

Progress Help Development Guide

© 2001 Progress Software Corporation. All rights reserved.

Progress® software products are copyrighted and all rights are reserved by Progress Software Corporation. This manual is also copyrighted and all rights are reserved. This manual may not, in whole or in part, be copied, photocopied, translated, or reduced to any electronic medium or machine-readable form without prior consent, in writing, from Progress Software Corporation.

The information in this manual is subject to change without notice, and Progress Software Corporation assumes no responsibility for any errors that may appear in this document.

The references in this manual to specific platforms supported are subject to change.

Progress, Progress Results, Provision and WebSpeed are registered trademarks of Progress Software Corporation in the United States and other countries. Apptivity, AppServer, ProVision Plus, SmartObjects, IntelliStream, and other Progress product names are trademarks of Progress Software Corporation.

SonicMQ is a trademark of Sonic Software Corporation in the United States and other countries.

Progress Software Corporation acknowledges the use of Raster Imaging Technology copyrighted by Snowbound Software 1993-1997 and the IBM XML Parser for Java Edition.

© IBM Corporation 1998-1999. All rights reserved. U.S. Government Users Restricted Rights — Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

Progress is a registered trademark of Progress Software Corporation and is used by IBM Corporation in the mark Progress/400 under license. Progress/400 AND 400® are trademarks of IBM Corporation and are used by Progress Software Corporation under license.

Java and all Java-based marks are trademarks or registered trademarks of Sun Microsystems, Inc. in the United States and other countries.

Any other trademarks and/or service marks contained herein are the property of their respective owners.

.

May 2001



Product Code: 4523
Item Number: 81092;9.1C

Contents

Preface . . . . . **ix**
Purpose ix
Audience ix
Organization of This Manual x
How to Use This Manual xi
Typographical Conventions xi
Syntax Notation xii
Example Procedures xvi
Progress Messages xvii
Other Useful Documentation xix
Getting Started xix
Development Tools xx
Reporting Tools xxi
4GL xxii
Database xxiii
DataServers xxiii
SQL-89/Open Access xxiv
SQL-92 xxiv
Deployment xxv
WebSpeed xxv
Reference xxvi

1. Providing Help for Progress Applications **1–1**
1.1 Help Strings 1–2
1.1.1 Associating Help Strings with Database Fields 1–3
1.1.2 Specifying Help Strings with the Progress 4GL 1–4
1.1.3 Defining Help Strings with the AppBuilder 1–5

1.2 ToolTips 1–5
1.2.1 Implementing ToolTips with the Progress 4GL 1–6
1.2.2 Implementing ToolTips with the AppBuilder 1–7
1.3 Overview of Online Help Systems 1–8
1.3.1 The SYSTEM-HELP Statement 1–10
1.3.2 Quitting Help When Exiting the Application. 1–13
1.3.3 Coding the Help Calling Interface 1–13
1.4 Accessing Online Help from the Menu Bar 1–17
1.4.1 Calling the Help Topics Dialog Box. 1–17
1.4.2 Calling the Index Function of the Help Topics Dialog Box 1–18
1.4.3 Calling the How to Use Help File 1–20
1.5 Accessing Online Help with a Help Button 1–20
1.6 Accessing Online Help with a Help Key 1–22
1.7 Context-sensitive Help 1–24
1.7.1 Implementing Context-sensitive Help with the Progress 4GL 1–25
1.7.2 Implementing Context-sensitive Help with the AppBuilder 1–29

2. Overview of Help System Development 2–1
2.1 Basic Features of Help Systems and Development Considerations 2–2
2.1.1 Help Topics and the Help Topics Dialog Box 2–2
2.1.2 Help Windows 2–6
2.1.3 Hypertext Links 2–10
2.2 Designing a Help System 2–13
2.2.1 Organizing Help Topics. 2–13
2.2.2 Topic Style Guidelines 2–14

3. Creating Help Topic Files 3–1
3.1 Help Topic Files 3–2
3.1.1 Managing Help Topic Files 3–4
3.1.2 Overview of Creating Help Topic Files 3–4
3.1.3 Separating Topics 3–5
3.1.4 Inserting Control Codes 3–6
3.1.5 Assigning Topic IDs 3–10
3.1.6 Assigning Topic Titles 3–10
3.1.7 Creating Topics for Context-sensitive Help 3–11
3.1.8 Writing Topic Text 3–11
3.1.9 Assigning Index Keywords 3–12
3.1.10 Creating Browse Sequences. 3–12
3.1.11 Assigning Build Tags. 3–14
3.1.12 Creating Non-scrolling Regions 3–15
3.2 Including Graphics in Help Topics 3–16
3.2.1 Tools for Creating Graphics 3–17
3.2.2 Placing Graphics By Reference 3–17

3.3 Adding Hypertext Links 3–19
3.3.1 Components of a Hypertext Link 3–20
3.3.2 Creating Jumps 3–21
3.3.3 Creating Popup Links 3–22
3.3.4 Creating Cross-file Links 3–24
3.3.5 Creating Secondary Window Links 3–24
3.3.6 Adding KLinks and ALinks 3–25
3.3.7 Adding Hotspots to Graphics 3–27
3.4 Using Help Macros 3–28
3.4.1 Help Macro Syntax 3–28
3.4.2 Defining Macros in Jumps 3–30
3.4.3 Defining Macros in Control Codes 3–31
3.5 Saving Help Topics in RTF Files 3–32

4. Completing Help Systems 4–1
4.1 Creating a Contents File 4–2
4.1.1 Contents File Commands 4–3
4.1.2 Headings and Topics 4–4
4.1.3 Creating a Contents File with Help Workshop 4–4
4.2 Creating a Project File 4–6
4.2.1 Project File Sections 4–7
4.2.2 Mapping Context-sensitive Help Topics 4–13
4.2.3 Project File Conventions 4–15
4.2.4 Creating a Project File with Help Workshop 4–16
4.3 Compiling a Help Project 4–18
4.3.1 Interpreting Compiler Messages 4–21
4.3.2 Running the Compiler from the Command Line 4–21
4.4 Testing a Help System 4–22

A. Files Mentioned in This Guide A–1

B. Options in the [Options] Section of the Project File B–1

Glossary Glossary–1

Index Index–1

Figures

Figure 1–1: Help String for a Field-level Widget 1–2
Figure 1–2: Defining a Help String in the Field Schema Definition 1–3
Figure 1–3: Defining a Help String in the Advanced Properties Dialog Box 1–5
Figure 1–4: Defining a ToolTip in the Property Sheet 1–7
Figure 1–5: Run-time Appearance of a ToolTip 1–7
Figure 1–6: Help Topics Dialog Box for Procedure Editor Help 1–18
Figure 1–7: Help Topics Dialog Box with Index Tab on Top 1–19
Figure 1–8: Dialog Box with a Help Button 1–20
Figure 1–9: Context-sensitive Help for the Buffer Information Dialog Box 1–21
Figure 1–10: HELP Event Hierarchy 1–23
Figure 1–11: Mouse Cursor Showing Context-sensitivity Enabled. 1–24
Figure 1–12: Context-sensitive Help Displayed. 1–24
Figure 1–13: Property Sheet for a Window with Context-sensitive Help 1–30
Figure 1–14: Property Sheet with a Help Topic Context ID Specified 1–31
Figure 2–1: Help Topics Dialog Box with Contents Tab on Top 2–3
Figure 2–2: Help Topics Dialog Box with Index Tab on Top 2–4
Figure 2–3: Help Topics Dialog Box with Find Tab on Top 2–5
Figure 2–4: Main Window for the Procedure Editor Help File 2–7
Figure 2–5: A Secondary Window 2–9
Figure 2–6: Hypertext Link Examples 2–10
Figure 2–7: Destination Topic for “File” Jump 2–11
Figure 2–8: Popup Link Activated 2–12
Figure 2–9: Printed Documentation Versus Online Help 2–14
Figure 3–1: Procedure Editor Help Topic 3–3
Figure 3–2: Help Topics Separated by Page Breaks 3–5
Figure 3–3: Example of Control Codes for a Help Topic 3–9
Figure 3–4: Browse Sequence Diagram 3–13
Figure 3–5: Example of a Non-scrolling Region 3–15
Figure 3–6: Graphic Displayed in a Help Window 3–17
Figure 3–7: Hypertext Link and Its Components 3–20
Figure 3–8: Help Macro in a Jump 3–30
Figure 3–9: Help Macro Viewed with MS-Word 3–31
Figure 4–1: Help Contents File in Help Workshop 4–6
Figure 4–2: [MAP] Section Example 4–14
Figure 4–3: Help Project File in Help Workshop 4–18
Figure 4–4: Compiling a Help Project 4–20

Tables

Table 2–1:	Main Window Menus	2–8
Table 2–2:	Main Window Buttons	2–8
Table 2–3:	Secondary Window Buttons	2–9
Table 2–4:	Help Topic Style Guidelines	2–15
Table 3–1:	Summary of Control Codes	3–6
Table 3–2:	Supported Graphics Formats	3–16
Table 3–3:	Help Statements for Placing Graphics	3–18
Table 4–1:	Contents File Commands	4–3
Table 4–2:	Project File Considerations	4–7
Table 4–3:	Summary of Project File Sections	4–8
Table 4–4:	Help Source Files	4–19
Table A–1:	Files in This Guide	A–1
Table B–1:	[OPTIONS] Section Options	B–1

Procedures

r-syshlp.p 1–15

Preface

Purpose

This document describes how to provide online help for Progress applications.

Audience

This manual is written for the Windows help development team. The team typically consists of two or more individuals who jointly develop the help system for a Progress application:

- A help author (technical writer) who composes and compiles the *help source files* which include:
  - Help topic files containing the information that explains how to use the application as well as hypertext jumps among topics and other commands and information.
  - Graphics files that are compiled into the final help file.
  - The help project file, which contains instructions for the compiler.
  - The contents file, which contains information that determines the appearance of the Contents tab of the Help Topics dialog box.
- A Progress programmer who provides access to help information in three ways:
  - By embedding help strings directly in the database schema and/or application code. Help strings appear in the status area of an application window.
  - By adding ToolTips to graphical items.
  - By providing software mechanisms to link an application to a help file that has been authored and compiled by the technical writer.

Organization of This Manual

Chapter 1, "Providing Help for Progress Applications"

This chapter provides instructions to the Progress application developer for creating help strings and integrating help into a Progress application using the SYSTEM-HELP statement and other Progress 4GL syntax. This chapter also briefly discusses the elements of a help system from the perspective of the application developer.

Chapter 2, "Overview of Help System Development"

This chapter discusses information design considerations for help files. This chapter also discusses the elements of the WinHelp 4.0 Help Topics dialog box and help viewer windows, and outlines the steps necessary for creating and compiling the help source files.

Chapter 3, "Creating Help Topic Files"

This chapter describes how to create help topic files using Microsoft Word, including creating topic text, coding hypertext links, and creating index keywords. This chapter also discusses graphic files and how to reference them in a help topic file.

Chapter 4, "Completing Help Systems"

This chapter describes how to create a help project file and how to compile the help source files into a binary help file, both using the Microsoft Help Workshop, which is supplied with your Progress products under license from Microsoft.

This chapter also describes how to create a contents file. A contents file contains the information used by the help engine to populate the Contents tab of the Help Topics dialog box, and other run-time commands used by the help engine.

Appendix A, "Files Mentioned in This Guide"

This appendix contains a list of the names, locations, and descriptions of the files mentioned in this guide.

Appendix B, "Options in the [Options] Section of the Project File"

This appendix contains a list of the options, syntax, and descriptions of the options in the [OPTIONS] section of the project (.hpj) file.

"Glossary"

How to Use This Manual

The roles of the help author and the Progress application programmer are distinct. The only way these roles overlap is that they must agree on what context IDs (discussed in Chapter 3) should be associated with what help topics. Therefore, read the chapters that pertain to your role in detail, and skim or read to your level of interest, the material in the other chapters as follows:

- Progress application programmers should focus primarily on Chapter 1, "Providing Help for Progress Applications".
- Help authors should focus on Chapter 2, "Overview of Help System Development," Chapter 3, "Creating Help Topic Files," and Chapter 4, "Completing Help Systems." If you are using a help authoring tool, the material in Chapters 3 and 4 will probably be covered in detail in your tool's documentation. Also note that using a help authoring tool enables you to create help files in a way that is less "manual" than the way described here.

  Help authors whose experience is limited to developing WinHelp 3.x (16-bit) help files, can learn about the new features of WinHelp 4.0 (32-bit) by reading the *Help Author's Guide* (`hcw.hlp`), the help file associated with the Microsoft Help Workshop (`hcw.exe`) and following the links under the What's New in Version 4.0 book icon.

The text examples and many of the illustrations in this guide are from the Progress Procedure Editor online help. The Progress Procedure Editor help topic, graphic, and project files are shipped with Progress. To gain more familiarity with the process of help authoring, you can open these files and experiment with editing, building, and compiling them on your own.

Typographical Conventions

This manual uses the following typographical conventions:

- **Bold typeface** indicates:
  - Commands or characters that the user types
  - That a word carries particular weight or emphasis
- *Italic typeface* indicates:
  - Progress variable information that the user supplies
  - New terms
  - Titles of complete publications

- `Monospaced typeface` indicates:
  - Code examples
  - System output
  - Operating system filenames and pathnames

The following typographical conventions are used to represent keystrokes:

- Small capitals are used for Progress key functions and generic keyboard keys.

  END-ERROR, **GET**, **GO**
  ALT, **CTRL**, **SPACEBAR**, **TAB**

- When you have to press a combination of keys, they are joined by a dash. You press and hold down the first key, then press the second key.

  CTRL-X

- When you have to press and release one key, then press another key, the key names are separated with a space.

  ESCAPE H
  ESCAPE CURSOR-LEFT

Syntax Notation

The syntax for each component follows a set of conventions:

- Uppercase words are keywords. Although they are always shown in uppercase, you can use either uppercase or lowercase when using them in a procedure.

  In this example, ACCUM is a keyword:

  SYNTAX

  ```
  ACCUM aggregate expression
  ```

- Italics identify options or arguments that you must supply. These options can be defined as part of the syntax or in a separate syntax identified by the name in italics. In the ACCUM function above, the *aggregate* and *expression* options are defined with the syntax for the ACCUM function in the *Progress Language Reference*.

- You must end all statements (except for DO, FOR, FUNCTION, PROCEDURE, and REPEAT) with a period. DO, FOR, FUNCTION, PROCEDURE, and REPEAT statements can end with either a period or a colon, as in this example:

```
FOR EACH Customer:
   DISPLAY Name.
END.
```

- Square brackets (**[ ]**) around an item indicate that the item, or a choice of one of the enclosed items, is optional.

  In this example, STREAM *stream*, UNLESS-HIDDEN, and NO-ERROR are optional:

  SYNTAX

```
DISPLAY [ STREAM stream ] [ UNLESS-HIDDEN ] [ NO-ERROR ]
```

  In some instances, square brackets are not a syntax notation, but part of the language.

  For example, this syntax for the INITIAL option uses brackets to bound an initial value list for an array variable definition. In these cases, normal text brackets ([ ]) are used:

  SYNTAX

```
INITIAL [ constant [ , constant ] ... ]
```

  NOTE: The ellipsis (**. . .**) indicates repetition, as shown in a following description.

- Braces (**{ }**) around an item indicate that the item, or a choice of one of the enclosed items, is required.

  In this example, you must specify the items BY and *expression* and can optionally specify the item DESCENDING, in that order:

  SYNTAX

```
{ BY expression [ DESCENDING ] }
```

  In some cases, braces are not a syntax notation, but part of the language.

For example, a called external procedure must use braces when referencing arguments passed by a calling procedure. In these cases, normal text braces ({ }) are used:

SYNTAX

```
{ &argument-name }
```

- A vertical bar (**|**) indicates a choice.

  In this example, EACH, FIRST, and LAST are optional, but you can only choose one:

  SYNTAX

  ```
  PRESELECT [ EACH | FIRST | LAST ] record-phrase
  ```

  In this example, you must select one of *logical-name* or *alias*:

  SYNTAX

  ```
  CONNECTED ( { logical-name | alias } )
  ```

- Ellipses (**. . .**) indicate that you can choose one or more of the preceding items. If a group of items is enclosed in braces and followed by ellipses, you must choose one or more of those items. If a group of items is enclosed in brackets and followed by ellipses, you can optionally choose one or more of those items.

  In this example, you must include two expressions, but you can optionally include more. Note that each subsequent expression must be preceded by a comma:

  SYNTAX

  ```
  MAXIMUM ( expression , expression [ , expression ] . . . )
  ```

  In this example, you must specify MESSAGE, then at least one of *expression* or SKIP, but any additional number of *expression* or SKIP is allowed:

  SYNTAX

  ```
  MESSAGE { expression | SKIP [ (n) ] } . . .
  ```

In this example, you must specify {*include-file*, then optionally any number of *argument* or &*argument-name* = "*argument-value*", and then terminate with }:

SYNTAX

```
{ include-file
    [ argument | &argument-name = "argument-value" ] ... }
```

- In some examples, the syntax is too long to place in one horizontal row. In such cases, optional items appear individually bracketed in multiple rows in order, left-to-right and top-to-bottom. This order generally applies, unless otherwise specified. Required items also appear on multiple rows in the required order, left-to-right and top-to-bottom. In cases where grouping and order might otherwise be ambiguous, braced (required) or bracketed (optional) groups clarify the groupings.

  In this example, WITH is followed by several optional items:

  SYNTAX

```
WITH [ ACCUM max-length ] [ expression DOWN ]
  [ CENTERED ] [ n COLUMNS ] [ SIDE-LABELS ]
  [ STREAM-IO ]
```

  In this example, ASSIGN requires one of two choices: either one or more of *field*, or one of *record*. Other options available with either *field* or *record* are grouped with braces and brackets. The open and close braces indicate the required order of options:

  SYNTAX

```
ASSIGN {   { [ FRAME frame ]
               { field [ = expression ] }
               [ WHEN expression ]
           } ...
         | { record [ EXCEPT field ... ] }
       }
```

Example Procedures

This manual provides example procedures that illustrate syntax and concepts. Examples use the following conventions:

- They appear in boxes with borders.
- If they are available online, the name of the procedure appears above the left corner of the box and starts with a prefix associated with the manual that references it, as follows:
  - e- - *Progress External Program Interfaces*, for example, e-ddeex1.p
  - lt- - *Progress Language Tutorial*, for example, lt-05-s3.p
  - p- - *Progress Programming Handbook*, for example, p-br01.p
  - r- - *Progress Language Reference*, for example, r-dynbut.p

  If the name does not start with a listed prefix, the procedure is not available online.
- If they are not available online, they compile as shown, but might not execute for lack of completeness.

Accessing Files in Procedure Libraries

Documentation examples are stored in procedure libraries, prodoc.pl and prohelp.pl, in the src directory where Progress is installed. The sample Progress procedure, r-syshlp.p, demonstrates various SYSTEM-HELP statement calls.

You must first create all subdirectories required by a library before attempting to extract files from the library. You can see what directories and subdirectories a library needs by using the PROLIB -list command to view the contents of the library. See the *Progress Client Deployment Guide* for more details on the PROLIB utility.

Extracting source files from a procedure library involves running PROENV to set up your Progress environment, creating the directory structure for the files you want to extract, and running PROLIB.

1 ♦ From the Control Panel or the Progress Program Group, double-click the Proenv icon.

2 ♦ The Proenv Window appears, with the proenv prompt.

Running Proenv sets the DLC environment variable to the directory where you installed Progress (by default, C:\Program Files\Progress). Proenv also adds the DLC environment variable to your PATH environment variable and adds the bin directory (PATH=%DLC%;%DLC%\bin;%PATH%).

3 ♦ Enter the following command at the proenv prompt to create the `prodoc` directory in your Progress working directory (by default, `C:\Progress\Wrk`):

```
MKDIR prodoc
```

4 ♦ Create the `langref` directory under `prodoc`:

```
MKDIR prodoc\langref
```

5 ♦ To extract all examples in a procedure library directory, run the PROLIB utility. Note that you must use double quotes because "Program Files" contains an embedded space:

```
PROLIB "%DLC%\src\prodoc.pl" -extract prodoc\langref\*.*
```

PROLIB extracts all examples into `prodoc\langref`.

To extract one example, run PROLIB and specify the file that you want to extract as it is stored in the procedure library:

```
PROLIB "%DLC%\src\prodoc.pl" -extract prodoc/langref/r-syshlp.p
```

PROLIB extracts `r-syshlp.p` into `prodoc\langref`.

Progress Messages

Progress displays several types of messages to inform you of routine and unusual occurrences:

- Execution messages inform you of errors encountered while Progress is running a procedure (for example, if Progress cannot find a record with a specified index field value).
- Compile messages inform you of errors found while Progress is reading and analyzing a procedure prior to running it (for example, if a procedure references a table name that is not defined in the database).
- Startup messages inform you of unusual conditions detected while Progress is getting ready to execute (for example, if you entered an invalid startup parameter).

After displaying a message, Progress proceeds in one of several ways:

- Continues execution, subject to the error-processing actions that you specify, or that are assumed, as part of the procedure. This is the most common action taken following execution messages.
- Returns to the Progress Procedure Editor so that you can correct an error in a procedure. This is the usual action taken following compiler messages.
- Halts processing of a procedure and returns immediately to the Procedure Editor. This does not happen often.
- Terminates the current session.

Progress messages end with a message number in parentheses. In this example, the message number is 200:

```
** Unknown table name table. (200)
```

Use Progress online help to get more information about Progress messages. Many Progress tools include the following Help menu options to provide information about messages:

- Choose Help→Recent Messages to display detailed descriptions of the most recent Progress message and all other messages returned in the current session.
- Choose Help→Messages, then enter the message number to display a description of any Progress message. (If you encounter an error that terminates Progress, make a note of the message number before restarting.)
- In the Procedure Editor, press the **HELP** key (**F2** or **CTRL-W**).

Other Useful Documentation

This section lists Progress Software Corporation documentation that you might find useful. Unless otherwise specified, these manuals support both Windows **and** Character platforms and are provided in electronic documentation format on CD-ROM.

Getting Started

Progress Electronic Documentation Installation and Configuration Guide (Hard copy only)

A booklet that describes how to install the Progress EDOC viewer and collection on UNIX and Windows.

Progress Installation and Configuration Guide Version 9 for UNIX

A manual that describes how to install and set up Progress Version 9.1 for the UNIX operating system.

Progress Installation and Configuration Guide Version 9 for Windows

A manual that describes how to install and set up Progress Version 9.1 for all supported Windows and Citrix MetaFrame operating systems.

Progress Version 9 Product Update Bulletin

A guide that provides a brief description of each new feature of the release. The booklet also explains where to find more detailed information in the documentation set about each new feature.

Progress Application Development Environment — Getting Started (Windows only)

A practical guide to graphical application development within the Progress Application Development Environment (ADE). This guide includes an overview of the ADE and its tools, an overview of Progress SmartObject technology, and tutorials and exercises that help you better understand SmartObject technology and how to use the ADE to develop applications.

Progress Language Tutorial for Windows and *Progress Language Tutorial for Character*

Platform-specific tutorials designed for new Progress users. The tutorials use a step-by-step approach to explore the Progress application development environment using the 4GL.

Progress Master Glossary for Windows and *Progress Master Glossary for Character* (EDOC only)

Platform-specific master glossaries for the Progress documentation set. These books are in electronic format only.

Progress Master Index and Glossary for Windows and *Progress Master Index and Glossary for Character* (Hard copy only)

Platform-specific master indexes and glossaries for the Progress hard-copy documentation set.

Progress Startup Command and Parameter Reference

A reference manual that describes the Progress startup commands and parameters in alphabetical order.

Welcome to Progress (Hard copy only)

A booklet that explains how Progress software and media are packaged. An icon-based map groups the documentation by functionality, providing an overall view of the documentation set. *Welcome to Progress* also provides descriptions of the various services Progress Software Corporation offers.

Development Tools

Progress ADM 2 Guide

A guide to using the Application Development Model, Version 2 (ADM 2) application architecture to develop Progress applications. It includes instructions for building and using Progress SmartObjects.

Progress ADM 2 Reference

A reference for the Application Development Model, Version 2 (ADM 2) application. It includes descriptions of ADM 2 functions and procedures.

Progress AppBuilder Developer's Guide (Windows only)

A programmer's guide to using the Progress AppBuilder visual layout editor. AppBuilder is a Rapid Application Development (RAD) tool that can significantly reduce the time and effort required to create Progress applications.

Progress Basic Database Tools (Character only; information for Windows is in online help)

A guide for the Progress Database Administration tools, such as the Data Dictionary.

Progress Basic Development Tools (Character only; information for Windows is in online help)

A guide for the Progress development toolset, including the Progress Procedure Editor and the Application Compiler.

Progress Debugger Guide

A guide for the Progress Application Debugger. The Debugger helps you trace and correct programming errors by allowing you to monitor and modify procedure execution as it happens.

Progress Translation Manager Guide (Windows only)

A guide that describes how to use the Progress Translation Manager tool to manage the entire process of translating the text phrases in Progress applications.

Progress Visual Translator Guide (Windows only)

A guide that describes how to use the Progress Visual Translator tool to translate text phrases from procedures into one or more spoken languages.

Reporting Tools

Progress Report Builder Deployment Guide (Windows only)

An administration and development guide for generating Report Builder reports using the Progress Report Engine.

Progress Report Builder Tutorial (Windows only)

A tutorial that provides step-by-step instructions for creating eight sample Report Builder reports.

Progress Report Builder User's Guide (Windows only)

A guide for generating reports with the Progress Report Builder.

Progress Results Administration and Development Guide (Windows only)

A guide for system administrators that describes how to set up and maintain the Results product in a graphical environment. This guide also describes how to program, customize, and package Results with your own products. In addition, it describes how to convert character-based Results applications to graphical Results applications.

Progress Results User's Guide for Windows and *Progress Results User's Guide for UNIX*

Platform-specific guides for users with little or no programming experience that explain how to query, report, and update information with Results. Each guide also helps advanced users and application developers customize and integrate Results into their own applications.

4GL

Building Distributed Applications Using the Progress AppServer

A guide that provides comprehensive information about building and implementing distributed applications using the Progress AppServer. Topics include basic product information and terminology, design options and issues, setup and maintenance considerations, 4GL programming details, and remote debugging.

Progress External Program Interfaces

A guide to accessing non-Progress applications from Progress. This guide describes how to use system clipboards, UNIX named pipes, Windows dynamic link libraries, Windows dynamic data exchange, Windows ActiveX controls, and the Progress Host Language Call Interface to communicate with non-Progress applications and extend Progress functionality.

Progress Internationalization Guide

A guide to developing Progress applications for markets worldwide. The guide covers both internationalization—writing an application so that it adapts readily to different locales (languages, cultures, or regions)—and localization—adapting an application to different locales.

Progress Language Reference

A three-volume reference set that contains extensive descriptions and examples for each statement, phrase, function, operator, widget, attribute, method, and event in the Progress language.

Progress Programming Handbook

A two-volume handbook that details advanced Progress programming techniques.

Database

Progress Database Design Guide

A guide that uses a sample database and the Progress Data Dictionary to illustrate the fundamental principles of relational database design. Topics include relationships, normalization, indexing, and database triggers.

Progress Database Administration Guide and Reference

This guide describes Progress database administration concepts and procedures. The procedures allow you to create and maintain your Progress databases and manage their performance.

DataServers

Progress DataServer Guides

These guides describe how to use the DataServers to access non-Progress databases. They provide instructions for building the DataServer modules, a discussion of programming considerations, and a tutorial. Each DataServer has its own guide, for example, the *Progress DataServer for ODBC Guide*, the *Progress DataServer for ORACLE Guide*, or the *Progress/400 Product Guide*.

MERANT ODBC Branded Driver Reference

The Enterprise DataServer for ODBC includes MERANT ODBC drivers for all the supported data sources. For configuration information, see the MERANT documentation, which is available as a PDF file in *installation-path*\odbc. To read this file you must have the Adobe Acrobat Reader Version 3.1 or higher installed on your system. If you do not have the Adobe Acrobat Reader, you can download it from the Adobe Web site at: `http://www.adobe.com/prodindex/acrobat/readstep.html`.

SQL-89/Open Access

Progress Embedded SQL-89 Guide and Reference

A guide to Progress Embedded SQL-89 for C, including step-by-step instructions on building ESQL-89 applications and reference information on all Embedded SQL-89 Preprocessor statements and supporting function calls. This guide also describes the relationship between ESQL-89 and the ANSI standards upon which it is based.

Progress Open Client Developer's Guide

A guide that describes how to write and deploy Java and ActiveX applications that run as clients of the Progress AppServer. The guide includes information about how to expose the AppServer as a set of Java classes or as an ActiveX server.

