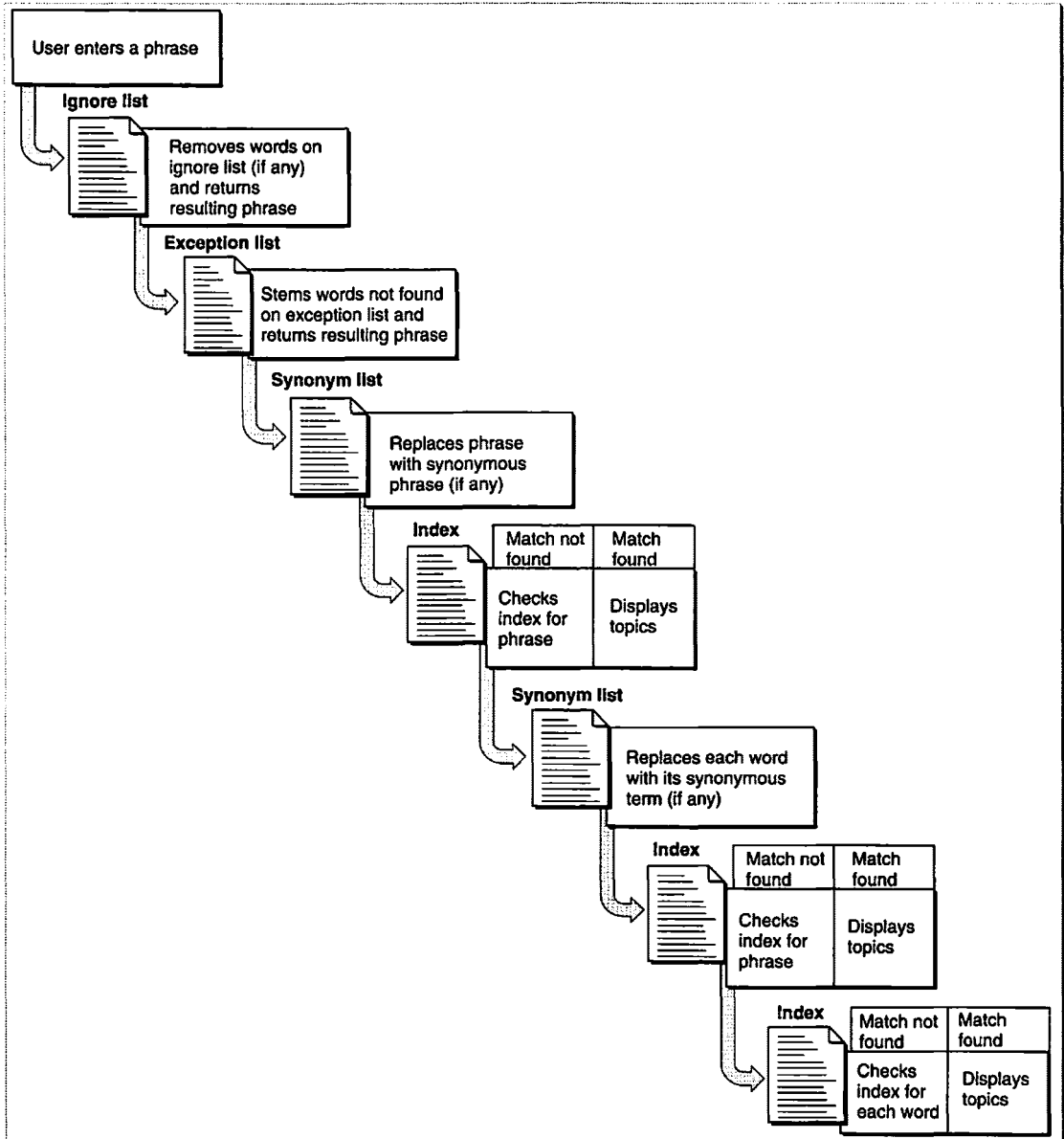
Stemmed suffix	Search word	Stemmed word
ion	Definition	Definit
al	Retrieval	Retriev
s	Privileges	Privilege
ies	Utilities	Util
ing	Removing	Remov
ing, "e" added	Typing	Type
ing, "e" not added	Pasting	Past
double letters	Trackball	Trackbal
ed	Shared	Share
er + ing	Recovering	Recover
ize	Organize	organ

IMPORTANT

Apple Guide uses the stemming methods described here only for guide files that specify the command <WorldScript> 0,0. By default, Apple Guide does not perform stemming for non-Roman and non-U.S. scripts. For additional information on localization, see "Designing for Localization" beginning on page 2-91. ▲

How Apple Guide Matches Search Phrases With Topics

After Apple Guide receives a search phrase from the user, it tries to find a match by looking in the index and in the associated ignore, exception, and synonym lists. Figure 3-4 shows the actions Apple Guide typically performs after receiving a search phrase.

Figure 3-4 An Apple Guide search in response to a search phrase

Notice that Apple Guide first checks the ignore list and removes designated ignore words, if any, from the search phrase. Next, Apple Guide checks the exception list to see which words it should not stem in the resulting phrase. Apple Guide then stems other words in the phrase, if appropriate.

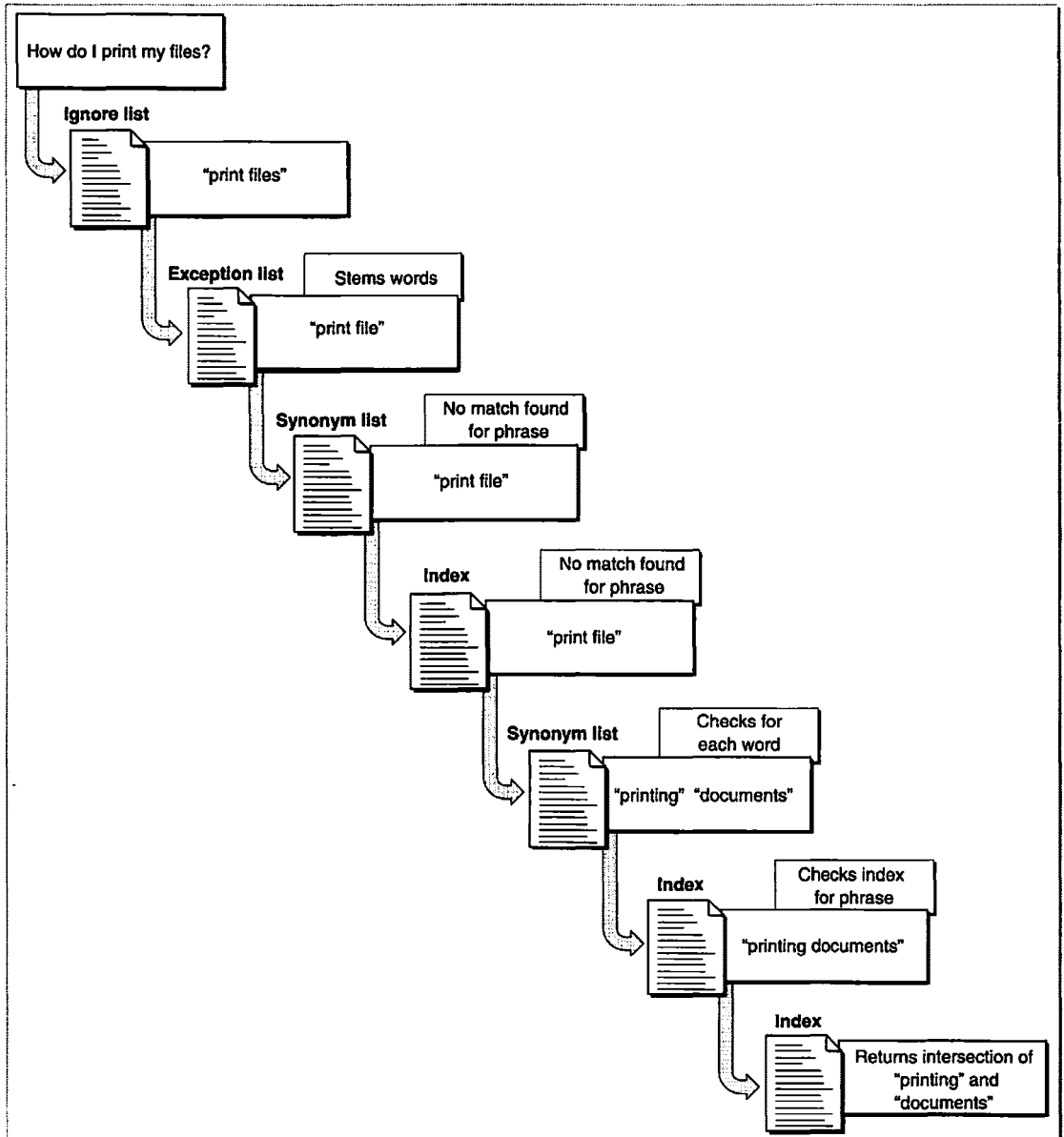
Now Apple Guide searches the synonym list for the phrase. If it finds a synonym, it replaces the phrase with the equivalent phrase and searches the index for a match. If Apple Guide finds a match for the entire phrase, it displays the associated topics; otherwise, it searches the synonym list for each word in the phrase and replaces each one with its index term equivalent. It then searches the index for the phrase and, if found, displays topics. If not found, it creates a list of topics for each word in the phrase, and displays the intersection of these lists.

If Apple Guide cannot find a match for one word in a multiple word search phrase, the entire search fails, and it displays a message telling the user to try again. Apple Guide's success matching multiple word search phrases depends on how effectively you use the synonym list.

For example, suppose that you have provided Apple Guide with these ignore and synonym words:

Ignore words	Synonym words
how	print = printing
do	file = documents
I	
my	

Now suppose that the user enters the search phrase "How do I print my files?" Figure 3-5 shows how Apple Guide searches for topics to match this phrase.

Figure 3-5 A typical Look For search

For this example, Apple Guide performs the following steps:

1. Apple Guide removes from the search phrase the four words specified in the ignore list ("how", "do", "I", and "my"). The resulting phrase is "print files".
2. Apple Guide checks the exception list to verify which words in the phrase "print files" it should not stem. The phrase contains no exception words.
3. Apple Guide stems the word "files" to "file". The resulting phrase is "print file".
4. Apple Guide searches the synonym list for the entire phrase "print file" and finds no synonyms. The phrase remains "print file".
5. Apple Guide searches the index for the phrase "print file".
6. Apple Guide searches the synonym list to match each word in the phrase. As a result, it replaces the word "print" with "printing", and replaces the word "file" with "documents".
7. Apple Guide searches the index for a match for the entire phrase, "printing documents", but doesn't find it.
8. Apple Guide searches the index for each word in the phrase "printing documents" (that is, "printing" and "documents") and finds matching index terms.
9. Apple Guide displays the intersection of these topics.

The rest of this section gives you guidelines for creating an index and the three associated lists. Apple strongly recommends that you also use the Look For feature in Guide Maker to test and develop your Look For content. For more information, see Chapter 6 in Part 2.

Creating a Guide File Index and Associated Lists

Your guide file index completes two features of the Full Access screen; it appears in the left column of the screen when Index features are selected, and Apple Guide uses it to associate user search phrases with topics when Look For features are selected. When you create a guide file index, you need to consider both these uses. For the Index feature, you should create index terms that correspond appropriately to your guide file content. For the Look For feature, you should determine the effects of Apple guide stemming and the interaction of the three associated lists—ignore, exception, and synonym—on the index

terms. If you do not understand all these components, you will get unexpected or erroneous results.

The rest of this section provides guidelines for creating a guide file index, including the use of invisible index terms. It then describes each of the associated lists.

Creating a Guide File Index

To retrieve topics for the user when Look For or Index features are selected in the Full Access screen, Apple Guide uses a guide file index. A **guide file index** is a list of terms you create that point to the topics appearing in the right column of the Full Access window.

If you have created an index for reference documentation, you are probably familiar with common indexing conventions. You generally create an index entry for all nouns or noun phrases that name a topic, and use conjunctions and prepositions only to clarify the relationship between a main entry and subentry. For example, to create an index entry for the section "Files and Folders", you ignore the word "and" and create a separate index entry for the words "files" and "folders". Your index typically includes entries for incorrect terms with a reference to the correct index term. For example, an index can include an entry that says "site dictionary, *see* dictionary."

In contrast, your guide file index consists of only main index terms and no subentries, a practice that virtually eliminates the use of conjunctions and prepositions. In addition, you generally create index terms only for words that actually appear in your topics. To accommodate user search phrases containing terms that do not appear in the index, you generally create synonyms. For more information, see "Creating a Synonym List" beginning on page 3-24.

You create guide file index terms using nouns, noun phrases, and verbs that name topics. In general, use the plural form of the noun, even if it does not appear in the topic name. For example, for the topic "How do I save a memo?", use the plural form "memos" for the index term. Similarly, use the form of the verb that ends with "ing", even if it does not appear in the topic name. For example, for the topics "How do I save my files?" and "How do I open a file?", use the forms "saving" and "opening" for index terms.

Table 3-2 shows some sample topics and their associated index terms.

Table 3-2 Examples of index terms derived from sample topics

Topic	Index terms
How do I save my document?	saving, documents
How do I add a word to the dictionary?	dictionary
How do I create a memo?	memos
How do I edit my files?	editing, documents
How do I copy my files?	copying, documents
How do I set the default margins?	margins
How do I set the page number prefix?	page numbers
How do I set default footer text?	footers
How do I set automatic page breaks?	page breaks
How do I print my document?	printing, documents

As you create your index, you need to answer the following questions:

- Does Apple Guide stem the term?
- Does the term have other forms? If so, are the stemmed versions of these forms inappropriate?
- Does the term have a plural form?
- Is there an alternative word, spelling, abbreviation, or acronym for the term?

If the answer to any of these questions is yes, then create a synonym for the word. (You can also use the exception list or invisible index term.) For example, consider the index term “printing”. Apple Guide stems this term to “print”. The term “printing” has other forms, “print” and “prints”, which Apple Guide also stems to “print”. To match these words to the appropriate index term, you can specify that “print” is a synonym for “printing”.

Alternative words or phrases that the user might enter for “printing” include “print command”, “printer”, and “spool”. By specifying each of these words as

Planning Your Help Content

synonyms for the index term “printing”, you can allow the user greater flexibility in entering search phrases and increase Apple Guide’s ability to provide a successful match.

For complete information, see “Creating a Synonym List” beginning on page 3-24, as well as the next section.

Invisible Index Terms

When users view the guide file index in the left column of the Full Access window with Index features selected, they should see only index terms that correspond to the terms used in the guide file topics. The terms in the left column are **visible index terms**. To accommodate Look For search phrases that do not correspond to actual index terms, use invisible index terms. Although an **invisible index term** does not appear in the left column of the Full Index screen with Index selected, Apple Guide uses the term to match a search phrase with guide file topics.

You should use invisible index terms to have Apple Guide provide results that it cannot derive from the synonym list or through normal intersections of topics. For example, you should create an invisible index term to have Apple Guide

- Accommodate search phrases that the user is likely to enter but that have no corresponding index term or appropriate synonyms. For example, if the word “customizing” is not synonymous with any index terms and Apple Guide cannot retrieve any of its associated topics through an intersection of other words, make it an invisible index term.
- Provide logical topic groupings that Apple Guide cannot derive from intersections. For example, suppose you specify “bitmap font” as a synonym for “fonts” and “bitmap graphic” as a synonym for “graphics”, and the index terms have no topics in common. In this case, you can create an invisible index term for “bitmap” that includes relevant topics from the index terms “fonts” and “graphics”.
- Provide topics if the user enters a search phrase that describes topics by class rather than by their specific names. For example, if the user enters the search word “tips”, you would provide all tip sequences in the guide file.

Creating an Ignore List

You should place in the ignore list words that you want Apple Guide to remove if they appear in the user search phrase. In general, these words should be ones that the user is likely to enter as a search phrase but that are inappropriate as index terms and do not warrant synonyms. The list should generally include

- all words from the phrases in your topic headings (for example, “how”, “do”, and “I”)
- other words commonly used in search phrases
- contractions (“didn’t”, “doesn’t”, “don’t”)
- prepositions (“of”, “with”, “in”, “on”, “without”, “during”, “except”, “for”, “from”)
- adverbs (“especially”, “exactly”, “frequently”, “forward”)
- articles (“a”, “an”, “the”)
- adjectives (“each”, “entire”, “few”, “fewer”, “first”)
- conjunctions (“and”, “but”, “than”)
- pronouns (“everything”, “each”, “my”, “that”, “those”, “their”, “these”, “your”, “its”)

Note

Apple Guide automatically ignores numerals. ♦

For example, Table 3-3 shows the ignore words for some sample guide file topics.

Table 3-3 Example of ignore words derived from sample topics

Topic	Ignore word candidates
How do I save my document?	How, do, I, my
How do I add a word to the dictionary?	How, do, I, a, to, the,
How do I create a memo?	How, do, I, a
How do I edit my files?	How, do, I, my

continued

Table 3-3 Example of ignore words derived from sample topics (continued)

Topic	Ignore word candidates
How do I set the default margins?	How, do, I, set, the
How do I set the page number prefix?	How, do, I, set, the
How do I set default footer text?	How, do, I, set
How do I set automatic page breaks?	How, do, I, set
How do I print my document?	How, do, I, my

Creating an Exception List

In the exception list, you should place only words that you do not want stemmed and for which the stemmed version matches another index term. For example, assume that you specify “custom” as a visible index term and “customizing” as an invisible index term, with different topics associated with each index term. To prevent Apple Guide from stemming “customizing” to “custom”, you should place “customizing” on the exception list.

You should use the exception list for no other reason. To handle unwanted effects of stemming, you generally should use the synonym list (described next). By default, Apple Guide does not use the exception list for non-Roman and non-U.S. script systems.

Creating a Synonym List

In the synonym list, you should place words that have identical meaning to index terms but that do not appear in the index. You generally use the synonym list more than the exception list or invisible index terms. It provides the most efficient method for handling unwanted effects of stemming and incorrect usage in search phrases.

You can create a synonym for regular and invisible index terms. You can also create a multiword synonym for an index term that is a compound phrase. When you do, however, the search obtains the expected result only if the user search phrase is the exact multiword synonym, with no other words. You should use these guidelines to create a synonym:

- For an index term that Apple Guide stems, specify its stemmed version as a synonym to that index term. For example, Apple Guide stems the term

"editing" for "edit". You therefore specify the word "edit" as a synonym to "editing". This convention applies to all plural forms of index terms, which Apple Guide stems to the root word. For example, Apple Guide stems the term "formats" to "format". You therefore specify "format" as a synonym for "formats". (Optionally, you can place a plural form of the index term on the exception list or create an invisible index term for the stemmed version.) Follow this convention even if the stemmed version of the index term is not an actual word. For example, Apple Guide stems the word "searching", to "searche". You should therefore specify "searche" as the synonym for "searching", even though it is an incorrect spelling.

- For an index term that has other forms that do not stem appropriately or match the term, specify these forms as synonyms for that index term. For example, the index term "retrieving" uses the forms "retrieve", "retrieves", and "retrieval". Apple Guide stems "retrieving" to "retriev", "retrieval" to "retriev", and "retrieves" to "retrieve", but it does not stem the word "retrieve". To ensure that Apple Guide finds a match if the user enters any of these three forms as search phrase, specify "retriev" and "retrieve" as synonyms for "retrieving." In some cases, you need only one synonym to cover several forms. For example, Apple Guide stems both "printing" and "prints" to "print". You can therefore cover all forms of "printing" by specifying "print" as its synonym.
- For an index term that has the same meaning as other words not included in the index, specify the other words as synonyms to that index term. For example, a user might enter the search phrase "duplicate" instead of the index term "copying". Note that "duplicate" is itself stemmed to "duplic"; therefore specify "duplic" as a synonym for "copying".
- For an index term that has alternative wordings or spellings, specify these as synonyms for the correct term. For example, specify "catalogue" as a synonym for "catalog" and specify "tool bar" as a synonym for "toolbar". If Apple Guide does not appropriately stem the alternative word, specify a synonym for this stemmed version.
- For an index term that has an abbreviated version, make it a synonym for the index term. For example, specify "info" as a synonym for "information".
- For an index term that has a foreign word with the same meaning, make the foreign word a synonym.

Planning Your Help Content

- For an index term that has an acronym, specify the acronym as a synonym for the spelled out version. For example, specify “RAM” as a synonym for “random access memory”.
- For an index word that has a slang word with the same meaning, make the slang word a synonym for the index term. For example, specify “crash” as a synonym for “system failure”.

Table 3-4 shows synonyms derived from sample topics.

Table 3-4 Examples of synonyms derived from sample topics

Topic	Index terms	Synonyms
How do I save my document?	saving	save
How do I add a word to the site dictionary?	dictionary	site dictionary
How do I create a memo?	memos	memo
How do I edit files?	editing	edit
How do I set the default margins?	margins	margin
How do I set the page number prefix?	page numbers	page numbers
How do I set default footer text?	footers	footer
How do I set automatic page breaks?	page breaks	page breaks
How do I print my document?	printing	print

PART TWO

Building Guide Files

Introduction to Guide Maker

Contents

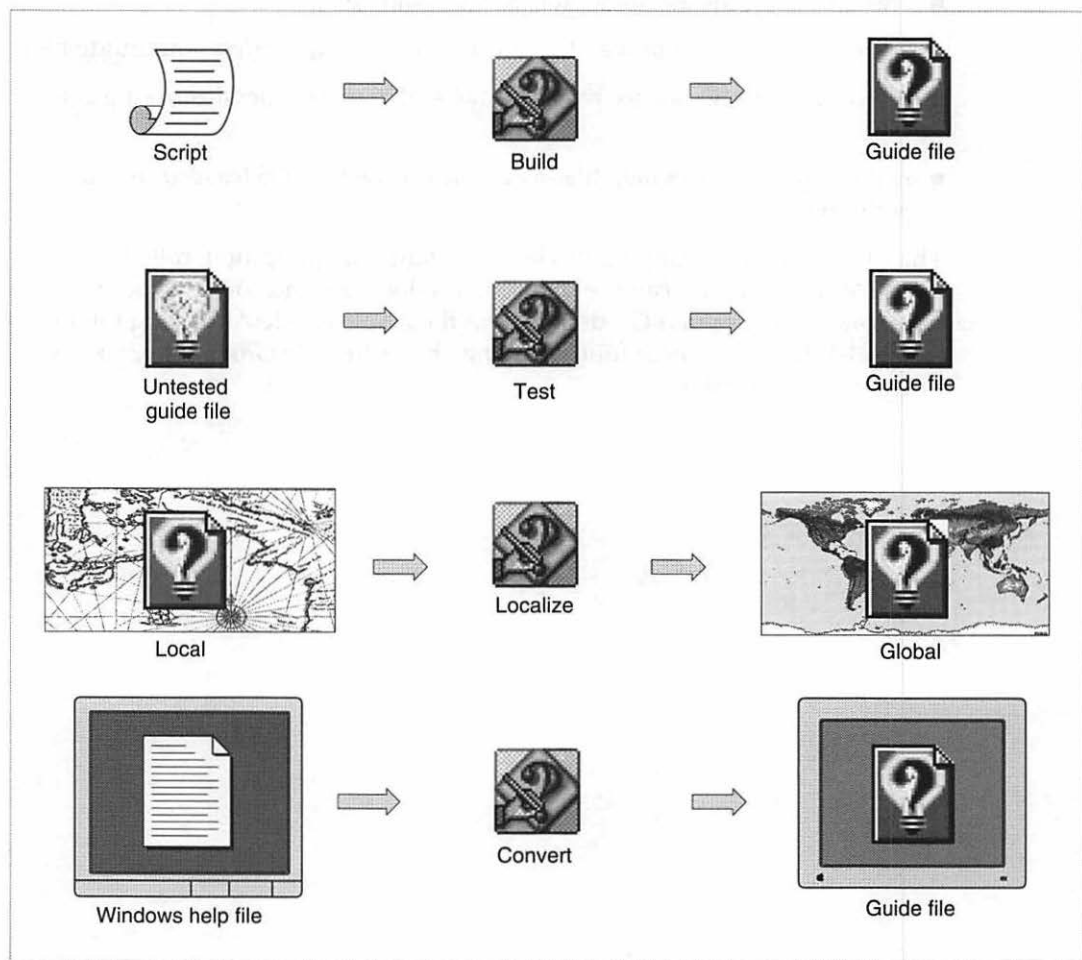
Guide Maker Overview	4-5
Which Chapter Should I Read?	4-8

In Part 1 of this book, you learned about Apple Guide and about designing your Apple Guide online assistance. This part of the book describes how to

- create your online assistance (*build* your guide file)
- ensure that your online assistance works as you expect (*test* your guide file)
- get your online assistance ready for the global market (*localize* your guide file)
- make your Windows help files available for the Mac OS (*convert* to Guide Script files)

The CD-ROM accompanying this book contains an application, called Guide Maker, that you can use to build, test, localize, and convert your online assistance. (You can find Guide Maker in the Apple Guide:Authoring folder.) Figure 4-1 illustrates these four tasks and shows the role Guide Maker plays in accomplishing them.

Figure 4-1 Building, testing, localizing, and converting your online assistance using Guide Maker

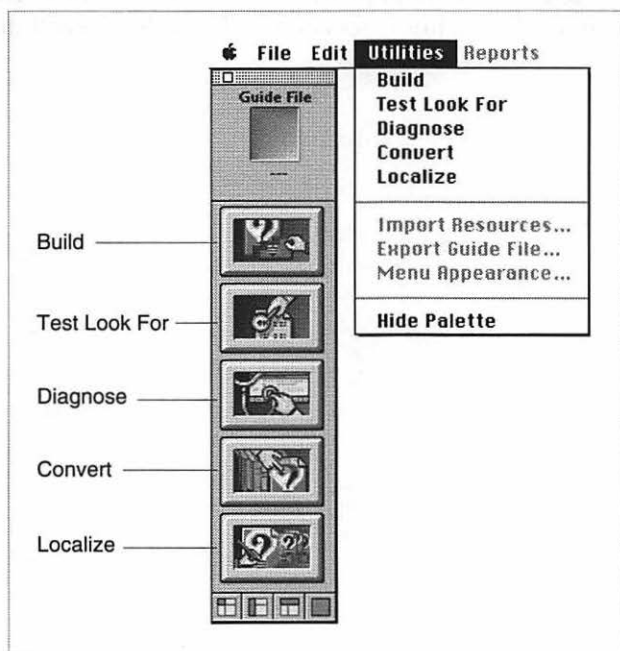


The next section gives an overview of Guide Maker.

Guide Maker Overview

When you launch Guide Maker, besides the menus, you see a palette with five buttons. This palette represents the Utilities menu graphically and gives you another way of selecting that menu's utilities. To access a utility, you can click the button that represents it in the palette or choose it from the menu itself (see Figure 4-2).

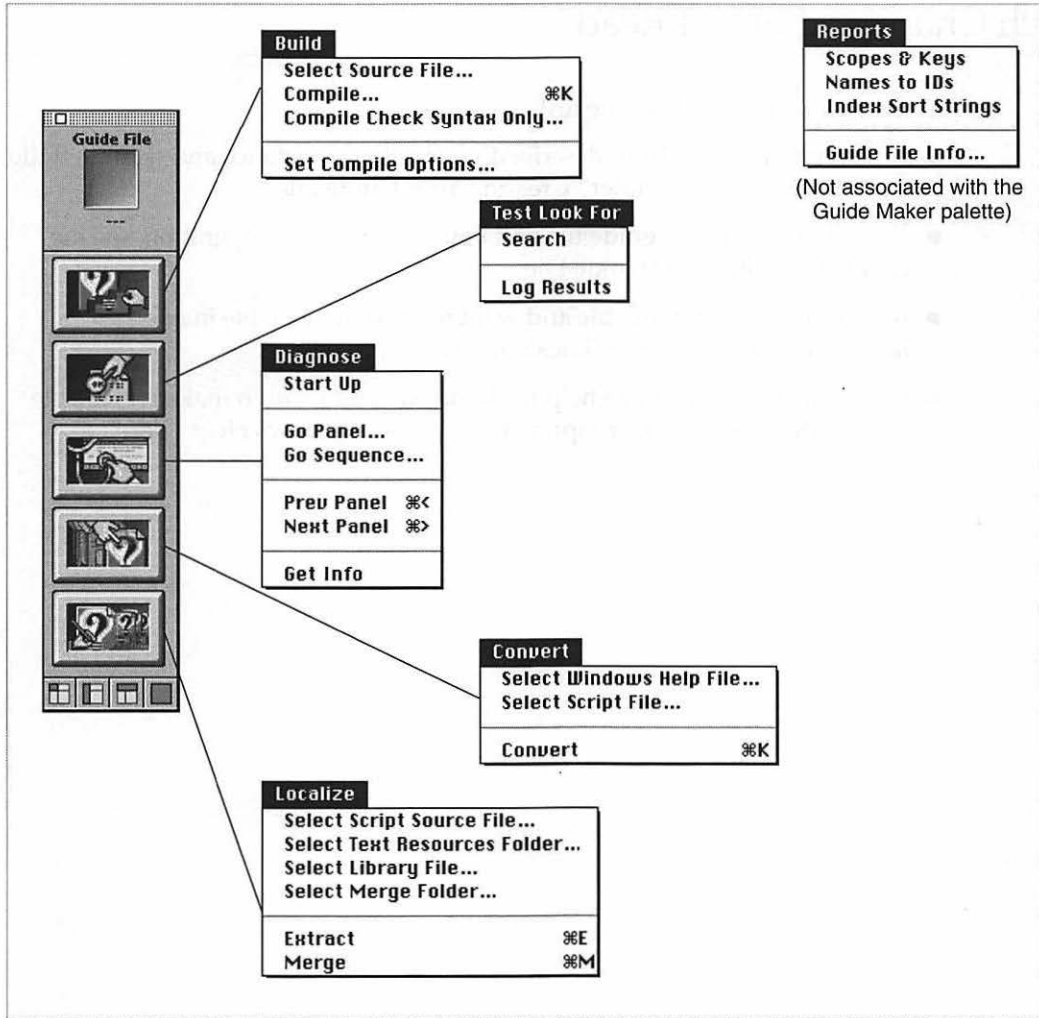
Figure 4-2 Accessing Guide Maker's utilities



Here's how your four tasks correspond to the five Guide Maker buttons:

- To build your guide file, use the Build utility.
- To test your guide file, use the Test Look For or the Diagnose utility (the difference between these two utilities is described in the chapter "Testing Your Guide File").
- To localize your guide file, use the Localize utility.
- To convert a Windows help file to a Guide Script file, use the Convert utility.

When you select a utility, Guide Maker displays a window specific to this utility and adds a corresponding menu to the menu bar. For example, if you choose the Build utility, Guide Maker displays the Build window and adds the Build menu to the menu bar. To accomplish your building tasks, you can use either the commands in the Build menu or the buttons in the Build window. Figure 4-3 illustrates the menus associated with Guide Maker's utilities.

Figure 4-3 Using Guide Maker's menus

Note that the Reports menu, shown to the right in Figure 4-3, is not directly associated with any of the utilities. It is, unlike the other menus, always visible in the menu bar. (The Reports menu is discussed in the chapter “Testing Your Guide File.”)

Which Chapter Should I Read?

You are now ready to read the rest of Part 2.

- If you have designed and described your online assistance and want to build a guide file, see the chapter “Creating Your Guide File.”
- If you have created a guide file and you want to test its operation, see the chapter “Testing Your Guide File.”
- If you have built a guide file and want to make it available in another language, see the chapter “Localizing Your Guide File.”
- If you have created online help for Windows and want to make it available for the Mac OS, read the chapter “Converting Windows Help Files.”

Creating Your Guide File

Contents

Preparing Your Source Files	5-3
Building Your Guide File in Four Steps	5-6
Setting Compile Options	5-8
Checking the Syntax of Your Source Files	5-9
Interpreting the Compile Messages	5-10
Other Utilities	5-11
Importing and Exporting Resources	5-12
Specifying Guide File Information	5-12
Creating a Mixin for Your Guide File	5-13

Once you have designed and scripted your online assistance, you are ready to create your guide file. This chapter tells you how.

It describes how to

- prepare your source files
- build your guide file (that is, compile your source files into a guide file)
- set compile options
- interpret compile messages

This chapter also describes how you can create a mixin for an existing guide file.

To get the most out of this chapter, you should have a basic understanding of Apple Guide, know how to script a Guide Script source file, and be familiar with the Guide Maker application. For more information on these subjects, see Part 1, Part 4, and the first chapter of Part 2, respectively.

Preparing Your Source Files

If your help content contains more than one panel sequence, you have most likely described it in multiple source files. For example, you might have a couple of source files that describe the panels and sequences, and a resource file that contains pictures for your panels. If you have divided your help content into multiple files, you need to merge the content together when you build your guide file.

Note

If your help content is in only one source file, you do not need to read this section. Instead, you can proceed to the next section; it describes how to build your guide file. ♦

To combine the help content together you need to create a file—called a **build file**—that merges together all of your source files. A **build file** is a source file that contains only `<Include>` and `<Resource>` commands. You use the `<Include>` command to specify your source files and the `<Resource>` command to specify a file containing resources for your guide file. (For more information on these commands, see the chapter “Guide Script Command Reference” in Part 4 of this book.)

Creating Your Guide File

For example, if you split your help content into two source files, you need to create a build file that specifies both of these files. Figure 5-1 illustrates a working example of a build file and its accompanying source files. This build file specifies five source files that describe

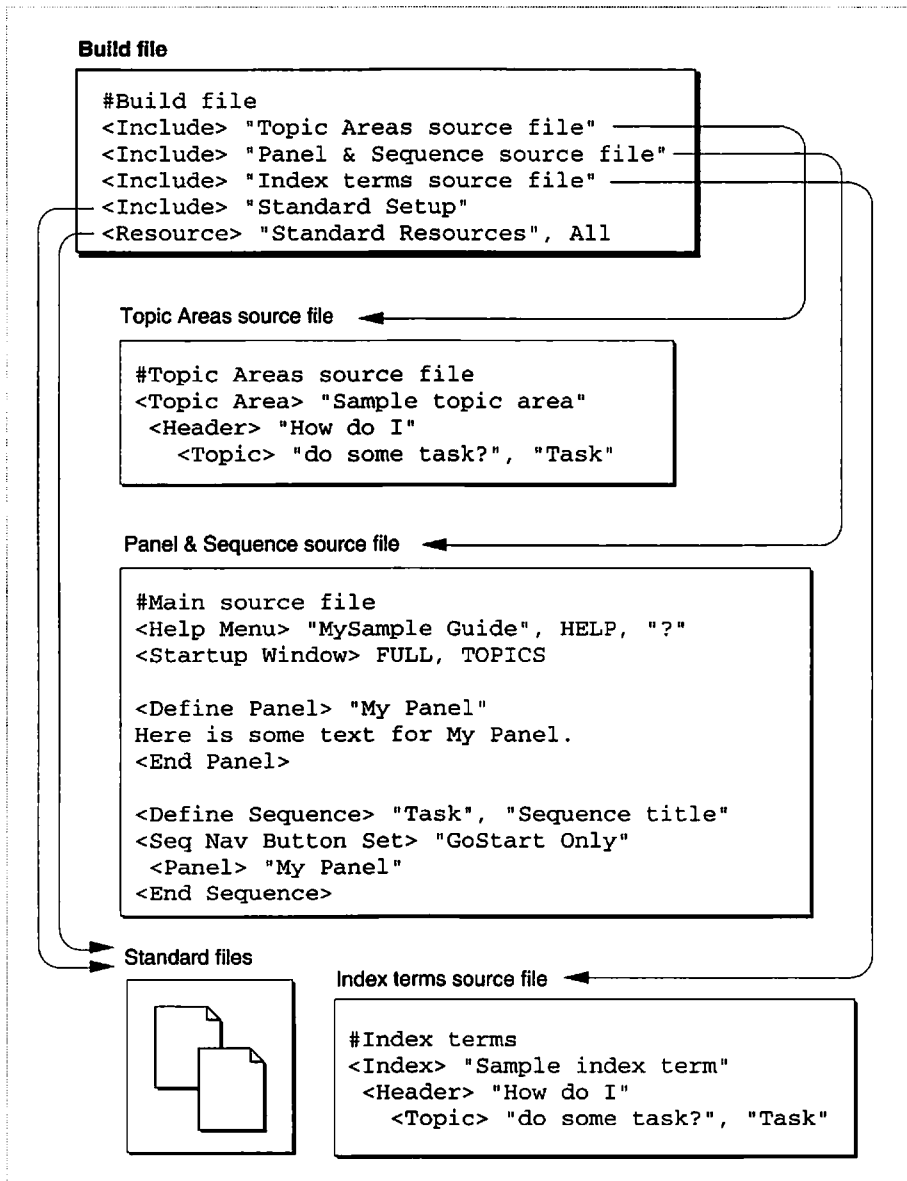
- the topic areas and topics of the guide file
- the panels and sequences for the guide file
- the index terms of the guide file
- the basic setup information for the guide file (the Standard Setup file)
- the resources used by the guide file (the Standard Resources file)

The two standard files, Standard Setup and Standard Resources, are provided with Guide Maker. The Standard Setup file contains the Guide Script commands that give basic information about a guide file, such as prompt set definitions, navigation bar button definitions, and format definitions for placing objects and text in your panels. The Standard Resources file contains the 'PICT' resources that describe the GoStart and Huh? buttons, and a sample application logo. The file also contains external modules for context checks. If you include the Standard Setup and Standard Resources files in your build file, you can automatically use these definitions and graphics in your guide file. For more information on these two files, see Appendix C.

Note

When you build your guide file, you must place any files referenced by <Include> or <Resource> commands in the same folder as your build file. If the files (or an alias to each of the files) are not in the same folder, you will not be able to build your guide file. ♦

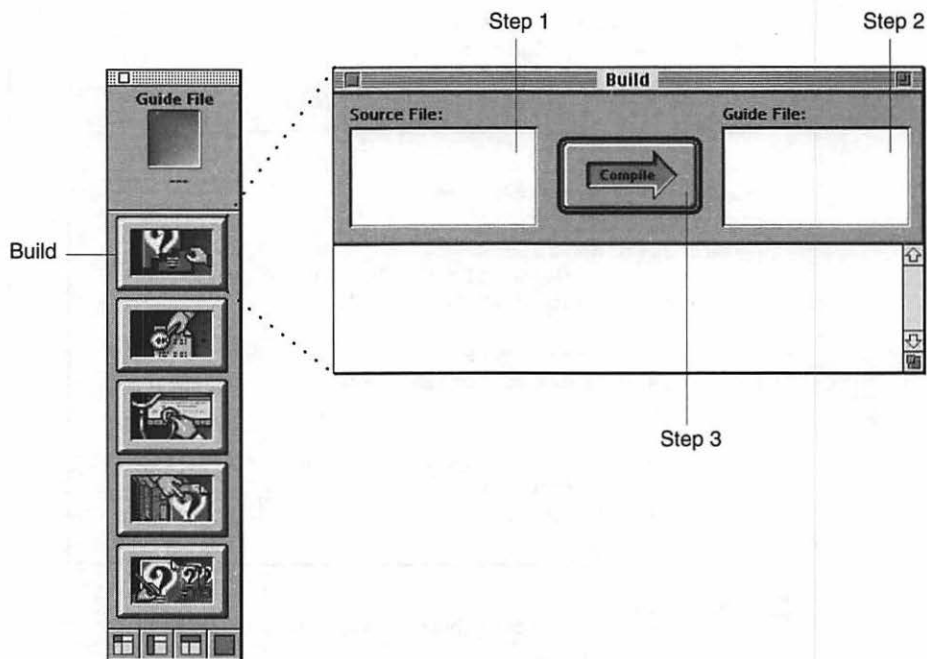
The next section describes how to build your guide file.

Figure 5-1 Creating a build file

Building Your Guide File in Four Steps

To build a guide file is a four-step process using Guide Maker's **Build utility**. Figure 5-2 illustrates the Build window (Guide Maker's building interface). Note that the steps in the illustration refer to the first three steps of the build process.

Figure 5-2 Building your guide file using Guide Maker's Build utility



To build your guide file, follow these four steps:

1. Select the source file to compile.

Click in the Source File area of the Build window (see Figure 5-2); a standard file dialog box appears, requesting that you select the source file to compile. Select your build file. When you compile your build file, Guide Maker merges all of your help content together and writes it to your guide file. If you don't have a build file (that is, if your help content is described in only one source file) then select that source file as the file to compile.

2. Select the guide file.

Click in the Guide File area of the Build window (see Figure 5-2); a dialog box appears, requesting that you create a new guide file or select an existing one.

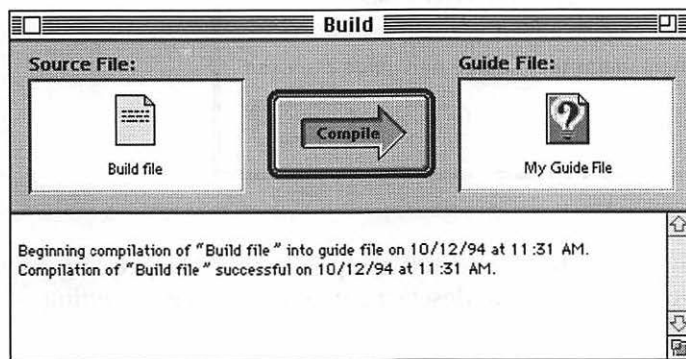
If you have never built a guide file before, or if you want to build another version of your guide file, click the "Create a new guide file" radio button and then name the new guide file. A new guide file is empty, meaning that it does not contain any help content. When you build with a new guide file, you simply add your help content to this previously empty guide file.

If you want to override the help content in an existing guide file, click the "Open an existing guide file" radio button and then select the guide file. When you build with an existing guide file, you override the previous content in the guide file with the content specified in the current source files.

3. Click the Compile arrow.

Guide Maker begins to compile your source files, reporting status and error messages in the status area of the Build window. Figure 5-3 shows the messages reported by Guide Maker after a successful compile.

Figure 5-3 A successfully compiled guide file



Creating Your Guide File

4. Save the guide file.

After you have successfully compiled your source files into a guide file, choose Save from the File menu to save it. (To automate this step, see the next section.)

Congratulations! By following the build steps, you have just created a guide file. To make the created guide file available for your application, place it (or an alias to it) in the same folder as the application.

Setting Compile Options

By specifying compile options, you can adjust the compile process to fit your needs. For example, you can specify when you want Guide Maker to save your guide file—whether you want it to save it immediately after your guide file has finished compiling or when you choose Save from the File menu.

To set compile options, select the Set Compile Options command from the Build menu; Guide Maker displays the compile options dialog box shown in Figure 5-4.

Figure 5-4 Compile options dialog box showing default settings

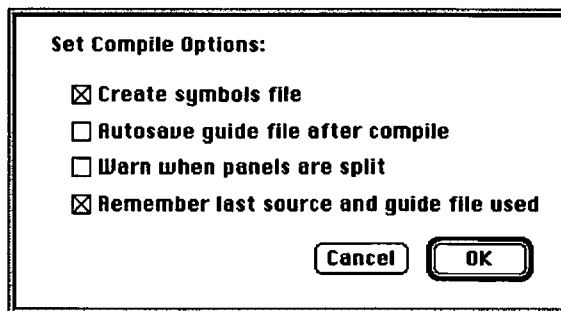


Figure 5-4 illustrates the default compile settings. Depending on your compile needs, you can select and deselect any of these compile settings.

- **Create symbols file.** A symbols file contains symbolic information that you use when you test your guide file. If you don't create a symbols file, you won't be able to generate the test reports—Names to IDs and Guide File Info—when you test your guide file. For information on these reports, see the chapter “Testing Your Guide File.” You should not include the symbols file when you distribute your guide file. Guide Maker automatically generates a symbols file. If you don't want it to, then deselect this compile option.
- **Autosave guide file after compile.** To spare yourself the last step in the compiling process (step 4, saving your guide file), you can, by checking this compile option, have Guide Maker automatically save your guide file.
- **Warn when panels are split.** If the content of one of your panels is larger than the panel itself, Guide Maker splits the panel into several to accommodate it. Guide Maker does not warn you when it does this. If you would like to receive a warning, select this compile option. (For information on how to extend the maximum height of a panel, see the chapter “Guide Script Command Reference.”)
- **Remember last source and guide file used.** Guide Maker keeps track of the last files you used in the compile process. Thus, once you've specified your build file and guide file, you needn't select them every time you open Guide Maker. If you don't want Guide Maker to remember the last files used, then deselect this option.

Checking the Syntax of Your Source Files

When you compile your source files, Guide Maker performs both a syntax check and a full compilation. To perform a syntax check without a full compilation, choose the Compile Check Syntax Only command from the Build menu.

Guide Maker doesn't write information to the guide file or symbols file when performing a syntax check. Thus, a syntax check is noticeably faster than a full compile, especially for large guide files. Use the Compile Check Syntax Only command if you are making small, incremental changes to your source files. You can quickly verify the syntax of your changes without having to perform a full compilation. When you are ready to build your guide file, use the

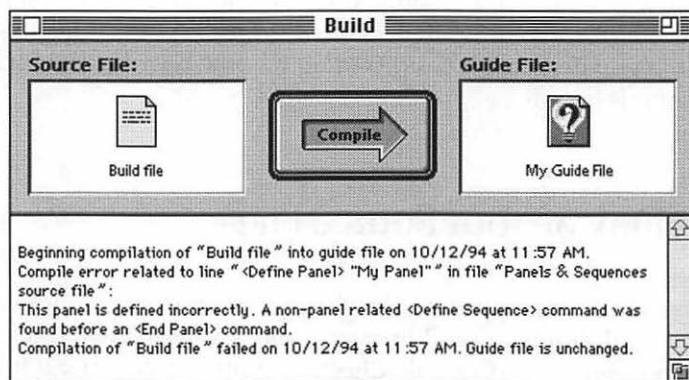
Compile command, as described in “Building Your Guide File in Four Steps” on page 5-6.

Interpreting the Compile Messages

When you compile your source files, you might get messages indicating that the compile process didn't go as you expected. There are two types of compile messages—error messages and warning messages.

If you get an *error message*, it means that Guide Maker was unable to compile your source files and you must correct the error before you complete the compile process. For example, if you accidentally leave out an <End Panel> command in one of your source files, you get an error message. Figure 5-5 illustrates this type of error message.

Figure 5-5 Guide Maker displaying a compile error message

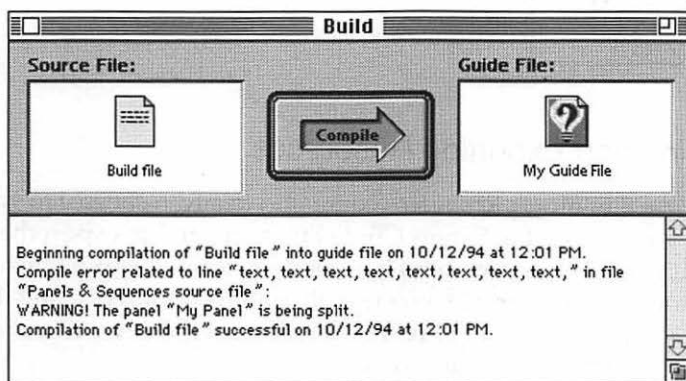


Note

If you get a compile error, your guide file does not change; it is updated only after you successfully compile it and save it. ♦

If you get a *warning message*, it indicates that Guide Maker found a potential problem with your guide file. It is a good idea to investigate the problem, even though it has not caused compilation to fail. For example, if the content of one of your panels is larger than the panel itself and you have selected the “Warn when panels are split” compile option, you get a warning message. Figure 5-6 illustrates this type of warning message.

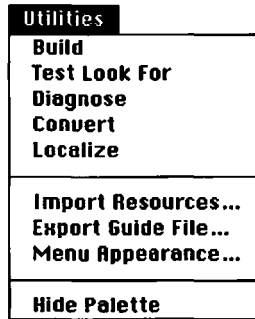
Figure 5-6 Guide Maker displaying a warning message



To keep a record of your error and warning messages, save and print them using the Save Text and Print Text commands of the File menu.

Other Utilities

When you create or open a guide file, three menu items—Import Resources, Export Guide File, and Menu Appearance—become available in the Utilities menu (see Figure 5-7).

Figure 5-7 The Utilities menu

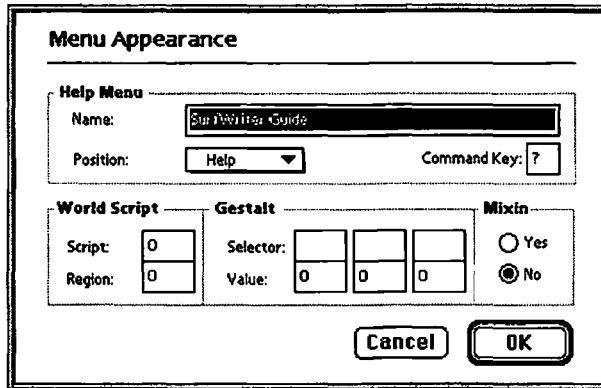
Importing and Exporting Resources

You can use the Import Resources command to add resources to your guide file, and you can use the Export Guide File command to export the resources in your guide file. These two menu items are especially helpful when you debug your guide file. For example, you usually import resources using the <Resource> command, but to reimport resources quickly without compiling again, use the Import Resources command.

Specifying Guide File Information

You can use the Menu Appearance command to modify information about your guide file that you usually set in your source file using Guide Script commands. The Menu Appearance command displays a dialog box (see Figure 5-8) which you can use to

- modify the attributes that determine how your guide file appears in the Help menu
- change the WorldScript, Gestalt, and mixin information in your guide file

Figure 5-8 The Menu Appearance dialog box

Usually you define these settings in your source files using the <Help Menu>, <World Script>, <Gestalt>, and <Mixin> Guide Script commands. If you don't, or if you want to modify the settings you specified in your source files, use the Menu Appearance command. (For information about these commands, see the chapter "Guide Script Command Reference.")

To override information that already exists in your guide file, type in the new information in the Menu Appearance dialog box and click the OK button. To add new information to your guide file, type in the new setting in the Menu Appearance dialog box and click the OK button. To apply these settings to your open guide file, choose Save from the File menu.

If you recompile the guide file, the settings in the Menu Appearance dialog box are updated with the settings specified in your source files. If your source files do not specify settings for the Help menu, WorldScript, Gestalt, or mixin attributes, the settings in the Menu Appearance dialog box remain as previously set.

Creating a Mixin for Your Guide File

If, after distributing a guide file, you need to add to or modify its content, you can create a Mixin guide file (also known as a guide file addition, or *mixin*).

Creating Your Guide File

This section gives information on how to create a Mixin guide file that revises the contents of an existing guide file.

Creating a Mixin guide file is much like creating any guide file: first you describe the help content in source files and then you compile the source files into a Mixin guide file (that is, you build your mixin). To specify that you want Guide Maker to create a Mixin guide file, use the <Mixin> command. You also specify the guide file you wish to *mix* information into using this command. (For information on how to use the <Mixin> command and for descriptions of the commands you can use in your mixin source file to replace, delete, or add content to a main guide file, see the chapter “Guide Script Command Reference.”)

Once you have created your source files, build your Mixin guide file using Guide Maker’s Build utility. For information on how to build a guide file, see the section “Building Your Guide File in Four Steps” beginning on page 5-6. Note that when you build your Mixin guide file, you typically give it the name of your guide file and append the word “Addition”. Figure 5-9 shows an icon for a guide file and one for its accompanying mixin. Note that the icon of a Mixin guide file has a plus (+) symbol.

Figure 5-9 A guide file and its guide file addition, or mixin



MySample Guide



MySample Guide Addition

A Mixin guide file contains additional information that Guide Maker automatically mixes into an existing guide file. You can restrict which mixins mix into a main guide file by using the <Mixin Match> command; for information on this command, see the chapter “Guide Script Command Reference.” A mixin and the guide file it mixes into must be kept in the same folder for the help information to appear in the application. Note that the mixin does not appear in the Help menu; the user sees only the name of the guide file.

Testing Your Guide File

Contents

Testing Your Guide File's Interface	6-3
Obtaining Navigation Information	6-5
Getting Debugging Information	6-6
Testing Your Look For Content	6-8
Generating Reports	6-13
The Scopes and Keys Report	6-13
The Names to IDs Report	6-14
The Index Sort Strings Report	6-15
The Guide File Info Report	6-16
Verifying Coachmarks, Context Checks, and Event Functions	6-18
Testing Coachmarks	6-19
Testing Context Checks	6-19
Testing Event Functions	6-19
Planning for User Testing	6-20

By now you have created your guide file, and you are ready to test it to see if it performs as you expect. You have designed the content of your guide file, described the design in Guide Script source files, and compiled these source files into a guide file. This chapter describes how you can test your guide file to make sure that it works as you expect.

The type of testing you should perform on your guide file includes:

- **interface testing**—checking that all topic areas appear in the access window, that the appropriate topics are associated with each topic area, that each topic brings up its associated sequence of panels, that the user can navigate from panel to panel in the order you intended, that buttons perform the expected action, and that panel content appears as you intended
- **Look For testing**—checking that user search phrases are correctly parsed and matched to the appropriate topics
- **coachmark testing**—verifying that coachmarks mark the item in the interface that you intended to coach
- **context check testing**—verifying that context checks perform as expected
- **event function testing**—verifying that actions that your guide file performs for the user work as expected
- **user testing**—observing users as they use your guide file

This chapter discusses each of these areas of testing. Note however, that it does not provide in-depth information on software testing methodology.

This chapter also includes information on how to use Guide Maker's Reports menu.

Once you test your guide file, if you find that it needs redesign, see the chapter "Designing Your Help Content." After testing your guide file, if you want to localize your guide file, see the chapter "Localizing Your Guide File."

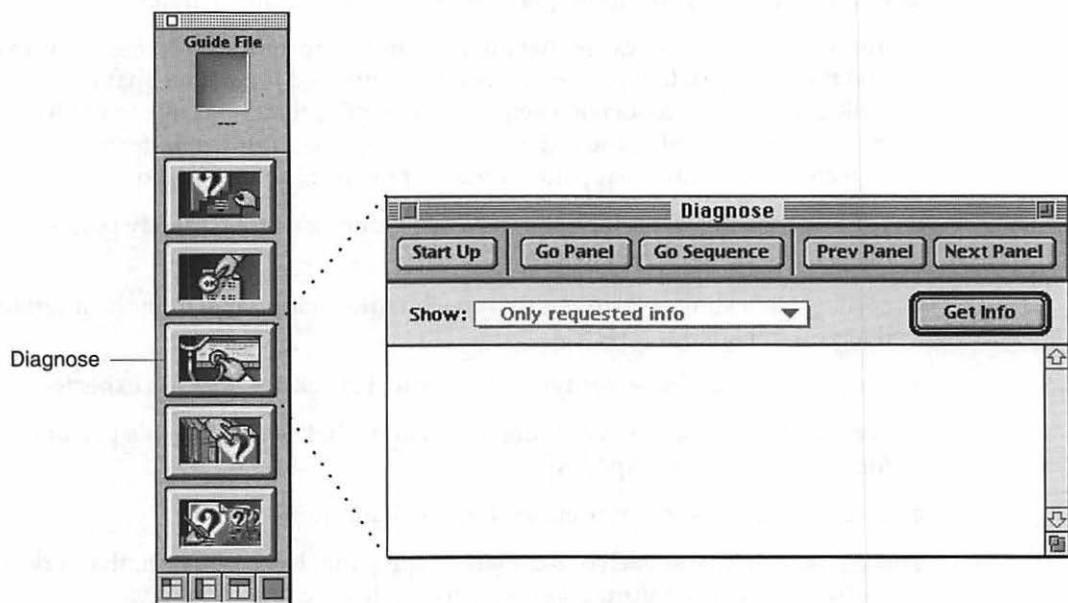
Testing Your Guide File's Interface

By navigating through your guide file, you can test its interface. For example, by navigating through the various panels in your guide file, you can determine whether they appear in the order you intended, whether the art on the panels

Testing Your Guide File

looks OK, and so on. To test your guide file's interface, use Guide Maker's **Diagnose** utility. Figure 6-1 shows the Diagnose window.

Figure 6-1 The Diagnose window



To navigate through your guide file, use the five buttons located on the top of the Diagnose window.

- **Start Up.** If you have just created a new guide file, it's a good idea to examine all of its help content. To do so, start up your guide file and navigate through it, just as if you were the user. To display your guide file's startup window, click the Start Up button. Once your guide file is active (that is, its startup window is showing), you can then navigate through all of its sequences by selecting, one-by-one, all of your topics and index terms.
- **Go Panel.** If you have modified or added content to just one panel in your guide file, you might want to examine that one only (and not the whole sequence it belongs to). To navigate to a specific panel in your guide file, click the Go Panel button. The Go Panel command displays a list of all the

Testing Your Guide File

panels in your guide file; from this list you can select the panel you want to examine. Note that the panels are listed by their names and IDs if you have the guide file's symbol file in the same folder as the guide file; otherwise, only the IDs are shown. You can obtain a mapping of panel IDs to panel names by generating a Names to IDs report. For more information on the Names to IDs report, see the section "Generating Reports" on page 6-13.

Note that when you view a panel using the Go Panel command, Guide Maker does not display the panel's sequence display title, its prompt, or its navigation bar buttons; this is because the panel might belong to more than one sequence.

- **Go Sequence.** If you have modified or added content to several panels in a sequence, it is a good idea to examine the entire sequence. To invoke the beginning of a specific sequence, click the Go Sequence button. The Go Sequence command displays a list of all the sequences in your guide file; from this list you can select the sequence you want to examine. Note that the sequences, just like the panels, are listed by their names and IDs if you have the guide file's symbol file in the same folder as the guide file; otherwise, only the IDs are shown. You can obtain a mapping of sequence IDs to sequence names by generating a Names to IDs report. For more information on the Names to IDs report, see the section "Generating Reports" on page 6-13.
- **Prev Panel.** If you are examining a panel and would like to look at the panel that comes before it without invoking any context checks, click the Prev Panel button.
- **Next Panel.** If you are examining a panel and would like to look at the one that comes after it without invoking any context checks, click the Next Panel button.

Obtaining Navigation Information

When you navigate through your guide file, you might want to get a quick update on exactly where in the guide file you are. For example, you might want to know the name and ID of the panel you are currently looking at, so that you can revisit it later in your testing phase. To get this type of information, click the Get Info button in the Diagnose window.

Testing Your Guide File

The Get Info command displays, in the status area of the Diagnose window, the exact position of your guide file. Here's an example of the information you can obtain by clicking the Get Info button:

Sequence "MySequence" (#2001) Panel "MyPanel" (2004) (2 of 3)

From this example you can tell that the current panel is MyPanel. Its panel ID is 2004, it is the second of three panels, and it belongs to the sequence MySequence (which has the sequence ID 2001). The next section describes how to get additional debugging information about your guide file.

Getting Debugging Information

To obtain even more information about your guide file as you navigate through it, install the Apple Guide Debug extension. This extension gives you the option of using the "All messages" command in the Show pop-up menu of the Diagnose window.

IMPORTANT

To install the Apple Guide Debug extension, first remove the Apple Guide extension from your Extensions folder; then install the Apple Guide Debug extension and reboot. Do not install both extensions. ▲

If the Apple Guide Debug extension is installed and you select the "All messages" option from the Show pop-up menu, Guide Maker displays running in-depth information about your guide file as you navigate through it. Guide Maker displays this information in the status area of the Diagnose window. This feature is useful for debugging context checks or event functions that use AppleScript. For example, when "All messages" is selected, Guide Maker provides information about the number of external modules in the guide file, reports when a context check is invoked for a panel, and reports any errors related to context checks, Apple events, or AppleScript processing.

Note

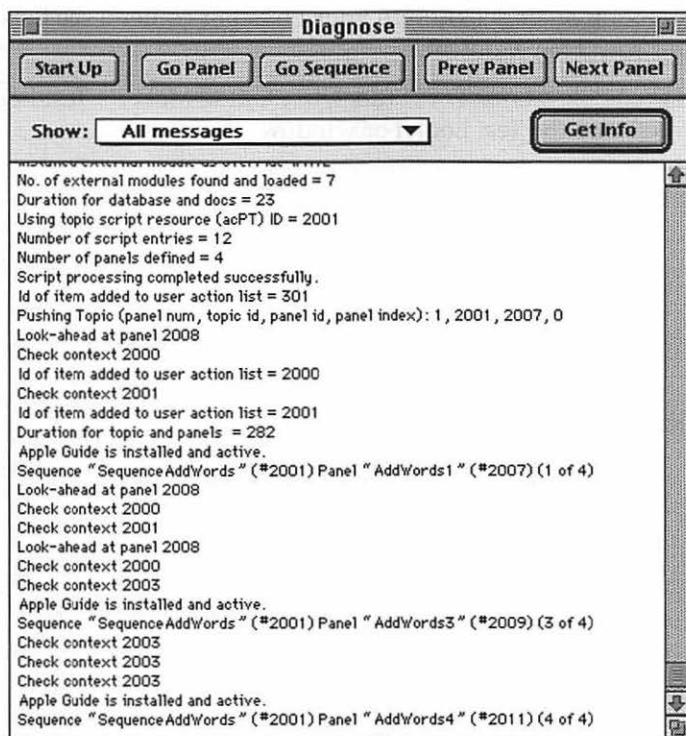
You can also click the Get Info button at any time, to record the current sequence ID and panel ID. ♦

Figure 6-2 shows messages displayed by Guide Maker when the "All messages" option is selected. This figure shows the navigation through four

Testing Your Guide File

panels of a sequence of a specific guide file, beginning with the first panel (panel ID 2007). The second panel (panel ID 2008) and third panel (panel ID 2009) both have two context checks associated with them, and the fourth panel (panel ID 2011) has one context check associated with it. This display shows how Apple Guide looks ahead to the next panel, invoking any context checks for that panel, to determine whether the next panel should be displayed. For example, because the second panel contains a <Skip If> command that evaluates to `true`, the second panel is skipped. The context checks for the third and fourth panels evaluate to `true`; so these panels are displayed as the user navigates to them.

Figure 6-2 The “All messages” debugging option



Testing Your Guide File

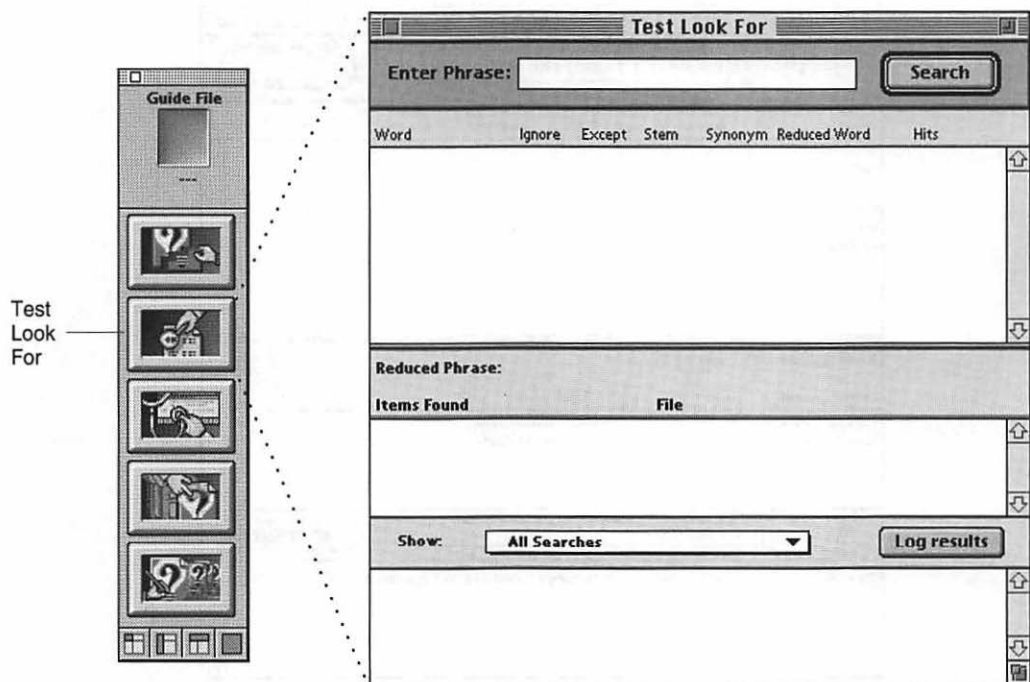
You can keep a record of the information reported in the status area by using the Save Text and Print Text commands of the File menu.

Testing Your Look For Content

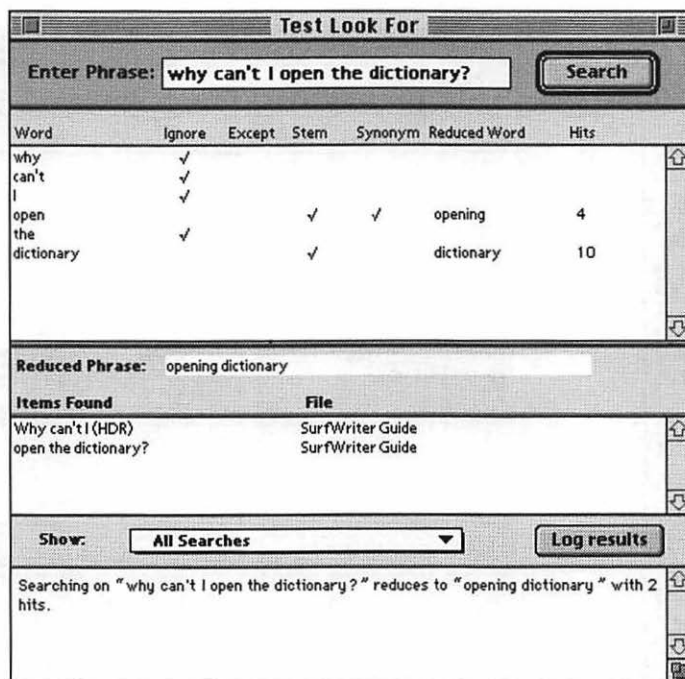
When a user enters a search phrase in the Look For window and clicks Search, Apple Guide parses this phrase and tries to match it with an index term in the guide file. To enhance Apple Guide's searching facility, in addition to index terms you can provide Look For content that Apple Guide uses as it parses the phrase. This section describes how you can test whether a search phrase matches up with the appropriate index term in your guide file.

For information on how to design your Look For content, see the chapter "Planning Your Help Content."

To test a guide file's Look For content, use Guide Maker's Test Look For utility. Figure 6-3 shows the Test Look For window.

Figure 6-3 The Test Look For window

Enter the phrase to search for in the Enter Phrase text box and then either click the Search button or press the Enter key. Figure 6-4 shows the results of entering a phrase in the Test Look For window.

Figure 6-4 A parsed phrase in the Test Look For window


Word	Ignore	Except	Stem	Synonym	Reduced Word	Hits
why	✓					
can't	✓					
I	✓					
open			✓	✓	opening	4
the	✓					
dictionary			✓		dictionary	10

Reduced Phrase: opening dictionary

Items Found	File
Why can't I (HDR)	SurfWriter Guide
open the dictionary?	SurfWriter Guide

Show: All Searches **Log results**

Searching on "why can't I open the dictionary?" reduces to "opening dictionary" with 2 hits.

As you can see from Figure 6-4, Guide Maker reports the following for each word:

- Whether it is on the ignore list. (If so, it puts a check in the Ignore column.)
- Whether it is on the exception list. (If so, it puts a check in the Except column.)
- Whether it is eligible for stemming. (If so, it puts a check in the Stem column.) Note that a check in the Stem column does not necessarily mean the word was stemmed.

Guide Maker then checks whether the phrase as parsed so far is a synonym, and if so, it puts a check in the Synonym column and also puts the associated word or phrase in the Reduced Word column. Using the example shown in Figure 6-4, the phrase as parsed at this point is "open dictionary". This phrase is not in the synonym list. (If the phrase is in the synonym list, Guide Maker

Testing Your Guide File

checks the index for the parsed phrase and, if it finds a matching index term, displays the results in the Items Found column and displays “n/a” in the Hits column [see Figure 6-5].)

If Guide Maker does not find the initial parsed phrase in the synonym list, it reports the following for each word:

- Whether it is a synonym for another word or phrase. (If so, it puts a check in the Synonym column.)
- The resulting word after it was parsed. (It displays it in the Reduced Word column.)

Guide Maker displays the resulting phrase in the Reduced Phrase area, checks the index for this phrase (“opening dictionary” in this example), and displays the results in the Items Found column. If it doesn’t find the reduced phrase in the index, it also reports the following for each word:

- Whether the index contains a matching index term for the word. (If so, it displays the number of headers and topics associated with the matched index term in the Hits column.)

Guide Maker then displays in the Items Found column the intersection of the words in the reduced phrase. In the example shown in Figure 6-4, Guide Maker displays the intersection of “opening” and “dictionary”.

Figure 6-5 shows another example of how Guide Maker parses a search phrase. As shown in this example, “site dictionary” is a synonym for “dictionary”. “Dictionary” is an index term, and thus Guide Maker displays in the Items Found column all the topics associated with the index term. Guide Maker also displays “n/a” in the Hits column when it finds a synonym for an entire phrase.

Figure 6-5 Results of a search

The screenshot shows a window titled "Test Look For". At the top, there is a text input field labeled "Enter Phrase:" containing the text "site dictionary", followed by a "Search" button. Below this is a table with the following columns: Word, Ignore, Except, Stem, Synonym, Reduced Word, and Hits. The first row of data shows "site dictionary" in the Word column, a checkmark in the Synonym column, "dictionary" in the Reduced Word column, and "n/a" in the Hits column. Below the table is a section labeled "Reduced Phrase:" with the text "dictionary". Underneath is a table with two columns: "Items Found" and "File". The "Items Found" column lists various phrases like "How do I (HDR)", "add a word to the dictionary?", and "look up a word in the dictionary?". The "File" column lists "SurfWriter Guide" for each item. At the bottom of the window, there is a "Show:" dropdown menu set to "All Searches" and a "Log results" button. Below these is a scrollable text area that says "Searching on 'site dictionary' reduces to 'dictionary' with 10 hits".

Word	Ignore	Except	Stem	Synonym	Reduced Word	Hits
site dictionary				✓	dictionary	n/a

Reduced Phrase: dictionary

Items Found	File
How do I (HDR)	SurfWriter Guide
add a word to the dictionary?	SurfWriter Guide
look up a word in the dictionary?	SurfWriter Guide
create a custom dictionary?	SurfWriter Guide
add or remove a dictionary?	SurfWriter Guide
Why can't I (HDR)	SurfWriter Guide
open the dictionary?	SurfWriter Guide
Definitions (HDR)	SurfWriter Guide
custom dictionary	SurfWriter Guide
standard dictionary	SurfWriter Guide

Show: All Searches Log results

Searching on "site dictionary" reduces to "dictionary" with 10 hits

Guide Maker gives a summary of the search results in the Log Results area (the scrollable area near the bottom of the Test Look For window). If you select All Searches from the Show pop-up menu, Guide Maker gives a summary of the current search each time you perform a search. If you want Guide Maker to report on only selected searches, choose "Only requested searches" from the Show pop-up menu. If you choose this item, then you must click the "Log results" button when you want Guide Maker to report the summary results of a search.

As you develop and test your Look For content, if you get unexpected results, you might find it helpful to create a guide file with only a single index term in it. You can then use this guide file to determine how a word stems by default. By comparing this with the search results from the Look For content of your guide file, you may be able to more quickly troubleshoot how the search phrase is parsing.

Generating Reports

By using Guide Maker's Reports menu, you can get a quick, in-depth look at your guide file. You can

- get a list of all topics and their associated sequence IDs (Scopes and Keys report)
- get a list of each panel's or sequence's name and ID (Names to IDs report)
- verify that index terms are correctly displayed (Index Sort Strings report)
- get a printout of your guide file (Guide File Info report)

The Scopes and Keys, Names to IDs, and Index Sort Strings reports appear in a new window. You can save or print this information using the Save Text or Print Text commands from the File menu. The Guide File Info report is written to a file that you name before generating the report.

Note

To create a Names to IDs report or a Guide File Info report, the guide file and its symbol file must be located in the same folder. ♦

The Scopes and Keys Report

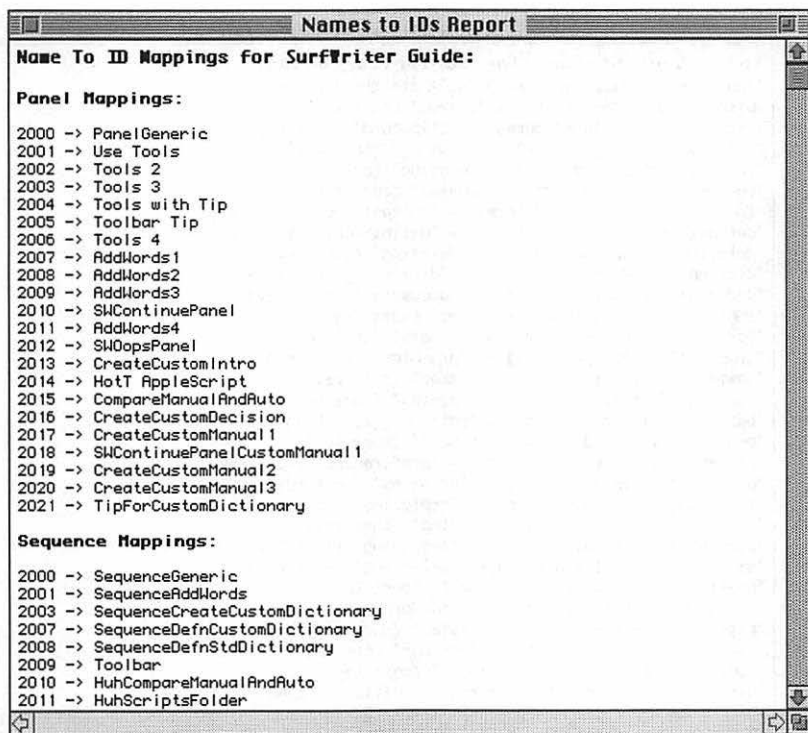
A Scopes and Keys report contains a complete list of the headers and topics for each topic area and each index term in a guide file and gives information about each associated sequence. A Scopes and Keys report also lists the words specified by the Guide Script commands <Exception>, <Ignore>, and <Synonym>. Figure 6-6 shows excerpts from a Scopes and Keys report for SurfWriter Guide. This excerpt shows information about the topic areas "Copying & Pasting", "Using the Dictionary", and "Using the Toolbar". It also shows information about the index terms "Apple menu" and "dictionary". Note that the number following the text "PresID:" is the sequence ID associated with the topic.

Figure 6-6 A Scopes and Keys report

Scopes and Keys Report	
Dump of Scopes & Keys for SurfWriter Guide:	
Copying & Pasting	Scope: T(?0 Key: HABA (TA) in SurfWriter Guide
How do I	Key: ABA (HDR) in SurfWriter Guide
placeholder for topic?	PresID: 2000 Key: ABB (TPC) in SurfWriter Guide
another placeholder for topic?	PresID: 2000 Key: ABC (TPC) in SurfWriter Guide
Using the Dictionary	Scope: T(?1 Key: HABB (TA) in SurfWriter Guide
How do I	Key: ABA (HDR) in SurfWriter Guide
add a word to the dictionary?	PresID: 2001 Key: ABB (TPC) in SurfWriter Guide
look up a word in the dictionary?	PresID: 2000 Key: ABC (TPC) in SurfWriter Guide
create a custom dictionary?	PresID: 2003 Key: ABD (TPC) in SurfWriter Guide
add or remove a dictionary?	PresID: 2000 Key: ABE (TPC) in SurfWriter Guide
Why can't I	Key: ABF (HDR) in SurfWriter Guide
open the dictionary?	PresID: 2000 Key: ABG (TPC) in SurfWriter Guide
Definitions	Key: ABH (HDR) in SurfWriter Guide
custom dictionary	PresID: 2007 Key: ABI (TPC) in SurfWriter Guide
standard dictionary	PresID: 2008 Key: ABJ (TPC) in SurfWriter Guide
Using the Toolbar	Scope: T(?9 Key: HABJ (TA) in SurfWriter Guide
How do I	Key: ABA (HDR) in SurfWriter Guide
use the tools in the toolbar?	PresID: 2009 Key: ABB (TPC) in SurfWriter Guide
Apple menu	Scope: I(?0 Key: Apple menu (Index) in SurfWriter Guide
How do I	Key: ABA (HDR) in SurfWriter Guide
placeholder for topic?	PresID: 2000 Key: ABB (TPC) in SurfWriter Guide
dictionary	Scope: I(?0 Key: dictionary (Index) in SurfWriter Guide
How do I	Key: ABA (HDR) in SurfWriter Guide
add a word to the dictionary?	PresID: 2001 Key: ABB (TPC) in SurfWriter Guide
look up a word in the dictionary?	PresID: 2000 Key: ABC (TPC) in SurfWriter Guide
create a custom dictionary?	PresID: 2003 Key: ABD (TPC) in SurfWriter Guide
add or remove a dictionary?	PresID: 2000 Key: ABE (TPC) in SurfWriter Guide
Why can't I	Key: ABF (HDR) in SurfWriter Guide
open the dictionary?	PresID: 2000 Key: ABG (TPC) in SurfWriter Guide

The Names to IDs Report

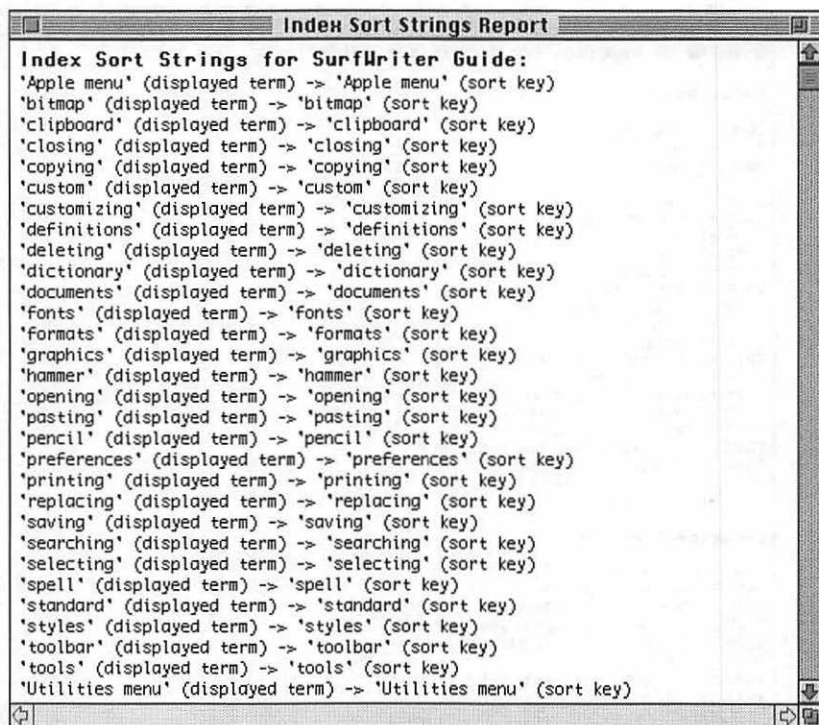
A Names to IDs report contains mappings of all panel IDs to their panel names, all sequence IDs to their sequence names, and all text block IDs to their text block names. Figure 6-7 shows a sample of a generated Names to IDs report.

Figure 6-7 A Names to IDs report

The Index Sort Strings Report

An Index Sort Strings report contains a list of all the index terms in a guide file and each index term's sort key. Figure 6-8 shows part of a generated Index Sort Strings report.

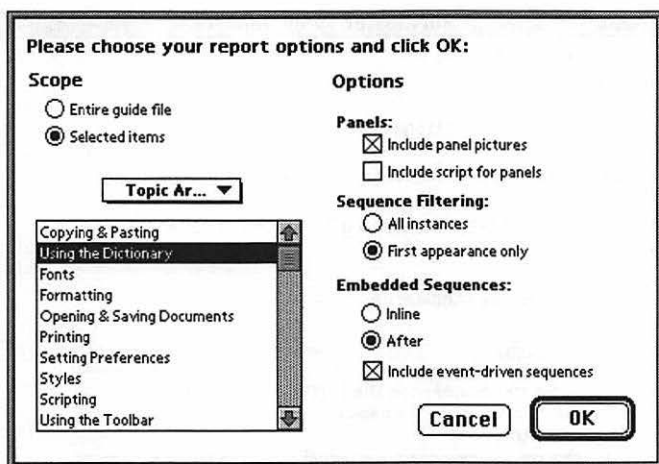
You can specify in your source file how you want Apple Guide to sort your index terms. For information about sorting index terms, see the chapter "Guide Script Command Reference," in Part 4.

Figure 6-8 An Index Sort Strings report

The Guide File Info Report

You can generate a complete report on your guide file by choosing the Guide File Info command from the Reports menu. The generated report includes sequence definitions, panel definitions, and pictures of the panels as they will appear onscreen.

After you choose the Guide File Info command, Guide Maker displays a dialog box asking you to specify your main source file (select your build file if you have multiple source files). It then displays another dialog box, asking you to choose a place to save the file and to choose a file format for the file. Next, it displays the Options dialog box. In it, you can specify the information you want your guide file report to contain. Figure 6-9 shows the Options dialog box.

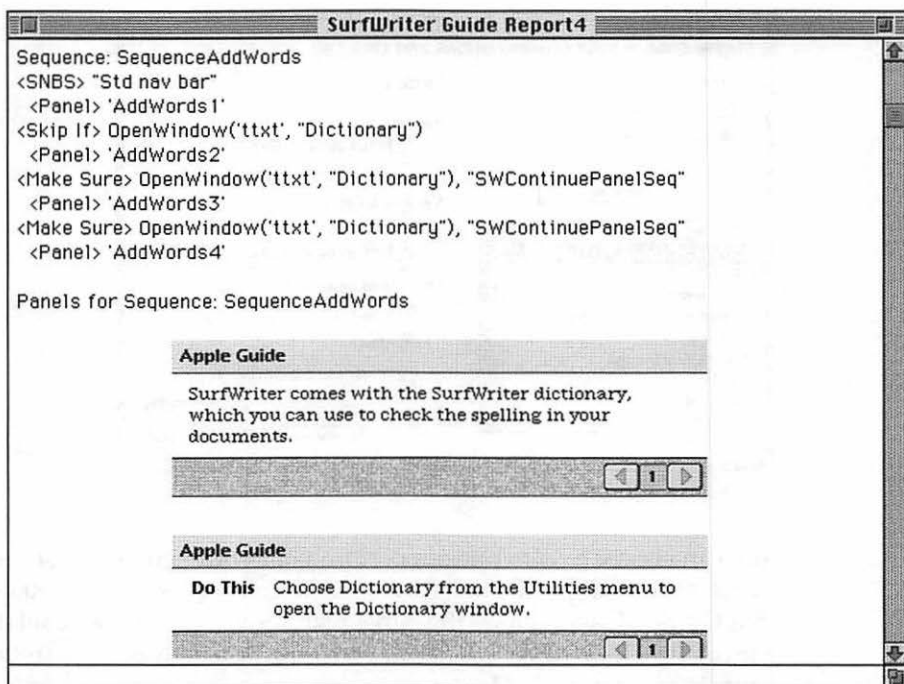
Figure 6-9 The Options dialog box

You can choose to generate a report on your entire guide file or only selected items, such as a specific topic area, index term, or sequence. You can request that Guide Maker include the panel pictures as well as the panel definitions in the report. Generating a report like this can be a convenient way to review the content of your guide file.

Note

To create a Guide File Info report, you must have XTND translators installed on your system. ♦

Figure 6-10 illustrates a sample guide file report, saved in TeachText format. As shown in this report, a sequence definition is followed by pictures of the panels in the sequence.

Figure 6-10 A Guide file Info report

Verifying Coachmarks, Context Checks, and Event Functions

To make sure your guide file works as expected, you need to verify that coachmarks, context checks, and event functions perform as you intended under a variety of conditions.

When testing coachmarks, context checks, and event functions, have several applications open at once and switch between them often to verify that your guide file works as expected. Your user might perform steps out of order, and

you need to allow for this possibility when you define your context checks for each panel.

Testing Coachmarks

If your guide file uses coachmarks, you need to verify, for each panel that specifies a coachmark, that the coachmark marks the desired item.

To begin, verify that the coachmark is directed to the correct application. For example, a menu coach designed to coachmark the File menu of your application should not coachmark the File menu of any other application.

Next, verify that the coachmark marks the desired item. For example, an item coach for an editable text item in a Global Changes dialog box should coachmark only that item. Verify that the coachmark marks the desired item when the dialog box is active, and verify that the coachmark isn't drawn when another window or dialog box is active.

Testing Context Checks

If your guide file uses context checks, you need to verify that the context checks work in any given condition of the user's environment. For example, if you define a context check that determines whether the user's dictionary is open, you need to test that the correct result is returned (and thus the appropriate panel displayed) when the dictionary is open and when it is closed.

Testing Event Functions

If your guide file uses event functions to perform an action for the user, you need to verify that the expected action occurs. For example, an event function that opens a dictionary for the user should both open the dictionary and make it the active document. By testing your event functions, you can also determine whether you have made an assumption that might not be valid given a particular user's environment.

Planning for User Testing

As with any product, you should plan for and conduct user testing of your guide file. Users can give valuable feedback as to the usefulness of your guide file. For example, you can ask users the following:

- Are the topic areas relevant and complete?
- Are the topics you are looking for available from the access window?
- Is the level of instruction appropriate?
- Are instructions clear and easy to follow?
- Is the use of navigation buttons and content area buttons clear and consistent?
- Did the guide file help you perform a task? If not, why not?
- Is the index complete?
- Do searches give the expected result?
- Are any topic areas or topics not covered?

You should plan to do user testing early enough in your design process so that you can incorporate feedback. For additional information on conducting user testing, see the *Macintosh Human Interface Guidelines*.

Localizing Your Guide File

Contents

The Localizing Process	7-3
Translating Text Strings	7-6

Localizing Your Guide File

By now you have created and tested your guide file, and are ready to create localized versions of it. This chapter describes how you can localize your guide file for other regions.

It tells you how to

- extract language-specific text strings from your source files
- translate the text strings using various tools
- merge the translated text strings back into your source files
- localize any other elements of your guide file, such as pictures, as necessary

Once you have localized your source files, you can use these files to build a new, localized version of your guide file.

For information on designing your guide file so that it can be more easily localized, see the chapter “Authoring Tips and Suggestions” in Part 1. For additional design and localization issues, see the *Macintosh Human Interface Guidelines*.

The Localizing Process

You need to localize all elements of your guide file that are language-specific, such as text strings and pictures. These include, for example, the name of your guide file (as it appears in the Help menu), topic areas and topics, titles of panels, text in panels, objects to coach (for example, the names of folders, menus, and menu items), button labels, index terms, and Look For content. Fortunately, Guide Maker provides the **Localize** utility to help you accomplish these tasks. Figure 7-1 shows the Localize window.

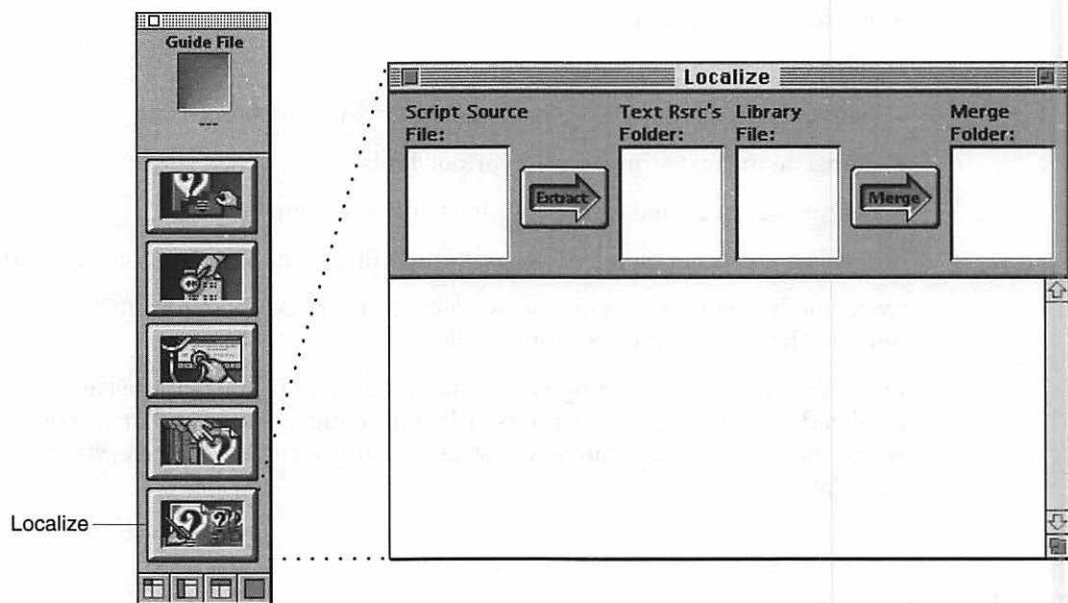
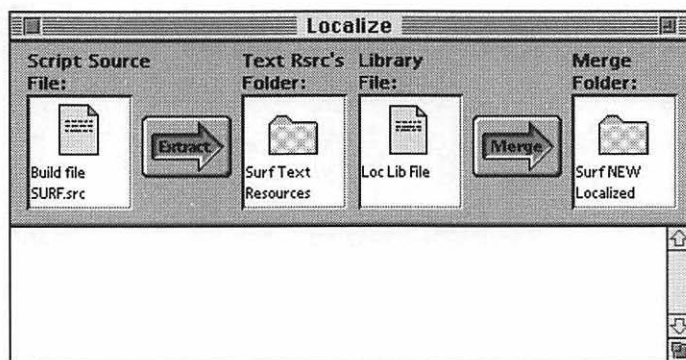
Figure 7-1 The Localize window

Figure 7-2 shows the Localize window after specifying everything you need to.

Figure 7-2 The Localize window with files and folders specified

To localize your guide file, open the Localize window and follow these 10 steps:

1. Select the source file to extract text strings from.

Click in the Script Source File area of the Localize window. Guide Maker displays a dialog box from which you can select your source file. You can select a single source file, or you can select your build file to extract text strings from all source files at once.

2. Select the folder to hold the files containing the extracted text strings.

Click in the Text Rsrc's Folder area. Guide Maker displays a dialog box from which you can select a folder to hold the extracted text resources. Guide Maker creates a resource file for each source file you localize; it places all of the resource files in the text resources folder that you specify.

3. Create a localization library file.

Click in the Library File area of the Localize window. Guide Maker displays a dialog box, in which you indicate whether you want to create a new localization library file or use an existing one. After you choose, Guide Maker prompts you to either name the new library file or select an existing library file.

The localization library file contains information about the position of the extracted text strings in the source files. When you merge the localization library file and the localized text strings, Guide Maker uses the information in the library file to make sure that the text strings are placed into the proper place in the localized script source file.

4. Extract the text strings.

Click the Extract arrow in the Localize window. Guide Maker begins extracting text strings from all source files listed in the build file. It creates a file for each source file, appending `.RSRC` to each of these filenames. It places these files in the text resources folder that you specified in step 2.

5. Translate the extracted text strings.

Use a resource editor such as ResEdit or AppleGlott to translate the extracted text strings in the `.RSRC` files.

Guide Maker stores the extracted text strings as resources of type `'TEXT'` with resource names that give information about the text string. By looking at a resource name, you can determine the Guide Script command associated with the text string, and from that information you can induce the structure of the text string. See the section "Translating Text Strings" for additional information on translating these text strings.

Localizing Your Guide File

6. Select the folder to hold the localized source files.

After you localize the extracted text strings, you can merge the translated text back into your source files. To do this, click the Merge Folder area of the Localize window. Guide Maker displays a dialog box from which you can select a folder to hold the new localized source files.

7. Merge the translated strings and localization library file.

Click the Merge arrow in the Localize window. Guide Maker begins merging the text strings from the .RSRC files back into the source files. It places the new localized source files in the folder you selected in step 6.

8. Localize any 'PICT' resources

For example, if your application logo contains text, you should localize the text using a graphics application.

9. Make any additional localization changes to your source files as needed.

For example, you might need to adjust formats or localize a QuickTime movie.

10. Build a guide file with the localized source files.

Build a new guide file (using the process described in the chapter "Creating Your Guide File"), only specify the build file that Guide Maker placed in the Merge folder. You must also copy any auxiliary files used by your source file (such as 'PICT' resources or scripts) into the Merge folder before compiling.

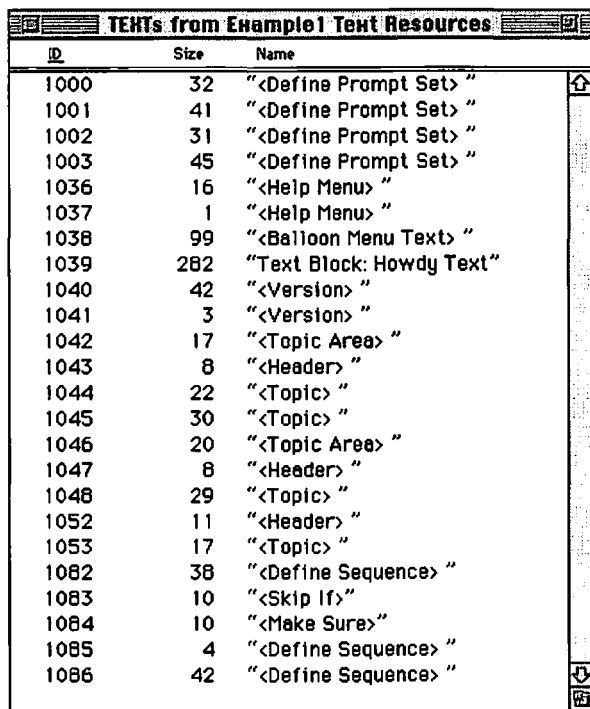
Congratulations, your guide file is now localized! After creating your localized guide file, you should test it, as described in the chapter "Testing Your Guide File."

Translating Text Strings

When you extract text from source files, Guide Maker stores each text string as a resource of type 'TEXT' and gives it a descriptive resource name (so that you can determine the origin of the text string) and a unique resource ID. For example, a text string that specifies an index term has the resource name <Index> and has a unique resource ID to distinguish it from other resources with the same resource name. Table 7-1 beginning on page 7-10 lists all of the possible resource names and gives a description of the text string contents you can expect to be associated with it.

Figure 7-3 and Figure 7-4 show examples (from the SurfWriter Guide source files) of various text resources that Guide Maker extracts. Figure 7-3 shows the ResEdit window of text resources associated with setup information (such as prompt sets and Help menu information), topic areas, and sequences.

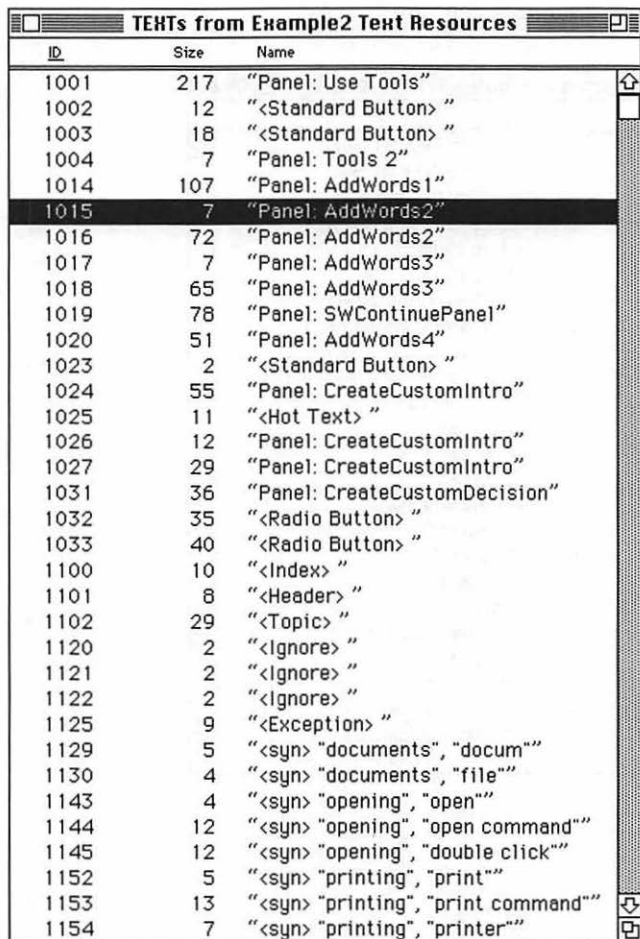
Figure 7-3 Examples of text resources



ID	Size	Name
1000	32	"<Define Prompt Set> "
1001	41	"<Define Prompt Set> "
1002	31	"<Define Prompt Set> "
1003	45	"<Define Prompt Set> "
1036	16	"<Help Menu> "
1037	1	"<Help Menu> "
1038	99	"<Balloon Menu Text> "
1039	282	"Text Block: Howdy Text"
1040	42	"<Version> "
1041	3	"<Version> "
1042	17	"<Topic Area> "
1043	8	"<Header> "
1044	22	"<Topic> "
1045	30	"<Topic> "
1046	20	"<Topic Area> "
1047	8	"<Header> "
1048	29	"<Topic> "
1052	11	"<Header> "
1053	17	"<Topic> "
1082	38	"<Define Sequence> "
1083	10	"<Skip If> "
1084	10	"<Make Sure> "
1085	4	"<Define Sequence> "
1086	42	"<Define Sequence> "

Figure 7-4 shows the ResEdit window of text resources associated with panels, index terms, and Look For content.

Figure 7-4 Text resources for panels, index terms, and Look For content



ID	Size	Name
1001	217	"Panel: Use Tools"
1002	12	"<Standard Button> "
1003	18	"<Standard Button> "
1004	7	"Panel: Tools 2"
1014	107	"Panel: AddWords1"
1015	7	"Panel: AddWords2"
1016	72	"Panel: AddWords2"
1017	7	"Panel: AddWords3"
1018	65	"Panel: AddWords3"
1019	78	"Panel: SWContinuePanel"
1020	51	"Panel: AddWords4"
1023	2	"<Standard Button> "
1024	55	"Panel: CreateCustomIntro"
1025	11	"<Hot Text> "
1026	12	"Panel: CreateCustomIntro"
1027	29	"Panel: CreateCustomIntro"
1031	36	"Panel: CreateCustomDecision"
1032	35	"<Radio Button> "
1033	40	"<Radio Button> "
1100	10	"<Index> "
1101	8	"<Header> "
1102	29	"<Topic> "
1120	2	"<Ignore> "
1121	2	"<Ignore> "
1122	2	"<Ignore> "
1125	9	"<Exception> "
1129	5	"<syn> "documents", "docum""
1130	4	"<syn> "documents", "file""
1143	4	"<syn> "opening", "open""
1144	12	"<syn> "opening", "open command""
1145	12	"<syn> "opening", "double click""
1152	5	"<syn> "printing", "print""
1153	13	"<syn> "printing", "print command""
1154	7	"<syn> "printing", "printer""

To begin translating text, follow these steps.

1. Open one of your .RSRC files using a resource editor (such as ResEdit).

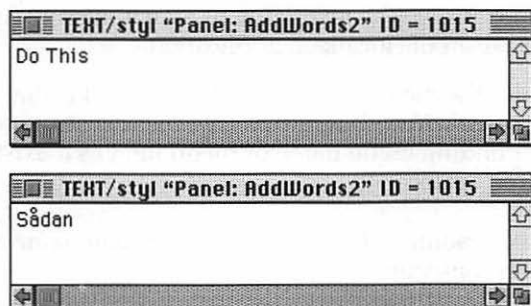
ResEdit displays the resource ID, size, and resource name of each text string in a window (see Figure 7-4).

2. Select the text string to translate.

ResEdit opens a window containing the text that needs translation. Translate the text and then close the window. Repeat this process for all text strings.

Figure 7-5 shows a ResEdit window for a text resource as it appears before and after the text is translated. This example shows the text string of the text resource with resource ID 1015 and resource name "Panel: AddWords2". The string represents panel text, in this case, the tag "Do This", which is translated to "Sådan".

Figure 7-5 Translating a text string



3. Save the .RSRC file containing the translated text strings.

Repeat steps 1 through 3 for all of your .RSRC files.

Table 7-1 lists the typical 'TEXT' resource names (in alphabetical order) and gives a description of the associated text strings.

Table 7-1 The 'TEXT' resource names and the associated text strings

Text resource name	Description of the associated text string
<Balloon Menu Text>	Specifies the text for the help balloon associated with your guide file's menu item name in your application's Help menu.
<Checkbox>	Specifies a label for a checkbox.
<Define Context Check>	Specifies an additional parameter in a context check and indicates a text string of type LPSTRING. The text string specifies a default value. You must translate this text string to match the corresponding default value as it exists in the localized application.
<Define Item Coach>	Specifies the item that is highlighted by a coachmark. You should translate the text string to match the corresponding item as it exists in the localized application.
<Define Menu Coach>	Specifies the menu name or menu item that is highlighted by a coachmark. You should translate the text string to match the corresponding menu name or menu item as it exists in the localized application. The text resource file does <i>not</i> list the menu name and its menu items in consecutive order.
<Define Prompt Set>	Specifies a navigation prompt that appears on the bottom of a panel. A prompt set consists of four navigation prompts; Guide Maker creates a text resource for each of these prompts.
<Define Sequence>	Specifies a sequence display title. This text appears in the title bars of all panels of the sequence.
<Define Window Coach>	Specifies a window in which a coachmark is drawn. You should translate the text string to match the corresponding window name as it exists in the localized application.

Table 7-1 The 'TEXT' resource names and the associated text strings (continued)

Text resource name	Description of the associated text string
<Exception>	<p>Specifies a word that should not be stemmed when Apple Guide parses a search phrase. To get a complete list of all exception words, create a Scopes and Keys report, as described in the chapter "Testing Your Guide File."</p> <p>Apple Guide performs stemming and uses the exception list only for guide files that specify the command <World Script> 0,0. Thus, when translating to other languages, you do not need to translate exception words. So you should delete <Exception> commands from your localized source files.</p>
<Header>	Specifies a header associated with a particular topic area or index term. This text appears in the right column of the access window.
<Help Menu>	Specifies the name of a guide file or the Command key shortcut. Guide Maker creates a text resource for the guide file name and one for the Command key shortcut. The guide file name and its shortcut are displayed in the Help menu.
<Hot Text>	Specifies hot text in a panel. You should translate this text string to match the corresponding text string as it appears in the localized panel.
<If>	<p>Specifies text (with data type LPSTRING) that is part of a condition. If the condition statement contains more than one LPSTRING, Guide Maker creates a text resource for each one. If the condition is part of a compound condition statement, Guide Maker creates one text resource for each LPSTRING in the compound condition.</p> <p>In some cases, the text string specifies a label of a radio button or checkbox.</p> <p>You must translate this text string to match the corresponding text string as it exists in the localized application.</p>
<Ignore>	Specifies a word or phrase that Apple Guide ignores when parsing a search phrase. To get a list of these words and phrases, create a Scopes and Keys report, as described in the chapter "Testing Your Guide File."

continued

Table 7-1 The 'TEXT' resource names and the associated text strings (continued)

Text resource name	Description of the associated text string
<Index>	Specifies an index term. The index term appears in the left column when Index is the active list.
<Look For Instruction>	Specifies an instruction that appears above the search phrase entry box in the Full Access window when Look For is the active list.
<Look For Results Instruction>	Specifies an instruction that appears above the list of topics in the Full Access window when Look For is the active list and the user has performed a successful search.
<Look For Search Btn Instruction>	Specifies an instruction that appears above the Search button in the Full Access window when Look For is the active list.
<Look For String>	Specifies text that appears in the search phrase entry box in the Full Access window when Look For is the active list.
<Make Sure>	<p>Specifies text (with data type LPSTRING) that is part of a condition. The evaluated condition determines whether the next panel should be displayed. If the condition statement contains more than one LPSTRING, Guide Maker creates a text resource for each of these. If the condition is part of a compound condition statement, Guide Maker creates one text resource for each LPSTRING in the compound condition.</p> <p>In some cases, the text string specifies a label of a radio button or checkbox.</p> <p>You must translate this text string to match the corresponding one as it exists in the localized application.</p>
Panel: <i>panel name</i>	Specifies the body text to be displayed in the panel specified by the <i>panel name</i> . If the text in the panel is divided into multiple pieces (for example, the text is split between a picture, a QuickTime movie, or a button, or the text uses more than one format), Guide Maker creates a text resource for each of the text parts. The created text resources have identical text resource names but different resource IDs.
<Radio Button>	Specifies a label for a radio button.

Table 7-1 The 'TEXT' resource names and the associated text strings (continued)

Text resource name	Description of the associated text string
<Skip If>	<p>Specifies text (with data type <code>LPSTRING</code>) that is part of a condition. The evaluated condition determines whether the next panel should be displayed. If the condition statement contains more than one <code>LPSTRING</code>, Guide Maker creates a text resource for each of these. If the condition is part of a compound condition statement, Guide Maker creates one text resource for each <code>LPSTRING</code> in the compound condition.</p> <p>In some cases, the text string specifies a label of a radio button or checkbox.</p> <p>You must translate this text string to match the corresponding text string as it exists in the localized application.</p>
<Standard Button>	Specifies a label for a standard button.
<Synonym> <i>index term, synonymous term</i>	<p>Specifies a synonymous term for the index term specified in the text resource name. If an index term has more than one synonym, Guide Maker creates a text resource for each of these. Note that the text resource file does <i>not</i> group an index term and its synonyms together (unless they're grouped together in the source file). To get a complete list of all index terms and their synonyms, create a Scopes and Keys report, as described in the chapter "Testing Your Guide File."</p>
Text Block: <i>text block name</i>	Specifies a block of text. This block of text typically appears in the body text of a howdy window.
<Topic>	Specifies a topic. This text string appears in the right column of the access window.
<Topic Area>	Specifies a topic area. This text appears in the left column of the Topics screen in the access window. Note that you are limited to 31 one-byte characters.

continued

Table 7-1 The 'TEXT' resource names and the associated text strings (continued)

Text resource name	Description of the associated text string
<Topic Areas Instruction>	Specifies an instruction or a label. This text appears above the list of topic areas in the Full Access window when Topics is the active list.
<Topics Instructions>	Specifies an instruction that appears above the list of topics in the Full Access window when Topics or Index is the active list.
<Version>	Specifies version information for the guide file. Guide Maker creates a text resource for both the long and short version information strings.

Converting Windows Help Files

Contents

Preparing Your Windows Help Files	8-3
Converting Your Windows Help Files in Three Steps	8-4
Creating an Interface for Your Help Content	8-8

Converting Windows Help Files

If you have created online help for Windows and you want to make it available for the Mac OS, read this chapter. It describes how to

- prepare your Windows Help files for conversion
- convert them into Guide Script source files
- create an access window, so that your users can access the converted help

To get the most out of this chapter, you should have a good understanding of Apple Guide, know how to script a Guide Script source file, and be familiar with the Guide Maker application. For more information on these subjects, see Part 1, Part 4, and the introduction to Part 2, respectively.

This chapter does not describe how to compile the created Guide Script source files into a guide file. For information on compiling a guide file, see the chapter “Creating Your Guide File.”

Preparing Your Windows Help Files

Before you convert your Windows Help files to Guide Script source files, make sure that your Windows Help files meet two conditions:

- Your Windows Help files *must* be RTF files of type 'TEXT'. Guide Maker (the application you use to convert your files) cannot convert files that aren't WinHelp RTF files. Check that the file type for all the RTF files you are converting are 'TEXT'; if the type isn't 'TEXT', Guide Maker isn't able to see them. You can set the file type (to 'TEXT') using the ResEdit application or another file editing tool.
- Your Windows Help files cannot contain underlined hidden text. If any underlined hidden text exists, remove the underline before converting the files.

In addition to these two requirements, you should note that the current version of Guide Maker

- removes page breaks from the Windows Help files
- ignores any tables in the Windows Help files
- removes occurrences of the text “>List” in Windows Help files

Converting Windows Help Files

- assumes that a Windows Help file topic, which is equivalent to a panel sequence in a Guide Script source file, begins with the footnote '#' and ends at the next '#' footnote
- requires the footnotes '#' and '\$' for every help topic
- ignores the footnotes '!' and '+'
- uses the footnote 'K' to create index terms
- converts the command `bmc` (the command to place bitmaps in Windows) to the commented out Guide Script command
`#<PICT> "NameOfPicture", CENTER`
The Guide Script command is commented out because Apple Guide isn't able to display your Windows Help bitmaps. Before you remove the comment sign (the # sign), convert your bitmaps to PICT files.

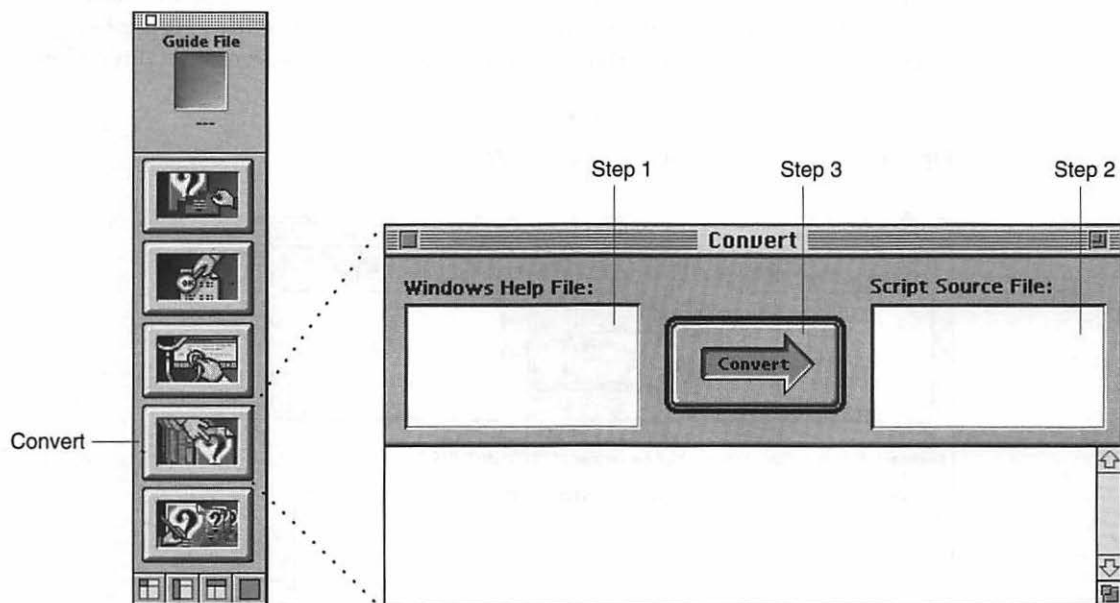
Note

All footnotes must appear on the same line. All other text and bitmaps on the first line are ignored and removed by Guide Maker. ♦

Once you have prepared your Windows Help files, you are ready to convert them, as described in the next section.

Converting Your Windows Help Files in Three Steps

Converting a Windows Help file to a Guide Script source file is a three-step process using Guide Maker's **Convert utility**. Figure 8-1 illustrates the Convert window (Guide Maker's converting interface). Note that the steps in the illustration refer to the three steps of the conversion process.

Figure 8-1 Converting your Windows Help files using Guide Maker's Convert utility

To convert your Windows Help files, follow these three steps:

1. Select the Windows Help file to convert.

To select the Windows Help file, click in the Windows Help File area of the Convert window (see Figure 8-1); a standard file dialog box appears, requesting that you select the Windows Help file to convert. Select the file you want to convert.

2. Select a name for the converted source file.

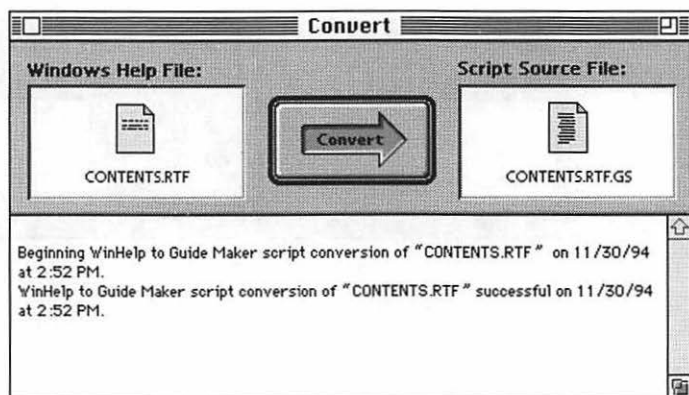
To name the converted source file, click in the Script Source File area of the Convert window (see Figure 8-1); a dialog box appears, requesting that you name the converted file. Typically, you give the converted source file the name of your Windows Help file and end the name in .GS (for Guide Script).

Converting Windows Help Files

3. Click the Convert arrow.

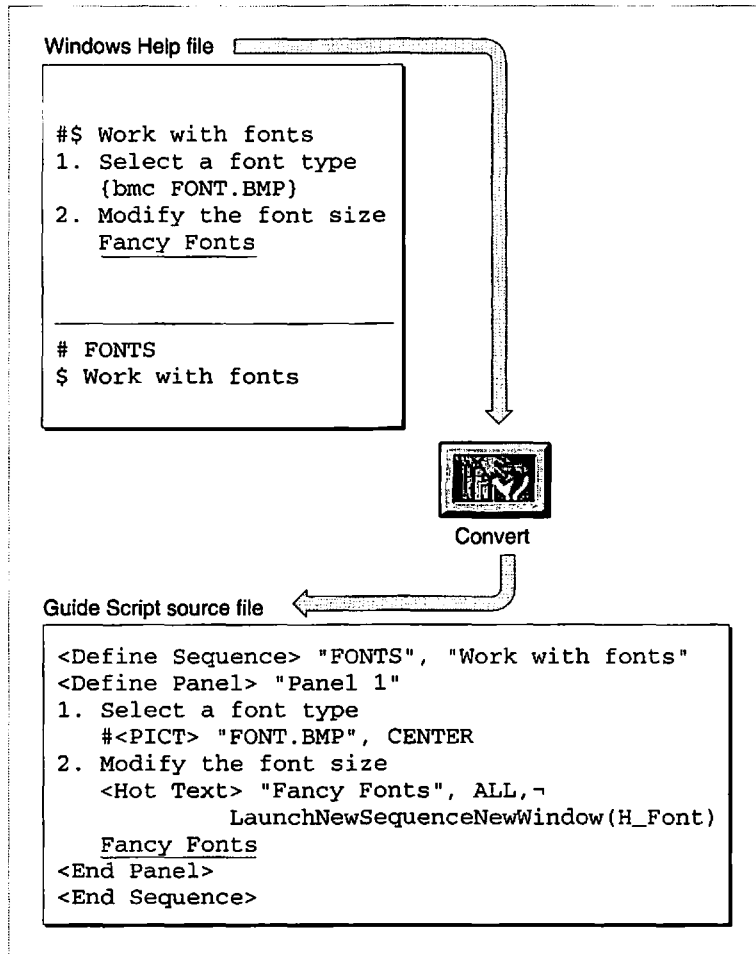
Guide Maker begins to convert your Windows Help file, reporting status and error messages in the status area of the Convert window. Figure 8-2 shows the messages reported by Guide Maker after a successful conversion.

Figure 8-2 A successfully converted Windows Help file



Congratulations! By following the conversion steps, you have converted a Windows Help file into a Guide Script source file. To convert all of your Windows Help files, repeat these three steps.

Figure 8-3 illustrates a sample Windows Help file and its converted Guide Script source file. Note that the hot text—Fancy Fonts—has been successfully converted; Guide Maker converts automatically all references to hot text and hot objects.

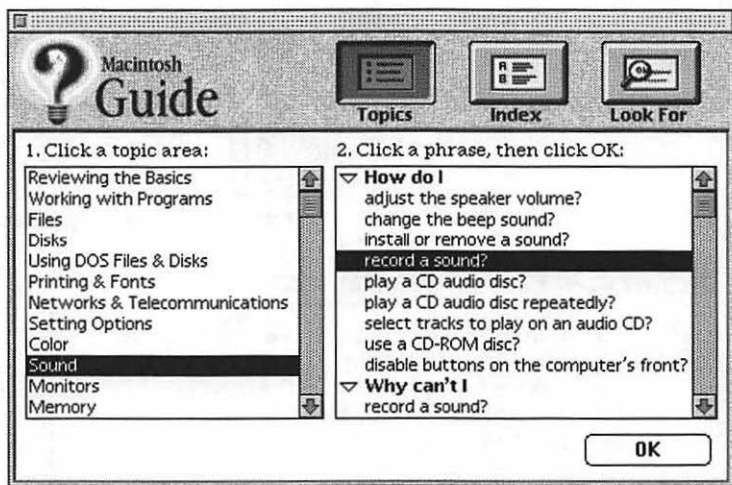
Figure 8-3 A Windows Help file and its converted Guide Script source file

When all of your Windows Help files are converted to Guide Script source files, you need to create an interface for your help content; see the next section for more information on this topic.

Creating an Interface for Your Help Content

Once you have converted all of your Windows Help files, you need to construct an interface (that is, an access window) so that your users can access your help content. Figure 8-4 shows a typical guide file interface, a Full Access window.

Figure 8-4 Creating an interface for your help content



To create an access window, first specify the type of window you want, using the <Startup Window> command. You can choose from three types of access windows: Full Access, Single List Access, and Simple Access. For information on these windows see the chapter "Authoring Tips and Suggestions" in Part 1.

Note

For information on the Guide Script commands used in this section, see the chapter "Guide Script Command Reference" in Part 4. ♦

Converting Windows Help Files

Once you have specified which type of access window you want, you need to define the topic areas, headers, and topics that are visible in the access window.

- For topic areas, use the <Topic Area> command.
- For headers, use the <Header> command.
- For the topics for the headers, use the <Topic> command.

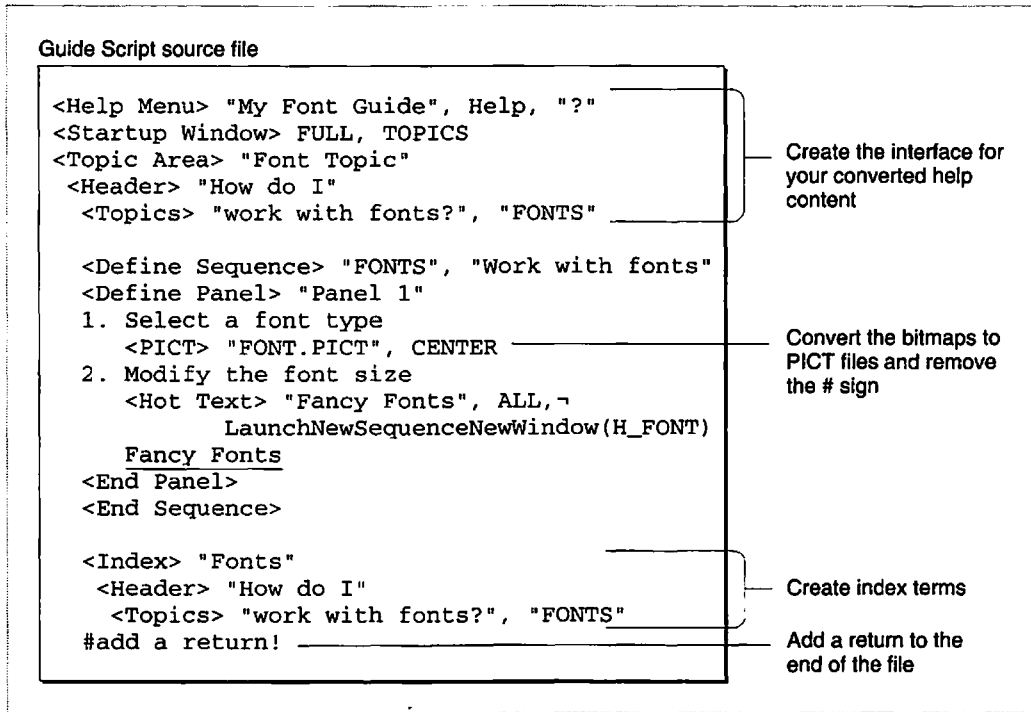
For example, if you have converted a panel sequence that describes how to record a sound, you might want to define a topic area called “Sound”, a header called “How do I”, and a topic called “record a sound?” (as illustrated in the Full Access window in Figure 8-4).

In addition to creating the access window, you need to

- name your guide file, using the <Help Menu> command
The guide file name appears in the Help menu whenever your guide file is available.
- convert your bitmaps to PICT files and remove the comment (#) sign from the <PICT> command
- add navigation buttons to your panels
- create an index for your help content, using the <Index>, <Header>, and <Topic> commands

Note that if your Windows Help file contains index terms (they are specified by the 'K' footnote), Guide Maker converts these automatically. The converted index terms do not, however, include titles for the <Header> command; you must provide these. For information on how to design an index, see the chapter “Planning Your Help Content” in Part 1.

Figure 8-5 illustrates a sample source file with its interface commands.

Figure 8-5 Constructing an interface for a sample source file

When you have constructed the interface, converted your bitmaps to PICT files, and created an index, you are ready to compile your source files into a guide file (this step is known as building your guide file). For information on how to build your guide file, see the chapter "Creating Your Guide File."

Integrating Guide Files

Apple Guide API

Contents

Introduction	9-3
Determining Whether Apple Guide Is Available	9-4
Starting Up Apple Guide	9-5
AGStart	9-5
AGQuit	9-6
AGGetStatus	9-6
Determining Which Guide Files Are Available	9-7
AGGetAvailableDBTypes	9-7
AGFileGetDBCCount	9-9
AGFileGetIndDB	9-10
Opening and Closing Guide Files	9-11
AGOpen	9-12
AGOpenWithView	9-13
AGOpenWithSearch	9-14
AGOpenWithSequence	9-16
AGClose	9-17
Working With Open Guide Files	9-18
AGIsDatabaseOpen	9-18
AGGetFrontWindowKind	9-19
AGGeneral	9-20
AGGetFSSpec	9-22
Getting Information About Guide Files	9-22
AGFileGetDBMenuName	9-23
AGFileGetHelpBalloonText	9-24
AGFileGetHelpMenuAppCreator	9-25
AGFileGetDBType	9-26

AGFileGetDBCountr	9-28
AGFileGetDBVersion	9-29
AGFileGetSelectorCount	9-30
AGFileGetSelector	9-31
AGFileIsMixin	9-32
AGFileGetMixinMatchSelector	9-32
Installing and Removing Coachmark Handlers	9-33
AGInstallCoachHandler	9-34
AGRemoveCoachHandler	9-35
Installing and Removing Context Check Handlers	9-36
AGInstallContextHandler	9-36
AGRemoveContextHandler	9-38
Application-Defined Routines	9-38
Providing Object Locations for Coachmarks	9-38
MyCoachReplyProc	9-39
Responding to Context Checks	9-40
MyContextReplyProc	9-40
Summary of Apple Guide API	9-45

You can provide the user with context-sensitive help from within your application by using the **Apple Guide application programming interface (API)**. This application programming interface is a set of functions that allow you to start up Apple Guide, access and work with guide files—all from within your application. An additional set of functions is provided in the AGFile library; you can use these functions to get information about guide files. You should read this chapter if you are a developer who wants to integrate Apple Guide with your application.

Usually the user controls when help is displayed by choosing a guide file from the Help menu. However, using the Apple Guide API, you can make help available from a button in a dialog box or offer to provide help to the user if the user repeatedly makes a mistake. For example, you can display to the user a sequence that is directly related to the task the user is performing.

This chapter begins by introducing you to the types of tasks your application can accomplish using the Apple Guide API and AGFile library. Then it gives a complete description of all of their functions.

Introduction

This chapter describes two sets of routines: those in the Apple Guide API and those in the AGFile library. The routines in the Apple Guide API are available in System 7.5 or later. The routines prefixed with “AGFile” are not included in System 7.5 but are part of the AGFile library; to use these routines you must link the AGFile library with your application when you build it.

You can use functions in the Apple Guide API and AGFile library to

- start up Apple Guide and get information about its status
- get the number and type of guide files available
- open and close guide files
- work with open guide files
- get information about guide files
- provide object location for coachmarks
- respond to context checks

Determining Whether Apple Guide Is Available

To determine whether Apple Guide is available, call the `Gestalt` function with the `gestaltHelpMgrAttr` selector and check the value of the response parameter. If the bit indicated by the constant `gestaltAppleGuidePresent` is set, then Apple Guide (and its API) is available. If the bit indicated by the constant `gestaltAppleGuideIsDebug` is set, then the Apple Guide Debug extension is installed (you use the Apple Guide Debug extension only for debugging your guide file).

```
enum {
    gestaltHelpMgrAttr      'help'    /*Gestalt selector for Help Mgr */
                                /* and Apple Guide*/
    gestaltAppleGuidePresent = 31,    /*Apple Guide API is available*/
    gestaltAppleGuideIsDebug = 30     /*Apple Guide Debug extension */
                                /* is installed*/
};
```

For example, this code determines whether Apple Guide is available:

```
long response = 0;
OSErr err;

err = Gestalt(gestaltHelpMgrAttr, &response);
if (err == noErr && (response & (1 << gestaltAppleGuidePresent)))
    /*Apple Guide is available*/
    kAppleGuideAvailable = 1
else
    /*Apple Guide is not available*/
    kAppleGuideAvailable = 0;
```

For information on the `Gestalt` function, see the chapter “Gestalt Manager” in *Inside Macintosh: Operating System Utilities*.

Starting Up Apple Guide

You can use three functions—`AGStart`, `AGQuit`, and `AGGetStatus`—to start up, stop, and determine the status of Apple Guide. Apple Guide contains two components: a stay-resident portion that is always in memory and an application portion that is required to be in memory only while a guide file is open. The application portion is launched in its own heap as a faceless, background application. If it is not already in memory, it is automatically launched when your application opens a guide file. Thus, your application doesn't usually need to call `AGStart`. However, if your application opens a guide file, it should always close the guide file and call `AGQuit` before quitting.

AGStart

Use the `AGStart` function to start up Apple Guide.

```
AGErr AGStart(void);
```

DESCRIPTION

The `AGStart` function launches the application portion of Apple Guide. If it is successfully launched or already in memory, the `AGStart` function returns the `noErr` result code.

SPECIAL CONSIDERATIONS

The Apple Guide functions that open guide files automatically call `AGStart`.

RESULT CODES

<code>noErr</code>	0	No error
--------------------	---	----------

AGQuit

Use the `AGQuit` function to quit Apple Guide.

```
AGErr AGQuit(void);
```

DESCRIPTION

The `AGQuit` function checks to see if any guide files are open. If it finds one open, it does not quit Apple Guide. If no guide files are open, it quits the application portion of Apple Guide.

If your application opens a guide file, it should always close the guide file and call `AGQuit` before quitting. Otherwise, if the application portion of Apple Guide is active or sleeping, it remains in memory.

SPECIAL CONSIDERATIONS

Don't force Apple Guide to quit when your application goes to the background. If your application has opened a guide file and the user switches to another application, your guide file should remain open until the user closes it, opens another guide file, or quits your application.

RESULT CODES

<code>noErr</code>	0	No error
<code>kAGErrDatabaseOpen</code>	-2961	No open guide file

AGGetStatus

Use the `AGGetStatus` function to determine the status of Apple Guide.

```
AGStatus AGGetStatus(void);
```

DESCRIPTION

The `AGGetStatus` function determines whether Apple Guide is active, sleeping, or not running by returning one of the following constants:

- `kAGIsActive`, if the application portion of Apple Guide is in memory and a guide file is open
- `kAGIsSleeping`, if the application portion of Apple Guide is in memory but no guide files are open
- `kAGIsNotRunning`, if the application portion of Apple Guide is not in memory

SEE ALSO

To determine whether the Apple Guide API is available, use the Gestalt Manager, as described in “Determining Whether Apple Guide Is Available” on page 9-4.

Determining Which Guide Files Are Available

To determine the number and types of guide files that are available, use the routines described in this section. After finding the desired guide file, you can obtain its file system specification record (you must provide a file system specification record to the Apple Guide functions that open guide files).

AGGetAvailableDBTypes

Use the `AGGetAvailableDBTypes` function to determine the types of guide files that are available in the current application’s Help menu.

```
UInt32 AGGetAvailableDBTypes(void);
```

DESCRIPTION

The `AGGetAvailableDBTypes` function returns a collection of bit flags that indicate the types of guide files that are available in the current application's Help menu:

```
enum AGDBTypeBit
{
    kAGDBBitAny           = 0x00000001, /*one or more guide */
                                   /* files are present*/
    kAGDBTypeBitHelp      = 0x00000002, /*Help guide file*/
    kAGDBTypeBitTutorial  = 0x00000004, /* file*/
    kAGDBTypeBitShortcuts = 0x00000008, /*Shortcuts guide */
                                   /* file*/
    kAGDBTypeBitAbout     = 0x00000010, /*About guide file*/
    kAGDBTypeBitOther     = 0x00000080, /*Other guide file*/
};
```

The `AGGetAvailableDBTypes` function returns

- `kAGDBBitAny` if one or more guide files are present
- `kAGDBTypeBitHelp` if a Help guide file is present
- `kAGDBTypeBitTutorial` if a Tutorial guide file is present
- `kAGDBTypeBitShortcuts` if a Shortcuts guide file is present
- `kAGDBTypeBitAbout` if an About guide file is present
- `kAGDBTypeBitOther` if an Other guide file is present
- 0 if no guide files are present

SEE ALSO

To count the number of guide files of a particular type, use the `AGFileGetDBCount` function, described next. To get the file system specification record for a guide file, use the `AGFileGetIndDB` function, as described on page 9-10.

AGFileGetDBCount

Use the `AGFileGetDBCount` function to count the number of guide files in a specified folder.

```
AGFileCountType AGFileGetDBCount
    (short vRefNum, long dirID,
     AGFileDBType databaseType,
     Boolean wantMixin);
```

vRefNum The volume reference of the volume on which the guide file is located.

dirID The directory ID of the directory where the guide file is located.

databaseType The guide file type. Use these constants to specify the type of guide file:

```
enum {
    kAGFileDBTypeAny          = 0, /*all types*/
    kAGFileDBTypeHelp         = 1, /*Help*/
    kAGFileDBTypeTutorial     = 2, /*Tutorial*/
    kAGFileDBTypeShortcuts    = 3, /*Shortcuts*/
    kAGFileDBTypeAbout        = 4, /*About*/
    kAGFileDBTypeOther        = 8 /*Other*/
};
```

wantMixin A flag. Set to `TRUE` if you want to count the number of main guide files and Mixin guide files. Set to `FALSE` if you want to count only the number of main guide files.

DESCRIPTION

The `AGFileGetDBCount` function returns the number of guide files available in the folder specified by the `vRefNum` and `dirID` parameters. This function counts only the guide files of the type specified in the `databaseType` parameter.

AGFileGetIndDB

Use the `AGFileGetIndDB` function to obtain the file system specification record of a guide file located in a specified folder.

`OSErr AGFileGetIndDB`

```
(short vRefNum, long dirID,
  AGFileType databaseType,
  Boolean wantMixin,
  short dbIndex, FSSpecType *fileSpec);
```

vRefNum The volume reference number of the volume on which the guide file is located.

dirID The directory ID of the directory where the guide file is located.

databaseType

The guide file type. Use these constants to specify the type of guide file:

```
enum {
  kAGFileDBTypeAny          = 0, /*any guide file*/
  kAGFileDBTypeHelp         = 1, /*Help*/
  kAGFileDBTypeTutorial     = 2, /*Tutorial*/
  kAGFileDBTypeShortcuts    = 3, /*Shortcuts*/
  kAGFileDBTypeAbout        = 4, /*About*/
  kAGFileDBTypeOther        = 8 /*Other*/
};
```

wantMixin A flag. Specify `TRUE` if the desired guide file is a Mixin guide file. Specify `FALSE` if the guide file is not a Mixin guide file.

dbIndex A number representing the guide file index. To access the first guide file of the specified type in the folder, set this parameter to 1. To access the second guide file of this type, specify 2, and so on.

fileSpec A pointer to a file system specification record. On return, this parameter refers to the file system specification record for the guide file.

DESCRIPTION

The `AGFileGetIndDB` function returns, through its `fileSpec` parameter, the file system specification record for the specified guide file. You can call `AGFileGetIndDB` repetitively to access the file system specification record for all of the guide files in a folder. To access all of the guide files, increment the `dbIndex` parameter by 1, until the value of the `dbIndex` parameter is equal to the number of guide files of the requested type in the folder, or until the function result is nonzero.

RESULT CODES

<code>noErr</code>	0	No error
<code>dirNFErr</code>	-12	Directory not found or incomplete pathname
<code>nsvErr</code>	-35	Volume doesn't exist

SEE ALSO

After obtaining a guide file's file system specification record, you can get information about the guide file by using the functions described in "Getting Information About Guide Files" beginning on page 9-22. You can also open a guide file, as described next.

Opening and Closing Guide Files

This section describes the functions you can use to open and close guide files. You can use these functions to

- open a guide file in its default active list (Topics, Index, Look For, or Howdy)
- open a guide file and specify which list the guide file should open in (Topics, Index, Look For, or Howdy)
- open a guide file and specify the sequence to display
- open a guide file with Look For active and immediately perform a search
- close a guide file

AGOpen

Use the `AGOpen` function to open a guide file in its default active list. You can use this function to open any guide file.

```
AGErr AGOpen(FSSpec *fileSpec, UInt32 flags,
             Handle mixinControl,
             ARefNum *resultRefNum);
```

fileSpec A pointer to the file system specification record of the guide file you wish to open. Specify `NIL` to open the first guide file of type `Help` that is available to the application.

flags Reserved. Specify 0 in this parameter.

mixinControl Reserved. Specify `NIL` in this parameter.

resultRefNum An address through which `AGOpen` returns a reference number for the guide file specified in the `fileSpec` parameter. You use this reference number to refer to the guide file in other Apple Guide functions.

DESCRIPTION

The `AGOpen` function opens the guide file specified in the `fileSpec` parameter. It opens the guide file in its default active list. (For a Full Access window, Topics, Index, Look For, or Howdy; for a Single List Access window, Topics or Howdy; for a Simple access window, the first panel of its sequence.) If the application portion of Apple Guide is not in memory, `AGOpen` calls `AGStart` to start up Apple Guide before it opens the specified guide file.

RESULT CODES

<code>noErr</code>	0	No error
<code>kAGErrCannotOpenAliasFile</code>	-2954	Unable to open alias
<code>kAGErrDatabaseNotAvailable</code>	-2956	Guide file is not available
<code>kAGErrInsufficientMemory</code>	-2962	Not enough memory

SEE ALSO

For a description of the `AGStart` function, see page 9-5.

AGOpenWithView

Use the `AGOpenWithView` function to open a guide file and specify its initial active list. You can use this function to open a guide file that uses a Full Access window or Single List access window.

```
AGErr AGOpenWithView (FSSpec *fileSpec, UInt32 flags,
                      Handle mixinControl,
                      short viewNum,
                      AGRefNum *resultRefNum);
```

fileSpec A pointer to the file system specification record for the guide file you wish to open. Specify `NIL` to open the first guide file of type Help that is available to the application.

flags Reserved. Specify 0 in this parameter.

mixinControl Reserved. Specify `NIL` in this parameter.

viewNum A value that indicates which list to display. You can use these constants to specify which list should be initially active:

```
enum {
    kAGViewFullHowdy      = 1,  /*full howdy*/
    kAGViewTopicAreas     = 2,  /*Topic Area*/
    kAGViewIndex          = 3,  /*Index*/
    kAGViewLookFor        = 4,  /*Look For*/
    kAGViewSingleHowdy    = 5,  /*Single List *
                                /* howdy*/
    kAGViewSingleTopics   = 6   /*Single List */
                                /* topics*/
};
```

Apple Guide API

resultRefNum

An address through which `AGOpenWithView` returns a reference number for the guide file specified in the `fileSpec` parameter. You use this reference number to refer to the guide file in other Apple Guide functions.

DESCRIPTION

The `AGOpenWithView` function opens the guide file specified in the `fileSpec` parameter, and displays the active list specified in the `viewNum` parameter. If the application portion of Apple Guide is not in memory, `AGOpenWithView` calls `AGStart` to start up Apple Guide before it opens the specified guide file.

RESULT CODES

<code>noErr</code>	0	No error
<code>kAGErrCannotOpenAliasFile</code>	-2954	Unable to open alias
<code>kAGErrNoAliasResource</code>	-2955	Unable to open resource alias
<code>kAGErrDatabaseNotAvailable</code>	-2956	Guide file is not available
<code>kAGErrInsufficientMemory</code>	-2962	Not enough memory

AGOpenWithSearch

Use the `AGOpenWithSearch` function to open a guide file and immediately start a search on a specified search phrase. You can use this function to open a guide file that uses a Full Access window.

```
AGErr AGOpenWithSearch(FSSpec *fileSpec, UInt32 flags,
                       Handle mixinControl,
                       ConstStr255Param searchString,
                       ARefNum *resultRefNum);
```

fileSpec	A pointer to the file system specification record for the guide file you wish to open. Specify <code>NIL</code> to open the first guide file of type Help that is available to the application.
flags	Reserved. Specify 0 in this parameter.

mixinControl

Reserved. Specify NIL in this parameter.

searchString

The search phrase to place in the search phrase entry box. When the guide file (specified in the **fileSpec** parameter) opens up, Apple Guide searches on this phrase.

resultRefNum

An address through which **AGOpenWithSearch** returns a reference number for the guide file specified in the **fileSpec** parameter. You use this reference number to refer to the guide file in other Apple Guide functions.

DESCRIPTION

The **AGOpenWithSearch** function opens the guide file, specified in the **fileSpec** parameter, in a Full Access window with Look For active. After the guide file opens, Apple Guide immediately starts to search the open guide file for any occurrences of the search phrase specified in the **searchString** parameter. If the application portion of Apple Guide is not in memory, **AGOpenWithSearch** first calls the **AGStart** function to start up Apple Guide before it opens the specified guide file.

RESULT CODES

noErr	0	No error
kAGErrCannotOpenAliasFile	-2954	Unable to open alias
kAGErrNoAliasResource	-2955	Unable to open resource alias
kAGErrDatabaseNotAvailable	-2956	Guide file is not available
kAGErrInsufficientMemory	-2962	Not enough memory

AGOpenWithSequence

Use the `AGOpenWithSequence` function to open a guide file and immediately display a panel sequence. You can use this function to open any guide file.

```
AGErr AGOpenWithSequence(FSSpec *fileSpec, UInt32 flags,
                        Handle mixinControl,
                        short sequenceID,
                        ARefNum *resultRefNum);
```

fileSpec A pointer to the file system specification record for the guide file you wish to open. Specify `NIL` to open the first guide file of type `Help` that is available to the application.

flags Reserved. Specify 0 in this parameter.

mixinControl Reserved. Specify `NIL` in this parameter.

sequenceID The sequence ID of the sequence to display. Guide Maker assigns sequence IDs to sequences when it compiles a guide file. You can obtain a list of all the sequence IDs in a guide file, by generating a Names to IDs report. For more information on how to generate a Names to IDs report, see the chapter “Testing Your Guide File” in Part 2.

resultRefNum An address through which `AGOpenWithSequence` returns a reference number for the guide file specified in the `fileSpec` parameter. You use this reference number to refer to the guide file in other Apple Guide functions.

DESCRIPTION

The `AGOpenWithSequence` function opens the guide file specified in the `fileSpec` parameter and immediately displays the first panel of the sequence specified in the `sequenceID` parameter. If the application portion of Apple Guide is not in memory, `AGOpenWithSequence` first calls `AGStart` to start up Apple Guide before it opens the specified guide file.

RESULT CODES

<code>noErr</code>	0	No error
<code>kAGErrCannotOpenAliasFile</code>	-2954	Unable to open alias
<code>kAGErrNoAliasResource</code>	-2955	Unable to open resource alias
<code>kAGErrDatabaseNotAvailable</code>	-2956	Guide file is not available
<code>kAGErrInsufficientMemory</code>	-2962	Not enough memory

AGClose

Use the `AGClose` function to close a specified guide file.

```
AGErr AGClose(AGRefNum *resultRefNum);
```

`resultRefNum`

A pointer to the reference number for the guide file you wish to close.

DESCRIPTION

The `AGClose` function closes the guide file specified in the `resultRefNum` parameter. You use `AGClose` to close a guide file that was opened by your application. If you attempt to close a guide file that was opened by another application, the `AGClose` function returns a nonzero result code. Note that a call to `AGClose` does not quit Apple Guide; it continues to run in the background.

If your application opens a guide file, it should always close the guide file and call `AGQuit` before quitting. Otherwise, if the application portion of Apple Guide is active or sleeping, it remains in memory.

SPECIAL CONSIDERATIONS

Don't close an open guide file or force Apple Guide to quit when your application goes to the background. If your application has opened a guide file and the user switches to another application, your guide file should remain open until the user closes it, opens another guide file, or quits your application.

RESULT CODES

<code>noErr</code>	0	No error
<code>kAGErrDatabaseNotOpen</code>	-2957	Guide file is not open
<code>kAGErrInvalidRefNum</code>	-2960	The guide file was opened by another application

SEE ALSO

For a description of the `AGQuit` function, see page 9-6.

Working With Open Guide Files

This section describes how to work with open guide files. Its four functions—`AGIsDatabaseOpen`, `AGGetFrontWindowKind`, `AGGeneral`, and `AGGetFSSpec`—describe, for any specified open guide file, how to

- verify that it is still open
- determine whether its access window or one of its panels is showing
- request Apple Guide to perform an action related to its display (such as showing the next panel)
- get its file system specification record

AGIsDatabaseOpen

Use the `AGIsDatabaseOpen` function to verify that a guide file is still open.

```
Boolean AGIsDatabaseOpen(AGRefNum refNum);
```

`refNum` The reference number for the guide file.

DESCRIPTION

If the guide file (specified in the `refNum` parameter) is open, `AGIsDatabaseOpen` returns `TRUE`, and if the guide file is closed, `AGIsDatabaseOpen` returns `FALSE`.

The user or another application can close a guide file that your application explicitly opens. A user can directly close your guide file by clicking in its close box. The user or another application can indirectly close your guide file by opening another guide file. (Before opening a guide file, Apple Guide closes the active guide file, if any.) Therefore, when your application switches from the background to the foreground, you should call `AGIsDatabaseOpen` to verify that your guide file is still open.

SEE ALSO

You can also verify that a guide file is open by using the function `AGGetFrontWindowKind` function. The `AGGetFrontWindowKind` function (described next) returns additional information about a guide file.