Progress SQL-89 Guide and Reference

A user guide and reference for programmers who use interactive Progress/SQL-89. It includes information on all supported SQL-89 statements, SQL-89 Data Manipulation Language components, SQL-89 Data Definition Language components, and supported Progress functions.

SQL-92

Progress Embedded SQL-92 Guide and Reference

A guide to Progress Embedded SQL-92 for C, including step-by-step instructions for building ESQL-92 applications and reference information about all Embedded SQL-92 Preprocessor statements and supporting function calls. This guide also describes the relationship between ESQL-92 and the ANSI standards upon which it is based.

Progress JDBC Driver Guide

A guide to the Java Database Connectivity (JDBC) interface and the Progress SQL-92 JDBC driver. It describes how to set up and use the driver and details the driver's support for the JDBC interface.

Progress ODBC Driver Guide

A guide to the ODBC interface and the Progress SQL-92 ODBC driver. It describes how to set up and use the driver and details the driver's support for the ODBC interface.

Progress SQL-92 Guide and Reference

A user guide and reference for programmers who use Progress SQL-92. It includes information on all supported SQL-92 statements, SQL-92 Data Manipulation Language components, SQL-92 Data Definition Language components, and Progress functions. The guide describes how to use the Progress SQL-92 Java classes and how to create and use Java stored procedures and triggers.

Deployment

Progress Client Deployment Guide

A guide that describes the client deployment process and application administration concepts and procedures.

Progress Developer's Toolkit

A guide to using the Developer's Toolkit. This guide describes the advantages and disadvantages of different strategies for deploying Progress applications and explains how you can use the Toolkit to deploy applications with your selected strategy.

Progress Portability Guide

A guide that explains how to use the Progress toolset to build applications that are portable across all supported operating systems, user interfaces, and databases, following the Progress programming model.

WebSpeed

Getting Started with WebSpeed

Provides an introduction to the WebSpeed Workshop tools for creating Web applications. It introduces you to all the components of the WebSpeed Workshop and takes you through the process of creating your own Intranet application.

WebSpeed Installation and Configuration Guide

Provides instructions for installing WebSpeed on Windows and UNIX systems. It also discusses designing WebSpeed environments, configuring WebSpeed Brokers, WebSpeed Agents, and the NameServer, and connecting to a variety of data sources.

WebSpeed Developer's Guide

Provides a complete overview of WebSpeed and the guidance necessary to develop and deploy WebSpeed applications on the Web.

WebSpeed Product Update Bulletin

A booklet that provides a brief description of each new feature of the release. The booklet also explains where to find more detailed information in the documentation set about each new feature.

Welcome to WebSpeed! (Hard copy only)

A booklet that explains how WebSpeed software and media are packaged. *Welcome to WebSpeed!* also provides descriptions of the various services Progress Software Corporation offers.

Reference

Pocket Progress (Hard copy only)

A reference that lets you quickly look up information about the Progress language or programming environment.

Pocket WebSpeed (Hard copy only)

A reference that lets you quickly look up information about the SpeedScript language or the WebSpeed programming environment.

1

Providing Help for Progress Applications

Online help provides users with immediate access to information while working with a software application. Progress provides three ways to supply online help to application users-with help strings, ToolTips, and online help systems.

This chapter explains how to implement:

- Help Strings
- ToolTips
- Context-sensitive help

1.1 Help Strings

A *help string* is a string that appears in the status bar of a window and describes the function of the field-level widget that has input focus. Help strings are a good way to provide cursory information about a database field or a field-level widget in your application. You define help strings for database fields in the field schema (using the Data Dictionary) and for field-level widgets in the appropriate property sheet (using the AppBuilder), or in 4GL code written in the Procedure Editor.

Progress provides the ability to associate help strings with database fields and those field-level widgets that can receive input focus. Help strings are easy to implement and produce a first line of help for your application. When you tab through certain field-level widgets (fill-in fields, buttons, combo boxes, selection lists, editors, toggle boxes, radio sets, and sliders) on an application interface, Progress displays a help string in the status area of the current window or screen as shown in Figure 1–1.

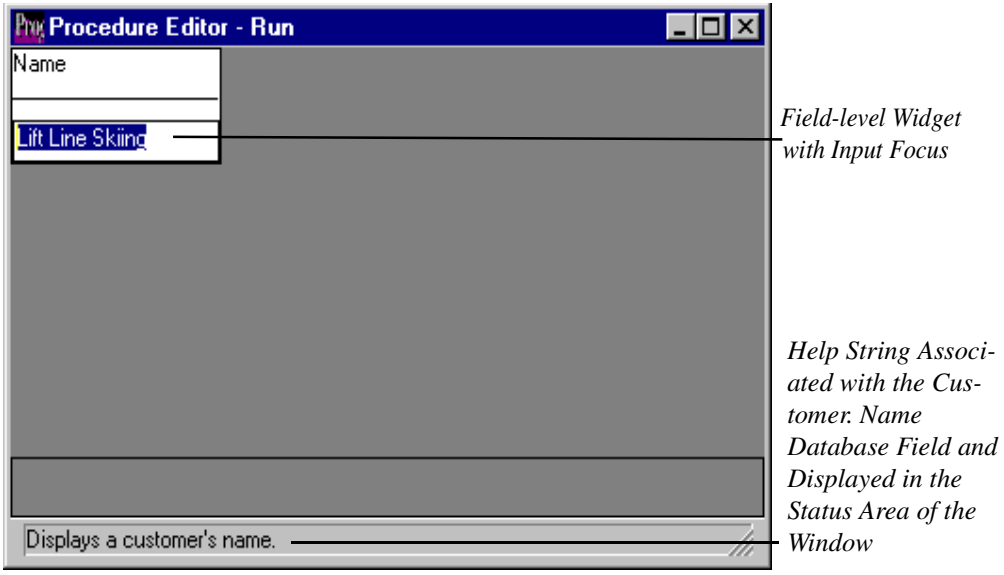


Figure 1–1: Help String for a Field-level Widget

A *status area* is an optional feature of a Progress application window that displays one line of message text at the bottom of the window. Its appearance is controlled by the STATUS-AREA attribute. Thus, if the STATUS-AREA is disabled on a window, no help message will appear. While a dialog box does not have a status area, help strings defined for fields on a dialog box are displayed in the status area of its parent window.

There are three methods for creating help strings:

- Defining help strings for database fields in the schema definition
- Creating help strings for field-level widgets with 4GL statements
- Creating help strings for field-level widgets in the Advanced Properties dialog box in the AppBuilder

The next three sections discuss these methods for creating help strings.

1.1.1 Associating Help Strings with Database Fields

By default, a data representation widget associated with a database field displays the help string defined for that database field. You can define and store a default help string as part of the schema definition of a database field using the Data Dictionary, as shown in Figure 1–2.

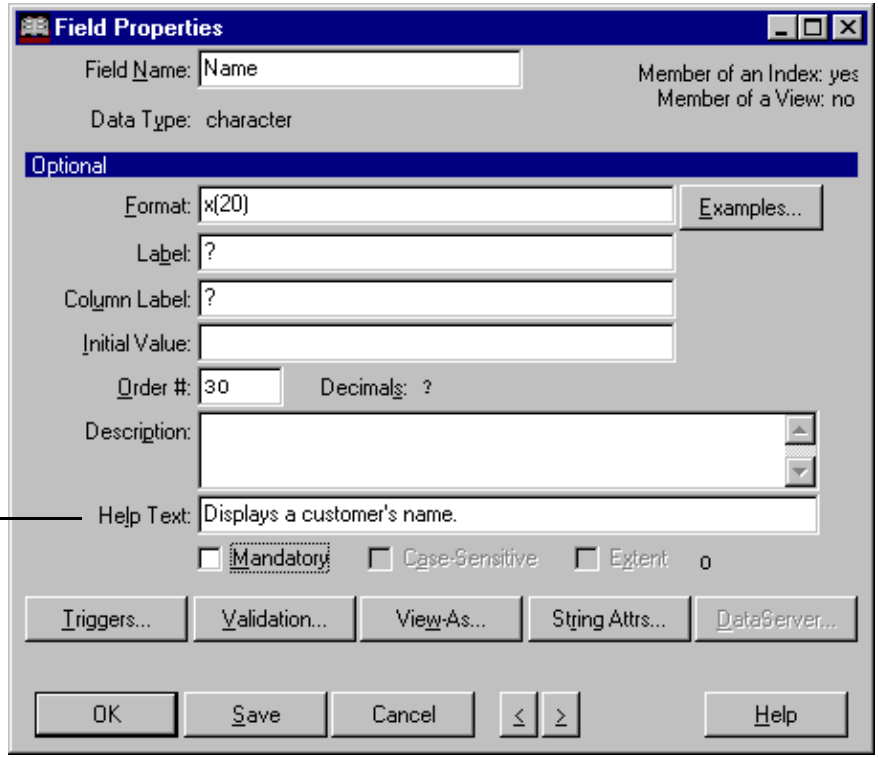


Figure 1–2: Defining a Help String in the Field Schema Definition

Creating help strings in the database schema provides a centralized location for help strings and makes them easy to maintain.

1.1.2 Specifying Help Strings with the Progress 4GL

The alternative to associating help strings with database fields is to define help strings as a part of field-level widgets that contain the data. To attach a help string to a field-level widget, use the HELP option of the Format phrase specified with the following 4GL statements: DEFINE FRAME, ENABLE, FORM, PROMPT-FOR, SET, or UPDATE. The following code shows an example using the DEFINE FRAME statement:

```
DEFINE FRAME FRAME-A
    BUTTON-1 AT ROW 4 COL 12
      HELP "Choose to cancel the operation and exit."
    WITH 1 DOWN NO-BOX OVERLAY SIDE-LABELS
      AT COL 1 ROW 1
      SIZE 74 BY 11.
```

In Progress, the Format phrase HELP option is the only way to define help strings for widgets not associated with database fields. Help strings defined with the HELP option override any help strings specified for associated database fields in the Data Dictionary. For more information on the HELP option, see the Format Phrase reference entry in the *Progress Language Reference*.

1.1.3 Defining Help Strings with the AppBuilder

Within the AppBuilder, you can specify a help string for a field-level widget in its Advanced Properties dialog box. Figure 1–3 shows a help string defined in the Advanced Properties dialog box for BUTTON-1.

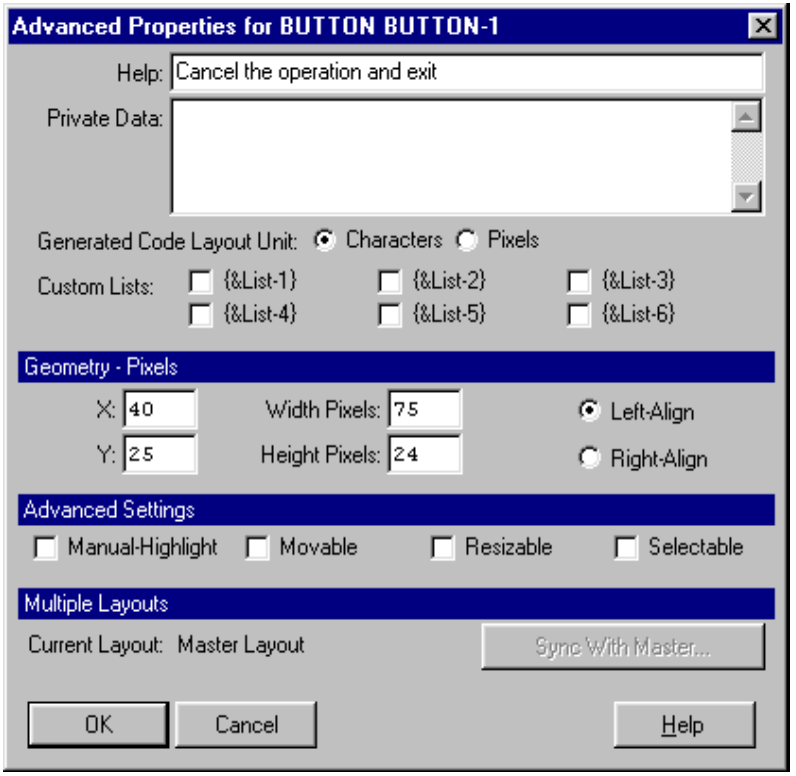


Figure 1–3: Defining a Help String in the Advanced Properties Dialog Box

The help string becomes part of a HELP option of a FORMAT phrase in the DEFINE FRAME statement generated by the AppBuilder.

1.2 ToolTips

A *ToolTip* is a short string that appears in an enclosing rectangle when the mouse pointer pauses over a field-level widget. ToolTips are widely used in Windows applications. For example, ToolTips are often used to provide labels for toolbar buttons.

You can assign a ToolTip to any Progress field-level widget such as a button, combo-box, editor, fill-in, image, radio-set, selection-list, slider, text, and toggle-box.

1.2.1 Implementing ToolTips with the Progress 4GL

To implement ToolTips, you specify the ToolTip text in the TOOLTIP attribute of the associated widget and set the TOOLTIPS attribute to TRUE for a SESSION system handle.

TOOLTIP Attribute

Each widget for which the TOOLTIP option is implemented has a run-time attribute, called TOOLTIP, established with read/write capabilities. If the TOOLTIP attribute is set to "" or the unknown value (?), then no ToolTip is displayed for that widget. The default is to not have a ToolTip. You can add a ToolTip to a widget at any time.

The following syntax is for the TOOLTIP Attribute:

SYNTAX

```
[ TOOLTIP tooltip ]
```

`tooltip`

A quoted string containing the text that displays when the mouse pointer pauses over the widget.

The following code example shows how to specify a ToolTip at run time:

```
Btn_OK:TOOLTIP = "Select this button to accept the information."
```

You can use the TOOLTIP option with the following 4GL elements:

- DEFINE BROWSE statement
- DEFINE BUTTON statement
- DEFINE IMAGE statement
- DEFINE RECTANGLE statement
- VIEW-AS phrase for combo-box, editor, fill-in, radio-set, selection-list, slider, text, and toggle-box widgets

TOOLTIPS Attribute

There is a session attribute called TOOLTIPS. The session default setting for TOOLTIPS is on (TRUE). To turn TOOLTIPS off for the session, set the TOOLTIPS session attribute to FALSE, for example:

```
SESSION:TOOLTIPS=FALSE
```

1.2.2 Implementing ToolTips with the AppBuilder

The AppBuilder allows you to define ToolTips by entering ToolTip text in an object's property sheet. To do this, first select a field-level widget (basic object) in the design window, then click the Object Properties button to display its property sheet. Enter the text into the ToolTip fill-in.

Figure 1–4 shows the Property Sheet for a button, indicating where you enter the ToolTip text.

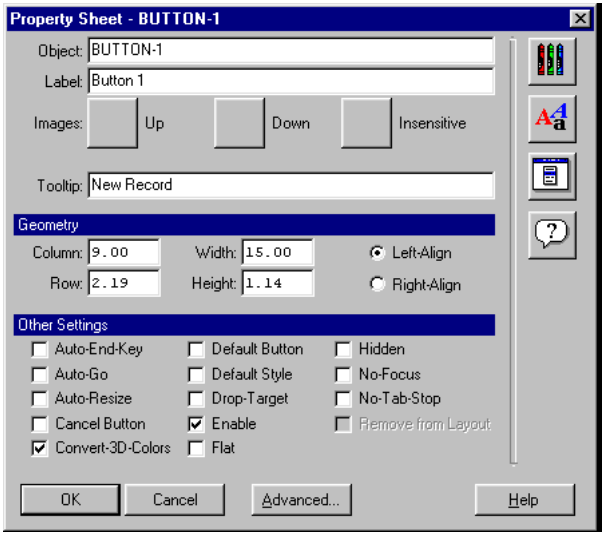


Figure 1–4: Defining a ToolTip in the Property Sheet

Figure 1–5 shows the appearance of the ToolTip when it is invoked from the running application.

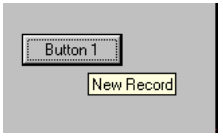


Figure 1–5: Run-time Appearance of a ToolTip

1.3 Overview of Online Help Systems

While help strings and ToolTips provide a first line of information for end users, the information they provide is limited. A help system can provide much larger chunks of information to the end user with a much more sophisticated delivery mechanism.

In a *help system*, help information is stored in a help file that is external to the application's source code. A help author creates the help file, which is divided into chunks of information called help topics. An application programmer uses the SYSTEM-HELP statement to direct Windows to either display a specific help topic in a help viewer window, or allow the end user to navigate through the help topics in a help file by displaying a Help Topics dialog box.

While a help system should appear to end users as a seamless part of a Progress application, the 4GL's SYSTEM-HELP statement actually calls a separate Windows application, `winhlp32.exe`.

Because the help information is stored in an external help file or files, a help system requires a coordinated effort of information design and application programming. The help author codes, compiles, and tests the help file(s), and the application programmer adds the appropriate Progress 4GL code to the application to invoke the help viewer and display the appropriate help topic when the end user requests help.

A help system consists of three elements:

- **Help information**—to assist users understand and use an application. The help information for an application resides in one or more help files and is displayed in chunks or units of information called *help topics*. Each help topic contains information about a single subject. For example, a help topic might describe a dialog box in your application, define a term, give instructions on how to perform a task, or describe a 4GL language element. Each help topic is associated with a unique identifier, called a *topic ID*.

  The help files are binary files, each of which is generated by compiling a set of help source files.

- **Help engine**—the executable file, `winhlp32.exe`, displays help topics in a help viewer and allows the user to navigate among help topics in a help file. The help engine reads help files and displays help topics in various types of help windows that are defined by the help file. A help window displays a single help topic at a time, while providing features that allow users to navigate between help topics in the help file.

  The help engine also displays and populates the Help Topics dialog box with information that helps the end user to navigate the help file.

- **Help calling interface**—the code in an application that allows users to access help information. It calls the help viewer and determines which help topic (if any) to display. Users can request and receive help information using a help keystroke, a help button, or a help menu.

From the end user's perspective, there are two general ways to access information in a help system:

- **Context sensitivity**—*Context-sensitive help* is information that is primarily intended to be accessed "on demand." Typically, the end user clicks a Help button in a dialog box, presses a help key (usually **F1**), or clicks the question mark icon to access a help topic that explains the current status or user interface of an application.

- **Navigation**—A help file can be opened in such a way that the end user is not immediately presented with a specific help topic. For example, the Help Topics item on the Help menu of an application window opens a Help Topics dialog box that allows the end user to access help topics in several different ways.

  Help topics containing conceptual information or reference information are typically accessible only through navigation.

  Individual help topics can also contain links to other topics that provide related information in the form of definitions and other help topics. Also, help viewer windows usually implement features that allow end users to navigate among help topics in various ways.

1.3.1 The SYSTEM-HELP Statement

A Progress application executes the Windows help engine (`Winhlp32.exe`) using the SYSTEM-HELP statement. This is the syntax for the SYSTEM-HELP statement:

SYNTAX

```
SYSTEM-HELP file-string
  [ WINDOW-NAME window-name ]
  {  CONTEXT int-expr
   | CONTENTS
   | SET-CONTENTS int-expr
   | FINDER
   | CONTEXT-POPUP int-expr
   | KEY string
   | PARTIAL-KEY string
   | MULTIPLE-KEY char TEXT string
   | COMMAND string
   | POSITION X x Y y WIDTH dx HEIGHT dy
   | POSITION MAXIMIZE
   | FORCE-FILE
   | HELP
   | QUIT
  }
```

`file-string`

The *filestring* parameter is a character expression that specifies the pathname of a help file. Help files have a .hlp file extension.

`WINDOW-NAME` *window-name*

The *windowname* parameter is a character expression that evaluates to the primary or secondary window name as defined in the [WINDOWS] section of the help project file. If the window name is omitted, or if “main” is specified, the primary help window is used.

`CONTEXT` *intexpr*

Displays the help topic that the context ID identifies. You define context IDs in the [MAP] section of the help project file.

The *int-expr* parameter is the context ID for the help topic.

CONTENTS

Displays the help topic defined as the contents in the [OPTIONS] section of the help project file.

Progress supports this option for backward compatibility with Windows 3.x (16-bit) help files. When calling help files compiled with the Help Workshop (32-bit), you should provide a contents file (.cnt) and use the FINDER option.

SETCONTENTS *intexpr*

Dynamically re-maps the contents help topic from what is defined in the [OPTIONS] section of the help project file. When a CONTENTS call is made, the new contents help topic is displayed.

The *int-expr* parameter is the context ID for the new contents help topic.

Progress supports this option for backward compatibility only.

FINDER

Displays the Help Topics dialog box, which contains an Index tab, a Find tab, and optionally a Contents tab, with the most recently used tab displayed on top.

If a contents file (.cnt) is present when you initially call the Help Topics dialog box, then the Contents tab displays on top. If a contents file is not present, then the dialog box displays with only the Index and Find tabs, and the Index tab is on top.

CONTEXTPOPUP *intexpr*

Displays the help topic in a popup window that the context ID identifies. If there is a non-scrolling region in the topic, the Windows Help Viewer displays only that region. You pair context IDs and topic IDs in the [MAP] section of the help project file.

The *int-expr* parameter is the context ID of the help topic.

KEY *string*

Displays the help topic matching the string found in the index keyword list. If there is more than one match, it displays the first topic containing the keyword. If there is no match or the string is omitted, it displays the Help Topics dialog box with the Index tab on top.

The *string* parameter is a character expression that evaluates to a keyword for the desired help topic.

PARTIALKEY *string*

Displays the help topic matching the string found in the keyword list. If there is more than one match, no match, or if the string is omitted, it displays the Help Topics dialog box with the Index tab on top.

The *string* parameter is a character expression that evaluates to a partial key for the desired help topic.

MULTIPLEKEY *char* TEXT *string*

Displays the help topic matching a keyword from an alternate keyword table.

The *char* parameter is a character expression that evaluates to the single character keyword table identifier for the required table.

The *string* parameter is a character expression that evaluates to the keyword that is located in the keyword table.

COMMAND *string*

Executes a help macro if the help file is already open.

The *string* parameter is a character expression that evaluates to the help macro to execute.

POSITION X *x* Y *y* WIDTH *dx* HEIGHT *dy*

Displays and positions the help window as specified. (The units are pixels.)

The *x* parameter is an integer expression that specifies the x coordinate for the help window.

The *y* parameter is an integer expression that specifies the y coordinate for the help window.

The *dx* parameter is an integer expression that specifies the width of the help window.

The *dy* parameter is an integer expression that specifies the height of the help window.

POSITION MAXIMIZE

Displays and maximizes the help window.

FORCEFILE

Ensures that the correct help file is open and displayed.

HELP

Displays the How to Use Help file.

QUIT

Informs the help application that help is no longer required. If no other applications are using help, the operating system closes the help viewer.

1.3.2 Quitting Help When Exiting the Application

You can terminate the help viewer with the QUIT parameter. As a matter of sound programming practice, the Progress application should execute the following line of code when it terminates:

```
SYSTEM-HELP myhelp.hlp QUIT.
```

If the help engine is running when the user terminates the Progress application, then the help viewer also terminates.

1.3.3 Coding the Help Calling Interface

This section explains some SYSTEM-HELP statement calls that are commonly used in the help calling interfaces of Progress applications. For more information on the Progress language elements described in the following sections, see the *Progress Language Reference*.

You can run the sample procedure, `r-syshlp.p`, to execute the help calls explained in this section. This procedure demonstrates several features of the SYSTEM-HELP statement with the Procedure Editor help files (`editeng.hlp` and `editeng.cnt`). The user can select a button to demonstrate each of the following SYSTEM-HELP options:

- CONTENTS
- CONTEXT
- PARTIAL-KEY
- HELP
- QUIT

Note that PARTIAL-KEY invokes the Help Topics dialog box with the Index tab on top.

The source code example r-syshlp.p, is in the library, prodoc.pl, in \Program Files\Progress\src or in the \src directory where Progress is installed. Follow the directions in the Preface for extracting files from libraries.

To execute r-syshlp.p, follow these steps:

1 ♦ Copy editeng.hlp and editeng.cnt from \Program Files\Progress\prohelp to your Progress working directory (by default, C:\Progress\Wrk).

2 ♦ Open r-syshlp.p in the Procedure Editor.

3 ♦ Press **F2** to run the file.

When you click the buttons on the sample interface, r-syshlp.p calls the Procedure Editor help files, editeng.hlp and editeng.cnt:

r-syshlp.p *(1 of 2)*

```
/* r-syshlp.p */

DEFINE VAR helpfile as CHAR.

DEFINE BUTTON b_help LABEL "FINDER Call".
DEFINE BUTTON b_h6 LABEL "CONTENTS Call".
DEFINE BUTTON b_h5 LABEL "CONTEXT Call".
DEFINE BUTTON b_h4 LABEL "PARTIAL-KEY Call-".
DEFINE BUTTON b_h3 LABEL "PARTIAL-KEY Call-'Tools'".
DEFINE BUTTON b_h2 LABEL "PARTIAL-KEY Call-'Tools Menu'".
DEFINE BUTTON b_h1 LABEL "HELP Call".
DEFINE BUTTON b_quit LABEL "QUIT Call".

FORM
    skip(1) space(1) b_help space(1)
    skip(1) space(1) b_h6 space(1)
    skip(1) space(1) b_h5 space(1)
    skip(1) space(1) b_h4 space(1)
    skip(1) space(1) b_h3 space(1)
    skip(1) space(1) b_h2 space(1)
    skip(1) space(1) b_h1 space(1)
    skip(1) space(1) b_quit space(1)
    skip(1) WITH FRAME x.
ENABLE ALL WITH FRAME x.

helpfile = "editeng.hlp".

/* The FINDER call brings up the Help Topics dialog box in its most
   recent state. */
ON CHOOSE OF b_help IN FRAME x
DO:
    SYSTEM-HELP helpfile FINDER.
END.
/* The CONTENTS call displays the main contents help topic. This
   is for backward compatability with Windows 3.x. */
ON CHOOSE OF b_help IN FRAME x
DO:
    SYSTEM-HELP helpfile CONTENTS.
END.

/* The CONTEXT call displays the help topic associated with the
   specified context ID of a help topic (in this case, 49154). */
ON CHOOSE OF b_h5 IN FRAME x
DO:
    SYSTEM-HELP helpfile CONTEXT 49154.
END.
```

r-syshlp.p *(2 of 2)*

```
/* The PARTIAL-KEY call brings up the Help Topics dialog box with
   the Index tab on top. When the string parameter is empty or is
   omitted altogether, the fill-in at the top of the Index tab is
   left blank.*/
ON CHOOSE OF b_h4 IN FRAME x
DO:
    SYSTEM-HELP helpfile PARTIAL-KEY "".
END.

/* In a PARTIAL-KEY call where the string parameter does not exactly
   match an index keyword of any help topic, the fill-in at the top
   of the Index tab is populated with the string that is passed in,
   and no help topic is automatically displayed. */
ON CHOOSE OF b_h3 IN FRAME x
DO:
    SYSTEM-HELP helpfile PARTIAL-KEY "Tools".
END.

/* In a PARTIAL-KEY call where the string parameter exactly
   matches a unique index keyword of a help topic, the help engine
   automatically launches a help viewer window and displays
   the matching topic. */
ON CHOOSE OF b_h2 IN FRAME x
DO:
    SYSTEM-HELP helpfile PARTIAL-KEY "Tools Menu".
END.

/* The HELP call brings up the Help Topics dialog box for the help file
   that explains how to use Windows Help (winhlp32.hlp). */
ON CHOOSE OF b_h1 IN FRAME x
DO:
    SYSTEM-HELP helpfile HELP.
END.

/* The QUIT call causes the help engine to terminate, unless another
   application is using help. */
ON CHOOSE OF b_quit IN FRAME x
DO:
    SYSTEM-HELP helpfile QUIT.
    RETURN.
END.

WAIT-FOR GO OF FRAME x.
```

1.4 Accessing Online Help from the Menu Bar

A common method of requesting help from an application is by selecting help items from the application menu bar. The menu bar usually includes a help pull-down menu and contains menu items such as:

- **Help Topics** — Instructs the help engine to display the Help Topics dialog box for the specified help file.
- **About or Version** — Provides application copyright and version number information.

The CHOOSE event occurs when a user chooses an option from a menu. In Progress, you use a CHOOSE event to code a trigger for menu items. A *trigger* is a block of code that executes whenever a specific event occurs (such as when the user chooses a menu item).

The following code example demonstrates a help trigger that executes when the user selects a menu item from the application's menu bar to access the help engine:

```
ON CHOOSE OF MENU-ITEM menu_help_contents
    DO:
      SYSTEM-HELP myhelp.hlp CONTENTS.
    END.
```

1.4.1 Calling the Help Topics Dialog Box

Using the FINDER option of the SYSTEM-HELP statement displays the Help Topics dialog box for the specified help file. You frequently use this for a Help Topics call from the help menu. The following code example demonstrates a Help Topics call:

```
ON CHOOSE OF MENU-ITEM Help_Topics:
    DO:
      SYSTEM-HELP myhelp.hlp FINDER.
    END.
```

For example, when the user chooses the Help Topics from the Help menu in the Procedure Editor's main window, the Help Topics dialog box appears, as shown in Figure 1–6.