AGGetFrontWindowKind

Use the `AGGetFrontWindowKind` function to determine, for a specified open guide file, the type of window that is currently being displayed.

```
AGWindowKind AGGetFrontWindowKind(AGRefNum refNum);
```

refNum A reference number for a guide file or the constant `kAGFrontDatabase` to specify the active guide file.

DESCRIPTION

The `AGGetFrontWindowKind` function returns information indicating whether the access window or a presentation window of the specified guide file is currently showing. It also indicates whether the guide file is open.

The `AGGetFrontWindowKind` function returns

- `kAGAccessWindow` if the guide file's access window is showing
- `kAGPresentationWindow` if a presentation window (panel) of the guide file is showing
- `kAGNoWindow` if the specified guide file is not open (which indicates that the application portion of Apple Guide is either sleeping or not running)

AGGeneral

Use the `AGGeneral` function to perform actions on an open guide file.

```
AGErr AGGeneral (AGRefNum refNum, AGEvent theEvent);
```

refNum A reference number of an open guide file.

theEvent A four-character sequence that indicates the action to perform. (These actions correspond to Apple Guide Apple events.) You specify the action using one of these constants:

```
enum {
    /*Apple Guide Apple events for guide files*/
    kAGEventDoCoach   = 'doco', /*coachmark*/
    kAGEventDoHuh     = 'dhuh', /*Huh? topic*/
    kAGEventGoNext    = 'gonp', /*go next*/
    kAGEventGoPrev    = 'gopp', /*go previous*/
    kAGEventHidePanel = 'pahi', /*hide panel*/
    kAGEventReturnBack= 'gobk', /*return back*/
    kAGEventShowPanel = 'pash', /*show panel*/
    kAGEventTogglePanel= 'patg' /*toggle panel*/
};
```

DESCRIPTION

The `AGGeneral` function sends an Apple event to Apple Guide, which then performs the requested action on the guide file specified in the `refNum` parameter.

The parameter `theEvent` specifies the action to perform, as indicated by the particular Apple event constant. You can specify any of these seven types of Apple events to perform the corresponding action related to an open guide file:

- `kAGEventDoCoach`, to draw a coachmark that exists for the active panel of the open guide file
- `kAGEventDoHuh`, to open a Huh? topic that belongs to the active panel of the open guide file
- `kAGEventGoNext`, to go to the next panel
- `kAGEventGoPrev`, to go to the previous panel
- `kAGEventShowPanel`, to expand the active panel
- `kAGEventTogglePanel`, to collapse or expand panel
- `kAGEventReturnBack`, to return from an Oops panel

You can direct Apple events only to a guide file that was opened by your application. If you attempt to direct an event to a guide file that was not opened by your application, the `AGGeneral` function returns a nonzero result code.

RESULT CODES

<code>noErr</code>	0	No error
<code>kAGErrDatabaseNotOpen</code>	-2957	Guide file is not open
<code>kAGErrInvalidRefNum</code>	-2960	The guide file was opened by another application

AGGetFSSpec

Use the `AGGetFSSpec` function to access the file system specification record of an open guide file.

```
AGErr AGGetFSSpec(AGRefNum refNum, FSSpec *fileSpec);
```

refNum The reference number for a guide file or the constant `kAGFrontDatabase` to specify the active guide file.

fileSpec A pointer to a file system specification record. The `AGGetFSSpec` function returns, through this parameter, the guide file's file system specification record.

DESCRIPTION

The `AGGetFSSpec` function returns (through the `fileSpec` parameter) the file system specification record for the guide file specified in the `refNum` parameter.

RESULT CODES

<code>noErr</code>	0	No error
<code>kAGErrDatabaseNotOpen</code>	-2957	Guide file is not open

SEE ALSO

To get the file system specification record for any guide file, use the `AGFileGetIndDB` function, as described on page 9-10.

Getting Information About Guide Files

This section describes how to get information about a guide file—how to

- find its Help menu item name
- get its associated help balloon text
- determine its creator

- determine its type
- get its script and region codes
- get its version
- get the number of its gestalt selectors
- access its gestalt selectors
- determine whether it's a Mixin guide file
- determine whether a Mixin guide file can be mixed with it

These functions require that you specify the file system specification record of the guide file you desire information about. To get a guide file's file system specification record, use the `AGFileGetIndDB` function (described on page 9-10).

The functions in this section do not require that the Apple Guide extension be installed; however, they do require that the application using them be built with the AGFile library.

AGFileGetDBMenuName

Use the `AGFileGetDBMenuName` function to obtain, for a specific guide file, the text that Apple Guide displays in an application's Help menu when the guide file is available.

```
OSErr AGFileGetDBMenuName
    (AGFileFSSpecType *fileSpec,
     AGFileDBMenuNamePtr menuItemNameStr);
```

fileSpec A pointer to the file system specification record for the guide file.

menuItemNameStr

A pointer to a text string. The `AGFileGetDBMenuName` function returns, through this parameter, the guide file's Help menu name.

DESCRIPTION

The `AGFileGetDBMenuName` function returns, in the `menuItemNameStr` parameter, the guide file's Help menu name.

RESULT CODES

<code>noErr</code>	0	No error
<code>nsvErr</code>	-35	Volume doesn't exist
<code>ioErr</code>	-36	I/O error
<code>fnOpnErr</code>	-38	File not open
<code>fnfErr</code>	-43	File or directory does not exist
<code>fLckdErr</code>	-45	File is locked
<code>rfNumErr</code>	-51	Bad reference number
<code>dirNFErr</code>	-120	Directory not found or incomplete pathname
<code>afpAccessDenied</code>	-5000	User does not have the correct access to the file

AGFileGetHelpBalloonText

Use the `AGFileGetHelpBalloonText` function to obtain the help balloon text associated with a guide file.

```
OSErr AGFileGetHelpBalloonText
    (AGFileFSSpecType *fileSpec,
     Str255 helpMenuBalloonString);
```

fileSpec A pointer to the file system specification record for the guide file.

helpMenuBalloonString
 A text string. The `AGFileGetHelpBalloonText` function
 returns, in this parameter, the help balloon text for the guide file.

DESCRIPTION

The `AGFileGetHelpBalloonText` function returns, in its `helpMenuBalloonString` parameter, the help balloon text associated with the guide file. Apple Guide displays this text in a help balloon when Balloon Help is on and the cursor is in the guide file's menu item in the Help menu.

RESULT CODES

<code>noErr</code>	0	No error
<code>nsvErr</code>	-35	Volume doesn't exist
<code>ioErr</code>	-36	I/O error
<code>fnOpnErr</code>	-38	File not open
<code>fnfErr</code>	-43	File or directory does not exist
<code>flckdErr</code>	-45	File is locked
<code>rfNumErr</code>	-51	Bad reference number
<code>dirNFErr</code>	-120	Directory not found or incomplete pathname
<code>afpAccessDenied</code>	-5000	User does not have the correct access to the file

AGFileGetHelpMenuAppCreator

Use the `AGFileGetHelpMenuAppCreator` function to retrieve the four-character value that specifies the application associated with this guide file.

```
OSErr AGFileGetHelpMenuAppCreator
    (AGFileFSSpecType *fileSpec,
     OSType *helpMenuAppCreator);
```

fileSpec A pointer to the file system specification record for the guide file.

helpMenuAppCreator

The `AGFileGetHelpMenuAppCreator` function returns, through this parameter, the four-character value that specifies the application associated with this guide file.

DESCRIPTION

The `AGFileGetHelpMenuAppCreator` function returns, through its `helpMenuAppCreator` parameter, the signature of the application that is associated with the guide file specified in the `fileSpec` parameter.

If a guide file specifies the application associated with it, then the value returned in the `helpMenuAppCreator` parameter must match the signature of the application for the guide file to appear in the application's Help menu. If the guide file does not specify the application that is associated with it, the `helpMenuAppCreator` parameter returns `NIL`.

RESULT CODES

<code>noErr</code>	0	No error
<code>nsvErr</code>	-35	Volume doesn't exist
<code>ioErr</code>	-36	I/O error
<code>fnOpnErr</code>	-38	File not open
<code>fnfErr</code>	-43	File or directory does not exist
<code>fLckdErr</code>	-45	File is locked
<code>rfNumErr</code>	-51	Bad reference number
<code>dirNFErr</code>	-120	Directory not found or incomplete pathname
<code>afpAccessDenied</code>	-5000	User does not have the correct access to the file

AGFileGetDBType

Use the `AGFileGetDBType` function to determine the type of a guide file.

```
OSErr AGFileGetDBType
    (AGFileFSSpecType *fileSpec,
     AGFileDBType *databaseType);
```

fileSpec A pointer to the file system specification record for the guide file.

databaseType

A pointer to the type of the guide file. The `AGFileGetDBType` function returns, through this parameter, a value that indicates the guide file type. The guide file type is indicated by one of these constants:

```
enum {
    kAGFileDBTypeHelp          = 1, /*Help*/
    kAGFileDBTypeTutorial      = 2, /*Tutorial*/
    kAGFileDBTypeShortcuts     = 3, /*Shortcuts*/
    kAGFileDBTypeAbout         = 4, /*About*/
    kAGFileDBTypeOther         = 8 /*Other*/
};
```

DESCRIPTION

The `AGFileGetDBType` function returns, in its `databaseType` parameter, the type of the specified guide file.

RESULT CODES

<code>noErr</code>	0	No error
<code>nsvErr</code>	-35	Volume doesn't exist
<code>ioErr</code>	-36	I/O error
<code>fnOpnErr</code>	-38	File not open
<code>fnfErr</code>	-43	File or directory does not exist
<code>fLckdErr</code>	-45	File is locked
<code>rflNumErr</code>	-51	Bad reference number
<code>dirNFErr</code>	-120	Directory not found or incomplete pathname
<code>afpAccessDenied</code>	-5000	User does not have the correct access to the file

AGFileGetDBCOUNTRY

Use the `AGFileGetDBCOUNTRY` function to obtain the script and region codes for a guide file.

```
OSErr AGFileGetDBCOUNTRY
    (AGFileFSSpecType *fileSpec,
     AGFileDBScriptType *script,
     AGFileDBRegionType *region);
```

<code>fileSpec</code>	A pointer to the file system specification record for the guide file.
<code>script</code>	A pointer to a short integer. On return, this parameter refers to the script code for the guide file specified in the <code>fileSpec</code> parameter.
<code>region</code>	A pointer to a short integer. On return, this parameter refers to the region code for the guide file specified in the <code>fileSpec</code> parameter.

DESCRIPTION

The `AGFileGetDBCOUNTRY` function returns, in the `script` and `region` parameters, the script and region code for a guide file. The guide file is specified in the `fileSpec` parameter.

RESULT CODES

<code>noErr</code>	0	No error
<code>nsvErr</code>	-35	Volume doesn't exist
<code>ioErr</code>	-36	I/O error
<code>fnOpnErr</code>	-38	File not open
<code>fnfErr</code>	-43	File or directory does not exist
<code>fLckdErr</code>	-45	File is locked
<code>rfNumErr</code>	-51	Bad reference number
<code>dirNFErr</code>	-12	Directory not found or incomplete pathname
<code>afpAccessDenied</code>	-5000	User does not have the correct access to the file

SEE ALSO

See the chapter “Script Manager” in *Inside Macintosh: Text* for a complete list of script codes and region codes.

AGFileGetDBVersion

Use the `AGFileGetDBVersion` function to obtain the version information of a guide file.

```
OSErr AGFileGetDBVersion
    (AGFileFSSpecType *fileSpec,
     AGFileMajorRevType *majorRev,
     AGFileMinorRevType *minorRev);
```

<code>fileSpec</code>	A pointer to the file system specification record for the guide file.
<code>majorRev</code>	A pointer to a short integer. On return, this parameter refers to the major version designation for the guide file specified in the <code>fileSpec</code> parameter.
<code>minorRev</code>	A pointer to a short integer. On return, this parameter refers to the minor version designation for the guide file specified in the <code>fileSpec</code> parameter.

DESCRIPTION

The `AGFileGetDBVersion` function returns in its parameters—`majorRev` and `minorRev`—the major and minor versions of the guide file.

RESULT CODES

<code>noErr</code>	0	No error
<code>nsvErr</code>	-35	Volume doesn't exist
<code>ioErr</code>	-36	I/O error
<code>fnOpnErr</code>	-38	File not open
<code>fnfErr</code>	-43	File or directory does not exist
<code>flckdErr</code>	-45	File is locked
<code>rfNumErr</code>	-51	Bad reference number
<code>dirNFErr</code>	-120	Directory not found or incomplete pathname
<code>afpAccessDenied</code>	-5000	User does not have the correct access to the file

AGFileGetSelectorCount

Use the `AGFileGetSelectorCount` function to obtain the number of gestalt selectors for a guide file.

```
AGFileSelectorCountType AGFileGetDBSelectorCount
    (AGFileFSSpecType *fileSpec);
```

`fileSpec` A pointer to the file system specification record for the guide file.

DESCRIPTION

The `AGFileGetSelectorCount` function returns the number of gestalt selectors for the guide file specified in the `fileSpec` parameter. This number corresponds to the number of <Gestalt> commands in the guide file's source files. A guide file can have at most three gestalt selectors.

AGFileGetSelector

Use the `AGFileGetSelector` function to access a guide file's gestalt selector and its associated value.

```
OSErr AGFileGetSelector
    (AGFileFSSpecType *fileSpec,
     AGFileSelectorIndexType selectorNumber,
     AGFileSelectorType *selector,
     AGFileSelectorValueType *value);
```

fileSpec A pointer to the file system specification record for the guide file.

selectorNumber

A selector index number. To access the first gestalt selector, specify 1; to access the second gestalt selector, specify 2; and to access the third gestalt selector, specify 3.

selector The `AGFileGetSelector` function returns, through this parameter, a four-character value gestalt selector for the gestalt selector specified by the `selectorNumber` parameter.

value The `AGFileGetSelector` function returns, through this parameter, a long integer associated with the gestalt selector returned in the `selector` parameter.

DESCRIPTION

The `AGFileGetSelector` function returns, in its `selector` and `value` parameters, the gestalt selector and its required value for the guide file specified in the `fileSpec` parameter. The gestalt selector and value correspond to those specified by a <Gestalt> command in the guide file's source files.

SEE ALSO

For a complete list of Gestalt selector codes, see the chapter "Gestalt Manager" in *Inside Macintosh: Operating System Utilities*.

AGFileIsMixin

Use the `AGFileIsMixin` function to determine if a guide file is a Mixin guide file, that is, whether it is used to add or modify content in a main guide file.

```
Boolean AGFileIsMixin
        (AGFileFSSpecType *fileSpec,
         AGFileDBMenuNamePtr menuItemNameStr);
```

fileSpec A pointer to the file specification record for the guide file.

menuItemNameStr

If the file is a mixin, `AGFileIsMixin` returns, through this parameter, the name of the main guide file associated with it.

DESCRIPTION

The `AGFileIsMixin` function returns `TRUE` if the guide file (specified in the `fileSpec` parameter) is a mixin and `FALSE` if it isn't.

AGFileGetMixinMatchSelector

Use the `AGFileGetMixinMatchSelector` function to determine if a Mixin guide file can be mixed in with a main guide file.

```
OSErr AGFileGetMixinMatchSelector
        (AGFileFSSpecType *fileSpec,
         OSType *mixinMatchSelector);
```

fileSpec A pointer to the file system specification record for the guide file.

mixinMatchSelector

The `AGFileGetMixinMatchSelector` function returns, through this parameter, a four-character value that usually corresponds to the signature of the application associated with this guide file.

DESCRIPTION

The `AGFileGetMixinMatchSelector` function returns, through its `mixinMatchSelector` parameter, a four-character value corresponding to the value specified by a `<Mixin Match>` command in the guide file's source files. A Mixin guide file can mix in with a main guide file if their four-character values (specified by `<Mixin Match>` commands) match.

In addition, if the value returned in the `mixinMatchSelector` parameter is `'****'`, the Mixin guide file can mix with any main guide file.

RESULT CODES

<code>noErr</code>	0	No error
<code>nsvErr</code>	-35	Volume doesn't exist
<code>ioErr</code>	-36	I/O error
<code>fnOpnErr</code>	-38	File not open
<code>fnfErr</code>	-43	File or directory does not exist
<code>fLckdErr</code>	-45	File is locked
<code>rfNumErr</code>	-51	Bad reference number
<code>dirNFErr</code>	-120	Directory not found or incomplete pathname
<code>afpAccessDenied</code>	-5000	User does not have the correct access to the file

Installing and Removing Coachmark Handlers

This section describes how to install and remove coachmark handlers using two functions, `AGInstallCoachHandler` and `AGRemoveCoachHandler`.

AGInstallCoachHandler

Use the `AGInstallCoachHandler` function to install a coachmark handler.

```
OSErr AGInstallCoachHandler
    (CoachReplyProcPtr CoachReplyProc,
     long refCon,
     AGCoachRefNum *resultRefNum);
```

CoachReplyProc

A pointer to your coachmark handler function. This application-defined function provides the location of the object to coachmark.

refCon An optional reference constant that your application can provide. Apple Guide passes this as a parameter to your coachmark handler.

resultRefNum

A reference number. The `AGInstallCoachHandler` function returns, through this parameter, a number that refers to the coachmark handler.

DESCRIPTION

The `AGInstallCoachHandler` function installs the coachmark handler and returns in the `resultRefNum` parameter a reference to it. Once a coachmark handler is installed, Apple Guide calls it in response to any panels that use an object coach. (The object coach must be one that specifies your application as a target in the <Define Object Coach> command.)

SPECIAL CONSIDERATIONS

Install only one coachmark handler for your application.

Your application should always remove any coachmark handler (using `AGRemoveCoachHandler`) it has installed before quitting.

RESULT CODES

<code>noErr</code>	0	No error
<code>kAGErrCannotInitCoach</code>	-2952	Unable to initialize coach handler

SEE ALSO

For information on writing a coachmark handler, see “Providing Object Locations for Coachmarks” on page 9-38. For information on the <Define Object Coach> command, see the chapter “Guide Script Command Reference.”

AGRemoveCoachHandler

Use the `AGRemoveCoachHandler` function to remove an installed coachmark handler.

```
OSErr AGRemoveCoachHandler
        (AGCoachRefNum *theRefNum);
```

`theRefNum`

A pointer to the reference number of the coachmark handler.

DESCRIPTION

The `AGRemoveCoachHandler` function removes the coachmark handler, specified by the `theRefNum` parameter.

RESULT CODES

<code>noErr</code>	0	No error
--------------------	---	----------

Installing and Removing Context Check Handlers

This section describes how to install and remove context check handlers using two functions, `AGInstallContextHandler` and `AGRemoveContextHandler`.

AGInstallContextHandler

Use the `AGInstallContextHandler` function to install a context check handler.

```
OSErr AGInstallContextHandler
        (ContextReplyProcPtr ContextReplyProc,
         AEEventID eventID, long refCon,
         AGContextRefNum *resultRefNum);
```

ContextReplyProc

A reference to the function called by Apple Guide in response to a context check specified in a guide file.

eventID A four-character value that should match the value of the *codeResSpec* parameter in the <Define Context Check> command for this context check.

refCon An optional reference constant that your application can provide. Apple Guide passes this as a parameter to your context check handler.

resultRefNum

A reference number. The `AGInstallContextHandler` function returns, through this parameter, a number that refers to the context handler.

DESCRIPTION

The `AGInstallContextHandler` function installs the context handler specified in the `ContextReplyProc` parameter and returns in the `resultRefNum` parameter a reference to it. Once a context check handler is installed, Apple Guide calls it in response to any context checks that specify your application as a target in the <Define Context Check> command. Apple Guide calls the context check handler only if the value specified in the command's *codeResSpec* parameter matches the value specified in the *eventID* parameter.

SPECIAL CONSIDERATIONS

Before quitting, your application should always remove any context check handlers (using `AGRemoveContextHandler`) that it has installed.

RESULT CODES.

<code>noErr</code>	0	No error
<code>kAGErrCannotInitContext</code>	-2953	Unable to initialize context handler
<code>kAGErrMissingAppInfoHdl</code>	-2958	No application information handler
<code>kAGErrMissingContextObject</code>	-2959	No context object

SEE ALSO

For information on writing a context check handler, see “Responding to Context Checks” on page 9-40. For information on the <Define Context Check> command, see the chapter “Guide Script Command Reference.”

AGRemoveContextHandler

Use the `AGRemoveContextHandler` function to remove an installed context handler.

```
OSErr AGRemoveContextHandler
      (AGContextRefNum *resultRefNum);
```

`resultRefNum`

A pointer to the reference number for the context handler.

DESCRIPTION

The `AGRemoveContextHandler` function removes the context handler specified by the `resultRefNum` parameter.

RESULT CODES

`noErr` 0 No error

Application-Defined Routines

This section describes two routines, a coachmark handler and context check handler, that you can provide for your application.

Providing Object Locations for Coachmarks

This section gives information on how to provide a coachmark handler function. Apple Guide can automatically draw coachmarks for menus, items in dialog boxes, and certain parts of a window, without assistance from your application. For those objects that require your application to explicitly specify the coordinates of a rectangle marking the coachmark's location, you can provide a coachmark handler function to do so.

You define coachmarks for these types of objects using the `<Define Object Coach>` command. You associate an object coach with a particular panel using the `<Coach Mark>` command.

When Apple Guide opens a panel that includes a `<Coach Mark>` command naming a defined object coach and that specifies your application as the target application, Apple Guide requests that your application return a rectangle for the named object. (You specify the name of the object to mark and your application's signature as parameters to the `<Define Object Coach>` command.) Once your application returns a rectangle for the object, Apple Guide draws the coachmark.

Your application handles object-location coachmark requests from Apple Guide by installing a coachmark handler function. Apple Guide calls your coachmark handler whenever it needs to coachmark an object in your application that is specified by a <Define Object Coach> command.

MyCoachReplyProc

A coachmark handler function should find and return the rectangle for a named object. Here is the syntax of a coachmark handler function:

```
pascal OSErr MyCoachReplyProc
                (Rect *pRect, Ptr name, long refCon);
```

pRect	An address through which your coachmark handler should return the rectangle of the object to coachmark, in global coordinates.
name	The name of the object to coachmark. The name is stored as a NULL-terminated string.
refCon	A reference constant, which your application sets when it installs the coachmark handler. Your coachmark handler can use this reference constant for any purpose.

DESCRIPTION

A coachmark reply function should return, through the **pRect** parameter, the global coordinates of the object to coachmark.

RESULT CODES

The **MyCoachReplyProc** function should return **noErr** if successful or an appropriate result code otherwise.

SEE ALSO

For information on installing a coachmark handler, see page 9-33.

Responding to Context Checks

This section gives information on how to provide a context check handler.

You can implement context checks using one of two methods. You can provide the code for a context check in an external module that you include as a resource in your guide file (this is the more typical method of providing a context check handler). Alternatively, you can provide the code in your application and make your context check handler available to Apple Guide using the `AGInstallContextHandler` function. Regardless of which method you use, a context check handler follows a specific syntax, as described next.

MyContextReplyProc

A context check handler checks the condition associated with a context check and returns the result.

Here is the syntax of a context check handler function:

```
pascal OSErr MyContextReplyProc
                (Ptr pInputData, Size inputDataSize,
                 Ptr *ppOutputData,
                 Size *pOutputDataSize,
                 AGAppInfoHdl hAppInfo);
```

pInputData

A pointer to the input data. Apple Guide places any parameters specified by the context check in the data area pointed to by this parameter. Apple Guide concatenates the parameters in a byte array (word-aligned); your context check handler function retrieves the data through this parameter. You can cast `pInputData` to a pointer to a data structure that describes the parameters specified by your <Define Context Check> command.

inputDataSize

The size of the input data in bytes.

ppOutputData

Your context check handler function should return, through this parameter, a pointer to a short integer. The short integer should contain the result of the context check, 1 for true and 0 for false.

pOutputDataSize

Your context check handler function should return, through this parameter, the size of the output data in bytes, usually `sizeof(short)`.

hAppInfo

A handle to a structure of type `AGAppInfo`.

```
typedef struct AGAppInfo
{
    AEEEventID    eventId;        /*event ID*/
    long          refCon;         /*app's refCon*/
    void          *contextObj;    /*private*/
}
```

Apple Guide places the event ID and reference constant (specified in the call to `AGInstallContextHandler`) into the `eventId` and `refCon` fields of this structure. The `eventId` field corresponds to the value of the *codeResSpec* parameter in the <Define Context Check> command for this context check. You can use the `refCon` field for any purpose. The `contextObj` field is used by Apple Guide and your application should not use or change this field.

DESCRIPTION

Your `MyContextReplyProc` function should perform the context check and return the result of the context check through the `ppOutputData` parameter.

EXAMPLES

Here's an example of a context check installed by the SurfWriter application. The SurfWriter application defines these context checks in its guide file using the <Define Context Check> command:

```
<DCC> "SWDictionaryIsOpen", 'SWdd', 'WAVE', short:1, LPSTRING
<DCC> "SWThesaurusIsOpen", 'SWdd', 'WAVE', short:2, LPSTRING
```


Here's how SurfWriter Guide uses one of these context checks to dynamically adjust the display of its panels:

```
<Panel> "Dictionary:intro"
<Skip If> SWDictionaryIsOpen("Standard")
  <Panel> "Dictionary:open"
<Panel> "Dictionary:lookupWord"
```

The SurfWriter application installs its context check handler using the `AGInstallContextHandler` function.

```
/*gEventID = 'SWdd'*/
myErr = AGInstallContextHandler(SWIsOpenContextCheck,
                                gEventID,
                                &gRefCon, &gResultRefNum);
```

This is how it defines its context check handler function. (A context check in an external module receives the same parameters, but it has a single entry point, `main`.)

```
pascal OSErr SWIsOpenContextCheck
    (Ptr inputDataPtr, Size inputDataSize,
     Ptr *ppOutputData, Size *pOutputSize,
     AGAppInfoHdl hAppInfo)
{
    Boolean isOpen = false;
    OSErr myErr = noErr;
    /*app-defined structure contains two fields, a short and a string*/
    MyContextCheckParams myCCParams;

    myCCParams = *((MyContextCheckParams *) inputDataPtr);

    switch (myCCParams.selector)
    {
    case 1: /*check whether a specified dictionary is open*/
        isOpen = MyCheckDictionary(myCCParams);
        break;
    case 2: /*check whether a specified thesaurus is open*/
```

```

    isOpen = MyCheckThesaurus(myCCParams);
    break;
default:
    break;
}
/*return result of context check (true or false) in the */
/* ppOutputData parameter*/
myErr = MySetContextResult(&isOpen, sizeof(Boolean),
                          ppOutputData, pOutputSize);
/*if error occurs, return appropriate function result */
/* indicating error*/
return(myErr);
}

OSErr MySetContextResult(void *theData, Size theSize,
                        Ptr *outMessage, Size *outSize)
{
    Ptr    p;

    /*the context check routine must return a pointer to a short */
    /* in the ppOutputData parameter */
    /* (Apple Guide will dispose of the pointer on return)*/
    /*(1 = true, 0 = false) indicates the result of the context check*/
    /*The routine must also return the size of the ppOutputData */
    /* in the pOutputSize parameter*/
    if (p = NewPtr(theSize))
    {
        BlockMove(theData, p, theSize);
        *outSize = theSize;
        *outMessage = p;
        return(noErr);
    }
    else
        return(MemError());
}

```

CHAPTER 9

Apple Guide API

RESULT CODES

The `MyContextReplyProc` function should return `noErr` if successful, or an appropriate result code otherwise.

Summary of Apple Guide API

Constants

```
enum {
    gestaltHelpMtrAttr      'help'    /*Gestalt selector for Help Mgr */
                                   /* and Apple Guide*/
    gestaltAppleGuidePresent= 31,    /*Apple Guide API is available*/
    gestaltAppleGuideIsDebug= 30    /*Apple Guide Debug extension */
                                   /* is installed*/
};

enum {
    /*guide file active list types*/
    kAGViewFullHowdy        = 1,    /*full howdy*/
    kAGViewTopicAreas       = 2,    /*Topic Area*/
    kAGViewIndex            = 3,    /*Index*/
    kAGViewLookFor          = 4,    /*Look For*/
    kAGViewSingleHowdy      = 5,    /*Single List howdy*/
    kAGViewSingleTopics     = 6     /*Single List topics*/
};

enum AGDBTypeBit
{
    /*guide file types returned by AGGetAvailableDBTypes*/
    kAGDBBitAny              = 0x00000001,/*one or more guide */
                                   /* files are present*/
    kAGDBTypeBitHelp        = 0x00000002,/*Help guide file*/
    kAGDBTypeBitTutorial    /*Tutorial guide */
    kAGDBTypeBitTutorial     = 0x00000004,/* file*/
    kAGDBTypeBitShortcuts   /*Shortcuts guide */
    kAGDBTypeBitShortcuts    = 0x00000008,/* file*/
}
```

Apple Guide API

```

    kAGDBTypeBitAbout      = 0x00000010, /*About guide file*/
    kAGDBTypeBitOther      = 0x00000080  /*Other guide file*/
};

enum {
    /* Apple Guide Apple events*/
    kAGEventDoCoach        = 'doco', /*coachmark event*/
    kAGEventDoHuh          = 'dhuh', /*huh event*/
    kAGEventGoNext         = 'gonp', /*go next event*/
    kAGEventGoPrev         = 'gopp', /*go previous event*/
    kAGEventHidePanel      = 'pahi', /*hide panel event*/
    kAGEventReturnBack     = 'gobk', /*return back event*/
    kAGEventShowPanel      = 'pash', /*show panel event*/
    kAGEventTogglePanel    = 'patg'  /*toggle panel event*/
};

enum {
    kAGFrontDatabase       = 1    /*refers to active guide file*/
};

enum {
    /*values returned by AGGetFrontWindowKind*/
    kAGNoWindow,           /*guide file isn't open*/
    kAGAccessWindow,       /*access window is showing*/
    kAGPresentationWindow  /*panel is showing*/
};

enum {
    /*values returned by AGGetStatus*/
    kAGIsNotRunning,       /*app portion of AG not in memory*/
    kAGIsSleeping,         /*app portion of AG in memory but */
                          /* no guide file is open*/
    kAGIsActive            /*app portion of AG in memory */
                          /* and a guide file is open*/
};

```

```
enum {
    /*guide file types, used by the functions AGFileGetIndDB */
    /* and AGFileGetDBCount, from AGFile*/
    kAGFileDBTypeAny          = 0,  /*any or all guide files*/
    kAGFileDBTypeHelp        = 1,  /*Help guide file*/
    kAGFileDBTypeTutorial    = 2,  /*Tutorial guide file*/
    kAGFileDBTypeShortcuts   = 3,  /*Shortcuts guide file*/
    kAGFileDBTypeAbout       = 4,  /*About guide file*/
    kAGFileDBTypeOther       = 8   /*Other guide file*/
};
```

Data Types

```
typedef unsigned long   UInt32;
typedef unsigned short  UInt16;
typedef signed   short  SInt16;

typedef UInt32  AGRefNum;
typedef UInt32  AGCoachRefNum;
typedef UInt32  AGContextRefNum;
typedef UInt16  AGStatus;
typedef UInt16  AGWindowKind;
typedef UInt32  AGEvent;

typedef SInt16  AGErr;

typedef struct AGAppInfo
{
    AEEEventID    eventId;          /*event ID*/
    long          refCon;           /*reference constant*/
    void          *contextObj;      /*private*/
}

typedef struct AGAppInfo, *AGAppInfoPtr, **AGAppInfoHdl;
```

```

typedef pascal OSErr (*CoachReplyProcPtr)(Rect *pRect, Ptr name,
                                           long refCon);

typedef pascal OSErr (*ContextReplyProcPtr)
    (Ptr pInputData, Size inputDataSize,
     Ptr *ppOutputData, Size *pOutputDataSize,
     AGAppInfoHdl hAppInfo);

/*typedefs from AGFile*/
typedef FSSpec          AGFileFSSpecType;
typedef short           AGFileSelectorCountType;
typedef short           AGFileSelectorIndexType;
typedef OSType          AGFileSelectorType;
typedef long            AGFileSelectorValueType;
typedef short           AGFileDBType;
typedef ConstStr63Param AGFileDBMenuNamePtr;
typedef short           AGFileDBScriptType;
typedef short           AGFileDBRegionType;
typedef short           AGFileMajorRevType;
typedef short           AGFileMinorRevType;
typedef short           AGFileCountType;

```

Functions

Starting Up Apple Guide

```

AGErr AGStart(void);
AGErr AGQuit(void);
AGStatus AGGetStatus(void);

```

Determining Which Guide Files Are Available

```

UInt32 AGGetAvailableDBTypes (void);

```

```

AGFileCountType  AGFileGetDBCount
                    (short vRefNum, long dirID,
                     AGFileDBType databaseType,
                     Boolean wantMixin);

OSErr AGFileGetIndDB (short vRefNum, long dirID,
                     AGFileDType databaseType,
                     Boolean wantMixin,
                     short dbIndex, FSSpecType *fileSpec);

```

Opening and Closing Guide Files

```

AGErr AGOpen (FSSpec *fileSpec, UInt32 flags,
              Handle mixinControl,
              ARefNum *resultRefNum);

AGErr AGOpenWithView (FSSpec *fileSpec, UInt32 flags,
                      Handle mixinControl,
                      short viewNum,
                      ARefNum *resultRefNum);

AGErr AGOpenWithSearch (FSSpec *fileSpec, UInt32 flags,
                        Handle mixinControl,
                        ConstStr255Param searchString,
                        ARefNum *resultRefNum);

AGErr AGOpenWithSequence (FSSpec *fileSpec, UInt32 flags,
                           Handle mixinControl,
                           short sequenceID,
                           ARefNum *resultRefNum);

AGErr AGClose (ARefNum *resultRefNum);

```

Working With Open Guide Files

```

Boolean AGIsDatabaseOpen (ARefNum refNum);

AGWindowKind AGGetFrontWindowKind
                    (ARefNum refNum);

AGErr AGGeneral (ARefNum refNum, AEvent theEvent);

AGErr AGGetFSSpec (ARefNum refNum, FSSpec *fileSpec);

```


Getting Information About Guide Files

```

OSError AGFileGetDBMenuName    (AGFileFSSpecType *fileSpec,
                                AGFileDBMenuNamePtr menuItemNameStr);

OSError AGFileGetHelpBalloonText
                                (AGFileFSSpecType *fileSpec,
                                Str255 helpMenuBalloonString);

OSError AGFileGetHelpMenuAppCreator
                                (AGFileFSSpecType *fileSpec,
                                OSType *helpMenuAppCreator);

OSError AGFileGetDBType        (AGFileFSSpecType *fileSpec,
                                AGFileDBType *databaseType);

OSError AGFileGetDBCOUNTRY      (AGFileFSSpecType *fileSpec,
                                AGFileDBScriptType *script,
                                AGFileDBRegionType *region);

OSError AGFileGetDBVersion      (AGFileFSSpecType *fileSpec,
                                AGFileMajorRevType *majorRev,
                                AGFileMinorRevType *minorRev);

AGFileSelectorCountType AGFileGetDBSelectorCount
                                (AGFileFSSpecType *fileSpec);

OSError AGFileGetSelector      (AGFileFSSpecType *fileSpec,
                                AGFileSelectorIndexType selectorNumber,
                                AGFileSelectorType *selector,
                                AGFileSelectorValueType *value);

Boolean AGFileIsMixin          (AGFileFSSpecType *fileSpec,
                                AGFileDBMenuNamePtr menuItemNameStr);

OSError AGFileGetMixinMatchSelector
                                (AGFileFSSpecType *fileSpec,
                                OSType *mixinMatchSelector);

```

Installing and Removing Coachmark Handlers

```

OSError AGInstallCoachHandler (CoachReplyProcPtr CoachReplyProc,
                                long refCon,
                                AGCoachRefNum *resultRefNum);

```

```
OSErr AGRemoveCoachHandler (AGCoachRefNum *theRefNum);
```

Installing and Removing Context Check Handlers

```
OSErr AGInstallContextHandler
    (ContextReplyProcPtr ContextReplyProc,
     AEEventID eventID, long refCon,
     AGContextRefNum *resultRefNum);
```

```
OSErr AGRemoveContextHandler (AGContextRefNum *resultRefNum);
```

Providing Object Locations for Coachmarks

```
pascal OSErr MyCoachReplyProc
    (Rect *pRect, Ptr name, long refCon);
```

Responding to Context Checks

```
pascal OSErr MyContextReplyProc
    (Ptr pInputData, Size inputDataSize,
     Ptr *ppOutputData,
     Size *pOutputDataSize,
     AGAppInfoHdl hAppInfo);
```

Result Codes

noErr	0	No error
dirNFErr	-12	Directory not found or incomplete pathname
nsvErr	-35	Volume doesn't exist
ioErr	-36	I/O error
fnOpnErr	-38	File not open
fnfErr	-43	File or directory does not exist
fLckdErr	-45	File is locked
rfNumErr	-51	Bad reference number
dirNFErr	-120	Directory not found or incomplete pathname
kAGErrCannotInitCoach	-2952	Unable to initialize coach handler

<code>kAGErrCannotInitContext</code>	-2953	Unable to initialize context handler
<code>kAGErrCannotOpenAliasFile</code>	-2954	Unable to open alias
<code>kAGErrNoAliasResource</code>	-2955	Unable to open resource alias
<code>kAGErrDatabaseNotAvailable</code>	-2956	Guide file is not available
<code>kAGErrDatabaseNotOpen</code>	-2957	Guide file is not open
<code>kAGErrMissingAppInfoHdl</code>	-2958	No application information handler
<code>kAGErrMissingContextObject</code>	-2959	No context object
<code>kAGErrInvalidRefNum</code>	-2960	The guide file was opened by another application
<code>kAGErrDatabaseOpen</code>	-2961	No open guide file
<code>kAGErrInsufficientMemory</code>	-2962	Not enough memory
<code>afpAccessDenied</code>	-5000	User does not have the correct access to the file

PART FOUR

Scripting Guide Files

Guide Script Command Reference

Contents

Guide Script Command Syntax	10-5
Guide Script Command Descriptions	10-8
Specifying Startup Information	10-8
<App Creator>	10-8
<Gestalt>	10-10
<Version>	10-11
<World Script>	10-13
<Help Menu>	10-14
<Balloon Menu Text>	10-16
<Comment>	10-17
<Include>	10-18
<Mixin>	10-19
<Mixin Match>	10-20
Specifying the Startup Window	10-21
<Startup Window>	10-21
<Howdy>	10-24
<App Logo>	10-25
<App Text>	10-27
Specifying Default Settings	10-28
<Max Height>	10-28
<Min Height>	10-29
<Default Format>	10-30
<Default Nav Button Set>	10-32
<Allow Prompts>	10-34
<Default Prompt Set>	10-35
<Define Prompt Set>	10-37

Creating Sequences	10-39
<Define Sequence>	10-39
<Sequence Prompt Set>	10-42
<Seq Nav Button Set>	10-43
<Panel>	10-45
<Insert Sequence>	10-46
<Jump Sequence>	10-47
<Launch New Sequence>	10-49
<Build Sequence>	10-50
<End Sequence>	10-51
Creating Panels	10-52
<Define Panel>	10-52
<Panel Prompt>	10-55
<End Panel>	10-56
Creating Buttons	10-57
<Standard Button>	10-57
<3D Button>	10-60
<Radio Button>	10-64
<Radio Button Launch New Seq>	10-66
<Checkbox>	10-69
<Define Nav Button>	10-71
<Dimmable Button Data>	10-78
<Define Nav Button Set>	10-80
Defining and Using Text Blocks	10-82
<Define Text Block>	10-82
<End Text Block>	10-83
Formatting Text and Objects in a Panel	10-84
<Define Format>	10-85
<Define Transparent Format>	10-91
<Format>	10-93
Specifying Pictures and Movies	10-94
<PICT>	10-95
<QuickTime>	10-98
Importing Resources	10-100
<Resource>	10-101
<Starting Res Number>	10-103

Creating CoachMarks	10-105
<Define Menu Coach>	10-105
<Define Item Coach>	10-108
<Define Object Coach>	10-111
<Define Window Coach>	10-113
<Define AppleScript Coach>	10-116
<Coach Mark>	10-118
Creating Hot Items	10-119
<Hot Object>	10-119
<Hot Rectangle>	10-120
<Hot Text>	10-122
Defining Topic Areas	10-124
<Topic Areas Instruction>	10-124
<Topic Area>	10-125
Defining Index Terms	10-127
<Index Instruction>	10-127
<Index>	10-128
<Sorting>	10-130
<Index Sorting>	10-132
Defining Topics for Topic Areas and Index Terms	10-133
<Topics Instruction>	10-134
<Header>	10-135
<Topic>	10-137
Specifying “Look For” Help	10-140
<Look For Instruction>	10-141
<Look For String>	10-143
<Look For Search Btn Instruction>	10-143
<Look For Results Instruction>	10-144
<Ignore>	10-145
<Exception>	10-147
<Synonym>	10-149
Specifying Conditional Execution	10-152
<If>	10-153
<Else>	10-156
<End If>	10-158
<Skip If>	10-160
<Make Sure>	10-162

<Start Making Sure>	10-168
<End Making Sure>	10-171
Defining and Using Context Checks	10-172
<Define Context Check>	10-172
checkBoxState	10-175
radioButtonState	10-176
Specifying Events	10-177
<Define Event>	10-178
<Define Event List>	10-181
<On Panel Create>	10-183
<On Panel Destroy>	10-184
<On Panel Show>	10-185
<On Panel Hide>	10-187
Built-in Event Functions	10-188
Working With Mixin Guide Files	10-190
<Replace Sequence>	10-190
<Insert Topic Area Header>	10-192
<Insert Topic Area Topic>	10-193
<Insert Index Header>	10-195
<Insert Index Topic>	10-196
<Delete Topic Area>	10-198
<Delete Topic Area Header>	10-199
<Delete Topic Area Topic>	10-200
<Delete Index>	10-201
<Delete Index Header>	10-202
<Delete Index Topic>	10-203

This chapter describes Guide Script commands. You can use these commands to specify the content of your guide file. For example, using these commands you can specify startup information, create panels and panel objects (such as buttons, text, and pictures), align panel objects on a panel, set prompt values, and define topic area information and index information. Each command description in this chapter provides the command syntax, a description of the command parameters, a description of the command itself, and examples of the use of the command.

Some Guide Script commands can be referenced using an abbreviated spelling in addition to the command's full name (<QT> for <QuickTime>, for example). For a list of abbreviations for all commands, see "Appendix A: Guide Script Command Abbreviations."

This chapter first gives a brief overview of the syntax governing all commands, then presents the command descriptions. A summary list of all commands and their parameters is given in "Appendix B: Guide Script Commands and Parameters Quick Reference."

Guide Script Command Syntax

A Guide Script command line begins with a command, comment, or text that is part of a text block or panel definition, and is terminated by a carriage return. Any number of blank spaces or tabs can precede a command or comment.

Guide Script commands are named keywords that are enclosed in angle brackets (< >). Command keywords are case-insensitive, although headline-style capitalization is used in this book. Most Guide Script commands are followed by a series of parameters, separated by commas. Parameters must be included on the same line as the Guide Script command; however, with some word processors you can use a soft return (Option-Return) to break parameters across lines when necessary. In this chapter, the character → is used to indicate a line that does not contain a hard return; you must either wrap these lines or use a soft return, depending on your word processor. You can also include any number of spaces or tabs between parameters to make command lines easier to read.

This chapter describes Guide Script commands in this format:

```
<Command Name> parameter1, parameter2 [ , optionalParameter1]
```

Here's a specific example:

```
<Help Menu> itemString, helpType [, helpCmdKey]
```

In this example, the command <Help Menu> indicates the Guide Script command. (You don't have to use this sequence of uppercase and lowercase characters. For example, you can use <help menu> if you prefer.) Parameters are shown in italicized words following the command name. Brackets ([]) are used to show optional parameters. In this example, *itemString* and *helpType* are required parameters, and *helpCmdKey* is an optional parameter.

You can omit optional parameters. If you do so and the optional parameter is followed by another parameter that you do specify, then you must include a comma for each parameter that you omit. For example:

```
<Define Item Coach> coachMarkName [, targetApp] [, coachStyle]  
                    [, targetWindow] , targetItem  
                    [, itemRectangle]
```

For the <Define Item Coach> command just shown, you must specify the *coachMarkName* parameter and *targetItem* parameter, but all other parameters are optional. To specify the *coachMarkName* and *coachStyle* and *targetItem* parameters (omitting the *targetApp*, *targetWindow*, and *itemRectangle* parameters), you could specify the command in this format:

```
<Define Item Coach> "coachName", , REDCIRCLE,,DialogID(2)
```

Parameters are one of the following types:

- **Strings.** Any string of characters enclosed by straight double quotes (" "). Curly double quotes (" ") are not valid as string delimiters; they are considered part of the text string if used. To include a line feed in a string, use the carriage return character between the string delimiters. Do not specify an empty string for a parameter; instead, specify at least one blank space in the string, such as " ".
- **Constants.** Values represented by constant names that are defined by Guide Maker for a particular parameter. For example, LEFT, CENTER, RIGHT, TRUE, or FALSE. All Guide Script constants are defined as single words (no spaces), and are case insensitive (for example, LEFT, Left, and left are all valid constants).

- **Four-character values** (also sometimes called `OSType` or `ResType`). A four-character value must be enclosed by straight single quotes (`'`). For example, `'MACS'`, `'WAVE'`.
- **Numbers**. Short, long, or unsigned integers. Each parameter description gives information on the specific type. You can specify numbers using either decimal or hexadecimal form (precede hexadecimal numbers by `0x`).
- **Structures**. Groups of related values that are linked together according to a defined Guide Script convention. For example, you can specify a point, rectangle, column location, or RGB color using these conventions: `Point(x, y)`; `RECT(top, left, bottom, right)`; `Column(top, left, right)`; and `RGBColor(integer, integer, integer)`. Note that no space is allowed between the name and the first parenthesis.
- **Event functions**. You can specify event functions in conjunction with the `<Standard Button>`, `<3D Button>`, `<Define Nav Button>`, `<On Panel Create>`, `<On Panel Destroy>`, `<On Panel Show>`, and `<On Panel Hide>` commands. Guide Maker provides the following built-in event functions: `DoScript`, `GoPanel`, `LaunchNewSequence`, `LaunchNewSequenceNewWindow`, `PlaySound`, `QuitTopicOops`, and `StartTopicOops`. In addition, you can create your own event functions using the `<Define Event>` command. When you specify an event function, you must also specify within parentheses all parameters required by that event function.
- **Condition functions**. You can specify condition functions in conjunction with the `<If>`, `<Skip If>`, `<Make Sure>`, and `<Start Making Sure>` commands. You define condition functions using the `<Define Context Check>` command.

You use the `#` character to indicate that the text that follows, up to the carriage return, should be treated as a comment. You can start a comment at the beginning or at the end of a command line, for example:

```
#here's a comment
<App Text> "SurfWriter Guide"    #here's another comment
```

To use the `"#"` character as the first character on a line of panel text, use `"##"`. To use the `"<"` character as the first character on a line of panel text use `"<<"`.

Guide Script Command Descriptions

The rest of this chapter presents the command descriptions for Guide Script commands, arranged by function.

Specifying Startup Information

This section describes the commands that determine whether Apple Guide lists your guide file in an application's Help menu and whether Apple Guide considers your guide file a main or mixin guide file. It also describes other commands that provide basic information about your guide file, such as your guide file's menu item text and command key in the Help menu.

<App Creator>

You can use the <App Creator> command to specify the application that is associated with this guide file.

<App Creator> *creator*

creator A four-character value that specifies the creator of the guide file. This value should correspond to the signature of the application associated with this guide file.

DESCRIPTION

When the user launches an application, Apple Guide searches for any guide files in the same folder as the application file. Apple Guide determines which guide files (if any) to put in the application's Help menu according to the criteria summarized next.

If a single guide file exists in the same folder and does not contain an <App Creator> command, Apple Guide includes that guide file as an item in the Help menu. If multiple guide files, each of differing types, exist in the same

folder, Apple Guide includes each guide file in the Help menu (except for any guide file whose *creator* parameter does not match the application's signature).

If multiple guide files of type **OTHER** exist in the same folder, Apple Guide includes each **OTHER** guide file in the Help menu (except for any **OTHER** guide file whose *creator* parameter does not match the application's signature).

For guide files of type **ABOUT**, **SHORTCUTS**, **HELP**, or **TUTORIAL**, Apple Guide includes at most one guide file of the same type, even if multiple guide files of the same type exist. If multiple guide files of one of these types exist, Apple Guide first determines which guide file's *creator* parameter (specified in the `<App Creator>` command) matches the application's signature (it also checks gestalt conditions). If more than one guide file of the same type exists in the same folder and meets these requirements, Apple Guide includes as an item in the Help menu the guide file whose filename appears first alphabetically.

Apple Guide also checks any conditions specified by `<Gestalt>` commands in the guide file before adding it to an application's Help menu. Apple Guide does not add a guide file to the Help menu if the conditions specified by the guide file's `<Gestalt>` commands are not met.

SPECIAL CONSIDERATIONS

The `<App Creator>` command should appear once at most in your source files for a specific guide file. If you omit this command, Apple Guide includes your guide file as an item in the Help menu only if it is in the same folder as the launched application and only if it is the only guide file of a specific type (except for **OTHER**) in that folder.

EXAMPLES

```
#this guide file is associated with the SurfWriter app
<App Creator> 'WAVE'      #SurfWriter's signature is 'WAVE'
```

<Gestalt>

You can use the <Gestalt> command to define the conditions Apple Guide uses to determine whether your guide file can run and therefore whether your guide file should be added to the Help menu.

<Gestalt> *selector, requiredValue*

selector A four-character code representing a gestalt selector code.

requiredValue A value that indicates whether your guide file should run. If this value matches the value returned by the `Gestalt` function in its `response` parameter, Apple Guide includes your guide file as an item in the Help menu. If this value does not match and if other <Gestalt> commands also do not match, Apple Guide does not include your guide file as an item in the Help menu.

DESCRIPTION

Before adding your guide file to the Help menu, Apple Guide calls the `Gestalt` function with the selector code specified by a <Gestalt> command and checks the `response` parameter against the value specified by the *requiredValue* parameter. If these values match (and if the creator specified by the <App Creator> command also matches), Apple Guide adds your guide file to the Help menu. You can specify additional <Gestalt> commands. If you do, Apple Guide performs an OR operation on any conditions specified by <Gestalt> commands in your guide file. If any one of the specified conditions is true, Apple Guide adds your guide file to the menu. If none of the conditions are true, Apple Guide does not add your guide file to the Help menu.

SPECIAL CONSIDERATIONS

You can specify up to three <Gestalt> commands in your help source files.

EXAMPLES

```
#specify conditions that must be true to run this guide
# file; if any of these are true, then add this guide file
```

```
#the Gestalt selector 'snd ' returns attributes related to
# sound and a value of 16 indicates a sound input device
# is present
<Gestalt> 'snd ', 16
#specify conditions related to sound input & speech mgr
<Gestalt> 'snd ', 16
<Gestalt> 'ttsc', 0
```

SEE ALSO

For a complete list of Gestalt selector codes, see the chapter “Gestalt Manager” in *Inside Macintosh: Operating System Utilities*.

<Version>

You can use the <Version> command to specify values for the version resources of your guide file.

```
<Version> longVers1BottomOfGetInfo, shortVers1ForFinderListView
          [, longVers2TopOfGetInfo]
```

longVers1BottomOfGetInfo

A string containing the information for your guide file’s long version message in its version resource with resource ID 1. When the user selects your guide file and then chooses Get Info from the File menu of the Finder, the Finder displays information from your long version resource in the version field at the bottom of the information window.

You typically provide the file version number of your guide file and your company’s copyright information in this parameter.

shortVers1ForFinderListView

A string containing the information for your guide file’s version number for its version resource with resource ID 1. The Finder displays information from this short version resource when it shows files in list view.

longVers2TopOfGetInfo

A string containing the information for your guide file's long version message in its version resource with resource ID 2. The Finder displays, near the top of the information window (beneath your guide file's name and next to its icon), information from this version resource. You typically provide the version number of your guide file in this parameter, or, if your guide file is part of a set of guide files that you distribute, you provide the version number of the superset of files that it belongs to in this parameter. This parameter is optional.

DESCRIPTION

The <Version> command sets the guide file's version resources to the specified values. Providing this information helps the user distinguish between different versions of your guide file.

SPECIAL CONSIDERATIONS

The <Version> command should appear no more than once in your source files.

EXAMPLES

```
<Version> "1.0 (US), © Apple Computer, Inc. 1994", "1.0",~
"(for SurfWriter 3.0)"
```

SEE ALSO

For information on the version resources, see the chapter "Finder Interface" in *Inside Macintosh: Macintosh Toolbox Essentials*.

You can also specify script and region information for your guide file, using the <World Script> command, described next.

<World Script>

You can use the <World Script> command to specify script and region information for your guide file.

<World Script> *scriptCode*, *regionCode*

<i>scriptCode</i>	An integer specifying the script code for your guide file. This value identifies the script system your guide file is intended for.
<i>regionCode</i>	An integer specifying the region code for your guide file. This value identifies the region your guide file is intended for.

DESCRIPTION

The <World Script> command sets the guide file's script code and region code to the values specified by the *scriptCode* and *regionCode* parameters. Apple Guide displays only guide files whose script codes match the script system currently in use.

SPECIAL CONSIDERATIONS

The <World Script> command should appear no more than once in your source files. If you omit this command, Guide Maker assigns your guide file a script code of 0 (Roman) and a region code of 0 (U.S.).

EXAMPLES

```
#specify Roman script system, U.S. region  
<World Script> 0, 0
```

SEE ALSO

See the chapter "Script Manager" in *Inside Macintosh: Text* for a complete list of script codes and region codes.

<Help Menu>

You can use the <Help Menu> command to define the item text that Apple Guide displays in your application's Help menu when your application's guide file is available.

<Help Menu> *itemString*, *helpType* [, *helpCmdKey*]

itemString A text string specifying the text of the menu item as it should appear in the application's Help menu. Do not specify any of these characters in your text string: ";", "^", "!", "<", "/", "-", "(", ")", ",", ":", or "&".

helpType A value indicating the type of help that this guide file provides or a value indicating the position of the menu item in the Help menu. You can specify any of these constants to indicate the help type:

ABOUT
TUTORIAL
HELP
SHORTCUTS
OTHER

If you specify a value other than these five constants, Apple Guide uses the specified value to determine the position of the menu item in the Help menu.

helpCmdKey A one-character string that identifies the Command-key equivalent that the user can use to invoke this guide file. This parameter is optional. However, note that Apple Guide always invokes a guide file of type **HELP** (if one is available) when the user presses Command-Shift-/ (which maps to Command-Shift-? on U.S. keyboards). For guide files of type **HELP**, you should always specify "?" in this parameter. You cannot assign a Command-key equivalent for guide files of type **ABOUT**.

DESCRIPTION

Apple Guide displays in the application's Help menu the string defined by the *itemString* parameter. (Apple Guide displays the string only if your guide file is

available to the application.) In addition, if you specify the *helpCmdKey* parameter, Apple Guide displays the Command-key symbol in the menu item followed by the character specified in the *helpCmdKey* parameter. Apple Guide invokes a guide file of type **HELP** when the user presses Command-Shift-? (regardless of the contents of the *helpCmdKey* parameter). If you provide a character in the *helpCmdKey* parameter for a guide file of type **HELP**, Apple Guide also invokes the guide file when the user presses Command and the specified character.

Four special positions are reserved near the top of the Help menu for guide files of type **ABOUT**, **TUTORIAL**, **HELP**, and **SHORTCUTS**. Only one of each of these types of guide files is displayed in the Help menu. However, you can have as many guide files of type **OTHER** as you choose. Apple Guide displays menu items for guide files of type **OTHER** in alphabetical order at the bottom of the Help menu.

SPECIAL CONSIDERATIONS

The **<Help Menu>** command should appear once at most in your source files for a specific guide file. If you omit this command when you compile a new guide file, Guide Maker assigns your guide file the type **HELP** and the name “Application Guide” as the menu item text and does not display the Command-key symbol or “?”. If you omit this command when recompiling a guide file, Guide Maker preserves any existing type or item text information.

EXAMPLES

```
#define menu item text for guide file of type HELP and
# specify the Command-key equivalent that appears in the
# Help menu for this guide file
<Help Menu> "SurfWriter Guide", HELP, "?"

#define menu item text for guide file of type TUTORIAL
<Help Menu> "SurfWriter Tutorial", TUTORIAL

#define menu item text for guide file of type OTHER
<Help Menu> "SurfWriter Write It For Me", OTHER
```

SEE ALSO

To specify balloon text for your guide file's menu item, use the <Balloon Menu Text> command, as described next.

<Balloon Menu Text>

You can use the <Balloon Menu Text> command to define the text for the help balloon that is displayed when Balloon Help is on and the cursor is in your guide file's menu item in your application's Help menu.

<Balloon Menu Text> *balloonText*

balloonText A text string specifying the text for the balloon associated with your guide file's menu item in the application's Help menu.

DESCRIPTION

When Balloon Help is on and the cursor is in a guide file's menu item in the application's Help menu, Apple Guide displays in a help balloon the string defined by the *balloonText* parameter.

SPECIAL CONSIDERATIONS

The <Balloon Menu Text> command should appear once at most in your source files for a specific guide file. If you omit this command when you compile a new guide file, Apple Guide will not display a help balloon for your guide file's menu item.

EXAMPLES

```
#define the balloon text for the guide file's menu item
# in the Help menu
<Balloon Menu Text> ~
"Select this item to get information about how to use
SurfWriter"
```

<Comment>

You can use the <Comment> command (or "#") to indicate a comment in your source files.

<Comment> or #

DESCRIPTION

When Guide Maker encounters a <Comment> command (or "#") it ignores the following text until the next carriage return. Comments specified with the <Comment> command must appear at the beginning of the line. Comments specified with the "#" character can begin a line or appear at the end of a command line. You cannot specify a comment in the middle of a command line.

Guide Maker interprets the "#" character as a comment if it appears as the first nonblank space on a line. To use the "#" character as the first character on a line of panel text, use "##". Guide Maker does not treat the "#" character as a special character if it appears within panel text.

EXAMPLES

```
<Comment> This is a comment
#here's a comment
    #this is also a comment
<App Text> "SurfWriter Guide" #here's another comment
<Define Panel> "Some panel" #a comment at end of command line
    Text goes here. #This is not a comment here, it's panel text.
    ##this is not a comment, it's panel text.
    <Standard Button> "My Button",      #illegal comment
                        LEFT, doAction() #legal comment
    <Standard Button> "My Button", LEFT, doAction() #ok comment
<End Panel>
```

<Include>

You can use the <Include> command to specify a source file that contains the Guide Script commands used to generate your guide file.

<Include> *sourceFileName*

sourceFileName

A filename of a file containing Guide Script commands for your guide file. This file must be located in the same folder as the file that specifies the <Include> command.

DESCRIPTION

You can specify the Guide Script commands that define your guide file in multiple source files. If you do this, you must create a build file and use the <Include> command to specify each source file. A build file is a file that contains only <Include> and <Resource> commands. When you compile your guide file, you specify your build file as the file to compile. All files referenced by <Include> commands must be located in the same folder as your build file.

SPECIAL CONSIDERATIONS

If you use an <Include> command, it must appear in a separate build file that contains only <Include> and <Resource> commands.

EXAMPLES

```
#build file for SurfWriter guide file
<Include> "Standard Setup"
<Include> "Panel definitions"
<Include> "Sequence definitions"
<Include> "Topic Areas definitions"
<Include> "Index definitions"
<Resource> "Standard Resources", ALL
```

SEE ALSO

For information on the <Resource> command, see page 10-101.

<Mixin>

You can use the <Mixin> command to specify whether the guide file generated by your source files is a Mixin guide file.

<Mixin> *symNameOrStartResNum*

symNameOrStartResNum

A text string supplying the name of the SYM file for the main guide file, or an integer that specifies the starting resource number.

DESCRIPTION

The <Mixin> command provides you with a method of modifying or adding content to an already existing guide file. If you do not specify a SYM file, you are limited to appending content to the end of a main guide file. If you specify a SYM file, you can use additional mixin-specific commands to change the behavior of the existing guide file. You also have the option of specifying a starting resource number instead of a SYM file. If you specify a starting resource number, then for resources that Guide Maker creates, it begins numbering them with the specified resource ID.

SPECIAL CONSIDERATIONS

If you use the <Mixin> command, it must be the first command in a source file. If you omit the <Mixin> command, Guide Maker assumes that the guide file is not a Mixin guide file. If you specify a starting resource number, the number must be between 2000 and 20,000.

EXAMPLES

```
#this is a Mixin guide file, and specify the main guide file .SYM file
<Mixin> "Master GF.SYM"
#this is a Mixin guide file, and specify a starting resource number
<Mixin> 8123
```

SEE ALSO

To modify or add content to an existing guide file, you can use the commands <Insert Topic Area Header>, <Insert Topic Area Topic>, <Insert Index Header>, and <Insert Index Topic>, as described on page 10-192, page 10-193, page 10-195, and page 10-196, respectively. You can also use other Guide Script commands to create sequences and panels for any topics that you add or modify.

<Mixin Match>

You can use the <Mixin Match> command to specify a creator for the Mixin guide file so that you can match Mixin guide files with a main guide file without conflict.

<Mixin Match> *matchingCreator*

matchingCreator

A four-character value that typically specifies the creator of the guide file. This usually corresponds to the signature of the application associated with this guide file. The special wild character sequence '****' indicates that this mixin file can match all guide files that meet all other required criteria (specified in the <App Creator> and <Gestalt> commands).

DESCRIPTION

The <Mixin Match> command provides you with a method of matching a Mixin guide file with one or more other guide files. You must specify the <Mixin Match> command in both the main guide file and the Mixin guide file.

If the values specified in each guide file's *matchingCreator* parameter match, and if both guide files satisfy other required criteria (as specified in the <App Creator> and <Gestalt> commands and if both are in the same folder), then the Mixin guide file is mixed in with the main guide file.

SPECIAL CONSIDERATIONS

The <Mixin Match> command should appear no more than once in your source files. If you omit the <Mixin Match> command, the Mixin guide file matches only those main guide files that also omit the <Mixin Match> command (and that satisfy the required criteria).

EXAMPLES

```
#specified in main guide file for SurfWriter Guide
<Mixin Match> 'WAVE'      #SurfWriter's signature
```

```
#specified in a Mixin guide file for SurfWriter
<Mixin Match> 'WAVE'      #SurfWriter's signature
```

Specifying the Startup Window

You define the appearance of your access window by using the commands described in this section.

<Startup Window>

You can use the <Startup Window> command to specify the type of window that Apple Guide should display when your guide file is first opened.

<Startup Window> *windowType, accessScreenOptions*

windowType A value that indicates whether the access window uses the full, single, or simple (presentation) access method.

You can use one of these constants in the *windowType* parameter:

FULL

SINGLE

PRESENTATION

The value you choose for this parameter determines the options that can be specified with the next parameter.

accessScreenOptions

A value that specifies the access screen options for the specified window type.

Full Access windows have options that you specify using these constants:

HOWDY

TOPICS

INDEX

LOOKFOR

Single List Access windows have options that you specify using these constants:

HOWDY

TOPICS

Simple Access windows (also called presentation windows) have no other options, although you must specify a string containing the presentation window's sequence name. For example:

"SurfWriter Command-Key Shortcuts Sequence"

DESCRIPTION

When a user opens a guide file, Apple Guide displays the access window of the indicated type and with the specified options, as defined by the guide file's <Startup Window> command.

If your guide file defines Topics, Index, and Look For help, use a Full Access window as your startup window. If you want to present the user with introductory or welcoming text (howdy text), then specify the constant **HOWDY**. If you specify howdy text, the user can dismiss the howdy text by clicking the Topics, Index, or Look For button. If you do not provide howdy text, then you can specify which help (Topics, Index, or Look For) should initially be active (that is, the text initially displayed in the left column) by using the corresponding constant in the second parameter. Often, you'll want the Topics

help to be initially active (so that the text specified by <Topic Area> commands appears in the left column).

If your guide file provides only Topics help, use a Single List Access window as your startup window. If you want to present the user with introductory or welcoming text (howdy text), then specify the constant `HOWDY`.

If your guide file does not provide Topics, Index, or Look For help, then specify the sequence name of your help content. Apple Guide displays this type of help in a Simple Access (presentation) window.

SPECIAL CONSIDERATIONS

The <Startup Window> command should appear once at most in your source files for a specific guide file. If you omit this command, Apple Guide displays a Full Access window with space for howdy text.

EXAMPLES

```
#specifies a Full Access window, no howdy text, and  
# with Topics initially active  
<Startup Window> FULL, TOPICS
```

```
#specifies a Single List Access window with howdy text  
<Startup Window> SINGLE, HOWDY
```

```
#specifies a presentation window and its sequence name  
<Startup Window> PRESENTATION, "SurfWriter ShortCuts  
Sequence"
```

SEE ALSO

The <Howdy> command is described next.

<Howdy>

You can use the <Howdy> command to specify the text block that Apple Guide displays for your guide file's howdy text.

<Howdy> *howdyTextBlockName*

howdyTextBlockName

A string identifying the name of a previously defined text block. (You use the <Define Text Block> and <End Text Block> commands to define and name a text block.)

DESCRIPTION

When a user opens a guide file, if you specified a Full or Single Access window and also specified the constant HOWDY (in the <Startup Window> command), Apple Guide displays the text defined by the <Howdy> command.

SPECIAL CONSIDERATIONS

The <Howdy> command should appear once at most in your source files for a specific guide file.

EXAMPLES

```
#specifies a Full Access window with howdy text
<Startup Window> FULL, HOWDY
```

```
#define howdy text block and give it a name
<Define Text Block> "Howdy Text"
    Welcome to personal help for SurfWriter.
```

```
To start, click Topics, Index, or Look For.
```

```
Topics shows general categories,
and Index lists key words.
```

```

    Look For lets you search for help
    according to key words you type.
<End Text Block>

```

```

#specify name of the text block that defines the howdy text
<Howdy> "Howdy Text"

```

SEE ALSO

For information on the <Define Text Block> and <End Text Block> commands, see page 10-82 and page 10-83, respectively.

<App Logo>

You can use the <App Logo> command to specify a file containing a picture that Apple Guide displays in the upper-left corner of a Full Access window.

```
<App Logo> colorLogo [ , B&WLogo ]
```

colorLogo The name of a file containing a color 'PICT' graphic; this graphic should help identify this guide file (for example, the application's logo) to the user.

B&WLogo The name of a file containing a black-and-white 'PICT' graphic; this graphic should help identify this guide file (for example, the application's logo) to the user. This parameter is optional.

DESCRIPTION

Guide Maker imports the 'PICT' resource from the file referenced by the parameter *colorLogo* and assigns it the resource ID 501, which is the resource ID reserved for this special graphic. If provided, Guide Maker also imports the 'PICT' resource from the file referenced by the parameter *B&WLogo* and assigns it the resource ID 502, which is the resource ID reserved for this special graphic. Note that the files containing the graphics should be located in the same folder as your help sources.

Guide Script Command Reference

As an alternative to using the <App Logo> command, you can assign your application's logos resource IDs of 501 and 502 and then import them using the <Resource> command.

The file Standard Resources is provided with Guide Maker. This file contains templates (two 'PICT' resources with IDs 501 and 502) that you can use to create your application logo picture. You should incorporate the lightbulb icon with your application logo and also use your application's name followed by the word Guide, as shown in Macintosh Guide and SurfWriter Guide.

Apple Guide displays the graphic defined by the <App Logo> command in the upper-left corner of a Full Access window. Note that you can include text in the graphic you provide for the <App Logo> command. The graphic should be at most 59 by 185 pixels. If both a color picture and black-and-white picture are provided, Apple Guide displays the color picture unless the monitor's bit depth is set to 4 bits or less.

SPECIAL CONSIDERATIONS

The <App Logo> command should appear once at most in your source files for a specific guide file. If you omit this command (and the <App Text> command), Apple Guide displays the default access window background in the upper-left corner of the Full Access window.

You should specify either the <App Logo> or <App Text> command for your guide file, but not both.

EXAMPLES

```
#directly specify your application logo
<App Logo> "SurfWriter ColorLogo", "SurfWriter B&WLogo"
#or, import PICT resources with resource IDs 501 & 502
<Resource> "Standard Resources", ALL
```

SEE ALSO

For information on the <Resource> command, see page 10-101. The <App Text> command is described next.

<App Text>

You can use the <App Text> command to specify a text string that Apple Guide displays in the upper-left corner of a Full Access window.

<App Text> *string*

string A text string, no longer than 64 characters, that helps identify this guide file (for example, the application's name) to the user.

DESCRIPTION

Apple Guide displays the string defined by the <App Text> command in the upper-left corner of a Full Access window. It displays this string in 16-point Espy Serif Bold; no other font or style options are available. You typically use this command only if you do not want to provide a graphic of your application's logo.

SPECIAL CONSIDERATIONS

The <App Text> command should appear once at most in your source files for a specific guide file. If you omit this command (and the <App Logo> command), Apple Guide displays the default access window background in the upper-left corner of the Full Access window.

You should specify either the <App Logo> or <App Text> command for your guide file, but not both.

EXAMPLES

<App Text> "SurfWriter Guide"

SEE ALSO

For information on the <App Logo> command, see page 10-25.

Specifying Default Settings

You can specify default settings for your guide file by using the commands described in this section. For example, you can specify the default maximum and minimum height of a panel, defaults for the font, style, size, and color of text in a panel, default navigation buttons, and a default prompt set.

<Max Height>

You can use the <Max Height> command to specify the maximum allowable panel height.

<Max Height> *height*

height A short integer specifying the maximum panel height in pixels.

DESCRIPTION

Apple Guide automatically sizes each panel to a height that is within the range specified by the <Max Height> and <Min Height> commands. If you omit the <Max Height> command, Apple Guide uses a default maximum panel height of 250 pixels.

If the contents of a panel exceeds the maximum specified by the <Max Height> command or exceeds the default maximum, Apple Guide splits the panel into multiple panels to accommodate all the panel objects.

To specify a fixed panel size for all panels in the guide file regardless of content, set the maximum height and minimum height to the same value.

SPECIAL CONSIDERATIONS

The <Max Height> command should appear only once in your source files.

EXAMPLES

```
#set the maximum height allowed to 350 pixels  
<Max Height> 350
```

```
#set all panels to a fixed height of 300 pixels  
<Max Height> 300  
<Min Height> 300
```

SEE ALSO

The `<Min Height>` command is described next.

`<Min Height>`

You can use the `<Min Height>` command to specify the minimum allowable panel height.

```
<Min Height> height
```

height A short integer specifying the minimum panel height in pixels.

DESCRIPTION

Apple Guide automatically sizes each panel to a height that is within the range specified by the `<Max Height>` and `<Min Height>` commands. If you omit the `<Min Height>` command, Apple Guide uses a default minimum panel height of 0 pixels.

If the contents of a panel requires less space than the amount specified by the `<Min Height>` command, Apple Guide allocates the minimum height for the panel.

To specify a fixed panel size for all panels in the guide file regardless of content, set the maximum height and minimum height to the same value.

SPECIAL CONSIDERATIONS

The `<Min Height>` command should appear only once in your source files.

EXAMPLES

```
#set the minimum height of a panel to 100 pixels  
<Min Height> 100
```

SEE ALSO

For information on the `<Max Height>` command, see page 10-28.