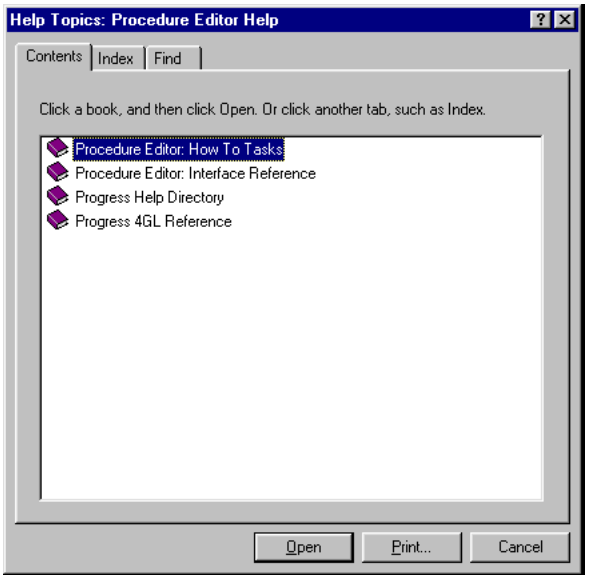


Figure 1–6: Help Topics Dialog Box for Procedure Editor Help

1.4.2 Calling the Index Function of the Help Topics Dialog Box

Like the Help Topics call, it is common to call the Index function of the Help Topics dialog box from the Help menu. The following pseudocode calls the Help Topics dialog box with the Index tab on top:

```
ON CHOOSE OF Button_Search IN FRAME frame_1
    DO:
      SYSTEM-HELP myhelp.hlp PARTIAL-KEY "".
    END.
```

Figure 1–7 shows the Help Topics dialog box for the Procedure Editor help file with the Index tab on top.

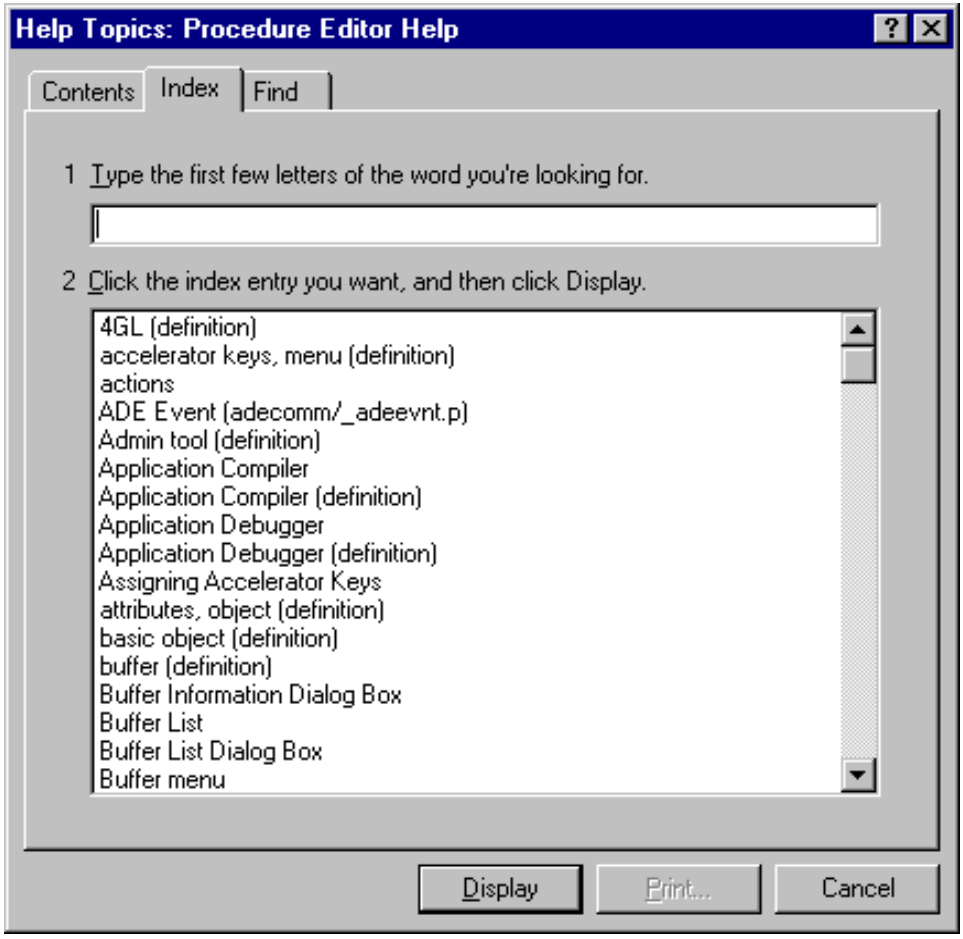


Figure 1–7: Help Topics Dialog Box with Index Tab on Top

1.4.3 Calling the How to Use Help File

Windows provides a help file with information about how to use the help engine itself (How to Use Help file, `\Program Files\Progress\prohelp\hoh.hlp`. for 16 bit applications or `winhlp32.hlp` for 32 bit applications).You call this How to Use Help file with the HELP parameter, as demonstrated in the following code example:

```
ON CHOOSE OF MENU-ITEM "How to Use Help":
    DO:
      SYSTEM-HELP myhelp.hlp HELP.
    END.
```

1.5 Accessing Online Help with a Help Button

Context-sensitive help should be made available when a user wants to learn about the purpose of a particular window, dialog box, or widget. You can code the Progress application to display a Help button, as shown in Figure 1–8. When the user chooses the button, the help viewer displays the help topic for the current window or dialog box.

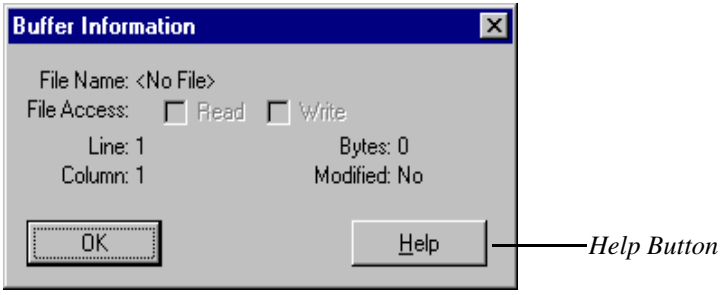


Figure 1–8: Dialog Box with a Help Button

For example, when the user chooses the Help button in the Procedure Editor Buffer Information dialog box, the user sees context-specific help for that dialog box, as shown in Figure 1–9.

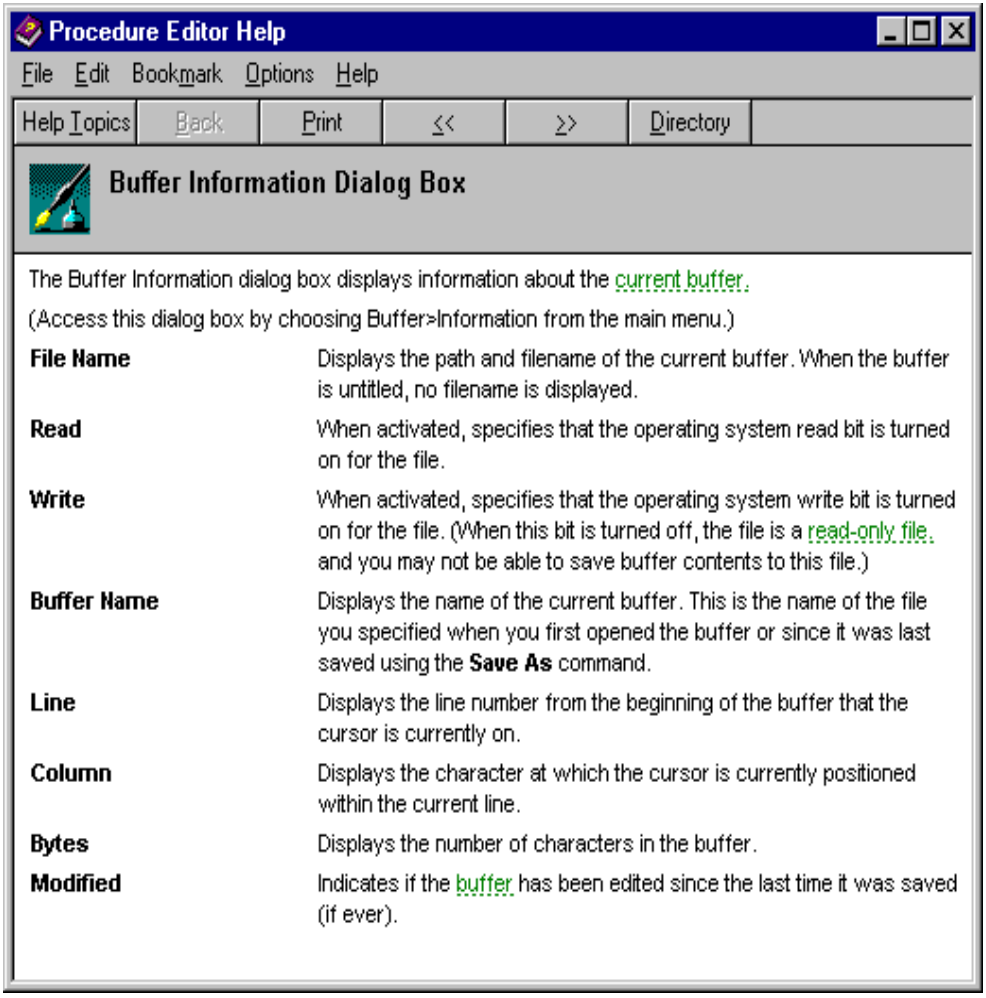


Figure 1–9: Context-sensitive Help for the Buffer Information Dialog Box

As with menu items, you attach triggers to buttons with the CHOOSE event. In this example, a help trigger is attached to the Help button. In the following code example, a help trigger executes when the user chooses a Help button to access help information from an application window or dialog box:

```
ON CHOOSE OF b_help IN FRAME x
    DO:
      SYSTEM-HELP myhelp.hlp CONTEXT 49154.
    END.
```

You can write a help trigger for a field-level widget, a frame, a dialog box, a window, or an application. The following code example demonstrates a help trigger that executes when the current application window has input focus and the user presses **HELP**:

```
ON HELP OF WINDOW-1:
    DO:
      SYSTEM-HELP myhelp.hlp CONTEXT 49154.
    END.
```

1.6 Accessing Online Help with a Help Key

When a user presses the designated help key (usually **F1** in Windows applications), a HELP event goes to the field-level widget with input focus in the current frame. If there is no trigger on the field-level widget, the HELP event goes to the current frame. The HELP event continues to move to the next level in the widget hierarchy as shown in Figure 1–10. If the help event does not find an associated widget, it runs `applhelp.p`, the Progress help calling interface. This procedure, `applhelp.p`, in `c:\program files\progress\src\`, displays the message, "No application help is available." A sample interface, `c:\program files\progress\src\samples\applhelp.p`, provides an example of how you can modify `applhelp.p` for your application.

Figure 1–10 depicts the behavior of the Progress HELP event.

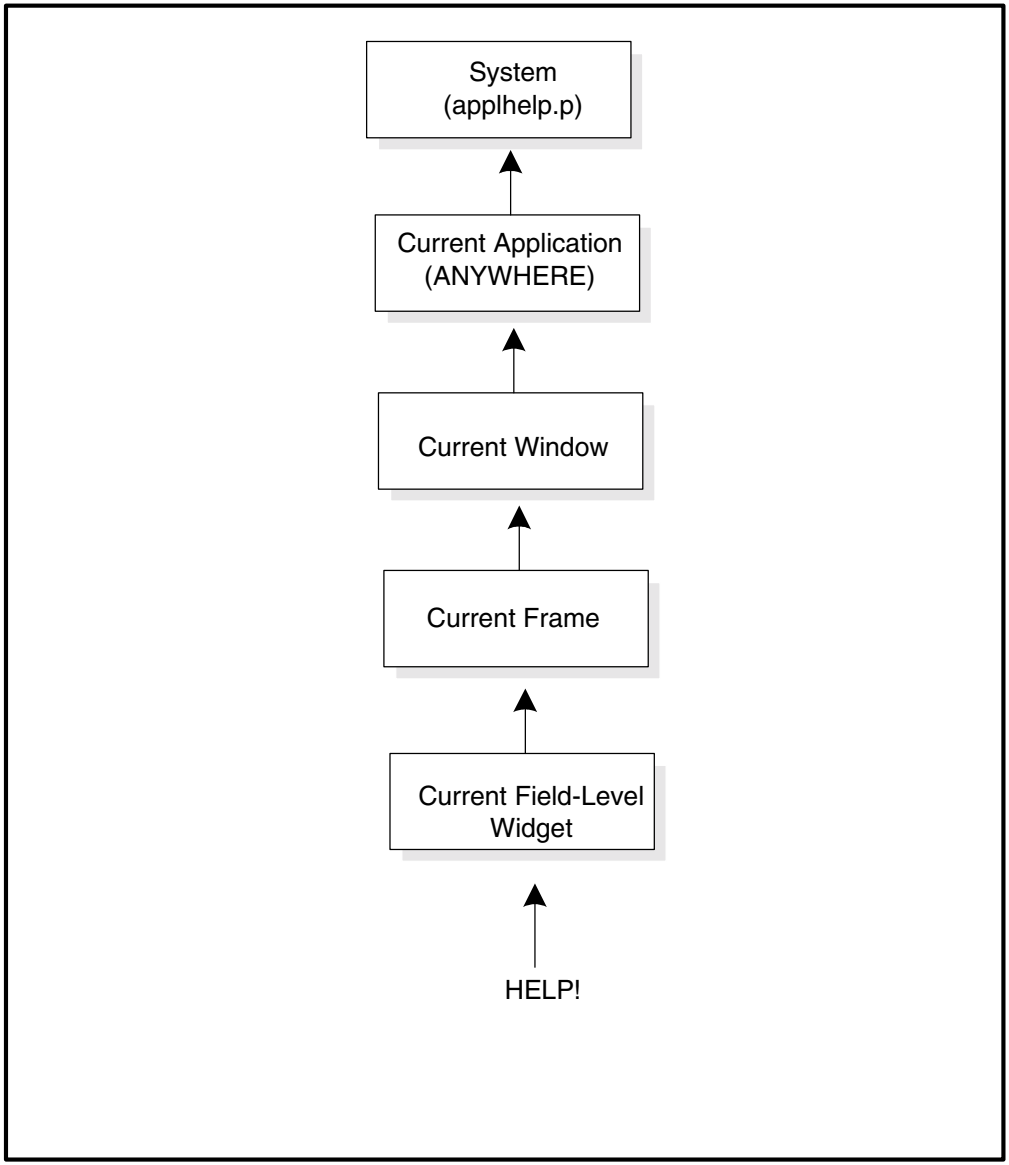


Figure 1–10: HELP Event Hierarchy

1.7 Context-sensitive Help

Progress supports Windows 95/98 and Windows NT context-sensitive help for dialog box and window widgets. When context-sensitive help is available in an application, a question mark icon appears in the title bar of the window or dialog box. If the user clicks on the question mark icon, the mouse cursor changes, as shown in Figure 1–11.



Figure 1–11: Mouse Cursor Showing Context-sensitivity Enabled.

If the user then clicks on a control, help text is displayed, as shown in Figure 1–12.



Figure 1–12: Context-sensitive Help Displayed.

1.7.1 Implementing Context-sensitive Help with the Progress 4GL

In the Progress 4GL, when the user clicks on the question mark icon to initiate a context-sensitive help operation, the following occurs:

- The mouse cursor changes to a pointer with a question mark. This cursor overrides any cursor that has been loaded by the LOAD-MOUSE-POINTER( ) method.
- If the user clicks outside of the dialog box or window, clicks within the dialog box or window but not on a field level widget, presses the **ESC** key, **CTRL-ESC**, or **ALT-TAB**, the operation is cancelled and the mouse cursor returns to its prior state.
- If the user clicks on a widget in the dialog box or window, the mouse cursor returns to its prior state and Progress displays the help topic by following these rules:
  - If the widget's CONTEXT-HELP-ID attribute has been set and a help file has been specified, the Progress 4GL calls the help system to display the help topic. If the help file does not contain an entry for the specified context ID, the help system displays the message "No Help topic is associated with this item." If the help file itself does not exist, the help system asks whether the user wants to try to locate it.
  - If the widget's HELP attribute contains a non-empty string, a pop-up window is displayed containing the help string.
  - Otherwise, the help system displays a pop-up window stating "No Help topic is associated with this item."

Context-sensitive help supports the following attributes, statements, and phrases:

- CONTEXT-HELP Attribute
- CONTEXT-HELP-FILE Attribute
- CONTEXT-HELP-ID Attribute
- DEFINE BROWSE Statement
- DEFINE BUTTON Statement
- DEFINE VARIABLE Statement
- Frame Phrase

CONTEXT-HELP Attribute

The CONTEXT-HELP attribute applies to window and dialog box widgets. It is a logical value. When CONTEXT-HELP is TRUE, a question mark icon displays in the title bar of the window or dialog box. CONTEXT-HELP is both readable and writable; however, it must be set before the window or dialog box is realized. The default value of CONTEXT-HELP is FALSE.

Microsoft Windows does not permit the question mark icon to appear, or it appears but does not function, when combined with these other attribute settings that affect a window's title bar:

- If CONTEXT-HELP = TRUE and SMALL-TITLE = TRUE, the question mark icon does not appear.
- If CONTEXT-HELP = TRUE and both MIN-BUTTON = TRUE and MAX-BUTTON = TRUE, the question mark icon does not appear.
- If CONTEXT-HELP = TRUE and either (but not both) of MIN-BUTTON or MAX-BUTTON = TRUE, the question mark icon appears but does not function.
- If CONTEXT-HELP = TRUE and CONTROL-BOX = FALSE, the question mark icon does not appear.

To summarize, you must set CONTEXT-HELP = TRUE, MIN-BUTTON = FALSE, and MAX-BUTTON = FALSE (leaving CONTROL-BOX and SMALL-TITLE at their default FALSE values) to successfully use this feature with a window widget.

NOTE: The preceding settings only apply to window widgets, not to dialog boxes. The question mark icon always functions correctly when used with a dialog box.

CONTEXT-HELP-FILE Attribute

The CONTEXT-HELP-FILE attribute specifies the path of a help file (.hlp) associated with a dialog box, window, or session. (The full path of the help file is required; Progress does not search for the help file.) This attribute is a character value and is readable and writable. It applies to the dialog box widget, window widget, and SESSION handle.

If the CONTEXT-HELP-FILE is not specified (is UNKNOWN) for a dialog box, the dialog box inherits the help file of its parent window. If the parent window's CONTEXT-HELP-FILE is also UNKNOWN, it inherits the session's help file (specified by SESSION:CONTEXT-HELP-FILE).

CONTEXT-HELP-ID Attribute

The CONTEXT-HELP-ID attribute specifies the context ID of a help topic in a help file. It is an integer value and is readable and writable. It applies to the following widgets:browse, button, combo-box, control frame, editor, fill-in, radio-set, selection-list, slider, and toggle box.

DEFINE BROWSE Statement

When the CONTEXT-HELP-ID attribute is used with the DEFINE BROWSE statement, it specifies the context ID of a help topic for this browse. The help file is specified at the session, window, or dialog box level using the CONTEXT-HELP-FILE attribute. The syntax for the DEFINE BROWSE statement follows:

SYNTAX

```
DEFINE [ [ NEW ] SHARED ] BROWSE browse-name
.
.
.
[ CONTEXT-HELP-ID expression ]
.
.
.
```

DEFINE BUTTON Statement

When the CONTEXT-HELP-ID attribute is used with the DEFINE BUTTON statement it specifies the context ID of a help topic for this button. The help file is specified at the session, window, or dialog box level using the CONTEXT-HELP-FILE attribute. The syntax for the DEFINE BUTTON statement follows:

SYNTAX

```
DEFINE BUTTON button
.
.
.
[ CONTEXT-HELP-ID expression ]
.
.
.
```

DEFINE VARIABLE Statement

When the CONTEXT-HELP-ID attribute is used with the DEFINE VARIABLE statement, it specifies the context ID of a help topic for this variable. The help file is specified at the session, window, or dialog box level using the CONTEXT-HELP-FILE attribute. The syntax for the DEFINE VARIABLE statement follows:

SYNTAX

```
DEFINE [ [ NEW [ GLOBAL ] ] SHARED ] VARIABLE variable
.
.
.
[ CONTEXT-HELP-ID expression ]
.
.
.
```

Frame Phrase

When the CONTEXT-HELP attribute of the Frame Phrase is used with the DEFINE FRAME and FORM statements, among others, it specifies that context-sensitive help is available for that frame.

When the CONTEXT-HELP-FILE attribute of the Frame Phrase is used with the DEFINE FRAME and FORM statements, among others, it specifies the complete path of a help file associated with that frame. If CONTEXT-HELP-FILE is specified without CONTEXT-HELP, CONTEXT-HELP is assumed. This behavior can be overridden by setting the dialog box's CONTEXT-HELP attribute to FALSE at run time. The syntax for the Frame Phrase follows:

SYNTAX

```
WITH
.
.
.
[ CONTEXT-HELP ] [ CONTEXT-HELP-FILE help-file-name ]
.
.
.
```

Example

The following example displays a dialog box with two fill-ins and two buttons. It demonstrates several ways help can be accessed:

- The first fill-in (A) has a HELP string defined in this procedure.
- The second fill-in (customer.credit-limit) inherits its HELP string from the database schema.
- The button labeled “Help Available” has CONTEXT-HELP-ID specified.
- The button labeled “No Help Available” has neither a HELP string nor CONTEXT-HELP-ID specified.

```
DEFINE VARIABLE a AS CHARACTER LABEL "Fill-in A".
DEFINE BUTTON b1 LABEL "Help Available" CONTEXT-HELP-ID 110.
DEFINE BUTTON b2 LABEL "No Help Available".
DEFINE FRAME f1
      a HELP "Help for Fill-in A"
      customer.credit-limit SKIP
      b1
      b2
      WITH SIDE-LABELS THREE-D VIEW-AS DIALOG-BOX
        TITLE "Context-sensitive help example"
        CONTEXT-HELP CONTEXT-HELP-FILE "c:\dlc\prohelp\uibeng.hlp".
ENABLE ALL WITH FRAME f1.
WAIT-FOR WINDOW-CLOSE OF FRAME f1.
```

1.7.2 Implementing Context-sensitive Help with the AppBuilder

The AppBuilder allows you to implement context-sensitive help in an object’s property sheet. The Context-Sensitive Help Section of the Property Sheet contains a check box labeled Context Help. Check this box to enable context-sensitive help, then enter the path and help file name in the Help File field (or click Browse to select the help file).

Figure 1–13 shows a property sheet for a window with context-sensitivity enabled and a help file specified.

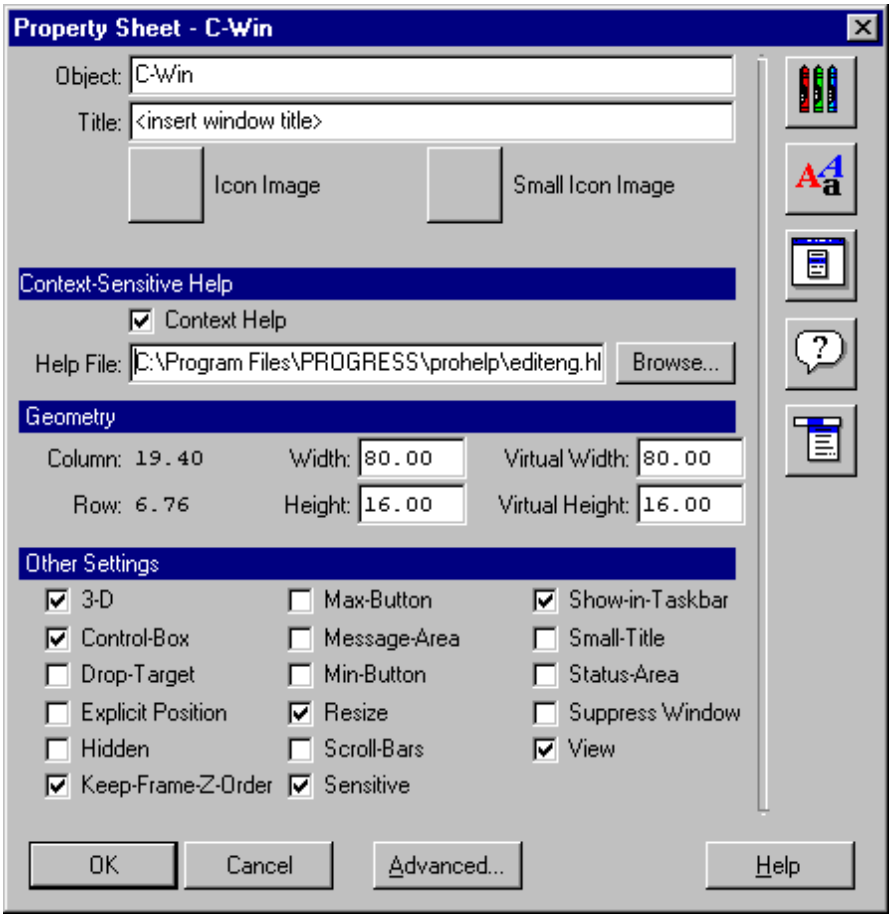


Figure 1–13: Property Sheet for a Window with Context-sensitive Help

As explained in the CONTEXT-HELP Attribute section, to enable context-sensitive help for windows and avoid known Windows bugs, you cannot set SMALL-TITLE, MIN-BUTTON, or MAX-BUTTON to TRUE, or CONTROL-BOX to FALSE.

The browse, button, combo-box, control frame, database field, editor, fill-in, radio-set, selection-list, slider, and toggle-box widgets inherit the help file specified in the window or dialog box. To implement context-sensitive help for one of these widgets, enter the Context ID (numeric) for the appropriate help topic in the widget's property sheet. Figure 1–14 shows a property sheet with the topic ID for a help topic specified.

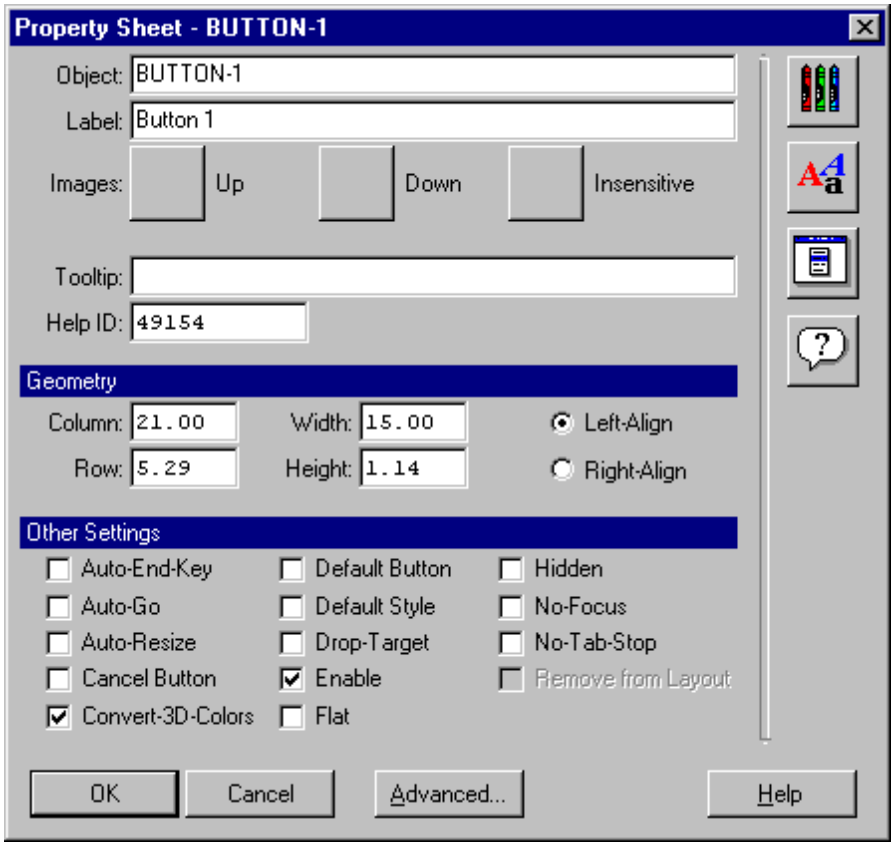


Figure 1–14: Property Sheet with a Help Topic Context ID Specified

2

Overview of Help System Development

This chapter reviews some of the most useful features of help viewers and the Help Topics dialog box. This chapter also discusses how to design a help system. This discussion is presented mainly from the perspective of the help author. The process of integrating a help system with a Progress application is described in Chapter 1, "Providing Help for Progress Applications."