`<Default Format>`

You can use the `<Default Format>` command to specify a defined format or transparent format that Guide Maker uses by default when placing text and objects in a panel.

`<Default Format>` *formatName*

formatName A text string specifying the name of the format or transparent format to use as the default format.

DESCRIPTION

A `<Default Format>` command specifies the default format that Guide Maker applies when placing text and objects in a panel. You can override the default format on a panel-by-panel basis by using a `<Format>` command in the panel's panel definition.

Any panel text or objects that appear before a `<Default Format>` command are aligned according to Guide Maker's default full-width panel format.

When Guide Maker encounters a `<Default Format>` command, it uses the format defined by *formatName* as the default format. Guide Maker uses the default format in a panel until encountering a `<Format>` command for that panel. It then uses the specified format to place all following text and objects in

that panel until another <Format> command is encountered or until an <End Panel> command is encountered. Guide Maker resets the format to the default format upon encountering an <End Panel> command.

Panel objects aligned according to the default format specified in a <Default Format> command are placed inline within the bounds specified by the format's column coordinates. Any text attributes specified by the format are applied to panel text, and any prompts are aligned according to the specified format.

You define formats that can be used by a <Default Format> command using the <Define Format> or <Define Transparent Format> commands.

The file Standard Setup is provided with Guide Maker. This file defines four formats:

- **Tag.** A format that provides a left column that you can use to format tags, that is, text such as "Do This" or "Oops".
- **Body.** A format that provides a right column and that is designed for use with the Tag format. Use the Body format to provide the information that goes in the right column of a panel that also has a tag in it.
- **Full.** A format that provides a full column. Use this format if your panel text requires a full-column width.
- **ResetPen.** A format that resets the format to a default format.

If you include the Standard Setup file in your build file, you can automatically use these formats as needed in your source files.

SPECIAL CONSIDERATIONS

The <Default Format> command should appear once at most in your source files.

EXAMPLES

```
#use the format with format name "Full" as the default format
# (the "Full" format is defined in the Standard Setup file)
```

```
# the "Full" format specifies a format with column coordinates of
# top = 6, left = 11, and right = 330
# and text attributes of Espy Serif, 10 point,
# plain text style, black text color, left aligned,
# and does not override the default alignment of the prompt.
<Define Format> "Full", Column(6, 11, 330), "Espy Serif", 10, ~
                PLAIN, , LEFT, FALSE
<Default Format> "Full"
```

SEE ALSO

For information on the <Define Format> and <Define Transparent Format> commands, see page 10-85 and page 10-91. For information on the <Format> command, see page 10-93.

<Default Nav Button Set>

You can use the <Default Nav Button Set> command to specify a default navigation button set that Apple Guide uses for all sequences in your guide file unless you override the default by using a <Seq Nav Button Set> command.

```
<Default Nav Button Set> navButtonSetName
```

navButtonSetName

A string specifying the name of the navigation button set to use as the default. You can use the constant **NONE** to indicate that Apple Guide should display no navigation buttons (other than the navigation arrows) by default.

DESCRIPTION

The <Default Nav Button Set> command defines the default set of navigation buttons (in addition to the navigation arrows) that Apple Guide displays for your guide file. You can define a navigation button set for specification in the *navButtonSetName* parameter using a <Define Nav Button Set> command.

Note that any navigation buttons that you define using a `<Define Nav Button Set>` command appear to the left of the navigation arrows. Apple Guide always displays the navigation arrows on each panel. For each panel, Apple Guide makes the right navigation arrow active or inactive according to whether the user can navigate to a following panel. Apple Guide makes the left navigation arrow active or inactive according to whether the user can navigate to a previous panel. Apple Guide also displays the panel number (as it appears in a sequence) between the left and right navigation arrows.

The file `Standard Setup` is provided with Guide Maker. This file contains descriptions of the Huh? and GoStart navigation buttons, and defines three navigation button sets:

- **“Standard Nav Bar”**. A navigation button set that specifies the GoStart and Huh? navigation buttons as buttons in the navigation bar.
- **“GoStart Only”**. A navigation button set that specifies only the GoStart navigation button as a button in the navigation bar. The GoStart button resembles a lightbulb.
- **“Huh? Only”**. A navigation button set that specifies only the Huh? navigation button as a button in the navigation bar.

The `Standard Setup` file also defines the “Standard Nav Bar” button set as the default navigation button set. If you include the `Standard Setup` file in your build file, you can automatically use these three navigation button sets as needed in your source files.

SPECIAL CONSIDERATIONS

The `<Default Nav Button Set>` command should appear only once in your source files. If you omit this command (or provide the constant `NONE` as a parameter), Apple Guide by default displays no navigation buttons for your guide file (but does display the left and right navigation arrows).

EXAMPLES

```
#use GoStart Only as default
<Default Nav Button Set> "GoStart Only"
#or use Huh? Only as default
<Default Nav Button Set> "Huh? Only"
```

```
#or use both GoStart and Huh? as default
<Default Nav Button Set> "Standard Nav Bar"
#or use no nav buttons by default
<Default Nav Button Set> NONE
#or use your own default navigation button set
<Default Nav Button Set> "my default nav buttons"
```

SEE ALSO

For information on the <Define Nav Button Set> and <Seq Nav Button Set> commands, see page 10-80, and page 10-43, respectively.

<Allow Prompts>

You can use the <Allow Prompts> command to specify whether Guide Maker should include space for prompts when it creates panels.

<Allow Prompts> *allow*

allow A Boolean value indicating whether Guide Maker should include space for prompts when determining the size of panels. Specify **TRUE** if you want Guide Maker to include space for prompts on your panels; specify **FALSE** otherwise. By default, Guide Maker includes space for prompts when determining the size of panels.

DESCRIPTION

When Guide Maker encounters the <Allow Prompts> command with *allow* set to **FALSE**, it does not include space for prompts when determining the size of any panel definitions following this command. For example, you might use this command if all your panels have full-size graphics.

SPECIAL CONSIDERATIONS

If you use the `<Allow Prompts>` command and set *allow* to **FALSE**, you should not use any other prompt-related command. If you do so, Guide Maker generates an error relating to this condition when compiling your source file.

If you omit the `<Allow Prompts>` command, Guide Maker includes space for prompts when determining the size of panels.

EXAMPLES

```
<Allow Prompts> FALSE
```

SEE ALSO

For information on the `<Define Prompt Set>` command, see page 10-37. For information on setting the prompt for a sequence or a particular panel, see the description of the `<Sequence Prompt Set>` and `<Panel Prompt>` commands on page 10-42 and page 10-55. The `<Default Prompt Set>` command is described next.

<Default Prompt Set>

You can use the `<Default Prompt Set>` command to specify a default set of navigation prompts that Apple Guide uses for all panels in all sequences in your guide file unless you override the default by using a `<Sequence Prompt Set>` or `<Panel Prompt>` command.

```
<Default Prompt Set> promptSetName
```

promptSetName

A string specifying the name of the navigation prompt set to use as the default. You can use the constant **NONE** to indicate that Apple Guide by default should display no navigation prompts.

DESCRIPTION

The `<Default Prompt Set>` command defines the default set of navigation prompts that Apple Guide displays for your guide file. You define the navigation prompt set (for specification in the *promptSetName* parameter) using a `<Define Prompt Set>` command. If you specify the constant `NONE` (indicating that panels don't use prompts by default), Guide Maker doesn't include space for prompts when it compiles panels, unless you override the default using the `<Sequence Prompt Set>` or `<Panel Prompt>` command.

SPECIAL CONSIDERATIONS

The `<Default Prompt Set>` command should appear only once in your source files. If you omit this command (or provide the constant `NONE` as a parameter), Guide Maker by default allocates no space for navigation prompts for your guide file.

EXAMPLES

```
#first define a prompt set
<Define Prompt Set> "default navigation prompts",~
    "To begin, click the right arrow.",~
    "Click the left arrow to go back or the right arrow to continue.",~
    "That's all, you're done!",~
    "Make your choice, then click the right arrow."
#use this prompt set as the default
<Default Prompt Set> "default navigation prompts"
```

SEE ALSO

For information on the `<Allow Prompts>` command, see page 10-34. For information on the `<Sequence Prompt Set>` and `<Panel Prompt>` commands, see page 10-42 and page 10-55, respectively. The `<Define Prompt Set>` command is described next.

<Define Prompt Set>

You can use the <Define Prompt Set> command to specify a set of prompts that can appear as navigation prompts on panels.

<Define Prompt Set> *promptSetName, promptFirstPanel, promptMiddlePanel, promptLastPanel, promptForPanelsWithControls*

promptSetName

A string specifying the name of this navigation prompt set. This name must be unique from all other navigation prompt sets that you define.

promptFirstPanel

A string specifying the text for use as the navigation prompt of the first panel in a sequence.

promptMiddlePanel

A string specifying the text for use as the navigation prompt of the middle panel in a sequence.

promptLastPanel

A string specifying the text for use as the navigation prompt of the last panel in a sequence.

promptForPanelsWithControls

A string specifying the text for use as the navigation prompt of panels containing any of these controls: checkboxes, radio buttons, or standard buttons.

DESCRIPTION

The <Define Prompt Set> command defines a set of navigation prompts. You can use this prompt set in panels and can control on which panels the prompts appear by using these commands:

- <Default Prompt Set>, to associate the prompt set with *all* panels in *all* sequences
- <Sequence Prompt Set>, to associate the prompt set with *all* panels in a *specific* sequence

- **<Panel Prompt>**, to associate the prompt set with only *one* specific panel in a sequence

If you provide a **<Default Prompt Set>** command that specifies a prompt set (other than **NONE**), Guide Maker allocates space for prompts on all panels and uses the specified prompt strings for each panel by default. You can override the default on a sequence or panel-by-panel basis.

If a sequence definition includes a **<Sequence Prompt Set>** command that specifies a prompt set (other than **NONE**), Guide Maker allocates space for prompts on all panels in that sequence by default, and uses the specified prompt strings for each panel in the sequence.

If a panel definition includes a **<Panel Prompt>** command that specifies a prompt set (other than **NONE**), Guide Maker allocates space for prompts on that panel and uses the specified prompt strings for that panel.

To specify that Guide Maker should not allocate space for panels, specify **NONE** as a parameter to the **<Default Prompt Set>**, **<Sequence Prompt Set>**, or **<Panel Prompt>** commands. If you want Guide Maker to allocate space for a prompt on a panel but do not want to provide text in the prompt string, specify the prompt string with at least one blank character, for example, " "; do not specify an empty string.

EXAMPLES

```
<Define Prompt Set> "special prompts", ~
"Click the right arrow to continue.", ~
"Click the left arrow to go back or the right arrow to continue.",~
"That's all, you're done!", ~
" "
```

```
<Define Panel> "Example Panel 1"
  #this panel doesn't use a prompt set
  <Panel Prompt> NONE
<End Panel>
```

```
<Define Panel> "Example Panel 2"
<End Panel>
```

```

<Define Sequence> "Example Sequence 2"
  #this sequence uses the prompt set
  # defined by "special prompts"
  <Sequence Prompt Set> "special prompts"
  <Panel> "Example Panel 1" #overrides sequence prompt set
  <Panel> "Example Panel 2" #uses sequence prompt set
<End Sequence>

```

SEE ALSO

For a description of the <Default Prompt Set>, <Sequence Prompt Set>, and <Panel Prompt> commands, see page 10-35, page 10-42, and page 10-55, respectively.

Creating Sequences

You can define a sequence—which specifies a set of panels and determines their order of display—by using the commands described in this section.

<Define Sequence>

You use the <Define Sequence> command to mark the beginning of a sequence definition. A sequence definition typically contains commands that specify the panels of the sequence and the conditions that control their order of display. Apple Guide displays a sequence when the user selects the sequence's associated topic from an access window.

```
<Define Sequence> sequenceName [ , seqDisplayTitle ]
```

sequenceName A text string specifying the name of the sequence. The sequence name can be up to 255 characters long, and the first 63 characters must be unique from all other sequence names in your file. If you do not specify the *seqDisplayTitle* parameter, Apple Guide uses the sequence name as the sequence display title, which appears in the title bar of the sequence's presentation window.

seqDisplayTitle

A text string specifying the display title of the sequence. The sequence display title can be up to 255 characters long and does not have to be unique. This parameter is optional. However, you must specify this parameter if you intend to localize your guide file using Guide Maker's localize feature.

If you specify this parameter, Apple Guide uses this string rather than the sequence name as the sequence display title. The sequence display title appears in the title bar of the sequence's presentation window.

DESCRIPTION

To build a sequence, Guide Maker collects all the commands between a <Define Sequence> and <End Sequence> command. Each sequence must have a unique sequence name. A sequence also has a sequence display title, which Apple Guide displays in the title bar of the sequence's presentation window. If you don't specify a sequence title in the second parameter of the <Define Sequence> command, Apple Guide uses the sequence name as the sequence display title. Apple Guide always displays a sequence title for a sequence, except in these two cases:

- Apple Guide does not display the sequence title of a subsequence. Instead, Apple Guide continues to display the main sequence title.
- Apple Guide does not display the sequence title of an Oops or Continue sequence. Instead, Apple Guide continues to display the main sequence title.

To include panels in a sequence, either use the <Panel> command (which references a defined panel) or define the panel within the sequence by placing the <Define Panel> and <End Panel> commands directly within the sequence itself. Either method works equally well; if you often reuse panels or want to see a quick synopsis of a sequence, use <Panel> commands. If you prefer to keep the definition of a panel with its sequence, define the panel directly within the sequence.

You can also specify conditional display of panels, using the <If>, <Else>, <End If>, <Skip If>, <Make Sure>, <Start Making Sure>, and <End Making Sure> commands.

SPECIAL CONSIDERATIONS

A sequence can have at most 32 panels, except for any panels added using the `<Jump Sequence>` command.

EXAMPLES

```
#sequence name and sequence title are different
<Define Sequence> "SequenceChangeWordFont", ~
    "How do I change the font of a word?" #seqTitle
<End Sequence>
```

```
#sequence name and sequence title are the same
<Define Sequence> "How do I open a file?"
<End Sequence>
```

```
#sequence name and sequence title are the same but both are
# specified for ease of localization
<Define Sequence> "How do I close a file?", ~
    "How do I close a file?"
<End Sequence>
```

```
#sequence with panels referenced by <Panel> commands
<Define Sequence> "How do I create index markers?"
    <Panel> "index intro"
    <Panel> "index tool"
    <Panel> "create index"
<End Sequence>
```

```
#sequence with panels defined within
<Define Sequence> "How do I create index markers?"
    <Define Panel> "creating index markers"
        To create index markers, select the index tool.
        #More text and commands for this panel here
    <End Panel>
<End Sequence>
```

SEE ALSO

For information on the <Panel>, <Define Panel>, and <End Panel> commands, see page 10-45, page 10-52, and page 10-56, respectively. For information on the conditional display of panels, see “Specifying Conditional Execution” beginning on page 10-152.

<Sequence Prompt Set>

You can use the <Sequence Prompt Set> command to specify a set of navigation prompt strings for all panels in a sequence.

<Sequence Prompt Set> *promptSetName*

promptSetName

The name of a defined prompt set, or the constant **NONE** to indicate that the sequence doesn’t require prompts.

DESCRIPTION

The <Sequence Prompt Set> command overrides the default navigation prompt set (you specify a default navigation prompt set using the <Default Prompt Set> command). When a sequence includes a <Sequence Prompt Set> command, Apple Guide uses the specified prompts for all panels in the sequence, except for panels that use a <Panel Prompt> command to override the sequence prompt set.

If you specify the constant **NONE** (indicating that prompts are not required by default for any panels in the sequence), Guide Maker doesn’t include space for prompts when it compiles panels in the sequence, unless you override the sequence prompt set for a particular panel using the <Panel Prompt> command.

SPECIAL CONSIDERATIONS

A sequence can have only one sequence prompt set associated with it. If you omit this command, Apple Guide uses the default prompt set. If Guide Maker encounters more than one <Sequence Prompt Set> command for a single sequence, it uses the last one encountered for that sequence.

If you use the <Sequence Prompt Set> command, it must always appear between the <Define Sequence> and <End Sequence> commands.

EXAMPLES

```
<Define Prompt Set> "prompts for special sequence", ↵
                    "Click the right arrow to begin.", ↵
                    " ", "That's all, you're done!", " "
```

```
<Define Sequence> "Example Sequence"
    #this sequence uses the prompt set defined
    # by "prompts for special sequence"
    <Sequence Prompt Set> "prompts for special sequence"
    <Panel> "Example panel 1"
    <Panel> "Example panel 2"
<End Sequence>
```

SEE ALSO

For a description of the <Define Prompt Set> command, see page 10-37.

<Seq Nav Button Set>

You can use the <Seq Nav Button Set> command to specify a navigation button set that Apple Guide uses for a particular sequence in your guide file. This command overrides any default set by a <Default Nav Button Set> command.

```
<Seq Nav Button Set> navButtonSetName
```

navButtonSetName

A string specifying the name of the navigation button set to use for this sequence. You can use the constant `NONE` to indicate that Apple Guide should display no navigation buttons (other than the navigation arrows) for this sequence.

DESCRIPTION

The <Seq Nav Button Set> command defines the set of navigation buttons that Apple Guide displays for a particular sequence. You define a navigation button set (for specification in the *navButtonSetName* parameter) using the <Define Nav Button Set> command.

SPECIAL CONSIDERATIONS

If you use the <Seq Nav Button Set> command, it must always appear between the <Define Sequence> and <End Sequence> commands.

EXAMPLES

```
#specify navigation buttons to use as a default
<Default Nav Button Set> "default navigation buttons"

<Define Sequence> "Example Sequence"
    #this sequence uses the default navigation buttons
<End Sequence>

<Define Sequence> "How do I spell-check a document?"
    #this sequence overrides the default navigation buttons
    <Seq Nav Button Set> "special sequence nav buttons"
    <Panel> "automatic spell-checking"
<End Sequence>
```

SEE ALSO

For information on the <Define Nav Button Set> and <Default Nav Button Set> commands, see page 10-80, and page 10-32, respectively.

<Panel>

You can use the <Panel> command to associate a panel with a particular sequence.

<Panel> *panelName*

panelName A text string specifying the name of a defined panel.

DESCRIPTION

A sequence definition consists of commands that reference one or more panels. Apple Guide displays panels according to the order in which the panels appear in a sequence definition and according to any conditions attached to each panel.

Note that Apple Guide uses the sequence name (or the optional sequence display title) as the panel's display title, not the string in the *panelName* parameter.

The *panelName* parameter must reference a named panel, that is, a panel defined with the <Define Panel> and <End Panel> commands. You can also directly place a panel definition within a sequence definition by using the <Define Panel> and <End Panel> commands. If you often reuse panels or want to see a quick synopsis of a sequence, use <Panel> commands in your sequence definitions. If you prefer to keep the definition of a panel with its sequence, define the panel directly within the sequence.

Note that panel definitions do not have to precede their use in a <Panel> command; they must simply appear somewhere within your source files.

EXAMPLES

```
#sequence with panels referenced by <Panel> commands
<Define Sequence> "How do I create index markers?"
    <Panel> "index intro"
    <Panel> "index tool"
    <Panel> "create index"
<End Sequence>
```

```
#sequence with panels defined within
<Define Sequence> "How do I create index markers?"
  <Define Panel> "creating index markers"
    To create index markers, select the index tool.
    #more text and commands for this panel here
  <End Panel>
<End Sequence>
```

SEE ALSO

For information on the <Define Panel> and <End Panel> commands see page 10-52 and page 10-56, respectively.

<Insert Sequence>

You can use the <Insert Sequence> command to include the commands from another sequence into the current sequence.

```
<Insert Sequence> sequenceName
```

sequenceName

A text string specifying the name of a sequence.

DESCRIPTION

The <Insert Sequence> command copies the information from the named sequence into the current sequence's definition. You can use this command to reuse sequences common to one or more help topics.

SPECIAL CONSIDERATIONS

Any panels inserted by an <Insert Sequence> command apply toward a sequence's 32-panel limit.

EXAMPLES

```
#sequence that is reused often
<Define Sequence> "How do I use the tool bar?"
  <Panel> "intro tool bar"
  <Panel> "tools"
<End Sequence>

#sequence that reuses another sequence
<Define Sequence> "How do I create footnotes?"
  <Panel> "intro footnotes"
  <Insert Sequence> "How do I use the tool bar?"
  <Panel> "editing footnotes"
<End Sequence>
```

SEE ALSO

The <Jump Sequence> command is described next.

<Jump Sequence>

You can use the <Jump Sequence> command to jump to another sequence from the current sequence.

```
<Jump Sequence> sequenceName
```

sequenceName

A text string specifying the name of the sequence to jump to.

DESCRIPTION

Unlike the <Insert Sequence> command, the <Jump Sequence> command jumps directly to the named sequence. The jumped-to sequence is treated as a separate entity in the guide file and is jumped to and returned from at the appropriate times in the calling sequence. Like the <Insert Sequence> command, the <Jump Sequence> command lets you reuse sequences that are

common to one or more help topics. You also might use <Jump Sequence> commands within <If> and <End If> commands.

SPECIAL CONSIDERATIONS

Any sequence that uses <Jump Sequence> commands must have at least one panel that is guaranteed to be shown before the first <Jump Sequence> command. You should also show at least one panel between <Jump Sequence> commands.

Any panels inserted by a <Jump Sequence> command do *not* apply toward a sequence's 32-panel limit.

You can use the <Jump Sequence> command if Guide Maker informs you that the current sequence exceeds limitations.

EXAMPLES

#sequence that is reused often

```
<Define Sequence> "How do I use the tool bar?"
    <Panel> "tool bar:intro"
    <Panel> "tool bar:tools"
<End Sequence>
```

#sequence that jumps to another sequence

```
<Define Sequence> "How do I create footnotes?"
    <Panel> "footnotes:intro"
    <Jump Sequence> "How do I use the tool bar?"
    <Panel> "footnotes:editing"
<End Sequence>
```

#sequence that jumps to a sequence depending on a condition

```
<Define Sequence> "How do I spell-check a document?"
    <Panel> "spell-checking:intro"
    <If> NOT isDictionaryOpen()
        <Jump Sequence> "How do I open the dictionary?"
    <End If>
```

```
<Panel> "spell-checking:options"  
<End Sequence>
```

SEE ALSO

For information on the <Insert Sequence> command, see page 10-46. The <Launch New Sequence> command is described next.

<Launch New Sequence>

You can use the <Launch New Sequence> command if you need to break complex sequences into smaller subsequences.

```
<Launch New Sequence> sequenceName
```

sequenceName

A text string specifying the name of the new sequence.

DESCRIPTION

The <Launch New Sequence> command displays the named sequence in the same window as the current sequence, and the user navigates to it using the right arrow button. However, the user cannot navigate back to the original sequence using the left arrow button after beginning the new sequence.

If you use the <Launch New Sequence> command, you usually specify it as the last command in the current sequence definition.

You can also use the <Radio Button Launch New Seq> command to launch new sequences in response to the user choosing a radio button. Alternatively, you can launch a new sequence as a result of the user choosing a button by linking the button to the built-in function `LaunchNewSequence`.

SPECIAL CONSIDERATIONS

Use the <Launch New Sequence> command only if Guide Maker informs you that the current sequence exceeds limitations.

EXAMPLES

```
#sequence to launch to
<Define Sequence> "How do I make a multibook index?"
    <Panel> "multibook setup"
    <Panel> "multibook combine"
    <Panel> "multibook generate"
<End Sequence>

#sequence that launches another sequence
<Define Sequence> "How do I create an index?"
    <Panel> "index techniques"
    <Panel> "editing index entries"
    <Launch New Sequence> "How do I make a multibook index?"
<End Sequence>
```

SEE ALSO

For information on the <Insert Sequence> and <Jump Sequence> commands, see page 10-46 and page 10-47, respectively. For information on the <Radio Button Launch New Seq> command, see page 10-66. For information on the built-in event functions `LaunchNewSequence` and `LaunchNewSequenceNewWindow`, see page 10-188.

<Build Sequence>

You can use the <Build Sequence> command to build a sequence that isn't accessed from another panel or whose topic doesn't appear in an access window, such as a help sequence called directly by your application to provide context-sensitive help.

<Build Sequence> *sequenceName*, [*seqResID*]

sequenceName

A string identifying the name of a defined sequence (that is, a sequence defined with the <Define Sequence> and <End Sequence> commands).

seqResID

A resource ID to be assigned to this sequence. Use a resource ID greater than 2000. This parameter is optional and, if it's not provided, Guide Maker assigns a resource ID to the sequence.

DESCRIPTION

The <Build Sequence> command compiles the specified sequence and stores it as a resource in the guide file. Your application can directly invoke a sequence built by the <Build Sequence> command by specifying its resource ID in the *sequenceID* parameter of the `AGOpenWithSequence` function.

EXAMPLES

```
<Build Sequence> "name of sequence", 2001
```

SEE ALSO

For information on the <Define Sequence> and <End Sequence> commands, see page 10-39 and page 10-51, respectively.

<End Sequence>

You can use the <End Sequence> command to mark the end of a sequence definition.

```
<End Sequence>
```

DESCRIPTION

Guide Maker collects all the commands between a `<Define Sequence>` and `<End Sequence>` command to build a sequence.

EXAMPLES

```
#sequence with panels referenced by <Panel> commands
<Define Sequence> "How do I create index markers?"
    <Panel> "index intro"
    <Panel> "index tool"
    <Panel> "create index"
<End Sequence>
```

SEE ALSO

For information on the `<Define Sequence>` command, see page 10-39.

Creating Panels

You can define a panel and the prompt for a specific panel by using the commands described in this section.

`<Define Panel>`

You can use the `<Define Panel>` command to mark the beginning of a panel definition. A panel definition typically contains commands that specify the objects to be displayed on the panel.

`<Define Panel>` *panelName*

panelName A text string specifying the name of the panel. The panel name can be up to 255 characters long, and the first 63 characters must be unique from all other panel names that you define. Apple Guide never displays the panel name to the user, so you should specify the panel name in a way that is most useful to you.

DESCRIPTION

To build a panel, Guide Maker collects all the commands between a `<Define Panel>` and `<End Panel>` command. A panel that you have defined with these two commands can be included in a sequence simply by referencing the panel name using the `<Panel>` command. Alternatively, you can directly define the panel within a sequence.

You can provide the content of a panel, for example, graphics, buttons, or a QuickTime movie, using Guide Script commands such as `<PICT>`, `<Standard Button>`, `<3D Button>`, and `<QuickTime>`. You specify text in a panel by placing it directly within the panel definition. Guide Maker treats any line in a panel definition that doesn't begin with a command or comment as panel text. (Specifically, Guide Maker scans for the first "nonblank" character, that is, a character other than a space or tab, and if this character is anything other than `"<"` or `"#"`, it is considered panel text.) Guide Maker formats panel text according to the format specified by the default format or the format specified by a `<Format>` command.

You can allow Guide Maker to automatically place panel text and objects in the panel, or you can specifically place text and objects yourself using `<Format>` commands and by providing placement information in Guide Script commands that place objects.

SPECIAL CONSIDERATIONS

Your panel definitions do not have to precede your sequence definitions in your source file. Thus, a sequence can reference any panel name, as long as that panel is defined somewhere in your source files.

EXAMPLES

```
#panel definition defined outside of a sequence
<Define Panel> "index intro"
    #text and commands that specify content of panel here
<End Panel>
```

```

#sequence with panels referenced by <Panel> commands
<Define Sequence> "How do I create index markers?"
    <Panel> "index intro"
    <Panel> "index tool"
    <Panel> "create index"
<End Sequence>

#panel definition that uses various commands to
# create panel content
<Define Panel> "creating footnotes"
    #text formatted by Guide Maker and using default format
    To create footnotes, select the Footnote tool.
    #Here's a graphic
    <PICT> "ColorPic1 File", CENTER, "B&WPic1 File"
    #text placed using a specific format
    <Format> "special format"
    Here's some more text.
    #Here's a button
    <Standard Button> "Display Footnotes", CENTER, ↵
                        doButton2Action()
    #More text and commands for this panel here
<End Panel>

```

SEE ALSO

For information on the <Panel> and <End Panel> commands, see page 10-45 and page 10-56, respectively. For more information on placing buttons, graphics, and objects in your panels, see “Creating Buttons” beginning on page 10-57, “Specifying Pictures and Movies” beginning on page 10-94, and “Creating Hot Items” beginning on page 10-119. For information on using formats in panels, see “Formatting Text and Objects in a Panel” beginning on page 10-84.

<Panel Prompt>

You can use the <Panel Prompt> command to specify a set of navigation prompt strings for the current panel.

<Panel Prompt> *promptSetName*

promptSetName

The name of a defined prompt set, or the constant `NONE` to indicate that the panel doesn't require prompts.

DESCRIPTION

The <Panel Prompt> command overrides the default navigation prompt set (the default for all panels in all sequences) and the default sequence prompt set (the default for all panels in a sequence). You define a navigation prompt set (for specification in the *promptSetName* parameter) using the <Define Prompt Set> command. If you indicate that the panel doesn't require prompts, Guide Maker allocates no space for prompts when it compiles the panel.

SPECIAL CONSIDERATIONS

A panel can have only one panel prompt set associated with it. If Guide Maker encounters more than one <Panel Prompt> command for a single panel, it uses the last one encountered for that panel.

If you use the <Panel Prompt> command, it must always appear between the <Define Panel> and <End Panel> commands.

EXAMPLES

```
<Define Prompt Set> "special prompts" ~
    "Click the right arrow to continue.", ~
    "Click the left arrow to go back or the right arrow to go on.", ~
    "That's all, you're done!", ~
    "Select one, then click the right arrow to continue."
```

```
<Define Panel> "Example Panel 1"
    #this panel uses the prompt set defined by "my panel prompts"
    <Panel Prompt> "my panel prompts"
<End Panel>

<Define Panel> "Example Panel 2"
    <Panel Prompt> NONE      #this panel doesn't require prompts
<End Panel>

<Define Panel> "Example Panel 3"
                                #this panel uses sequence prompts
<End Panel>

<Define Sequence> "Example Sequence 3"
    #this sequence uses the prompt set defined by "special prompts"
    <Sequence Prompt Set> "special prompts"
    <Panel> "Example Panel 1" #overrides sequence prompt set
    <Panel> "Example Panel 2" #uses no prompts
    <Panel> "Example Panel 3" #uses sequence prompt set
<End Sequence>
```

SEE ALSO

For a description of the <Define Prompt Set> command, see page 10-37.

<End Panel>

You can use the <End Panel> command to mark the end of a panel definition.

```
<End Panel>
```

DESCRIPTION

Guide Maker collects all the commands between a <Define Panel> and <End Panel> command to build a panel.

EXAMPLES

```
#panel definition for use in a sequence
<Define Panel> "index intro"
    #text and commands that specify content of panel here
<End Panel>
```

SEE ALSO

For information on the <Define Panel> command, see page 10-52.

Creating Buttons

You can create standard buttons, three-dimensional buttons, radio buttons, and checkboxes in the content area of panels, as well as navigation buttons in the navigation bar, by using the commands described in this section.

<Standard Button>

You can use the <Standard Button> command to place a standard 2D button on a panel. You must specify the button's title, its location on the panel, and the event function that Apple Guide calls when the user clicks the button.

<Standard Button> *buttonTitle*, *buttonLoc*, *buttonEvent* [, *buttonFont*]

buttonTitle A string specifying the button's title.

buttonLoc A constant specifying either the button's general location or its coordinates relative to the current pen location.

Guide Script Command Reference

To describe a button's coordinates, use the **Point** function. The current pen location's horizontal coordinate is the left edge of the current format; its vertical coordinate corresponds to the bottom edge of the last object not specifically placed using coordinates.

You can also use these constants to describe the button location:

LEFT

CENTER

RIGHT

If you specify one of these constants, Guide Maker justifies the button accordingly within the current format.

buttonEvent A name of an event function or event list. Include any parameters expected by the function in parentheses following its name. You define event functions using the **<Define Event>** or **<Define Event List>** command. Guide Maker also provides built-in functions that you can specify in this parameter. When the user clicks the button, Apple Guide calls the function defined by this parameter; usually such a function is used to send a specific Apple event.

buttonFont A constant specifying the font of the button's title. You can use either of these constants to specify the corresponding font:

APPLEGUIDE Specifies 10-point Espy Serif.

SYSTEM Specifies the system font of the current script system. For example, for Roman script systems, indicates 12-point Chicago.

This parameter is optional; if you omit this parameter Apple Guide displays the button title in the font of the current script system.

DESCRIPTION

The **<Standard Button>** command places a button with the specified title on a panel. Apple Guide performs the action indicated by the *buttonEvent* parameter when the user clicks the button. When you specify a button using the **<Standard Button>** command, Guide Maker creates a button that is 20 pixels high with a minimum width of 59 pixels (the width of a standard OK button). Guide Maker sizes the button to fit the button's title (the width of the text plus 10 pixels on each side).

Buttons appear inline with the surrounding text and are positioned within the current format. You can specify the button's location generally, using the constants `LEFT`, `CENTER`, or `RIGHT`. If you specify one of these constants, Guide Maker positions the button in the current format, and left-justifies, right-justifies, or centers the button, accordingly. You can also specify the button's location relative to the current pen position by specifying a specific point. For example, if you specify the button location as `Point(50,100)`, Guide Maker positions the button 50 pixels to the right and 100 pixels down from the current pen location.

EXAMPLES

```
#define doButton1Action as an event function that sends to
# the app with signature 'WAVE' (SurfWriter) the Apple event
# defined by event class 'sfwr' and event ID 'act1'
<Define Event> "doButton1Action", 'WAVE', 'sfwr', 'act1'

#define doButton2Action as an event function
<Define Event> "doButton2Action", 'WAVE', 'sfwr', 'act2'

<Define Panel> "Example Panel"

    #place "Create Book Index" button right-justified on panel
    <Standard Button> "Create Book Index", RIGHT, doButton1Action()

    #place "Create Chapter Index" button relative to the current
    # pen location, that is,
    # 50 pixels to the right, 100 pixels down
    <Standard Button> "Create Chapter Index", Point(50,100), ~
        doButton2Action()

<End Panel>
```

SEE ALSO

For information on using formats when placing buttons, see “Formatting Text and Objects in a Panel” beginning on page 10-84. For information on the

<Define Event> and <Define Event List> commands, see page 10-178 and page 10-181, respectively. For information on creating radio buttons and checkboxes, see the descriptions of the <Radio Button> and <CheckBox> commands on page 10-64 and page 10-69, respectively. The <3D Button> command is described next.

<3D Button>

You can use the <3D Button> command to place a 3D button on a panel. You must specify the graphic associated with this button, the location of the button on the panel, and an event function that Apple Guide calls when the user clicks the button.

<3D Button> *buttonUpPict*, *buttonDownPict*, *buttonLoc*, *buttonEvent*
 [, *b&wUp*] [, *b&wDown*]

buttonUpPict A resource ID, resource name, or filename that identifies the picture that defines the appearance of the button in its normal state (that is, when the button is not pressed). If you specify a resource ID or resource name, you must make the resource available to Guide Maker using the <Resource> command. If you specify a filename, the file must be in the same folder as your source files in order for Guide Maker to find the file.

buttonDownPict A resource ID, resource name, or filename that identifies the picture that defines the appearance of the button when the cursor is in the button and the user presses the mouse button. If you specify a resource ID or resource name, you must make the resource available to Guide Maker using the <Resource> command. If you specify a filename, the file must be in the same folder as your source files in order for Guide Maker to find the file.

buttonLoc A constant specifying either the button's general location or its coordinates relative to the current pen location.

To describe the button's coordinates, use the `Point` function. The current pen location's horizontal coordinate is the left edge of the current format; the vertical coordinate corresponds to the bottom edge of the last object not specifically placed using coordinates.

You can also use these constants to describe the button location:

`LEFT`

`CENTER`

`RIGHT`

If you specify one of these constants, Guide Maker justifies the button accordingly within the current format.

<i>buttonEvent</i>	A name of an event function or event list. Include any parameters expected by the function in parentheses following its name. You define event functions using the <code><Define Event></code> or <code><Define Event List></code> command. Guide Maker also provides built-in functions that you can specify in this parameter. When the user clicks the button, Apple Guide calls the function defined by this parameter; usually such a function is used to send a specific Apple event.
<i>b&wUp</i>	A filename that, if provided, Apple Guide uses in place of the button appearance described by the <i>buttonUpPict</i> parameter only if the bit depth of the user's monitor is set to 4 bits or less. This parameter is optional and can be used only if you also specified a filename for the <i>buttonUpPict</i> parameter. However, if you specify a resource ID or resource name in the <i>buttonUpPict</i> parameter, you can still provide a black-and-white picture by importing a 'PICT' graphic whose resource ID is one greater than the resource ID of the color graphic.
<i>b&wDown</i>	A filename that, if provided, Apple Guide uses in place of the button appearance described by the <i>buttonDownPict</i> parameter only if the bit depth of the user's monitor is set to 4 bits or less. This parameter is optional and can be used only if you also specified a filename for the <i>buttonDownPict</i> parameter. However, if you specify a resource ID or resource name in the <i>buttonDownPict</i> parameter, you can still provide a black-and-white picture by importing a 'PICT' graphic whose resource ID is one greater than the resource ID of the color graphic.

DESCRIPTION

The `<3D Button>` command places a button on a panel and specifies an event function that Apple Guide calls when the user clicks the button. The button's appearance is determined by the *buttonUpPict* and *buttonDownPict* parameters. In addition, you can specify replacement pictures using the *b&wUp*, and *b&wDown* parameters, which Apple Guide uses according to the bit depth of the user's monitor. All pictures describing the button's appearance should have the exact same size. Note that if Apple Guide displays a black-and-white button, it uses the frame created for the color button. In general, you should always provide both a color and black-and-white version of the button.

Buttons appear inline with the surrounding text. If you specify the button's location using the constants `LEFT`, `CENTER`, or `RIGHT`, Guide Maker positions the button in the current format, and left-justifies, right-justifies, or centers the button, accordingly. You can also specify the button's location relative to the current pen position by specifying a specific point. For example, if you specify the button location as `Point (50 , 100)`, Guide Maker positions the button 50 pixels to the right and 100 pixels down from the current pen location.

SPECIAL CONSIDERATIONS

If you use the `<3D Button>` command, it must always appear between the `<Define Panel>` and `<End Panel>` commands.

If you do not explicitly specify a black-and-white picture for the button's up or down appearance and the guide file happens to contain a 'PICT' graphic whose resource ID is one greater than one of the button's color 'PICT' graphics, Apple Guide uses this 'PICT' graphic as the button's black-and-white picture.

EXAMPLES

```
#define doButton1Action as an event function that sends to
# the app with signature 'WAVE' (SurfWriter) the Apple event
# defined by event class 'sfwr' and event ID 'act1'
<Define Event> "doButton1Action", 'WAVE', 'sfwr', 'act1'

#define doButton2Action as an event function
<Define Event> "doButton2Action", 'WAVE', 'sfwr', 'act2'
```

```
<Define Panel> "Example Panel"
```

```
  To accomplish this task, do this:
```

```
  Very informative instructions here.
```

```
  #place Button1 right-justified on panel
```

```
  <3D Button> "Button1UpPict", "Button1DownPict", RIGHT, ~
              doButton1Action(), ~
              "Button1B&WUpPict", "Button1B&WDownPict"
```

```
  #Button2's appearance is defined by:
```

```
  # the 'PICT' w/ resource ID 2010 -- the color buttonUpPict
```

```
  # the 'PICT' w/ resource ID 2012 -- the color buttonDownPict
```

```
  # Guide Maker automatically looks in the guide file for 'PICT's
```

```
  # with resource ID's one greater than the specified color 'PICT's;
```

```
  # in this case, it looks for resource IDs of 2011 for the b&wUp
```

```
  # and 2013 for b&wDown and uses these as the black & white 'PICT's
```

```
  #place Button2 relative to the current pen location, that is,
```

```
  # 50 pixels to the right, 100 pixels down
```

```
  <3D Button> 2010, 2012, Point(50,100), doButton2Action()
```

```
<End Panel>
```

```
#define doOpenDocmt as an event function that sends to
```

```
# the app with signature 'WAVE' (SurfWriter) the
```

```
# Open Documents event. Note that when doOpenDocmt is called
```

```
# it expects a parameter (the name of the file to open)
```

```
<Define Event> "doOpenDocmt", 'WAVE', 'aevt', 'odoc',, '----'
```

```
<Define Panel> "Example Panel 3"
```

```
  To accomplish this task, do this:
```

```
  #place Open button left-justified on panel
```

```
  <3D Button> "OpenButtonUpPict", "OpenButtonDownPict", LEFT, ~
              doOpenDocmt("HD:SurfWriter folder:SampleReport"),~
              "OpenButtonB&WUpPict", "OpenButtonB&WDownPict"
```

```
<End Panel>
```

SEE ALSO

For information on using formats when placing buttons, see “Formatting Text and Objects in a Panel” beginning on page 10-84. For information on the <Define Event> and <Define Event List> commands, see page 10-178 and page 10-181, respectively.

<Radio Button>

You can use the <Radio Button> command to place a radio button on a panel. You must specify the button’s title and its default state (on or off). You can optionally specify a sequence to insert into the existing sequence based on the radio button’s state. You can also specify an anchor point for the radio button and the font of the button title.

```
<Radio Button> buttonTitle, buttonState [, seqTrue] [, seqFalse]
                [, buttonAnchor] [, buttonFont]
```

<i>buttonTitle</i>	A string specifying the radio button’s title.
<i>buttonState</i>	A Boolean constant indicating the default state of the radio button. Specify TRUE to set the radio button to on, specify FALSE to set the radio button to off. Only one radio button in a set can have its default state set to TRUE .
<i>seqTrue</i>	The name of the sequence to insert if the radio button is on (TRUE). If you specify a sequence name in this parameter, you should omit the <i>seqFalse</i> parameter. This parameter is optional.
<i>seqFalse</i>	The name of the sequence to insert if the radio button is off (FALSE). You usually insert a sequence only if the state of the radio button is true, thus you typically do not specify the <i>seqFalse</i> parameter. If you do specify this parameter, you should omit the <i>seqTrue</i> parameter. This parameter is optional.
<i>buttonAnchor</i>	A point that indicates the anchor point of the radio button. To describe a specific point, use the Point function. This parameter is optional.
<i>buttonFont</i>	A constant specifying the font of the button’s title. You can use either of these constants to specify the corresponding font:

APPLEGUIDE Specifies 10-point Espy Serif.

SYSTEM Specifies the system font of the current script system. For example, for Roman script systems, indicates 12-point Chicago.

This parameter is optional; if you omit this parameter Apple Guide displays the button title in the system font.

DESCRIPTION

The `<Radio Button>` command places a radio button with the specified title on a panel. If you specify the *seqTrue* or *seqFalse* parameter, Apple Guide inserts the specified sequence into the current sequence, according to the setting of the radio button. This lets you easily accomplish sequence branching. When you use this method, Apple Guide displays the panel containing the radio buttons; when the user navigates to another panel, Apple Guide inserts the named sequence according to the current setting of the radio buttons. Alternatively, you can explicitly check the radio button settings yourself, using the `<If>` command and the built-in function `radioButtonState`.

Radio buttons appear inline with the surrounding text and are justified within the current format unless you specify an optional anchor point.

Always provide in your panel a label identifying the group of choices that your radio buttons offer. You can place this label on the panel and then use the `<Radio Button>` command to place your radio buttons.

EXAMPLES

```
<Define Panel> "Index Choices"
  #label for this group of radio buttons
  What type of index do you want to create?

  <Radio Button> "Book Index", TRUE, ~
    "How do I create a book index?" ~
    ,, APPLEGUIDE
  <Radio Button> "Chapter Index", FALSE, ~
    "How do I create a chapter index?" ~
    ,, APPLEGUIDE
```

<End Panel>

#alternative method of checking settings of radio buttons

<Define Panel> "Index Choices 2"

 #label for this group of radio buttons

 What type of index do you want to create?

 <Radio Button> "Book Index", TRUE, , , , APPLEGUIDE

 <Radio Button> "Chapter Index", FALSE, , , , APPLEGUIDE

<End Panel>

<Define Sequence> "How do I create an index?"

 <Panel> "Index Choices 2"

 <If> radioButtonState("Book Index", "Index Choices 2")

 <Panel> "How do I create a book index?"

 <Else>

 <Panel> "How do I create a chapter index?"

 <End if>

<End Sequence>

SEE ALSO

For information on creating checkboxes, see the description of the <CheckBox> command on page 10-69. For information on other buttons, see the description of the <Standard Button> and <3D Button> commands on page 10-57 and page 10-60, respectively.

<Radio Button Launch New Seq>

You can use the <Radio Button Launch New Seq> command to place a radio button on a panel and to launch a new sequence according to the settings of the

radio buttons. You should usually use the `<Radio Button>` command instead of the `<Radio Button Launch New Seq>` command.

```
<Radio Button Launch New Seq> buttonTitle, buttonState
                                [, seqTrue] [, seqFalse]
                                [, buttonAnchor] [, buttonFont]
```

<i>buttonTitle</i>	A string specifying the radio button's title.
<i>buttonState</i>	A Boolean constant indicating the default state of the radio button. Specify TRUE to set the radio button to on, specify FALSE to set the radio button to off. Only one radio button in a set can have its default state set to TRUE .
<i>seqTrue</i>	The name of the sequence to launch if the radio button is on (TRUE). If you specify a sequence name in this parameter, you should omit the <i>seqFalse</i> parameter. This parameter is optional.
<i>seqFalse</i>	The name of the sequence to launch if the radio button is off (FALSE). You usually do not specify this parameter, but if you do, you should omit the <i>seqTrue</i> parameter. This parameter is optional.
<i>buttonAnchor</i>	A point that indicates the anchor point of the radio button. To describe a specific point, use the Point function. This parameter is optional.
<i>buttonFont</i>	A constant specifying the font of the button's title. You can use either of these constants to specify the corresponding font: APPLEGUIDE Specifies 10-point Espy Serif. SYSTEM Specifies the system font of the current script system. For example, for Roman script systems, indicates 12-point Chicago. This parameter is optional; if you omit this parameter Apple Guide displays the button title in the system font.

DESCRIPTION

The `<Radio Button Launch New Seq>` command places a radio button with the specified title on a panel. If you specify the *seqTrue* or *seqFalse* parameter, Apple Guide inserts the specified sequence into the current sequence, according to the setting of the radio button.

Guide Script Command Reference

The <Radio Button Launch New Seq> command displays the named sequence in the same window as the current sequence, and the user navigates to it using the right arrow. However, the user cannot navigate back to the original sequence using the left arrow after beginning the new sequence.

Radio buttons appear inline with the surrounding text and are justified within the current format unless you specify an optional anchor point.

Always provide in your panel a label identifying the group of choices that your radio buttons offer. You can place this label on the panel and then use the <Radio Button Launch New Seq> command to place your radio buttons.

SPECIAL CONSIDERATIONS

Use the <Radio Button Launch New Seq> command only if Guide Maker informs you that the current sequence exceeds limitations.

EXAMPLES

```
<Define Panel> "Index Choices"
  #label for this group of radio buttons
  What type of index do you want to create?

  #similar to <Radio Button> command, only sequences are
  # launched (user can't go back to original sequence)
  # rather than inserted
  <Radio Button Launch New Seq> "Book Index", TRUE, ~
      "How do I create a book index?"
  <Radio Button Launch New Seq> "Chapter Index", FALSE, ~
      "How do I create a chapter index?"
<End Panel>
```

SEE ALSO

For information on the <Radio Button> command, see page 10-64. For information on the <Launch New Sequence> command, see page 10-49.

<Checkbox>

You can use the <Checkbox> command to place a checkbox on a panel. You must specify the button's title and its default state (on or off). You can optionally specify a sequence to insert into the new sequence, based on the checkbox's state. You can also specify an anchor point for the checkbox and its title font.

```
<Checkbox> checkboxTitle, checkboxState [, seqTrue] [, seqFalse]
            [, checkboxAnchor] [, checkboxFont]
```

checkboxTitle A string specifying the checkbox's title.

checkboxState A Boolean constant indicating the default state of the checkbox. Specify **TRUE** to set the checkbox to on, specify **FALSE** to set the checkbox to off. Any number of checkboxes in a set can have their default state set to **TRUE**.

seqTrue The name of the sequence to insert if the checkbox is on (**TRUE**). This parameter is optional.

seqFalse The name of the sequence to insert if the checkbox is off (**FALSE**). You usually do not provide this parameter if you provide the *seqTrue* parameter. This parameter is optional.

checkboxAnchor A point that indicates the anchor point of the checkbox. To describe a specific point, use the **Point** function. This parameter is optional.

checkboxFont A constant specifying the font of the checkbox's title. You can use either of these constants to specify the corresponding font:

APPLEGUIDE Specifies 10-point Espy Serif.

SYSTEM Specifies the system font of the current script system. For example, for Roman script systems, indicates 12-point Chicago.

This parameter is optional; if you omit this parameter, Apple Guide displays the checkbox title in the system font.

DESCRIPTION

The `<Checkbox>` command places a checkbox with the specified title on a panel. If you specify the *seqTrue* or *seqFalse* parameter, Apple Guide inserts the specified sequence into the current sequence, according to the setting of the checkbox. You can easily accomplish sequence branching by using these two parameters. When you use this method, Apple Guide displays the panel containing any checkboxes; when the user navigates to another panel, Apple Guide inserts one or more named sequences according to the current setting of the checkboxes. Alternatively, you can explicitly check the settings of the checkboxes yourself, using the `<If>` command and the built-in function `checkBoxState`.

Checkboxes appear inline with the surrounding text and are justified within the current format unless you specify an optional anchor point.

Note that the titles of checkboxes should reflect two clearly opposite states, because a checkbox should allow a user to turn a particular setting on or off.

EXAMPLES

```
<Define Panel> "Index Choices"
```

```
    Index choices:
```

```
    <Checkbox> "Include See Also entries", TRUE, ~
                "How do I create See Also entries?"
```

```
    <Checkbox> "Include starting and ending page ranges", FALSE, ~
                "How do I create page ranges for an index entry?"
```

```
<End Panel>
```

```
#alternative method of checking settings of checkboxes
```

```
<Define Panel> "Index Choices 2"
```

```
    Index choices:
```

```
    <Checkbox> "Include See Also entries", TRUE
```

```
    <Checkbox> "Include starting and ending page ranges", FALSE
```

```
<End Panel>
```

```

<Define Sequence> "How do I create an index?"
  <Panel> "Index Choices 2"
  <If> checkBoxState("Include See Also entries", ~
    "Index Choices 2")
    <Panel> "How do I create See Also entries?"
  <End if>
  <If> checkBoxState("Include starting and ending page ranges", ~
    "Index Choices 2")
    <Panel> "How do I create page ranges for an index entry?"
  <End if>
<End Sequence>

```

SEE ALSO

For information on creating radio buttons, see the description of the `<Radio Button>` command on page 10-64. For information on other buttons, see the description of the `<Standard Button>` and `<3D Button>` commands on page 10-57 and page 10-60, respectively.

<Define Nav Button>

You can use the `<Define Nav Button>` command to define a navigation button and to specify an event function that Apple Guide calls when the user clicks the button.

```

<Define Nav Button> buttonName, buttonUpPict, buttonDownPict,
                   dimmedButtonPict, buttonEvent
                   [, b&wUp] [, b&wDown] [, b&wDimmed]

```

buttonName A text string specifying the name to associate with this button. Note that this name is not used by Apple Guide; you provide a button name only so that you can reference this button definition in the `<Default Nav Button Set>` or `<Seq Nav Button Set>` commands.

buttonUpPict A resource ID, resource name, or filename that identifies the picture that defines the appearance of the navigation button in its active state (when the button is active and is not pressed). If you specify a resource ID or resource name, you must make the resource available to Guide Maker using the <Resource> command. If you specify a filename, the file must be in the same folder as your source files in order for Guide Maker to find the file.

buttonDownPict A resource ID, resource name, or filename that identifies the picture that defines the appearance of the navigation button when the button is active, the cursor is in the button, and the user presses the mouse button. If you specify a resource ID or resource name, you must make the resource available to Guide Maker using the <Resource> command. If you specify a filename, the file must be in the same folder as your source files in order for Guide Maker to find the file.

dimmedButtonPict A resource ID, resource name, or filename that identifies the picture that defines the appearance of the navigation button when the button is inactive (dimmed). Only navigation buttons that specify the constant `DIMMABLE` (rather than an event function) in the *buttonEvent* parameter are dimmable. Dimmable navigation buttons are inactive by default. To make a dimmable navigation button active on a specific panel, use the <Dimmable Button Data> command in the panel's definition.

If you specify a resource ID or resource name, you must make the resource available to Guide Maker using the <Resource> command. If you specify a filename, the file must be in the same folder as your source files in order for Guide Maker to find it.

buttonEvent A name of an event function, event list, or a constant. Include any parameters expected by the function in parentheses following its name. You define event functions using the <Define Event> or <Define Event List> command. Guide Maker also provides built-in functions that you can specify in this parameter. When the user clicks the button, Apple Guide calls the function defined by this parameter; usually such a function is used to send a specific Apple event.

Rather than specifying an event function or event list, you can use the constant **DIMMABLE** in this parameter. Use the constant **DIMMABLE** to indicate that Apple Guide should launch a new sequence (as specified in a subsequent <Dimmable Button Data> command) when this button is active and the user clicks the button. You can specify only one dimmable navigation button per guide file. A navigation button that is defined using the constant **DIMMABLE** is called a dimmable navigation button.

- b&wUp* A filename that, if provided, Apple Guide uses in place of the button appearance described by the *buttonUpPict* parameter only if the bit depth of the user's monitor is set to 4 bits or less. This parameter is optional and can be used only if you also specified a filename for the *buttonUpPict* parameter. However, if you specify a resource ID or resource name in the *buttonUpPict* parameter, you can still provide a black-and-white picture by importing a 'PICT' graphic whose resource ID is one greater than the resource ID of the color graphic.
- b&wDown* A filename that, if provided, Apple Guide uses in place of the button appearance described by the *buttonDownPict* parameter only if the bit depth of the user's monitor is set to 4 bits or less. This parameter is optional and can be used only if you also specified a filename for the *buttonDownPict* parameter. However, if you specify a resource ID or resource name in the *buttonDownPict* parameter, you can still provide a black-and-white picture by importing a 'PICT' graphic whose resource ID is one greater than the resource ID of the color graphic.
- b&wDimmed* A filename that, if provided, Apple Guide uses in place of the button appearance described by the *dimmedButtonPict* parameter only if the bit depth of the user's monitor is set to 4 bits or less. This parameter is optional and can be used only if you also specified a filename for the *dimmedButtonPict* parameter. However, if you specify a resource ID or resource name in the *dimmedButtonPict* parameter, you can still provide a black-and-white picture by importing a 'PICT' graphic whose resource ID is one greater than the resource ID of the color graphic.

DESCRIPTION

The <Define Nav Button> command defines a navigation button that you can later associate with a sequence. To do this, after defining the navigation button, you must define a navigation button set by using the <Define Nav Button Set> command and then associate this navigation button set with a specific sequence using the <Default Nav Button Set> or <Seq Nav Button Set> commands.

The appearance of the button is determined by the *buttonUpPict*, *buttonDownPict*, and *dimmedButtonPict* parameters. In addition, you can specify replacement pictures using the *b&wUp*, *b&wDown*, and *b&wDimmed* parameters, which Apple Guide uses according to the bit depth of the user's monitor. Note that if Apple Guide displays a black-and-white button, it uses the frame created for the color button. All pictures describing the button's appearance should have the exact same size. Because the navigation bar is a fixed height and because navigation buttons should use the Apple Guide font, the navigation buttons you define should have a height of 18 pixels. The navigation bar has a fixed width and height, and the navigation arrows always appear on the right; therefore you should size and design your buttons to fit within this area. You should use 10-point Esy Serif font for any text in navigation buttons (for Roman script systems).

When creating a navigation button in a graphics application, make sure your button graphic doesn't contain any extra white space around the edges (such as a bounding box); that is, when the button is copied, the graphic should represent its precise shape. When Guide Maker places a button graphic in the navigation bar, it places the graphic and then fills the remaining area with the navigation bar background pattern.

Navigation buttons are always associated with a sequence and are displayed on every panel in the sequence.

For buttons that specify the **DIMMABLE** constant in the *buttonEvent* parameter, Apple Guide displays the navigation button in its inactive state. Therefore, for each panel definition that uses this kind of navigation button, you must include a <Dimmable Button Data> command on those panels in which the navigation button is active. Navigation buttons such as the Huh? button typically specify the **DIMMABLE** constant, as the Huh? button is often active on only a subset of panels in a sequence.

For buttons that specify event functions or event lists in the *buttonEvent* parameter, Apple Guide displays the navigation button in its active state by default. Therefore, for each panel definition that uses this kind of navigation

button, the button must always be active. Navigation buttons such as the GoStart button are typically always active.

Navigation buttons appear in the navigation bar, in the location determined by the <Define Nav Button Set> command.

The file Standard Setup is provided with Guide Maker. This file contains descriptions of the Huh? and GoStart navigation buttons, and defines three navigation button sets. If you include the Standard Setup file in your build file, you can automatically use these two navigation buttons or any of the three navigation button sets, as needed in your source files.

SPECIAL CONSIDERATIONS

If you do not explicitly specify a black-and-white picture for the button's up, down, or dimmed appearance, and the guide file contains a 'PICT' graphic whose resource ID is one greater than one of the button's color 'PICT' graphics, Apple Guide uses this 'PICT' graphic as the button's black-and-white picture.

EXAMPLES

```
#example that uses the Huh? button.
# The Huh? button is defined in the Standard Setup file as:
# (Standard Setup also contains the black-and-white versions for
# the button in 'PICT' resources with resource IDs 1102, 1112, 1122)
#<Define Nav Button> "Huh?", 1101, 1111, 1121, DIMMABLE

<Define Panel> "Panel 2"
#the Huh? button should be active on this panel,
# so use the <Dimmable Button Data> command
# (when this navigation button is active and the user clicks
# this button, Apple Guide launches a new sequence in a new window)
    <Dimmable Button Data> "Huh?", "Name of sequence to launch"
<End Panel>
```

```

<Define Sequence> "Sequence with Huh? button"
    #use this nav button set for this sequence
    <Seq Nav Button Set> "Huh? Only"
    <Panel> "Panel 1"      #Huh? button inactive
    <Panel> "Panel 2"      #Huh? button active
    <Panel> "Panel 3"      #Huh? button inactive
<End Sequence>

```

```

#example of a navigation button that uses the DIMMABLE constant
# (this kind of navigation button is inactive by default)

```

```

<Define Nav Button> "Why?", "upWhyPict", "downWhyPict", ~
                    "dimmedWhyPict", DIMMABLE, ~
                    "b&wUpWhyPict", "b&wDownWhyPict", ~
                    "b&wDimmedWhyPict"

```

```

#define a nav button set that uses the Why? button and
# specify the Why? button as the middle navigation button

```

```

<Define Nav Button Set> "Why Nav Button Set", ~
                        "GoStart", "Why?"

```

```

<Define Panel> "Panel 2"

```

```

#the Why? button should be active on this panel,

```

```

# so use the <Dimmable Button Data> command

```

```

# (when this navigation button is active and the user clicks

```

```

# this button, Apple Guide launches the sequence in a new window)

```

```

    <Dimmable Button Data> "Why?", "sequence to launch"

```

```

<End Panel>

```

```

#Another example of a navigation button

```

```

# This button uses an event function.

```

```

# This kind of button is always active.

```

```

# The GoStart button is defined in Standard Setup file as:

```

```

#<Define Nav Button> "GoStart", 1103, 1113, 1123, GoStart()

```

```

# the GoStart function is also defined in Standard Setup as:

```

```

#<Define Event> "GoStart", 's***', 'help', 'stac'

```



```

<Define Sequence> "Sequence with GoStart button"
  #use the GoStart Only nav button set for this sequence
  <Seq Nav Button Set> "GoStart Only"
  <Panel> "Panel A"      #GoStart button always active
  <Panel> "Panel B"      #GoStart button always active
<End Sequence>

```

#Another example of a navigation button.

#This button uses an event function,

this kind of button is always active.

```

<Define Nav Button> "Other", "upOtherPict","downOtherPict",~
                    "dimmedOtherPict", DoOther(), ~
                    "b&wUpOtherPict", "b&wDownOtherPict",~
                    "b&wDimmedOtherPict"

```

#define nav button set that uses an Other button

```

<Define Nav Button Set> "Other Nav Button Set", ~
                        "Other"

```

```

<Define Sequence> "Sequence with Other button"
  #use this nav button set for this sequence
  <Seq Nav Button Set> "Other Nav Button Set"
  <Panel> "Panel A"      #Other button always active
  <Panel> "Panel B"      #Other button always active
<End Sequence>

```

SEE ALSO

For information on <Define Nav Button Set>, <Default Nav Button Set>, or <Seq Nav Button Set> commands, see page 10-80, page 10-32, and page 10-43, respectively.

For information on the <Define Event> and <Define Event List> commands, see page 10-178 and page 10-181, respectively.

The <Dimmable Button Data> command is described next.

<Dimmable Button Data>

You can use the <Dimmable Button Data> command to specify that a dimmable navigation button should be active on a particular panel.

<Dimmable Button Data> *buttonName*, *sequenceName*

buttonName A string specifying the navigation button's name.

sequenceName The name of the sequence to launch when the user clicks the button defined by the *buttonName* parameter.

DESCRIPTION

The <Dimmable Button Data> command makes the navigation button specified by the *buttonName* parameter active. The navigation button is active only for a panel definition that includes this command.

You must first define the navigation button using the <Define Nav Button> command and specify the constant **DIMMABLE** in the *buttonEvent* parameter. You then associate the navigation button with a sequence using the <Default Nav Button Set> or <Seq Nav Button Set> commands.

EXAMPLES

#example that uses the Huh? button.

The Huh? button is defined in the Standard Setup file as:

<Define Nav Button> "Huh?", 1101, 1111, 1121, DIMMABLE

<Define Panel> "Panel 2"

#the Huh? button should be active on this panel,

so use the <Dimmable Button Data> command

(when this navigation button is active and the user clicks

this button, Apple Guide launches a new sequence in a new window)

 <Dimmable Button Data> "Huh?", "Name of sequence to launch"

<End Panel>

```

<Define Sequence> "Sequence with Huh? button"
  #use this nav button set for this sequence
  <Seq Nav Button Set> "Huh? Only"
  <Panel> "Panel 1"      #Huh? button inactive
  <Panel> "Panel 2"      #Huh? button active
  <Panel> "Panel 3"      #Huh? button inactive
<End Sequence>

```

#example of a navigation button that uses the DIMMABLE constant
(this kind of navigation button is inactive by default)

```

<Define Nav Button> "Why?", "upWhyPict", "downWhyPict", ~
                    "dimmedWhyPict", DIMMABLE, ~
                    "b&wUpWhyPict", "b&wDownWhyPict", ~
                    "b&wDimmedWhyPict"

```

#define a nav button set that uses the Why? button and
specify the Why? button as the middle navigation button

```

<Define Nav Button Set> "Why Nav Button Set", ~
                        "GoStart", "Why?"

```

```

<Define Sequence> "Sequence with Why? button"
  <Seq Nav Button Set> "Why Nav Button Set"
  <Define Panel> "Panel 2"
    #the Why? button should be active on this panel,
    # so use the <Dimmable Button Data> command
    # (when this navigation button is active and the user clicks
    # this button, Apple Guide launches the specified sequence
    # in a new window)
    <Dimmable Button Data> "Why?", "sequence name to launch"
  <End Panel>
<End Sequence>

```

SEE ALSO

For information on creating other buttons, such as standard buttons, radio buttons, or checkboxes, see “Creating Buttons” beginning on page 10-57.

<Define Nav Button Set>

You can use the <Define Nav Button Set> command to specify a set of up to three navigation buttons that can appear together in the navigation bar of each panel in a sequence.

```
<Define Nav Button Set> navButtonSetName
                        [ , leftNavButton ] [ , midNavButton ]
                        [ , rightNavButton ]
```

navButtonSetName

A string specifying the name of this navigation button set. This name must be unique from all other navigation button sets that you define.

leftNavButton A string specifying the name of the navigation button to place in the left position of the navigation bar. This parameter is required if you provide the *midNavButton* parameter.

midNavButton A string specifying the name of the navigation button to place in the middle position of the navigation bar. This parameter is required if you provide the *rightNavButton* parameter.

rightNavButton

A string specifying the name of the navigation button to place in the right position of the navigation bar (but to the left of the navigation arrows). This parameter is optional.

DESCRIPTION

The <Define Nav Button Set> command defines a set of navigation buttons and specifies their placement in the navigation bar. A navigation button can be placed in the left, middle, or right positions of the navigation bar. You use the <Define Nav Button> command to define the navigation buttons that you specify in the *leftNavButton*, *midNavButton*, and *rightNavButton* parameters.

To associate this navigation button set with a specific sequence, use the <Default Nav Button Set> or <Seq Nav Button Set> commands.

Note that any navigation buttons that you define using a <Define Nav Button Set> command appear to the left of the navigation arrows. Apple Guide always displays the navigation arrows on each panel. For each panel, Apple Guide

makes the right navigation arrow active or inactive according to whether the user can navigate to a following panel. Apple Guide makes the left navigation arrow active or inactive according to whether the user can navigate to a previous panel. Apple Guide also displays the panel number (as it appears in a sequence) between the left and right navigation arrows.

Because the navigation bar is a fixed height and because navigation buttons should use the Apple Guide font, the navigation buttons you define (in `<Define Nav Button>` commands) should have a height of 18 pixels. The navigation bar has a fixed width and height, and the navigation arrows always appear on the right; therefore you should size and design your buttons to fit within this area.

The file `Standard Setup` is provided with Guide Maker. This file contains descriptions of the `Huh?` and `GoStart` navigation buttons, and defines three navigation button sets:

- **“Standard Nav Bar”**. A navigation set that specifies the `GoStart` and `Huh?` navigation buttons as buttons in the navigation bar.
- **“GoStart Only”**. A navigation set that specifies only the `GoStart` navigation button as a button in the navigation bar.
- **“Huh? Only”**. A navigation set that specifies only the `Huh?` navigation button as a button in the navigation bar.

The `Standard Setup` file also defines the “Standard Nav Bar” button set as the default navigation button set. If you include the `Standard Setup` file in your build file, you can automatically use these two navigation buttons or any of the three navigation button sets as needed in your source files.

EXAMPLES

```
# The Huh? button is defined in the Standard Setup file as:
#<Define Nav Button> "Huh?", 1101, 1111, 1121, DIMMABLE
# The GoStart button is defined in the Standard Setup file as:
#<Define Nav Button> "GoStart", 1103, 1113, 1123, GoStart()
#define another nav button specific to this guide file
<Define Nav Button> "Another Nav Button", 2210, 2220, ↵
                    2230, DoNav()
```

#define a navigation button set with three nav buttons

```
<Define Nav Button Set> "My Nav Bar", -  
                        "GoStart", "Huh?", "Another Nav Button"
```

SEE ALSO

For information on the <Define Nav Button>, <Default Nav Button Set>, and <Seq Nav Button Set> commands, see page 10-71, page 10-32, and page 10-43, respectively.

Defining and Using Text Blocks

You can define text used with the <Howdy> command by using the commands described in this section.

<Define Text Block>

You can use the <Define Text Block> command to mark the beginning of a named block of text.

```
<Define Text Block> textBlockName
```

textBlockName

A string assigning the name of this text block. Each text block name must be unique from all other text block names that you define.

DESCRIPTION

The <Define Text Block> command defines the beginning of a named block of text. You indicate the end of the text block using the <End Text Block> command. After defining a text block with the <Define Text Block> and <End Text Block> commands, you can later reference this text block in those commands that accept named text blocks as parameters. For example, the <Howdy> command requires a named text block as a parameter.

EXAMPLES

```
#define a text block and give it a name
<Define Text Block> "Howdy Text"
    To start, click Topics, Index, or Look For.

    Topics shows general categories, ~
    and Index lists key words. ~
    Look For lets you search for help ~
    according to key words you type.

    To learn basic skills, choose Tutorial from the ? menu.
<End Text Block>

#specify name of the text block that defines the Howdy text
<Howdy> "Howdy Text"
```

SEE ALSO

For information on the <Howdy> command, see page 10-24. The <End Text Block> command is described next.

<End Text Block>

You can use the <End Text Block> command to mark the end of a named block of text.

```
<End Text Block>
```

DESCRIPTION

The <End Text Block> command marks the end of a text block definition. You mark the beginning of a named text block using the <Define Text Block> command. After defining a text block with the <Define Text Block> and <End Text Block> commands, you can later reference this text block in those

commands that accept named text blocks as parameters. For example, the <Howdy> command requires a named text block as a parameter.

EXAMPLES

```
#define a text block and give it a name
<Define Text Block> "Howdy Text"
    To start, click Topics, Index, or Look For.

    Topics shows general categories, ↵
    and Index lists key words. ↵
    Look For lets you search for help ↵
    according to key words you type.
<End Text Block>

#specify name of the text block that defines the Howdy text
<Howdy> "Howdy Text"
```

SEE ALSO

For information on the <Howdy> command, see page 10-24. The <Define Text Block> command is described on page 10-82.

Formatting Text and Objects in a Panel

If you don't use the <Default Format> or <Define Format> commands to specify one or more formats, Guide Maker applies its own default formatting when placing objects within panels. Guide Maker applies full-width panel formatting by default. This built-in default format uses the full panel width (less an 11-pixel margin at each side) as a single column. It uses the default text attributes 10-point Espy Serif plain, black. Guide Maker also aligns the prompt on a panel with the left edge of the text object that appears first in the panel definition; so by default, the prompt is aligned to the panel's left edge (less 11 pixels). Guide Maker uses these defaults as the panel's format for any panel text or objects that appear before a <Default Format> or <Format> command in your source file.

<Define Format>

You can use the <Define Format> command to define a format for either immediate or later use to place text and objects in a panel.

```
<Define Format> formatName, columnCoords [, txFmt] [, txSize]
                [, txStyle] [, txColor] [, txAlign] [, alignPrompt]
```

formatName A text string specifying the name of this format. Each format name must be unique from all other format names that you define.

columnCoords

A column specifier giving the top, left, and right coordinates that define the bounds for this format. All objects placed using a specific format are placed relative to the bounds of that format. For example, `Column(50,75,275)` defines a format whose bounds are defined by the column beginning with a top coordinate of 50 and a left coordinate of 75 that extends horizontally for 200 pixels. You don't need to specify the bottom coordinate, because Guide Maker automatically extends the bottom as you place objects on a panel. Any objects placed using this format are aligned within the defined bounds.

txFmt A text string that specifies a font name (such as "Palatino" or "Geneva"). This parameter is optional; provide this parameter only when you want to override the text font specified by text attributes of your source files.

txSize A short integer specifying the font size. This parameter is optional; provide this parameter only when you want to override the text size specified by text attributes of your source files.

txStyle A constant specifying the text style. You can specify only one of these constants:

```
PLAIN
BOLD
ITALIC
UNDERLINE
OUTLINE
```

SHADOW
CONDENSE
EXTEND

This parameter is optional; provide this parameter only when you want to override the text style specified by text attributes of your source files.

txColor

A constant or RGB specifier describing the text color. You can specify only one of these constants:

BLACK
YELLOW
MAGENTA
RED
CYAN
GREEN
BLUE
WHITE

To specify the color using RGB values, use the form `RGBColor (red, green, blue)`; for example: `RGBColor (30000, 30000, 30000)`.