This chapter includes the following sections:

- Basic features of help systems and development considerations
- Designing a help system

2.1 Basic Features of Help Systems and Development Considerations

The sections that follow explain how to use some of the major help viewer menu commands and functions. You can re-create the examples shown in these sections by following these steps:

1 ♦ Start the Progress Procedure Editor.

2 ♦ Invoke the Help Topics dialog box by choosing Help→ Help Topics.

2.1.1 Help Topics and the Help Topics Dialog Box

The basic building block of a help system is the help topic, which is a concise, modular piece of information about a single subject. A help window displays one help topic at a time. Examples of help topics include: a description of a dialog box in your application, a definition of a term, a set of instructions to complete a task, and a description of a 4GL language element.

The Help Topics dialog box helps users to navigate in a help system. It appears when a user chooses Help→ HelpTopics. The Help Topics dialog box appears in the form of a tab folder with three tabs: Contents, Index, and Find. The next three sections discuss the functions of these three tabs. Also, by convention, each help viewer window contains a Help Topics button, allowing the user to return directly to the Help Topics dialog box from any help topic.

The Contents Tab

Figure 2–1 shows the Help Topics dialog box for the Procedure Editor help system with the Contents tab on top. This is how it appears the first time it is called from the Help→ HelpTopics menu item.

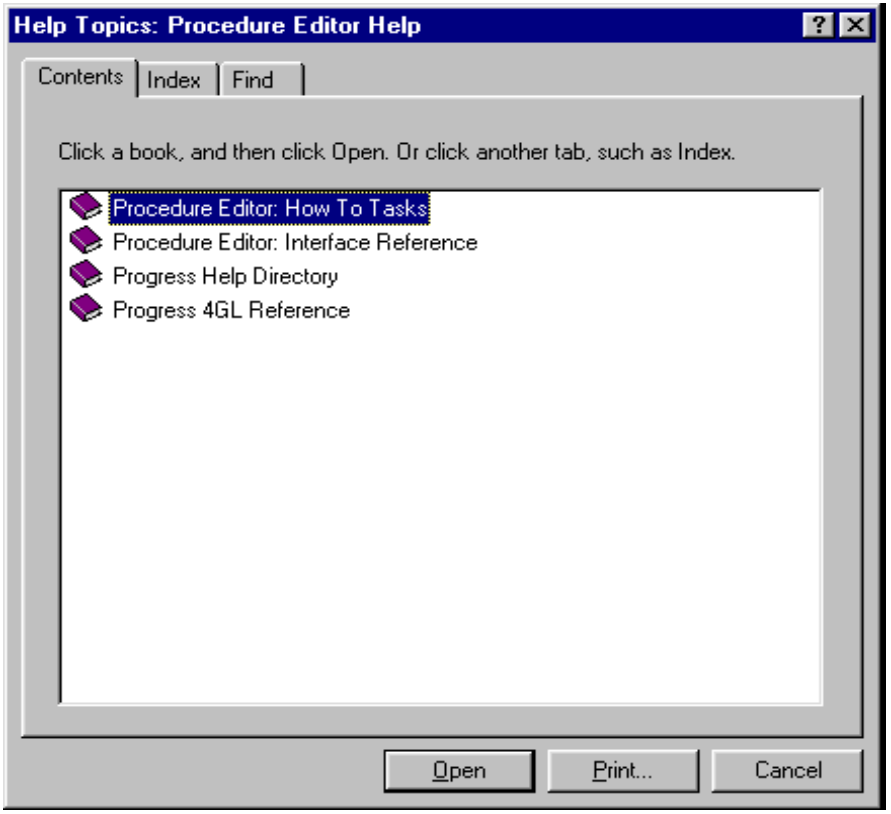


Figure 2–1: Help Topics Dialog Box with Contents Tab on Top

The Contents tab is populated by two types of icons: topic icons and book icons. Each topic icon represents a jump to a specific help topic in the help file. This means that when you double-click a topic icon, the help engine launches a help viewer window and displays the specified topic in it. Alternatively, topic icons can be used to jump to a help topic or the Help Topics dialog box in another help file.

Book icons are used to organize the Contents tab. Book icons do not cause you to jump anywhere outside the Contents tab of the Help Topics dialog box. They can contain only other topic or book icons. When you double-click a book icon, the display on the Contents tab expands to show the icons that book contains.

The appearance of the Contents tab is determined by a file known as the contents file. A contents file has the same name as the help file with which it is associated, except that the contents file has a .cnt extension. Unlike other help source files that are compiled into a single binary help file (.hlp), the contents file remains separate from the compiled help file, and must be present in the same directory as the compiled help file. If the help engine does not find a contents file, it displays the Help Topics dialog box with only the Index and Find tabs. For detailed instructions on creating contents files, see Chapter 4, "Completing Help Systems."

The Index Tab

Figure 2–2 shows the Index tab of the Help Topics dialog box for the Procedure Editor help system. Certain SYSTEM-HELP calls cause the Help Topics dialog box to appear with the Index tab on top. (See Chapter 1, "Providing Help for Progress Applications.")

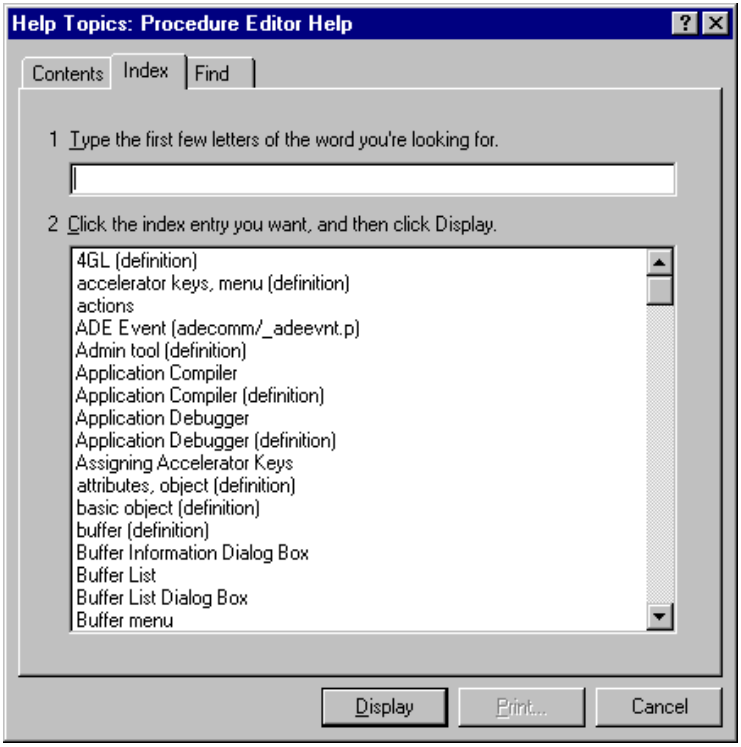


Figure 2–2: Help Topics Dialog Box with Index Tab on Top

The Index tab in a help system is similar to an index at the back of a book. The Index tab displays an alphabetized list of all the index keywords in the help file. When you double-click on an entry in the Index tab, you jump to the help topic associated with that keyword or to the Topics Found dialog box with a list of topics associated with that keyword As the help author, you associate

index keywords with help topics. See Chapter 3, "Creating Help Topic Files," for detailed information on creating index keywords.

The Find Tab

Figure 2–3 shows the Find tab of the Help Topics dialog box for the Procedure Editor help system. The Find tab comes to the top only when you click on it.

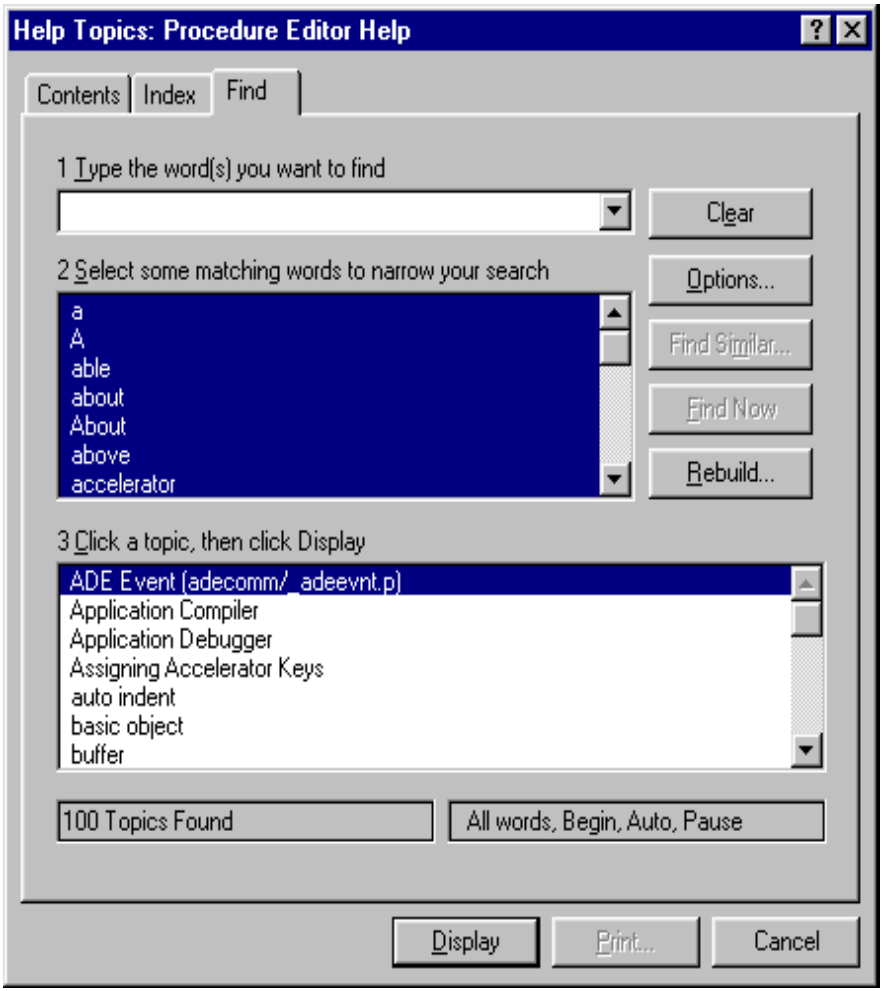


Figure 2–3: Help Topics Dialog Box with Find Tab on Top

The Find tab provides full-text search capability. When a help file is accessed for the first time, the help engine reads through the file and creates an alphabetized list of every word in the file. This list is stored in a file with an .fts extension in the same directory as the .hlp file and the .cnt file. You do not need to do anything as the help author to create the .fts file.

2.1.2 Help Windows

You can define more than one type of help window for a help file. The help engine recognizes two broad categories of window types: main and secondary. By default, there is only one main window and it must have the name, “main.” However, you can use several secondary window types, or none at all, depending on the types of information you need to present in help topics.

It is helpful to think of window types as corresponding to different types of information, so that end users will associate each window type (distinguished by size, location, color scheme, and/or viewer features) with its particular type of information. You should define and use secondary window types sparingly.

All the help files for the Progress ADE tools (including the Procedure Editor help file) use only a main window. In the Progress help files, this main window appears as a relatively large window centered on the screen.

The next two sections present the features of a main window and a secondary window as they might be implemented for the Progress Procedure Editor help system. You can also customize the menus and the buttons that appear in any help window, as well as its size and placement and other properties. You do this through entries in the help project file and by embedding macros in the project file or help source files. See Chapter 4, “Completing Help Systems,” for a more detailed discussion of the methods for customizing help windows.

The Main Window

Figure 2–4 shows the main window for the Procedure Editor help file. The menus and buttons shown are standard for Windows help and appear in the main window for almost all help files.

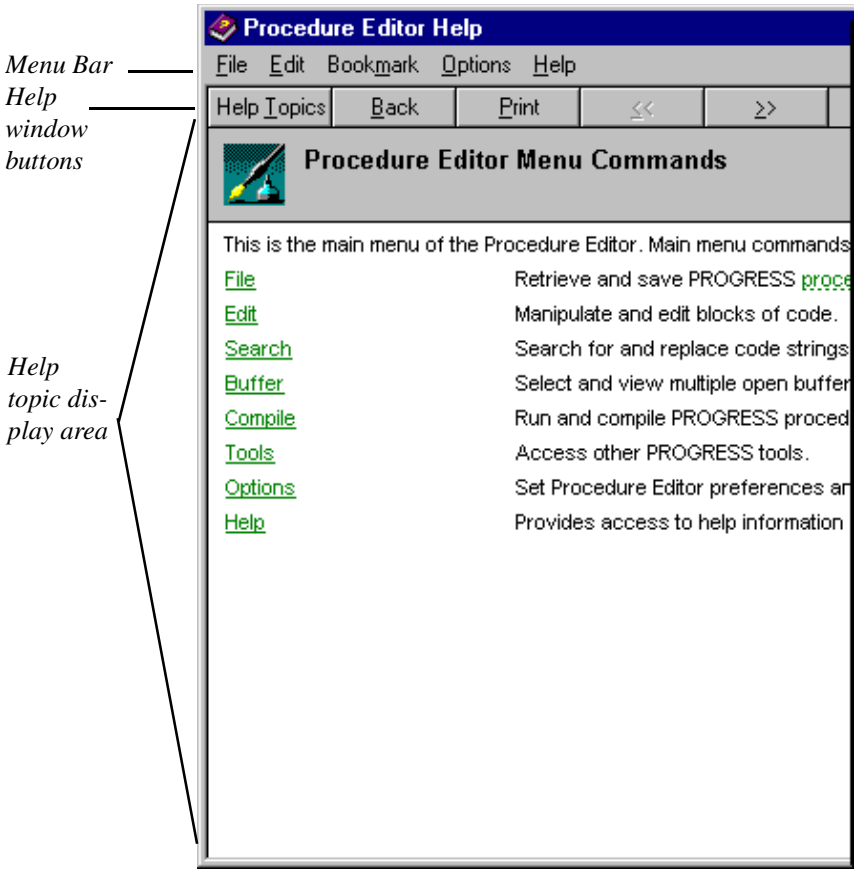


Figure 2–4: Main Window for the Procedure Editor Help File

Table 2–1 summarizes the standard main window menus. Also note that if you right-click anywhere in the help topic display area, a popup menu appears that duplicates some of the commands from the main window menus.

Table 2–1: Main Window Menus

Menu	Purpose
File	Opens a help file and prints help topics.
Edit	Copies or annotates a help topic. An annotated topic displays a paper clip icon that you can select to view the annotation.
Bookmark	Marks a help topic so you can access it directly from the Bookmark menu.
Options	Sets any of several options that affect the display properties of the window.
Help	Accesses version information about the help engine and the help file.

Table 2–2 summarizes the standard main window buttons.

Table 2–2: Main Window Buttons

Function	Purpose
Help Topics	Displays the Help Topics dialog box for the current help file.
Back	Displays the most recently viewed help topic.
Print	Displays the Print dialog box, allowing the user to print the current topic.
<<	Moves backward in a browse sequence.
>>	Moves forward in a browse sequence.

Secondary Windows

Figure 2–5 shows how a secondary window could appear in the Procedure Editor help file. The buttons shown are standard for Windows help and appear in the secondary window for almost all help files.

A secondary window, by definition, does not have a menu bar. However, the Options button that appears in a secondary window displays a popup menu that includes some of the most frequently used commands in the main window's menus. The Options button in the secondary window performs the same function as right-clicking in the display area of a window.

Secondary windows offer the advantage that a help author can specify that they are "sized to fit." This type of secondary window sizes itself to fit its content. Also, multiple secondary windows (up to nine) can remain open at the same time.

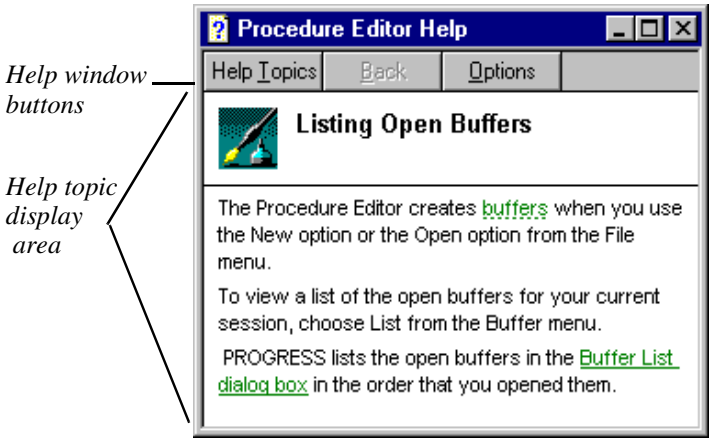


Figure 2–5: A Secondary Window

Table 2–3 summarizes the standard secondary window buttons.

Table 2–3: Secondary Window Buttons

Function	Purpose
Help Topics	Displays the Help Topics dialog box for the current help file.
Back	Displays the most recently viewed help topic.
Options	Displays a popup menu that duplicates some of the commands from the main window menus. This menu is the same as the menu that appears when you right-click in the display area of either a main or secondary window.

2.1.3 Hypertext Links

Hypertext links connect help information in a nonlinear fashion, causing the help engine to display a related topic when the user selects the link. For detailed instructions on creating hypertext links, see the "Adding Hypertext Links" section in Chapter 3, "Creating Help Topic Files."

Figure 2–6 shows examples of two basic link types-jumps and popups. The following sections provide more detail on various types of links.

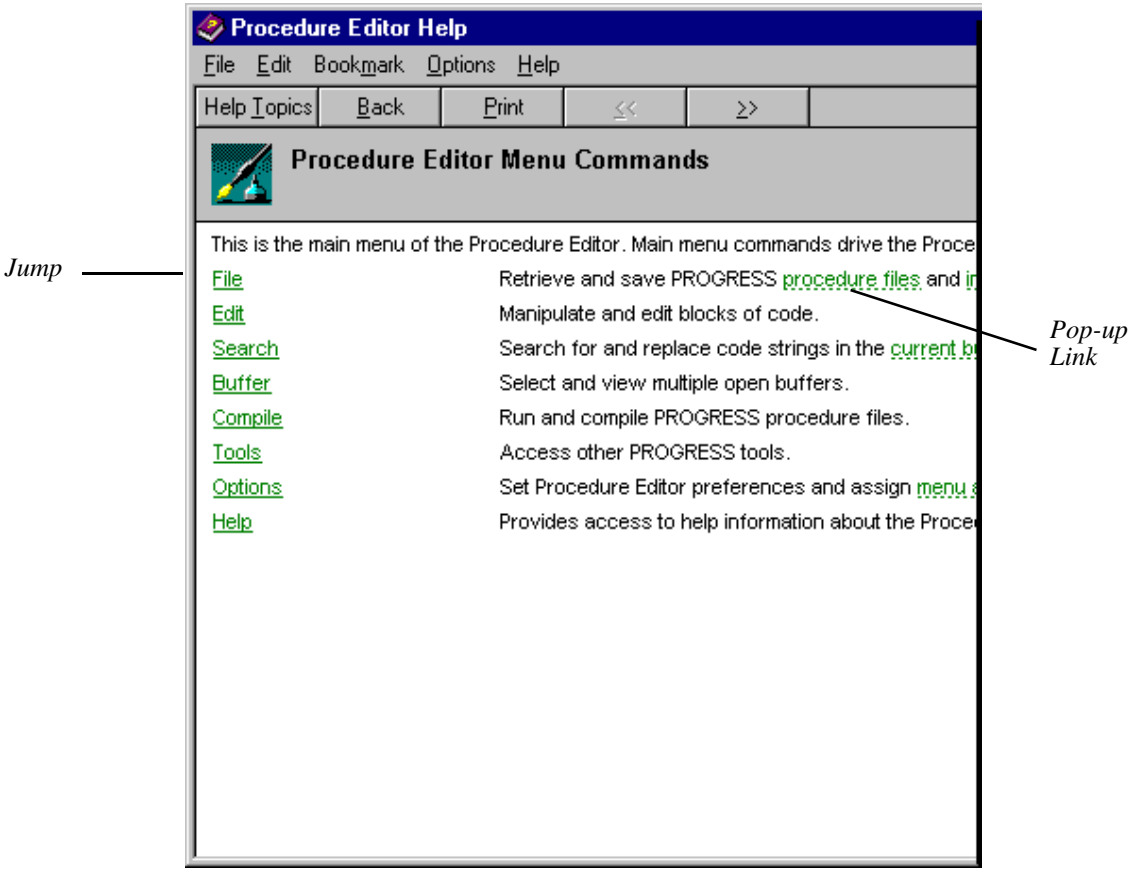


Figure 2–6: Hypertext Link Examples

Jumps

A *jump* appears with a solid underline in the help window. Jumps work like automatic cross-references; if you select a jump, the help engine immediately displays the linked help topic. In the Procedure Editor help file (Figure 2–6), the word "File" is single underlined, indicating that it is a jump. When you select it, the help engine displays the topic shown in Figure 2–7.

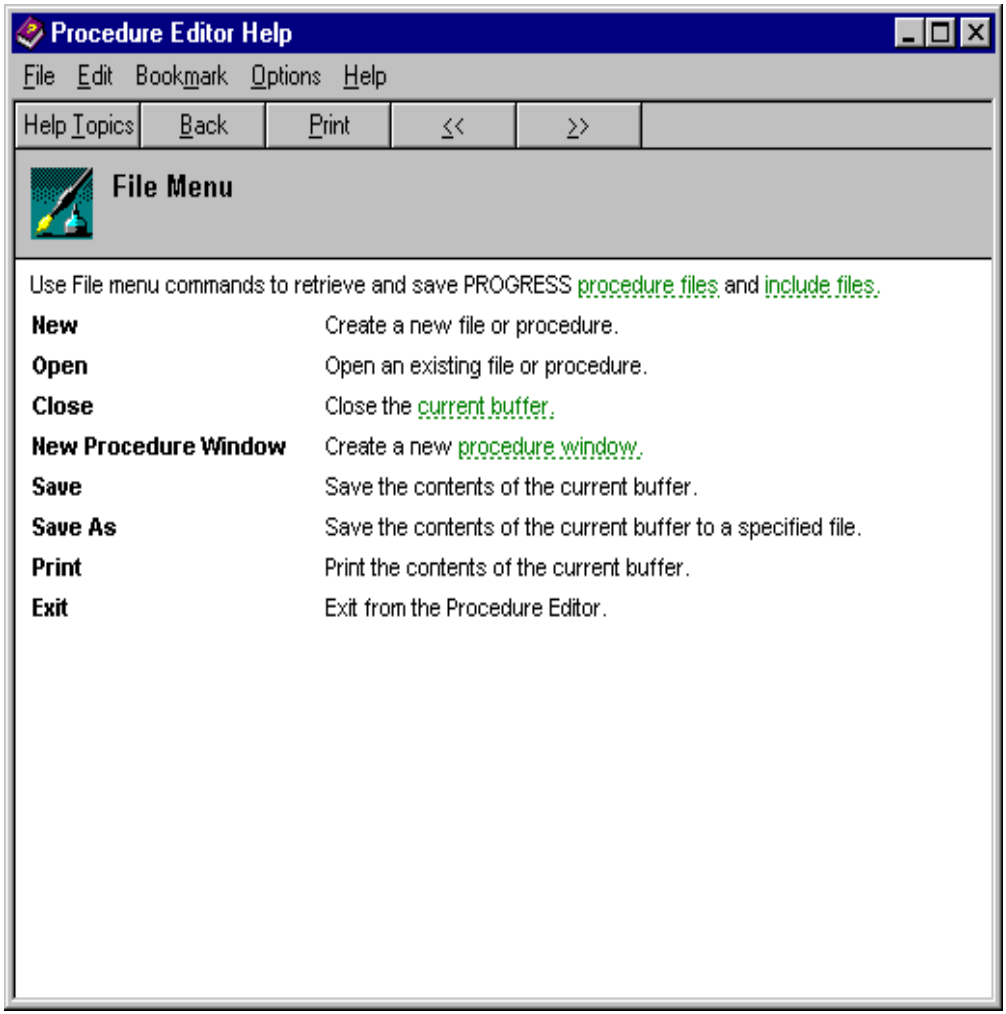


Figure 2–7: Destination Topic for "File" Jump

For more information about jumps, see the "Creating Jumps" section in Chapter 3, "Creating Help Topic Files."

Popups

A *popup link* appears with a dotted underline in the help window. If you select a popup link, text or a graphic displays in a popup help window overlaying the window containing the link. In the Procedure Editor help file, when you choose the text, "procedure files" (Figure 2–6), the help viewer displays a definition for procedure files in a popup window, as shown in Figure 2–8. To close the popup window, you click anywhere on the screen.

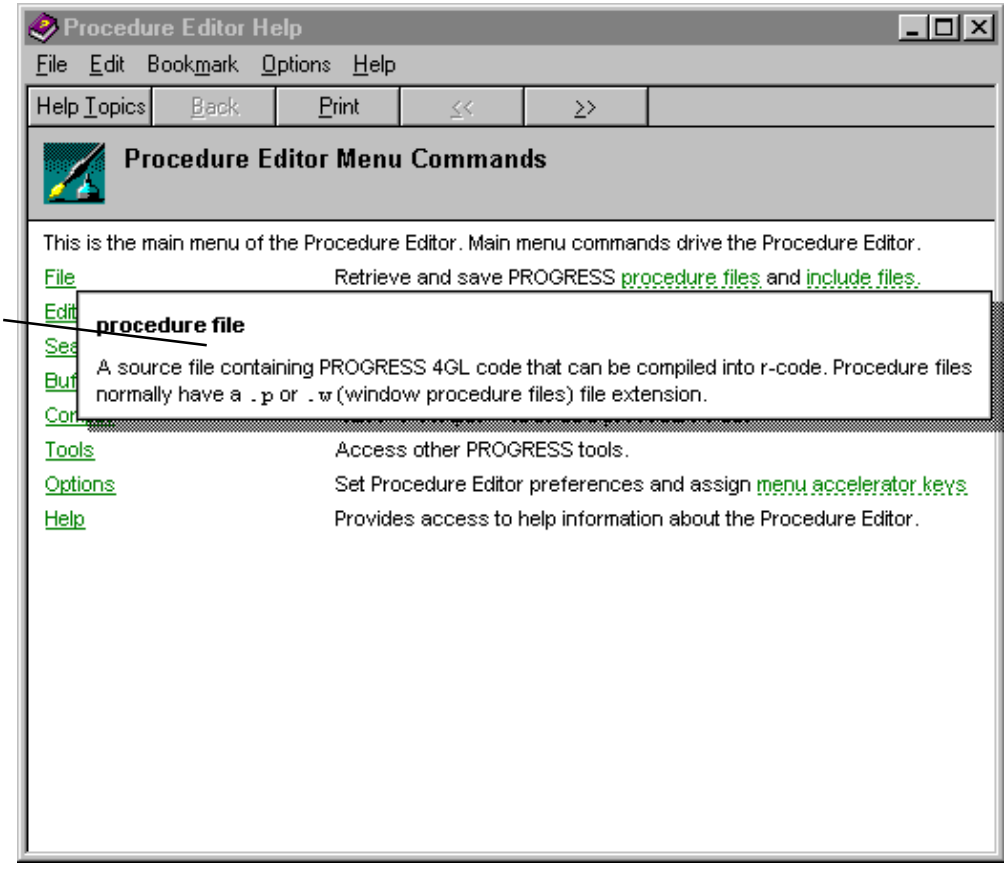


Figure 2–8: Popup Link Activated

For more information about popup links, see the"Creating Popup Links" section in Chapter 3, "Creating Help Topic Files."

2.2 Designing a Help System

The most important thing to consider in designing a help system is the information needs of the end user. There are two main steps involved in designing a help system:

1. Plan the help access methods in the application.

   Determine when the user will need help and what mechanisms you will use to supply help. For context-sensitive help, users can click a help key or a help button to access a topic in the help system or click the question mark icon and then click a control to access control-level help. For immediate access to the Help Topics dialog box, use a help menu or a help key call from the application.

2. Design the help topic structure.

   Your combined knowledge of the information needs of the end user and the access methods you plan helps determine the number and order of help topics you need to create. Usually, you create one context-sensitive topic per application context (normally a window, frame, or dialog box). You design reference help topics like reference information in a printed reference manual-modular, concise, and accessed primarily through the Help Topics dialog box, browse sequences, and jumps.

2.2.1 Organizing Help Topics

Before writing any help text or entering any control codes, you, as the help author, must develop a basic organizational plan for your help system.

If you already have a printed manual, use the Table of Contents as a guide for creating the contents file, which contains the information that populates the Contents tab of the Help Topics dialog box. You can nest second-level book icons inside first-level book icons to represent a hierarchical organization of the material in the help file. This approach allows the user to navigate easily to the appropriate topic.

The individual help topics in the help file can contain any number of popup links and jumps. The user can view definitions and jump to other topics that provide more detailed information about the current subject or information about a related subject. Keywords and browse sequences provide users additional ways to access help information.

Because help supports both text and graphics as jumps, you can also create a pictorial help topic using, for example, a schematic diagram, floor plan, or other diagram with hypertext links to related topics.

Help topic information can often be derived from printed documentation. If you already have a printed manual for your application, you can save time by using the source files created for the manual as the starting point for your online help system. Your first step is to reorganize the information into modular help topics. Then you can add hypertext links that allow the user to access these modular information chunks in a nonlinear fashion.

Figure 2–9 illustrates the difference between the way printed documentation is organized and the way online help is organized.

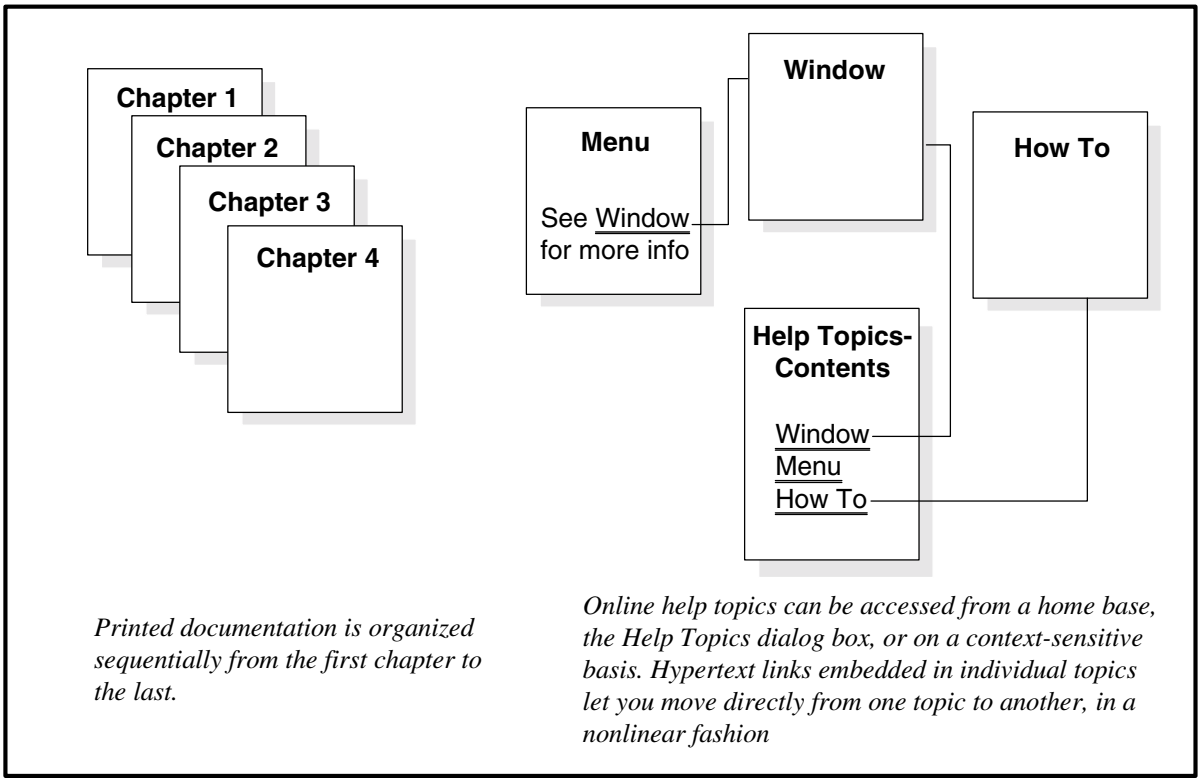


Figure 2–9: Printed Documentation Versus Online Help

If you have no printed documentation to use as the source of the help text, you have to plan and outline the steps that are normally involved in creating a manual. But there are some important differences involved in creating a successful online help system. The following section briefly discusses some general guidelines to use when organizing and creating help topics.

2.2.2 Topic Style Guidelines

In planning your help topics, keep them as brief as possible. To keep topics modular, avoid lengthy introductory paragraphs, complex concepts and comparisons, transitional passages, and text-based cross-references. Many of the techniques that make for good printed material do not translate well into online help.

Instead of writing long help topics that must be scrolled into view while the user reads the text, try to limit each help topic to the amount of information that can fit within its window type. Use hypertext features to lead the user to more information on related subjects.

Whether you are working with previously written material or developing your help topics from scratch, it is important to edit your text rigorously and to format it for maximum readability online. Table 2–4 summarizes some basic style guidelines.

Table 2–4: Help Topic Style Guidelines

Design Element	Guideline
Amount of text	Use a minimum of text. Reading speed of online text is generally 30% slower than printed text.
Paragraph length and white space	Reading online text is harder than reading a printed page. Keep paragraphs as short as possible. Use margins, inter-paragraph spacing, and other white space liberally.
Fonts and text highlighting	A sans serif typeface such as Arial or Helvetica improves readability. Body text should be at least 8 points. Avoid using all uppercase text, multiple fonts, or too many colors.
Graphics and icons	Although help lets you place graphics anywhere in a help topic, graphics can dramatically increase the size of your help file. Use graphics sparingly to aid comprehension. Avoid using graphics for decoration.
Design consistency	To reduce possible confusion and to build end-user confidence in the information your help files provide, be sure to use type, spacing, colors, terminology, hypertext links, and graphical design elements consistently.

The remaining two chapters explain how to create topic, contents, and project files, and how to compile them to create a help project.

The instructions in these chapters assume that you are using only the utilities that are provided with your Progress product, plus MS-Word and accessories that are supplied with Windows. Note that there are many help authoring tools available that can save you time by automating many of the tasks described in this chapter.

For complete reference information on all aspects of Windows help development, see the *Help Author's Guide*, an online help file associated with Help Workshop, `\Program Files\Progress\bin\hcw.hlp`. Help Workshop, `\Program Files\Progress\bin\hcw.exe`, is a Microsoft utility provided with Progress.

3

Creating Help Topic Files

This chapter explains how to create help topic files, one type of help source files. *Help source files* contain the help information for a help system and include help topic files and graphics files, which are described in this chapter, as well as contents files, and help project files, which are described in Chapter 4, “Completing Help Systems.”

This chapter includes the following sections:

- Help topic files
- Including graphics in help topics
- Adding hypertext links
- Using help macros
- Saving help topics in RTF files

3.1 Help Topic Files

The instructions in this guide assume that you are using only the utilities that are provided with your Progress product, plus MS-Word and the accessories that are supplied with Windows. Note that there are many help authoring tools available that can save you time by automating many of the tasks described in this chapter.

A help topic file is a rich-text format (RTF) file containing the content text as well as footnotes and special formatting properties such as underlining and hidden text. You can use MS-Word (or any text editor that can save text as RTF files) or use a help authoring tool that works as an add-on to MS-Word or can import RTF files.

Figure 3–1 shows how a Procedure Editor help topic appears while the help author edits the text and inserts control codes in MS-Word.

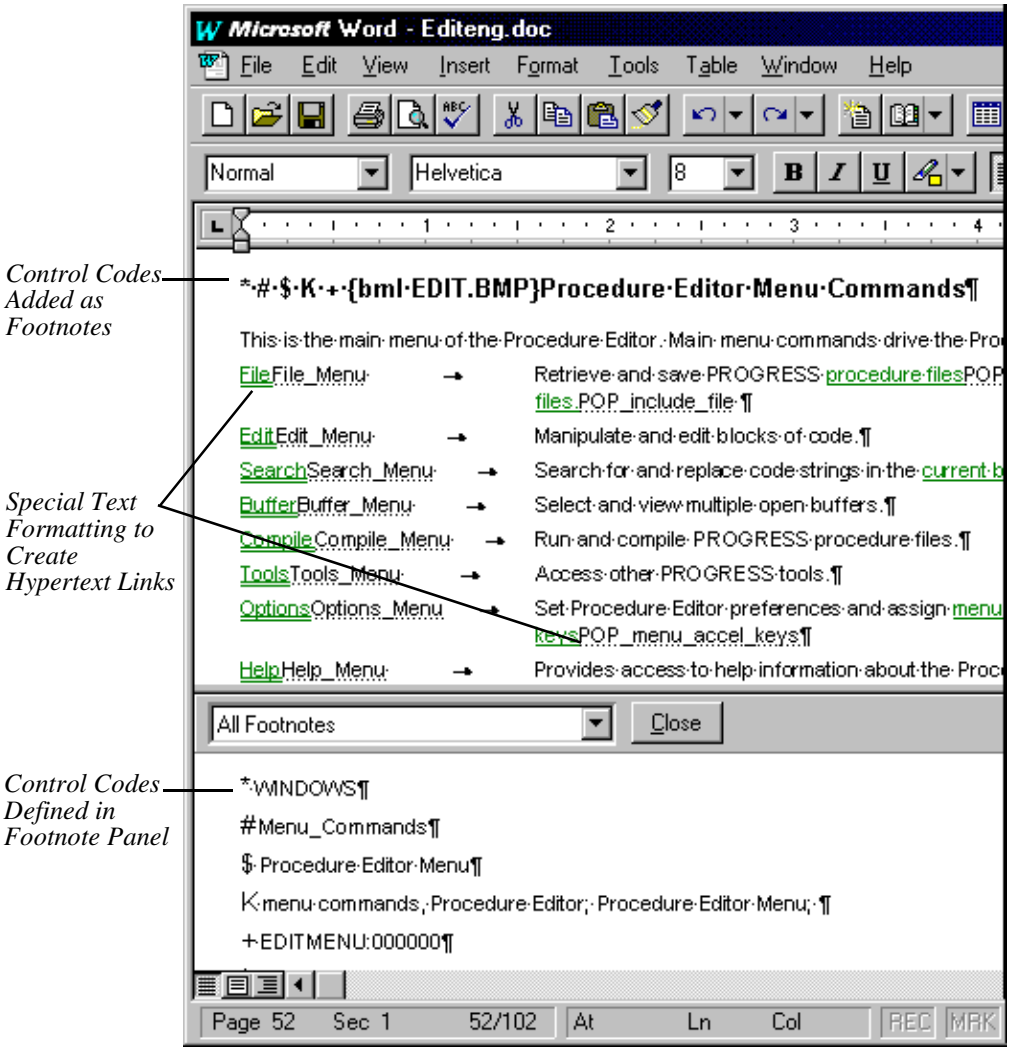


Figure 3–1: Procedure Editor Help Topic

3.1.1 Managing Help Topic Files

A help system can be compiled from one or more help topic files. As the help author, you must decide how many help topic files to create. If you are working alone, you may choose to put all your help topics in a single file. Conversely, if you are working with several writers, you might find it more convenient to create a separate help topic file for each writer. Or you might create help topic files based on the structure or content of the information, such as creating a separate file for each menu in your application or for each printed manual that is shipped with your application.

As long as you code each help topic correctly and identify all the help topic files required in your help system within the help project file, the help compiler will successfully compile all the information into a single help file.

3.1.2 Overview of Creating Help Topic Files

In a help topic file, each help topic (except the very first one) begins with a hard page break. You enter information essential for hypertext linking, keyword indexing, and browse sequencing by inserting special control codes into help topics. You insert these control codes as footnotes using special characters such as the pound (#) and dollar ($) signs.

The following sections provide more information on completing these steps in creating help topic files:

- Open a file in a text editor.
- Separate each topic with a page break.
- Create a title for each topic.
- Assign control codes to each help topic.
- Write the topic text.
- Assign index keywords, browse sequences, and build tags (all optional).
- Create non-scrolling regions for certain topics (optional).
- Insert graphics throughout the topic file (optional).
- Create hypertext links among help topics (optional).
- Save the topics in one or more RTF files.

3.1.3 Separating Topics

After opening a file in a text editor, you can start creating topics. Within the help topic file, a help topic begins with one or more footnote control codes and ends with a page break. Figure 3–2 shows three topics in the same help topic file. Notice that each help topic in the topic file is separated by a page break.

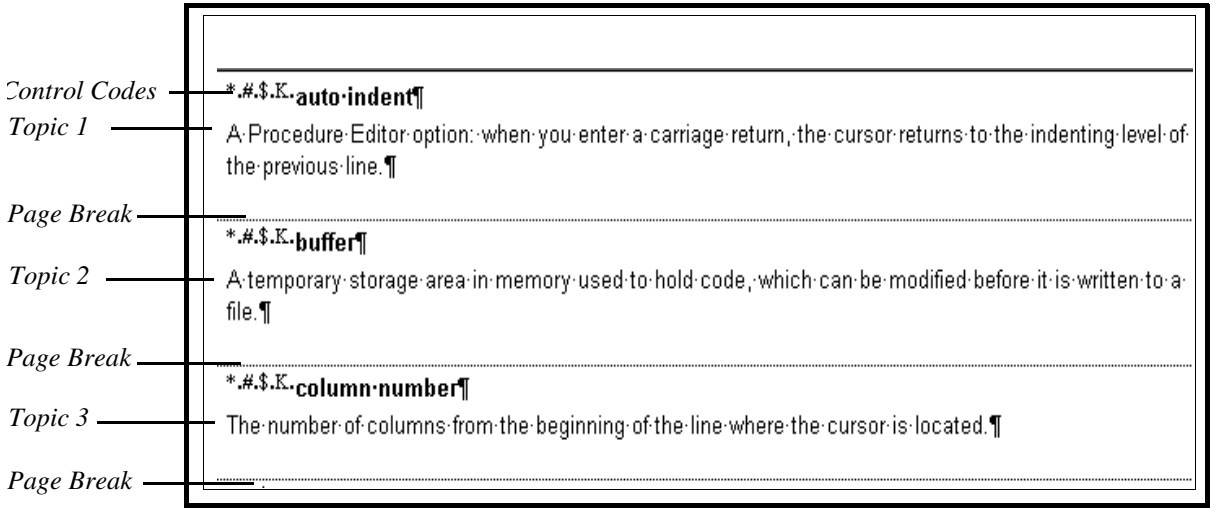


Figure 3–2: Help Topics Separated by Page Breaks

Follow these steps to insert a page break between topics using MS-Word. (If you are using a different RTF text editor, please see its documentation for how to perform the equivalent text formatting.)

1 ♦ Place the insertion point between the topics.

2 ♦ Choose Insert→ Break.

The Break dialog box appears.

3 ♦ Choose Page Break.

4 ♦ Choose OK.

You should see a dotted line (see Figure 3–2) indicating that MS-Word has inserted a page break.

3.1.4 Inserting Control Codes

Control codes are footnotes that the WinHelp compiler uses to build the help file. Table 3–1 summarizes the control codes. See the sections following for additional information on the individual control codes.

Table 3–1: Summary of Control Codes *(1 of 2)*

Control Code	Purpose	Comments
Build tag *(asterisk)	Tells the help compiler whether to include a help topic in a project build. If a topic is not assigned a build tag, it is automatically included in each build. Build tags are optional.	Can contain any character except a space. If you want to assign multiple build tags to a help topic, you must separate them with a semicolon (;). Example: `* WINDOWS;WIN16`
Topic ID # (pound sign)	Uniquely identifies a help topic.	Can contain any alphanumeric characters, a period, and/or an underscore, without spaces. Although topic IDs can contain up to 255 characters, keep them as short as possible. A recommended practice is to start topic IDs for context-sensitive help topics with “IDH_”. If using numbers, start with a positive number greater than 0. Example: `# IDH_File_Menu`
Topic title $ (dollar sign)	Assigns a title to a help topic. Topic titles appear on the Bookmark menu, the History dialog box, and in the list of topics displayed on the Index tab of the Help Topics dialog box. Topic titles are optional.	Can contain any character. Do not use any special formatting for the topic title text. Example: `$ File Menu`
Keyword K (capital K)	Assigns a keyword to a help topic so it can be found quickly using the Index tab of the Help Topics dialog box. Keywords are optional. NOTE: The help engine is case insensitive when performing a keyword search.	Can contain any ANSI character, including spaces and accented characters, and can contain up to 255 characters. A keyword can be a single word or a phrase. Do not use any special formatting for the keyword text. If you want to assign multiple keywords to a help topic, you must separate them with a semicolon (;). Example: `K File Menu; menus;`

Table 3–1: Summary of Control Codes *(2 of 2)*

Control Code	Purpose	Comments
A-keyword A (capital A)	Assigns a keyword to a help topic so it can be found quickly by an ALink search (which is usually associated with a Related Topics button).	Same rules as for (K-) keywords. A-Keywords are optional and do not appear in the Procedure Editor help file. Example: `A interface-elements;`
Browse sequence + (plus sign)	Assigns the help topic a position in a browse sequence. Browse sequences are optional.	Can contain the name of a browse sequence followed by a colon and the assigned sequence number. The sequence name is an optional name so that the help compiler can distinguish different browse sequences. (If you don't assign a sequence name, the topic is automatically assigned to a default (unnamed) sequence.) As you assign topics to a browse sequence, its a good idea to skip numbers so you can add more help topics to the sequence later. Example: `+ EDITMENU:00005`
Window type > (greater than sign)	Specifies the window type a topic will appear in when the topic is opened from the Index or Find tab of the Help Topics dialog box, or from an ALink or KLink macro.	The window type must be defined in the [WINDOWS] section of the help project file. Example: `>(w95sec)`

Follow these steps to assign a build tag, context string, topic title, keyword, and browse sequence to a help topic using MS-Word. Figure 3–3 shows the relationship between the footnote markers and the content of their corresponding footnotes for the most common control codes.

1 ♦ Place the cursor immediately to the left of the first character in the help topic text with no space between the cursor and the first character in the heading.

2 ♦ Choose Insert→ Footnote.

The Footnote dialog box appears.

3 ♦ Select the Custom Footnote Mark radio button.

4 ♦ Enter the appropriate control code (* for build tag, # for context string, $ for topic title, K or A for keywords, or + for browse sequence) in the Custom Footnote Mark field.

5 ♦ Choose OK.

MS-Word automatically performs the following actions:

- Inserts a superscript copy of the footnote symbol and one space to the left of the first character in the help topic
- Opens a footnote pane at the bottom of the screen, moves the cursor into this window, and inserts the footnote symbol

6 ♦ Enter the appropriate footnote text in the footnote pane following the footnote symbol.

For example, in Figure 3–3, “Menu_Commands” is a context string that is entered after the # symbol in the footnote window to uniquely identify the Procedure Editor Menu Commands help topic. There should be one space only between the footnote symbol and the footnote text.

7 ♦ When you finish entering the footnote text, choose the Close button in the Footnote pane.

NOTE: You can also insert help macros as a control code in a similar manner. For more information, see the “Using Help Macros” section.

Figure 3–3 shows the appearance of the control codes and their corresponding footnotes for a typical help topic.

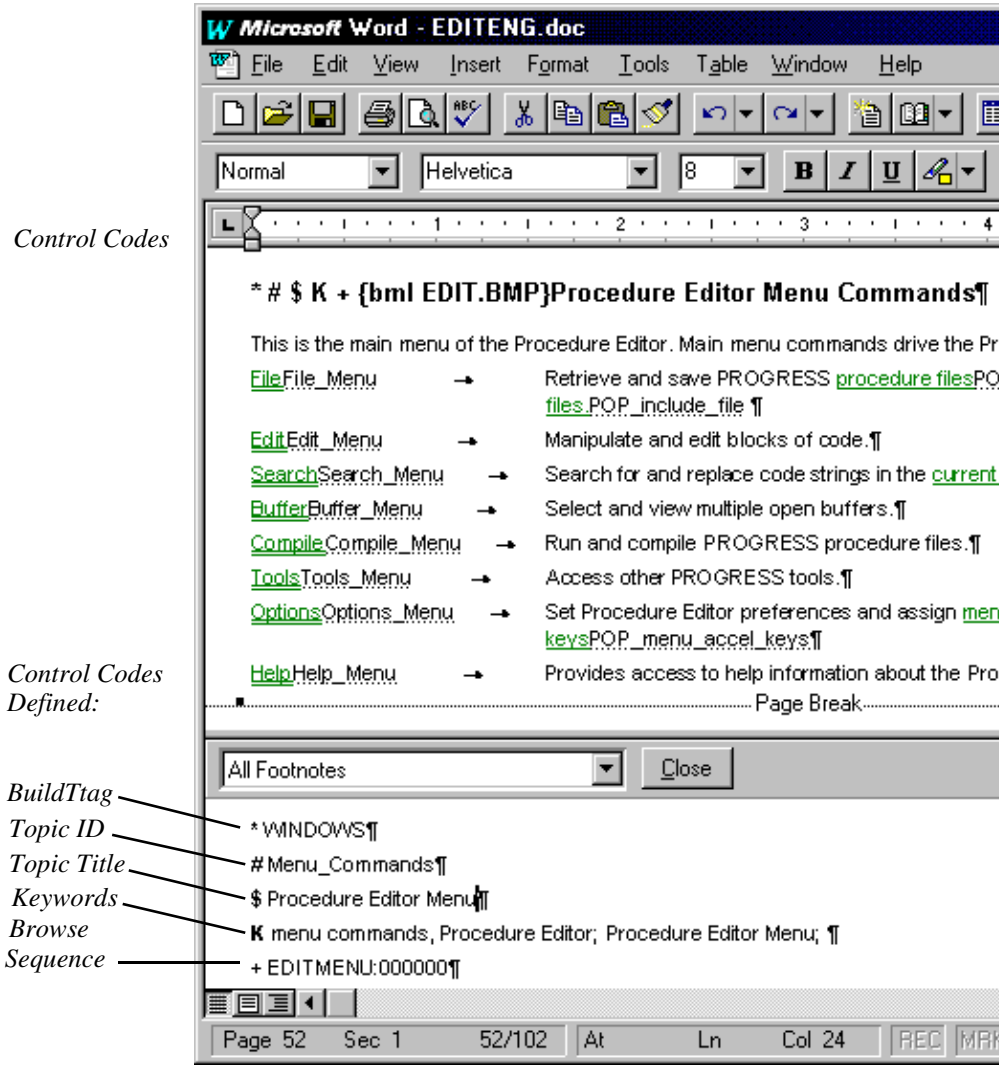


Figure 3–3: Example of Control Codes for a Help Topic

3.1.5 Assigning Topic IDs

A *topic ID* is a unique character string that identifies each topic in a help system. The topic ID tells the help engine which topic to display when the user chooses a popup link, jump, or a user interface feature that has been assigned context-sensitive help.

A topic ID footnote may contain alphanumeric characters, the period, and the underscore. Other non-alphanumeric characters, such as parentheses, dashes, and spaces, are not allowed. One space separates the control code (#) and the topic ID footnote text:

```
# Menu_Commands
```

Each help topic can be assigned only one topic ID. Although not mandatory, each help topic should be assigned a topic ID. A topic that is not assigned a topic ID can only be accessed by an index keyword search or as part of a browse sequence.

3.1.6 Assigning Topic Titles

A *topic title* is a word or phrase that identifies a help topic in the help viewer. The help viewer uses the topic title string to represent the topic on the Index tab of the Help Topics dialog box and in navigation functions like the History or Bookmark functions.

One space separates the control code ($) and the topic title footnote text. Do not use any special formatting for the topic title text:

```
$ Procedure Editor Menu
```

Note that the topic title text you embed in the $ footnote is not automatically used by the help engine to display a title for the topic in the help window. You must include the title that will appear in the help window at the beginning of the help topic itself. For the sake of clarity, the topic title text in the $ footnote should be identical to the title text that appears at the beginning of the help topic in the help window.

Although the help compiler accepts a topic title that is up to 128 characters long, each function has a different limit to how many characters it can display. A long topic title may be truncated by the Bookmark function or on the Index tab of the Help Topics dialog box.

3.1.7 Creating Topics for Context-sensitive Help

To create context-sensitive help topics, determine the application contexts that require help topics and create one help topic for each context. An *application context* is any currently displayed user-interface element that has input focus, for example, a window, dialog box, browse, button, combo-box, control frame, editor, fill-in, radio-set, selection-list, slider, and toggle box. There are two types of context-sensitive help:

- Context-sensitive help at the window and dialog box level calls a help topic and the help viewer remains open until the user closes it.
- Context-sensitive help at the control level calls a popup topic and the help viewer closes after the user clicks again on the topic or the interface.

For context-sensitive help topics to be called from the application interface, each topic must have a *context ID*, a unique identifying number. The context ID is paired with the topic ID in the Map section of the project file. (See "Mapping Context-sensitive Help Topics" in Chapter 4, "Completing Help Systems," for more information on this process.)

If you are creating topics for context-sensitive help, you can integrate the help file with the application it document before you complete all the help topic text. You can create preliminary, empty topics called stubs, by inserting page breaks, giving each topic a title, and adding footnote control codes.

To complete the integration of context-sensitive help with the application, you need to provide a list of the help topic IDs and context IDs (from the Map section of the project file) to your development team to include in the calling interface for each context. Note that the development team does not need to use your topic ID, but must use the same context ID. Note that on some software development projects, the help author provides the context IDs and on other projects, the development team provides the context IDs.

As soon as you create the stubs, you can compile your help system, and provide the help system to include with the product for preliminary testing. (For information on compiling, see the "Compiling a Help Project" section in Chapter 4, "Completing Help Systems.")

3.1.8 Writing Topic Text

If you have created help stubs for context-sensitive help, you can write the topic text as soon as information becomes available. For information on designing and writing your topic text, see the "Topic Style Guidelines" section in Chapter 2, "Overview of Help System Development."

Use manual numbering rather than MS-Word's autonumbering feature. Using manual numbering avoids formatting problems in your compiled help system, such as random or deleted tabs.

3.1.9 Assigning Index Keywords

A *keyword* is a word or phrase the help author assigns to a topic so a user can quickly find that topic using the Index tab of the Help Topics dialog box.

A help topic can be assigned one or more keywords by inserting the keyword after the K control code. A keyword footnote may contain any ANSI character, including spaces and accented characters, and can be up to 255 characters long. One space separates the control code and the keyword footnote text. Keywords are not required:

```
K menu commands
```

If you want to assign multiple keywords to a help topic, you must separate them with a semicolon (;) as shown below:

```
K menu commands, Procedure Editor; Procedure Editor Menu;
```

3.1.10 Creating Browse Sequences

A *browse sequence* is a set of related help topics that are meant to be read in a specific order. The user navigates though a browse sequence by clicking the backward (<<) and forward (>>) help window buttons.

For example, in the Procedure Editor help file, when you browse forward from the Procedure Editor Menu Commands help topic displayed in Figure 3–6, the help engine displays the next topic in the browse sequence-the File Menu help topic, displayed in Figure 3–7.

The help information for the Procedure Editor menus is designed as a browse sequence. Each help topic in the browse sequence is assigned a consecutive browse sequence number that specifies where it appears in the sequence. The Editor main menu is assigned the number 000000, the Editor File menu 000005, the Editor Edit menu 000010, and so on. You can define more than one browse sequence in a help file, but an individual help topic can reside in only in one browse sequence.

Figure 3–4 diagrams a browse sequence used in the Procedure Editor help file.