This parameter is optional; provide this parameter only when you want to override the text color specified by text attributes of your source files.

txAlign

A constant that specifies the text alignment. You can specify only one of these constants:

LEFT
CENTER
RIGHT
SYSTEM

This parameter is optional; provide this parameter only when you want to override the text alignment specified by the default format.

alignPrompt

A Boolean constant (TRUE or FALSE) that specifies whether the prompt should be aligned to this format. This parameter is optional; provide this parameter only when you want to override the alignment of the prompt as specified by the default format.

DESCRIPTION

The `<Define Format>` command creates a column with certain attributes, which you can use to simplify the layout of your panel objects. You can use formats to control the placement of most panel objects:

- paragraphs of text
- checkboxes and radio buttons
- 'PICT' graphics and QuickTime movies
- labels for graphics

You can specify a format that you define with the `<Define Format>` command in a `<Format>` or `<Default Format>` command.

Panel text objects that follow a `<Format>` command are placed inline within the bounds specified by the format's column and according to the format's attributes. For example, a `<PICT>` command that uses `LEFT` as the picture's location is aligned at the left edge of the format's column, not at the left edge of the panel. Text placed after a `<Format>` command uses the text attributes defined by the specified format. Any text attributes specified in a `<Define Format>` command and then used by a `<Format>` command override Guide Maker's default formatting or any default formatting you previously described with the `<Default Format>` command.

If your source file is plain text and you don't specify a default format, or if you define a default format (using the `<Default Format>` command) that doesn't specify a particular text attribute, Guide Maker uses its own default for that text attribute unless you override the default using a `<Define Format>` command. If you omit any of the `txFont`, `txSize`, `txStyle`, or `txColor` parameters in a `<Define Format>` command (and your source file is plain text), Guide Maker uses default text attributes of Esy Serif, 10-point, plain, and black, accordingly.

If your source file is styled text, by default Guide Maker uses the font, size, style, and color of the text as it appears in the source file. However, when you define a format you can override one or more of these text attributes. If a format specifies a text attribute, Guide Maker uses that text attribute rather than the one specified in your source files. For example, if a format specifies the font size as 10 and the style as bold, then all text that uses that format will appear with a size of 10 and in the bold style, regardless of the size and style of the text in the source file. That same text retains its font and color from the source file.

By default, Apple Guide aligns the prompt on a panel with the left edge of the text object that appears first in the panel definition. To align the prompt with another text object, specify `TRUE` in the *alignPrompt* parameter of the `<Define Format>` command for the format used by that text object. If more than one format on a panel has its *alignPrompt* parameter set to `TRUE`, Guide Maker reports an error. If a panel does not contain a text object, Apple Guide aligns the prompt to Guide Maker's default format (11 pixels in from the left edge of the panel).

Some Guide Script commands allow you to specifically place an object by specifying a coordinate location using the `Point` function. When you specifically place an object in this way, specify the object's location relative to the current pen location. The current pen location's horizontal coordinate is the left edge of the current format; its vertical coordinate corresponds to the bottom edge of the last object not specifically placed using coordinates. For example, to place a button 50 pixels to the right and 20 pixels down from the current pen location, specify `Point(50,20)` as a parameter to the `<Standard Button>` command. Guide Maker does not reset the current pen location after placing an object that specified coordinates. This feature allows you to place objects relative to a known location (the current pen location). Guide Maker does change the current pen location as it places objects that don't specify coordinates; however, note that the horizontal coordinate of the current pen location always refers to the left edge of the current format.

The file `Standard Setup` is provided with Guide Maker. This file defines four formats:

- **Tag.** A format that provides a left column that you can use to format tags, that is, text such as "Do This" or "Oops".
- **Body.** A format that provides a right column and that is designed for use with the Tag format. Use the Body format to provide the information that goes in the right column of a panel that also has a tag in it.
- **Full.** A format that provides a full column. Use this format if your panel text requires a full-column width.
- **ResetPen.** A format that resets the format to a default format.

If you include the `Standard Setup` file in your build file, you can automatically use these formats as needed in your source files.

EXAMPLES

```
#specifies a format with column coordinates of
# top = 6, left = 0, and right = 54
# and text attributes of Espy Sans Bold, 10 point, Plain,
# default text color, and right aligned.
# And does not override the default alignment of the prompt.
# (the "Tag" format is defined in the Standard Setup file)
<Define Format> "Tag", Column(6, 0, 54), "Espy Sans Bold", 10, ~
                PLAIN, , RIGHT, FALSE
```

```
#specifies a format with column coordinates of
# top = 6, left = 65, and right = 330
# and text attributes of Espy Serif, 10 point, Plain,
# default text color, and left aligned.
# And the prompt should be aligned to leftmost edge of this format.
# (the "Body" format is defined in the Standard Setup file)
<Define Format> "Body", Column(6, 65, 330), "Espy Serif", 10, ~
                PLAIN, , LEFT, TRUE
```

```
#use the "Tag" and "Body" formats in a panel
<Define Panel> "Some Panel"
    <Format> "Tag"
    Do This
    <Format> "Body"
    Give instruction here.
<End Panel>
```

```
#specifies a format with column coordinates of
# top = 0, left = 0, and right = 250
# and default text attributes (Espy Serif, 10 point, Plain, Black)
# and default alignments
<Define Format> "Reset", Column(0, 0, 250)
```

```
#specifies a format with
# text attributes of Palatino, 12 point, Bold, Red text color,
# and left aligned. The prompt should be aligned to the
# leftmost edge of this format (aligned to pixel 50).
<Define Format> "Left", Column(50, 50, 125), "Palatino", 12, ~
    BOLD, RED, LEFT, TRUE
```

```
#specifies a format with
# text attributes of Palatino, 12 point, Bold, default text color,
# and right aligned.
<Define Format> "Right", Column(50, 150, 330), "Palatino", 12, ~
    BOLD, , RIGHT, FALSE
```

```
#specifies a format with
# text attributes of Palatino, 12 point, Bold, RGB values
# for the text color, and right aligned.
<Define Format> "Right2", Column(50, 150, 330), "Palatino", 12, ~
    BOLD, RGBColor(30000,30000,30000), RIGHT, FALSE
```

```
<Define Panel> "An example panel with buttons"
    <Format> "Tag"
    Do This
    <Format> "Body"
    Give instruction here.
    #the current pen location at this point is
    # for the x coordinate the left edge of the "Body" format
    # and for the y coordinate the bottom of the last placed text
    # this button is placed 50 pixels to the right and 20 pixels
    # down of the current pen location
    <Standard Button> "Some button", Point(50,20), doAction()
    #the current pen location is not reset
    # the next button is placed 50 pixels to the right and 80 pixels
    # down of the current pen location
    <Standard Button> "Another button", Point(50,80), doAction2()
    #the current pen location is not reset
```

```

Here's some more instruction for the panel.
This text appears starting from the current pen location,
so it could potentially overlap "Some button"
#now the vertical pen location has changed to account for
# the just placed text
#the last button is placed using Guide Maker's formatting,
# requesting the button be placed on the right in the
# current format
<Standard Button> "Last button", RIGHT, doAction3()
<End Panel>

```

SEE ALSO

For information on the <Default Format> command, see page 10-30. For information on the <Format> command, see page 10-93.

<Define Transparent Format>

You can use the <Define Transparent Format> command to define a transparent format for either immediate or later use to place text and objects in a panel.

```

<Define Transparent Format> formatName, columnCoords
                        [, txFmt] [, txSize] [, txStyle]
                        [, txColor] [, txAlign]
                        [, alignPrompt]

```

The parameters of the <Define Transparent Format> command are identical to those of the <Define Format> command. See the <Define Format> command beginning on page 10-85 for a complete description of the command parameters.

DESCRIPTION

The `<Define Transparent Format>` command provides one more feature than the `<Define Format>` command. Panel text formatted according to a format defined by a `<Define Transparent Format>` command can overlap other panel contents; if the text does overlap, the part of its background that overlaps appears transparent.

You can specify a transparent format that you define with the `<Define Transparent Format>` command in a `<Format>` or `<Default Format>` command.

SPECIAL CONSIDERATIONS

If your panel definition contains multiple formats and multiple text objects and you use a transparent format that specifies `TRUE` in the *alignPrompt* parameter, the text object using this format might not appear to be transparent. In this case, rewrite your panel content so that the transparent text object appears as the first text object in the panel definition.

EXAMPLES

```
#specifies a transparent format (text can overlap format bounds)
# and text attributes of Palatino, 12 point, Bold, Red text color,
# and centered
<Define Transparent Format> "PicTitle", Column(50, 50, 125), ~
                                "Palatino", 12, BOLD, ~
                                RED, CENTER, FALSE
<Define Panel> "Another example panel"
    <PICT> 2528, CENTER
    <Format> "PicTitle"
    Figure caption
<End Panel>
```

SEE ALSO

For information on the `<Define Format>` command, see page 10-85. For information on the `<Default Format>` command, see page 10-30. The `<Format>` command is described next.

<Format>

You can use the <Format> command to apply a specific format to any commands following it that place text and objects in a panel. The specified format applies until the next <Format> or <End Panel> command.

<Format> *formatName*

formatName A text string specifying the name of the format or transparent format to apply to any commands that follow the <Format> command and place text and objects in a panel.

DESCRIPTION

When Guide Maker encounters a <Format> command, it applies the format defined by *formatName* to any commands following it that place text and objects in a panel until the next <Format> or <End Panel> command is encountered. The format specified by a <Format> command overrides all other defined formats, including any format specified by the <Default Format> command.

Panel objects that follow a <Format> command are placed inline within the bounds specified by the format's column coordinates. Any text attributes specified by the format are applied to panel text, and any prompts are aligned according to the specified format.

Any panel text or objects that appear before a <Format> command are aligned according to the format specified by a <Default Format> command or previous <Format> command. If a <Default Format> or <Format> command has not been specified, then panel text and objects are aligned according to Guide Maker's default full-width panel format.

Once Guide Maker encounters a <Format> command, the new format is applied to all following panel text and objects in that panel, until another <Format> command or <End Panel> command is encountered. Guide Maker resets the format to the default format upon encountering an <End Panel> command.

Thus, you can use the <Default Format> command to specify a format that Guide Maker uses as a default for all panels, and then you can override the default format for a specific panel as needed by using the <Format> command.

EXAMPLES

```
<Define Panel> "Example panel"  
  <Format> "Column1"  
  Text that is formatted according to "Column1" format.  
  <Format> "Column2"  
  Text that is formatted according to "Column2" format.  
<End Panel>  
  
<Define Panel> "Example panel 2"  
  Text that is formatted using the default format.  
<End Panel>  
  
<Define Panel> "Example panel 3"  
  #panel that uses the "tag" and "body" formats  
  <Format> "Tag"  
  Do This  
  <Format> "Body"  
  Give instruction here.  
<End Panel>
```

SEE ALSO

For information on the <Define Format> command, see page 10-85. For information on the <Default Format> command, see page 10-30.

Specifying Pictures and Movies

You can place pictures and movies in the content area of a panel by using the commands described in this section.

<PICT>

You can use the <PICT> command to specify a picture in a panel.

<PICT> *pictGraphic*, *location* [, *b&wPict*]

pictGraphic A resource ID, resource name, or filename that identifies the picture to place on the panel. If you specify the picture by resource ID or resource name, you must make the resource available to Guide Maker using the <Resource> command. If you specify a filename, the file must be in the same folder as your source files in order for Guide Maker to find it.

location A constant specifying the general location of the picture or a specific point describing the coordinates of the picture relative to the current pen location.

To specify a specific point (relative to the current pen location), use the **Point** function. The current pen location's horizontal coordinate is the left edge of the current format; its vertical coordinate corresponds to the bottom edge of the last object not specifically placed using coordinates.

You can use these constants to describe the picture location:

LEFT

CENTER

RIGHT

If you specify one of these constants, Guide Maker justifies the picture accordingly within the current format.

b&wPict A filename that, if provided, Apple Guide uses in place of the 'PICT' graphic described by the *pictGraphic* parameter only if the bit depth of the user's monitor is set to 4 bits or less. If you provide a black-and-white picture, be sure that it does not contain any color information. This parameter is optional and can be used only if you also specified a filename for the *pictGraphic* parameter. However, if you specify a resource ID or resource name in the *pictGraphic* parameter, you can still provide a black-and-white picture by importing a 'PICT' graphic whose resource ID is one greater than the resource ID of the color graphic.

DESCRIPTION

The <PICT> command places a picture on a panel. The picture's appearance is determined by the *pictGraphic* and *location* parameters. In addition, you can specify a replacement picture using the *b&wPict* parameter, which Apple Guide uses according to the bit depth of the user's monitor. All pictures describing the graphic's appearance should have the exact same size. Note that if Apple Guide displays a black-and-white picture, it uses the frame created for the color picture. In general, you should always provide both a color and black-and-white version of the picture.

The picture appears inline with the surrounding text. To specify the picture's general location use the constants **LEFT**, **CENTER**, or **RIGHT**. If you do this, Guide Maker justifies the picture according to the current format. You can also specify the picture's location relative to the current pen position by specifying a specific point. For example, if you specify the picture location as `Point(50, 100)`, Guide Maker positions the picture 50 pixels to the right and 100 pixels down from the current pen location.

Note that Guide Maker does not change the current pen position if you place an object using a point specifier. For example, if the current pen location is at (25, 25) (indicating the left edge of the current format and the bottom edge of the last object that was not specifically placed), and you specify the picture's location as `Point(50, 100)`, Guide Maker places the picture as just described. If you then place text, the text appears starting at location (25, 25); after placing the text, Guide Maker updates the pen location to account for the placed text.

Guide Maker searches for the resource specified by the *pictGraphic* parameter in this manner. If the *pictGraphic* parameter contains a number, Guide Maker searches for a resource with the specified resource ID. If the parameter contains a name, Guide Maker looks first for a resource using the specified string as the resource name. If it fails to find such a resource, it then looks for a file using the specified string as the filename.

If the resource is contained in a file, Guide Maker imports the 'PICT' resource from the file referenced by the parameter *pictGraphic* and assigns it a resource ID. If provided, Guide Maker also imports the 'PICT' resource from the file referenced by the parameter *b&wPict* and assigns it a sequential resource ID. Note that the files containing the graphics should be located in the same folder as your help sources.

If you specify a color 'PICT' by resource ID or resource name, use a resource editor to assign it a resource ID greater than 2000 and also mark the resource as

purgeable, then import it using the <Resource> command. Apple Guide reserves the use of 'PICT' resources with resource IDs less than 2000.

SPECIAL CONSIDERATIONS

Resource IDs 501 and 502 are reserved (for the application's color logo and black-and-white logo, respectively).

In general, in addition to a color picture you should always explicitly specify a black-and-white picture (by filename or by importing a 'PICT' graphic whose resource ID is one greater than the color 'PICT' graphic). If you do not explicitly specify a black-and-white picture and the guide file happens to contain a 'PICT' graphic whose resource ID is one greater than the color 'PICT' graphic, Apple Guide uses this 'PICT' graphic as the black-and-white picture.

EXAMPLES

```
#import resources used in this panel
<Resource> "MyResources", 'PICT'
<Resource> "MyResources2", 'PICT', 2528
<Define Panel> "Example Panel"
    To accomplish this task, do this:
    #very informative instructions here
    #place a picture on the panel (specified by filename)
    # and also include B&WPIC1 filename
    <PICT> "ColorPic1 File", CENTER, "B&WPic1 File"

    #place another picture on the panel (specified by res name)
    <PICT> "ColorPic2Resource", CENTER
    #place a picture on the panel (specified by resource ID)
    <PICT> 2528, CENTER
    #place a picture on the panel (specified by resource ID), and
    # specify its placement relative to the current pen location
    # as 75 pixels to the right, 30 pixels down
    <PICT> 2530, Point(75, 30)
<End Panel>
```

SEE ALSO

For information on the <Resource> command, see page 10-101.

<QuickTime>

You can use the <QuickTime> command to specify a QuickTime movie in a panel.

<QuickTime> *QTMovie*, *location*, *QTcontroller* [, *moviePict*]

QTMovie A filename of the file containing the movie to place on the panel. This file must be located in the same folder as your source files when you compile your guide file. A guide file and any movies used by the guide file must reside in the same folder.

location A constant specifying the general location of the movie or a specific point describing the coordinates of the movie relative to the current pen location.

To describe a specific point (relative to the current pen location), use the `Point` function. The current pen location's horizontal coordinate is the left edge of the current format; its vertical coordinate corresponds to the bottom edge of the last object not specifically placed using coordinates.

You can use these constants to describe the movie location:

LEFT

CENTER

RIGHT

If you specify one of these constants, Guide Maker justifies the movie accordingly within the current format.

QTcontroller A constant specifying the type of controller to use with the QuickTime movie. You can use these constants to describe the type of controller:

CONTROL

BADGE

PLAIN

Use the **CONTROL** constant to display the standard movie controller with the movie. The user can use the elements in the movie controller to control the playback of the movie. Use the **BADGE** constant to display a badge, a visual element that appears on the movie when the controller is not showing or the movie is not playing. When the user double-clicks a badge, a controller appears. Use the **PLAIN** constant if you do not want to provide a controller or a badge with the movie. Regardless of which constant you specify, the user can double-click the movie to play the movie, and click again to stop playing the movie.

moviePict A resource ID or resource name of a 'PICT' graphic, or a filename of a file containing a 'PICT' graphic, that Apple Guide uses in place of the QuickTime movie described by the *QTMovie* parameter only if QuickTime is not loaded or the movie file cannot be found. The replacement picture should have the same size as the QuickTime movie. This parameter is optional.

If you specify the picture by resource ID or resource name, you must make the resource available to Guide Maker using the `<Resource>` command. If you specify a filename, the file must be in the same folder as your source files in order for Guide Maker to find the file.

DESCRIPTION

The `<QuickTime>` command places a QuickTime movie on a panel.

The movie appears inline with the surrounding text. To specify the movie's general location use the constants **LEFT**, **CENTER**, or **RIGHT**. If you do this, Guide Maker justifies the movie according to the current format. You can also specify the movie's location relative to the current pen position by specifying a specific point. For example, if you specify the movie location as `Point(50, 100)`, Guide Maker positions the movie 50 pixels to the right and 100 pixels down from the current pen location.

SPECIAL CONSIDERATIONS

You can use only one QuickTime movie per panel. In Apple Guide, the amount of memory available for a QuickTime movie is limited; large movies might exceed the limit.

EXAMPLES

```
<Define Panel> "Example Panel With Movie 1"
    #place a QuickTime movie on the panel (specified by filename),
    # centered in current format, display badge,
    # and specify replacement picture by filename
    <QuickTime> "My QT Movie", CENTER, BADGE, "My Movie PICT"
<End Panel>

<Resource> "My Movie Pict 2", 'PICT', "Movie Pict"
<Define Panel> "Example Panel With Movie 2"
    #place a QuickTime movie on the panel (specified by filename),
    # centered in current format, display controller,
    # and specify replacement picture by resource name
    <QuickTime> "My QT Movie", CENTER, BADGE, "Movie Pict"
<End Panel>

<Resource> "My Pict for Movie", 'PICT', 2228
<Define Panel> "Example Panel With Movie 3"
    #place a QuickTime movie on the panel (specified by filename),
    # centered in current format, no controller and no badge,
    # and specify replacement picture as resource w/ res ID 2228
    <QuickTime> "My QT Movie", CENTER, PLAIN, 2228
<End Panel>
```

SEE ALSO

The <Resource> command is described in the next section.

Importing Resources

You can import resources such as pictures, movies, and external modules into your guide file by using the commands described in this section. Guide Maker makes the resources that you import available for use by other Guide Script commands.

<Resource>

You can use the <Resource> command to specify a file containing one or more resources to be included in your guide file. Use this command to include all resources in a file, all resources of a given resource type, all resources with a given resource ID or resource name, or a single resource.

<Resource> *fileName*, *resType* [, *whichResource*]

<i>fileName</i>	A filename of the file containing the resource to import. This file must be located in the same folder as your source files when you compile your guide file. A guide file and any resources used by the guide file must reside in the same folder.
<i>resType</i>	A four-character value specifying the resource type or the constant ALL. Use the constant ALL to include all resources in the file in your guide file. Specify a specific resource type to include only resources of that type.
<i>whichResource</i>	A resource ID or resource name of a resource. The resource included depends on the value that you specify in the <i>resType</i> parameter. To include a single resource, specify the resource's resource type in the <i>resType</i> parameter and the resource ID or resource name of the specific resource in the <i>whichResource</i> parameter. If you specify the resource's resource type in the <i>resType</i> parameter and omit the <i>whichResource</i> parameter, all resources of that type are included. This parameter is optional.

DESCRIPTION

When Guide Maker encounters a <Resource> command, it opens the file referenced by the *fileName* parameter and reads in the resources described by the *resType* and *whichResource* parameters. These resources are then compiled into the guide file and are available for use by other Guide Script commands, such as the <PICT> and <QuickTime> commands.

The file Standard Resources is provided with Guide Maker. This file contains 'PICT' resources that define the Continue, Huh?, and GoStart buttons and provides templates (the 'PICT' resources with IDs 501 and 502) that you can use to create your application logo picture. It also contains 'extm' resources; these are external modules (the Standard Setup file contains corresponding

<Define Context Check> commands for each external module). External modules referenced through a <Define Context Check> command can be used to specify a condition in <Make Sure> and other commands.

SPECIAL CONSIDERATIONS

For the resources that you plan to use in your guide file, you should assign each resource a resource ID between 2000 and 20,000. Apple Guide reserves the use of resource IDs less than 2000.

If you specify a <Resource> command, it must appear outside of any sequence or panel definitions.

For 'PICT' resources, you can provide a color version and a black-and-white version. You should assign the black-and-white 'PICT' a resource ID that is one greater than the resource ID of its corresponding color graphic.

EXAMPLES

```
#read in all resources of type 'PICT' in the file "All My Picts"
<Resource> "All My Picts", 'PICT'
```

```
#read in the 'PICT' resource with resource ID 2218
<Resource> "My Picture", 'PICT', 2218
```

```
#read in all resources from the file "Lots of Resources"
<Resource> "Lots of Resources", ALL
```

```
#read in the 'PICT' resource with resource ID 2228
# from the file "My Pict for Movie"
<Resource> "My Pict for Movie", 'PICT', 2228
#this 'PICT' resource is now available for use
<Define Panel> "Example Panel With Movie 3"
    #place a QuickTime movie on the panel and specify
    # replacement picture as 'PICT' resource with res ID 2228
    <QuickTime> "My QT Movie", CENTER, PLAIN, 2228
<End Panel>
```

```
#read in resources from the file "Standard Resources"
<Resource> "Standard Resources", ALL
```

SEE ALSO

For information on the <PICT> command, see page 10-95. For information on the <QuickTime> command, see page 10-98.

<Starting Res Number>

You can use the <Starting Res Number> command to specify the beginning resource number that Guide Maker should use when numbering any resources that it creates.

<Starting Res Number> *resID*

resID A short integer specifying the beginning resource ID that Guide Maker should use for resources that it creates for your guide file. This number must be between 2000 and 20,000.

DESCRIPTION

If you specify a <Starting Res Number> command, then for resources that Guide Maker creates, it begins numbering them using the specified resource ID.

For resources specified by filename in all Guide Script commands except the <Resource> command, Guide Maker reads the resource and assigns it a new resource ID, based on the value you specify in the <Starting Res Number> command.

For resources that you import in a <Resource> command or for resources that you specify by resource ID or resource name in other Guide Script commands, Guide Maker does not assign the resource a new resource ID. When Guide Maker encounters a <Resource> command, it opens the file referenced by the *fileName* parameter and reads in the resources described by the *resType* and *whichResource* parameters. It uses the resource's assigned resource ID and does not change or reassign it. These resources are then compiled into the guide file

and are available for use by other Guide Script commands, such as the <PICT> and <QuickTime> commands.

Using a <Starting Res Number> command can be useful when your guide file includes a Mixin guide file; using this command can help prevent resource ID conflicts between resources created by Guide Maker for your main guide file and resources created when you compile a Mixin guide file.

SPECIAL CONSIDERATIONS

The <Starting Res Number> command should appear once at most in your source files for a specific guide file. If you use this command, it must appear as the first command specified in the first file referenced by your build file. The <Starting Res Number> command is valid only for main guide files.

EXAMPLES

```
#Build file for SurfWriter guide file
<Include> "Initial Setup"
<Include> "Panel and Sequence definitions"
#End of Build file for SurfWriter guide file
```

```
#In the file named "Initial Setup"
#for resources specified by filename, assign resource IDs
# beginning with 3000
<Starting Res Number> 3000
#read in the 'PICT' resource with resource ID 2228
# from the file "My Pict for Movie"
<Resource> "My Pict for Movie", 'PICT', 2228
#End of "Initial Setup"
```

```
#In the file named "Panel and Sequence definitions"
<Define Panel> "Example Panel With Movie 3"
    #place a QuickTime movie on the panel and specify
    # replacement picture as 'PICT' resource with res ID 2228
    <QuickTime> "My QT Movie", CENTER, PLAIN, 2228
```

```
#place a picture on the panel (specified by filename)
#Guide Maker gets the color pict from the file and
# assigns it a new resource ID; it assigns the b&w pict
# the next sequential resource ID
<PICT> "ColorPic1 File", CENTER, "B&WPic1 File"
<End Panel>
```

SEE ALSO

For information on the <Resource> command, see page 10-101. For information on specifying a starting resource number for a Mixin guide file, see the description of the <Mixin> command, on page 10-19.

Creating Coachmarks

You can define coachmarks for menus, dialog items, and objects in a window by using the commands described in this section.

<Define Menu Coach>

You can use the <Define Menu Coach> command to define a menu coach for a specific menu and menu item.

```
<Define Menu Coach> coachMarkName [, targetApp] [, coachStyle]
                    [, targetMenu] [, targetItem]
                    [, itemCoachColor] [, itemCoachStyle]
```

coachMarkName

A text string specifying the name of this menu coach.

targetApp

A four-character sequence specifying the signature of the target application or the constant **FRONT** to specify the frontmost application. This parameter is not required. If you omit this parameter, Apple Guide uses **FRONT** as the default.

Guide Script Command Reference

<i>coachStyle</i>	A value indicating the coach style to use for the menu. You specify how Apple Guide should draw the coachmark for the menu using the constant REDCIRCLE or REDUNDERLINE . The <i>coachStyle</i> parameter is optional. If you omit this parameter, Apple Guide uses REDCIRCLE as the default.
<i>targetMenu</i>	The menu title or number of the menu associated with this coachmark. Apple Guide numbers the menus from left to right, beginning with the Apple menu which has a number of 1.
<i>targetItem</i>	The menu item name or number of the menu item associated with this coachmark. Menu items are numbered beginning with 1 for the first menu item. This parameter is optional. If you omit this parameter, Apple Guide draws a coachmark for the menu specified in the <i>targetMenu</i> parameter but does not use a coachmark for any menu item.
<i>itemCoachColor</i>	<p>A constant that specifies a color for the menu item coach. You can use one of these constants to specify the corresponding color:</p> <p>BLACK BLUE CYAN GREEN MAGENTA RED WHITE YELLOW</p> <p>The <i>itemCoachColor</i> parameter is optional. If you omit this parameter and specify a menu item coach in the <i>targetItem</i> parameter, Apple Guide uses a default of RED.</p>
<i>itemCoachStyle</i>	<p>A constant that specifies a text style for the menu item coach. You can use one of these constants to specify the corresponding text style:</p> <p>PLAIN BOLD CONDENSE EXTEND ITALIC</p>

OUTLINE
SHADOW
UNDERLINE

The *itemCoachStyle* parameter is optional. If you omit this parameter and specify a menu item coach in the *targetItem* parameter, Apple Guide uses a default of PLAIN.

DESCRIPTION

The <Define Menu Coach> command defines a menu coach. You associate a menu coach with a particular panel using the <Coach Mark> command. When Apple Guide opens a panel that includes a <Coach Mark> command that names a defined menu coach, Apple Guide uses the specified coach style and coach color to draw a coachmark for the specified menu. When the user pulls down the menu, Apple Guide uses the specified color and text style for the specified menu item.

SPECIAL CONSIDERATIONS

The display of menu coaches can be unpredictable if additional, unexpected menus have been inserted into the menu bar by system extensions, especially if the menu is referenced by number rather than by name.

Apple Guide may not be able to display menu coaches for applications that use custom menu definition procedures or a custom menu bar definition function.

Menu coaches do not appear if the target application is not active.

EXAMPLES

```
#define a menu coach for the Close command in the File menu
<Define Menu Coach> "FileCloseCoach", 'WAVE', REDCIRCLE, ~
    "File", "Close", RED, UNDERLINE
<Define Panel> "closing documents"
    <Coach Mark> "FileCloseCoach"
    To close a document, select Close from the File menu.
<End Panel>
```

```
#define a menu coach for the Open command in the File menu
#to specify an item with ellipsis, use Option-semicolon
<Define Menu Coach> "FileOpenCoach", 'WAVE', REDCIRCLE, ~
    "File", "Open...", RED, UNDERLINE
<Define Panel> "opening documents"
    <Coach Mark> "FileOpenCoach"
    To open a document, select Open... from the File menu.
<End Panel>
```

SEE ALSO

For information on the <Coach Mark> command, see page 10-118.

<Define Item Coach>

You can use the <Define Item Coach> command to define a coachmark for an item in a dialog box or for some other interface element that can be described by a help balloon rectangle.

```
<Define Item Coach> coachMarkName [, targetApp] [, coachStyle]
                    [, targetWindow] , targetItem
                    [, itemRectangle]
```

coachMarkName

A text string specifying the name of this item coach.

targetApp

A four-character sequence specifying the signature of the target application or the constant `FRONT` to specify the frontmost application. In general, you should always specify the signature of the target application to ensure that the coachmark is directed to the correct application. This parameter is optional. If you omit this parameter, Apple Guide uses `FRONT` as the default.

coachStyle

A value indicating the coach style to use for the item. You specify how Apple Guide should draw the coachmark for the item using a constant or red arrow specifier:

Constant

REDCIRCLE, REDUNDERLINE, or GREENX

Red arrow specifier

RedArrow(*start*, *end*)

Use the **RedArrow** function to draw a red arrow beginning at a location specified by *start* and ending at a location specified by *end*. The values for *start* and *end* are integers 1 through 8, where each value indicates a general location in a rectangle:

Location	Value
Top left	1
Top center	2
Top right	3
Middle right	4
Bottom right	5
Bottom center	6
Bottom left	7
Middle left	8

These values indicate the eight possible points for the start and end of the arrow. These locations can be interpreted as:

1	2	3
8		4
7	6	5

The *coachStyle* parameter is optional. If you omit this parameter, Apple Guide uses REDCIRCLE as the default.

- targetWindow* The window associated with this coachmark. For dialog items, you can specify the window's title, the constant FRONTWINDOW, or the constant DESKTOP. For items associated with help-balloon rectangles, if you provide this parameter you must specify the constant FRONTWINDOW. This parameter is optional. If you omit this parameter, Apple Guide uses FRONTWINDOW as the default.
- targetItem* An item specifier for an item in a dialog box or an ID specifier for a help balloon. You must provide the *targetItem* parameter.

To specify the rectangle of an item in a dialog box, use the built-in function `DialogID(itemNo)`. To specify the rectangle of an item that has a help balloon, use the built-in function `BalloonID(balloonNo)`.

itemRectangle

A subrectangle, relative to the upper-left of the item's rectangle, that further specifies the area in which the coachmark should be drawn. This parameter is optional. If you omit this parameter, Apple Guide draws the coachmark according to the item's rectangle.

DESCRIPTION

The <Define Item Coach> command defines an item coach for an item in a dialog box or an interface element in a window (or dialog box) that already has a help balloon associated with it. You associate an item coach with a particular panel using the <Coach Mark> command. When Apple Guide opens a panel that includes a <Coach Mark> command naming a defined item coach, Apple Guide uses the specified coach style to draw a coachmark for the specified item.

To provide a coachmark for an item in a dialog box, use the `DialogID` function to specify the item's item number in the *targetItem* parameter. (You can determine an item's item number using a resource editor, such as ResEdit.)

To provide a coachmark for an element in a window or dialog box that already has a help balloon associated with it, use the `BalloonID` function to specify the index number of the help balloon (corresponding to its position in the help resource) in the *targetItem* parameter.

Both an item in a dialog box and a help balloon for an interface element have rectangles that define the item or element's location. Usually this rectangle should suffice as the rectangle that defines where the coachmark should be drawn. However, you can further specify the rectangle in which Apple Guide draws the coachmark by providing information in the *itemRectangle* parameter.

EXAMPLES

```
#define an item coach for item (item no.7) in a dialog box
<Define Item Coach> "SpellCheckCoach", 'WAVE', REDCIRCLE,-
                    "Spell Check Options", DialogID(7)
```

```

<Define Panel> "spell-checking dialog box"
  <Coach Mark> "SpellCheckCoach"
    To spell-check a document and ignore valley slang,
    click the Ignore valley slang checkbox.
<End Panel>

#define an item coach based on a help balloon
<Define Item Coach> "page number balloon", 'WAVE', ~
    REDCIRCLE, FRONTWINDOW, BalloonID(8)

#define another item coach based on a help balloon
<Define Item Coach> "current style balloon", 'WAVE', ~
    RedArrow(1,7), ~
    FRONTWINDOW, BalloonID(4)

#define an item coach for item (item no.5) in a dialog box
# and specify a subrectangle for the coachmark
<Define Item Coach> "SlangCheckCoach", 'WAVE', REDCIRCLE, ~
    "Spell Check Options", DialogID(5), ~
    Rect(20,20,70,70)

```

SEE ALSO

For information on the <Coach Mark> command, see page 10-118. For information on help resources, see the chapter “Help Manager” in *Inside Macintosh: More Macintosh Toolbox*.

<Define Object Coach>

You can use the <Define Object Coach> command to define a coachmark for an object, based on a rectangle that your application returns for the named object.

```

<Define Object Coach> coachMarkName, targetApp [, coachStyle]
                      [, objectName]

```

coachMarkName

A text string specifying the name of this object coach.

targetApp

A four-character sequence specifying the signature of the target application.

coachStyle

A value indicating the coach style to use for the item. You specify how Apple Guide should draw the coachmark for the item using a constant or red arrow specifier:

Constant

REDCIRCLE, REDUNDERLINE, or GREENX

Red arrow specifier

RedArrow(*start*, *end*)

Use the **RedArrow** function to draw a red arrow beginning at a location specified by *start* and ending at a location specified by *end*. The values for *start* and *end* are integers 1 through 8, where each value indicates a general location in a rectangle:

Location	Value
Top left	1
Top center	2
Top right	3
Middle right	4
Bottom right	5
Bottom center	6
Bottom left	7
Middle left	8

These values indicate the eight possible points for the start and end of the arrow.

The *coachStyle* parameter is optional. If you omit this parameter, Apple Guide uses REDCIRCLE as the default.

objectName

The object associated with this coachmark.

DESCRIPTION

The <Define Object Coach> command defines an object coach. You associate an object coach with a particular panel using the <Coach Mark> command. When Apple Guide opens a panel that includes a <Coach Mark> command naming a

defined object coach, Apple Guide sends an Apple event to your application, requesting it to return a rectangle for the named object. Once your application returns a rectangle for the object, Apple Guide draws the coachmark.

EXAMPLES

```
#define an object coach
<Define Object Coach> "InfoButtonCoach", 'WAVE',
                      REDCIRCLE, "infoBox"
<Define Panel> "the info button"
  <Coach Mark> "InfoButtonCoach"
<End Panel>
```

SEE ALSO

For information on the <Coach Mark> command, see page 10-118.

<Define Window Coach>

You can use the <Define Window Coach> command to define a coachmark for a specific area of a window.

```
<Define Window Coach> coachMarkName [, targetApp] [, coachStyle]
                      [, targetWindow]
                      , windowRectangle [, rectOrigin]
```

coachMarkName

A text string specifying the name of this window coach.

targetApp

A four-character sequence specifying the signature of the target application or the constant `FRONT` to specify the frontmost application. This parameter is optional. If you omit this parameter, Apple Guide uses `FRONT` as the default.

coachStyle

A value indicating the coach style to use for the item. You specify how Apple Guide should draw the coachmark for the item using a constant or red arrow specifier:

Constant

REDCIRCLE, REDUNDERLINE, or GREENX

Red arrow specifier

RedArrow(*start*, *end*)

Use the **RedArrow** function to draw a red arrow beginning at a location specified by *start* and ending at a location specified by *end*. The values for *start* and *end* are integers 1 through 8, where each value indicates a general location in a rectangle:

Location	Value
Top left	1
Top center	2
Top right	3
Middle right	4
Bottom right	5
Bottom center	6
Bottom left	7
Middle left	8

These values indicate the eight possible points for the start and end of the arrow.

The *coachStyle* parameter is optional. If you omit this parameter, Apple Guide uses REDCIRCLE as the default.

targetWindow The window associated with this coachmark. You can specify the window's title, the constant FRONTWINDOW, or the constant DESKTOP. This parameter is optional. If omitted, Apple Guide uses FRONTWINDOW as the default.

windowRectangle

A subrectangle that further specifies the area in which the coachmark should be drawn. You can specify a rectangle using **Rect** (*top*, *left*, *bottom*, *right*). You can also use these constants to specify a specific element of a window:

GROWBOX
ZOOMBOX
CLOSEBOX
TITLEBAR

You can use the constant **BOOTDISK** to specify a coachmark for the startup disk (use this constant only if you also specify the Finder as the target application).

rectOrigin

A constant that gives the origin for the subrectangle. You can use one of these constants to specify the origin for the subrectangle:

TOPLEFT

TOPRIGHT

BOTTOMLEFT

BOTTOMRIGHT

The *rectOrigin* parameter is optional. If you omit this parameter, Apple Guide uses a default of **TOPLEFT**.

DESCRIPTION

The <Define Window Coach> command defines a window coach. You associate a window coach with a particular panel using the <Coach Mark> command. When Apple Guide opens a panel that includes a <Coach Mark> command naming a defined window coach, Apple Guide uses the specified coach style to draw a coachmark based on the location of the coachmark, as provided in the *windowRectangle* parameter.

EXAMPLES

```
#define a window coach for an area in a window
<Define Window Coach> "RulerTabCoach", 'WAVE', REDCIRCLE, ~
    FRONTWINDOW, Rect(30, 30, 50, 50)
<Define Panel> "using the ruler"
    <Coach Mark> "RulerTabCoach"
    To set a tab in your document,
    click the tab tool in the ruler bar.
<End Panel>
```

SEE ALSO

For information on the <Coach Mark> command, see page 10-118.

<Define AppleScript Coach>

You can use the <Define AppleScript Coach> command to define a coachmark that uses an AppleScript script to determine the object to mark.

```
<Define AppleScript Coach> coachMarkName [, coachStyle]
                        , AppleScriptID
```

coachMarkName

A text string specifying the name of this AppleScript coach.

coachStyle

A value that indicates the coach style to use for the item. You specify how Apple Guide should draw the coachmark for the item using a constant or red arrow specifier:

Constant

REDCIRCLE, REDUNDERLINE, or GREENX

Red arrow specifier

RedArrow(*start*, *end*)

Use the RedArrow function to draw a red arrow beginning at a location specified by *start* and ending at a location specified by *end*. The values for *start* and *end* are integers 1 through 8, where each value indicates a general location in a rectangle:

Location	Value
Top left	1
Top center	2
Top right	3
Middle right	4
Bottom right	5
Bottom center	6
Bottom left	7
Middle left	8

These values indicate the eight possible points for the start and end of the arrow.

The *coachStyle* parameter is optional. If you omit this parameter, Apple Guide uses REDCIRCLE as the default.

AppleScriptID The resource ID of a script or the name of a script file. You can specify a filename in the same folder as your source files or you can specify the file and its pathname, relative to the folder containing Guide Maker (for example, ":My Scripts:Some Coach Script"). The script that you specify should return a rectangle identifying the location for the coachmark.

DESCRIPTION

The <Define AppleScript Coach> command defines an AppleScript coach. You associate an AppleScript coach with a particular panel using the <Coach Mark> command. When Apple Guide opens a panel that includes a <Coach Mark> command naming a defined AppleScript coach, Apple Guide executes the specified script. Once the script returns a rectangle for the object, Apple Guide draws the coachmark.

If you specify a script by resource ID, you must make it available to Guide Maker by using a <Resource> command.

SPECIAL CONSIDERATIONS

If AppleScript is not installed on the user's system, Apple Guide does not attempt to execute the script and thus no coachmark will be drawn.

EXAMPLES

```
#define an AppleScript coach
<Define AppleScript Coach> "PageNumber", REDCIRCLE, ↵
    ":My Scripts:Find Rect for PageNo"
<Define Panel> "page number info"
    <Coach Mark> "PageNumber"
<End Panel>
```

SEE ALSO

The <Coach Mark> command is described next.

<Coach Mark>

You can use the <Coach Mark> command to associate a coachmark for display with a particular panel.

<Coach Mark> *coachMarkName*

coachMarkName

A text string specifying the name of a defined coachmark.

DESCRIPTION

The <Coach Mark> command associates a coachmark with a particular panel. Apple Guide displays this coachmark when it opens the panel.

The *coachMarkName* parameter must reference a defined coachmark, that is, a coachmark defined with these commands: <Define Menu Coach>, <Define Item Coach>, <Define Object Coach>, <Define Window Coach>, and <Define AppleScript Coach>.

SPECIAL CONSIDERATIONS

If you use the <Coach Mark> command, it must always appear between the <Define Panel> and <End Panel> commands.

You can associate one coachmark per panel; if more than one <Coach Mark> command appears in a panel definition, Apple Guide uses the last one encountered.

EXAMPLES

```
<Define Panel> "creating index markers"  
    <Coach Mark> "CoachIndexTool"  
    To create index markers, select the index tool.  
<End Panel>
```

SEE ALSO

For information on <Define Menu Coach>, <Define Item Coach>, <Define Object Coach>, <Define Window Coach>, and <Define AppleScript Coach> commands, see page 10-105, page 10-108, page 10-111, page 10-113, and page 10-116, respectively.

Creating Hot Items

You can specify as “hot” a particular object, rectangle, or text in the content area of a panel by using the commands described in this section. A “hot” item is an item in a panel that has the properties of a button; that is, when the user clicks this area it is highlighted and a specified action occurs.

<Hot Object>

You can use the <Hot Object> command to create a hot button on a panel.

<Hot Object> *eventFunction*

eventFunction

A name of an event function or event list. You must also include any parameters expected by the function in parentheses following the event function name. You define event functions using the <Define Event> or <Define Event List> command. Guide Maker also provides built-in functions that you can specify in this parameter.

DESCRIPTION

The <Hot Object> command creates a hot button using the rectangle of the next object (text or graphic) specified in the panel definition. The area encompassing this rectangle acts as a “hot” button; that is, when the user clicks in this rectangle, Apple Guide calls the function defined by the *eventFunction* parameter. Usually such a function is used to send a specific Apple event.

EXAMPLES

```
<Define Panel> "spell-checking a document"  
    Select a dictionary to open:  
  
    <Hot Object> openDictionary("Sharon's Super Dictionary")  
    Sharon's Super Dictionary  
  
    <Hot Object> openDictionary("Daphne's Fantastic Dictionary")  
    Daphne's Fantastic Dictionary  
  
<End Panel>
```

SEE ALSO

For information on creating a hot rectangle or hot text, see the descriptions of the <Hot Rectangle> and <Hot Text> commands. For information on event functions, see "Specifying Events" beginning on page 10-177.

<Hot Rectangle>

You can use the <Hot Rectangle> command to create a hot rectangle on a panel.

<Hot Rectangle> *hotRect*, *eventFunction*

hotRect The coordinates of a rectangle, relative to the current pen location, that define the hot rectangle. The top of the coordinate system is defined by the bottom of the immediately preceding text, and the left edge of the coordinate system is defined by the left edge of the current format.

eventFunction A name of an event function or event list. You must also include any parameters expected by the function in parentheses following the event function name. You define event functions

using the <Define Event> or <Define Event List> command. Guide Maker also provides built-in functions that you can specify in this parameter.

DESCRIPTION

The <Hot Rectangle> command creates a hot rectangle using the rectangle specified in the *hotRect* parameter. The area encompassing this rectangle acts as a “hot” area; that is, when the user clicks in this rectangle, Apple Guide calls the function defined by the *eventFunction* parameter. Usually such a function is used to send a specific Apple event.

The <Hot Rectangle> command provides more precise control over placement of the rectangle than the <Hot Object> command, which assumes the rectangle of the following object.

EXAMPLES

<Define Panel> "spell-checking a document"

Select a dictionary to open:

#use rectangle coordinates relative to the current pen location

<Hot Rectangle> Rect(0, 0, 12, 180), ↵

openDictionary("Sharon's Super Dictionary")

Sharon's Super Dictionary

<Hot Rectangle> Rect(0, 0, 12, 180), ↵

openDictionary("Daphne's Fantastic Dictionary")

Daphne's Fantastic Dictionary

<Hot Rectangle> Rect(0, 0, 52, 180), ↵

openDictionary("Any")

<PICT> "GenericDictionaryPict", LEFT

Choose another dictionary

<End Panel>

SEE ALSO

For information on creating a hot object or hot text, see the descriptions of the <Hot Object> command on page 10-119 and the <Hot Text> command, described next.

<Hot Text>

You can use the <Hot Text> command to create a hot button around specified text on a panel.

<Hot Text> *hotText, whichOccurrence, eventFunction*

hotText A string specifying the hot text.

whichOccurrence

A value identifying which occurrence of the text specified in the *hotText* parameter should be considered hot. You can use the constants **FIRST**, **LAST**, and **ALL** in this parameter. The occurrence of the hot text applies only to the text in the immediately following text object, not all text objects on the panel.

eventFunction

A name of an event function or event list. You must also include any parameters expected by the function in parentheses following the event function name. You define event functions using the <Define Event> or <Define Event List> command. Guide Maker also provides built-in functions that you can specify in this parameter.

DESCRIPTION

The <Hot Text> command creates a hot button for the text specified in the *hotText* parameter. The area encompassing this rectangle acts as a “hot” area; that is, when the user clicks in the rectangle surrounding this text, Apple Guide calls the function defined by the *eventFunction* parameter. Usually such a function is used to send a specific Apple event.

The <Hot Text> command provides a method of specifying particular text and a specific occurrence of this text, such as all occurrences of the text within the text object, as hot text, as opposed to the <Hot Rectangle> command, which assumes the rectangle of the following object. Note that the hot text applies only to strings within the following text object (a string of text ended by a carriage return) not to all text in the panel.

EXAMPLES

```
<Define Panel> "spell-checking a document"
  #define hot text, and specify that every occurrence
  # of the text in the following text object is hot
  <Hot Text> "Sharon's Super Dictionary", ALL, ~
              openDictionary("Sharon's Super Dictionary")
  This panel provides information on Sharon's Super Dictionary.
  <Pict> "Super Dictionary Pict", CENTER, "B&WPic File"

  <Hot Text> "Sharon's Super Dictionary", ALL, ~
              openDictionary("Sharon's Super Dictionary")
  Sharon's Super Dictionary is a super dictionary. ~
  It provides lightning fast checking of all text in a document. ~
  In addition, Sharon's Super Dictionary supports WorldScript, ~
  and can spell-check a document in over 20 different scripts.

<End Panel>
```

SEE ALSO

For information on creating a hot object or hot text, see the descriptions of the <Hot Object> and <Hot Rectangle> commands on page 10-119 and page 10-120, respectively. For information on event functions, see "Specifying Events" beginning on page 10-177.

Defining Topic Areas

You can define topic areas and a topic areas instruction for an access window by using the commands described in this section.

<Topic Areas Instruction>

You can use the <Topic Areas Instruction> command to specify a phrase that appears above the list of topic areas in the Full Access window when Topics is the active list.

<Topic Areas Instruction> *topicAreaInstruction*

topicAreaInstruction

A text string that specifies a phrase to display above the list of topic areas.

DESCRIPTION

Apple Guide displays, above the list of topic areas, the phrase defined in the *topicAreaInstruction* parameter. This command is optional. If you omit this parameter, Apple Guide displays the phrase, "1. Click a topic area:" above the list of topic areas.

SPECIAL CONSIDERATIONS

The <Topic Areas Instruction> command should appear once at most in your source files for a specific guide file.

EXAMPLES

#specify a topic areas instruction

<Topic Areas Instruction> "1.Click a topic area of interest:"

<Topic Area>

You can use the <Topic Area> command to specify a phrase that appears in the left column (the topic area column) in the Full Access window when Topics is the active list.

<Topic Area> *topicAreaPhrase* [, *mixinOrder*]

topicAreaPhrase

A text string that specifies a phrase to display in the list of topic areas.

mixinOrder

You use this parameter only for Mixin guide files to specify the sort order for this topic area after it is mixed in. You can either specify the name of a topic area in the main guide file that this topic area should follow or you can specify the constant `FIRST` or `LAST` to insert the topic area at the beginning or end of the main guide file's topic areas.

DESCRIPTION

Apple Guide displays, in the list of topic areas, the phrase defined in the *topicAreaPhrase* parameter. When the user clicks the Topics button, Apple Guide displays the phrases defined by <Topic Area> commands. When the user selects a particular topic area, Apple Guide displays the topics and any headers associated with the selected topic area. You associate topics and headers with a particular topic area by following a <Topic Area> command by <Header> and <Topic> commands.

Apple Guide displays topic areas in the same order as they are defined in your help source files.

SPECIAL CONSIDERATIONS

A <Topic Area> command must be followed by at least one <Header> or <Topic> command.

A Single List Access window contains topics and (optionally) headers. Although Apple Guide does not display topic areas for a Single List Access window, you must include one <Topic Area> command (followed by the <Header> and <Topic> commands) for your Single List Access guide file.

EXAMPLES

#specify a topic area

```
<Topic Area> "Fonts"
```

```
    #specify header and topics for this topic area
```

```
    <Header> "How do I"
```

```
        <Topic> "change the font of a word?", "SeqFont1"
```

```
        <Topic> "change the font of a paragraph?", "SeqFont2"
```

```
        <Topic> "change the font of a document?", "SeqFont3"
```

```
        <Topic> "change the default font?", "SeqFont4"
```

#specify a topic area

```
<Topic Area> "Styles"
```

```
    #specify headers and topics for this topic area
```

```
    <Header> "How do I"
```

```
        <Topic> "change the style of a word?", "SeqStyle1"
```

```
        <Topic> "change the style of a paragraph?", "SeqStyle2"
```

```
        <Topic> "change the style of a document?", "SeqStyle3"
```

```
        <Topic> "change the default style?", "SeqStyle4"
```

```
    <Header> "Definitions"
```

```
        <Topic> "style", "DefnsStyle"
```

#specify a topic area

```
<Topic Area> "Index markers"
```

```
    #specify header and topics for this topic area
```

```
    <Header> "How do I"
```

```
        <Topic> "create an index marker?", "CreateIndexMarkerSeq"
```

```
        <Topic> "select an index marker?", "SelectIndexMarkerSeq"
```

```
        <Topic> "remove an index marker?", "RemoveIndexMarkerSeq"
```

```
        <Topic> "generate an index?", "GenerateIndexSequence"
```

SEE ALSO

For information on the <Header> and <Topic> commands, see page 10-135 and page 10-137, respectively.

Defining Index Terms

You can define index terms and specify sorting order information of index terms by using the commands described in this section.

<Index Instruction>

You can use the <Index Instruction> command to specify a phrase that appears above the list of index terms in the Full Access window when Index is the active list.

<Index Instruction> *indexInstruction*

indexInstruction

A text string that specifies a phrase to display above the list of index terms.

DESCRIPTION

Apple Guide displays, above the list of index terms, the phrase defined in the *indexInstruction* parameter.

This command is optional. If you omit this command, Apple Guide displays the phrase, "1. Click an index entry:" above the list of index terms.

To define a phrase that Apple Guide displays above the right column, use the <Topics Instruction> command. You use this command to specify a phrase that appears above the list of topics regardless of whether Topics, Index, or Look For is active.

SPECIAL CONSIDERATIONS

The <Index Instruction> command should appear once at most in your source files for a specific guide file.

EXAMPLES

```
#specify an index instruction  
<Index Instruction> "1. Click an index term:"
```

SEE ALSO

For information on the <Topics Instruction> command, see page 10-134.

<Index>

You can use the <Index> command to specify a phrase that appears in the left column (the index column) in the Full Access window when Index is the active list.

```
<Index> indexTerm [ , visible ] [ , key ]
```

indexTerm A text string that specifies an index term for display in the list of index terms.

visible A Boolean constant that indicates whether the index term is visible (TRUE) or invisible (FALSE). This parameter is optional. If you omit this parameter, Apple Guide uses TRUE as the default.

key A text string that specifies a key that Apple Guide should use when sorting this index term among other index terms. This parameter is optional and should be supplied only if you also specify an <Index Sorting> command.

DESCRIPTION

Apple Guide displays, in the list of index terms, the phrase defined in the *indexTerm* parameter. When the user clicks the Index button, Apple Guide displays the index terms defined by <Index> commands. When the user selects a particular index term, Apple Guide displays the topics and any headers associated with the selected index term. You associate topics and headers with

a particular index term by following an `<Index>` command by `<Header>` and `<Topic>` commands.

The *visible* parameter determines whether the index term is visible or invisible. Apple Guide displays visible index terms in the list of index terms. Apple Guide does not display invisible index terms in the list of index terms; but when a user uses the Look For facility to search for an index term, Apple Guide searches both visible and invisible index terms. Invisible index terms allow you to specify key phrases that you want the user to be able to search for but that you do not want included in the list of displayed index terms.

By default, Apple Guide displays index terms in alphabetical order regardless of the order in which they appear in your help source files. To specify that Apple Guide should display terms in the order they appear in your source files or to indicate that you are providing keys for each index term, use the `<Index Sorting>` command.

SPECIAL CONSIDERATIONS

An `<Index>` command must be followed by at least one `<Header>` or `<Topic>` command. You must define at least one index term if you use the Full Access window for your guide file.

EXAMPLES

`#specify index term, then header and topics`

```
<Index> "Page width"
  #specify header and topics for this index term
  <Header> "How do I"
    <Topic> "change the default page width?", -
      "How do I change the default page width?"
    <Topic> "give each page a different page width?", -
      "How do I give each page a different page width?"
```

`#specify index term then header and topics`

```
<Index> "Fonts"
  #specify header and topics for this index term
  <Header> "How do I"
```

```

<Topic> "change the default font?", ~
        "How do I change the default font?"
<Topic> "increase the size of a font?", ~
        "How do I increase the size of a font?"

```

#specify invisible index term then header and topics

```

<Index> "Setting page width", FALSE
    #specify header and topics for this invisible index term
    <Header> "How do I"
        <Topic> "change the default page width?", ~
                "How do I change the default page width?"
        <Topic> "give each page a different page width?", ~
                "How do I give each page a different page width?"

```

SEE ALSO

For information on the <Header> and <Topic> commands, see page 10-135 and page 10-137, respectively. For information on specifying index information for Mixin guide files, see the descriptions of the <Insert Index Header> and <Insert Index Topic> commands on page 10-195 and page 10-196, respectively. To provide more control over the sorting and sorting order of index terms, use the <Sorting> command (described next) and the <Index Sorting> command (described on page 10-132).

<Sorting>

You can use the <Sorting> command to specify the method Apple Guide should use when sorting index terms in your guide file.

<Sorting> *method*

indexTerm A constant, either SCRIPTSORT or USASCISORT, indicating the sorting method. If you specify SCRIPTSORT, Apple Guide sorts index terms according to the international sorting method

for the script specified by the <World Script> command. If you specify **USASCIIISORT**, Apple Guide sorts index terms using a fast, case-insensitive ASCII comparison routine.

DESCRIPTION

Apple Guide sorts index terms before displaying them. Apple Guide sorts these terms according to the sorting method specified by the <Sorting> command.

SPECIAL CONSIDERATIONS

The <Sorting> command is optional. If you omit it, Apple Guide uses **SCRIPTSORT** as the default. Apple Guide retrieves information much more quickly from your guide file if you specify the **USASCIIISORT** sorting method, especially for non-Roman script systems. However, if you specify **USASCIIISORT** you should provide hidden keys for your index terms to obtain best results. You may also need to customize the sliding letter bar that appears above the list of index terms.

EXAMPLES

```
#sort index terms according to the sorting method of the
# script specified by the World Script command
<Sorting> SCRIPTSORT
#sort index terms using the fast, case-insensitive
# sorting method
<Sorting> USASCIIISORT
```

SEE ALSO

You can also specify the order in which index terms appear in the displayed list of index terms by using the <Index Sorting> command, described next.

<Index Sorting>

You can use the <Index Sorting> command to specify how Apple Guide should order your index terms.

<Index Sorting> *orderingKey*

orderingKey A constant that specifies how Apple Guide should order index terms when it sorts them for display. You can specify one of four possible constants: `USEDISPLAYEDTERM`, `NONE`, `USEHIDDENKEY`, or `USEHIDDENKEYWITHOUTIGNORE`.

DESCRIPTION

Apple Guide displays index terms in alphabetical order regardless of the order in which they appear in your help source files, unless you specify the <Index Sorting> command. Using this command, you can more directly control the order in which index terms appear in the list.

If you specify the constant `USEDISPLAYEDTERM`, Apple Guide displays index terms in alphabetical order regardless of the order in which they appear in your help source files. Apple Guide uses this as a default if you do not provide an <Index Sorting> command in your source files.

If you specify the constant `NONE`, Apple Guide displays index terms in the order in which they appear in your help source files.

If you specify the constant `USEHIDDENKEY` or `USEHIDDENKEYWITHOUTIGNORE`, you must also specify the third parameter of all <Index> commands. You can use the third parameter of the <Index> command to specify a key that Apple Guide uses as that term's key when sorting index terms. Note that the key does not appear in the list of index terms.

When you use the constant `USEHIDDENKEY`, the key is used for sorting purposes only and is not searchable by the user. In this case, the user can find the index term by entering it as a search phrase but the user cannot find the index term by entering the index term's key as a search phrase. When you use the constant `USEHIDDENKEYWITHOUTIGNORE`, the user can find the index term by entering either the index term or the key as a search phrase.

SPECIAL CONSIDERATIONS

The <Index Sorting> command is optional. If you omit this command, Apple Guide uses `USEDISPLAYEDTERM` as the default. If you specify this command, it should appear before any of your index terms and should appear near the beginning of your source file. You typically use this command only for non-Roman script systems when you need direct control over the sorting order.

EXAMPLES

```
#specify that all index terms include a key
<Index Sorting> USEHIDDENKEY
#specify index term with a key that determines sort order
<Index> "32-bit addressing",,"Thirty-two bit addressing"
  #specify header and topics for this index term
  <Header> "How do I"
    <Topic> "set 32-bit addressing?", -
      "How do I set 32-bit addressing?"
#specify index term with a key that determines sort order
<Index> "24-bit addressing",,"Twenty-four bit addressing"
  #specify header and topics for this index term
  <Header> "How do I"
    <Topic> "set 24-bit addressing?", -
      "How do I set 24-bit addressing?"
```

SEE ALSO

For information on the <Index> and <Sorting> commands, see page 10-128 and page 10-130, respectively.

Defining Topics for Topic Areas and Index Terms

You can define topics and headers for topic areas and index terms by using the commands described in this section.

<Topics Instruction>

You can use the <Topics Instruction> command to specify a phrase that appears above the list of topics in the Full Access window when Topics or Index is the active list.

<Topics Instruction> *topicsInstruction*

topicsInstruction

A text string that specifies a phrase to display above the list of topics.

DESCRIPTION

Apple Guide displays, above the list of topics, the phrase defined in the *topicsInstruction* parameter.

This command is optional. If you omit this command, Apple Guide displays the phrase "2. Click a phrase, then click OK:" above the list of topics when Topics or Index is the active list.

SPECIAL CONSIDERATIONS

The <Topics Instruction> command should appear once at most in your source files for a specific guide file.

Note that when Look For is the active list and the user has performed a successful search, Apple Guide displays the phrase "3. Click a phrase, then click OK:" above the list of topics; you can change this text using the <Look For Results Instruction> command.

EXAMPLES

```
#specify a topics instruction
```

```
<Topics Instruction> "2. Click an item, then click OK."
```

SEE ALSO

For information on specifying the instruction that appears above the list of topic areas, see the description of the `<Topic Areas Instruction>` command on page 10-124. For information on specifying the instruction that appears above the list of index terms, see the description of the `<Index Instruction>` command on page 10-127.

`<Header>`

You can use the `<Header>` command to specify a phrase that appears in the right column (the help topics column) in the Full Access window when Topics or Index is the active list and the user selects the header's associated topic area or index term. You can also use the `<Header>` command for a Single List Access window.

`<Header>` *headerPhrase*

headerPhrase A text string that specifies a header to display in the list of topics. Apple Guide displays headers in bold.

DESCRIPTION

Apple Guide displays, in the list of topics, the phrase defined in the *headerPhrase* parameter. When the user clicks the Topics button, Apple Guide displays the phrases defined by `<Topic Area>` commands. When the user clicks the Index button, Apple Guide displays in the index terms area the phrases defined by `<Index>` commands. When the user selects a particular topic area or index term, Apple Guide displays any headers (and topics) associated with the selected topic area or index term. You associate a header with a particular topic area by following either a `<Topic Area>` or an `<Index>` command with a `<Header>` command.

The text specified in the *headerPhrase* parameter is typically a general header that groups several related topics or that provides a question prefix for several related topics, such as "How do I" or "Why can't I".

Headers are not required. However, if you do define a header for a topic area or index term, all topics for that topic area or index term must be grouped by

headers. Alternatively, a topic area or index term can have all single topics associated with it.

Apple Guide places an animated triangle next to any header. When the user clicks the triangle, Apple Guide animates it and expands or compacts the topics under the header.

SPECIAL CONSIDERATIONS

If you define a header for a topic area or index term, all topics for that topic area or index term must be grouped by headers.

EXAMPLES

#specify a topic area

```
<Topic Area> "Fonts"
  #specify header and topics for this topic area
  <Header> "How do I"
    <Topic> "change the font of a word?", "SeqFont1"
    <Topic> "change the font of a paragraph?", "SeqFont2"
    <Topic> "change the font of a document?", "SeqFont3"
    <Topic> "change the default font?", "SeqFont4"
  <Header> "Definitions"
    <Topic> "fonts", "DefnsFontsSequence"
```

#specify a topic area

```
<Topic Area> "Index markers"
  #specify header and topics for this topic area
  <Header> "How do I"
    <Topic> "create an index marker?", "CreateIndexMarkerSeq"
    <Topic> "select an index marker?", "SelectIndexMarkerSeq"
    <Topic> "remove an index marker?", "RemoveIndexMarkerSeq"
    <Topic> "generate an index?", "GenerateIndexSequence"
```

SEE ALSO

For information on the <Index> command, see page 10-128.

<Topic>

You can use the <Topic> command to specify a phrase that appears in the right column (the topics column) in the Full Access window when Topics or Index is the active list and the user selects the header's associated topic area or index term. You can also use the <Topic> command for a Single List Access window.

<Topic> *topicPhrase*, *sequenceName*

topicPhrase A text string that specifies a topic to display in the list of topics. The topic phrase must not be greater than 63 characters. The actual number of characters displayed depends on the font and the available width of the topics column.

sequenceName A text string that specifies a sequence for Apple Guide to display when the user chooses the topic specified by the *topicPhrase* parameter. The sequence name must not be greater than 255 characters and must be unique within the first 63 characters.

DESCRIPTION

Apple Guide displays, in the list of topics, the phrase defined in the *topicPhrase* parameter. Apple Guide displays the topic under its associated header, if any. When the user clicks the Topics button, Apple Guide displays the phrases defined by <Topic Area> commands. When the user clicks the Index button, Apple Guide displays in the index terms area the phrases defined by <Index> commands. When the user selects a particular topic area or index term, Apple Guide displays any headers (and topics) associated with the selected topic area or index term. You associate a topic with a topic area or index term by following a <Topic Area> or <Index> command by a <Topic> command; or you can associate a topic with a particular header by following a <Header> command by a <Topic> command.

The text specified in the *topicPhrase* parameter is typically a specific topic or a question suffix that follows a header. For example, if a `<Header>` command specifies the text "How do I", a `<Topic>` command might specify the text "change the font?".

Headers are not required. However, if you do define a header for a topic area or index term, all topics for that topic area or index term must be grouped by headers. Alternatively, a topic area or index term can have all single topics associated with it.

Apple Guide places an animated triangle next to any header. When the user clicks the triangle, Apple Guide animates it and expands or compacts the topics under the header.

SPECIAL CONSIDERATIONS

A `<Topic>` command must be preceded by a `<Topic Area>`, `<Index>`, or `<Header>` command. If you define a header for a topic area or index term, all topics for that topic area or index term must be grouped by headers.

EXAMPLES

```
#specify a topic area with headers and topics
<Topic Area> "Settings"
  #specify header and topics for this topic area
  <Header> "How do I"
    #specify topic phrase and sequence name
    <Topic> "change the default page width?", "SeqDefaultWidth"
    #specify topic phrase and sequence name
    <Topic> "change the default font?", -
      "How do I change the default font?"
    #specify topic phrase and sequence name
    <Topic> "change the default tabs?", -
      "How do I change the default tabs?"
  <Header> "Why can't I"
    <Topic> "remove a tab?", "Why can't I remove a tab?"
    <Topic> "use smart tabs?", "Why can't I use smart tabs?"
```

#specify index term then header and topics

```
<Index> "Page width"
  #specify header and topics for this index term
  <Header> "How do I"
    <Topic> "change the default page width?", "SeqDefaultWidth"
    <Topic> "give each page a different page width?", "SeqPageWidth"
```

#specify index term then header and topics

```
<Index> "Fonts"
  #specify header and topics for this index term
  <Header> "How do I"
    <Topic> "change the default font?", -
      "How do I change the default font?"
    <Topic> "increase the size of a font?", -
      "How do I increase the size of a font?"
```

#specify a topic area with topics only (no headers)

```
<Topic Area> "Settings"
  #specify topics for this topic area
  <Topic> "How do I change the default settings?", -
    "How do I change the default settings?"
  <Topic> "How do I apply settings from another document?", -
    "How do I apply settings from another document?"
  <Topic> "How do I save settings for a document?", -
    "How do I save settings for a document?"
```

SEE ALSO

For information on the <Topic Area>, <Header>, and <Index> commands, see page 10-125, page 10-135, and page 10-128, respectively. For information on specifying topics for index terms and topic areas for Mixin guide files, see the descriptions of the <Insert Index Header>, <Insert Index Topic>, <Insert Topic Area Header>, and <Insert Topic Area Topic> commands on page 10-195, page 10-196, page 10-192, and page 10-193, respectively.

Specifying “Look For” Help

You can specify the text that appears above the search phrase entry box in the Full Access window and the search string using the <Look For Instruction> and <Look For String> commands. You can also specify text instructing the user to click the Search button and to select a topic using the <Look For Search Btn Instruction> and <Look For Results Instruction> commands.

You can use the <Ignore>, <Exception>, and <Synonym> commands to control how Apple Guide searches your index terms when the user directs it to. When the user enters text in the search phrase entry box and then clicks Search, Apple Guide searches your index terms for a matching entry. You can optionally provide a list of words that Apple Guide should ignore (remove from the user’s entered text string) when searching, a list of words that Apple Guide should not stem, and a list of words that are synonymous with another word.

In general, Apple Guide first

- removes from the string words that are specified in the ignore list.
- stems the remaining words, except for words on the exception list.
- searches the list of synonyms for the parsed *phrase* and, if it is found, replaces it with the equivalent index term and searches the index for this phrase. If it finds the phrase in the index, Apple Guide displays the topics associated with the matched index term.

If Apple Guide does not find the phrase in the synonym list, then Apple Guide

- searches the list of synonyms again, this time for each word in the parsed phrase. If it finds a word on the synonym list, it replaces the synonym with its equivalent index term.
- searches the index for the entire *phrase* (with synonyms for individual words replaced by their equivalent index term). If it finds this phrase in the index, Apple Guide displays the topics associated with the index term.
- searches for each word in the list of index terms. If Apple Guide finds a matching index term for more than one word in the phrase, Apple Guide intersects the results and displays any topics that are common to all words in the phrase.

For example, consider the phrase “How do I view files?”

“How do I view files?”	Original phrase entered by user
“view files”	Apple Guide removes words on ignore list (assuming “how”, “do”, and “I” are on ignore list)
“view file”	Apple Guide stems words (does not stem words on exception list)
“view file”	Apple Guide looks for phrase in synonym list and replaces with its index term if found (in this case, “view file” is not in the synonym list)

If the phrase is found in the synonym list, Apple Guide looks in the index for a matching index term and displays the topics associated with the phrase.

If the phrase is not found in the synonym list, Apple Guide searches again.

“view” “file”	Apple Guide looks for synonyms
“open” “file”	Apple Guide replaces each word with its index term if the word is a synonym (assuming “view” is a synonym for “open”)
“open file”	Apple Guide looks for “open file” in list of index terms; if it doesn’t find this term, Apple Guide continues searching
“open”	Apple Guide looks for “open” in list of index terms
“file”	Apple Guide looks for “file” in list of index terms

If more than one word has a matching index term, Apple Guide displays the intersection of topics (if any). If Apple Guide finds a match for only one word, it displays a message indicating that it couldn’t find an intersection for the word that didn’t match and instructs the user to narrow the search phrase.

<Look For Instruction>

You can use the <Look For Instruction> command to specify a phrase that appears above the search phrase entry box in the Full Access window when Look For is active.

<Look For Instruction> *lookForInstruction*

lookForInstruction

A text string that specifies a phrase to display above the search phrase entry box.

DESCRIPTION

Apple Guide displays, above the search phrase entry box, the phrase defined in the *lookForInstruction* parameter. Two lines of space are available for the Look For instruction; Apple Guide automatically wraps the text to fit. You can include a return character in the Look For instruction if you want to wrap the text at a specific location. If you specify an empty string, Apple Guide does not display a Look For instruction.

This command is optional. If you omit this command, Apple Guide displays the phrase, "1. Click the arrow button to begin, then type one or more words to search for:" above the search phrase entry box.

SPECIAL CONSIDERATIONS

The <Look For Instruction> command should appear once at most in your source files for a specific guide file.

EXAMPLES

```
#specify a look for instruction  
<Look For Instruction> "1. Click the arrow button, then ↵  
enter a search phrase:"
```

SEE ALSO

For information on specifying instructions for the Search button and the list of topics that appear in the Look For window, see the descriptions of the <Look For Search Btn Instruction> and <Look For Results Instruction> commands, on page 10-143 and page 10-144, respectively.

<Look For String>

You can use the <Look For String> command to specify the default search phrase that appears in the search phrase entry box in the Full Access window when Look For is active.

<Look For String> *searchPhrase*

searchPhrase A text string that specifies a phrase to display in the search phrase entry box.

DESCRIPTION

Apple Guide displays, in the search phrase entry box, the phrase defined in the *searchPhrase* parameter.

This command is optional. If you omit this command, Apple Guide does not display any text in the search phrase entry box.

EXAMPLES

```
#specify a default search phrase  
<Look For String> "Enter string to search for here"
```

<Look For Search Btn Instruction>

You can use the <Look For Search Btn Instruction> command to specify the default instruction that appears above the Search button in the Full Access window when Look For is active.