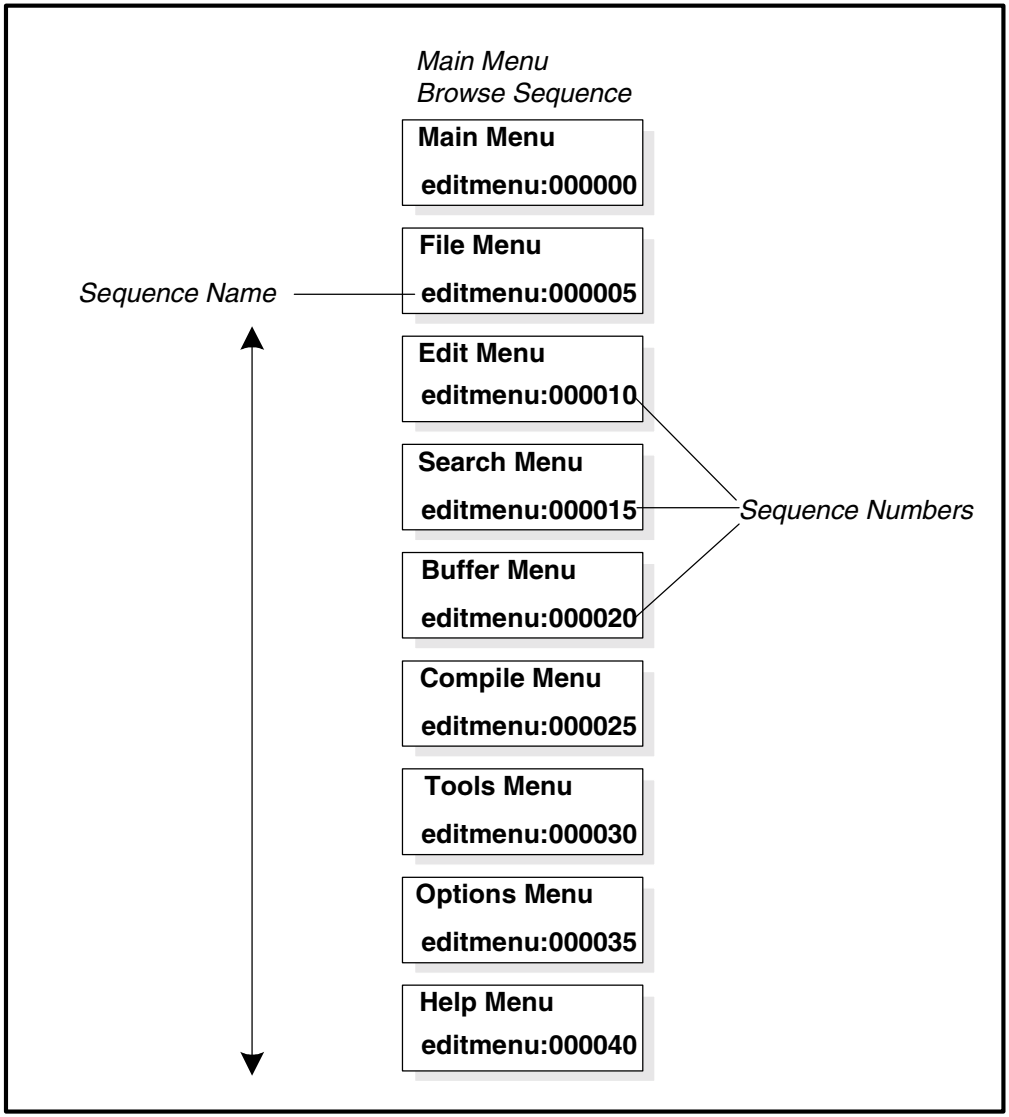


Figure 3–4: Browse Sequence Diagram

A help topic can be assigned to a browse sequence by inserting a sequence name, a colon, and a sequence number in the footnote text following the control code. One space separates the control code (+) and the keyword footnote text. A help topic can be assigned to no more than one browse sequence:

```
+ editmenu:000000
```

Assigning Browse Sequence Names

The *sequence name* is an optional name you assign so the help compiler can distinguish among different browse sequences. A sequence name cannot contain embedded spaces. If you do not assign a sequence name to a topic, the topic is automatically assigned to a null (unnamed) sequence.

Assigning Browse Sequence Numbers

The *sequence number* is a unique alphanumeric string that the help author assigns to a help topic in a browse sequence. The help compiler uses the sequence number to do an ASCII (ascending) sort of all help topics defined in a particular browse sequence. For example, when the user selects the next help topic in a browse sequence, the help viewer displays the help topic with the next highest sequence number in that sequence.

Because the help compiler sorts sequence numbers by ASCII value, the length of each string should be consistent across all your help topic files. If you use only numerical characters in your browse sequence numbers, make sure all the numbers contain the same number of digits. (For instance, the number 100 precedes the number 99 because the compiler compares only the first two digits in this pair of numbers).

As you assign sequence numbers to topics, it is a good idea to skip numbers so you can insert new help topics into the sequence at a later time.

3.1.11 Assigning Build Tags

A *build tag* tells the help compiler whether to include a topic in the help file when the compiler processes the help topic files. Build tags are used to create multiple versions of a help file from a single help topic file. By default, if a topic is not assigned a build tag, it is automatically included in each build. Build tags are optional.

A build tag footnote may contain any character except a space. One space separates the control code (*) and the build tag footnote text:

```
* WINDOWS
```

If you want to assign multiple build tags to a help topic, you must separate them with a semicolon (;), as shown below:

```
* sample_build; build_3; build_final
```

NOTE: You also must define the build tag for a build in the help project file. (See the "[BUILDTAGS] Section" section in Chapter 4, "Completing Help Systems.")

3.1.12 Creating Non-scrolling Regions

Help allows you to create a non-scrolling region at the start of a help topic. The *non-scrolling region* displays part of the help topic (such as the topic heading and/or a graphic) continuously while the user scrolls through the other text in the help topic as shown in Figure 3–5.

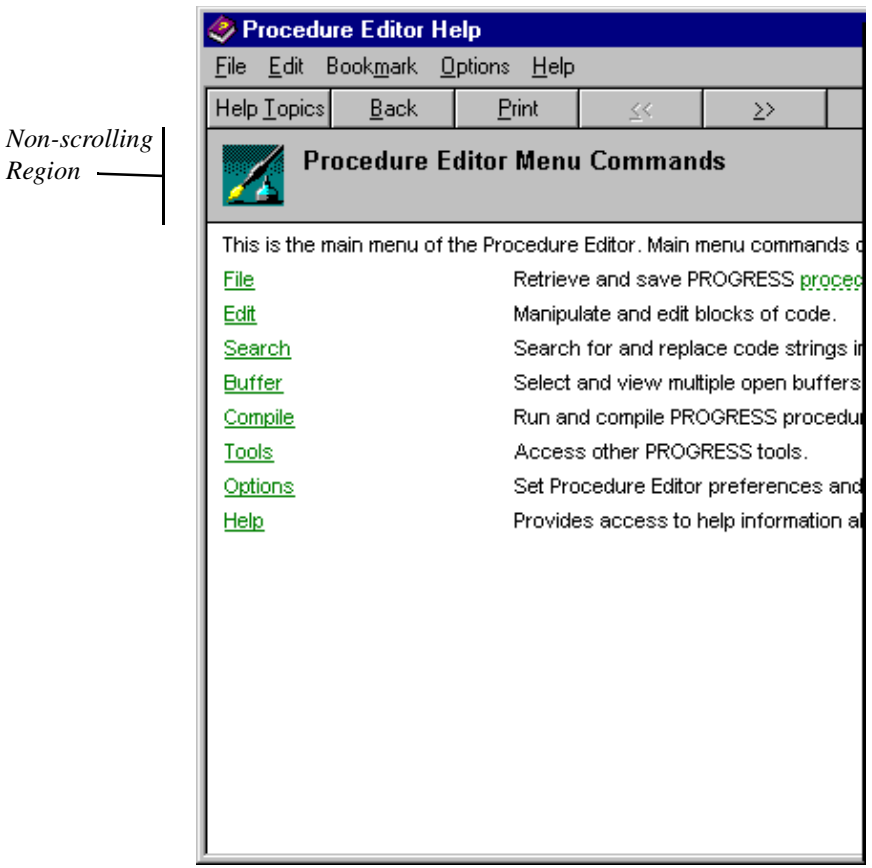


Figure 3–5: Example of a Non-scrolling Region

Help topics that you want the help engine to display in an overlapping, popup window, should **not** contain non-scrolling regions; otherwise, only the non-scrolling region will display.

Follow these steps to create a non-scrolling region for a help topic with MS-Word:

1 ♦ Select all the lines of help text that you want to include in the non-scrolling region.

The selected text must include the first line of the help topic. The selection may include a graphic reference.

2 ♦ Choose Format→ Paragraph.

The Paragraph dialog box appears.

3 ♦ Activate the Keep With Next toggle box.

4 ♦ Choose OK.

3.2 Including Graphics in Help Topics

Help allows you to use graphic images to illustrate your help text. A help topic that contains a graphic can function as a standard illustration, or it can be coded as a jump, a popup link, or a destination topic. (Linking graphics to help topics is discussed later in this chapter.) This section lists the graphic formats the help viewers support and explains how to insert graphics into help topic files.

Graphics files consist of images in certain file formats. You reference graphics files inside the help topic file where you want them to appear, and then they become a part of the finished help file at compile time.

The help engine supports several graphic formats. Table 3–2 lists the supported graphic formats.

Table 3–2: Supported Graphics Formats

Graphics Format	File Extension
Windows Bitmap	.bmp or .dib
Segmented Hypergraphic	.shg
Windows Metafile	.wmf
Multiple-resolution Bitmap	.mrb

Whenever file size is an issue, using segmented hypergraphics can reduce file size dramatically. (For more information about segmented hypergraphics, see the "Adding Hotspots to Graphics" section.)

3.2.1 Tools for Creating Graphics

There are many tools that you can use to create graphics files for a help system. Many help authors use the graphics editing tool that Microsoft ships with Windows (Paint) for creating Windows bitmaps. You can create multiple-resolution bitmaps with the Microsoft Multiple-Resolution Bitmap Compiler (MRBC), and use the Windows Hotspot Editor (also known as the System Help Editor utility or `shed.exe`) to create segmented hypergraphics. (For more information, see the "Adding Hotspots to Graphics" section .)

As a sound design practice, you should create graphics for display on the most basic computer monitors with which your users are working. For example, images that you design for monitors with higher resolutions (such as super VGAs) might not appear the same on lower resolution monitors (such as VGAs). Certain colors do not translate to visible gray scales on some monochrome monitors. Plan for the most limited environment, and whenever possible, test images on the monitor type of your user base.

3.2.2 Placing Graphics By Reference

You include graphics in help files using a method called *placement by reference*. This allows you to specify the filename and alignment of the graphic in a help topic, as opposed to embedding the graphic directly in the topic. The compiler then builds the actual graphic image into the finished help file. The example below references the graphic file, `edit.bmp`:

```
The Procedure Editor {bmc edit.bmp} is an interactive tool that
```

Figure 3–6 shows how the same line of text appears with the graphic displayed:

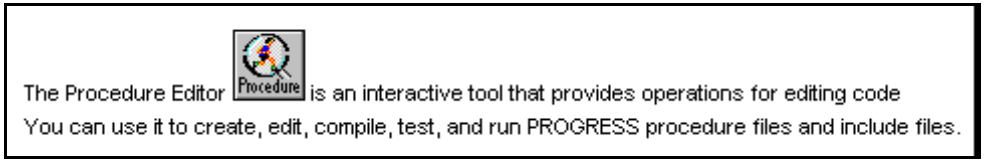


Figure 3–6: Graphic Displayed in a Help Window

Although you can only view the graphic after the help file is compiled, placement by reference lets you include a given graphic as often as you want without substantially increasing the size of the help file.

Help supports three techniques for placing graphics by reference. You can place the graphic:

- As a character in a line of text
- Along the left margin of text
- Along the right margin of text

For each technique, use the appropriate help statement followed by the graphic filename enclosed in curly braces ({ }). Table 3–3 lists the help statements used for placing graphics.

Table 3–3: Help Statements for Placing Graphics

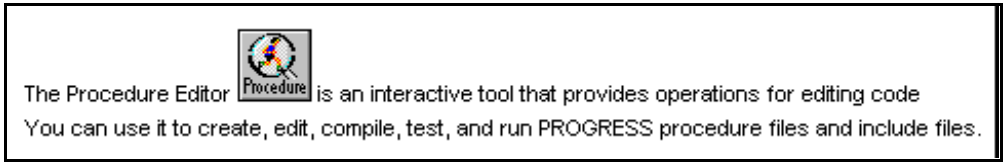
Graphic Location in Text	Help Statement Syntax
Treated as a character	{bmc *filename}*
Left margin	{bml *filename}*
Right margin	{bmr *filename}*

NOTE: When using the placement by reference technique, do not include the graphic's pathname in the graphic placement command. By default, the help compiler expects all graphics to be in the same directory as the help topic file. The full pathname of each graphic that is not stored in the help topic file's directory must be included in the [BITMAPS] section of the help project file. For more information, see the "[BITMAPS] Section" section in Chapter 3, "Creating Help Topic Files."

The following sections demonstrate each graphic placement technique.

Placing a Graphic as a Character in a Line of Text

Help text can precede or follow the graphic on the same line, as shown in the following example, but it does not automatically wrap around the graphic:



Placing a Graphic Along the Left Margin

Help text automatically wraps around the right side of the image as shown in the example below:



The Procedure Editor is an interactive tool that provides operations for editing code. You can use it to create, edit, compile, test, and run PROGRESS procedure files and include files.

Placing a Graphic Along the Right Margin

Help text automatically wraps around the left side of the image as shown in the example below:

The Procedure Editor is an interactive tool that provides operations for editing code. You can use it to create, edit, compile, test, and run PROGRESS procedure files and include files.



3.3 Adding Hypertext Links

After you create the help topics for your help system, the next step is to provide users with ways to navigate among these topics. *Hypertext links* are sections of text or graphics that you code to direct the help engine to display a specified topic or choice of topics when the user selects the hypertext link. This section describes various types of links in text and graphics and how to create them.

3.3.1 Components of a Hypertext Link

Hypertext links consist of two parts: *link text*, a visual cue to the user identifying the link; and a *destination topic identifier* that informs the help viewer which topic or topics to find when the user activates the link. The destination topic identifier is the topic ID of the topic you are linking or a KLink or ALink macro listing keywords or A-keywords to use in a topic search. Figure 3–7 shows a jump as it might appear in a help topic file.

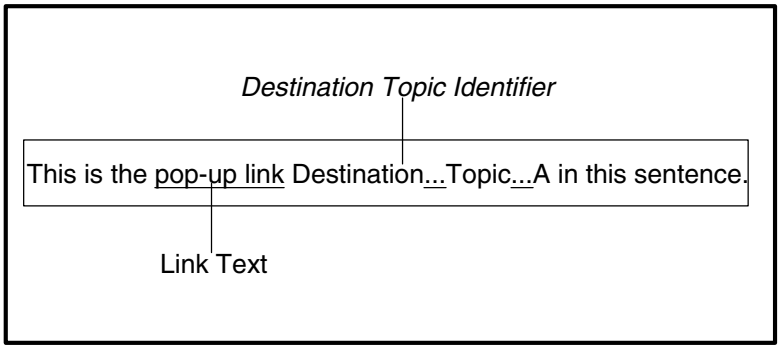


Figure 3–7: Hypertext Link and Its Components

In a help window, which is displaying topics from the compiled help file, the user sees only the link text. The help engine does not display the destination topic identifier. The appearance of the link in the help viewer indicates the link type, as shown below.

In a help window, jumps, KLinks, and ALinks are identified as text with a solid underline:



Popup links display as text with a dotted underline (hidden text in MS-Word):



All link text is displayed in green by default.

3.3.2 Creating Jumps

A *jump* is a word, phrase, or graphic in a help topic that causes the help engine to immediately display a different help topic. If the window type specified for the second help topic is the same as that of the original topic, then the second topic replaces the original help topic in the help window. If the window type for the second topic is different from that of the original topic, the help engine launches a window of that type and displays the second topic in it. (For an example of how the user executes a jump, see the "Hypertext Links" section in Chapter 2, "Overview of Help System Development.")

NOTE: You can also create jumps that execute help macros. For more information, see the "Using Help Macros" section in this chapter.

Follow these steps to create a jump using MS-Word:

1 ♦ Highlight the word, phrase, or graphic reference you want to use as a jump.

2 ♦ Choose Format→ Font.

The Font dialog box appears.

3 ♦ Select Double underline (or strikethrough).

4 ♦ Choose OK to close the Font dialog box.

5 ♦ Move the cursor to the end of the underlined text. Be certain there is no space between the word and the cursor and be careful not to include the paragraph mark.

6 ♦ Choose Format→ Font (again).

The Font dialog box appears.

7 ♦ Choose Hidden.

8 ♦ Choose OK to close the Font dialog box.

9 ♦ Enter the topic ID of the destination topic for the underlined word, phrase, or graphic reference.

The text you enter should appear with a dotted underline, to indicate that it has the hidden formatting attribute.

In the example shown below, the phrase “jump” (formatted as double underline text) is the jump text. The text string “Destination_Topic_B” (formatted as hidden text) is the topic ID of the destination topic assigned to the jump:

This is the go-to linkDestination Topic B in this sentence.

The following example shows a jump created with a graphic. When the user clicks on the graphic, the help engine displays the topic identified by the topic ID “Destination_Topic_B”:



When creating a jump with a graphic that you have placed by reference, the entire reference-including the help statement and the curly braces-serves as the link text. (For information about referencing graphics in a help topic, see the “Placing Graphics By Reference” section in this chapter.)

3.3.3 Creating Popup Links

A *popup link* is a word or phrase in a help topic that immediately displays a second help topic in a popup window when you choose it. (For an example of how the user executes a popup link, see the “Hypertext Links” section in Chapter 2, “Overview of Help System Development.”)

Follow these steps to create a popup link using MS-Word:

1 ♦ Select the word, phrase, or graphic reference you want to use as a popup link.

2 ♦ Choose Format→ Font.

The Font dialog box appears.

3 ♦ Select Single underline.

4 ♦ Choose OK to close the Font dialog box.

5 ♦ Move the cursor to the end of the underlined text. Be certain there is no space between the word and the cursor and be careful not to include the paragraph mark.

6 ♦ Choose Format→ Font (again).

The Font dialog box appears.

7 ♦ Choose Hidden.

8 ♦ Choose OK to close the Font dialog box.

9 ♦ Enter the topic ID of the destination topic for the underlined word, phrase, or graphic reference.

The text you enter should appear with a dotted underline, to indicate that it has the hidden formatting attribute.

NOTE: Do not use help topics with non-scrolling regions as destination topics for popup links. When you use a topic containing a non-scrolling region, the help engine displays only the non-scrolling region and nothing more.

In the example shown below, the phrase “popup link” (formatted as single underline text) is the popup link text. The text string “Destination_Topic_A” (formatted as hidden text) is the topic ID of the destination topic assigned to the popup link:



The following example shows a popup link created with a graphic reference. When the user clicks the mouse on the graphic, the help viewer displays the topic identified by the topic ID “Destination_Topic_A”:



When creating a pop-up link with a graphic that you have placed by reference, the entire reference-including the help statement and the curly braces-serves as the link text. (For information about referencing graphics in a help topic, see the “Placing Graphics By Reference” section .)

3.3.4 Creating Cross-file Links

Cross-file links are jumps and popups that have destination topics located in external help files:

SYNTAX

```
link-textdestination-topic@pathname
```

You format the text for the popup or jump in the regular way (single or double underlined followed by hidden text) except that after the destination topic, you insert an at sign (@) followed by the name of the help file containing the destination topic:

```
SYSTEM-HELP statementsystemhelpstatement@lgrfeng.hlp
```

In this example, when the user chooses the jump (SYSTEM-HELP statement), the help viewer displays the help topic for the SYSTEM-HELP statement located in the help file `lgrfeng.hlp`.

NOTE: At run time, the help file that contains the destination topics must reside in the path specified in the link itself. If you do not specify a pathname, the help viewer assumes that the help file resides in the current working directory.

3.3.5 Creating Secondary Window Links

You can use a technique called *window redirection* to display a help topic in a secondary window or to force a topic to be displayed in the main window. *Secondary windows*, unlike the main help viewer window, cannot contain a menu bar:

SYNTAX

```
link-textdestination-topic>window-name
```

You format the popup or jump in the same way as destination topics (single or double underlined followed by hidden text), except that after the destination topic you insert a greater-than sign (>) followed by the name of a window type that you have defined in the [WINDOWS] section of your help project file:

```
databasePOP_database>glossary
```

In this example, when the user chooses the jump (database), the help viewer displays the help topic for the string "database" in a dynamically created secondary window (glossary). The help writer defined this secondary window in the help project file. For more information on defining windows, see the "[WINDOWS] Section" in Chapter 4, "Completing Help Systems."

3.3.6 Adding KLinks and ALinks

KLinks and ALinks are hypertext links from words, phrases, or graphics in help topics that direct the help engine to run a macro that searches for other help topics based on specified keywords. With KLinks and ALinks, you do not need to specify any destination topic IDs.

KLinks and ALinks work exactly the same way, except for the types of keywords they use to identify matching topics, and the help macros they use to find sets of matching topics. KLinks use the KLink macro to search for topics associated with a K-keyword, while ALinks use the ALink macro to search for topics associated with an A-keyword.

An advantage of using KLinks and ALinks is that you can create them and compile the help file that contains them without knowing exactly what topics will serve as their targets. Also, because KLinks and ALinks use keywords instead of topic IDs, they can identify more than one topic as a target. When multiple topics are found, their titles are displayed in a Topics Found dialog box.

A disadvantage of using KLinks and ALinks as opposed to jumps, is that it is possible to create a KLink or an ALink that does not find any topic. When this happens, the help engine either displays a default topic (which you specify) or displays a message box with the text "No additional information is available."

The difference between KLinks and ALinks lies in the way their corresponding keyword types are meant to be used in help systems.

KLinks

A KLink works by finding a set of topics, each of which is associated with one or more specified index keywords. Each index K-keyword is stored in the K footnote of a topic and it says something about that topic. For example, a help topic about the File menu might have as an index keyword, "saving files," because the task of saving files is something you can do from the File menu.

Index keywords (K-keywords) are the keywords that appear on the Index tab of the Help Topics dialog box, and they are intended to function in exactly the same way as the keywords that appear in the index in the back of a printed book. In fact, the results of a KLink topic search for any keyword will be exactly the same as when the user double-clicks on that keyword in the Index Tab.

ALinks

An ALink works by finding a set of topics, each of which is associated with one or more specified A-keywords. Each A-keyword is stored in the A footnote of a topic and it says something about another topic that this topic is related to. For example, in a command reference help file, if Command-1 is related to Command-2, then the topic explaining Command-1 might have an A-keyword, "Command-2." The topic on Command-2 would then have a Related Topics button or text hotspot designed to find all other topics that reference Command-2 in their A footnotes.

An ALink is often associated with a graphical hotspot that looks like a button, and is labeled Related Topics. (Note that a Related Topics "button" is not the kind of button you create with a CreateButton macro in the [CONFIG] section of the help project file; it is a bitmap that you turn into a graphical hotspot using the Hotspot Editor. (See the "Adding Hotspots to Graphics" section.)

An advantage of ALinks, is that when you add a new topic to an existing help file, you can identify it as related to an existing topic without editing that topic. Instead of adding a hard-coded "See also" jump to the old topic that refers back to the new topic, you add an A-keyword to the new topic, indicating that it is related to the old one.

Creating KLinks and ALinks

Follow these steps to create a KLink or an ALink:

1 ♦ Decide which word, phrase, or graphic reference will serve as the link text (the visual cue that identifies the jump to the user).

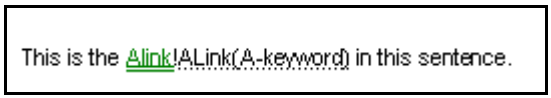
2 ♦ Format the text or graphic reference as double underline (or strikethrough) text.

3 ♦ Enter the KLink or ALink macro as hidden text immediately following the link text.

In the example shown below, the phrase "KLink" (formatted as double underline text) is the KLink link text. The text string "!KLink(keyword)" (formatted as hidden text) is the KLink macro that performs the search for all topics that reference the keyword in their K footnotes:



In the example shown below, the phrase “ALink” (formatted as double underline text) is the ALink link text. The text string “!ALink(keyword)” (formatted as hidden text) is the ALink macro that performs the search for all topics that reference the keyword in their A footnotes:



For more detailed information on the KLink and ALink macros, see the *Help Author’s Guide* (`hcw.hlp`), the help file for Help Workshop.

3.3.7 Adding Hotspots to Graphics

Help viewers allow you to use graphic images that contain hotspots. A *hotspot* is a rectangular region that functions like a jump or a popup link. A graphic image that contains one or more hotspots is called a *segmented hypergraphic*.

You can add hotspots to a Windows bitmap (*.bmp) using the Windows Hotspot Editor (also known as the System Help Editor (SHED), `\Program Files\Progress\bin\shed.exe`. You can also use SHED to create segmented hypergraphics and to compress bitmap files. You can find more information about SHED in *Using Hotspot Editor*, the SHED help file, `shed.hlp`.

Follow these steps to add a hotspot using SHED:

1 ♦ Run SHED.

2 ♦ Choose File→ Open.

The Open File dialog box appears.

3 ♦ Enter the name of the bitmap file and choose OK.

The bitmap appears on the screen.

4 ♦ Click and drag the mouse pointer to draw a rectangular area on the bitmap.

SHED automatically displays the region you have drawn with resize handles. Resize the region as necessary.

5 ♦ Choose Edit→ Attributes.

The Attributes dialog box appears.

6 ♦ Enter the topic ID of the destination topic.

7 ♦ Specify whether you want the region to function like a popup or a jump, or to run a macro.

8 ♦ Specify the region border as visible or invisible.

9 ♦ Choose File→ Save.

SHED saves the bitmap as a segmented hypergraphics file (.shg).

3.4 Using Help Macros

It is possible to control and customize the functionality of help files by using help macros. For example, you can customize the features of help windows by adding buttons to the button bar or menu items to the menus. This section discusses help macros and illustrates some ways to use them.

You can define macros in several places:

- In the help project file (.hlp), you can define a macro so that it runs when the help file is opened, when a particular window type is launched, when the user chooses a keyword from the Index tab of the Help Topics dialog box, or when the user chooses a menu item or button. In the help project file, you define help macros in the [CONFIG] and [MACROS] sections.
- In the contents file (.cnt), you can define a macro so that it runs when the user double-clicks an associated page icon on the Contents tab of the Help Topics dialog box.
- In the help topic file, you can define a macro in a footnote to run whenever the topic is displayed, or in a hotspot, to run when the user clicks the hotspot. KLink and ALink topic searches are examples of ways to use macros in hotspots.

3.4.1 Help Macro Syntax

A help macro consists of a macro name and parameters enclosed in parentheses. Macro names specify the action to take, such as creating buttons or inserting menu items. Macro parameters specify the files, buttons, menus, or topics on which to carry out the action.

A help macro and all of its parameters cannot exceed 512 characters.

Help macros can be combined into macro strings by separating the macros with semicolons (;). The help compiler processes the macro string as a unit and execute the individual macros sequentially. Macros can also be used as parameters in other macros. In most cases, embedded macros must be enclosed in quotation marks.

Macro parameters must be enclosed in parentheses and separated by spaces. Parameters in many macros must also be enclosed in quotation marks. This is especially true if the parameter contains space characters.

Special characters must be preceded by a backslash (\). Special characters include double quotation marks ("), open and closed single quotation marks (‘ ’), and backslashes.

For example, the following code adds a menu item entitled “Yes \ No” to the “Options” menu:

```
AppendItem("MNU_OPTIONS", "Yesno", "Yes \\ No", "JumpHelpOn()")
```

In Windows, the valid quotation characters are the matching double quotation marks (" ") and the open and closed single quotation marks (‘ ’). When a using a pair of single quotation marks, the first one must be the open quotation mark and the second one must be the closed quotation mark.

NOTE: The single open quotation mark is different from the single closed quotation mark. The single open quotation mark (‘) is paired with the tilde (~) above the **TAB** key on extended keyboards; the single closed quotation mark (’) is the same as the apostrophe.

If a quotation character is needed as part of a parameter, the parameter should be enclosed in single quotation marks. When using single quotation marks in this manner, you can omit the backslash escape character for the double quotation marks, as shown in the following example:

```
‘command "string as parameter"’
```

If the embedded macro also has quoted parameters, the quotation character that is used must be different than the quotation characters enclosing the macro. The following examples examples show two correct ways to use nested quotation marks:

```
CreateButton("time_btn", "&Time", "ExecProgram(‘clock’, 0)")
```

```
CreateButton("time_btn", "&Time", ‘ExecProgram("clock", 0)’)
```

In both examples, the quotation marks enclosing the “clock” parameter is different from the type of quotation mark used to embed the macro “ExecProgram.”

In help topic files, you can define a help macro in two ways:

- As a hotspot in a jump, in which case the macro runs when the user clicks on the link text.
- As a control code in a footnote, in which case the macro runs whenever the help engine displays the topic.

3.4.2 Defining Macros in Jumps

You define help macros as jumps in much the same way as you define regular jumps. The example below shows a jump that runs a macro that calls the help file on how to use Windows Help.

NOTE: The examples given in this section and the next section are not from the Procedure Editor help file.



Figure 3–8: Help Macro in a Jump

To define a macro in a jump, follow these steps.