<Look For Search Btn Instruction> *buttonInstruction*

buttonInstruction

A text string that specifies a phrase to display above the Search button.

DESCRIPTION

Apple Guide displays, above the Search button, the phrase defined in the *buttonInstruction* parameter.

This command is optional. If you omit this command, Apple Guide displays the phrase "2. Click Search:" above the Search button.

EXAMPLES

#specify a default instruction for the Search button

<Look For Search Btn Instruction> "2. Click the Search button"

<Look For Results Instruction>

You can use the <Look For Results Instruction> command to specify a phrase that appears above the list of topics in the Full Access window when Look For is the active list and the user has performed a successful search.

<Look For Results Instruction> *resultsInstruction*

resultsInstruction

A text string that specifies a phrase to display above the list of topics.

DESCRIPTION

Apple Guide displays, above the list of topics, the phrase defined in the *resultsInstruction* parameter.

This command is optional. If you omit this command, Apple Guide displays the phrase "3. Click a phrase, then click OK:" above the list of topics when Look For is the active list.

SPECIAL CONSIDERATIONS

The <Look For Results Instruction> command should appear once at most in your source files for a specific guide file.

EXAMPLES

#specify an instruction above list of topics in Look For
<Look For Results Instruction> "3. Click an item, then click OK."

SEE ALSO

For information on specifying the instruction that appears above the search phrase entry box, see the description of the <Look For Instruction> command on page 10-141. For information on specifying the instruction that appears above the Search button, see the description of the <Look For Search Btn Instruction> command on page 10-143.

<Ignore>

You can use the <Ignore> command to specify a word or phrase that Apple Guide should ignore when parsing a search phrase entered by the user when Look For is active.

<Ignore> *ignoreWord*

ignoreWord A text string that specifies a word or phrase to ignore when parsing the phrase entered by the user in the search phrase entry box.

DESCRIPTION

When a user enters a phrase in the search phrase entry box and then clicks Search, Apple Guide parses the entered phrase, removing any words specified by <Ignore> commands. (Then it stems common word variations to a root word; see the <Exception> command, described next, for more details.) Apple Guide also searches the list of synonyms and finally the list of index terms for an index term matching the parsed phrase. For example, if the user enters "How do I view files" and "How", "do", and "I" are on the ignore list (specified by <Ignore> commands), Apple Guide parses these words from the entered text, resulting in a search phrase "view files". Apple Guide then stems words (except for words in the exception list) in the phrase (resulting in "view

file”), and looks for this phrase in the synonym list. If Apple Guide doesn’t find the phrase in the synonym list, it continues the search process, as described earlier.

Apple Guide automatically ignores (removes from the search phrase) numerals and punctuation marks. Using the previous example, Apple Guide parses the similar phrase “How, do I view 3 files?” to “view file”.

EXAMPLES

```
#specify words to ignore when parsing the phrase entered
# by the user in the search phrase entry box
<Ignore> "a"
<Ignore> "how"
<Ignore> "do"
<Ignore> "I"
<Ignore> "am"
<Ignore> "an"
<Ignore> "are"
<Ignore> "aren't"
<Ignore> "aren't"
<Ignore> "at"
<Ignore> "be"
<Ignore> "can"
<Ignore> "cannot"
<Ignore> "can't"
<Ignore> "can't"
<Ignore> "did"
<Ignore> "for"
<Ignore> "in"
<Ignore> "it"
<Ignore> "its"
<Ignore> "it's"
<Ignore> "it's"
<Ignore> "like"
<Ignore> "of"
```

```
<Ignore> "on"  
<Ignore> "or"  
<Ignore> "not"  
<Ignore> "real"  
<Ignore> "really"  
<Ignore> "that"  
<Ignore> "the"  
<Ignore> "this"  
<Ignore> "to"  
<Ignore> "what"  
<Ignore> "why"
```

<Exception>

You can use the <Exception> command to specify a word that should not be stemmed when Apple Guide parses a search phrase. You typically place a word on the exception list *only* if the stemmed form of the word matches an existing index term.

<Exception> *exceptionWord*

exceptionWord

A text string that specifies a word that should not be stemmed when Apple Guide parses the search phrase.

DESCRIPTION

After a user enters a phrase in the search phrase entry box and then clicks Search, Apple Guide parses it. After removing any words specified by <Ignore> commands, Apple Guide parses the remaining string. When parsing, Apple Guide parses common word variations to a root word. If Apple Guide incorrectly stems a word, you usually specify the incorrectly stemmed word as a synonym for the index term. For example, Apple Guide stems “documents” to “docum”, so you typically specify “docum” as a synonym for “documents” using the <Synonym> command. Alternatively, you could prevent

Apple Guide from stemming the word “documents” by placing it on the exception list. But it is usually more convenient to use synonyms.

Thus, you typically use the <Exception> command only if the stemmed form of the word matches an existing index term. For example, assume “custom” and “customizing” are defined as two separate index terms. To prevent Apple Guide from stemming “customizing” to its root form (“custom”), you can include “customizing” on the exception list. Thus, if the user enters “customizing” as a search phrase Apple Guide reports the correct matching index term.

Also note that if a word or phrase in the exception list is a synonym (that you have specified using the <Synonym> command), Apple Guide replaces it with its equivalent index term when parsing. For example, if you have specified the word “editing” in an <Exception> command and also specified in a <Synonym> command that “editing” is synonymous with “copy editing”, when the user enters “editing” in the search phrase, Apple Guide does not stem “editing” but does replace it in the search phrase with “copy editing”. Apple Guide then looks for an index term matching “copy editing”.

If Apple Guide finds a match, it displays the topics associated with the index term.

SPECIAL CONSIDERATIONS

Apple Guide performs stemming and uses the exception list only for guide files that specify the command <World Script> 0,0.

EXAMPLES

```
#specify exception words (word that shouldn't be stemmed)
#you usually place an index term on the exception list
# ONLY when the term would otherwise stem to another
# index term or whose root form has an alternate meaning
<Exception> "customizing"
<Exception> "customize"
<Exception> "editing"
<Exception> "edition"
<Exception> "editions"
```


<Synonym>

You can use the <Synonym> command to specify a synonymous term for an existing index term. Apple Guide uses the synonym list when parsing a search phrase.

<Synonym> *indexTerm*, *synonym*

indexTerm A text string that specifies an existing index term.

synonym A text string specifying a word or phrase that is synonymous with the index term specified in the *indexTerm* parameter.

DESCRIPTION

After a user enters a phrase in the search phrase entry box and clicks Search, Apple Guide parses it. After removing any words specified by <Ignore> commands and, except for words on the exception list, stemming common word variations to a root word, Apple Guide examines the list of synonyms for the phrase. If the phrase is a synonym specified by a <Synonym> command, Apple Guide replaces the synonym with its equivalent index term and searches for the phrase in the list of index terms. (Note that at this point in the search, Apple Guide searches the list of index terms only if it found the phrase on the synonym list.)

If Apple Guide does not find the phrase in the synonym list, then Apple Guide looks in the synonym list again, this time for each word in the parsed phrase. If Apple Guide finds a word on the synonym list, it replaces the word in the phrase with its equivalent index term. Apple Guide then looks in the index for a term matching the parsed phrase (this time any words that are synonyms have been replaced by their equivalent index terms). If Apple Guide finds a match for the phrase, Apple Guide displays the topics for the index term. If it does not find the phrase in the index, Apple Guide searches the list of index terms for each word. If Apple Guide finds a matching index term for more than one word in the phrase, Apple Guide intersects the results and displays any topics that are common to both words.

By using <Synonym> commands, you can increase the likelihood that the search phrase entered by the user will match an index term. For example, you can use a <Synonym> command to specify that the index term “copying” has a synonym “duplic” (which is the stemmed form of the word “duplicate”). In

this case, when the user enters the search phrase “duplicate” Apple Guide stems the word to “duplic”, finds “duplic” on the synonym list and replaces it with “copying”, then looks in the index for a term matching “copying”.

If Apple Guide finds a match, it displays the topics associated with the index term.

SPECIAL CONSIDERATIONS

Apple Guide first looks in the synonym list for the reduced phrase, and only later (if necessary) looks for each word in the synonym list. For example, using the synonym list shown in Examples, if the user enters the phrase “How do I open a file and print it?”, Apple Guide parses this search phrase to “open file print” and looks for this phrase in the synonym list. Apple Guide does not find this phrase in the synonym list. Note that in this case Apple Guide does not replace “open file” with “opening” because Apple Guide looks in the synonym list for the entire phrase, not for phrases within the phrase.

Because the phrase “open file print” isn’t in the synonym list, Apple Guide looks in the list again, this time for each word (“open”, “file”, and “print”). The word “open” is a synonym (for “opening”), “file” is a synonym (for “documents”), and “print” is a synonym (for “printing”). After replacing each word with its equivalent index term (if any), the parsed phrase becomes “opening documents printing”. Apple Guide looks in the index for this phrase, and in this example, does not find an index term for the phrase. So Apple Guide looks in the index for each word in the phrase and displays the intersection of the result.

If no topics are common to both words, Apple Guide instructs the user to narrow the search. For example, the user might narrow the search to “How do I open a file?” In this case, Apple Guide reduces the phrase to “open file” and does replace the phrase with its equivalent index term, “opening”.

Note that for a multi-word index term that contains one or more words that are synonyms, you need to create a synonym which is an exact equivalent of the index term. For example, assume the synonym list is as shown in Examples. Because this list doesn’t include a synonym for the index term “file server”, when the user enters this phrase Apple Guide looks in the synonym list for the entire phrase. If it fails to find a synonym for the entire phrase, it looks in the synonym list for each word, replacing “file” with “documents”. Apple Guide then looks in the index for the phrase “documents server”. If it fails to find the phrase in the index, it then looks in the index for each word and

intersects the results. Thus, as demonstrated in this example, Apple Guide will not report a successful search for the phrase “file server”. Therefore, you must create a synonym which is an exact equivalent of the index term

```
<Synonym> "file server", "file server"
```

In summary, you only need to create a synonym that is an exact equivalent of the index term if the index term contains multiple words, and one or more words in the phrase are synonyms for other index terms.

EXAMPLES

```
#specify words that are synonymous with existing index term
#specify an existing index term, then the synonym
<Synonym> "copying", "clone"
<Synonym> "copying", "duplic"
<Synonym> "copying", "copy"
<Synonym> "copying", "Copy command"
<Synonym> "deleting", "Clear command"
<Synonym> "deleting", "cut"
<Synonym> "deleting", "Cut command"
<Synonym> "deleting", "delet"
<Synonym> "deleting", "delete"
<Synonym> "deleting", "get rid"
<Synonym> "documents", "docum"
<Synonym> "documents", "file"
<Synonym> "memory", "RAM"
<Synonym> "memory", "memory available"
<Synonym> "opening", "double click"
<Synonym> "opening", "open"
<Synonym> "opening", "Open command"
<Synonym> "opening", "open file"
<Synonym> "opening", "execute"
<Synonym> "opening", "run"
```

<Synonym> "Page Setup", "landscape"
 <Synonym> "Page Setup", "landscape print"
 <Synonym> "Page Setup", "page format"
 <Synonym> "Page Setup", "Page Setup command"
 <Synonym> "Page Setup", "paper size"
 <Synonym> "Page Setup", "portrait"
 <Synonym> "Page Setup", "printer choice"
 <Synonym> "Page Setup", "printer option"
 <Synonym> "Page Setup", "print choice"
 <Synonym> "Page Setup", "print option"
 <Synonym> "pop-up menu", "pop up"
 <Synonym> "pop-up menu", "pop up menu"
 <Synonym> "pop-up menu", "popup"
 <Synonym> "pop-up menu", "popup menu"
 <Synonym> "printer drivers", "print driver"
 <Synonym> "printer drivers", "print software"
 <Synonym> "printer drivers", "printer driver"
 <Synonym> "printer drivers", "printer software"
 <Synonym> "printing", "print"
 <Synonym> "printing", "printer"
 <Synonym> "printing", "print file"
 <Synonym> "printing", "print window"
 <Synonym> "printing", "Print command"
 <Synonym> "printing", "ImageWrit"
 <Synonym> "printing", "LaserWrit"
 <Synonym> "printing", "spool"

Specifying Conditional Execution

You can dynamically adjust the display of panels according to conditions that you specify by using the commands described in this section. For these commands (<If>, <Skip If>, <Make Sure>, and <Start Making Sure>), you can specify as a condition function any condition defined with the <Define Context Check> command as well as Guide Maker's two built-in condition functions, `checkBoxState` and `radioButtonState`.

<If>

You can use the **<If>** command to specify a condition that determines whether statements that follow it are executed.

<If> *condition*

condition A condition function, either single or compound, that returns a Boolean value. Guide Script provides several built-in condition functions, such as `radioButtonState` and `checkBoxState`. You can also define your own condition functions using the **<Define Context Check>** command.

DESCRIPTION

You can use the **<If>** command, in combination with the **<End If>** command, to specify conditional execution of one or more commands. You can also use an **<Else>** command with an **<If>** command. Apple Guide executes the statements between an **<If>** command and an **<End If>** (or **<Else>**) command only if the condition evaluates to **TRUE**. Apple Guide executes the statements between an **<Else>** command and an **<End If>** command only if the condition evaluates to **FALSE**.

This is the general structure of conditional execution:

```
<If> condition
    statement(s)      #executed if true
<Else>
    statement(s)      #executed if false
<End if>
```

You typically use conditional statements to dynamically adjust the sequence of panels presented to the user by specifying conditions that check the current state of user settings (such as radio buttons or checkboxes), other settings (such as the current date), or the state of the user's environment (such as whether a particular folder is open).

You can specify a single condition function or combine several condition functions into a compound condition using Guide Script's built-in compound

operators AND, OR, and NOT; you can also use parentheses to further qualify the condition. For example:

```
#example of AND    functionA() AND functionB()
#example of OR     functionA() OR functionB()
#example of NOT    functionA() AND NOT(functionB())
#example of all
                    functionA() AND (NOT(functionB()) OR functionC())
```

The Standard Setup file defines various context checks (such as `OpenWindow`, `InSystemFolder`, and `ControlPanelWinActive`) that you can use to specify a condition in the `<If>` command (and also in the `<Skip If>`, `<Make Sure>`, and `<Start Making Sure>` commands).

SPECIAL CONSIDERATIONS

Every `<If>` command must be balanced with a corresponding `<End If>` command. You cannot nest `<If>` commands more than four levels deep.

EXAMPLES

```
<Define Sequence> "How do I create an index?"
  <Panel> "1st panel always display"
  <If> radioButtonState("Book Index", "1st panel always display")
    <Panel> "Panel 2 if true"
    <Panel> "Panel 3 if true"
  <Else>
    <Panel> "Panel 2 if false"
    <Panel> "Panel 3 if false"
  <End if>
  <Panel> "last panel always display"
<End Sequence>
```

```

<Define Sequence> "How do I create an index?"
  <Panel> "Index Choices 2"
  <If> radioButtonState("Book Index", "Index Choices 2")
    <Panel> "How do I create a book index?"
  <Else>
    <Panel> "How do I create a chapter index?"
  <End if>
<End Sequence>

<Define Sequence> "How do I create an index example 2?"
  <Panel> "Index Choices 2"
  <If> radioButtonState("Book Index", "Index Choices 2")
    <Panel> "Book index:intro"
    <Skip If> myCheckTemplateIsOpen("index")
    <Panel> "index template:open"
    <Panel> "Book index:create"
  <End if>
<End Sequence>

<Define Sequence> "How do I create an index example 3?"
  <Panel> "Index Choices 2"
  <If> radioButtonState("Book Index", "Index Choices 2")
    <Panel> "Book index:intro"
    <If> NOT myCheckTemplateIsOpen("index")
      <Panel> "index template:open"
      <Panel> "index template:create new"
    <End if>
    <Panel> "Book index:create"
  <End if>
<End Sequence>

```

```
<Define Sequence> "How do I add a word to the dictionary?"  
  <Panel> "AddWords intro"  
  #OpenWindow is a context check defined in Standard Setup file  
  <If> NOT OpenWindow('WAVE', "Dictionary")  
    <Panel> "AddWords open dictionary"  
  <End if>  
  <Make Sure> OpenWindow('WAVE', "Dictionary"), "SwContinueSeq"  
  <Panel> "AddWords summary"  
<End Sequence>
```

SEE ALSO

For information on the <Define Context Check> command, see page 10-172.
For information on the <End If> command, see page 10-158. The <Else> command is described next.

<Else>

You can use the <Else> command to specify statements to execute when a condition specified by a previous <If> command evaluates to **FALSE**.

<Else>

DESCRIPTION

You can use the <Else> command, preceded by an <If> command, to specify conditional execution of one or more commands. Use the <End If> command to signal the end of the conditional execution of the <Else> branch of the condition. Apple Guide executes the statements between an <If> command and an <Else> command only if the condition evaluates to **TRUE**. Apple Guide executes the statements between an <Else> command and an <End If> command only if the condition evaluates to **FALSE**.

This is the general structure of conditional execution:


```
<If> condition
      statement(s)      #executed if true
<Else>
      statement(s)      #executed if false
<End if>
```

You typically use conditional statements to dynamically adjust the sequence of panels presented to the user by specifying conditions that check the current state of user settings (such as radio buttons or checkboxes), other settings (such as the current date), or the state of the user's environment (such as whether a particular folder is open).

SPECIAL CONSIDERATIONS

Every <Else> command must be preceded with a corresponding <If> command and followed by an <End If> command.

EXAMPLES

```
<Define Sequence> "How do I create an index?"
  <Panel> "1st panel always display"
  <If> radioButtonState("Book Index", "1st panel always display")
    <Panel> "Panel 2 if true"
    <Panel> "Panel 3 if true"
  <Else>
    <Panel> "Panel 2 if false"
    <Panel> "Panel 3 if false"
  <End if>
  <Panel> "last panel always display"
<End Sequence>
```

```
<Define Sequence> "How do I create an index?"  
  <Panel> "Index Choices 2"  
  <If> radioButtonState("Book Index", "Index Choices 2")  
    <Sequence> "How do I create a book index?"  
  <Else>  
    <Sequence> "How do I create a chapter index?"  
  <End if>  
<End Sequence>
```

SEE ALSO

For information on the `<Define Context Check>` command, see page 10-172. For information on the `<If>` command, see page 10-153.

<End If>

You use the `<End If>` command to specify the end of a sequence of commands in a conditional execution.

```
<End If>
```

DESCRIPTION

You can use the `<End If>` command, in combination with an `<If>` command, to specify conditional execution of one or more commands. You can also use an `<Else>` command with an `<If>` command. Apple Guide executes the statements between an `<If>` command and an `<End If>` (or `<Else>`) command only if the condition evaluates to **TRUE**. Apple Guide executes the statements between an `<Else>` command and an `<End If>` command only if the condition evaluates to **FALSE**.

This is the general structure of conditional execution:

```

<If> condition
    statement(s)           #executed if true
<Else>
    statement(s)           #executed if false
<End if>

```

SPECIAL CONSIDERATIONS

Every <If> command must be balanced with a corresponding <End If> command. You cannot nest <If> commands more than four levels deep.

EXAMPLES

```

<Define Sequence> "How do I create an index?"
  <Panel> "1st panel always display"
  <If> radioButtonState("Book Index", "1st panel always display")
    <Panel> "Panel 2 if true"
    <Panel> "Panel 3 if true"
  <Else>
    <Panel> "Panel 2 if false"
    <Panel> "Panel 3 if false"
  <End if>
  <Panel> "last panel always display"
<End Sequence>

```

```

<Define Sequence> "How do I create an index?"
  <Panel> "Index Choices 2"
  <If> radioButtonState("Book Index", "Index Choices 2")
    <Sequence> "How do I create a book index?"
  <Else>
    <Sequence> "How do I create a chapter index?"
  <End if>
<End Sequence>

```

SEE ALSO

For information on the `<If>` and `<Else>` commands, see page 10-153 and page 10-156, respectively.

`<Skip If>`

You can use the `<Skip If>` command to specify a condition that determines whether the next panel should be displayed.

`<Skip If>` *condition*

condition A condition function, either single or compound, that returns a Boolean value. Guide Script provides several built-in condition functions, such as `radioButtonState` and `checkBoxState`. You can also define your own condition functions using the `<Define Context Check>` command.

DESCRIPTION

You can use the `<Skip If>` command to specify conditional display of the panel specified in the following `<Panel>` command. Apple Guide skips the panel (does not display it) only if the condition evaluates to `TRUE`. Apple Guide displays the panel if the condition evaluates to `FALSE`.

This is the general structure of conditional display of a panel:

```
<Skip If> condition
#skip this panel if condition is true, display if false
  <Panel> "skip panel if condition true"
#continue with other commands
#always show this panel
<Panel> "example panel"
```

You typically use conditional statements to dynamically adjust the sequence of panels presented to the user by specifying conditions that check the current state of user settings (such as radio buttons or checkboxes), other settings (such

as the current date), or the state of the user's environment (such as whether a particular folder is open).

You can specify a single condition function or combine several condition functions into a compound condition using Guide Script's built-in compound operators AND, OR, and NOT; you can also use parentheses to further qualify the condition. For example:

```
#example of AND    functionA() AND functionB()
#example of OR     functionA() OR functionB()
#example of NOT    functionA() AND NOT(functionB())
#example of all
                    functionA() AND (NOT(functionB()) OR functionC())
```

SPECIAL CONSIDERATIONS

If a <Skip If> command immediately precedes a <Make Sure> command, the <Make Sure> command is not evaluated if the <Skip If> condition is true. For example:

```
<Skip If> condition
#skip this panel (and any <Make Sure> commands) if condition is
# true; if false, evaluate <Make Sure> commands
<Make Sure> makeSureCondition, oopsSeq #eval if condition false
<Make Sure> makeSureCondition2, oopsSeq #eval if condition false
    <Panel> "skip panel if condition true"
#continue with other commands
#always show this panel
<Panel> "example panel"
```

EXAMPLES

```
<Define Sequence> "How do I use the dictionary?"
    #check whether the dictionary file is already open -
    # if it isn't open, tell the user how to open it
    #(isDictionaryOpen is application-defined context check)
```

```

<Skip If> isDictionaryOpen("SurfWriter Dictionary")
  <Panel> "instruct user how to open dictionary"
<Make Sure> isDictionaryOpen("SurfWriter Dictionary"), "SwContSeq"
  <Panel> "how to use the dictionary"
<End Sequence>

<Define Sequence> "How do I add a word to the thesaurus?"
  <Panel> "AddWords intro"
  #OpenWindow is a context check defined in Standard Setup file
  <Skip If> OpenWindow('WAVE', "Thesaurus")
    <Panel> "AddWords open thesaurus"
  <Make Sure> OpenWindow('WAVE', "Thesaurus"), "SwContinueSeq"
    <Panel> "AddWords summary"
<End Sequence>

```

SEE ALSO

For information on the <Define Context Check> command, see page 10-172.

<Make Sure>

You can use the <Make Sure> command to specify a condition that must be true in order for the next panel to be displayed and to specify a sequence to display (the sequence to display is referred to as an Oops or Continue sequence) if the condition isn't true.

<Make Sure> *condition, oopsOrContinueSequenceName*

condition A condition function, either single or compound, that returns a Boolean value. Guide Script provides several built-in condition functions, such as `radioButtonState` and `checkBoxState`. You can also define your own condition functions using the <Define Context Check> command.

oopsOrContinueSequenceName

A text string specifying the sequence name of the sequence to display if the condition evaluates to **FALSE**.

DESCRIPTION

You can use the **<Make Sure>** command to specify conditional display of the panel specified in the **<Panel>** command that follows it. Apple Guide displays the panel only if the condition evaluates to **TRUE**. Apple Guide displays the specified sequence if the condition evaluates to **FALSE**.

If the condition is false, the sequence Apple Guide displays is referred to as an Oops sequence (if the sequence instructs the user to correct the problem) or a Continue sequence (if the sequence performs the task for the user). Note that Apple Guide does not display the sequence title of an Oops or Continue sequence. Instead, Apple Guide displays the main sequence title as the Oops or Continue sequence title.

Oops sequences should follow special rules. An Oops sequence should generally consist of one panel, giving the user information on how to correct the problem. Use a tag and indented body format for panels in an Oops sequence. Use the tag "Oops" or its localized equivalent. You should also provide an OK button centered under the panel text. You should provide a function that, when the user clicks the OK button in a panel of an Oops sequence, either returns the user to a previous panel giving instructions on how to correct the problem, or closes the Oops sequence and returns the user to the next panel in the original sequence. You can use the **GoBack** event function to do this.

Continue sequences should also follow special rules. A Continue sequence should consist of one panel, telling the user that Apple Guide is performing the task for them. Use the full format for panel text and provide a Continue button centered under the panel text. To perform the task for the user, your panel definition typically includes an **<On Panel Show>** command. When Apple Guide displays a Continue panel with this command, it executes the specified event function, which should perform the task for the user. For example, an event function might send one or more Apple events to direct the target application to accomplish the task. You should also provide a function that closes the Continue sequence and returns the user to the next panel in the original sequence when the user clicks the Continue button in a panel of a Continue sequence. You can use the **GoBack** event function to do this.

This is the general structure of conditional display of a panel using the <Make Sure> command:

```
<Make Sure> condition, oopsOrContinueSequenceName
#(if condition is false,
# display the oopsOrContinueSequenceName sequence)
# if condition is true, display this panel
    <Panel> "show panel if condition true"
#continue with other commands
#always show this panel
<Panel> "example panel"
```

You typically use the <Make Sure> command to ensure that a particular condition is true before allowing the user to continue. You define an Oops sequence to guide the user toward correcting the problem and a Continue sequence if you can perform the task for the user. For example, if a panel requires that a particular window be open, you can use a <Make Sure> command with a condition function that tests whether the window is open. If so, the user can continue with the next panel. Otherwise, you can either provide an Oops sequence that instructs the user to open the window before continuing or provide a Continue sequence that opens the window for the user.

The <Make Sure> command applies only to the next panel in the sequence (however, the GoBack function may update the condition, as explained in the following paragraphs). You can combine <Make Sure> and <Skip If> commands and apply them to a single panel. You should not compound functions in the *condition* parameter for the <Make Sure> command. Instead, if needed you can apply several <Make Sure> conditions, each with their own Oops or Continue sequence, to a single panel.

To apply a Make Sure condition to several panels, use the <Start Making Sure> and <End Making Sure> commands.

When the user clicks a navigation arrow to move to the next panel and the next panel is preceded by a <Make Sure> command, Apple Guide checks the condition associated with the <Make Sure> command before displaying the next panel. If the condition is false, Apple Guide displays the associated Oops or Continue sequence. If you use the GoBack event function, when the user clicks OK in the Oops sequence or Continue in the Continue sequence the GoBack function determines whether the condition has been met. If so, the GoBack function returns the user to the next panel in the original sequence.

If the condition has not been met, then the `GoBack` function works backwards from the original panel until it finds a panel that it can show.

For example, consider a sequence that contains these statements:

```
<Panel> "first panel"
<Make Sure> condition1, oopsSequenceName
# if condition1 is true, display this panel
    <Panel> "show panel if condition true"
#continue with other commands
<Panel> "some panel"
<Make Sure> condition2, oops2SequenceName
# if condition2 is true, display this panel
    <Panel> "example panel"
```

If the condition specified by *condition1* is true, Apple Guide displays the following panel and other panels as the user navigates through them. If the user is viewing the panel named "some panel" in this example and then clicks the right arrow to navigate to the panel named "example panel", Apple Guide evaluates the condition specified by *condition2*. If the condition is true, Apple Guide displays the next panel ("example panel"). If the condition isn't true, Apple Guide displays the sequence named *oops2SequenceName* and allows the user to correct the problem. If the user fixes the problem and then clicks OK, the `GoBack` function proceeds to the next panel ("example panel"). If the user clicks OK but has not fixed the problem, the `GoBack` function returns to "some panel".

The Oops sequence should tell the user how to correct the problem associated with the condition. Apple Guide hides the current panel while the Oops sequence is displayed. Once the user satisfies the condition specified in the `<Make Sure>` command, you can close the Oops sequence and return to the original sequence (the panel following the `<Make Sure>` command).

The Continue sequence should inform the user that Apple Guide is performing the task for the user. Note that if the user clicks the Continue button before the task completes, Apple Guide may return the user to a previous panel rather than the next panel in the sequence. Thus, your Continue panel should always instruct the user to wait until the task completes before clicking the Continue button. Once the task is complete and the user clicks Continue, you can close

the Continue sequence and return to the original sequence (the panel following the <Make Sure> command).

The file Standard Setup is provided with Guide Maker. This file defines the GoBack function. If you include the Standard Setup file in your build file, you can automatically use the GoBack function as needed in your source files.

The Standard Setup file also defines various context checks (such as OpenWindow, InSystemFolder, and ControlPanelWinActive) that you can use to specify a condition in the <Make Sure> command (and also in the <If>, <Skip If>, and <Start Making Sure> commands).

SPECIAL CONSIDERATIONS

You cannot apply a <Make Sure> condition to the first panel in a <Jump> sequence. For this reason, be careful when applying <Skip If> and <Make Sure> commands to introductory panels because the restriction on the <Make Sure> command applies to any panel that may appear first.

If you specify more than one <Make Sure> command for a panel, Apple Guide evaluates the conditions in the reverse order from the way they appear in your source file. That is, the <Make Sure> command closest to the <Panel> command is evaluated first.

EXAMPLES

```
#sequence definition for an Oops sequence
<Define Sequence> "instruct user to open dictionary"
  <Sequence Prompt Set> NONE
  <Define Panel> "Oops panel: Open dictionary"
    <Format> "OopsTag"    #a defined format
      Oops
    <Format> "OopsText" #a defined format
      You need to open the SurfWriter Dictionary.
      Click OK for instructions (or open the
      dictionary, then click OK).
    <Standard Button> "OK", Center, GoBack()
  <End Panel>
<End Sequence>
```

```
#sequence definition that uses <Make Sure> and Oops
<Define Sequence> "How do I use the dictionary?"
    <Panel> "intro to dictionary"
    #now make sure that the dictionary file is open before
    # allowing the user to go to the next panel;
    # if it isn't open, tell the user how to open it
    # by providing an Oops sequence
    #(isDictionaryOpen is application-defined context check)
    <Make Sure> isDictionaryOpen("SurfWriter Dictionary"),
        "instruct user to open dictionary" #oops seq.
    <Panel> "finding a word in the dictionary"
    <Panel> "special dictionaries"
<End Sequence>
```

```
#sequence definition for a Continue sequence
<Define Sequence> "open dictionary for the user"
    <Seq Nav Button Set> NONE
    <Define Panel> "Continue panel: Opening dictionary"
        <Format> "Full"          #a defined format
        Please wait a moment. Apple Guide is assisting you by
        opening the SurfWriter dictionary.

    #this 3D button (Continue) is in Standard Resources
    <3D Button> 1070, 1072, Center, GoBack()
    #use prompt text: "Wait until the dictionary is open,
    # then click Continue."
    <Panel Prompt> "Wait while AG opens dictionary"
    #specify event function that Apple Guide executes
    # upon showing the panel; specify your own event
    # function or a built-in event function
    <On Panel Show> DoScript("openSWDictionary")
    <End Panel>
<End Sequence>
```

```
#sequence definition that uses <Make Sure> and Continue
<Define Sequence> "How do I use the dictionary?"
    <Panel> "intro to dictionary"
    #now make sure that the dictionary file is open before
    # allowing the user to go to the next panel;
    # if it isn't open, open it for the user
    # by providing a Continue sequence
    #(isDictionaryOpen is application-defined context check)
    <Make Sure> isDictionaryOpen("SurfWriter Dictionary"),
        "open dictionary for the user" #continue seq.
    <Panel> "finding a word in the dictionary"
    <Panel> "special dictionaries"
<End Sequence>
```

SEE ALSO

For information on the <Define Context Check>, <Define Event>, and <On Panel Show> commands, see page 10-172, page 10-178, and page 10-185.

<Start Making Sure>

You can use the <Start Making Sure> command to specify a condition that must be true for all panels preceding the next <End Making Sure> command in order for each panel to be displayed. Like the <Make Sure> command, you also specify an Oops or Continue sequence to display if the condition isn't true.

<Start Making Sure> *condition, oopsOrContinueSequenceName*

condition A condition function, either single or compound, that returns a Boolean value. Guide Script provides several built-in condition functions, such as `radioButtonState` and `checkBoxState`. You can also define your own condition functions using the <Define Context Check> command.

oopsOrContinueSequenceName

A text string specifying the sequence name of the sequence to display.

DESCRIPTION

You can use the <Start Making Sure> command to specify conditional display of all panels that precede the next <End Making Sure> command. Apple Guide displays a panel only if the condition evaluates to **TRUE**. Apple Guide displays the specified sequence if the condition evaluates to **FALSE**.

This is the general structure of conditional display of a panel using the <Make Sure> command:

```
#specify a condition that must be true for all panels
# between <Start Making Sure> and <End Making Sure>
<Start Making Sure> condition, oopsOrContinueSequenceName
#for each panel, display the panel if the condition is true
(if condition is false, display oopsOrContinueSequenceName seq.)
    <Panel> "show panel 1 if condition true"
    <Panel> "show panel 2 if condition still true"
    <Panel> "show panel 3 if condition still true"
<End Making Sure>
#always show this panel
<Panel> "example panel"
```

You typically use the <Start Making Sure> command to ensure that a particular condition is true for a series of panels. You define an Oops sequence to guide the user toward correcting the problem before allowing the user to continue. You define a Continue sequence to perform the task for the user. For example, if all panels in a series require that a particular window remain open, you can use a <Start Making Sure> command with a condition function that tests whether the window is open. If it is, the user can continue with the next panel. Otherwise, you can either provide an Oops sequence that instructs the user to open the window before continuing or provide a Continue sequence that opens the window for the user.

The <Start Making Sure> command applies to all panels preceding an <End Making Sure> command. You can also apply <Make Sure>, <If>, and <Skip If>

commands to any panel in the sequence. You should not compound functions in the *condition* parameter for the <Start Making Sure> command. Instead, if needed you can apply several <Start Making Sure> conditions, each with their own Oops or Continue sequence, to a single panel.

Apple Guide continues to update conditions specified in <Start Making Sure> commands as the user moves through a sequence.

Apple Guide hides the current panel while the Oops or Continue sequence is displayed. Once the condition specified in the <Start Making Sure> command is performed and the user clicks OK or Continue, you can close the Oops or Continue sequence and return the user to the original sequence (the panel following the <Make Sure> command).

SPECIAL CONSIDERATIONS

You cannot apply a <Start Making Sure> condition to the first panel in a <Jump> sequence. Be careful when applying <Skip If> commands and <Start Making Sure> commands to introductory panels; the restriction on the <Start Making Sure> command applies to any panel that may appear first.

A <Start Making Sure> command must always be matched by a following <End Making Sure> command.

If you specify a series of commands that includes a <Start Making Sure> command, a <Jump Sequence> command, and an <End Making Sure> command, the <Start Making Sure> condition will not apply to any panels referenced through the <Jump Sequence> command.

EXAMPLES

```
<Define Sequence> "How do I use the dictionary?"
  <Panel> "intro to dictionary"
  #specify a condition that must be true for all panels
  # between <Start Making Sure> and <End Making Sure>
  #(isDictionaryOpen is an application-defined context check)
  <Start Making Sure> isDictionaryOpen("SurfWriter Dictionary"),
    "instruct user to open dictionary" #oops
  <Panel> "finding a word in the dictionary"
  <Panel> "finding synonyms in the dictionary"
```

```

    <Panel> "getting a list of adverbs from the dictionary"
  <End Making Sure>
<End Sequence>

```

SEE ALSO

For information on the <Define Context Check> command, see page 10-172.

<End Making Sure>

You can use the <End Making Sure> command to end the condition checking specified in a previous <Start Making Sure> command.

```
<End Making Sure>
```

DESCRIPTION

The <End Making Sure> command specifies the end of condition checking begun by a previous <Start Making Sure> command. Apple Guide stops evaluating the condition specified by a <Start Making Sure> command when it encounters an <End Making Sure> command.

SPECIAL CONSIDERATIONS

A <Start Making Sure> command must always be matched by a following <End Making Sure> command.

EXAMPLES

```

<Define Sequence> "How do I use the dictionary?"
  <Panel> "intro to dictionary"
  #specify a condition that must be true for all panels
  # between <Start Making Sure> and <End Making Sure>
  #(isDictionaryOpen is an application-defined context check)

```

```

<Start Making Sure> isDictionaryOpen("SurfWriter Dictionary"),
    "instruct user how to open dictionary" #oops
    <Panel> "finding a word in the dictionary"
    <Panel> "finding synonyms in the dictionary"
    <Panel> "getting a list of adverbs from the dictionary"
<End Making Sure>
<End Sequence>

```

SEE ALSO

For information on the <Start Making Sure> command, see page 10-168.

Defining and Using Context Checks

You can define your own condition functions (context checks) using the <Define Context Check> command. In addition, Guide Maker provides two built-in condition functions, `checkBoxState` and `radioButtonState`. You can specify condition functions in the <If>, <Skip If>, <Make Sure>, and <Start Making Sure> commands.

<Define Context Check>

You can use the <Define Context Check> command to define a context check that you can use in later commands to dynamically adjust the display of panels.

```

<Define Context Check> contextCheckName, codeResSpec [, targetApp]
                        [, additionalParam] [, additionalParam]
                        [, ... ]

```

contextCheckName

A text string specifying the name of this context check. The name must be a single-word string (no spaces) and should be as descriptive as possible.

<i>codeResSpec</i>	A four-character sequence specifying the resource name of an external code module that contains the code that performs the context check. (An external module must have a four-character resource name, a resource type of 'extm', and a resource ID greater than 2000.)
<i>targetApp</i>	A four-character sequence specifying the signature of the target application or the constant <code>FRONT</code> . This parameter is optional. If it is omitted, Apple Guide uses <code>FRONT</code> as the default.
<i>additionalParam</i>	<p>One or more additional parameters, of the form</p> <p><i>data type</i> [:<i>default value</i>]</p> <p>where <i>data type</i> is a constant with an optional default value. To specify the data type, use one of these constants:</p> <p><code>SHORT</code> <code>LONG</code> <code>PSTRING</code> <code>LPSTRING</code> <code>OSTYPE</code></p> <p>To specify a default value, include a colon followed by the value after the data type. By default, if you specify a value as <code>SHORT</code> or <code>LONG</code>, the value is treated as a decimal number. To specify a hexadecimal number, precede the number by <code>0x</code>. If you provide a default value for a parameter, then the caller does not specify this parameter when the context check is used in another command.</p> <p>For strings, you should specify <code>LPSTRING</code> if you intend for the string to be localized. Guide Maker's Localize utility extracts from the <Define Context Check> command strings that are specified by <code>LPSTRING</code> but not strings specified by <code>PSTRING</code>.</p> <p>Avoid providing a default value if you specify <code>LPSTRING</code> or <code>PSTRING</code> as the data type of the additional parameter. Instead, require that the caller of the context check provide this information.</p>

DESCRIPTION

The <Define Context Check> command defines a context check. You typically use a context check to dynamically adjust the display of panels based on

conditions that Apple Guide or your context check can detect. For example, Guide Maker provides two built-in context checks, `checkBoxState` and `radioButtonState`, that return the state of a checkbox or radio button.

After defining a context check, you can reference the context check in `<If>`, `<Skip If>`, `<Make Sure>`, and `<Start Making Sure>` commands.

Apple Guide passes parameters specified in *additionalParam* and following parameters to the context check. The context check should take the appropriate action and then return a value of `TRUE` or `FALSE`.

The file `Standard Resources` is provided with Guide Maker. This file contains external code modules, defined as resources of type `'extm'`. The `Standard Setup` file contains `<Define Context Check>` commands that reference each external module. The context checks defined by these commands can be used to specify a condition in `<If>`, `<Skip If>`, `<Make Sure>`, and `<Start Making Sure>` commands.

EXAMPLES

```
#define a context check called "isSomethingActive"
# (with resource name 'MyEM')
# that has 3 parameters: the first (short) and
# third (long) are specified by the context check to have
# default values of 10 and 30
# the second parameter is of type LPSTRING and the caller
# is required to provide this value
<Define Context Check> "isSomethingActive", 'MyEM', 'WAVE',
    SHORT:10, LPSTRING, LONG:30
<Define Sequence> "using the ruler"
    <If> isSomethingActive("Object1")
        <Panel> "Panel to display if context check is true"
    <Else>
        <Panel> "Panel to display if context check is false"
    <End If>
<End Sequence>
```

SEE ALSO

For information on the <If>, <Skip If>, <Make Sure>, and <Start Making Sure> commands, see page 10-153, page 10-160, page 10-162, and page 10-168, respectively.

checkBoxState

You can use Guide Maker's built-in context check for checkboxes, `checkBoxState`, to determine the state of a checkbox.

`checkBoxState (buttonTitle, panelName)`

buttonTitle A string specifying the checkbox's title, as defined in a <Checkbox> command.

panelName A string specifying on which panel the checkbox appears.

DESCRIPTION

The `checkBoxState` context check returns the state of the specified checkbox. It returns **TRUE** if the checkbox state is on, **FALSE** if the checkbox is off. You usually use the `checkBoxState` context check in conjunction with <If>, <Else>, or <Skip If> commands.

EXAMPLES

```
<Define Panel> "Index Choices 2"
```

```
  Index choices:
```

```
    <Checkbox> "Include See Also entries", TRUE
```

```
    <Checkbox> "Include starting and ending page ranges", FALSE
```

```
<End Panel>
```

```

<Define Sequence> "How do I create an index?"
  <Panel> "Index Choices 2"
  <If> checkBoxState("Include See Also entries", ~
    "Index Choices 2")
    <Panel> "How do I create See Also entries?"
  <End if>
  <If> checkBoxState("Include starting and ending page ranges", ~
    "Index Choices 2")
    <Panel> "How do I create page ranges for an index entry?"
  <End if>
<End Sequence>

```

SEE ALSO

For information on the <If>, <Else>, and <Skip If> commands, see page 10-153, page 10-156, and page 10-160, respectively. For information on the <Checkbox> command, see page 10-69.

radioButtonState

You can use Guide Maker's built-in context check for radio buttons, `radioButtonState`, to determine the state of a radio button.

`radioButtonState (buttonTitle, panelName)`

buttonTitle A string specifying the radio button's title, as defined in a <Radio Button> command.

panelName A string specifying on which panel the radio button appears.

DESCRIPTION

The `radioButtonState` context check returns the state of the specified radio button. It returns **TRUE** if the radio button is on, **FALSE** if the radio button is off. You usually use the `radioButtonState` context check in conjunction with <If>, <Else>, or <Skip If> commands.

EXAMPLES

```

<Define Panel> "Index Choices 2"
    #Label for this group of radio buttons
    What type of index do you want to create?

    <Radio Button> "Book Index", TRUE
    <Radio Button> "Chapter Index", FALSE
<End Panel>

<Define Sequence> "How do I create an index?"
    <Panel> "Index Choices 2"
    <If> radioButtonState("Book Index", "Index Choices 2")
        <Panel> "How do I create a book index?"
    <Else>
        <Panel> "How do I create a chapter index?"
    <End if>
<End Sequence>

```

SEE ALSO

For information on the <If>, <Else>, and <Skip If> commands, see page 10-153, page 10-156, and page 10-160, respectively. For information on the <Radio Button> command, see page 10-64.

Specifying Events

You can specify an event function and associate it with a particular panel by using the commands described in this section. This section also describes Guide Maker's built-in event functions.

<Define Event>

You can use the <Define Event> command to define an event function for use with the following commands: <Standard Button>, <3D Button>, <Define Nav Button>, <Hot Text>, <Hot Rectangle>, <Hot Object>, <On Panel Create>, <On Panel Destroy>, <On Panel Show>, and <On Panel Hide>.

<Define Event> *eventName*, *targetApp*, *eventClass*, *eventID*
 [, *IOPTData*] [, *optKey*] [, *optData*]

<i>eventName</i>	A text string specifying the name of this event function.
<i>targetApp</i>	A four-character sequence specifying the signature of the target application or the constant <code>FRONT</code> . You can use the constant <code>FRONT</code> to specify the frontmost application. You can use the signature 's***' to send an Apple event to Apple Guide.
<i>eventClass</i>	A four-character sequence that identifies the event class.
<i>eventID</i>	A four-character sequence that identifies the event ID.
<i>IOPTData</i>	A short integer that provides data for the 'IOPT' keyword. This parameter is optional. If you provide it, Guide Maker adds to the Apple event a parameter whose keyword is 'IOPT' and whose data contains the value you specify in the <i>IOPTData</i> parameter.
<i>optKey</i>	A four-character sequence that identifies an additional keyword for the event. This parameter is optional.
<i>optData</i>	A string providing additional data for the Apple event parameter whose keyword is specified by <i>optKey</i> . This parameter is optional. If you provide the <i>optKey</i> and <i>optData</i> parameters, Guide Maker adds to the Apple event a parameter whose keyword is specified by <i>optKey</i> and whose data contains the value you specify in the <i>optData</i> parameter.

DESCRIPTION

The <Define Event> command defines an event function. You typically associate an event function with another button-defining command, such as <Standard Button> or <3D Button>. Event functions are used to send an Apple event to a target application, requesting it to perform some action. When the

user clicks a button, Apple Guide calls the event function associated with that particular button and then sends the event to the specified target application.

In addition to specifying the event class and event ID, you can add two Apple event parameters to the event you send by using the *IOPTData*, *optKey*, and *optData* parameters. One parameter is identified by the 'IOPT' keyword, and you provide data for this Apple event parameter in the *IOPTData* parameter. In the *optKey* parameter, you provide the keyword for the second parameter and in the *optData* parameter you provide data for this Apple event parameter.

For example, if an event requires a direct object, specify the keyword for the direct object in the *optKey* parameter and specify its data in the *optData* parameter. Optionally, you can allow the caller of the event function to provide the data for the *optData* parameter. If you do this, the event definition must leave the *optData* parameter blank, and the caller must provide a string surrounded by quotes inside of the parentheses of the event function. As an example, for an event function called *doOpenDocmt*, the caller specifies the data for the direct object like this: *doOpenDocmt* ("HD:Reports folder:Quarter 1 Report").

Guide Maker provides the following built-in event functions: *DoScript*, *GoPanel*, *LaunchNewSequence*, *LaunchNewSequenceNewWindow*, *PlaySound*, *QuitTopicOops*, and *StartTopicOops*. Each built-in event function corresponds to a specific Apple event that is directed to Apple Guide.

Apple Guide also supports a number of other events. These events are defined in the Standard Setup file. If your build file includes the Standard Setup file, then you can specify any of the events defined in that file.

EXAMPLES

```
#define doButton1Action as an event function that sends to
# the app with signature 'WAVE' (SurfWriter) the Apple event
# defined by event class 'sfwr' and event ID 'act1'
<Define Event> "doButton1Action", 'WAVE', 'sfwr', 'act1'

<Define Panel> "Example Panel"
  <Standard Button> "Create Chapter Index", Point(50,100),
    doButton1Action()
<End Panel>
```

```
#define doOpenDocmt as an event function that sends to
# the app with signature 'WAVE' (SurfWriter) the
# Open Documents event. Note that the Open Documents event
# expects a direct object (the name of the file to open)
# as a parameter. This event definition specifies the keyword of the
# direct object and the caller specifies its data.
<Define Event> "doOpenDocmt", 'WAVE', 'aevt', 'odoc',, '-----'
```

```
<Define Panel> "Example Panel 3"
  To accomplish this task, do this:
  Very informative instructions here.
```

```
  #place Open button left-justified on panel
  <3D Button> "OpenButtonUpPict", "OpenButtonDownPict", LEFT, ~
    doOpenDocmt("HD:SurfWriter folder:SampleReport")
<End Panel>
```

```
#define an event function that sends to
# the app with signature 'WAVE' (SurfWriter) the Apple event
# defined by event class 'sfwr' and event ID 'act2'
# This event expects two parameters: the number 2530 is the
# data for the 'IOPT' keyword; the other additional parameter has
# the keyword 'kysf' and "info for kysf parameter" as data.
<Define Event> "doButton2Action", 'WAVE', 'sfwr', 'act2', ~
  2530, ~
  'kysf', "info for kysf parameter"
```

```
<Define Panel> "Example Panel"
  <Standard Button> "Create Chapter Index", Point(50,100),~
    doButton2Action()
<End Panel>
```


SEE ALSO

For information on Guide Maker’s built-in event functions, see “Built-in Event Functions” on page 10-188. For information on the <Standard Button> and <3D Button> commands, see page 10-57 and page 10-60, respectively. For information on specifying event functions for hot objects, see “Creating Hot Items” beginning on page 10-119. For information on specifying event functions for the <On Panel Create> and related commands, see page 10-183.

<Define Event List>

You can use the <Define Event List> command to specify a sequence of events (an event list). You can use an event list as a parameter for the <Standard Button>, <3D Button>, <Define Nav Button>, and <On Panel Create> commands.

```
<Define Event List> eventListName, event1 [, event2] [, event3]
                    [, event4] [, event5] [, event6]
```

eventListName A text string specifying the name of this event list.

event1 A name of an event function. Include any parameters expected by the function in parentheses following its name.

event2 -event6 The parameters *event2* through *event6* can each specify the name of an event function (and any parameters expected by the function in parentheses following its name). These parameters are optional.

DESCRIPTION

The <Define Event List> command specifies one or more event functions. You typically associate an event list with another button-defining command, such as <Standard Button> or <3D Button>. When the user clicks a button, Apple Guide calls the event function or event list associated with that particular button. Event lists are usually used to perform a series of actions, such as sending one or more Apple events to a target application, requesting it to perform various actions.

You can specify up to six event functions in an event list. All events in an event list must be static, that is, they cannot require any parameters from the caller (other than the parameters already provided in the specification of the event in the event list).

Guide Maker provides these built-in event functions: `DoScript`, `GoPanel`, `LaunchNewSequence`, `LaunchNewSequenceNewWindow`, `PlaySound`, `StartTopicOops`, and `QuitTopicOops`.

Apple Guide also supports a number of other events, defined in the Standard Setup file. If your build file includes the Standard Setup file, then you can specify any of the events defined in that file.

SPECIAL CONSIDERATIONS

Any event functions specified in parameters *event1* through *event6* must refer to a event function defined with the `<Define Event>` command or to one of Guide Maker's built-in event functions.

EXAMPLES

```
#an event list specifying a series of event functions
<Define Event List> "ClickAndGo", PlaySound(1000),
                    LaunchNewSequence("My sequence name"),
                    doMyButtonAction(),
                    DoScript("AppleScriptOne")

<Define Panel> "Example Panel"
    <Standard Button> "Create Chapter Index", Point(50,100),
                    ClickAndGo()

<End Panel>
```

SEE ALSO

For information on Guide Maker's built-in event functions, see "Built-in Event Functions" on page 10-188. For information on the `<Standard Button>` and `<3D Button>` commands, see page 10-57 and page 10-60, respectively.

<On Panel Create>

You can use the <On Panel Create> command to define an event function that Apple Guide executes before displaying a panel.

<On Panel Create> *eventFunction*

eventFunction An event function or event list.

DESCRIPTION

The <On Panel Create> command defines an event function or event list for a panel. Apple Guide executes the event function when Apple Guide creates the panel to which it is attached. Apple Guide executes the event function before displaying the panel on the screen. Event functions are usually used to send an Apple event to a target application, requesting it to perform some action.

You can use multiple <On Panel Create> commands per panel. Apple Guide executes any event functions in the order in which the <On Panel Create> command appears in your panel definition.

EXAMPLES

```
#define doExamplePanelAction as an event function that sends to
# the app with signature 'WAVE' (SurfWriter) the Apple event
# defined by event class 'sfwr' and event ID 'act1'
<Define Event> "doExamplePanelAction", 'WAVE', 'sfwr', 'act1'

<Define Panel> "Example Panel"
    <On Panel Create> doExamplePanelAction()
    <On Panel Create> PlaySound(1000)
<End Panel>
```

SEE ALSO

For information on Guide Maker's built-in event functions, see "Built-in Event Functions" on page 10-188. For information on the <Define Event> and <Define Event List> commands, see page 10-178 and page 10-181, respectively.

<On Panel Destroy>

You can use the <On Panel Destroy> command to define an event function that Apple Guide executes when destroying a panel. It executes the event after removing the panel from the screen.

<On Panel Destroy> *eventFunction*

eventFunction An event function or event list.

DESCRIPTION

The <On Panel Destroy> command defines an event function or event list for a panel. Event functions are usually used to send an Apple event to a target application, requesting it to perform some action. Apple Guide executes the event function when Apple Guide destroys the panel to which it is attached.

You can use multiple <On Panel Destroy> commands per panel. Apple Guide executes any event functions in the order in which the <On Panel Destroy> command appears in your panel definition.

EXAMPLES

```
#define doExamplePanelAction as an event function that sends to
# the app with signature 'WAVE' (SurfWriter) the Apple event
# defined by event class 'sfwr' and event ID 'act1'
<Define Event> "doExamplePanelAction", 'WAVE', 'sfwr', 'act1'
<Define Panel> "Example Panel"
    <On Panel Destroy> doExamplePanelAction()
    <On Panel Destroy> PlaySound(1000)
<End Panel>
```

SEE ALSO

For information on Guide Maker's built-in event functions, see "Built-in Event Functions" on page 10-188. For information on the <Define Event> and <Define Event List> commands, see page 10-178 and page 10-181, respectively.

<On Panel Show>

You can use the <On Panel Show> command to define an event function that Apple Guide executes when opening a panel and when expanding a panel.

<On Panel Show> *eventFunction* [, *firstOrAlways*]

eventFunction An event function or event list.

firstOrAlways A constant that indicates whether Apple Guide should execute the event function only when it first shows the panel (**FIRST**) or both when it first shows the panel and whenever the panel is expanded (**ALWAYS**). This parameter is optional. If you omit this parameter, Apple Guide uses **ALWAYS** as the default.

DESCRIPTION

The <On Panel Show> command defines an event function or event list for a panel. Apple Guide executes the event function, based on the *firstOrAlways* parameter, when Apple Guide shows or expands the panel to which it is attached. Event functions are usually used to send an Apple event to a target application, requesting it to perform some action.

You can use multiple <On Panel Show> commands per panel. Apple Guide executes event functions in the order in which the <On Panel Show> command appears in your panel definition.

You typically use the <On Panel Show> command to perform a task that is required by a Continue panel. One of the parameters to the <Make Sure> command is the sequence to display if the specified condition isn't true. In this case, Apple Guide displays the sequence, showing the first panel in the sequence. If you include an <On Panel Show> command in this panel definition, Apple Guide executes the specified event function. This event function should perform the task for the user.

EXAMPLES

```
#simple panel that plays a sound when opened or expanded
<Define Panel> "Example Panel"
    #always play a sound when panel is opened and also
    # when it is expanded
    <On Panel Show> PlaySound(1000), ALWAYS
<End Panel>
```

```
#sequence definition for a Continue sequence
<Define Sequence> "open dictionary for the user"
    <Seq Nav Button Set> NONE
    <Define Panel> "Continue panel: Opening dictionary"
        <Format> "Full"          #a defined format
        Please wait a moment. Apple Guide is assisting you by
        opening the SurfWriter dictionary.

        #this 3D button (Continue) is in Standard Resources
        <3D Button> 1070, 1072, Center, GoBack()
        #use prompt text: "Wait until the dictionary is open,
        # then click Continue."
        <Panel Prompt> "Wait while AG opens dictionary"
        #specify event function that Apple Guide executes
        # upon showing the panel; specify your own event
        # function or a built-in event function
        <On Panel Show> DoScript("openSWDictionary")
    <End Panel>
<End Sequence>
```

```
#sequence definition that uses <Make Sure> and Continue
<Define Sequence> "How do I use the dictionary?"
    <Panel> "intro to dictionary"
    #now make sure that the dictionary file is open before
    # allowing the user to go to the next panel
    # if it isn't open, open it for the user
```

```
# by providing a Continue sequence
#(isDictionaryOpen is application-defined context check)
<Make Sure> isDictionaryOpen("SurfWriter Dictionary"),
    "open dictionary for the user" #continue seq.
    <Panel> "finding a word in the dictionary"
    <Panel> "special dictionaries"
<End Sequence>
```

SEE ALSO

For information on Guide Maker's built-in event functions, see "Built-in Event Functions" on page 10-188. For information on the <Define Event> and <Define Event List> commands, see page 10-178 and page 10-181, respectively. For information on the <Make Sure> command, see page 10-162.

<On Panel Hide>

You can use the <On Panel Hide> command to define an event function that Apple Guide executes when hiding a panel.

<On Panel Hide> *eventFunction* [, *firstOrAlways*]

eventFunction An event function or event list.

firstOrAlways A constant that indicates whether Apple Guide should execute the event function only when it first hides the panel (**FIRST**) or both when it first hides the panel and whenever the panel is minimized (**ALWAYS**). This parameter is optional. If you omit this parameter, Apple Guide uses **ALWAYS** as the default.

DESCRIPTION

The <On Panel Hide> command defines an event function or event list for a panel. Apple Guide executes the event function, based on the *firstOrAlways* parameter, when Apple Guide hides the panel to which it is attached. Hiding a panel refers to Apple Guide either closing the panel or minimizing the panel.

Event functions are usually used to send an Apple event to a target application, requesting it to perform some action.

You can use multiple <On Panel Hide> commands per panel. Apple Guide executes event functions in the order in which the <On Panel Hide> command appears in your panel definition.

EXAMPLES

```
<Define Panel> "Example Panel"  
    #play a sound the first time Apple Guide hides the panel  
    <On Panel Hide> PlaySound(1000), FIRST  
<End Panel>
```

SEE ALSO

For information on the <Define Event> and <Define Event List> commands, see page 10-178 and page 10-181, respectively. Guide Maker's built-in event functions are described next.

Built-in Event Functions

Guide Maker provides the following built-in event functions: DoScript, GoPanel, LaunchNewSequence, LaunchNewSequenceNewWindow, PlaySound, StartTopicOops, and QuitTopicOops.

Apple Guide also supports a number of other events, defined in the Standard Setup file. If your build file includes the Standard Setup file, then you can specify any of the events defined in that file.

You can specify an event function when using the following commands: <Standard Button>, <3D Button>, <Define Nav Button>, <Hot Text>, <Hot Rectangle>, <Hot Object>, <On Panel Create>, <On Panel Destroy>, <On Panel Show>, and <On Panel Hide>.

The built-in event functions and their parameters are described here.

DoScript(*scriptResource*)

scriptResource A resource ID of a 'scpt' resource previously specified in a <Resource> command or the filename of a compiled script that resides in the same folder as the help source files. Apple Guide runs the referenced script when it invokes the DoScript event function. The DoScript event function can also be accessed using the name DoAppleScript.

GoPanel(*panelNumber*)

panelNumber A panel number identifying a panel in a sequence. Panels are numbered beginning with 1 for the first panel in a sequence, 2 for the second panel, and so on. Apple Guide displays the panel.

LaunchNewSequence(*sequenceName*)

sequenceName A text string specifying the sequence to launch. Apple Guide closes the current topic, if any, and continues with the named sequence. Calling this event function closes the current access window or panel and opens a new one. The guide file of the target application must already be open before calling this event function.

LaunchNewSequenceNewWindow(*sequenceName*)

sequenceName A text string specifying the sequence to launch. Apple Guide does not close the current topic but instead opens a new window. The guide file of the target application must already be open before calling this event function.

PlaySound(*soundResource*)

soundResource A resource ID or resource name of a 'snd' resource previously specified in a <Resource> command, or the filename of a System 7 sound file that is in the same folder as your help source files. Apple Guide plays the sound resource asynchronously when it invokes the PlaySound event function.

StartTopicOops(*sequenceName*)

sequenceName A text string specifying the Oops sequence to launch. Apple Guide hides the current topic and opens the Oops sequence in a new window. The guide file of the target application must already be open before calling this event function.

QuitTopicOops([*panelNumber*])

panelNumber A panel number identifying a panel in a sequence. Panels are numbered beginning with 1 for the first panel in a sequence, 2 for the second panel, and so on. The *panelNumber* parameter is optional. If it is provided, Apple Guide closes the Oops topic and returns to the specified panel. If it is omitted, Apple Guide closes the Oops topic and returns to the parent topic.

Working With Mixin Guide Files

You can modify or add content to an already existing guide file by creating a Mixin guide file. The source file for a Mixin guide file must contain a <Mixin> command and optionally, a <Mixin Match> command. A main guide file may also contain a <Mixin Match> command to specify which Mixin guide files can be mixed in with it.

This section describes the commands you can use in your Mixin source file to replace, delete, or add content to a main guide file.

<Replace Sequence>

You can use the <Replace Sequence> command to specify a sequence in a main guide file that is to be replaced by a new sequence in a Mixin guide file.

<Replace Sequence> *oldSequenceName*, *newSequenceName*

oldSequenceName

A text string specifying the name of the sequence in the main guide file.

newSequenceName

A text string specifying the name of the sequence in the Mixin guide file.

DESCRIPTION

When Guide Maker compiles a source file for a Mixin guide file and encounters a `<Replace Sequence>` command, it looks for a sequence specified by the *oldSequenceName* parameter in the main guide file. If it finds the sequence, it replaces it with the sequence specified by the *newSequenceName* parameter.

SPECIAL CONSIDERATIONS

The `<Replace Sequence>` command can be used only in the source file for a Mixin guide file that also includes the `<Mixin>` command. To use the `<Replace Sequence>` command, you must specify a SYM file in the *symName* parameter of the `<Mixin>` command.

EXAMPLES

```
<Replace Sequence> "creating index markers", -  
                  "New creating index markers"
```

SEE ALSO

For information on the `<Mixin>` command, see page 10-19. To modify or add new topic areas, topics, or index entries to an existing guide file, you can use the commands `<Insert Topic Area Header>`, `<Insert Topic Area Topic>`, `<Insert Index Header>`, and `<Insert Index Topic>`, as described on page 10-192, page 10-193, page 10-195, and page 10-196, respectively.

<Insert Topic Area Header>

You can use the <Insert Topic Area Header> command to insert a header from a Mixin guide file and associate this new header with an existing topic area in a main guide file.

<Insert Topic Area Header> *header*, *topicArea* [, *sortOrder*]

<i>header</i>	A text string specifying a new header.
<i>topicArea</i>	A text string specifying the topic area in a main guide file. Apple Guide associates the new header with the specified topic area.
<i>sortOrder</i>	A value specifying where the new header should appear in the list of headers for the specified topic area. You can use the constant FIRST or LAST to sort the header at the beginning or end of existing headers for the specified topic area. You can also specify in this parameter a text string of the header that should immediately precede the new header. This parameter is optional. If you omit it, Apple Guide uses LAST as the default.

DESCRIPTION

When Guide Maker compiles a source file for a Mixin guide file and encounters an <Insert Topic Area Header> command, it looks for a topic area specified by the *topicArea* parameter in the main guide file. If it finds the topic area, it associates the new header with it and inserts the header into the list of headers according to the sort order specified by the *sortOrder* parameter.

To define the topics for a new header, use the <Insert Topic Area Topic> command.

SPECIAL CONSIDERATIONS

The <Insert Topic Area Header> command can be used only in the source file for a Mixin guide file that also includes the <Mixin> command. To use the <Insert Topic Area Header> command, you must specify a SYM file in the *symName* parameter of the <Mixin> command.

EXAMPLES

```
#in a Mixin guide file, specify a new header for an
# existing topic area in a main guide file
<Insert Topic Area Header> "When should I", ~
                        "Setting Options", FIRST
```

SEE ALSO

For information on the <Mixin> command, see page 10-19. For information on the <Topic Area>, <Header>, and <Topic> commands, see page 10-125, page 10-135, and page 10-137, respectively.

<Insert Topic Area Topic>

You can use the <Insert Topic Area Topic> command to insert a topic from a Mixin guide file and associate this new topic with an existing topic area in a main guide file.

```
<Insert Topic Area Topic> topic, seqName, topicArea [, sortOrder]
```

<i>topic</i>	A text string specifying a new topic.
<i>seqName</i>	A text string specifying the sequence name associated with the new topic.
<i>topicArea</i>	A text string specifying the topic area in a main guide file. Apple Guide associates the new topic with the specified topic area.
<i>sortOrder</i>	A value specifying where the new topic should appear in the list of topics for the specified topic area. You can use the constant FIRST or LAST to sort the topic at the beginning or end of existing topics for the specified topic area. You can also specify in this parameter a text string of the topic that should immediately precede the new topic. This parameter is optional. If you omit it, Apple Guide uses LAST as the default.

DESCRIPTION

When Guide Maker compiles a source file for a Mixin guide file and encounters an `<Insert Topic Area Topic>` command, it looks for a topic area specified by the *topicArea* parameter in the main guide file. If it finds the topic area, it associates the new topic with it and inserts the topic into the list of topics, according to the sort order specified by the *sortOrder* parameter.

To define a header for a new topic, use the `<Insert Topic Area Header>` command.

SPECIAL CONSIDERATIONS

The `<Insert Topic Area Topic>` command can be used only in the source file for a Mixin guide file that also includes the `<Mixin>` command. To use the `<Insert Topic Area Topic>` command, you must specify a SYM file in the *symName* parameter of the `<Mixin>` command.

EXAMPLES

```
#in a Mixin guide file, specify a new topic for an
# existing topic area in a main guide file
<Insert Topic Area Topic> "change the default font?", ~
    "How do I change the default font?", ~
    "Setting Options", ~
    "change the standard footer?"
```

SEE ALSO

For information on the `<Mixin>` command, see page 10-19. For information on the `<Topic Area>`, `<Header>`, and `<Topic>` commands, see page 10-125, page 10-135, and page 10-137, respectively.

<Insert Index Header>

You can use the <Insert Index Header> command to insert a header from a Mixin guide file and associate this new header with an existing index term in a main guide file.

<Insert Index Header> *header*, *indexTerm* [, *sortOrder*]

<i>header</i>	A text string specifying a new header.
<i>indexTerm</i>	A text string specifying the index term in a main guide file. Apple Guide associates the new header with the specified index term.
<i>sortOrder</i>	A value specifying where the new header should appear in the list of headers for the specified index term. You can use the constant <code>FIRST</code> or <code>LAST</code> to sort the header at the beginning or end of existing headers for the specified index term. You can also specify in this parameter a text string of the header that should immediately precede the new header. This parameter is optional. If you omit it, Apple Guide uses <code>LAST</code> as the default.

DESCRIPTION

When Guide Maker compiles a source file for a Mixin guide file and encounters an <Insert Index Header> command, it looks for an index term specified by the *indexTerm* parameter in the main guide file. If it finds the index term, it associates the new header with it and inserts the header into the list of headers, according to the sort order specified by the *sortOrder* parameter.

To define the topics for a new header, use the <Insert Index Topic> command.

SPECIAL CONSIDERATIONS

The <Insert Index Header> command can be used only in the source file for a Mixin guide file that also includes the <Mixin> command. To use the <Insert Index Header> command, you must specify a SYM file in the *symName* parameter of the <Mixin> command.

EXAMPLES

```
#in a Mixin guide file, specify a new header for an
# existing index term in a main guide file
<Insert Index Header> "When should I", ↵
                        "Math operations", FIRST
```

SEE ALSO

For information on the <Mixin> command, see page 10-19. For information on the <Index>, <Header>, and <Topic> commands, see page 10-128, page 10-135, and page 10-137, respectively.

<Insert Index Topic>

You can use the <Insert Index Topic> command to insert a topic from a Mixin guide file and associate this new topic with an existing topic area in a main guide file.

```
<Insert Index Topic> topic, seqName, indexTerm [, sortOrder]
```

<i>topic</i>	A text string specifying a new topic.
<i>seqName</i>	A text string specifying the sequence name associated with the new topic.
<i>indexTerm</i>	A text string specifying the index term in a main guide file. Apple Guide associates the new topic with the specified index term.
<i>sortOrder</i>	A value specifying where the new topic should appear in the list of topics for the specified topic area. You can use the constant FIRST or LAST to sort the topic at the beginning or end of existing topics for the specified topic area. You can also specify in this parameter a text string of the topic that should immediately precede the new topic. This parameter is optional. If you omit it, Apple Guide uses LAST as the default.

DESCRIPTION

When Guide Maker compiles a source file for a Mixin guide file and encounters an `<Insert Index Topic>` command, it looks for an index term specified by the *indexTerm* parameter in the main guide file. If it finds the index term, it associates the new topic with it and inserts the topic into the list of topics, according to the sort order specified by the *sortOrder* parameter.

To define a header for a new topic, use the `<Insert Index Header>` command.

SPECIAL CONSIDERATIONS

The `<Insert Index Topic>` command can be used only in the source file for a Mixin guide file that also includes the `<Mixin>` command. To use the `<Insert Index Topic>` command, you must specify a SYM file in the *symName* parameter of the `<Mixin>` command.

EXAMPLES

```
#in a Mixin guide file, specify a new topic for an
# existing index term in a main guide file
<Insert Index Topic> "use hexadecimal numbers?", -
    "When should I use hexadecimal numbers?", -
    "Math operations", FIRST
```

SEE ALSO

For information on the `<Mixin>` command, see page 10-19. For information on the `<Index>`, `<Header>`, and `<Topic>` commands, see page 10-128, page 10-135, and page 10-137, respectively.

<Delete Topic Area>

You can use the <Delete Topic Area> command to specify that a topic area defined in a main guide file should not be displayed if the main guide file is mixed in with your Mixin guide file.

<Delete Topic Area> *topicArea*

topicArea A text string specifying the topic area that should not be displayed when the mixin is mixed in with the main guide file.

DESCRIPTION

If a Mixin guide file includes a <Delete Topic Area> command, when Apple Guide mixes in the main guide file with the Mixin, the topic area specified by the *topicArea* parameter will not be shown in the list of topic areas when Topics is active. If the main guide file is not mixed in with the Mixin guide file, the topic area is shown in the list.

SPECIAL CONSIDERATIONS

The <Delete Topic Area> command can be used only in the source file for a Mixin guide file that also includes the <Mixin> command. To use the <Delete Topic Area> command, you must specify a SYM file in the *symName* parameter of the <Mixin> command.

EXAMPLES

```
#do not show this topic area if the main guide file is
# mixed in with this guide file
<Delete Topic Area> "Printing"
```

SEE ALSO

For information on the <Mixin> command, see page 10-19. For information on the <Topic Area>, <Header>, and <Topic> commands, see page 10-125, page 10-135, and page 10-137, respectively.

<Delete Topic Area Header>

You can use the <Delete Topic Area Header> command to specify that a header associated with a topic area defined in a main guide file should not be displayed if the main guide file is mixed in with your Mixin guide file.

<Delete Topic Area Header> *topicArea*, *topicAreaHeader*

topicArea A text string specifying the topic area associated with the header.

topicAreaHeader

A text string specifying the header that should not be displayed when the Mixin guide file is mixed in with the main guide file.

DESCRIPTION

If a Mixin guide file includes a <Delete Topic Area Header> command, when Apple Guide mixes in the main guide file with the Mixin guide file, the topic area header specified by the *topicAreaHeader* parameter will not be displayed. If the main guide file is not mixed in with the Mixin guide file, the topic area header is displayed.

SPECIAL CONSIDERATIONS

The <Delete Topic Area Header> command can be used only in the source file for a Mixin guide file that also includes the <Mixin> command. To use the <Delete Topic Area Header> command, you must specify a SYM file in the *symName* parameter of the <Mixin> command.

EXAMPLES

```
#do not show this header if the main guide file is mixed in
# with this guide file
<Delete Topic Area Header> "Printing", "How do I"
```

SEE ALSO

For information on the <Mixin> command, see page 10-19. For information on the <Topic Area>, <Header>, and <Topic> commands, see page 10-125, page 10-135, and page 10-137, respectively.

<Delete Topic Area Topic>

You can use the <Delete Topic Area Topic> command to specify that a topic associated with a topic area defined in a main guide file should not be displayed if the main guide file is mixed in with your Mixin guide file.

<Delete Topic Area Topic> *topicArea*, *topicAreaTopic*

topicArea A text string specifying the topic area associated with the topic.

topicAreaTopic

A text string specifying the topic that should not be displayed when the Mixin guide file is mixed in with the main guide file.

DESCRIPTION

If a Mixin guide file includes a <Delete Topic Area Topic> command, when Apple Guide mixes in the main guide file with the Mixin guide file, the topic specified by the *topicAreaTopic* parameter will not be displayed. If the main guide file is not mixed in with the Mixin guide file, the topic is displayed.

SPECIAL CONSIDERATIONS

The <Delete Topic Area Topic> command can be used only in the source file for a Mixin guide file that also includes the <Mixin> command. To use the <Delete Topic Area Topic> command, you must specify a SYM file in the *symName* parameter of the <Mixin> command.

EXAMPLES

```
#do not show this topic if the main guide file is mixed in  
# with this guide file  
<Delete Topic Area Topic> "Printing", "select a printer?"
```

SEE ALSO

For information on the <Mixin> command, see page 10-19. For information on the <Topic Area>, <Header>, and <Topic> commands, see page 10-125, page 10-135, and page 10-137, respectively.

<Delete Index>

You can use the <Delete Index> command to specify that an index term defined in a main guide file should not be displayed if the main guide file is mixed in with your Mixin guide file.

<Delete Index> *indexTerm*

indexTerm A text string specifying the index term that should not be displayed when the mixin is mixed in with the main guide file.

DESCRIPTION

If a Mixin guide file includes a <Delete Index> command, when Apple Guide mixes in the main guide file with the mixin, the index term specified by the *indexTerm* parameter will not be shown in the list of index terms when Index is active. If the main guide file is not mixed in with the Mixin guide file, the index term is shown in the list.

SPECIAL CONSIDERATIONS

The <Delete Index> command can be used only in the source file for a Mixin guide file that also includes the <Mixin> command. To use the <Delete Index> command, you must specify a SYM file in the *symName* parameter of the <Mixin> command.

EXAMPLES

```
#do not show this index term if the main guide file is  
# mixed in with this guide file  
<Delete Index> "printer drivers"
```

SEE ALSO

For information on the <Mixin> command, see page 10-19. For information on the <Topic Area>, <Header>, and <Topic> commands, see page 10-125, page 10-135, and page 10-137, respectively.

<Delete Index Header>

You can use the <Delete Index Header> command to specify that a header associated with an index term in a main guide file should not be displayed if the main guide file is mixed in with your Mixin guide file.

<Delete Index Header> *indexTerm*, *indexHeader*

indexTerm A text string specifying the index term associated with the header.

indexHeader A text string specifying the header that should not be displayed when the mixin is mixed in with the main guide file.

DESCRIPTION

If a Mixin guide file includes a <Delete Index Header> command, when Apple Guide mixes in the main guide file with the mixin, the header specified by the *indexHeader* parameter will not be shown in the list of headers for the specified index term. If the main guide file is not mixed in with the Mixin guide file, the header is shown in the list.

SPECIAL CONSIDERATIONS

The `<Delete Index Header>` command can be used only in the source file for a Mixin guide file that also includes the `<Mixin>` command. To use the `<Delete Index Header>` command, you must specify a SYM file in the *symName* parameter of the `<Mixin>` command.

EXAMPLES

```
#do not show this index term if the main guide file is
# mixed in with this guide file
<Delete Index Header> "printer drivers", "How do I"
```

SEE ALSO

For information on the `<Mixin>` command, see page 10-19. For information on the `<Topic Area>`, `<Header>`, and `<Topic>` commands, see page 10-125, page 10-135, and page 10-137, respectively.

<Delete Index Topic>

You can use the `<Delete Index Topic>` command to specify that a topic associated with an index term in a main guide file should not be displayed if the main guide file is mixed in with your Mixin guide file.

`<Delete Index Topic> indexTerm, indexTopic`

<i>indexTerm</i>	A text string specifying the index term associated with the topic.
<i>indexTopic</i>	A text string specifying the topic that should not be displayed when the mixin is mixed in with the main guide file.

DESCRIPTION

If a Mixin guide file includes a `<Delete Index Topic>` command, when Apple Guide mixes in the main guide file with the mixin, the topic specified by the *indexTopic* parameter will not be shown in the list of topics for the specified

index term. If the main guide file is not mixed in with the Mixin guide file, the topic is shown.

SPECIAL CONSIDERATIONS

The <Delete Index Topic> command can be used only in the source file for a Mixin guide file that also includes the <Mixin> command. To use the <Delete Index Topic> command, you must specify a SYM file in the *symName* parameter of the <Mixin> command.

EXAMPLES

```
#do not show this index term if the main guide file is
# mixed in with this guide file
<Delete Index Topic> "printer drivers", "choose a printer"
```

SEE ALSO

For information on the <Mixin> command, see page 10-19. For information on the <Topic Area>, <Header>, and <Topic> commands, see page 10-125, page 10-135, and page 10-137, respectively.

Appendixes

Guide Script Command Abbreviations

Most Guide Script commands can be specified by their full name or an abbreviation. Abbreviations for all Guide Script commands are shown here, in alphabetical order of the full command name.

Table A-1 Command abbreviations

Full command name	Abbreviated command name
<3D Button>	<3DB>
<Allow Prompts>	<AP>
<App Creator>	<AC>
<App Logo>	<AL>
<App Text>	<AT>
<Balloon Menu Text>	<BMT>
<Build Sequence>	<BS>
<Checkbox>	<CB>
<Coach Mark>	<CM>
<Comment>	#
<Default Format>	
<Default Nav Button Set>	<DefaultNBS>
<Default Prompt Set>	<DefaultPS>
<Define AppleScript Coach>	<DAC>
<Define Context Check>	<DCC>

continued

Guide Script Command Abbreviations

Table A-1 Command abbreviations (continued)

Full command name	Abbreviated command name
<Define Event>	<DE>
<Define Event List>	
<Define Format>	<DF>
<Define Item Coach>	<DIC>
<Define Menu Coach>	<DMC>
<Define Nav Button>	<DNB>
<Define Nav Button Set>	<DNBS>
<Define Object Coach>	<DOC>
<Define Panel>	<DP>
<Define Prompt Set>	<DPS>
<Define Sequence>	<DS>
<Define Text Block>	<DTB>
<Define Transparent Format>	<DTF>
<Define Window Coach>	<DWC>
<Delete Index>	<DI>
<Delete Index Header>	<DIH>
<Delete Index Topic>	<DIT>
<Delete Topic Area>	<DTA>
<Delete Topic Area Header>	<DTAH>
<Delete Topic Area Topic>	<DTAT>
<Dimmable Button Data>	<DBD>
<Else>	
<End If>	<EI>
<End Making Sure>	<EMS>

Table A-1 Command abbreviations (continued)

Full command name	Abbreviated command name
<End Panel>	<EP>
<End Sequence>	<ES>
<End Text Block>	<ETB>
<Exception>	<EXC>
<Format>	
<Gestalt>	
<Header>	
<Help Menu>	<HM>
<Hot Object>	<HO>
<Hot Rectangle>	<HR>
<Hot Text>	<HT>
<Howdy>	
<If>	
<Ignore>	<IGN>
<Include>	
<Index>	
<Index Instruction>	<II>
<Index Sorting>	<IS>
<Insert Index Header>	<IIH>
<Insert Index Topic>	<IIT>
<Insert Sequence>	<IS>
<Insert Topic Area Header>	<ITAH>
<Insert Topic Area Topic>	<ITAT>

continued

Table A-1 Command abbreviations (continued)

Full command name	Abbreviated command name
<Jump Sequence>	<JS>
<Launch New Sequence>	<LNS>
<Look For Instruction>	<LFI>
<Look For Results Instruction>	<LFRI>
<Look For Search Btn Instruction>	<LFSBI>
<Look For String>	<LFS>
<Make Sure>	<MS>
<Max Height>	
<Min Height>	
<Mixin>	
<Mixin Match>	<MM>
<On Panel Create>	<OPC>
<On Panel Destroy>	<OPD>
<On Panel Hide>	<OPH>
<On Panel Show>	<OPS>
<Panel>	
<Panel Prompt>	<PP>
<PICT>	
<QuickTime>	<QT>
<Radio Button>	<RB>
<Radio Button Launch New Seq>	<RBLNS>
<Replace Sequence>	<RS>
<Resource>	
<Seq Nav Button Set>	<SNBS>

Guide Script Command Abbreviations

Table A-1 Command abbreviations (continued)

Full command name	Abbreviated command name
<Sequence Prompt Set>	<SPS>
<Skip If>	<SI>
<Sorting>	
<Standard Button>	<SB>
<Starting Res Number>	<SRN>
<Start Making Sure>	<SMS>
<Startup Window>	<SW>
<Synonym>	<SYN>
<Topic>	
<Topic Area>	<TA>
<Topic Areas Instruction>	<TAI>
<Topics Instruction>	<TI>
<Version>	
<World Script>	<WS>

Guide Script Commands and Parameters Quick Reference

A summary of all Guide Script commands and their parameters, in alphabetical order, follows.

Table B-1 Commands quick reference

Command	Parameters
<3D Button>	<i>buttonUpPict, buttonDownPict, buttonLoc, buttonEvent</i> [, <i>b&wUp</i>] [, <i>b&wDown</i>]
<Allow Prompts>	<i>allow</i>
<App Creator>	<i>creator</i>
<App Logo>	<i>colorLogo</i> [, <i>B&WLogo</i>]
<App Text>	<i>string</i>
<Balloon Menu Text>	<i>balloonText</i>
<Build Sequence>	<i>sequenceName, seqResID</i>
<Checkbox>	<i>checkBoxTitle, checkBoxState</i> [, <i>seqTrue</i>] [, <i>seqFalse</i>] [, <i>checkBoxAnchor</i>] [, <i>checkBoxFont</i>]
<Coach Mark>	<i>coachMarkName</i>
<Comment> or #	
<Default Format>	<i>formatName</i>
<Default Nav Button Set>	<i>navButtonSetName</i>
<Default Prompt Set>	<i>promptSetName</i>

continued

Guide Script Commands and Parameters Quick Reference

Table B-1 Commands quick reference (continued)

Command	Parameters
<Define AppleScript Coach>	<i>coachMarkName</i> [, <i>coachStyle</i>] [, <i>AppleScriptID</i>]
<Define Context Check>	<i>contextCheckName</i> , <i>codeResSpec</i> [, <i>targetApp</i>] [, <i>additionalParam</i>] [, <i>additionalParam</i>] [, ...]
<Define Event>	<i>eventName</i> , <i>targetApp</i> , <i>eventClass</i> , <i>eventID</i> [, <i>IOPTData</i>] [, <i>optKey</i>] [, <i>optData</i>]
<Define Event List>	<i>eventListName</i> , <i>event1</i> [, <i>event2</i>] [, <i>event3</i>] [, <i>event4</i>] [, <i>event5</i>] [, <i>event6</i>]
<Define Format>	<i>formatName</i> , <i>columnCoords</i> [, <i>txFmt</i>] [, <i>txSize</i>] [, <i>txStyle</i>] [, <i>txColor</i>] [, <i>txAlign</i>] [, <i>alignPrompt</i>]
<Define Item Coach>	<i>coachMarkName</i> [, <i>targetApp</i>] [, <i>coachStyle</i>] [, <i>targetWindow</i>] , <i>targetItem</i> [, <i>itemRectangle</i>]
<Define Menu Coach>	<i>coachMarkName</i> [, <i>targetApp</i>] [, <i>coachStyle</i>] , <i>targetMenu</i> [, <i>targetItem</i>] [, <i>itemCoachColor</i>] [, <i>itemCoachStyle</i>]
<Define Nav Button>	<i>buttonName</i> , <i>buttonUpPict</i> , <i>buttonDownPict</i> , <i>dimmedButtonPict</i> , <i>buttonEvent</i> [, <i>b&wUp</i>] [, <i>b&wDown</i>] [, <i>b&wDimmed</i>]
<Define Nav Button Set>	<i>navButtonSetName</i> [, <i>leftNavButton</i>] [, <i>midNavButton</i>] [, <i>rightNavButton</i>]

Table B-1 Commands quick reference (continued)

Command	Parameters
<Define Object Coach>	<i>coachMarkName</i> , <i>targetApp</i> [, <i>coachStyle</i>] [, <i>objectName</i>]
<Define Panel>	<i>panelName</i>
<Define Prompt Set>	<i>promptSetName</i> , <i>promptFirstPanel</i> , <i>promptMiddlePanel</i> , <i>promptLastPanel</i> , <i>promptForPanelsWithControls</i>
<Define Sequence>	<i>sequenceName</i> [, <i>seqDisplayTitle</i>]
<Define Text Block>	<i>textBlockName</i>
<Define Transparent Format>	<i>formatName</i> , <i>columnCoords</i> [, <i>txFmt</i>] [, <i>txSize</i>] [, <i>txStyle</i>] [, <i>txColor</i>] [, <i>txAlign</i>] [, <i>alignPrompt</i>]
<Define Window Coach>	<i>coachMarkName</i> [, <i>targetApp</i>] [, <i>coachStyle</i>] [, <i>targetWindow</i>] [, <i>windowRectangle</i>] [, <i>rectOrigin</i>]
<Delete Index>	<i>indexTerm</i>
<Delete Index Header>	<i>indexTerm</i> , <i>indexHeader</i>
<Delete Index Topic>	<i>indexTerm</i> , <i>indexTopic</i>
<Delete Topic Area>	<i>topicArea</i>
<Delete Topic Area Header>	<i>topicArea</i> , <i>topicAreaHeader</i>
<Delete Topic Area Topic>	<i>topicArea</i> , <i>topicAreaTopic</i>
<Dimmable Button Data>	<i>buttonName</i> , <i>sequenceName</i>
<Else>	
<End If>	
<End Making Sure>	
<End Panel>	

continued

Guide Script Commands and Parameters Quick Reference

Table B-1 Commands quick reference (continued)

Command	Parameters
<End Sequence>	
<End Text Block>	
<Exception>	<i>exceptionWord</i>
<Format>	<i>formatName</i>
<Gestalt>	<i>selector, requiredValue</i>
<Header>	<i>headerPhrase</i>
<Help Menu>	<i>itemString, helpType</i> <i>[, helpCmdKey]</i>
<Hot Object>	<i>eventFunction</i>
<Hot Rectangle>	<i>hotRect, eventFunction</i>
<Hot Text>	<i>hotText, whichOccurrence,</i> <i>eventFunction</i>
<Howdy>	<i>howdyTextBlockName</i>
<If>	<i>condition</i>
<Ignore>	<i>ignoreWord</i>
<Include>	<i>sourceFileName</i>
<Index>	<i>indexTerm [, visible] [, key]</i>
<Index Instruction>	<i>indexInstruction</i>
<Index Sorting>	<i>orderingKey</i>
<Insert Index Header>	<i>header, indexTerm [, sortOrder]</i>
<Insert Index Topic>	<i>topic, seqName, indexTerm</i> <i>[, sortOrder]</i>
<Insert Sequence>	<i>sequenceName</i>
<Insert Topic Area Header>	<i>header, topicArea [, sortOrder]</i>
<Insert Topic Area Topic>	<i>topic, seqName, topicArea</i> <i>[, sortOrder]</i>

Table B-1 Commands quick reference (continued)

Command	Parameters
<Jump Sequence>	<i>sequenceName</i>
<Launch New Sequence>	<i>sequenceName</i>
<Look For Instruction>	<i>lookForInstruction</i>
<Look For Results Instruction>	<i>resultsInstruction</i>
<Look For Search Btn Instruction>	<i>buttonInstruction</i>
<Look For String>	<i>searchPhrase</i>
<Make Sure>	<i>condition</i> , <i>oopsOrContinueSequenceName</i>
<Max Height>	<i>height</i>
<Min Height>	<i>height</i>
<Mixin>	<i>symNameOrStartResNum</i>
<Mixin Match>	<i>matchingCreator</i>
<On Panel Create>	<i>eventFunction</i>
<On Panel Destroy>	<i>eventFunction</i>
<On Panel Hide>	<i>eventFunction</i> [, <i>firstOrAlways</i>]
<On Panel Show>	<i>eventFunction</i> [, <i>firstOrAlways</i>]
<Panel>	<i>panelName</i>
<Panel Prompt>	<i>promptSetName</i>
<PICT>	<i>pictGraphic</i> , <i>location</i> [, <i>b&wPict</i>]
<QuickTime>	<i>QTMovie</i> , <i>location</i> , <i>QTcontroller</i> [, <i>moviePict</i>]
<Radio Button>	<i>buttonTitle</i> , <i>buttonState</i> [, <i>seqTrue</i>] [, <i>seqFalse</i>] [, <i>buttonAnchor</i>] [, <i>buttonFont</i>]

continued

Guide Script Commands and Parameters Quick Reference

Table B-1 Commands quick reference (continued)

Command	Parameters
<Radio Button Launch New Seq>	<i>buttonTitle</i> , <i>buttonState</i> [, <i>seqTrue</i>] [, <i>seqFalse</i>] [, <i>buttonAnchor</i>] [, <i>buttonFont</i>]
<Replace Sequence>	<i>oldSequenceName</i> , <i>newSequenceName</i>
<Resource>	<i>fileName</i> , <i>resType</i> [, <i>whichResource</i>]
<Seq Nav Button Set>	<i>navButtonSetName</i>
<Sequence Prompt Set>	<i>promptSetName</i>
<Skip If>	<i>condition</i>
<Sorting>	<i>method</i>
<Standard Button>	<i>buttonTitle</i> , <i>buttonLoc</i> , <i>buttonEvent</i> [, <i>buttonFont</i>]
<Starting Res Number>	<i>resID</i>
<Start Making Sure>	<i>condition</i> , <i>oopsOrContinueSequenceName</i>
<Startup Window>	<i>windowType</i> , <i>accessScreenOptions</i>
<Synonym>	<i>indexTerm</i> , <i>synonym</i>
<Topic>	<i>topicPhrase</i> , <i>sequenceName</i>
<Topic Area>	<i>topicAreaPhrase</i> [, <i>mixInOrder</i>]
<Topic Areas Instruction>	<i>topicAreaInstruction</i>
<Topics Instruction>	<i>topicsInstruction</i>
<Version>	<i>longVers1BottomOfGetInfo</i> , <i>shortVers1ForFinderListView</i> [, <i>longVers2TopOfGetInfo</i>]
<World Script>	<i>scriptCode</i> , <i>regionCode</i>

SurfWriter Guide and Its Source Files

This appendix provides a specific example—using SurfWriter Guide—to help guide you through the process of scripting your source files. This appendix integrates much of the information provided up to now, by showing a specific implementation of a guide file. Read this appendix when you're ready to start scripting your source files.

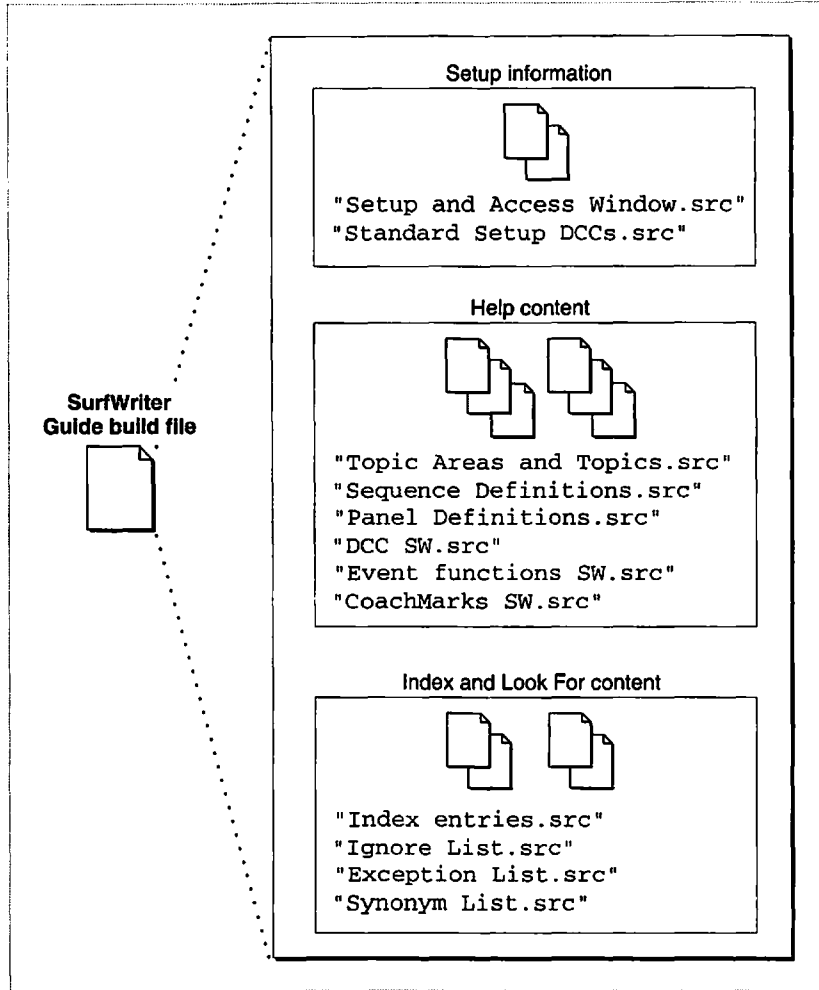
This appendix shows the guide file for the SurfWriter application and includes the source files containing the Guide Script commands used to create this guide file. It takes you through each source file used to create SurfWriter Guide, giving explanatory text where necessary.

The source files used in this example are also provided online on the CD-ROM accompanying this book, in the folder Apple Guide:Authoring:Documentation:Example Source Files Appdx C. You can find the compiled guide file in the folder Apple Guide:Authoring:Documentation:Example Guide.

Getting Started

Figure C-1 illustrates the structure of the source files for SurfWriter Guide. These source files can be grouped into three distinct areas: files that specify setup information, files that provide help content, and files that provide Index and Look For content.

Figure C-1 The organization of the source files for SurfWriter Guide



The rest of this section describes the files used to specify the initial setup information and gives explanatory text where necessary. "Help Content" beginning on page C-11 and "Index and Look For Content" beginning on page C-39 describe other files used to create SurfWriter Guide and give

additional information related to the specific commands specified in the source files.

SurfWriter Guide Build File

The source files for SurfWriter Guide are organized in small sections, to make the files convenient to use and easier to manage. For example, separate files are used for SurfWriter Guide's index terms, synonym list, ignore list, and exception list. Similarly, the source files containing sequence and panel definitions are organized so that their information can be readily accessed.

The build file for SurfWriter Guide is shown in Listing C-1. The rest of the sections in this appendix elaborate on these files.

Listing C-1 A build file ("Build file SURF.src" file)

```
#the following file specifies event, nav button, & format definitions
# plus prompts, coachmarks, Help menu info, and
# the type of access window
<Include> "Setup and Access Window.src"
#the following file specifies the standard context checks
<Include> "Standard Setup DCCs.src"
#the following file specifies application-defined context checks
<Include> "DCC SW.src"
#the following file specifies application-defined event functions
<Include> "Event functions SW.src"
#the following file specifies coachmarks
<Include> "CoachMarks SW.src"
#the following files specify the guide file content
<Include> "Topic Areas and Topics.src"
<Include> "Sequence Definitions.src"
<Include> "Panel Definitions.src"
#use the following file only if you have an XTND translator installed
#<Include> "Panels with StyleInfo.src"
#the following files specify the guide file index and Look For content
<Include> "Index entries.src"
```

```
<Include> "Ignore List.src"  
<Include> "Exception List.src"  
<Include> "Synonym List.src"
```

Using Standard Files

Included on the CD-ROM that accompanies this book are two files that are separate from the SurfWriter Guide source files but that are commonly used in creating guide files. These two files, Standard Setup and Standard Resources, are in the Standard Includes folder, which is located in the Apple Guide:Authoring folder. These two standard files are typically used as templates. For example, SurfWriter Guide uses information from these two files and customizes the information when necessary, as described in "Customizing the Setup Information" beginning on page C-5.

The Standard Setup File

The Standard Setup file contains the Guide Script commands that give basic information about a guide file, such as

- the name of the guide file as it appears in the Help menu
- the balloon text for the guide file's menu item
- the application associated with the guide file
- the type of access window (full, single, or simple) and the access screen that is initially active (howdy, Topics, Index, or Look for)
- howdy text for the initial access window
- application logo information for the access window
- Finder version resources (to display in the Get Info box of the guide file)
- prompt set definitions, including the default prompt set
- navigation bar button definitions, including the Huh? and GoStart buttons, as well as the default navigation bar button set
- format definitions, including the Full, Tag, and Body formats
- specifications of the maximum and minimum height allowed for a panel

- event definitions for the Huh? button ("DoHuh"), the GoStart button ("GoStart"), and the Continue or OK buttons in Continue and Oops panels ("GoBack")
- context check definitions for the external modules provided in the Standard Resources file

By editing the Standard Setup file, you can quickly create a guide file that already has standard elements (such as the Huh? and GoStart buttons) defined and customize specific elements (such as your guide file's menu item, application logo, howdy text, and type of access window) as appropriate for your guide file.

The Standard Resources File

The Standard Resources file contains:

- 'PICT' resources for the Continue, Huh?, and GoStart buttons
- templates of 'PICT' resources for an application logo
- 'extm' resources (external modules) containing the code for the context checks defined in the Standard Setup file

Customizing the Setup Information

The file "Setup and Access Window.src" specifies the basic setup information for SurfWriter Guide. It is based on the Standard Setup file, with additions that are specific for this guide file. Note that the file "Standard Setup DCCs.src" contains the context check definitions from the Standard Setup file.

Listing C-2 through Listing C-6 show the "Setup and Access Window.src" file.

Navigation Information and Formats

As shown in Listing C-2, the "Setup and Access Window.src" file first imports the resources from the Standard Resources file, then defines the same events, navigation buttons, and formats as the Standard Setup file. It also defines additional formats specific to SurfWriter Guide.

Listing C-2 Events, navigation buttons, and formats (from the "Setup and Access Window.src" file)

```

<Resource> "Standard Resources", ALL

#these events & definitions are also defined in "Standard Setup"
<Define Event> "DoHuh", 's***', 'help', 'dhuh'
<Define Event> "GoStart", 's***', 'help', 'stac'
<Define Event> "GoBack", 's***', 'help', 'gobk'

<Define Nav Button> "Huh?", 1101, 1111, 1121, DIMMABLE
<Define Nav Button> "GoStart", 1103, 1113, 1123, GoStart()
<Define Nav Button Set> "Std nav bar", "GoStart", "Huh?"

<Define Format> "Tag", Column(6,0,54),"Espy Sans Bold",10,plain,,
right, false
<Define Format> "Body", Column(6,65,330),"Espy Serif",10,plain,,
left, true
<Define Format> "Full", Column(6,11,330),"Espy Serif",10,plain,,Left,
false
#if you use styles in your help content and your source files are
# styled text, omit "plain" from format
<Define Format> "Full2",Column(6,11,330),"Espy Serif",10,,,Left,false
<Default Format> "Full"
#define other formats that specify style information (if your
# source files are plain text rather than styled text)
# (these formats are used only with the "CreateCustomIntro" panel)
<Define Format> "PlainText", Column(6,7,330),"Espy Serif",
10,plain,,,false
<Define Format> "UnderlineText", Column(20,41,115),"Espy Sans Bold",
10,underline,,,false
<Define Format> "PlainTextReset", Column(20,116,330),"Espy Serif",
10,plain,,,false
<Define Format> "PlainTextNormal", Column(34,7,330),"Espy Serif",
10,plain,,,false

```

Prompt Sets

Listing C-3 defines the prompt sets used by SurfWriter Guide.

Listing C-3 Prompt sets (from the "Setup and Access Window.src" file)

```
#Define your prompt sets.
#Provide four strings to <Define Prompt Set>; the strings specify
# prompts for:
# 1. the first panel in a sequence
# 2. any middle panel in a sequence that does not have
#    radio buttons, checkboxes, or standard buttons
# 3. the last panel in a sequence
# 4. any panel with controls (radio buttons, checkboxes,
#    or standard buttons)
#
<Define Prompt Set> "standard" , "To begin, click the right arrow.",
"Do this step, then click the right arrow.", "Do this step, then
you're done.", "Make your choice, then click the right arrow."

<Define Prompt Set> "standard2" , "To begin, click the right arrow.",
"Click the left arrow to go back or the right arrow to continue.",
"That's all, you're done!", "Make your choice, then click the right
arrow."

<Default Prompt Set> "standard2"

<Define Prompt Set> "introprompts" , "To begin, click the right
arrow. ", " ", " ", " "

<Define Prompt Set> "doThisprompt" , "Do this step, then click the
right arrow.", "Do this step, then click the right arrow.", " ", "Do
this step, then click the right arrow."
```

SurfWriter Guide and Its Source Files

```
<Define Prompt Set> "YouAreDone" , " ", " ", "That's all, you're
done.", " "
```

```
<Define Prompt Set> "YouAreDone2" , " ", " ", "That's all, you're
done.", "That's all, you're done."
```

```
<Define Prompt Set> "continuePrompt" , "After the dictionary opens,
click Continue.", "After the dictionary opens, click Continue.",
"After the dictionary opens, click Continue.", " "
```

```
<Define Prompt Set> "Defn&HuhPrompts" , "Read this information, then
you're done.", "Read this information, then you're done.", "Read this
information, then you're done.", " "
```

Help Menu Information

Listing C-4 defines the information specific to SurfWriter's Help menu (the menu item text and balloon text for the guide file).

Listing C-4 Help menu information (from the "Setup and Access Window.src" file)

```
# ***** Help menu information *****
#define how the name of this guide file should appear in
# SurfWriter's Help menu
#the menu item name is "SurfWriter Guide"
#the guide file for this menu item is of type HELP
#if you use the cmd like this, it does NOT give you Cmd-Key symbol
# or ? in the Help menu
#<Help Menu> "SurfWriter Guide", HELP
#therefore, specify the command like this
<Help Menu> "SurfWriter Guide", HELP, "?"
```

```
#use this command to specify the text that appears for this guide file
# in your application's Help menu; that is, when the cursor is in
# the SurfWriter Guide menu item and Balloon Help is on,
```

```
# the Finder displays this text in a help balloon
<Balloon Menu Text> "Provides information and instructions to assist
you in accomplishing specific tasks with SurfWriter"

#To specify that your guide file appear only in the Help menu
# of your application, use this command and specify your application's
# creator, for example:
# <App Creator> 'WAVE'
```

Access Window Information

Listing C-5 defines the access window information for SurfWriter Guide. Note that because this source file specifies the <App Logo> command, the Standard Resources file must not contain 'PICT' resources with resource IDs 501 and 502 (these resources have been removed from the Standard Resources file provided in the Example Source Files Appdx C folder). This guide file uses a Full Access window with howdy text initially active, as shown by the <Startup Window> command.

Listing C-5 Access window startup (from the "Setup and Access Window.src" file)

```
# *****Access window startup information*****
#Define either SurfWriter app logo or app text.
#Apple Guide displays the app logo or app text
# in the upper-left corner of SurfWriter Guide's Full Access window.
#<App Text> "SurfWriter Guide"
#if you choose to use app logo instead,
# define the filename that contains a PICT of your app's logo
# for example, <App Logo> "MyAppLogoPict", "MyAppLogoB&WPict"
#(Note that the file "Standard Resources" contains templates with
# PICT resource IDs 501 and 502 that you can modify appropriately and
# use as the application logo picture associated with your guide).
#If you modify the PICT resources with IDs 501 and 502 then import the
# "Standard Resources" file, you can omit the <App Logo> command.
#If you store your app logo in a separate file,
```

APPENDIX C

SurfWriter Guide and Its Source Files

```
# then remove the PICT resources with resource IDs 501 and 502
# from the "Standard Resources" file.
```

```
<App Logo> "SurfWriter App Color Logo", "SurfWriter App B&W Logo"
```

```
#define startup window, this is the window that Apple Guide
# first displays when the user chooses the menu item SurfWriter Guide
# If you specify a full access window, you can display HOWDY info
# OR you can specify which of the three buttons is initially active
# this example specifies HOWDY
<Startup Window> FULL, HOWDY
```

```
#You can choose FULL, SINGLE, or PRESENTATION as the startup window
#Full access windows allow topics, index, and look for, and Howdy
#Single access windows allow only topics and howdy
#Presentation allows a single sequence only
```

```
#define the text for the Howdy screen
<Define Text Block> "Howdy Text"
Welcome to personal help for SurfWriter.
```

To start, click Topics, Index, or Look For.

Topics shows general categories and Index lists key words.
Look For lets you search for help according to key words you type.

```
To learn basic skills, choose the
Tutorial item from the ? menu.
<End Text Block>
```

```
#now specify the defined howdy text
<Howdy> "Howdy Text"
```

Finder Version Information

Listing C-6 provides customized strings for the Finder version resources.

Listing C-6 Finder version information (from the "Setup and Access Window.src" file)

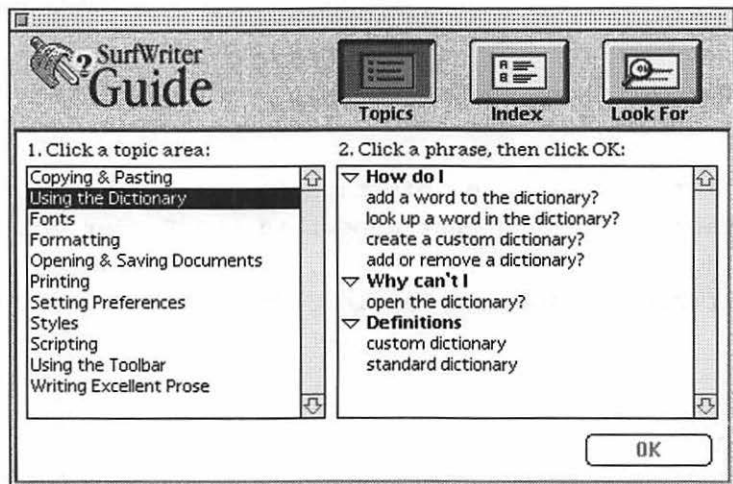
```
# *****Finder file information*****
#Specify the version strings that appear in the Finder Get Info window
# for your guide file
<Version> "SurfWriter Guide 1.0 ©Apple Computer, Inc.", "1.0"
```

Help Content

This section focuses on SurfWriter Guide by describing its source files for topic areas and topics, sequences, panels, coachmarks, context checks, and event functions. Illustrations of SurfWriter Guide's access window and panels are also shown, so that you can map this onscreen help to SurfWriter Guide's source file content.

Topic Areas and Topics

The file "Topic Areas and Topics.src" specifies the topic areas and topics for SurfWriter Guide. Figure C-2 shows these topic areas and shows the topics for one specific topic area, "Using the Dictionary".

Figure C-2 The access window with Topics selected

The Guide Script commands that define the topic areas for SurfWriter Guide's access window are shown in Listing C-7. For each topic area, the associated headers and topics are also defined. SurfWriter Guide provides two complete topic areas, "Using the Dictionary" and "Using the Toolbar". Other topic areas use placeholders for topics, so that the source file is easier to follow.

Listing C-7 Topic areas and topics ("Topic Areas and Topics.src" file)

```
#define topic areas -- when the user chooses the Topics button,
# Apple Guide displays the topic areas defined
# by <Topic Area> commands
#Apple Guide displays topic areas on the left side of a full
# access screen

#Topic areas are often followed by <Header> commands
# and <Topic> commands
#You use header commands to group one or more topics
# in the same category
```



```

#Headers are not required --
# you can choose to have all topics with no groupings by headers
# or you can have headers with each topic grouped with a header
#Note that a topic area MUST have at least 1 topic associated with it

# A header command defines the bold headers that Apple Guide displays
# in the right column when the user selects its associated topic area

# A topic command defines a topic that Apple Guide displays
# in the right column; the topic can either appear by itself or
# have a header associated with it
#Note that you should either have all single topics or
# have all topics grouped with headers
#Apple Guide expands and compresses the topics under a header
# when the user clicks the triangle to the left of the header

#topic areas are displayed in the same order as they appear
# in the source file

#Tip: You can use style or color information in Guide Script
# commands to make your source files easier to read

<Topic Area> "Copying & Pasting" #topic area on left column of screen
    <Header> "How do I"           #header for right column when user
                                # chooses "Copying & Pasting"
    #topics displayed/expanded/etc with "Copying & Pasting"
        <Topic> "placeholder for topic?","SequenceGeneric"
        <Topic> "another placeholder for topic?","SequenceGeneric"

```

```

<Topic Area> "Using the Dictionary"
  <Header> "How do I"
    <Topic> "add a word to the dictionary?","SequenceAddWords"
    <Topic> "look up a word in the dictionary?","SequenceGeneric"
    <Topic> "create a custom dictionary?","
SequenceCreateCustomDictionary"
    <Topic> "add or remove a dictionary?","SequenceGeneric"
  <Header> "Why can't I"
    <Topic> "open the dictionary?","SequenceGeneric"
  <Header> "Definitions"
    <Topic> "custom dictionary","SequenceDefnCustomDictionary"
    <Topic> "standard dictionary","SequenceDefnStdDictionary"

<Topic Area> "Fonts"
  <Header> "How do I"
    <Topic> "placeholder for topic?","SequenceGeneric"
<Topic Area> "Formatting"
  <Header> "How do I"
    <Topic> "placeholder for topic?","SequenceGeneric"
<Topic Area> "Opening & Saving Documents"
  <Header> "How do I"
    <Topic> "placeholder for topic?","SequenceGeneric"
<Topic Area> "Printing"
  <Header> "How do I"
    <Topic> "placeholder for topic?","SequenceGeneric"
<Topic Area> "Setting Preferences"
  <Header> "How do I"
    <Topic> "placeholder for topic?","SequenceGeneric"
<Topic Area> "Styles"
  <Header> "How do I"
    <Topic> "placeholder for topic?","SequenceGeneric"
<Topic Area> "Scripting"
  <Header> "How do I"
    <Topic> "placeholder for topic?","SequenceGeneric"

```

```

<Topic Area> "Using the Toolbar"
  <Header> "How do I"
    <Topic> "use the tools in the toolbar?","Toolbar"
<Topic Area> "Writing Excellent Prose"
  <Header> "How do I"
    <Topic> "placeholder for topic?","SequenceGeneric"

```

Sequences

This section focuses on the file “Sequence Definitions.src”, which specifies the sequences for each topic in SurfWriter Guide. The main sequences for each topic area are shown in Listing C-8 through Listing C-12.

The Placeholder Sequence

SurfWriter Guide uses the placeholder sequence shown in Listing C-8 for all topics other than those associated with “Using the Dictionary” and “Using the Toolbar”. Thus, the guide file displays all topic areas and topics, while allowing the actual sequences to be added incrementally.

Listing C-8 Placeholder sequence (from the “Sequence Definitions.src” file)

```

#Here's a sequence definition used as a placeholder
<Define sequence> "SequenceGeneric", "How do I *do this task*?"
<SNBS> "Std nav bar"
  <Panel> "PanelGeneric"
<End sequence>

```

“How do I use the tools in the toolbar?” Sequence

The sequence definition shown in Listing C-9 specifies the navigation button set for the sequence and then lists all the panels that are part of the sequence. This sequence has one panel (“Tools with Tip”) that launches another sequence if the user clicks the panel’s Tip button. The sequence “Toolbar TipSeq” shows the sequence for the Tip panel.

Listing C-9 Sequence for "How do I use the tools in the toolbar?" (from the "Sequence Definitions.src" file)

```
#A sequence defn for a topic of the topic area "Using the Toolbar"
<Define sequence> "Toolbar", "How do I use the tools in the toolbar?"
  <SNBS> "Std nav bar"
  <Panel> "Use Tools"
  <Panel> "Tools 2"
  <Panel> "Tools 3"
  <Panel> "Tools with Tip"
  <Panel> "Tools 4"
<End sequence>

#A sequence defn for a Tip of "How do I use the tools in the toolbar?"
<Define sequence> "Toolbar TipSeq", "Tip: How do I use the tools in
the toolbar?"
  <SNBS> "Std nav bar"
  <Panel> "Toolbar Tip"
<End sequence>
```

"How do I add a word to the dictionary?" Sequence

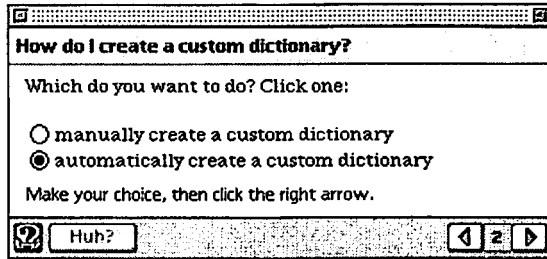
The sequence definition shown in Listing C-10 includes one <Skip If> command and three <Make Sure> commands. The second panel in the sequence is skipped if the condition specified in the command is true. This sequence uses Continue panels, as shown by the <Make Sure> commands. If the condition specified in the <Make Sure> command is not true, a Continue panel is displayed. (See the source file on the CD-ROM for an example of a sequence that uses an Oops panel.) See Figure C-4 on page C-25 for an illustration of the panels in this sequence.

Listing C-10 Sequence for “How do I add a word to the dictionary?” (from the “Sequence Definitions.src” file)

```
#A sequence defn for a topic of the topic area "Using the Dictionary"
<Define sequence> "SequenceAddWords", "How do I add a word to the
dictionary?"
    <SNBS> "Std nav bar"
    <Panel> "AddWords1"
<Skip If> ActiveWindow('ttxt', "Dictionary") AND ActiveAppIs('ttxt')
    <Panel> "AddWords2"
<Make Sure> ActiveAppIs('ttxt'), "SWContinuePanelSeq"
<Make Sure> ActiveWindow('ttxt', "Dictionary"), "SWContinuePanelSeq"
    <Panel> "AddWords3"
<Make Sure> ActiveWindow('ttxt', "Dictionary"), "SWContinuePanelSeq"
    <Panel> "AddWords4"
<End sequence>
#*****Example of a Continue sequence*****
#A sequence definition for a Continue sequence in the above topic,
# "How do I add a word to the dictionary?"
<Define Sequence> "SWContinuePanelSeq", "Oops"
    <Panel> "SWContinuePanel"
<End Sequence>
```

“How do I create a custom dictionary?” Sequence

SurfWriter Guide includes the topic “How do I create a custom dictionary?” The sequence for this topic includes a panel containing radio buttons, as shown in Figure C-3.

Figure C-3 A panel with radio buttons

SurfWriter Guide allows the user to create a custom dictionary either automatically or manually. To implement this, a panel with radio buttons is used and the sequence definition (shown in Listing C-11) includes an `<If>` command that tests the state of a radio button and then branches accordingly.

When using SurfWriter Guide, if the user chooses to create a custom dictionary automatically (by clicking the appropriate radio button then clicking the right arrow button), the `<If>` command and the statements immediately following the `<If>` command are executed (see Listing C-11). Note that the panels in this branch include `<Skip If>` and `<Make Sure>` commands. This source file uses the `ActiveWindow`, `OpenWindow`, and `ActiveAppIs` condition functions (these context checks are defined in the Standard Setup file) to determine whether the folder is active and if not, it automatically activates or opens the folder for the user.

The panels corresponding to the user choosing the manual method follow the `<Else>` command. The `ActiveWindow` and `ActiveAppIs` condition functions are also used in this branch to determine whether a particular window is active and open. Both branches use Continue panels if `ActiveWindow` returns false. Note that it's important that SurfWriter Guide determine not only whether a window is open but also whether it is active, especially when using coachmarks for a window. Otherwise, the coachmark may be drawn in an inappropriate window.

Finally, Listing C-11 shows that the sequence uses a common closure panel, which is presented to the user no matter which branch they chose. See Figure C-5 on page C-29 and Figure C-6 on page C-30 for illustrations of the panels in this sequence.

Listing C-11 Sequence for "How do I create a custom dictionary?" (from the "Sequence Definitions.src" file)

```
#A sequence defn for a topic of the topic area "Using the Dictionary"
<Define sequence> "SequenceCreateCustomDictionary", "How do I create
a custom dictionary?"
    <SNBS> "Std nav bar"
    <Panel> "CreateCustomIntro"
    <Panel> "CreateCustomDecision"
#branch based on state of radio buttons in previous panel
<IF> radioButtonState("automatically create a custom dictionary",
"CreateCustomDecision")
#if user chose "automatically" do this sequence of panels
<Skip If> ActiveWindow('MACS', "SurfWriter Scripts") AND
ActiveAppIs('MACS')
    <Panel> "CreateCustomAuto1"
<Make Sure> ActiveWindow('MACS', "SurfWriter Scripts"),
"SWContinueSeqCustomAuto1"
<Make Sure> OpenWindow('MACS', "SurfWriter Scripts"),
"SWContinueSeqCustomAuto1"
<Make Sure> ActiveAppIs('MACS'), "SWContinueSeqCustomAuto1"
    <Panel> "CreateCustomAuto2"
<Else>
    #if user chose "manually" do this sequence of panels
<Skip If> ActiveWindow('ttxx', "Dictionary") AND ActiveAppIs('ttxx')
    <Panel> "CreateCustomManual1"
<Make Sure> ActiveAppIs('ttxx'), "SWContinuePanelSeq"
<Make Sure> ActiveWindow('ttxx', "Dictionary"),
"SWContinueSeqCustomManual1"
    <Panel> "CreateCustomManual2"
<Make Sure> ActiveAppIs('ttxx'), "SWContinuePanelSeq"
<Make Sure> ActiveWindow('ttxx', "Dictionary"),
"SWContinueSeqCustomManual1"
    <Panel> "CreateCustomManual3"
<End If>
```

```
#common closure panel
  <Panel> "CreateCustomAllDone"
<End sequence>

#A sequence definition for a Continue sequence in the topic,
# "How do I create a custom dictionary (automatically)?"
<Define Sequence> "SWContinueSeqCustomAuto1", "Oops"
  <Panel> "SWContinuePanelCustomAuto1"
<End Sequence>

#A sequence defn for another Continue sequence in the topic,
# "How do I create a custom dictionary (manually)?"
<Define Sequence> "SWContinueSeqCustomManual1", "Oops"
  <Panel> "SWContinuePanelCustomManual1"
<End Sequence>
```

The remaining sequence definitions used by panels of the sequence “How do I create a custom dictionary?” are self-explanatory and are shown in Listing C-12.

Listing C-12 Sequence definitions for Huh?, Definition, and Related Topics (from the “Sequence Definitions.src” file)

```
#A sequence defn for "Definition:AppleScript"
<Define sequence> "Defns:AppleScript", "Definition: AppleScript"
  <SNBS> "Std nav bar"
  <Panel> "HotT AppleScript"
<End Sequence>

#A sequence definition for a Huh? sequence
<Define sequence> "HuhCompareManualAndAuto", "Comparison of manual
and automatic methods"
  <SNBS> "Std nav bar"
  <Panel> "CompareManualAndAuto"
<End Sequence>
```



```
#A sequence defn for a Tip sequence of creating a custom dictionary
<Define sequence> "Tip:CustomDictionary", "Tip: How do I create a
custom dictionary?"
```

```
    <SNBS> "Std nav bar"
```

```
    <Panel> "TipForCustomDictionary"
```

```
<End sequence>
```

```
#A seq defn for Related Topics sequence of creating custom dictionary
<Define sequence> "Related Topics:CustomDictionary", "Related Topics:
How do I create a custom dictionary?"
```

```
    <SNBS> "Std nav bar"
```

```
    <Panel> "RelatedTopicsForCustomDictionary"
```

```
<End sequence>
```

```
#A sequence definition for a Huh? sequence
```

```
<Define sequence> "HuhScriptsFolder", "Scripts in the SurfWriter
Scripts folder"
```

```
    <SNBS> "Std nav bar"
```

```
    <Panel> "ScriptsFolder"
```

```
<End sequence>
```

```
#Here's a seq defn for the panel that defines "Standard Dictionary"
```

```
<Define sequence> "SequenceDefnStdDictionary", "Definition: standard
dictionary"
```

```
    <SNBS> "Std nav bar"
```

```
    <Panel> "PanelDefineStandardDictionary"
```

```
<End sequence>
```

```
#Here's a sequence defn for the panel that defines "Custom Dictionary"
```

```
<Define sequence> "SequenceDefnCustomDictionary", "Definition: custom
dictionary"
```

```
<SNBS> "Std nav bar"
```

```
    <Panel> "PanelDefineCustomDictionary"
```

```
<End sequence>
```

Panels

This section focuses on the file "Panel Definitions.src", which specifies the panels for SurfWriter Guide.

The Placeholder Panel

SurfWriter Guide uses the placeholder panel shown in Listing C-13 for all topics other than those associated with "Using the Dictionary" and "Using the Toolbar". Thus the guide file displays a generic panel for these topics. In this way, the actual content of the panels can be added incrementally and at a later time.

Listing C-13 Placeholder panel (from the "Panel Definitions.src" file)

```
<Define Panel> "PanelGeneric"  
Placeholder for information that you supply.  
<End Panel>
```

"How do I use the tools in the toolbar?" Panels

The panel definitions shown in Listing C-14 are simple examples of various Apple Guide features. They show how to place a standard button on a panel, how to use the Tag and Body formats, how to specify prompts, and how to use a Tip or Related Topics button to launch a new sequence.

Listing C-14 Panels for "How do I use the tools in the toolbar?" (from the "Panel Definitions.src" file)

```
<Define Panel> "Use Tools"  
<Panel Prompt> "introToolsprompts"  
The SurfWriter toolbar contains tools that you use to create and  
manipulate graphics.
```

For an overview of each tool in the toolbar, click **Toolbar Tour**. For background information on graphics, click **Designing Graphics**.

```

#When you specifically place an object you place it relative to the
# current pen location. Note that the current pen location isn't reset
# after an object is absolutely placed. The current pen location's
# horizontal coordinate is the left edge of the current format;
# the vertical coordinate corresponds to the last object not
# specifically placed using coordinates.
# it's (x, y)
#The first button is placed 50 pixels to the right and 20 pixels
# down of the current pen location; the second button is placed 50
# pixels to the right and 80 pixels down of the current pen location.
<Standard Button> "Toolbar Tour", Point(50,20), doNothingEvent()
<Standard Button> "Designing Graphics", Point(50,80), doNothingEvent()
<End Panel>

<Define Panel> "Tools 2"
<Panel Prompt> "doThisprompt"
<Format> "Tag"
Do This
<Format> "Body"
Click the Pencil icon in the toolbar.
<End Panel>

<Define Panel> "Tools 3"
#panel that shows default features
<Panel Prompt> NONE

.

<End Panel>

```

```
<Define Panel> "Tools with Tip"
#panel that shows a Tip button
<Panel Prompt> "doThisprompt"
<Format> "Tag"
Do This
<Format> "Body"
Click the Pen icon in the toolbar.
```

For information about quickly selecting tools, click the Tip button.

```
<Standard Button> "Tip", RIGHT, LaunchNewSequenceNewWindow("Toolbar
TipSeq")
<End Panel>
```

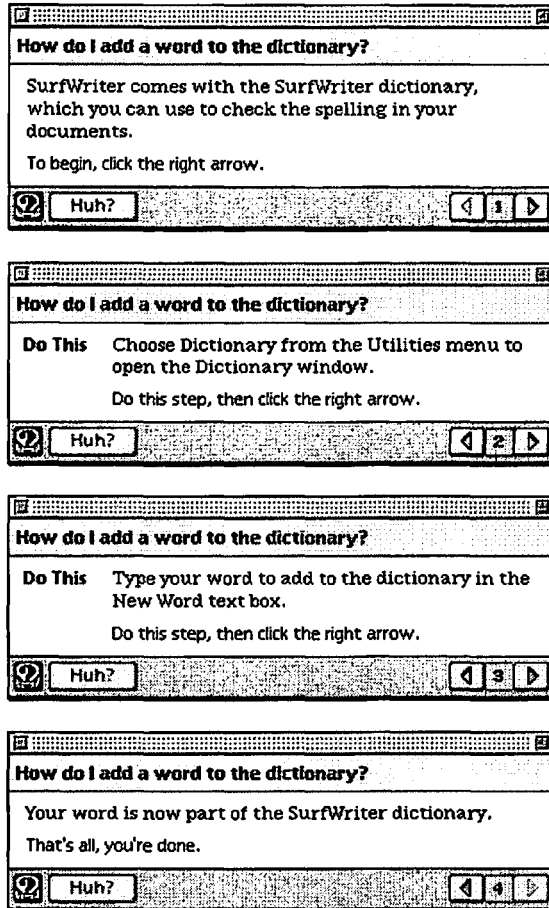
```
<Define Panel> "Toolbar Tip"
<Panel Prompt> "Defn&HuhPrompts"
You can also select tools by using Command-key equivalents. See the
Shortcuts Guide for a complete list of these keys.
<End Panel>
```

```
<Define Panel> "Tools 4"
<Panel Prompt> "YouAreDone2"
<Format> "Tag"
Do This
<Format> "Body"
Close the toolbar.
```

```
<Standard Button> "Related Topics", CENTER, doNothingEvent()
<End Panel>
```

"How do I add a word to the dictionary?" Panels

SurfWriter Guide includes the topic "How do I add a word to the dictionary?"
Figure C-4 shows illustrations of the panels associated with this topic.

Figure C-4 “How do I add a word to the dictionary?” panels

The panel definitions for this topic (see Listing C-15) show use of coachmarks and show how to implement a Continue panel. Note that to perform an action for the user, the Continue panel includes an <On Panel Show> command that specifies an event function. This guide file uses the `DoAppleScript` built-in event function to execute a specified script. This particular script opens a file containing a picture of the dictionary. Note that this guide file was originally

designed for use with the SurfWriter application; however, for illustrative purposes SimpleText is used in this example.

Listing C-15 Panels for "How do I add a word to the dictionary?" (from the "Panel Definitions.src" file)

```
<Define Panel> "AddWords1"
  <Panel Prompt> "introprompts"
SurfWriter comes with the SurfWriter dictionary, which you can use to
check the spelling in your documents.
<End Panel>

<Define Panel> "AddWords2"
  <Panel Prompt> "doThisPrompt"
<Coach Mark> "UtilsOpenDictionary"
<Format> "Tag"
Do This
<Format> "Body"
Choose Dictionary from the Utilities menu to open the Dictionary
window.
<End Panel>

<Define Panel> "AddWords3"
  <Panel Prompt> "doThisprompt"
<Format> "Tag"
Do This
<Format> "Body"
Type your word to add to the dictionary in the New Word text box.
#for SurfWriter, you would use:
#<Coach Mark> "DictionaryNewWord"
#for an application similar to SimpleText:
<Coach Mark> "DictionaryNewWordSimpleText"
<End Panel>

<Define Panel> "SWContinuePanel"
```

```
#open the dictionary for the user, by sending an Apple event to the
# SurfWriter application requesting it to open the Dictionary window
#Apple Guide sends this event to SurfWriter when it shows this panel
#<On Panel Show> SWOpenDictionary("Dictionary")
```

```
<Panel Prompt> "continuePrompt"
```

Please wait a moment. Apple Guide is assisting you by opening the dictionary.

```
<3D Button> 1070, 1072, Center, GoBack()
```

```
#
#(For illustrative purposes, this example uses SimpleText instead of
# SurfWriter.)
```

```
<On Panel Show> DoAppleScript(":SurfWriter Scripts
src:OpenDictionarySimpleText")
```

```
<End Panel>
```

```
<Define Panel> "AddWords4"
```

```
<Panel Prompt> "YouAreDone"
```

Your word is now part of the SurfWriter dictionary.

```
<End Panel>
```

```
#Example of an Oops panel
```

```
 #(If you use Oops instead of Continue panel in the above sequence)
```

```
<Define Panel> "SWOopsPanel"
```

```
<Panel Prompt> NONE
```

```
<Format> "Tag"
```

```
Oops
```

```
<Format> "Body"
```

You did not open the dictionary. Click OK for instructions (or open the dictionary, then click OK).

```
<Standard Button> "OK", Center, GoBack()
```

```
<End Panel>
```

“How do I create a custom dictionary?” Panels

SurfWriter Guide also includes the topic “How do I create a custom dictionary?” Figure C-5 and Figure C-6 show illustrations of the panels associated with this topic. Figure C-5 shows the panels a user views when using the automatic method, and Figure C-6 shows the panels a user views when using the manual method.

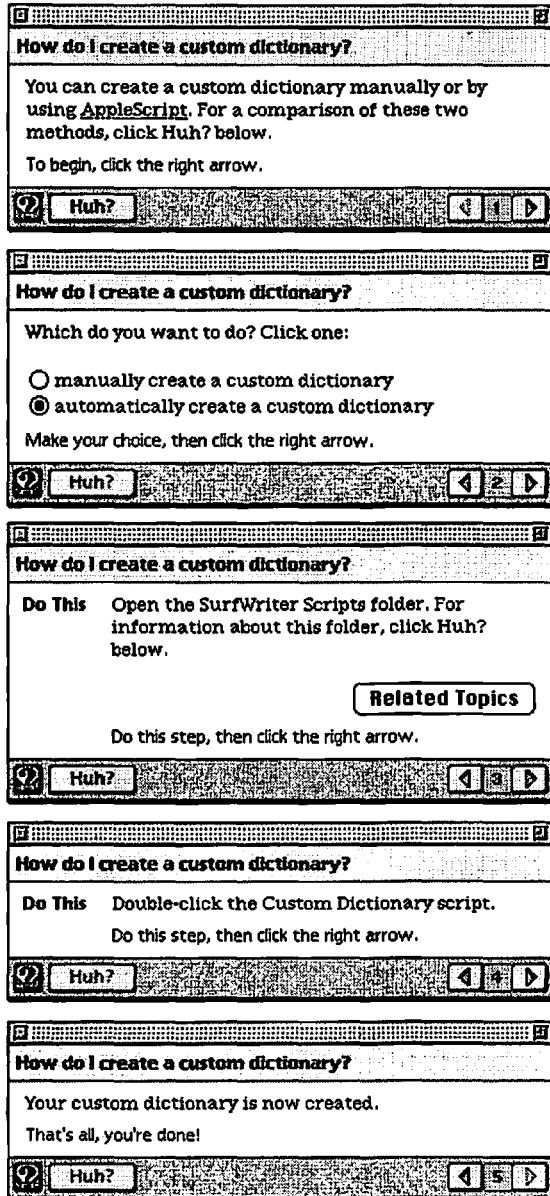
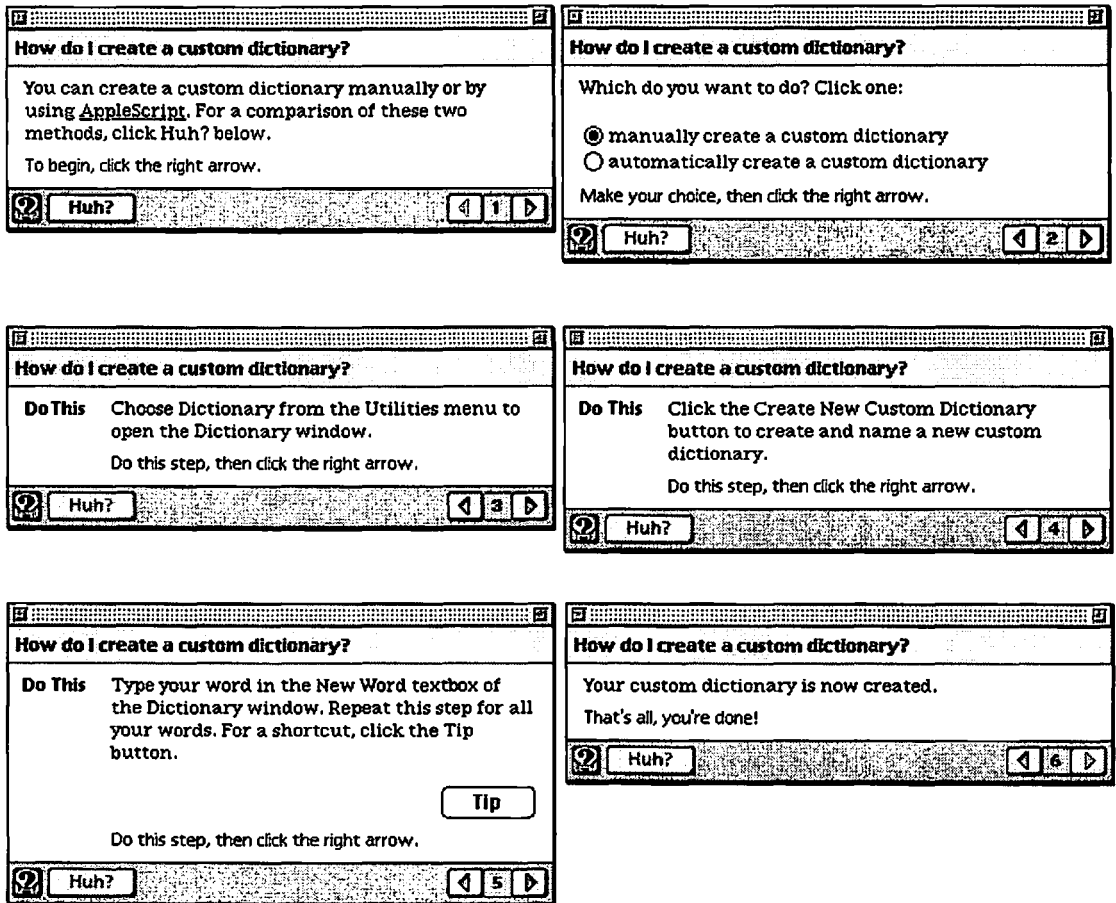
Figure C-5 “How do I create a custom dictionary?” panels (automatic branch)

Figure C-6 “How do I create a custom dictionary?” panels (manual branch)



The panel definitions for this topic, shown in Listing C-16, show use of hot text with a Definition panel and also illustrate how to place radio buttons on a panel. In addition, Listing C-16 shows use of **Huh?**, **Tip**, and **Related Topics** panels.

Listing C-16 Panels for "How do I create a custom dictionary?" (from the "Panel Definitions.src" file)

```

#To use styled text in a panel, you can either directly embed the
# style information in the file (if you have the appropriate XTND
# translator) or you can specify the style information
# using format commands, as shown here.
#introduction panel
<Define Panel> "CreateCustomIntro"
#this panel has an active Huh? button
<Dimmable Button Data> "Huh?", "HuhCompareManualAndAuto"
  <Panel Prompt> "introprompts"
<Format> "PlainText"
You can create a custom dictionary manually or by using
<Format> "UnderlineText"
<Hot Text> "AppleScript", FIRST,
LaunchNewSequenceNewWindow("Defns:AppleScript")
AppleScript.
<Format> "PlainTextReset"
For a comparison of these two
<Format> "PlainTextNormal"
methods, click Huh? below.
<End Panel>

#Alternatively to using format commands for styled text, you can
# define the panels that contain styled text in a separate file
# and save the file in a format for which you have an XTND translator,
# to more easily assign style information to the hot text.
#Here's an example (from a file saved in an appropriate format,
# "Panels with StyleInfo.src")
#<Define Panel> "CreateCustomIntro"
#this panel has an active Huh? button
#<Dimmable Button Data> "Huh?", "HuhCompareManualAndAuto"
  #<Panel Prompt> "introprompts"
#<Format> "Full2"

```

```
#Here's the panel text:
#You can create a custom dictionary manually
#<Hot Text> "AppleScript", FIRST,
LaunchNewSequenceNewWindow("Defns:AppleScript")
#or by using AppleScript. For a comparison of these two
#methods, click Huh? below.
#<End Panel>
```

```
#definition panel for AppleScript
<Define Panel> "HotT AppleScript"
  <Panel Prompt> "Defn&HuhPrompts"
AppleScript is a scripting language that lets you
create sets of written instructions ("scripts")
to automate tasks you perform in an application.
<End Panel>

#definition panel for Huh? panel that compares Auto vs Manual methods
<Define Panel> "CompareManualAndAuto"
  <Panel Prompt> "Defn&HuhPrompts"
You can create a custom dictionary by individually entering each word
yourself or by using a script that searches an open document for
tagged words and places them in the dictionary for you.
<End Panel>
```

```
#decision panel
<Define Panel> "CreateCustomDecision"
  <Panel Prompt> "standard"
Which do you want to do? Click one:
```

```
#for <Radio Button> command, provide title of button,
# default setting, and font
<Radio Button> "manually create a custom dictionary", FALSE, ,,
APPLEGUIDE
<Radio Button> "automatically create a custom dictionary", TRUE, ,,
APPLEGUIDE
<End Panel>
```

```
#first panel for "manually create a custom dictionary" branch
<Define Panel> "CreateCustomManual1"
<Coach Mark> "UtilsOpenDictionary"
  <Panel Prompt> "doThisPrompt"
<Format> "Tag"
Do This
<Format> "Body"
Choose Dictionary from the Utilities menu to open the Dictionary
window.
<End Panel>
```

```
#Continue panel for "CreateCustomManual1" panel
<Define Panel> "SWContinuePanelCustomManual1"
#open the Dictionary window for the user, by sending an Apple event to
# the SurfWriter application requesting it to perform this action
#[for example, <On Panel Show> SWOpenDictionary("Dictionary") ]
#Apple Guide sends this event to SurfWriter when it shows this panel
#
<Panel Prompt> "continuePrompt"
Please wait a moment. Apple Guide is assisting you by opening the
Dictionary window.

<3D Button> 1070, 1072, Center, GoBack()
#
#(For illustrative purposes, this example uses SimpleText instead of
# SurfWriter.)
<On Panel Show> DoAppleScript(":SurfWriter Scripts
src:OpenDictionarySimpleText")
<End Panel>
```

```
#second panel for "manually create a custom dictionary" branch
<Define Panel> "CreateCustomManual2"
  <Panel Prompt> "doThisPrompt"
<Format> "Tag"
Do This
```

SurfWriter Guide and Its Source Files

```
<Format> "Body"
```

Click the Create New Custom Dictionary button to create and name a new custom dictionary.

```
#For SurfWriter, you would use:
```

```
#<Coach Mark> "CustomDictionary"
```

```
<Coach Mark> "CustomDictionaryButtonSimpleText"
```

```
<End Panel>
```

```
#third panel for "manually create a custom dictionary" branch
```

```
<Define Panel> "CreateCustomManual3"
```

```
<Panel Prompt> "doThisPrompt"
```

```
<Format> "Tag"
```

```
Do This
```

```
<Format> "Body"
```

Type your word in the New Word textbox of the Dictionary window.

Repeat this step for all your words. For a shortcut, click the Tip button.

```
<Coach Mark> "DictionaryNewWordSimpleText"
```

```
<Standard Button> "Tip", RIGHT,
```

```
LaunchNewSequenceNewWindow("Tip:CustomDictionary")
```

```
<End Panel>
```

```
#Tip panel for "manually create a custom dictionary" branch
```

```
<Define Panel> "TipForCustomDictionary"
```

```
<Panel Prompt> "Defn&HuhPrompts"
```

To quickly add a word to the dictionary, select the word in your document and then click the Add Word button in the Dictionary window.

```
<End Panel>
```

#first panel for "automatically create a custom dictionary" branch

```
<Define Panel> "CreateCustomAutol"
```

#this panel has an active Huh? button

```
<Dimmable Button Data> "Huh?", "HuhScriptsFolder"
```

```
<Panel Prompt> "doThisPrompt"
```

```
<Format> "Tag"
```

Do This

```
<Format> "Body"
```

Open the SurfWriter Scripts folder. For information about this folder, click Huh? below.

```
<Standard Button> "Related Topics", RIGHT,
```

```
LaunchNewSequenceNewWindow("Related Topics:CustomDictionary")
```

```
<End Panel>
```

#Related Topics panel for "automatically create a custom dictionary"

```
<Define Panel> "RelatedTopicsForCustomDictionary"
```

```
<Panel Prompt> "Defn&HuhPrompts"
```

Also see these topic areas:

Scripting

Writing excellent prose

```
<End Panel>
```

#Huh panel for "automatically create a custom dictionary" branch

```
<Define Panel> "ScriptsFolder"
```

```
<Panel Prompt> "Defn&HuhPrompts"
```

The SurfWriter Scripts folder contains three scripts: Custom Dictionary, Create Glossary, and Create Bibliography.

```
<End Panel>
```

#Continue panel for "CreateCustomAutol" panel

```
<Define Panel> "SWContinuePanelCustomAutol"
```

#Open the SurfWriter Scripts folder for the user,

by providing a script that performs the action for the user.

Use the DoAppleScript event function to run the script.

SurfWriter Guide and Its Source Files

```
<Panel Prompt> "continuePrompt"
```

Please wait a moment. Apple Guide is assisting you by opening the SurfWriter Scripts folder.

```
<3D Button> 1070, 1072, Center, GoBack()
```

```
#
```

```
<On Panel Show> DoAppleScript(":SurfWriter Scripts  
src:OpenSurfWriterScriptsFolder")
```

```
<End Panel>
```

```
#second panel for "automatically create a custom dictionary" branch
```

```
<Define Panel> "CreateCustomAuto2"
```

```
<Panel Prompt> "doThisPrompt"
```

```
<Format> "Tag"
```

```
Do This
```

```
<Format> "Body"
```

```
Double-click the Custom Dictionary script.
```

```
<Coach Mark> "Custom Dictionary File"
```

```
<End Panel>
```

```
#closure panel for "creating a custom dictionary"
```

```
<Define Panel> "CreateCustomAllDone"
```

```
<Panel Prompt> "standard2"
```

```
Your custom dictionary is now created.
```

```
<End Panel>
```

```
# *****panels for Definition panels*****
```

```
<Define Panel> "PanelDefineStandardDictionary"
```

```
<Panel Prompt> "Defn&HuhPrompts"
```

```
Placeholder for information that you supply.
```

```
<End Panel>
```

```
<Define Panel> "PanelDefineCustomDictionary"
```

```
<Panel Prompt> "Defn&HuhPrompts"
```

```
Placeholder for information that you supply.
```

```
<End Panel>
```


Coachmarks

The file "CoachMarks SW.src" specifies the coachmarks for the guide file. Listing C-17 shows the contents of this file. The commands that define a coachmark first specify the name of the coachmark, the signature of the target application, and the coachmark style (red circle or red underline, for example). For menu coaches, the menu title and menu item to coach are also provided. For item coaches, the name of the dialog box and the dialog item to coach are also provided. For window coaches, the name of the window and location to coach are also provided.