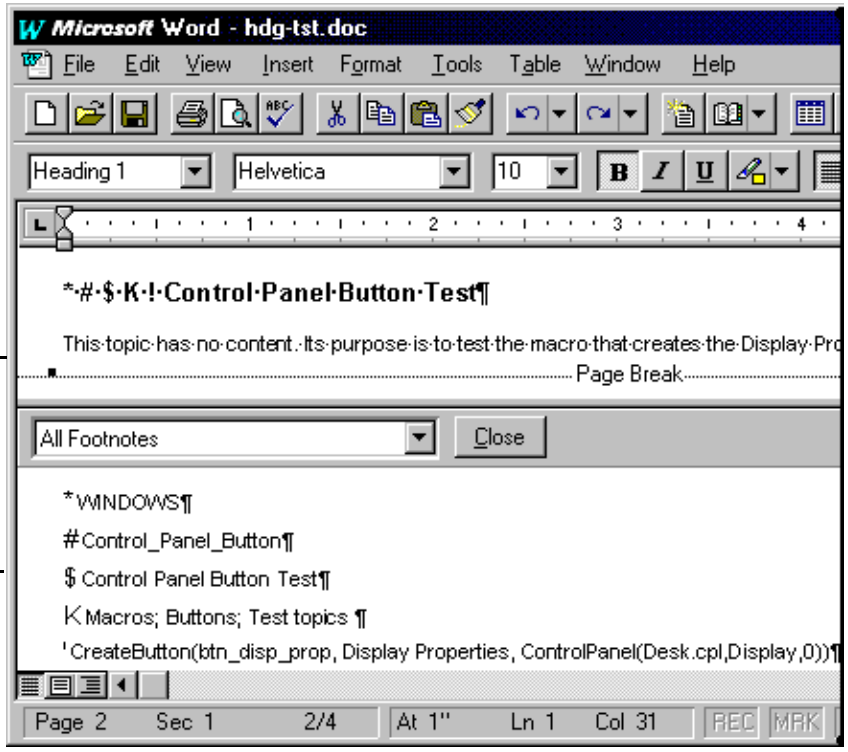
1 ♦ Decide which word, phrase, or graphic reference will serve as the link text (the visual cue that identifies the jump to the user).

2 ♦ Format the text or graphic reference as double underline (or strikethrough) text.

3 ♦ Enter an exclamation point (!) and the macro name as hidden text immediately following the link text.

3.4.3 Defining Macros in Control Codes

A help macro can also be entered as a control code, as shown in Figure 3–9. The macro adds a button labelled "Display Properties" to the help window's button bar. When the user clicks the Display Properties button, the help engine launches the Display applet with the first tab (tab 0) on top.



Help Macro Referenced with the Exclamation point

Footnote Character

Figure 3–9: Help Macro Viewed with MS-Word

You can reference more than one help macro in a control code as long as you separate each macro with a semicolon (;). For instructions on how to add a control code using MS-Word, see the "Inserting Control Codes" section in this chapter.

Each supported macro is explained in detail in the *Help Author's Guide* (`hcw.hlp`), the help file for Help Workshop.

3.5 Saving Help Topics in RTF Files

When you finish your topic text in MS-Word, you need to save the topics in one or more RTF files.

1 ♦ In MS-Word, select File→ Save As Type.

2 ♦ Choose Rich Text Format (*.rtf).

Chapter 4, "Completing Help Systems," explains how to create a contents file and a project file and then how to compile these source files into a help project.

4

Completing Help Systems

After you create your help topics, you need to create a help project file (.hpj) to direct the help compiler to compile the help source files into the finished help file (.hlp). You also create a contents file (.cnt) which, together with the compiled help file, constitutes the completed help project.

This chapter tells how to use Help Workshop to create contents files and help project files and to compile your help project.

This chapter includes the following sections:

- Creating a contents file
- Creating a project file
- Compiling a help project
- Testing a help system

4.1 Creating a Contents File

The *contents file* contains information that populates the Contents tab of the Help Topics dialog box as well as commands that are used by the help engine at run time. Contents files are plain text (ASCII) files that can be created with the Help Workshop (`hcw.exe`), or in a text editor.

The topic icons on the Contents tab of the Help Topics dialog box (icons resembling pages with question marks) represent links to individual help topics in the compiled help file. This type of link works in much the same way as a jump that you create in a help topic-it references the topic ID (context string) of the linked topic.

While you do not have to create the entire help topic file before creating the contents file, it is helpful to at least create those topics that will be the destinations of jumps from the Contents tab before you create the contents file. That way, the required topic IDs will already exist in the help topic file when you begin work on the contents file.

Strictly speaking, a contents file is not a required element of a help project. But if you create a help file and run it without a contents file, the Help Topics dialog box will display without a Contents tab.

The contents file (*.cnt) is an ASCII text file that contains the information that populates the Contents tab of the Help Topics dialog box. The contents file can also include commands that direct the help engine to do such things as display keywords from multiple help files on the Index tab or use a specified window type as a default.

If you are documenting a modular application with several help systems, you can use the contents file to combine the various help systems, each with its own contents file (.cnt) and compiled help file (.hlp). *Modular help* uses the contents file to present a unified help system to users with combined Contents and Index tabs and full text searching. Implementing modular help is beyond the scope of this manual; for more information, see the the *Help Author's Guide* or one of the many third party help development guides.

The elements of the contents file fall into three categories: commands, headings, and commands. The following sections discuss these elements in more detail.

4.1.1 Contents File Commands

Table 4–1 summarizes the commands you can use in contents files. Note that a contents file command is always preceded by a colon.

Table 4–1: Contents File Commands

Command (with syntax)	Direction to Help Engine
:Base *HLP-filename*[> *window-name*]	To use *HLP-filename* as the default help file to search for topics referenced by topic entries. If you specify *window-name*, the help engine will use that window type as the default window type for displaying help topics.
:Include *cnt-filename*	To find the contents file specified by *cnt-filename* and add its headings and topics to the Contents tab based on the position of the :Include command in this contents file. If the specified file is not found, no error message is generated.
:Index *help-title = HLP-filename*	To include the keywords from *HLP-filename* in the Index tab, and the unique text words from *HLP-filename* in the Find tab of the Help Topics dialog box. The help engine uses *help-title* when it displays a topic from *HLP-filename*.
:Link *HLP-filename*	To search in *HLP-filename* (as well as the default help file) for help topics during a KLink or ALink search.
:Tab *tab-name = DLL-name*	To add a custom tab to the Help Topics dialog box and call the .dll specified by *DLL-name* to determine its functionality.
:Title *title*	To use *title* in the title bar of the Help Topics dialog box, and in the title bar of any secondary window that does not specify a title.

4.1.2 Headings and Topics

Headings and topics are the elements that give the Contents tab its book-and-page appearance. In the contents file, each heading and topic is preceded by a number from 1 through 9, representing its level. Headings, represented by book icons, are used to categorize topics and lower levels of headings. Topics, represented by page icons, represent jumps to help topics, either as jumps, cross-file jumps, or KLink or ALink topic searches.

If a numbered line in the contents file contains only text, the help engine recognizes that line as a heading. If a line contains text followed by an equals sign and a macro or jump, the help engine recognizes that line as a topic.

You might want to make a topic icon appear directly on the Contents tab, so that the user does not have to double-click on a book icon to see it. To do this, place the topic at the beginning of the list of heading and topic entries in the contents file; you cannot place it after a book icon.

For examples of the appearance of heading and topic lines in a contents file, and the effect of their placement on the appearance of the contents tab, compare Figure 4–1 to the contents file text fragment immediately below it.

4.1.3 Creating a Contents File with Help Workshop

This section provides an orientation to creating contents files with Help Workshop by describing the elements unique to the contents file window. This section also illustrates the contents file window by showing you a portion of the Procedure Editor contents file as it appears in Help Workshop.

For a detailed description of the menu bar and tool bar of Help Workshop, consult the *Help Author's Guide* (`hcw.hlp`). Also note that the contents file window and all the dialog boxes associated with it use field-level help. Click in any field and press **F1** to see the help text for that field.

The area of the contents file window below the tool bar is divided into three sections. The topmost section contains two text areas and and Edit button. When you choose the Edit button, the Default Help Information dialog box appears. This dialog box has three text areas described below:

- **Default Help filename** — If you enter a filename in this area, Help Workshop creates a :Base command in the contents file.
- **Default window** — If you enter a window type in this area, Help Workshop adds this to the :Base command.
- **Default title** — If you enter text in this area, Help Workshop creates a :Title command.

Use the center section to create, arrange, and edit heading and topic entries. This section features a graphical display showing how your heading and topic entries will appear on the Contents tab, as well as the following buttons:

- **Edit** — Allows you to edit an existing entry. Select the entry in the graphical display area, then choose the Edit button.
- **Remove** — Removes a selected entry from the contents file.
- **Add Above/Add Below** — Make your first entry by clicking either the Add Above or Add Below button to display the Edit Contents Tab Entry dialog box. This dialog box features a radio set at the top that gives you the choice of creating a heading or topic entry, calling a macro, or including the entries from a second contents file (which embeds an :Include command in the contents file). Note that the lower portion of the dialog box changes appearance depending on which radio button you select.

  Insert new entries in the list by selecting an entry and choosing the appropriate Add button.
- **Move Right/Move Left** — The Move Right and Move Left buttons allow you to change the heading level of heading and topic entries.

The bottom section contains three buttons which are described below:

- **Tabs** — Displays the Custom Tabs dialog box which allows you to add custom tabs to the Help Topics dialog box. Each entry in the Custom Tabs dialog box creates a :Tab command in the contents file.
- **Link Files** — Displays the Help Link Files dialog box, which allows you to specify files to include in KLink and ALink searches. Each entry in the Help Link Files dialog box creates a :Link command in the contents file.
- **Index Files** — Displays the Index Files dialog box which allows you to specify help files whose keywords and text words are to be included in the Index and Find tabs of the Help Topics dialog box. Each entry in the Index Files dialog box creates an :Index command in the contents file.

Figure 4–1 shows a portion of the contents file for the Procedure Editor help file as it appears in Help Workshop.

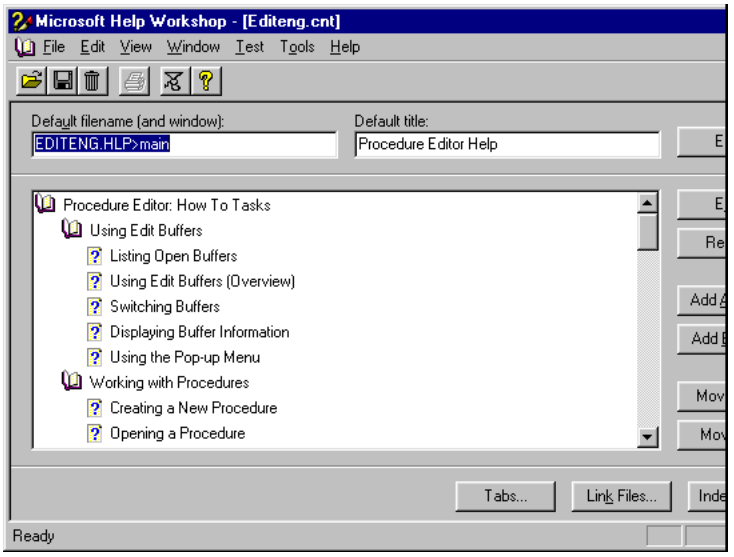


Figure 4–1: Help Contents File in Help Workshop

The text of the Procedure Editor contents file corresponding to the portion of the file shown in Figure 4–1 follows:

```
:Base EDITENG.HLP>main
:Title Procedure Editor Help
1 Procedure Editor: How To Tasks
2 Using Edit Buffers
3 Listing Open Buffers=Listing_Open_Buffers
3 Using Edit Buffers (Overview)=Using_Edit_Buffers
3 Switching Buffers=Switching_Buffers
3 Displaying Buffer Information=Displaying_Buffer_Information
3 Using the Pop-up Menu=Using_the_Pop_up_Menu
2 Working with Procedures
3 Creating a New Procedure=Creating_a_New_Procedure
3 Opening a Procedure=Opening_a_Procedure
```

4.2 Creating a Project File

A *project file* is a text file that contains a variety of information, such as the name of each source file, specifications for each window type, and copyright information, all of which, tells the help compiler how to build a help file from specified source files. Help project files must have a .hpj file extension.

A project file consists of sections, or categories, of compiler instructions. Before you create the project file, you need to analyze the help system to determine which sections of compiler instructions you need to include in the project file.

Table 4–2 tells you which project file section you'll need for each of several commonly-used help functions. By analyzing the considerations in the left-hand column, you can determine the minimum set of sections to include in your project file.

Table 4–2: Project File Considerations

Help System Consideration	Project File Section Required
Context-sensitive help calls	[MAP]
Graphics	[BITMAPS]
Table of Contents	[OPTIONS]
Special help viewer features, such as browse buttons	[CONFIG]
Secondary windows?	[WINDOWS]
Build tag (*) control code for conditional compile (build) options	[BUILDTAGS]

Note that Help Workshop creates an [OPTIONS] section by default. A particularly useful option in the [OPTIONS] section is COMPRESS, in which you can specify compression to considerably reduce the size of a help file.

4.2.1 Project File Sections

Table 4–3 summarizes the project file sections, explaining the function of each one. Some of the individual sections are explained in more detail later in this chapter.

Table 4–3: Summary of Project File Sections *(1 of 2)*

Section	Description
[ALIAS]	Assigns one or more topic IDs to the same help topic. This section is optional.
[BAGGAGE]	Lists files (typically multimedia elements) that the Windows Help compiler stores within the help file's internal file system. This section is optional.
[BITMAPS]	This section is recognized by the help compiler for backward compatibility only. It was formerly used to list graphic files to include in the build. The help compiler now uses the BMROOT option in the [OPTIONS] section to specify a directory where graphic files are located..
[BUILDTAGS]	The [BUILDTAGS] section defines the valid build tags for a help file. The compiler uses the build tags to determine which topics to include or exclude when building the Help file.
[CONFIG]	This section is used to specify author-defined menus and buttons to be used with the help file, and to register DLLs and DLL functions used as macros in the help file that define non-standard menus, buttons, and macros used in the help file. This section is required if the help file uses any of these features.
[CONFIG:*name*]	This section is used in the same way as the [CONFIG] section, except that the content of the section only applies to the window type indicated by *name*.
[EXCLUDE]	This section specifies build tags that identify topics that should be excluded from the help file.
[FONTS]	This section specifies the fonts in topic files that are to be replaced with different fonts, point sizes, or character sets.
[FILES]	Specifies the help topic files to include in the build. This section is required.
[INCLUDE]	This section specifies build tags that identify topics that should be included in the help file.
[MACROS]	This section specifies macros to run whenever a user selects one of the keywords (also listed in this section) from the Index tab of the Help Topics dialog box.
[MAP]	Associates topic IDs with context IDs. This section is optional.

Table 4–3: Summary of Project File Sections *(2 of 2)*

Section	Description
[OPTIONS]	Specifies options that control the build process. This section is required, and should appear first in the project file, so the options will apply during the entire compilation.
[WINDOWS]	Defines the size, location, and colors for the main help window and any secondary window types used in a help file. This section is required if the help file uses secondary windows.

The following sections describe the various sections of the help project file and use the project file for the Procedure Editor help file, (`editeng.hpj`) as an example. (See the "Accessing Files in Procedure Libraries" section in the Preface for information on how to access this project file.)

[ALIAS] Section

The [ALIAS] section of the help project file specifies alternate names for topic IDs. The [ALIAS] section is optional.

Normally, if you combine, remove, or replace any of your help topics, you have to manually update all the affected topic IDs to maintain the intended hypertext functions. By including an [ALIAS] section in the help project file, you can tell the help compiler to substitute a newer topic ID for a topic ID that you no longer want to use.

For instance, if you combined the Cutting_Code, Copying_Code, and Pasting_Code help topics into a single new help topic named Drag_and_Drop, you could automatically update the hypertext references by adding the following statements to the [ALIAS] section in the help project file:

```
[ALIAS]
Cutting_Code=Drag_and_Drop
Copying_Code=Drag_and_Drop
Pasting_Code=Drag_and_Drop
```

If you include this [ALIAS] section in the help project file, the help compiler links all references to any of the three help topics to the new replacement help topic Drag_and_Drop.

The [ALIAS] section can also be used to resolve compiler errors if two or more topic IDs are assigned to the same help topic. For example, if one help author assigns the topic ID Cut_and_Paste to an Edit menu topic and another author uses Drag_and_Drop to reference the same help topic, an error would occur. To fix the problem, simply include the following statement in the [ALIAS] section:

```
[ALIAS]
Cut_and_Paste=Drag_and_Drop
```

[BAGGAGE] Section

The [BAGGAGE] section lists files (typically multimedia elements) that the help compiler stores within the help file's internal file system. The help engine can access data files stored in the help file more efficiently than it can access files in the normal file system, since it doesn't have to read the file allocation table:

```
[BAGGAGE]
meep.wav
```

If a file is listed in the [BAGGAGE] section, you refer to that file in the topic file by prefixing the filename with an exclamation point (!). The filename must appear exactly the same in the topic file as it appears in the [BAGGAGE] section. To avoid having to specify a full path, use the ROOT option in the [OPTIONS] section to specify the path. All filenames that you specify in the topic file are relative to the ROOT path.

This example is from the [BAGGAGE] section of the Procedure Editor help project file:

```
[BAGGAGE]
[Map]
#include <EDITENG.HH>
```

[BITMAPS] Section

The [BITMAPS] section of the help project file is no longer used by the help engine. If you are using an old help project file that contains a [BITMAPS] section, the help compiler will ignore it.

By default, the compiler looks for graphics files in the same directory as the help project file, then in the same directory as the .rtf file that specified the graphics, and then in all directories specified by the ROOT option. The BMROOT option is necessary only if graphics files are not in one of these locations. You can specify multiple BMROOT options.

[BUILDTAGS] Section

The [BUILDTAGS] section specifies all valid build tags for a particular build.

This section is used in conjunction with the build tag footnote (*) in help topic files, and the BUILD option in the [OPTIONS] section of the help project file. The build tag footnote associates a particular build tag with a given topic.

The help compiler processes those help topics assigned a build tag plus any help topics that contain no build tag. The compiler ignores only help topics containing build tags that are not defined in the project file's [BUILDTAGS] section.

If you do not include a [BUILDTAGS] section in your project file, the help compiler processes all the help topics in all the topic files listed in the [FILES] section. The [BUILDTAGS] section is optional.

Note that Help Workshop does not provide a way to modify the [BUILDTAGS] section in your project file. To create or modify a [BUILDTAGS] section, load the project file into a text editor, and then add or edit the section.

Also note that Help Workshop does not provide a way to specify the BUILD option in your project file. Instead, it provides a Build Tags tab in the Options dialog box that enables you to specify what build tags to include and exclude. These build tags are placed in the [INCLUDE] and [EXCLUDE] sections respectively. If there is an [INCLUDE] or [EXCLUDE] section in your project file when you compile a file, Help Workshop ignores any BUILD option you may have specified.

To use a BUILD option instead of an [INCLUDE] or [EXCLUDE] section, you must load the project file into a text editor and add the option to the [OPTIONS] section. You must also add a [BUILDTAGS] section specifying all the build tags that you will be using:

```
[BUILDTAGS]
WINDOWS
```

[CONFIG] Section

The [CONFIG] section of the help project file specifies which help macros are executed when the help engine first loads the help file.

This section can contain macros that carry out actions, such as creating buttons or menus, and macros that register routines in external DLLs as WinHelp macros. These routines can then be used the same as WinHelp macros.

The [CONFIG] section is optional.

A *help macro* is a predefined routine that executes a specific action, such as adding, modifying, or removing buttons and/or menu options displayed in the help viewer. In the following example, browse buttons are enabled in the help viewer:

```
[CONFIG]
BrowseButtons()
```

For detailed information about the function, parameters, and correct syntax for each macro, see the *Help Author's Guide*, `hcw.hlp`, the help file for Help Workshop.

[FILES] Section

The [FILES] section of the help project file specifies all the help topic files (`.rtf`) you want the help compiler to process when creating the help file. A help project file must have a [FILES] section.

If a file is not on the defined path and cannot be found, the help compiler generates an error message.

[MAP] Section

The [MAP] section of the help project file supports context-sensitive help. The [MAP] section is optional. See the "Mapping Context-sensitive Help Topics" section for more information on this process.

[OPTIONS] Section

The [OPTIONS] section includes options that control how a help file is compiled and what feedback the compilation process displays. This section, along with the [FILES] section, is required in all project (.hpj) files. The [OPTIONS] section should be placed first in the project file so that the specified options apply during the entire compilation process.

Options are specified using the following syntax:

SYNTAX

```
option name = status of option
```

The following example shows the [OPTIONS] section of the Procedure Editor help project file:

```
[OPTIONS]
BUILD=WINDOWS
CNT=editeng.cnt
TITLE=Procedure Editor Help
COPYRIGHT=Copyright c 1999 Progress Software Corp.
OLDKEYPHRASE=NO
REPORT=NO
OPTCDROM=0
DBCS=0
NOTES=1
COMPRESS=12
BMROOT=..\gfx
ERRORLOG=M:\checkin\EDITENG\EDITENG.err
```

For a complete of [OPTIONS] section options, see Appendix B, Appendix B, "Options in the [Options] Section of the Project File."

[WINDOWS] Section

The [WINDOWS] section defines the size, location, and colors for the main help window and any secondary window types used in a help file. You can define up to 255 secondary window types (although the help engine can display no more than nine secondary window types at one time).

You can use window definitions established in the [WINDOWS] section with the SYSTEM-HELP statement or as a part of the hypertext link definition to display help information in a customized secondary help window. For more information about the SYSTEM-HELP statement, see the "The SYSTEM-HELP Statement" section in Chapter 1, "Providing Help for Progress Applications." For information about window redirection in a hypertext link definition, see the"Creating Secondary Window Links" section inChapter 3, "Creating Help Topic Files."

You can also associate a window type with a topic in the help topic file by specifying the window type in a footnote coded with the greater-than sign (>). This footnote type specifies the window type a topic will appear in when the topic is opened from the Index or Find tab, or from an ALink or KLink macro.

Help Workshop provides a useful graphical interface for creating or editing the [WINDOWS] section. It is much easier to use Help Workshop to define window types, rather than using a text editor.

For detailed reference information on defining window types, see the *Help Author's Guide*, `hcw.hlp`, the help file for Help Workshop.

4.2.2 Mapping Context-sensitive Help Topics

Context-sensitive help relies on the [MAP] section, where the help author pairs a context ID with a topic ID for a specific topic. A *context ID* is a unique number that the help author or application programmer assigns to each application context in the Progress application that requires online help. An *application context* is any currently displayed window, dialog box, frame, button, field-any user interface element-that has input focus.

When the user requests context-sensitive help from a running application, the help calling interface passes the context ID to the help engine which then displays the topic associated with the topic ID.

Figure 4–2 shows the simplest way to pair a help topic's topic ID with its assigned context ID in the [MAP] section.

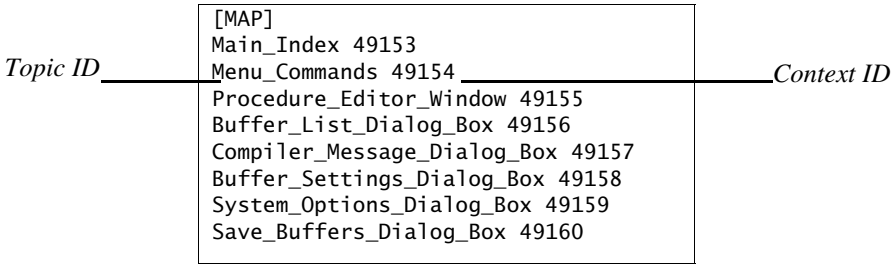


Figure 4–2: [MAP] Section Example

The Help button in the Procedure Editor Buffer List dialog box is coded to request the topic whose context ID is 49156. When a user clicks the Help button, the help engine immediately displays the help topic whose Topic ID is Buffer_List_Dialog_Box, based on the [MAP] section entry shown inFigure 4–2.

In addition to the method shown in Figure 4–2, you can use one of the following methods to specify context-sensitive help topics in the [MAP] section of the help project file:

Method 1—Use the C language `#define` directive to assign context IDs to topic IDs which are also defined in a C include file. (In the example below, the context ID is in hexadecimal form. The special characters /* and accompanying text are C language comments.)

```
[MAP]
#define Main_Index 0x000                    /* Contents topic */
#define Menu_Commands 0x001                 /* Menu Commands topic */
#define Procedure_Editor_Window 0x002       /* Main Editor window */
#define Buffer_List_Dialog_Box 0x003        /* Buffer List dialog */
#define Compiler_Message_Dialog_Box 0x004   /* Compiler Message dialog */
#define Buffer_Settings_Dialog_Box 0x005    /* Buffer Settings dialog */
#define System_Options_Dialog_Box 0x006     /* System Options dialog */
#define Save_Buffers_Dialog_Box 0x007       /* Save Buffers dialog */
```

NOTE: If context IDs use the `#define` directive, and the file that contains the `#define` statements is included in both the application code and the help file, then updates made to the context IDs are automatically reflected in the next help build.

Method 2—Use the C language `#include` directive to tell the help compiler to link a separate header file that contains the context-sensitive mapping information. The filename of the included file must be enclosed in double quotes (" ") or angle brackets (< >).

This example is from the [MAP] section of the Procedure Editor help project file:

```
[MAP]
#include <EDITENG.HH>
```

The format of the header file should be either that shown in Method 1 or Method 2.

Be careful to follow these conventions when creating the [MAP] section:

- Use either decimal or hexadecimal numbers formatted in standard C notation to specify context IDs.
- Assign no more than one context ID to a topic ID or alias. Assigning the same context ID to more than one topic ID will generate a compiler error.
- You can use the C `#include` directive to include other files that map the context-sensitive help topics. In addition, the [MAP] section supports an extended format that lets you include C files with an H extension directly.
- As in other project file sections, use the semicolon (;) to begin each line of comment text in the [MAP] section. Entries using the C `#define` directive may contain comments in C format, as in the following example:

```
[MAP]
#define topic_ID context_ID /* comment */
```

4.2.3 Project File Conventions

The next section shows you how to create a project file using Help Workshop. If you choose to create your project files in a text editor, make sure you observe the following conventions:

- All project filenames must have an .hpj extension.
- Each section must begin with a section name that is enclosed in brackets ([ ]).
- All project files require a [FILES] section, which the help compiler uses to identify the help topic files to include in the build, and an [OPTIONS] section, which specifies options that control the build process.

NOTE: To compile the help project (.hpj) file, the help compiler must know where to find the help source files. By default, the compiler looks for help source files in the directory where the project file resides. For help topic files that reside in another directory, you should use the ROOT option in the [OPTIONS] section. For graphic files that reside in another directory, you should use the BMROOT option the [OPTIONS] section of the project file. See the "[OPTIONS] Section" section for more information.

- Sections can appear in any order, except that the [OPTIONS] section should appear first and the [ALIAS] section must precede the [MAP] section if both of these sections are present.

- To insert a comment, use a semicolon to precede the comment text. The help compiler ignores all text from the semicolon to the end of the line in which it occurs.

4.2.4 Creating a Project File with Help Workshop

This section provides an orientation to creating help project files with Help Workshop by describing the elements unique to the project file window. This section also illustrates the project file window by showing you the full text of the Procedure Editor project file as it appears in Help Workshop.

Help Workshop, `\Program Files\Progress\bin\hcw.exe`, is supplied with Progress. It is a utility for creating help contents files and help project files, and for running the help compiler, `\Program Files\Progress\bin\hcrtf.exe`.

For complete reference information on all aspects of online help development including compiler notes, warnings, and error messages, see the *Help Author's Guide*, an online help file, `hcw.hlp`,that accompanies the Help Workshop. The *Help Author's Guide* also includes a detailed description of the menu bar and tool bar of Help Workshop.

The project file window and all the dialog boxes associated with it use field-level help. Click in any field and press **F1** to see the help text for that field.

The project file window displays two text areas and a vertically arranged set of buttons. (Refer to Figure 4–3 for the placement of these elements.) The small text area, labeled Help File, displays the directory pathname and file name to use for the compiled help file. You can edit this text area directly, or on the Files tab of the Options dialog box, as described below.

The large text area displays the text of the project file. As you create new project file entries using the dialog boxes associated with the project file window buttons, Help Workshop inserts them into this display area. If you want to edit an existing entry, you can double-click on it in this display area, and Help Workshop will launch the appropriate dialog box (with the correct tab on top, where applicable).

The remainder of this section lists the buttons that appear in the project file window and describes the dialog boxes they launch:

- **Options** — Launches the Options dialog box, consisting of a number of tabbed pages. Note that not all of the entries you make in the Options dialog box end up in the [OPTIONS] section of the project file.

- **Files** — Launches the Topic Files dialog box, which allows you to specify the `.rtf` files that contain the help topics for the project.
- **Windows** — Launches the Window Properties dialog box, consisting of a number of tabbed pages that allow you to create and set properties for all the window types required for your project. Defining window types in the Window Properties dialog box is very convenient and straightforward compared to defining window types in a text editor.
- **Bitmaps** — Launches the Bitmap Folders dialog box, which allows you to specify a folder (directory), other than the project file's directory, where you have stored graphics files.
- **Map** — Launches the Map dialog box, which allows you to create entries mapping topic IDs to context IDs for context-sensitive calls into the help file. Note that if you want to use an #include statement to reference an external file containing context mappings, you cannot do this using Help Workshop. You have to insert an #include statement manually, using a text editor.
- **Alias** — Launches the Topic ID Alias dialog box, which allows you to create topic ID aliases.
- **Config** — Launches the Configuration Macros dialog box, which allows you to specify macros that run whenever a user opens a help file. For example, if you wanted to add optional buttons or menu items for the main window, you would enter the appropriate macros here.
- **Data Files** — Launches the Files Used by Help DLLs dialog box, which allows you to specify the names of author-created DLLs that serve to extend the help engine's internal macro set.
- **Save and Compile** — This is the only button of this group of buttons that does not launch a dialog box. The Compile button on the tool bar, however, does launch the Compile a Help File dialog box, which allows you to set several options that affect the behavior of Help Workshop during the compilation.