Listing C-17 Coachmarks ("CoachMarks SW. src" file)

```
#coach SurfWriter's Dictionary menu item in the Utilities menu
<Define Menu Coach> "UtilsOpenDictionary", 'WAVE', REDCIRCLE,
"Utilities", "Dictionary...", RED

#coach dialog item 3 in SurfWriter's "Dictionary" dialog box
<Define Item Coach> "DictionaryNewWord", 'WAVE', REDUNDERLINE,
"Dictionary", DialogID(3)
#coach dialog item 8 in SurfWriter's "Dictionary" dialog box
<Define Item Coach> "CustomDictionary", 'WAVE', REDCIRCLE,
"Dictionary", DialogID(8)

#coach a specific area in a window opened by SimpleText
<Define Window Coach> "DictionaryNewWordSimpleText", 'ttxt',
REDUNDERLINE, "Dictionary", Rect(74,250,82,395)

#coach a specific area in a window opened by SimpleText
<Define Window Coach> "CustomDictionaryButtonSimpleText", 'ttxt',
REDCIRCLE, "Dictionary", Rect(250,155,262,360)

#coach a specific area in the SurfWriter Scripts folder
<Define AppleScript Coach> "Custom Dictionary File",
REDCIRCLE,":SurfWriter Scripts src:Finder Coach Surf Custom File"
```

Context Checks

SurfWriter Guide uses context checks to dynamically adjust the sequence of display of its panels based on the state of the user's environment. For example, SurfWriter Guide verifies that a certain window is open and active before displaying a panel that coachmarks an item within the window. For panels that contain radio buttons, it also adjusts the display of panels that follow it. These panels are displayed based on the state of the radio buttons.

SurfWriter Guide defines its condition functions (context checks) with the `<Define Context Check>` command. It then specifies that condition function in commands related to conditional execution, such as the `<If>`, `<Skip If>`, `<Make Sure>`, and `<Start Making Sure>` commands.

This guide file uses the `radioButtonState` built-in condition function and also uses the `ActiveWindow`, `OpenWindow`, and `ActiveAppIs` condition functions defined in the Standard Setup file. See Listing C-10 on page C-17 and Listing C-11 on page C-19 for examples of the use of these condition functions. The file "DCC SW.src" on the CD-ROM shows additional examples of condition functions.

Event Functions

SurfWriter Guide uses event functions to perform some action for the user. SurfWriter Guide defines its event functions using the `<Define Event>` command and then specifies these event functions or built-in event functions when using the `<Standard Button>`, `<Hot Text>`, and `<On Panel Show>` commands.

For example, SurfWriter Guide often uses a Tips or Related Topics button. SurfWriter Guide uses the `LaunchNewSequenceNewWindow` built-in event function to display a new sequence to the user when the user clicks the button.

SurfWriter Guide also uses Continue panels. By using the `<On Panel Show>` command to associate an event function (such as the built-in event function `DoAppleScript`) with the display of a Continue panel, SurfWriter Guide performs an action for the user.

See Listing C-16 on page C-31 for examples of the use of these event functions. The file "Event Functions SW.src" on the CD-ROM shows additional examples of event functions.

Index and Look For Content

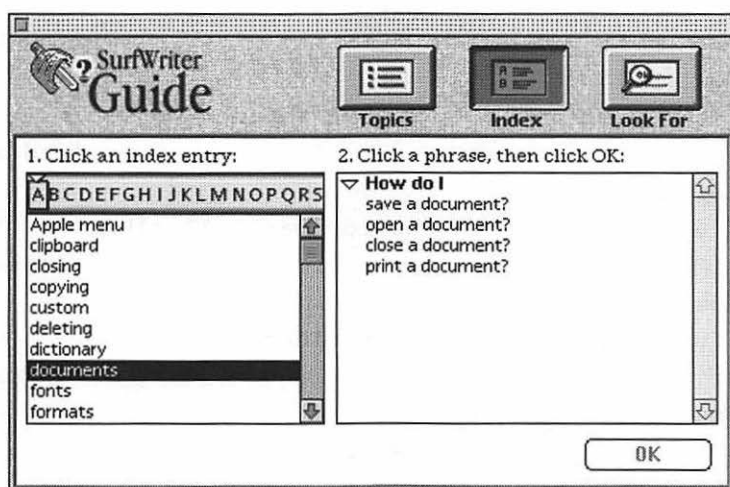
This section focuses on the files that specify the Index and Look For content for SurfWriter Guide. Illustrations of SurfWriter Guide's access window when Index is active and when Look For is active are shown, along with explanatory text describing how SurfWriter Guide implements its Index and Look For content.

SurfWriter Guide provides a list of its index terms. And to enhance Apple Guide's searching facility, in addition to its index terms, SurfWriter Guide provides three lists: the ignore list, exception list, and synonym list.

Index Terms

The file "Index Entries.src" specifies the index terms for SurfWriter Guide. Figure C-7 shows SurfWriter Guide's access window when Index is active.

Figure C-7 The Index window



SurfWriter Guide's index terms, and the headers and topics associated with each index term, are shown in Listing C-18.

Listing C-18 Index terms ("Index Entries.src" file)

```

<Index> "dictionary"                                #index entry
# specify headers and topics for this index term
  <Header> "How do I"                                #header
                                                    #topics
    <Topic> "add a word to the dictionary?","SequenceAddWords"
    <Topic> "look up a word in the dictionary?","SequenceGeneric"
    <Topic> "create a custom dictionary?","SequenCreatCustDictionry"
    <Topic> "add or remove a dictionary?","SequenceGeneric"

  <Header> "Why can't I"
    <Topic> "open the dictionary?","SequenceGeneric"
  <Header> "Definitions"
    <Topic> "custom dictionary","SequenceDefnCustomDictionary"
    <Topic> "standard dictionary","SequenceDefnStdDictionary"

<Index> "toolbar"                                    #index entry
  <Header> "How do I"
    <Topic> "use the tools in the toolbar?","Toolbar"
<Index> "tools"                                       #index entry
  <Header> "How do I"
    <Topic> "use the tools in the toolbar?","Toolbar"
<Index> "pencil"                                      #index entry
  <Header> "How do I"
    <Topic> "use the tools in the toolbar?","Toolbar"
<Index> "hammer"                                     #index entry
  <Header> "How do I"
    <Topic> "use the tools in the toolbar?","Toolbar"

```

```

<Index> "opening"                                #index entry
  <Header> "How do I"
    <Topic> "open a document?","SequenceGeneric"
  <Header> "Why can't I"
    <Topic> "open the dictionary?","SequenceGeneric"
<Index> "custom"                                #index entry
  <Header> "How do I"
    <Topic> "create a custom dictionary?","SequenCreatCustDictionary"
  <Header> "Definitions"
    <Topic> "custom dictionary","SequenceDefnCustomDictionary"
<Index> "standard"                              #index entry
  <Header> "Definitions"
    <Topic> "standard dictionary","SequenceDefnStdDictionary"
<Index> "Utilities menu"                        #index entry
  <Header> "How do I"
    <Topic> "use the dictionary?","SequenceGeneric"
    <Topic> "use the thesaurus?","SequenceGeneric"
#specify invisible index terms.
#invisible index terms do not appear in the index, but are included
# when the user searches using the Look For feature
<Index> "customizing", FALSE                    #index entry (invisible)
  <Header> "How do I"
    <Topic> "add a word to the dictionary?","SequenceAddWords"
    <Topic> "add or remove a dictionary?","SequenceGeneric"
    <Topic> "create a custom dictionary?","SequenCreatCustDictionary"
  <Header> "Definitions"
    <Topic> "custom dictionary","SequenceDefnCustomDictionary"
<Index> "spell", FALSE                          #index entry (invisible)
  <Header> "How do I"
    <Topic> "look up a word in the dictionary?","SequenceGeneric"
<Index> "bitmap", FALSE                        #index entry (invisible)
  <Header> "How do I"
    <Topic> "use bitmapped fonts?","SequenceGeneric"
    <Topic> "create bitmapped graphics?" ,"SequenceGeneric"
    <Topic> "placeholder for topic?","SequenceGeneric"

```

APPENDIX C

SurfWriter Guide and Its Source Files

```
<Index> "definitions", FALSE #index entry (invisible)
  <Header> "Definitions"
    <Topic> "custom dictionary","SequenceDefnCustomDictionary"
    <Topic> "standard dictionary","SequenceDefnStdDictionary"

#other index terms (that use placeholders for now)
<Index> "Apple menu" #index entry
  <Header> "How do I"
    <Topic> "placeholder for topic?","SequenceGeneric"
<Index> "clipboard" #index entry
  <Header> "How do I"
    <Topic> "placeholder for topic?","SequenceGeneric"
<Index> "closing" #index entry
  <Header> "How do I"
    <Topic> "close a document?","SequenceGeneric"
<Index> "copying" #index entry
  <Header> "How do I"
    <Topic> "placeholder for topic?","SequenceGeneric"
<Index> "deleting" #index entry
  <Header> "How do I"
    <Topic> "placeholder for topic?","SequenceGeneric"
<Index> "documents" #index entry
  <Header> "How do I"
    <Topic> "save a document?","SequenceGeneric"
    <Topic> "open a document?","SequenceGeneric"
    <Topic> "close a document?","SequenceGeneric"
    <Topic> "print a document?","SequenceGeneric"
<Index> "fonts" #index entry
  <Header> "How do I"
    <Topic> "use bitmapped fonts?","SequenceGeneric"
    <Topic> "placeholder for topic?","SequenceGeneric"
<Index> "formats" #index entry
  <Header> "How do I"
    <Topic> "placeholder for topic?","SequenceGeneric"
```

```

<Index> "graphics"                                #index entry
  <Header> "How do I"
    <Topic> "create bitmapped graphics?","SequenceGeneric"
    <Topic> "placeholder for topic?","SequenceGeneric"
<Index> "pasting"                                  #index entry
  <Header> "How do I"
    <Topic> "placeholder for topic?","SequenceGeneric"
<Index> "preferences"                              #index entry
  <Header> "How do I"
    <Topic> "placeholder for topic?","SequenceGeneric"
<Index> "printing"                                 #index entry
  <Header> "How do I"
    <Topic> "placeholder for topic?","SequenceGeneric"
    <Topic> "print a document?","SequenceGeneric"
<Index> "replacing"                                #index entry
  <Header> "How do I"
    <Topic> "placeholder for topic?","SequenceGeneric"
<Index> "saving"                                   #index entry
  <Header> "How do I"
    <Topic> "save a document?","SequenceGeneric"
<Index> "searching"                                #index entry
  <Header> "How do I"
    <Topic> "placeholder for topic?","SequenceGeneric"
<Index> "selecting"                                #index entry
  <Header> "How do I"
    <Topic> "placeholder for topic?","SequenceGeneric"
<Index> "styles"                                   #index entry
  <Header> "How do I"
    <Topic> "placeholder for topic?","SequenceGeneric"
<Index> "writing"                                  #index entry
  <Header> "How do I"
    <Topic> "placeholder for topic?","SequenceGeneric"

```

Note that several invisible index terms are defined in the "Index Entries.src" file. Apple Guide does not display invisible index terms in the Index window.

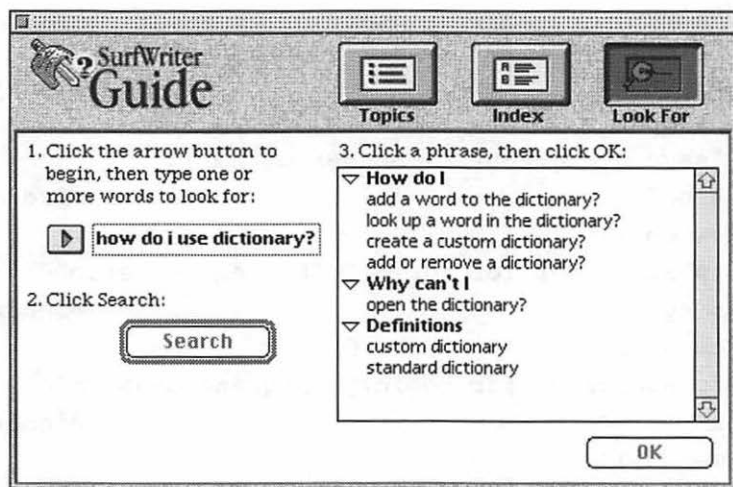
SurfWriter Guide and Its Source Files

However, Apple Guide includes both visible and invisible index terms when it searches the index as a result of the user entering a search phrase in the Look For window.

The Ignore List

The file "Ignore List.src" specifies the words in SurfWriter Guide's ignore list. When first parsing a search phrase entered by the user, Apple Guide removes from the phrase any words that appear on the ignore list. For example, if the user enters the phrase "how do i use dictionary?" Apple Guide removes "how", "do", "I", and "use" from the phrase, resulting in the phrase "dictionary". "Dictionary" is an index term and, after checking the exception and synonym list, Apple Guide reports a match, as shown in Figure C-8.

Figure C-8 Matching a search phrase



Apple Guide automatically removes from the user's search phrase numerals and other special characters, such as "\$", "&", "+", "%", "?", and "-". For example, Apple Guide reduces the phrase "32-bit addressing" to "bit address". Listing C-19 shows words on SurfWriter Guide's ignore list.

Listing C-19 Ignore words ("Ignore List.src" file)

```
<ignore> "how"  
<ignore> "do"  
<ignore> "I"  
<ignore> "my"  
<ignore> "the"  
<ignore> "and"  
<ignore> "or"  
<ignore> "real"  
<ignore> "really"  
<ignore> "a"  
<ignore> "also"  
<ignore> "any"  
<ignore> "in"  
<ignore> "into"  
<ignore> "on"  
<ignore> "an"  
<ignore> "as"  
<ignore> "by"  
<ignore> "for"  
<ignore> "from"  
<ignore> "of"  
<ignore> "if"  
<ignore> "is"  
<ignore> "isn't"  
<ignore> "to"  
<ignore> "it"  
<ignore> "its"  
<ignore> "it's"  
<ignore> "look"  
<ignore> "up"  
<ignore> "this"  
<ignore> "that"  
<ignore> "why"
```

```
<ignore> "when"  
<ignore> "these"  
<ignore> "can"  
<ignore> "can't"  
<ignore> "cannot"  
<ignore> "won't"  
<ignore> "create"  
<ignore> "use"  
<ignore> "add"  
<ignore> "remove"  
<ignore> "lookup"  
<ignore> "word"
```

The Exception List

The file "Exception List.src" specifies the words in SurfWriter Guide's exception list. Listing C-20 shows the contents of this file. SurfWriter Guide places a word on its exception list *only* if the stemmed form of the word matches an existing index term. For example, SurfWriter Guide defines "custom" and "customizing" as two separate index terms. To prevent Apple Guide from stemming "customizing" to its root form ("custom"), "customizing" is included on the exception list. Thus, if the user enters "customizing" as a search phrase Apple Guide reports the correct matching index term. "Customize" is also placed on the exception list, to prevent stemming to "custom". In addition, "customize" is placed on the synonym list, associating it with the index term "customizing".

Listing C-20 Words on the exception list ("Exception List.src" file)

```
#you usually place an index term on the exception list  
# ONLY when the term would otherwise stem to another  
# index term or whose root form has an alternate meaning
```

```
<exception> "customize"  
<exception> "customizing"  
<exception> "preferences"
```

The Synonym List

The file "Synonym List.src" specifies the words in SurfWriter Guide's synonym list. As you recall, when parsing a phrase, after removing words on the ignore list and stemming words except those on the exception list, Apple Guide then checks the synonym list. Apple Guide first checks whether the synonym list specifies the entire phrase as a synonym. If Apple Guide finds the phrase on the synonym list, it replaces the synonym with its associated phrase (usually its index term) then checks the index for a term matching this phrase. If it finds a matching index term, Apple Guide reports a match.

For example, if the user enters "How do I create a site dictionary?" Apple Guide removes words on the ignore list and stems words not on the exception list. This results in the phrase "site dictionary". Apple Guide then checks the synonym list and replaces the synonym "site dictionary" with its associated index term, "dictionary". Apple Guide then searches the index for this phrase and finds a matching index term. Also note that one index term can have many synonyms; for example, SurfWriter Guide defines "palette" and "tool bar" as synonyms for "toolbar".

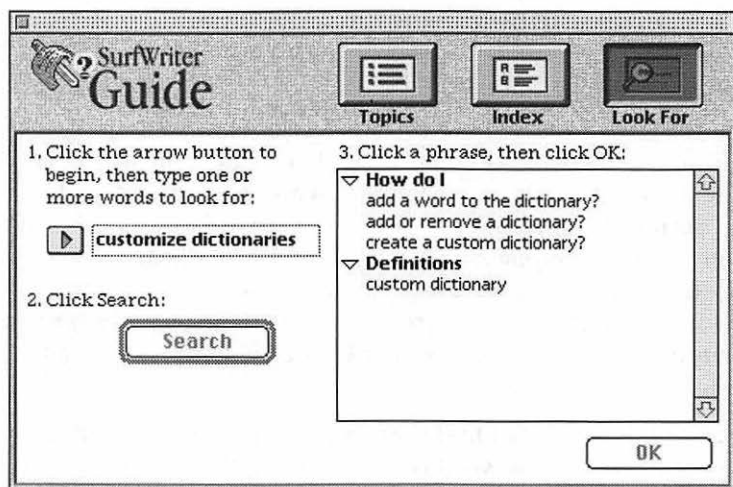
If Apple Guide does not find the phrase in the synonym list, then for each individual word, if the word is a synonym in the synonym list, it replaces the synonym with its index term. It then looks in the index for a term matching the parsed phrase (this time any words that are synonyms have been replaced by their equivalent index terms). If Apple Guide finds a match for the phrase, it displays the topics for the index term. If it doesn't find the phrase in the index, it looks in the list of index terms for each word. If Apple Guide finds a matching index term for more than one word in the phrase, Apple Guide intersects the results and displays any topics that are common to both words.

For example, if the user enters the search phrase "customize dictionaries", Apple Guide stems "dictionaries" to "dictionary" (it does not stem "customize" because this word is on the exception list). Apple Guide then looks in the synonym list for the phrase "customize dictionary". Because "customize dictionary" is not on the synonym list, Apple Guide then checks the synonym list again, this time for each word in the phrase. "Customize" is on the synonym list, so Apple Guide replaces it with the index term

SurfWriter Guide and Its Source Files

"customizing". "Dictionary" is not on the synonym list so Apple Guide leaves "dictionary" as is. Next, Apple Guide looks in the index for the phrase "customizing dictionary". Because this phrase isn't in the index, Apple Guide then looks in the index for the term "dictionary" and for the term "customizing", and finally displays the intersection of this list. Figure C-9 shows the results of this search. Note that Apple Guide displays only those topics that are common to both index terms.

Figure C-9 Results of an intersection between two index terms



Listing C-21 shows the words on the SurfWriter Guide's synonym list.

Listing C-21 Words on the synonym list ("Synonym List.src" file)

```
#specify the index term, then specify the synonym
<syn> "clipboard", "clip"
<syn> "clipboard", "show clipboard"
<syn> "clipboard", "hide clipboard"
<syn> "clipboard", "scrap"
```

SurfWriter Guide and Its Source Files

```

<syn> "closing", "close"
<syn> "closing", "closebox"
<syn> "closing", "close box"
<syn> "closing", "close window"
<syn> "copying", "clone"
<syn> "copying", "copy"
<syn> "copying", "copy command"
<syn> "copying", "duplic"
<syn> "customizing", "customize"
<syn> "definitions", "definit"
<syn> "deleting", "clear"
<syn> "deleting", "clear command"
<syn> "deleting", "cut"
<syn> "deleting", "cut command"
<syn> "deleting", "delet"
<syn> "deleting", "delete"
<syn> "dictionary", "site dictionary"
<syn> "documents", "docum"
<syn> "documents", "file"
<syn> "fonts", "font"
<syn> "fonts", "bitmap font"
<syn> "fonts", "outline font"
<syn> "formats", "format"
<syn> "graphics", "graphic"
<syn> "graphics", "bitmap graphic"
<syn> "hammer", "hammer icon"
<syn> "opening", "open"
<syn> "opening", "open command"
<syn> "opening", "double click"
<syn> "pasting", "past"
<syn> "pasting", "paste"
<syn> "pencil", "pencil icon"
<syn> "printing", "print"
<syn> "printing", "print command"
<syn> "printing", "printer"

```

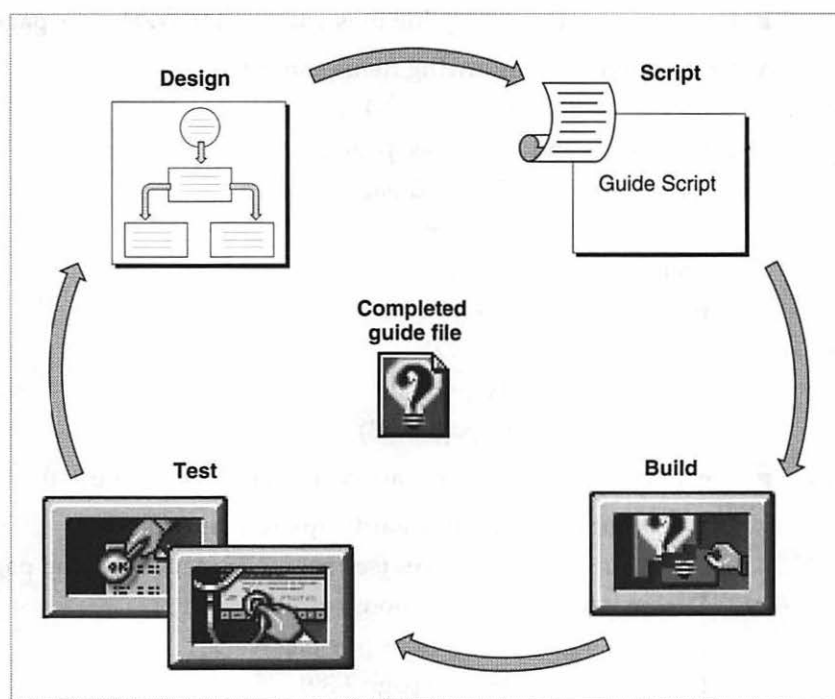
SurfWriter Guide and Its Source Files

```
<syn> "printing", "LaserWrit"  
<syn> "printing", "ImageWrit"  
<syn> "printing", "spool"  
<syn> "replacing", "replac"  
<syn> "replacing", "replace"  
<syn> "saving", "save"  
<syn> "saving", "save command"  
<syn> "saving", "save work"  
<syn> "saving", "store"  
<syn> "searching", "searche"  
<syn> "searching", "search"  
<syn> "selecting", "select"  
<syn> "styles", "style"  
<syn> "styles", "bold"  
<syn> "styles", "ital"  
<syn> "styles", "underline"  
<syn> "styles", "shadow"  
<syn> "tools", "tool"  
<syn> "toolbar", "palette"  
<syn> "toolbar", "tool bar"  
<syn> "utilities menu", "util"  
<syn> "utilities menu", "util menu"  
<syn> "writing", "write"
```

Checklist

This appendix is a checklist you can use to design, script, build, and test guide files; modify and localize them; and integrate them into your application. It consists of a series of actions for accomplishing the given tasks and includes page numbers referring you to information within the book. The first four sections show how to perform the tasks required to create a guide file—designing, scripting, building, and testing—as shown in Figure D-1.

Figure D-1 Tasks required to create a guide file



Checklist

The last section shows three tasks you can perform on your guide files after you complete them. These are

- localizing your guide file
- modifying your guide file using a mixin
- integrating your guide file into your application

Designing Your Guide File Content

You should perform these tasks before you begin scripting your guide file:

- **Determine which information is appropriate for your guide files (see page 3-4)**
- **Decide which types of guide files you want to create (see page 2-5)**
- **Find out how the following items should look:**
 - access windows (see page 2-15)
 - topic areas and topics (see page 2-30)
 - panels (see page 2-35 and page 2-50)
 - sequences (see page 2-66)
 - branches (see page 2-67)
 - buttons (see page 2-70)
 - hot areas (see page 2-77)
 - coachmarks (see page 2-79)
 - context checks (see page 2-83)
- **Create your guide file topic areas and topics (see page 3-4)**
- **Design your guide file in a hard-copy format (page 3-7)**
 - Plan panels and sequences (see page 2-35, page 2-50, and page 2-66)
 - Design navigation bar buttons (see page 2-71)
 - Design content area buttons (see page 2-72)
 - Design prompt sets (see page 2-39)
 - Decide whether to use Oops or Continue panels for context checks (see page 2-85)

Checklist

- ☐ Design your context checks and associated panels (see page 2-83)
- **Design your access window features**
 - ☐ Design your application logo (see page 2-18)
 - ☐ Compose your howdy text, if used (see page 2-28)
 - ☐ Create a preliminary guide file index for the Index and Look For features in the Full Access window (see page 3-19)
 - ☐ Create preliminary ignore, exception, and synonym lists for the Look For feature in the Full Access window (see page 3-23 and page 3-24))
- **Design your panel features**
 - ☐ Plan the panel layout and format (see page 2-43 and page 2-45)
 - ☐ Compose your help instructions (see page 2-38)
 - ☐ Prepare your graphics (see page 2-46)
 - ☐ Determine all navigation and content buttons (see page 2-71 and page 2-72)
 - ☐ Assign the appropriate prompt (see page 2-39)
 - ☐ Plan for localization (see page 2-91)

Scripting Your Source Files

After designing your guide file, script your source files as follows:

- **Set up the organization of your source files**
 - ☐ Split your content into multiple source files (see page 5-3)
 - ☐ Include the Standard Setup file (see page 5-4)
 - ☐ Import resources from the Standard Resources file (see page 5-4)
 - ☐ Create your build file (see page 5-5)
- **Customize the information in the Standard Setup file**
 - ☐ Define your prompt sets (see page 10-35 and page 10-37)
 - ☐ Define your formats (see page 10-30 and page 10-85)
 - ☐ Define your navigation button sets (see page 10-32, page 10-71, and page 10-80)

Checklist

- **Provide basic information about your guide file**
 - ☐ Import your application logo (see page 10-25)
 - ☐ Specify howdy text, if any (see page 10-24)
 - ☐ Specify the type of access window (see page 10-21)
 - ☐ Specify the menu item text that should appear in the Help menu when your guide is available (see page 10-14)
 - ☐ Specify help balloon text for your guide's item in the Help menu (see page 10-16)
 - ☐ Specify the application associated with your guide file (see page 10-8)
 - ☐ Specify WorldScript information (see page 10-13)
 - ☐ Specify version information for your guide file (see page 10-11)
 - ☐ Specify whether your guide is a mixin (see page 10-19)
 - ☐ Specify whether other guides can mix into your guide file or whether it can be mixed in with other guides (see page 10-20)
- **Specify your topic areas and topics (see page 10-125, page 10-135, and page 10-137)**
- **For mixins, add or modify topic areas, index terms, and topics (see page 10-190)**
- **Define your sequences (see page 10-39)**
 - ☐ Specify the sequence prompt set (see page 10-42)
 - ☐ Specify the sequence navigation button set (see page 10-43)
 - ☐ Specify panels and conditional display of panels (see page 10-45, page 10-52, and page 10-152)
 - ☐ Provide condition functions for context checks (see page 10-172)
- **Provide the content of your panels**
 - ☐ Text (see page 10-52 and page 10-85)
 - ☐ Graphics (see page 10-95)
 - ☐ Movies (see page 10-98)
 - ☐ Standard buttons (see page 10-57)
 - ☐ Three-dimensional buttons (see page 10-60)
 - ☐ Radio buttons (see page 10-64)
 - ☐ Checkboxes (see page 10-69)
 - ☐ Hot text (see page 10-122)

Checklist

- ☐ Event functions (page 10-178)
- ☐ Coachmarks (see page 10-118)
- ☐ Prompts (see page 10-55)
- **Define your index terms**
 - ☐ Index terms (see page 10-128)
 - ☐ Associated help topics (see page 10-135)
 - ☐ Topics associated with headings (see page 10-137)
- **Define your Look For content**
 - ☐ Ignore list (see page 10-145)
 - ☐ Exception list (see page 10-147)
 - ☐ Synonym list (see page 10-149)

Building Your Guide File

Compile your source files into a guide file by performing these tasks:

- **Open Guide Maker** (see page 4-5)
- **Compile your source files** (see page 5-6)
- **Save your guide file** (see page 5-8)

Testing Your Guide File

After building your guide file, test it as follows:

- **Test your Look For content** (see page 6-8)
- **Test your guide file's interface** (see page 6-3)
- **Verify coachmarks, context checks, and event functions** (see page 6-18)
- **Do user testing** (see page 6-20)

Additional Guide File Tasks

After you complete your guide file, at some point you will want to modify its contents; you do this using a mixin. Apple recommends that you also localize your guide file for other regions. In addition to making your guide file available from your application's Help menu, you can integrate help into your application. These three guide file tasks are listed next.

■ Localize your guide file

- ☐ Extract language-specific text strings from your source files using Guide Maker's Localize utility (see page 7-3)
- ☐ Translate the text strings using various tools (see page 7-6)
- ☐ Merge the translated text strings back into your source files using Guide Maker's Localize utility (see page 7-6)
- ☐ Localize any other elements of your guide file, such as pictures, as necessary (see page 7-6)
- ☐ Compile your translated source files into a new guide file (see page 5-6)

■ Modify a guide file by creating a mixin source file (see page 2-14)

- ☐ Specify that the guide file is a mixin (see page 10-19)
- ☐ Indicate whether it can be mixed in with any guide or only a specific guide (see page 10-20)
- ☐ Specify additions and modifications to the main guide file's content (see page 10-190)
- ☐ Compile your mixin source files into a Mixin guide file (see page 5-6)
- ☐ Save your Mixin guide file (see page 5-8)

■ Integrate your guide file with your application

Refer to Chapter 9 for a description of the complete set of functions that allow you to provide the user with context-sensitive help from within your application.

Glossary

About Apple Guide A guide file included with system software that describes the help system provided with the Macintosh computer.

About guide file A type of guide file that describes the available help systems in an application. See also **guide file**, **guide file types**, and **Macintosh Guide**.

access window The window—Full, Single List, or Simple—that appears whenever the user selects a guide file from the Help Menu and from where the user selects to view (or goes directly to view) help topics. See also **Full Access window**, **Single List Access window**, and **Simple Access window**.

action panel A panel that presents a single step in a procedure (for example, to tell the user to open a menu).

additions See **Mixin guide file**.

Apple Guide A system extension provided with system software that provides an onscreen help system.

Apple Guide API A set of functions, or application programming interface, that lets you start up, access, and work with Apple Guide from within your application.

Apple Guide font The attributes, 10-point Espy Serif Plain black, automatically applied by Apple Guide to text on the panel content area.

AppleScript coach A coachmark for an object whose location is determined by a script. See also **coachmark**, **item coach**, **menu coach**, **object coach**, **window coach**.

Balloon Help A System 7 (and higher) feature that provides users information about such objects on the Macintosh screen as icons, windows, and commands.

Body format A format that provides a right column and that is designed for use with the Tag format.

branch A panel sequence within another sequence.

build To compile a help source file using Guide Maker.

build file A file that contains only <Include> and <Resource> commands.

Build utility The Guide Maker utility you use to build guide files. See **build** and **guide file**.

closure panel At the end of a sequence, a panel that summarizes the information covered by that sequence.

coachmark An onscreen graphic that circles or points to an item on the screen. See also **coachmark types**.

coachmark types The five different kinds of coaches—menu, item, object, window, and AppleScript—that you can create with Apple Guide. See also **menu coach**, **item coach**, **object coach**, **window coach**, and **AppleScript coach**.

compile See **build**.

content area The part of a panel between the panel sequence title area and the navigation bar that contains help instructions, possibly including text, control features (such as radio buttons and checkboxes), 'PICT' graphics, and QuickTime movies. See also **sequence display title area** and **navigation bar**.

content area button Control features that appear in the content area of a panel and that are generally associated with an event or navigation route specific to that panel. See also **content area**.

context checks Functions that verify the user's environment so that Apple Guide can dynamically skip or show certain panels to the user based on their appropriateness.

Continue panel A panel that offers to complete for the user a condition that has not been met when the user attempts to continue to the next panel in the sequence. Compare **Oops panel**.

Convert utility The Guide Maker utility you use to convert a Windows Help file to a Guide Script source file.

decision panel A panel from which the user can select one or more tasks by using radio buttons or checkboxes. Each button or checkbox has a distinct panel branch associated with it.

default prompt set The prompt set that Apple Guide automatically applies to panels in a sequence unless you override it. See also **prompt** and **prompt set**.

definition panel A panel that defines terminology appearing on a panel.

"Definitions" heading A heading for term definitions that relate directly to the selected topic area. See also **"How do I" heading** and **"Why can't I" heading**.

Diagnose utility The Guide Maker utility you use to step through the panels of a guide file.

exception list A list of words that Apple Guide does not reduce to a root word if entered as a user search phrase with Look For features selected in the Full Access window. See also **ignore list**, **synonym list**, and **Look For buttons**.

first-level panels In a sequence, the panels that lead the user directly through the task or concept of the associated topic. See also **second-level panels**.

first previous panel In a sequence, the panel that Apple Guide displays when the user clicks the OK button on an Oops panel without making the condition true. This panel is the first panel that Apple Guide finds, searching backward through the sequence, that meets one of two criteria: it does not have a <Make Sure> command specified for it, or it has a <Make Sure> command whose condition evaluates to true. See also **Oops panel**.

Full Access window A window that includes three built-in buttons—Topics, Index, and Look For—from which the user makes selections or enters a search phrase. See also **Simple Access window** and **Single List Access window**.

Full format A one-column format that Apple Guide applies by default to all panels. See also **Tag and Body format**.

GoStart button A lightbulb-shaped button that appears in the navigation bar of a panel and that takes the user back to the access screen. See also **navigation button**.

guide file A single file containing help content that conforms to one of five guide file types supported by Apple Guide. See also **guide file types**.

guide file index A list of terms that point to the topics appearing in the right column of the Full Access window. Apple Guide uses the index to retrieve topics for the user when Look For or Index features are selected on the Full Access screen. See also **Index button** and **Look For button**.

guide file types Five guide files specified by the Apple Guide Human Interface Guidelines—About, Tutorial, Help, Shortcuts, and Other—that each have a particular focus, content, naming convention, and Help menu location. See also **About guide file**, **Help guide file**, **Shortcuts guide file**, **Tutorial guide file**, and **Other guide file**.

Guide Maker A tool for building and testing guide files.

Guide Script An authoring language for developing guide files.

Guide Script source files See **help source files**.

Help guide file A type of guide file that provides the main information in a help system through a wide range of task-oriented information about an application. See also **guide file**, **guide file types**, and **Macintosh Guide**.

help source files Files containing Guide Script commands that define the look, content, and navigation path of all panels in a guide file. See also **Guide Script**.

hot area On a panel, the area containing a rectangle, object, or text that the user can click to view another panel containing related information.

hot object On a panel, a hot area specified by the rectangle of the next object (either text or a graphic) in the panel definition.

hot rectangle On a panel, a hot area that is a specific rectangle.

hot text On a panel, a hot area that consists of designated text. See also **hot area**, **hot object**, and **hot rectangle**.

"How do I" heading In the Full Access window, a heading for topics that show the user how to accomplish a task. See also **"Definitions" heading** and **"Why can't I" heading**.

howdy text Text that describes a guide file and that appears in the access window's instructions.

Huh? button A button that appears in the navigation bar of a panel and that, when active, the user can click to view an associated panel with crucial information.

ignore list A list of words you tell Apple Guide to remove from a user search phrase entered with Look For features selected in the Full Access window. See also **Look For button**, **exception list**, and **synonym list**.

Index button On the Full Access screen, a built-in button that the user can click to display a list of index terms for the guide file contents. This list appears in the left column of the window.

information panel A panel that provides brief conceptual explanations.

introductory panel A panel that begins a panel sequence or branch and that describes its contents.

invisible index term A term that does not correspond to a visible term in the Look For index but that Apple Guide uses to match a search phrase with guide file topics. See also **visible index term**.

item coach A coachmark for an item in a dialog box or other interface element in a window (or dialog box). See also **coachmark**, **menu coach**, **object coach**, **window coach**, and **AppleScript coach**.

Localize utility The Guide Maker utility you use to localize all elements of your guide files that are language-specific, such as text strings and pictures.

Look For button In the Full Access window, a built-in button that the user can click to access information in the guide file using a search phrase.

Macintosh Guide A guide file included with system software that provides its main help through step-by-step instructions for a variety of tasks.

Macintosh Guide additions Four guide files—PowerBook Guide, Speech Guide, Video Guide, Video Player—that add content to Macintosh Guide about a specific piece of hardware attached to the Macintosh computer or about certain system software features.

main guide file A guide file containing help instructions that can be modified using a Mixin guide file. See also **Mixin guide file**.

menu coach A coachmark for a specific menu or menu item. See also **AppleScript coach**, **coachmark**, **item coach**, **object coach**, and **window coach**.

Mixin guide file A file you use to add to or modify the contents of a main guide file. See also **main guide file**.

navigation bar On the lower portion of a panel, the bar that displays the left and right navigation arrows that the user clicks to move between panels. This bar may contain additional navigation buttons (such as GoStart and Huh?). See also **content area** and **sequence display title area**.

navigation button A button that always appears in the navigation bar of a panel that takes the user to different parts of the guide file. See also **GoStart button** and **Huh? button**.

object coach A coachmark for an object based on a rectangle that an application returns for the named object. See also **coachmark**, **menu coach**, **item coach**, **window coach**, and **AppleScript coach**.

Oops panel A panel that tells the user a condition specified on a previous panel was not met. The user needs to make the condition true to continue to the next panel. Compare **Continue panel**.

Other guide file A guide file that does not conform to the content guidelines for the four other Apple Guide guide files (About, Tutorial, Help, and Shortcuts) or that is a particularly advanced or specialized version of these other types. See also **guide file types**.

panel An Apple Guide help window that describes a concept or step.

panel associated with a Huh? button A panel that appears when the user clicks a Huh? button on a panel. This associated panel provides information crucial to understanding the original panel.

panel branch See **branch**.

panel height On a panel, the distance between the title area and the navigation bar.

panel number The number appearing on a panel between the left and right navigation arrows. Guide Maker automatically assigns this number in a dynamic sequence.

panel sequence See **sequence**.

panel width The area across the panel, which is a fixed measurement of 344 pixels.

presentation panel See **panel**.

prompt The panel text that tells the user what to do and where to go (for example, click the right arrow to continue).

prompt set A collection of four prompts that Apple Guide chooses to display based on what a panel's position in the sequence is (first, middle, or last) or whether it contains controls (radio buttons, checkboxes, or standard buttons). See also **default prompt set** and **prompt**.

random access A method of access in which the user can select help topics in any order.

related topics panel A panel that describes topics in a guide file pertaining to either a specific panel or sequence.

ResetPen Format A format that resets the current format to the default format.

second-level panels The panels that provide supplemental help, and Oops and Continue panels. See also **first-level panels**.

sequence A set of related panels that the user can access linearly using left and right navigation arrows.

sequence display title area In the upper portion of a panel, the bar that contains the panel title. See also **content area** and **navigation bar**.

sequential access A method of access in which the user can select help topics in a structured order.

Shortcuts A guide file included with system software that provides Macintosh keyboard commands and tips.

Shortcuts guide file A guide file that provides condensed reference material similar to that found on a quick reference card. See also **guide file types** and **guide file**.

Simple Access window A window that takes the user directly to the help information unless you provide your own access method. See also **Full Access window** and **Single List Access window**.

Single List Access window A window that provides a single scrollable list of topics for the user to choose from. See also **Full Access window** and **Simple Access window**.

standard button A two-dimensional button, drawn by the Macintosh toolbox, that has an event associated with it. See also **content area button**.

standard panel types A set of panel designs that apply to and should be used for specific categories of help information.

stemming The process by which Apple Guide reduces common word variations to root words in search phrases entered by the user.

synonym list A list containing words that have identical meaning to index terms but that do not appear in the index. See also **exception list**, **ignore list**, and **Look For button**.

Tag In the left column of a panel, a bold phrase that describes the Body text that appears in the right column of the panel. See also **Tag and Body format**.

Tag and Body format A two-column format for panels created by two Guide Script commands. See also **Full format**, **Tag**, **Tag format**, and **Body format**.

Tag format A format that provides a left column that you can use to format tags and that is designed for use with the Body format.

three-dimensional button A button whose appearance is determined by a graphic that you place in it. See also **content area button**, **navigation button**, and **standard button**.

tip panel A panel that gives a hint about how to perform an action or use an application feature.

topic A category of help information in a guide file that the user views from an access window.

topic area A broad category of help that breaks into one or more topics. It appears in the left column of the Full Access window when the user selects the Topics button.

GLOSSARY

Topics button A built-in button on the Full Access window that the user clicks to display topic areas in the left column of the window.

transition panel A panel that connects parts of multipart panel sequences.

Tutorial A guide file included with system software that provides training in Macintosh skills.

Tutorial guide file A type of guide file that leads users through the basic features of an application. See also **guide file types** and **guide files**.

visible index term A term that appears in the left column of the Full Access window with Index features selected. See also **invisible index terms**.

“Why can’t I” heading In the Full Access window, a heading for topics that explain why a certain action cannot be performed. See also **“Definitions” heading** and **“How do I” heading**.

window coach A coachmark for a specific area of a window. See also **coachmark**, **menu coach**, **item coach**, **object coach**, and **AppleScript coach**.

Index

Symbols

command 10-17

Numerals

<3D Button> command 10-60 to 10-64

A

About Apple Guide 1-5

About guide files 2-7, 2-9 to 2-10

access windows 2-15 to 2-30

 features of 1-9 to 1-13

 Full 2-16 to 2-24

 creating Look For features for 3-12 to 3-26

 defined 2-15

 features of 2-16 to 2-19

 headings for 2-33 to 2-35

 for Help guide files 2-11

 howdy text on 2-28 to 2-30

 designing Index features for 2-20 to 2-21

 designing Look For features for 2-21 to 2-24

 topic areas for 2-31 to 2-32

 designing Topics features for 2-19 to 2-20

 topics for 2-31 to 2-32

Simple

 buttons in 2-26

 defined 2-15

 features of 2-26 to 2-28

 howdy text on 2-28 to 2-30

 for Shortcuts guide files 2-14

 topics for 2-33

Single List

 defined 2-15

 features of 2-25 to 2-26

 howdy text on 2-28 to 2-30

 topics for 2-32 to 2-33

action panels 2-53 to 2-54

additions. *See* Mixin guide files

AGClose function 9-17 to 9-18

AGFileGetDBCCount function 9-9

AGFileGetDBCcountry function 9-28 to 9-29

AGFileGetDBMenuName function 9-23 to 9-24

AGFileGetDBType function 9-26 to 9-27

AGFileGetDBVersion function 9-29 to 9-30

AGFileGetHelpBalloonText function 9-24 to 9-25

AGFileGetHelpMenuAppCreator
 function 9-25 to 9-26

AGFileGetIndDB function 9-10 to 9-11

AGFileGetMixinMatchSelector
 function 9-32 to 9-33

AGFileGetSelectorCount function 9-30

AGFileGetSelector function 9-31

AGFileIsMixin function 9-28 to 9-32

AGFile library 9-3

AGGeneral function 9-20 to 9-21

AGGetAvailableDBTypes function 9-7 to 9-8

AGGetFrontWindowKind function 9-19 to 9-20

AGGetFSSpec function 9-22

AGGetStatus function 9-6 to 9-7

AGInstallCoachHandler function 9-34 to 9-35

AGInstallContextHandler function 9-36 to 9-37

AGIsDatabaseOpen function 9-17 to 9-19

AGOpen function 9-12 to 9-13

AGOpenWithSearch function 9-14 to 9-15

AGOpenWithSequence function 9-16 to 9-17

AGOpenWithView function 9-13 to 9-14

AGQuit function 9-6

AGRemoveCoachHandler function 9-35

INDEX

`AGRemoveContextHandler` function 9-37 to 9-38
`AGStart` function 9-5
<Allow Prompts> command 10-34 to 10-35
<App Creator> command 10-8 to 10-9
Apple Guide
 compared with Balloon Help 1-5
 components of 1-4 to 1-5
 features of 1-3 to 1-8
 typical help session with 1-9 to 1-13
 uses of 1-8
Apple Guide API
 functions in 9-5 to 9-38
Apple Guide Debug extension 6-6, 9-4
Apple Guide extension 9-4
 determining if installed 9-4
Apple Guide font 2-48
Apple Guide icon 2-19
AppleScript
 and coachmarks 2-80
 determining location of coachmarks with 10-116 to 10-117
 localizing 2-95
 running a compiled script 10-189
application creator 10-8 to 10-9
application logo 2-18 to 2-19, 2-25, 10-25 to 10-26
application programming interface for Apple Guide. *See* Apple Guide API
application signature 10-8 to 10-9
<App Logo> command 10-25 to 10-26
<App Text> command 10-27
authoring tips 2-5 to 2-95
autosaving guide files, compile option 5-9

B

Balloon Help 1-5
`BalloonID` function 10-110, 10-111
<Balloon Menu Text> command 10-16
Body format 2-44
branches 2-67 to 2-69
build files

 compiling 5-6
 creating 5-3 to 5-4
 and the <Include> command 10-18
building
 source files into a guide file 5-6 to 5-8
Build menu 4-7
<Build Sequence> command 10-50 to 10-51
Build utility 5-6
Build window 5-6
built-in event functions 10-188 to 10-190
buttons
 content area 2-55, 2-70, 2-72
 creating 10-57 to 10-82
 dimnable 2-72, 10-78 to 10-79
 formatting 2-72 to 2-74
 GoStart 2-71
 Huh? 2-71
 Index 2-17, 2-20
 localizing 2-94
 Look For 2-17, 2-21
 navigation 2-70, 2-71 to 2-72, 10-71 to 10-82
 radio 2-76 to 2-77, 10-64 to 10-68
 on decision panels 2-52
 on related topics panels 2-58
 titles for 2-76
 standard 2-70, 2-74, 10-57 to 10-60
 three-dimensional 2-70, 2-74 to 2-76, 10-60 to 10-64
 Topics 2-17, 2-19
 in Tutorial guide file 2-13
 types of 2-70

C

<Checkbox> command 10-69 to 10-71
checkboxes
 on decision panels 2-52
 titles for 2-76
`checkBoxState` context check 10-175 to 10-176
closure panels 2-60
<Coach Mark> command 10-118 to 10-119
coachmarks

- AppleScript 10-116 to 10-117
- associating with a panel 10-118 to 10-119
- item 10-108 to 10-111
- localizing 2-95
- menu 10-105 to 10-108
- object 10-111 to 10-113
- styles of 2-80 to 2-83
- testing 6-19
- types of 2-80
- window 10-113 to 10-115
- Column function 10-7, 10-85
- <Comment> command 10-17
- company name in access window 2-18, 2-25
- Compile arrow 5-7
- compile messages in Guide Maker
 - errors 5-10
 - interpreting 5-10 to 5-11
 - warnings 5-11
- compile options
 - autosaving guide files 5-9
 - creating symbol files 5-9
 - setting 5-8 to 5-9
 - warning when panels split 5-9
- compiling help source files 5-6 to 5-11, 5-14
- compiling help source files. *See* building help source files
- composing help instructions 2-39, 2-54
- condition functions
 - checkboxState 10-175 to 10-176
 - defining 10-172 to 10-175
 - radioButtonState 10-176 to 10-177
- content area buttons 2-70, 2-72
- content area of panel 2-39
- context checks
 - and Continue panels 2-85, 2-87
 - defining 10-172 to 10-175
 - determining 2-85 to 2-88
 - displaying branches based on 2-69
 - evaluation of 2-89 to 2-91
 - and first previous panels 2-89 to 2-90
 - in flowcharts 3-12
 - introduced 1-7 to 1-8
 - localizing 2-95
 - and Oops panels 2-85, 2-87 to 2-88, 2-91
 - testing 6-19
 - using 2-83 to 2-91
- Continue panels 2-65 to 2-66
 - compared with Oops panels 2-66
 - and context checks 2-85, 2-87
- control features. *See* buttons
- Convert arrow 8-6
- converting with Guide Maker 8-5 to 8-6
- Convert menu 4-7
- Convert utility 8-5
- Convert window 8-5

D

- decision panels 2-52 to 2-53, 2-69
- <Default Format> command 10-30 to 10-32
- <Default Nav Button Set> command 10-32 to 10-34
- default prompt set 2-40 to 2-43
- <Default Prompt Set> command 10-35 to 10-36
- <Define AppleScript Coach> command 10-116 to 10-117
- <Define Context Check> command 10-172 to 10-175
- <Define Event> command 10-178 to 10-181
- <Define Event List> command 10-181 to 10-182
- <Define Format> command 10-85 to 10-91
- <Define Item Coach> command 10-108 to 10-111
- <Define Menu Coach> command 10-105 to 10-108
- <Define Nav Button> command 10-71 to 10-77
- <Define Nav Button Set> command 10-80 to 10-82
- <Define Object Coach> command 10-111 to 10-113
- <Define Panel> command 10-52 to 10-54
- <Define Prompt Set> command 10-37 to 10-39
- <Define Sequence> command 10-39 to 10-42
- <Define Text Block> command 10-82 to 10-83
- <Define Transparent Format> command 10-91 to 10-92

<Define Window Coach> command 10-113 to 10-115
 definition panels 2-56 to 2-58
 "Definitions" heading 2-57
 <Delete Index> command 10-201 to 10-202
 <Delete Index Header> command 10-202 to 10-203
 <Delete Index Topic> command 10-203 to 10-204
 <Delete Topic Area> command 10-198
 <Delete Topic Area Header> command 10-199 to 10-200
 <Delete Topic Area Topic> command 10-200 to 10-201
 Diagnose menu 4-7
 Diagnose utility 6-3 to 6-7
 dialog boxes
 coaching items in 10-108 to 10-115
 DialogID function 10-110, 10-111
 <Dimmable Button Data> command 10-78 to 10-79
 DoAppleScript event function 10-189
 DoScript event function 10-189

E

<Else> command 10-156 to 10-158
 <End If> command 10-158 to 10-160
 <End Making Sure> command 10-171 to 10-172
 <End Panel> command 10-56 to 10-57
 <End Sequence> command 10-51 to 10-52
 <End Text Block> command 10-83 to 10-84
 Espy font 2-47
 event functions
 built-in 10-188 to 10-190
 defining 10-178 to 10-181
 testing 6-19
 event lists
 defining 10-181 to 10-182
 <Exception> command 10-147 to 10-148
 exception lists 3-13
 Export Guide File command 5-12

F

Finder, and system help 1-5
 first-level panels 2-66 to 2-67
 first previous panel 2-89 to 2-90
 flowcharts, designing help content with 3-7 to 3-12
 fonts
 Espy 2-47
 recommended 2-47
 footnotes in Windows Help files 8-4
 <Format> command 10-93 to 10-94
 formats
 Body 2-44
 creating 10-85 to 10-91
 default 10-30 to 10-32
 designing 2-45 to 2-46
 Full 2-43 to 2-45, 2-50
 localizing 2-93
 recommended 2-43 to 2-45
 specifying 10-93 to 10-94
 Tag 2-44
 Tag and Body 2-43 to 2-45
 transparent 10-91 to 10-92
 formatting
 buttons 2-72 to 2-74
 graphics 2-46 to 2-47
 panel text 2-47 to 2-48
 Full Access windows
 creating Look For features for 3-12 to 3-26
 defined 2-15
 features of 2-16 to 2-19
 headings for 2-33 to 2-35
 for Help guide files 2-11
 howdy text on 2-28 to 2-30
 designing Index features for 2-20 to 2-21
 designing Look For features for 2-21 to 2-24
 topic areas for 2-31 to 2-32
 designing Topics features for 2-19 to 2-20
 topics for 2-31 to 2-32
 Full format 2-43 to 2-45, 2-50
 for introductory panels 2-51

G

<Gestalt> command 10-10 to 10-11
 GoBack event function 10-163 to 10-167
 GoPanel event function 10-189
 GoStart button
 and the <Define Nav Button> command 10-71 to 10-77
 using 2-71
 graphics in panels
 formatting 2-46 to 2-47
 green "X" character coachmarks 2-80 to 2-82
 guide file additions. *See* Mixin guide files
 guide file index 3-19 to 3-22
 Guide File Info report 6-16 to 6-18
 guide files
 About 2-7, 2-9 to 2-10
 compiling 5-6 to 5-11
 designing 2-5 to 2-6
 designing with flowcharts 3-7 to 3-12
 developing content for 3-3 to 3-26
 Help 2-7
 in Help menu 2-7 to 2-8
 inappropriate material for 3-4
 indexes. *See* indexes
 invoking
 from Help menu 1-10 to 1-11
 with Command key shortcuts 2-8
 localizing 7-3 to 7-14
 main 2-14
 Mixin 2-5, 2-14 to 2-15
 navigating through 2-48 to 2-49
 Other 2-7, 2-14
 Shortcuts 2-7, 2-13 to 2-14
 system software 1-4
 testing 6-3 to 6-20
 Tutorial 2-7, 2-11 to 2-13
 types of 2-6 to 2-7
 uses of 1-8
 Guide Maker
 Build utility 5-6 to 5-8
 Convert utility 8-3 to 8-10
 Diagnose utility 6-3 to 6-7
 introduction to 4-3 to 4-7

Localize utility 7-3 to 7-14
 menus 4-6 to 4-7
 Reports feature 6-13 to 6-17
 Test Look For utility 6-8 to 6-12
 Guide Script 1-5
 Guide Script commands
 abbreviations A-1 to A-5
 descriptions 10-5 to 10-204
 quick reference to B-1 to B-6
 Guide Script source files. *See* help source files

H

<Header> command 10-135 to 10-137
 headings 2-33 to 2-35
 height
 of navigation buttons 10-74
 of panels
 default maximum 10-28
 default minimum 10-29
 of standard buttons 10-58
 help content
 deriving from reference documentation 3-4 to 3-6
 designing with flowcharts 3-7 to 3-12
 inappropriate material for 3-4
 Help guide files 2-7
 Help menu 2-7 to 2-8
 <Help Menu> command 10-14 to 10-16
 help source files 2-6
 hot areas 2-77
 <Hot Object> command 10-119 to 10-120
 hot objects 2-77 to 2-78
 <Hot Rectangle> command 10-120 to 10-122
 hot rectangles 2-77 to 2-78
 hot text 2-77 to 2-79
 compared with panels associated with Huh?
 buttons 2-78
 on related topics panels 2-58
 <Hot Text> command 10-122 to 10-123
 "How do I" heading 2-33
 <Howdy> command 10-24 to 10-25

howdy text 2-16, 2-28 to 2-30
 Huh? button
 and the <Define Nav Button> command 10-71 to 10-77
 and the <Dimmable Button Data> command 10-78 to 10-79
 panels associated with 2-61 to 2-63
 using a 2-71

I

<If> command 10-153 to 10-156
 <Ignore> command 10-145 to 10-147
 ignore lists 3-12, 3-23 to 3-24
 Import Resources command 5-12
 <Include> command 10-18 to 10-19
 Index button 2-17, 2-20
 <Index> command 10-128 to 10-130
 indexes
 designing 3-20 to 3-22
 terms in
 invisible 3-22
 visible 3-20 to 3-22
 testing 6-8 to 6-12
 <Index Instruction> command 10-127 to 10-128
 Index instructions 2-17
 <Index Sorting> command 10-132 to 10-133
 Index Sort Strings report 6-15 to 6-16
 index terms
 commands for 10-128 to 10-130, 10-130 to 10-133
 invisible 3-22
 visible 3-20 to 3-22
 information panels 2-54 to 2-55
 <Insert Index Header> command 10-195 to 10-196
 <Insert Index Topic> command 10-196 to 10-197
 <Insert Sequence> command 10-46 to 10-47
 <Insert Topic Area Header> command 10-192 to 10-193
 <Insert Topic Area Topic> command 10-193 to 10-194

introducing users to help system 2-16, 2-28 to 2-30
 introductory panels 2-51
 invisible index terms 3-22
 item coachmarks 10-108 to 10-111

J, K

<Jump Sequence> command 10-47 to 10-49

L

<Launch New Sequence> command 10-49 to 10-50
 LaunchNewSequence event function 10-189
 LaunchNewSequenceNewWindow event function 10-189
 Localize menu 4-7
 localizing
 AppleScript 2-95
 buttons 2-94
 coachmarks 2-95
 context checks 2-95
 formats 2-93
 panel text 2-92 to 2-93
 sequence display area 2-94
 source files 7-3 to 7-14
 and stemming 3-14
 Look For button 2-17, 2-21
 Look For features
 designing 2-21 to 2-24, 3-12 to 3-26
 exception lists 3-13
 ignore lists 3-13, 3-23 to 3-24
 matching search phrases with topics 3-15 to 3-19
 stemming 3-12 to 3-14
 synonym lists 3-13
 testing 6-8 to 6-12
 <Look For Instruction> command 10-141 to 10-142
 Look For instructions 2-17

- <Look For Results Instruction>
 command 10-144 to 10-145
- <Look For Search Btn Instruction>
 command 10-143 to 10-144
- <Look For String> command 10-143

M

- Macintosh Guide 1-5
- Macintosh Guide additions 1-4 to 1-5
- Macintosh Operating System, and system
 help 1-5
- main guide files 2-14
- <Make Sure> command 10-162 to 10-168
- <Max Height> command 10-28 to 10-29
- Menu Appearance command 5-12 to 5-13
- menu coachmarks 2-80, 10-105 to 10-108
- <Min Height> command 10-29 to 10-30
- <Mix in> command 10-19 to 10-20
- Mix in guide files
 - additional commands for 10-19 to 10-20,
 10-190 to 10-204
 - and topic areas and topics 2-31
 - creating 5-13 to 5-14
 - designing 2-14 to 2-15
 - introduced 2-5
- <Mix in Match> command 10-20 to 10-21
- movies
 - specifying 10-98 to 10-100

N

- Names to IDs report 6-14 to 6-15
- navigation arrows 2-39
- navigation bar 2-39
- navigation buttons 2-70, 2-71 to 2-72
- navigation button set
 - specifying 10-32 to 10-34, 10-43 to 10-44, 10-80
 to 10-82
- navigation methods 2-48 to 2-49

O

- object coachmarks 2-80, 10-111 to 10-113
- <On Panel Create> command 10-183 to 10-184
- <On Panel Destroy> command 10-184 to 10-185
- <On Panel Hide> command 10-187 to 10-188
- <On Panel Show> command 10-185 to 10-187
- Oops panels 2-63 to 2-65
 - compared with Continue panels 2-65
 - and context checks 2-85, 2-87 to 2-88, 2-91
- Other guide files 2-7, 2-14

P

- panel branches. *See* branches
- <Panel> command 10-45 to 10-46
- panel number 2-39
- <Panel Prompt> command 10-55 to 10-56
- panels 2-39 to 2-66
 - buttons in 2-70 to 2-77
 - characteristics of 2-35 to 2-39
 - commands for creating 10-52 to 10-57
 - content area of 2-39
 - default features of 2-37 to 2-39
 - features of 1-5 to 1-6
 - first level 2-67
 - first-level 2-66 to 2-67
 - first previous 2-89 to 2-90
 - formatting 2-43 to 2-48
 - formatting text in 2-47 to 2-48
 - graphics in
 - formatting 2-46 to 2-47
 - localizing 2-94
 - height of 2-36
 - navigating through 2-48 to 2-49
 - navigation bar on 2-39
 - numbering of 2-39
 - prompts for 2-39 to 2-43
 - second level 2-67
 - second-level 2-67. *See also* standard panel
 types
 - sequence display title area 2-38

panels (*continued*)
 width of 2-36
 panels associated with Huh? buttons 2-61 to 2-63
 compared with hot text 2-78
 panel sequences. *See* sequences
 <PICT> command 10-95 to 10-98
 'PICT' graphics 2-45
 pictures
 application logo 10-25 to 10-26
 specifying 10-95 to 10-98
 PlaySound event function 10-189
 Point function 10-7, 10-58, 10-59
 PowerBook Guide 1-5
 presentation panels. *See* panels
 presenting help instructions. *See* panels
 presenting topics to users. *See* access windows
 prompts
 default 2-40 to 2-43
 defining 10-34 to 10-39, 10-55 to 10-56
 designing 2-39 to 2-43
 formatting 2-40
 overriding 2-42 to 2-43
 sets of 2-40 to 2-42

Q

<QuickTime> command 10-98 to 10-100
 QuickTime movies 2-45

R

<Radio Button> command 10-64 to 10-66
 <Radio Button Launch New Seq>
 command 10-66 to 10-68
 radio buttons 2-76 to 2-77
 in Simple Access windows 2-26
 on decision panels 2-52
 on related topics panels 2-58
 title for 2-76
 radioButtonState context check 10-176 to 10-177

random access 2-16
 RECT function 10-7
 red arrow coachmarks 2-80 to 2-81
 RedArrow function 10-109
 red circle coachmarks 2-80 to 2-81
 red underline coachmarks 2-80 to 2-82
 reference documentation, deriving help content
 from 3-4 to 3-6
 related topics panels 2-58 to 2-59
 <Replace Sequence> command 10-190 to 10-191
 reports
 generating from Guide Maker 6-13 to 6-17
 Reports menu 4-7
 <Resource> command 10-101 to 10-103
 resources
 importing 10-100 to 10-105
 movie 10-98 to 10-100
 picture 10-95 to 10-98
 script 10-189
 sound 10-189
 version 10-11 to 10-12
 resource types
 'PICT' 10-95 to 10-98
 'scpt' 10-189
 'snd' 10-189
 RGBColor function 10-7, 10-86

S

saving guide files 5-8
 Scopes and Keys report 6-13 to 6-14
 second-level panels 2-67
 <Seq Nav Button Set> command 10-43 to 10-44
 sequence display title area 2-38
 <Sequence Prompt Set> command 10-42 to 10-43
 sequences
 commands for creating 10-39 to 10-52
 designing 2-66 to 2-69
 within other sequences 2-67 to 2-69
 sequential access 2-16
 Set Compile Options command 5-8
 Shortcuts 1-5

Shortcuts guide files 2-7, 2-13 to 2-14

Simple Access windows

- buttons in 2-26
- defined 2-15
- features of 2-26 to 2-28
- howdy text on 2-28 to 2-30
- for Shortcuts guide files 2-14
- topics for 2-33

Single List Access windows

- defined 2-15
- features of 2-25 to 2-26
- howdy text on 2-28 to 2-30
- topics for 2-32 to 2-33

<Skip If> command 10-160 to 10-162

<Sorting> command 10-130 to 10-131

sounds

- specifying 10-189

source files

- compiling 5-6

Speech Guide 1-5

split panels 5-9

<Standard Button> command 10-57 to 10-60

standard buttons 2-70, 2-74

standard panel types

- action 2-53 to 2-54
- closure 2-60
- Continue 2-65 to 2-66, 2-85
 - and context checks 2-87
- decision 2-52 to 2-53, 2-69
- definition 2-56 to 2-58
- information 2-54 to 2-55
- introductory 2-51
- Oops 2-63 to 2-65, 2-85
 - and context checks 2-87 to 2-88, 2-91
- panels associated with Huh? buttons 2-61 to 2-63, 2-79
- related topics 2-58 to 2-59
- tip 2-55 to 2-56
- transition 2-59 to 2-60

Standard Resources file 5-4, C-5

Standard Setup file 5-4, C-4 to C-5

<Starting Res Number> command 10-103 to 10-105

<Start Making Sure> command 10-168 to 10-171

<Startup Window> command 10-21 to 10-23

stemming 2-32, 3-12 to 3-14

strings

- specifying 10-6

styled text 2-47. *See also* formats

styles 2-47

SurfWriter Guide

- source files for C-1 to C-50

symbol files

- compile option 5-9
- creating 5-9

<Synonym> command 10-149 to 10-152

synonym lists 3-13

system software guide files 1-4

T

Tag and Body format 2-43 to 2-45

- for action panels 2-51, 2-54
- for long introductory panels 2-51

Tag format 2-44

tags 2-47, 2-54. *See also* Tag and Body format

testing guide files 6-3 to 6-20

Test Look For menu 4-7

Test Look For utility 6-8 to 6-12

text attributes

- and the <Define Format> command 10-85 to 10-91
- default for panel text 10-84
- of checkboxes 10-69
- of radio buttons 10-64 to 10-65
- of standard buttons 10-57 to 10-58

text blocks

- defining 10-82 to 10-84

text in panels

- formatting 2-47 to 2-48

<3D Button> command 10-60 to 10-64

three-dimensional buttons 2-26, 2-70, 2-74 to 2-76

tip panels 2-55 to 2-56

title area 2-18 to 2-19, 2-25

<Topic Area> command 10-125 to 10-126

topic areas 2-30 to 2-35

INDEX

<Topic Areas Instruction> command 10-124
<Topic> command 10-137 to 10-139
Topic instructions 2-17
topics 2-30 to 2-35
Topics button 2-17, 2-19
<Topics Instruction> command 10-134 to 10-135
transition panels 2-59 to 2-60
transparent formats 10-91 to 10-92
Tutorial, component of Apple Guide 1-5
Tutorial guide files 2-7, 2-11 to 2-13

preparing for conversion 8-3
underlined hidden text in 8-3
windows. *See* access windows
<World Script> command 10-13
<WorldScript> command 3-14

U

user interface 2-5 to 2-95
utilities
 Build 5-6
 Convert 8-5
 Diagnose 6-3 to 6-8
 Localize 7-3 to 7-6
 Test Look For 6-8 to 6-12
Utilities palette 4-5 to 4-7

V

<Version> command 10-11 to 10-12
version resource 10-11 to 10-12
Video Guide 1-5
Video Player 1-5
visible index terms 3-20 to 3-22

W, X, Y, Z

"Why can't I" heading 2-33
window coachmarks 2-80
windows
 coaching items in 10-108 to 10-113, 10-113 to 10-115
Windows Help files
 footnotes in 8-4

This Apple manual was written, edited, and composed on a desktop publishing system using Apple Macintosh computers and FrameMaker software. Proof pages were created on an Apple LaserWriter Pro printer. Final page negatives were output directly from text files on an Optrotech SPrint 220 imagesetter. Line art was created using Adobe Illustrator™ and Adobe Photoshop™. PostScript™, the page-description language for the LaserWriter, was developed by Adobe Systems Incorporated.

Text type is Palatino® and display type is Helvetica®. Bullets are ITC Zapf Dingbats®. Some elements, such as program listings, are set in Apple Courier.

Special thanks to Peter Commons, Brian Glenn, Jeremy Hewes, Glenn Katz, Stephanie Koester, and John Powers.

Acknowledgments to Shemin Gau, Devon Hubbard, James Miyake, Georgiann Puckett, and Susan Torres.

WRITERS

Sharon Everson, Ulla Hald, Daphne Steck

EDITOR

Jeanne Woodward

PROOFREADER

Wendy Krafft

ILLUSTRATORS

Ruth Anderson, Deborah Dennis

COVER ART/COVER DESIGN

Ruth Anderson

PRODUCTION EDITOR

Gerri Gray

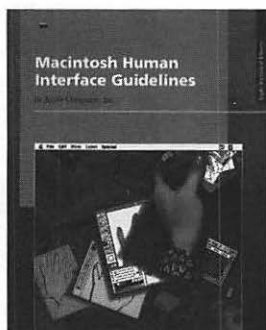
LEAD WRITER

Sharon Everson

PROJECT MANAGER

Patricia Eastman

Also available from Addison-Wesley and Apple Computer, Inc.



Macintosh Human Interface Guidelines Apple Computer, Inc.

This book provides authoritative information on the theory behind the Macintosh "look and feel" and the practice of using individual interface components. Anyone designing or creating a product for Macintosh computers needs to understand these core principles. Key topics address how people interact with computers, effective use of color, guidelines for using language clearly and consistently, and suggestions for creating an effective design process.

0-201-62216-5 \$29.95 paperback, 416 pages

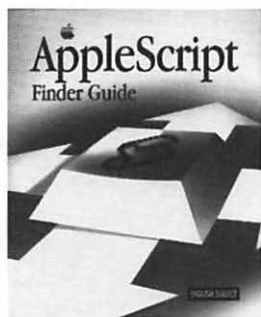


Electronic Guide to Macintosh Human Interface Design

Apple Computer, Inc.

This CD-ROM combines the full electronic text of the classic book, *Macintosh Human Interface Design* with the multimedia presentations of *Making It Macintosh* on one convenient and easy-to-use disc. This interactive guide contains more than 100 animated examples that demonstrate the correct use of Macintosh interface elements, including visual examples illustrating the appearance and behavior of menus, windows, dialog boxes, and icons.

0-201-40916-X \$49.95, Macintosh CD-ROM

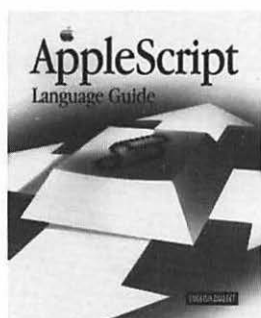


AppleScript Finder Guide

Apple Computer, Inc.

The definitive description of how the exciting new AppleScript scripting language works with the Finder, the application that controls the Macintosh desktop. You'll learn how to automate some of the most commonly performed activities on the Macintosh, such as opening and closing folders, and manipulating files.

0-201-40910-0 \$19.95 paperback, 176 pages

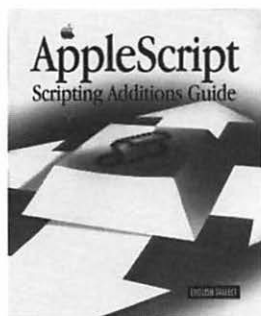


AppleScript Language Guide

Apple Computer, Inc.

This invaluable introduction to AppleScript contains an exhaustive reference section that describes how to use the value classes, commands, object classes, reference forms, expressions, control statements, handlers, and script objects recognized by AppleScript. Three handy appendices summarize AppleScript terms, define terms understood by the Scriptable Text Editor, and detail error messages.

0-201-40735-3 \$29.95 paperback, 416 pages



AppleScript Scripting Additions Guide

Apple Computer, Inc.

A companion to *AppleScript Language Guide*, this book provides all the information a programmer needs to begin writing effective scripting additions. Together with introductory material, this book shows how to install any scripting additions and invoke their commands, write scripting additions, and use the standard scripting additions commands.

0-201-40736-1 \$18.95 paperback, 144 pages

Order Information

Available wherever computer books are sold. Please call 1-800-822-6339 for the location of your nearest bookstore or to place your order. If you would like information about bulk quantity discounts direct from the publisher, please call our Corporate, Government, and Special Sales Department at 1-800-238-9682.

Addison-Wesley warrants the enclosed disc to be free of defects in materials and faulty workmanship under normal use for a period of ninety days after purchase. If a defect is discovered in the disc during this warranty period, a replacement disc can be obtained at no charge by sending the defective disc, postage prepaid, with proof of purchase to:

Addison-Wesley Publishing Company
Editorial Department
Trade Computer Books Division
One Jacob Way
Reading, MA 01867

After the 90-day period, a replacement will be sent upon receipt of the defective disc and a check or money order for \$10.00, payable to Addison-Wesley Publishing Company.

Addison-Wesley makes no warranty or representation, either express or implied, with respect to this software, its quality, performance, merchantability, or fitness for a particular purpose. In no event will Addison-Wesley, its distributors, or dealers be liable for direct, indirect, special, incidental, or consequential damages arising out of the use or inability to use the software. The exclusion of implied warranties is not permitted in some states. Therefore, the above exclusion may not apply to you. This warranty provides you with specific legal rights. There may be other rights that you may have that vary from state to state.



The CD contains essential tools and resources to develop guide files, including:

- Guide Maker, the software you need to build, test, and localize guide files
- sample guide files
- sample source files
- sample context checks
- Apple Guide interface files
- a searchable command reference to all Guide Script commands



System requirements:

- A Macintosh with at least 4 MB of memory and a CD-ROM drive.
- Macintosh System 7.5 or later.