Figure 4–3 shows the Procedure Editor help project file as it appears in Help Workshop.

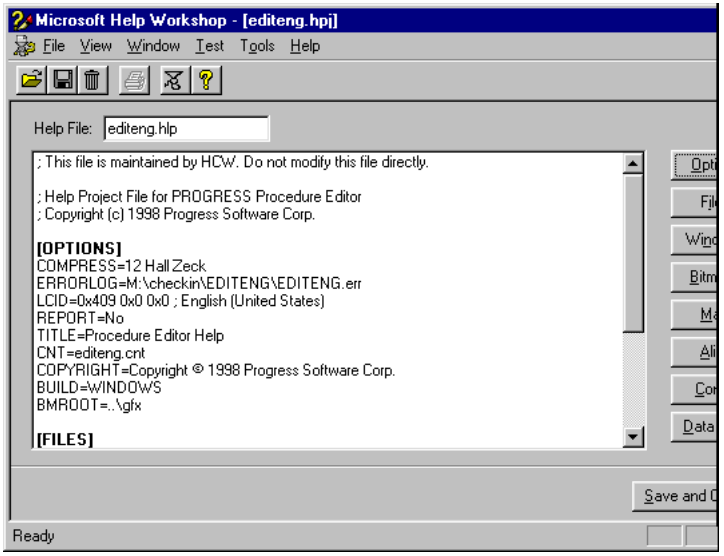


Figure 4–3: Help Project File in Help Workshop

4.3 Compiling a Help Project

The *help compiler* converts the help source files using the instructions in the help project file into a binary help file that can be read by the Windows help application, `winhlp32.exe`, supplied with Windows. Help Workshop (`\Program Files\Progress\bin\hcw.exe`) runs the Windows help compiler (`\Program Files\Progress\bin\hcrtf.exe`). While it is possible to run the help compiler directly, it is more convenient to use Help Workshop to run the help compiler.

Table 4–4 summarizes the various help source files.

Table 4–4: Help Source Files

File	Extension	Purpose
Topic files	.rtf	Topic files are RTF files containing all of the text that will be displayed in the compiled help file. The help topic files also contain special control codes and text formatting that define topics and create hypertext links.
Graphics files	.bmp, .shg, .wmf, .mrb	Graphics files are included in help topic files by referencing their filenames with help statements.
Contents file	.cnt	The contents file is a text file containing information used by the help engine to determine the behavior of the Help Topics dialog box and to populate the Contents tab of the Help Topics dialog box.
Project file	.hpj	The project file is a text file that organizes the source files and contains all of the information the compiler needs to create the binary help file. The help project file lists all the files (RTF, CNT, and graphics files) and includes any special information required to configure the help viewer.

When you finish specifying all the help topic files, graphic files, build tags, macros, and other options you want to include in the help project file, you are ready to build the final help file. The help compiler processes all the help source files listed in the help project file and generates the compiled help file.

By default, the help compiler places the finished help file in the same directory as the project file, and assigns it a name based on the name of the project file, but with an *.hlp extension. The simplest way to compile a help file is to use Help Workshop in graphical mode, accepting all the compilation defaults.

Figure 4–4 illustrates the process of compiling a help project.

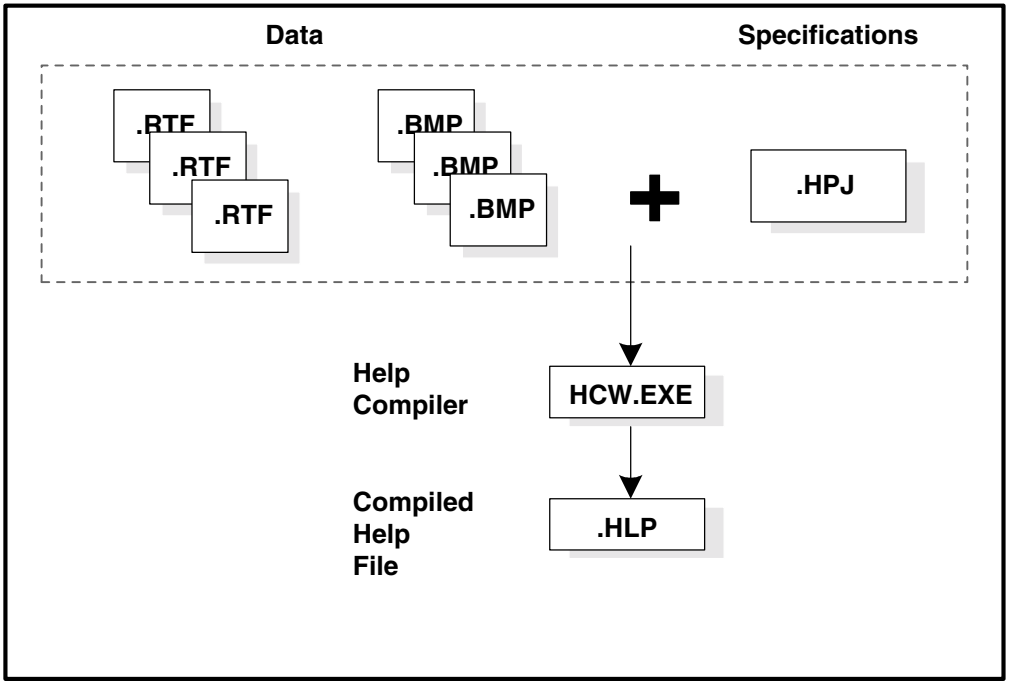


Figure 4–4: Compiling a Help Project

Follow these steps to compile a help project with Help Workshop:

1 ♦ Execute `hcw.exe` from the `Program Files\Progress\bin` directory.

2 ♦ From the main menu, choose File→ Open. The Open dialog box appears.

3 ♦ In the Open dialog box, select Help Project Files (.hpj) in Files of Type, and browse for your project file. Choose the Open button.

Help Workshop opens the project file and displays it as shown in Figure 4–3.

4 ♦ Choose the Compile button in the tool bar. Help Workshop runs the compiler. By default, Help Workshop minimizes while compiling.

5 ♦ When compilation is complete, Help Workshop places the compiled file in the same directory as the project file (or other directory if specified in the project file).

Help Workshop displays a Compilation window and populates it with compiler messages and any notes, warnings, or errors that are generated by the compiler. If you want to save this compilation information in a log file, choose File→ Save when the Compilation window is displayed. You can also instruct the compiler to save its messages to a file by using the ERRORLOG option in the [OPTIONS] section of the project file.

To prevent Help Workshop from minimizing while compiling, or to change other compilation preferences, choose File→ Compile from the main menu instead of the Compile button in Step 4 above.

4.3.1 Interpreting Compiler Messages

While compiling the source files, the help compiler always generates progress messages (like "Resolving browse sequences") and statistical information about the compilation process.

In addition to this information, it might display messages that fall into one or more of the following categories:

- Notes, which are conditions you should be aware of but which will probably not cause serious problems in the help file.
- Warnings, which are conditions that will not prevent the help file from compiling, but will result in a defective help file.
- Errors, which are conditions that prevent the help file from compiling.

See the *Help Author's Guide* (`hcw.hlp`) for detailed information on all notes, warnings and errors. They are indexed by number on the Index tab of the Help Topics dialog box.

Whenever possible, the compiler displays the name of the file that contains the error, as well as the number used to identify the specific line of the project file or the topic that produced the error. Since topics are not actually numbered, the topic number supplied with each error message refers to that topic's sequential position in the topic file.

4.3.2 Running the Compiler from the Command Line

You can also compile help files by running either Help Workshop or the help compiler itself from the DOS prompt in an MS-DOS window. For more information, see the HCW Command and the HCRTF Command-line Switches topics in the *Help Author's Guide* (`hcw.hlp`).

4.4 Testing a Help System

The final step in providing help for an application is to test the help system. The following items should be included in your plan for testing your help system:

- Test context-sensitive help calls from the application.
- Test the help menu.
- Verify that the help information is accurate.
- Look for user interfaces, instructions, and concepts that are not documented.
- Test the contents page links.
- Test all jumps and popups
- Test the index.
- Run reports. (Choose File→ Report in Help Workshop.)

For more information on testing, see the topic, Features for Testing Help Files, in the *Help Author's Guide* (`hcw.hlp`) . This topic also contains links to information on Help Author mode, an additional feature which facilitates analyzing and testing help topics.

A

Files Mentioned in This Guide

Table A–1 lists the names, locations, and descriptions of the files mentioned in this guide.

Table A–1: Files in This Guide *(1 of 2)*

File	Location	Description
applhelp.p	\Program Files\Progress\gui	Progress help calling interface
applhelp.p	\Program Files\Progress\src\samples\	Progress sample help calling interface
editeng.cnt	\Program Files\Progress\prohelp\	Progress Procedure Editor help contents file
editeng.hlp	\Program Files\Progress\prohelp\	Progress Procedure Editor help file
editeng.hpj	Packed file in prodoc.pl, in \Program Files\Progress\src	Progress Procedure Editor help project file
hcw.exe	\Program Files\Progress\bin\	Microsoft Help Workshop
hcw.hlp	\Program Files\Progress\bin\	Microsoft *Help Author's Guide*
hcrtf.exe	\Program Files\Progress\bin\	Microsoft Help Compiler

Table A–1: Files in This Guide *(2 of 2)*

File	Location	Description
hoh.hlp.	\Program Files\Progress\prohelp\	Microsoft How to Use Help file (16 bit)
r-syshlp.p	Packed file in prodoc.pl, in \Program Files\Progress\src	Sample Progress help calling interface
shed.exe	\Program Files\Progress\bin\	Microsoft Hotspot Editor (SHED)
shed.hlp	\Program Files\Progress\bin\	Microsoft Hotspot Editor help file
winhlp32.hlp	Supplied with Windows	Microsoft How to Use Help file (32 bit)
winhlp32.exe	Supplied with Windows	Windows help engine (32 bit)

B

Options in the [Options] Section of the Project File

Table B–1 describes the options in the [OPTIONS] section of a help project file.

Table B–1: [OPTIONS] Section Options *(1 of 3)*

Option (with syntax)	Description
BMROOT=*pathname*	Where graphic files are stored. Use this option if any graphic files are stored in a different directory from the directory that contains the help project file.
BUILD=*expression*	What topics to include in the build. This option works exactly like the [BUILDTAGS] section of the project file. Note that if you use this option, you must also include the build name in the [BUILDTAGS] section.
CHARSET=*charset-value*	The default character set to use for the fonts in the help file.
CITATION=*text*	To include a citation in the help file, that will be appended to the end of any information that is copied for printed from the help file. This option is commonly used to attach a copyright notice to information copied or printed from your help file.
COMPRESS=*compression-level*	To compress the generated help file. See the *Help Author's Guide* (`hcw.hlp`) for a list of possible values for *compression-level*.

Table B–1: [OPTIONS] Section Options *(2 of 3)*

Option (with syntax)	Description
CONTENTS=*topic-id*	The topic identified by *topic-id* is the default topic for the help file. This topic is displayed by default if the help engine encounters a broken jump or if the contents file associated with the help file is missing.
CNT=*contents-file*	The name of the contents file to associate with the help file.
COPYRIGHT=*copyright-notice*	Adds a custom copyright message for the help file to the Version dialog box.
DBCS=*value*	Indicates that the source files use a double-byte character set.
DEFFONT=*fontname,font-size,charset*	The default font for the lists that appear on the Contents, Index, and Find tabs of the Help Topics dialog box and in the Topics Found dialog box.
ERRORLOG=*error-filename*	To write all error messages to the specified file.
FORCEFONT=*fontname*	To force all authored fonts in the topic files to appear in the specified font when displayed in a help window. See the *Help Author's Guide* (`hcw.hlp`) for a list of possible values for *fontname*.
FTS=*fts-value*	The type of index file to create for the full-text search feature represented by the Find tab of the Help Topics dialog box.
HCW=*value*	This option is reserved for internal use by the help compiler.
HLP=*help-file*	The name of the compiled help file.
INDEX-SEPARATORS=*"separators"*	What characters to recognize as delimiters for keyword entries.
LCID=*language* *case-insensitive* *case-sensitive*	What language the help file uses.
MAPFONTSIZE=*m*[*-n*]*:p*	Changes the font size in the compiled version to ensure readability at different video resolutions.

Table B–1: [OPTIONS] Section Options *(3 of 3)*

Option (with syntax)	Description
MULTIKEY=*footnote-character*	The control characters used to identify alternate keyword tables.
NOTES=[*0/1/yes/no*]	To display notes while compiling the help file.
OLDKEYPHRASE=*value*	To use the existing key-phrase table or create a new one during the build. The OLDKEYPHRASE option can be assigned one of the following parameters: No (create new table) or Yes (use existing table). The default is No.
REPLACE= *old-prefix=new-prefix*	To replace pathname prefixes with a new prefix. By using this option you can avoid editing pathnames elsewhere in the project file, or in included files, if any source files move to a different server or folder.
REPORT=[*yes/no*]	To display messages while compiling the help file. These messages indicate when the help compiler is performing the different phases of the build, including compiling the file, resolving jumps, and verifying browse sequences.
ROOT=*pathname*	Where to look for the help topic files.
TMPDIR=*folder*	The name of a folder to use for storing temporary files created while compiling the help file.
TITLE=*titlename*	The title of the help file. The help engine displays the text specified in *titlename* in several places: The title bar of the Help Topics dialog box, if a title is not specified in the contents file. The title bar of the main help window whenever it displays the help file if a caption has not already been defined for it and no title was specified in the contents file. The Topics Found dialog box, if you have set up your keyword jump to display the title of the Help file with the topic title. The Version dialog box if the COPYRIGHT option was not specified in the project file.

Glossary

ALink

A hypertext link that locates target topics based on A-keywords (keywords stored in A footnotes) rather than context strings. ALinks are typically used to find a set of related topics, and are often associated with a Related Topics button in a help topic.

Application Context

Any currently displayed user-interface element that has input focus, for example, a window, dialog box, browse, button, combo-box, control frame, editor, fill-in, radio-set, selection-list, slider, and toggle box. The application context is the basis for context-sensitive help. (See also **Context-Sensitive Help, Context ID**.)

Browse Sequence

A series of related help topics that are designed for the user to access in a sequential order. When viewing a help topic defined as part of a browse sequence, the Browse buttons (<< and >>) are active, and the user can click them to see the previous or next topic. You define browse sequences in help topics with the plus sign (+) control code. Each help topic in a browse sequence has a sequence name and a unique sequence number. (See also **Control Code, Sequence Name, Sequence Number.)**

Contents File

An ASCII file containing information used by the help engine to populate the Contents tab of the Help Topics dialog box, as well as any or all of six contents file commands that determine the behavior of the Help Topics dialog box.

Context ID

A number assigned to an application context and the corresponding help topic. The help author maps context IDs to topic IDs in the [MAP] section of the help project file to provide context-sensitive help for a user interface. (See also **Application Context, Context-Sensitive Help, Help Project File, Topic ID.)**

Context-sensitive Help

Information designed to help users understand their current status or position in the context of an application. It is a quick access method, often used to help a user interact with a current dialog box, frame, screen, or window. (See also **Application Context, Context ID. Contrast with Reference Help.**)

Context String

See **Topic ID.**

Control Code

An MS-Word footnote that the help compiler uses to serve a particular purpose within the help system. For example, the help compiler includes all the keywords included in K control codes in the Index tab of the Help Topics dialog box.

Cross-file Link

A jump or popup link that, when selected, displays a help topic that is located in a different help resource file.

Destination Topic Identifier

The part of the hypertext link hidden from the user that informs the help engine which topic(s) to find when the user activates the link. A destination topic identifier is the topic ID of a particular help topic or a KLink or ALink macro.

Event

A keyword representation of a user or system action in Progress. An example of a user event is a keystroke.

Graphics Files

Files that contain images that are compiled along with help topic files into the finished help file.

Help Calling Interface

The Progress code that allows users to request and receive help information using a help keystroke, a help button, or a help menu. The calling interface invokes the help engine and directs it to display the specified help information.

Help Compiler

A Windows-supplied program that converts help source files into a finished help file using the instructions in the help project file.

Help Engine

An application (`winhlp32.exe`) designed to display help windows, populate them with help topics, and navigate help topics.

Help File

A binary file that contains the compiled help information for the help engine to read and display. Help files have an .hlp file extension.

Help Source Files

Files that contain the help information for a help system. Help source files consist of help topic files and graphics files.

Help Statements

An extended subset of tokens defined by the rich text format (RTF) standard. When you author help topic files in MS-Word, and save the file as RTF, the necessary help statements are automatically generated for you.

Help String

A string that appears in the status bar of a window and describes the function of the Progress field-level widget that has input focus. You define help strings for database fields in their schema definitions. You define help strings for field-level widgets, using the HELP option in the Format phrase of certain 4GL language statements.

Help System

A multi-component system that delivers large units of help information to application users and provides tools that allow users to navigate through help information.

Help Topic

A concise, modular piece of information about a single subject. A description of a dialog box in your application, a definition of a term, a set of instructions to complete a task, or a description of a 4GL language element are examples of help topics. In the help topic file, each help topic begins with one or more control codes and ends with a page break.

Help Topics Dialog Box

A dialog box that serves as “home base” for a help file. The Help Topics dialog box includes, by default, three tabbed pages labeled Contents, Index, and Find. The Contents tab serves as a table of contents for the help file, and displays only if there is a contents file that contains at least one valid topic entry (hypertext link). The Index tab lists the keywords contained in the K footnotes of the help file. The Find tab provides a full-text search feature. Users typically access the Help Topics dialog from the Help menu in an application window or from the Help Topics button in a help window. The Help Topics

dialog box can be customized by including one or more contents file commands in the contents file.

Help Window

A window the characteristics of which are defined in the help project file, created by the help engine to display a help topic. One or more window types can be defined in the project file. Every help project must define at least one widow type, named "main." All other window types are called secondary windows (which see).

Help Workshop

A Microsoft Windows utility, `\Program Files\Progress\bin\hcw.exe`, for creating help files, creating and editing project and contents files, and testing and reporting on help files. Help Workshop uses information in the project (.hpj) file to compile the topic (.rtf) files and graphics files into a binary help file (.hlp). The *Help Author's Guide*, `hcw.hlp`, provides online help for Help Workshop.

Hotspot

The origin of a hypertext link. It usually refers to a rectangular region of a segmented hypergraphics file (.shg) that contains a hypertext link. (See also **Hypertext Links, Segmented Hypergraphics File.)**

Hypertext Link

Text or a graphic that is coded to connect help information in a non-linear fashion, causing the help engine to display a related topic when the user selects the link. Hypertext links consist of two parts: **Link Text**, which is formatted to cue the user to recognize it as such, and a **Destination Topic Identifier** that informs the help engine which topic(s) to find when the user activates the link. (See also **Jump, Hotspot, Popup Link.**)

Input Focus

The interface element that will receive the next user event (input), provided the user does not navigate to another element.

Jump

Text or graphics that, when selected, cause the help engine to display a specified related help topic. Jumps in text appear in help windows with a solid underline. (See also **Segmented Hypergraphics File**. Contrast with **Popup Link**.)

Keyword

A word or phrase associated with one or more help topics. You can search for information in the Index tab of the Help Topics dialog box using keywords. You use the K control code to associate one or more keywords with a help topic. (See also **Control Code**.)

KLink

A hypertext link that locates target topics based on index keywords (keywords stored in K footnotes) rather than context strings.

Link Text

Text or a graphic reference formatted to serve as a visual cue to the user identifying it as a hypertext link.

Map ID

See **Context ID, Topic ID.**

Non-scrolling Region

A help window feature that continuously displays the header region of a help topic while the user scrolls through the topic. The help author can determine the size of this region, and turn the feature on or off. The non-scrolling region is controlled by the Keep With Next paragraph attribute in MS-Word. If a popup topic defines a non-scrolling region, only the non-scrolling region will appear when the topic is displayed.

Placement By Reference

A technique for including graphic files in a help topic file, by specifying the filename and alignment of the graphic in the help topic file, as opposed to embedding the graphic directly in the topic file.

Popup Link

Text or graphics that when selected, cause the help engine to display text or graphics in a smaller, overlapping window. The help viewer displays pop-up links in text with a dotted underline. (See also **Segmented Hypergraphics File**. Contrast with **Jump.**)

Project File

A text file that contains instructions that tell the help compiler how to build a help file from a specified set of help source files. Help project files must have an `.hpj` file extension.

Reference Help

Reference help provides general technical and usage information about an application. Reference information is not designed for a specific application context. The primary access for reference help is through the Help Topics dialog box. (Contrast with **Context-Sensitive Help**.)

Secondary Window

A help viewer window in which help topics are displayed. Unlike the main help window, a secondary window cannot contain a menu bar; however, the help author can control a

secondary window's buttons, style, and placement in the [WINDOWS] section of the help project file. (See also **Window Redirection**.)

Segmented Hypergraphics file

A graphics file that contains one or more hotspots. You can use the Hotspot Editor (`shed.exe`), to add hotspots to a standard Windows bitmaps. Segmented hypergrahics files have a .shg extension.

Sequence Name

The name the help compiler uses to distinguish different browse sequences in a help resource file. (See also **Browse Sequence, Sequence Number**.)

Sequence Number

A unique alphanumeric character that the help author assigns a help topic in a browse sequence. The help compiler uses the sequence number to do an ASCII (ascending) sort of all help topics defined in a particular browse sequence. For example, when the user selects the next help topic in a browse sequence, the help viewer displays the help topic with a higher sequence number. (See also **Browse Sequence, Sequence Name.**)

Status Area

An optional feature of a Progress window that displays one line of message text at the bottom of the window. Progress controls its appearance with the STATUS-AREA attribute.

Topic ID

A unique character string that identifies each topic in the help system. The topic ID identifier tells the help viewer which topic to display when the user chooses a popup link, jump, or a user interface feature that has been assigned context-sensitive help. You define topic IDs in help topics with the pound sign (#) control code. (See also **Application Context, Context ID, Control Code.)**

Topic Title

A word or phrase that identifies a help topic in four help engine functions: searching the list of keywords on the Index tab of the Help Topics dialog box, conducting a full-text search using the Find tab of the Help Topics dialog box, defining a bookmark using the Bookmark menu; and backtracking through previously viewed topics using the History dialog box. You define topic titles with the dollar sign ($) control code. (See also **Control Code**.)

Trigger

A block of Progress code that executes whenever a specific event occurs (such as when the user chooses a menu item).

Widget

A visual element of a graphical user interface such as a menu, a frame, or a fill-in field.

Window Redirection

The technique used for displaying a help topic in another help viewer window. (See also **Secondary Window**.)

Index

Symbols

:Base command 4–3

:Include command 4–3

:Index command 4–3

:Link command 4–3

:Tab command 4–3

:Title command 4–3

A

Accessing help 1–9

ALIAS section 4–8, 4–9, 4–16

ALinks 3–25, 3–26

AppBuilder 1–5, 1–7, 1–29

applhelp.p 1–22

Application context 1–24, 3–11, 4–13

Audience ix

B

Back button 2–8

BAGGAGE section 4–8, 4–10

BITMAPS section 4–8, 4–10

Bitmaps. *See* Graphics

bmc statement 3–18

bml statement 3–18

BMP files. *See* Graphics

bmr statement 3–18

BMROOT option 4–8, 4–15, B–1

Bold typeface
  as typographical convention xi

Bookmark menu 2–8

Bookmarks 3–6

Browse buttons 2–8, 3–12

Browse sequences 3–7, 3–12
  assigning 3–10
  diagram 3–13
  inserting 3–7
  sequence names 3–14

sequence numbers 3–14
sort order 3–14
syntax 3–14

BUILD option B–1

Build tags 3–6, 3–14
inserting 3–7
multiple 3–15
syntax 3–14

BUILDTAGS section 4–8, 4–10, B–1

C

C language 4–14, 4–15

Calling help 1–9

Case sensitivity 3–6

CD-ROMs 4–10

CHARSET option B–1

CHOOSE event 1–17, 1–18, 1–20, 1–22

CITATION option B–1

CNT option B–2

Coding help calls 1–13
See also Help calling interface

Coding help strings 1–4

Colors 2–15

COMMAND option
SYSTEM-HELP statement 1–12

Comments 4–14, 4–15, 4–16

Compiler messages 4–21

Compiling help source files 4–18

COMPRESS option B–1

Compressing
graphics 3–17
help topic files 4–7, B–1

Conditional compiling. *See* Build tags

CONFIG section 3–28, 4–8, 4–11

Contents file 2–4
commands 4–3
creating with Help Workshop 4–4
elements 4–2
headings 4–4
topics 4–4

CONTENTS option B–2
SYSTEM-HELP statement 1–11

Contents tab 2–3

Contents topic
CONTENTS option B–2

Context IDs 4–13, 4–14, 4–15
header files 4–14
include files 4–14

CONTEXT option
SYSTEM-HELP statement 1–10

CONTEXT-HELP attribute 1–26

CONTEXT-HELP-FILE attribute 1–26

CONTEXT-HELP-ID attribute 1–26

CONTEXT-POPUP option
SYSTEM-HELP statement 1–11

Context-sensitive help 1–9, 1–20, 1–21, 1–22, 1–24, 1–25, 1–29, 4–13, 4–14, 4–15

Control codes
browse sequences 3–12
build tags 3–14
help macros 3–31

help source files 4–10
keywords 3–12
multimedia files 4–10
topic IDs 3–10
topic titles 3–10

Converting manuals to help 2–13

COPYRIGHT option B–2

Creating
graphics 3–17
help topics 3–4
jumps 3–21
popups 3–22

Project files
creating with Help Workshop 4–16

Cross-file links 3–24

D

Data Dictionary 1–3, 1–4

Database fields
help 1–3
help strings 1–2

DBCS option B–2

DEFFONT option B–2

DEFINE BROWSE statement 1–6, 1–27

DEFINE BUTTON statement 1–6, 1–27

DEFINE FORM statement 1–28

DEFINE FRAME statement 1–4, 1–28

DEFINE IMAGE statement 1–6

DEFINE RECTANGLE statement 1–6

#define statement (C) 4–14, 4–15

DEFINE VARIABLE statement 1–28

Destination topic identifiers 3–20

Dialog boxes
help 1–2

E

Edit menu 2–8

ENABLE statement 1–4

Error messages B–2, B–3
displaying descriptions xvii

ERRORLOG option B–2

Example procedures xvi

EXCLUDE section 4–8

Exiting help from application 1–13

F

Field-level widgets 1–2

File locations 3–24, 4–12, 4–19, A–1, B–1, B–3

File menu 2–8

Filename extensions
help input files 4–19

FILES section 4–8, 4–12

Find tab 2–5

Fonts 2–15
changing B–2
changing size B–2
FORCEFONT option B–2
MAPFONTSIZE option B–2
size, changing B–2

FONTS section 4–8

Footnotes
inserting 3–6

FORCE-FILE option
SYSTEM-HELP statement 1–12

FORCEFONT option B–2

FORM statement 1–4

Format phrase 1–4

Frame Phrase 1–28

FTS option B–2

Full-text search 2–6

G

Graphics 2–15
BMROOT option B–1
compressing 3–17
creating 3–17
creating jumps 3–22
creating popups 3–23
creating segmented hypergraphic bitmaps 3–27
designing 3–17
examples 3–18
file locations 3–18
file locations, defining 4–15, B–1, B–3
filename extensions 3–16
files 3–16, 4–19
hypertext links 3–17
placement by reference 3–17
porting 3–17
referencing 3–17
size issues 3–17
tools for creating 3–17
wrapping text around 3–18

H

HCW option B–2

Header files 4–14, 4–15

Help
xvii

Help access methods 1–17, 1–22, 2–13

Help Author's Guide 2–15, 4–16

Help buttons 1–22

Help calling interface 1–9
coding 1–13
Help Topics dialog box 1–17

Help compiler 4–18
comments 4–16
conditional compiling 3–14
errors, resolving 4–9
locating source files 4–15
running from the command line 4–21
sorting sequence numbers 3–14
topic ID errors 4–9, 4–15

HELP events 1–22, 1–23

Help information 2–13

Help keys 1–22

Help macros 4–11
conventions 3–28
declaring 4–11
length restrictions 3–28
multiple, using 3–29
nesting 3–29
syntax 3–28
viewed with Word 3–31

Help menu 2–8

HELP option 1–4
  SYSTEM-HELP statement 1–13

Help project files 4–6, 4–19
  comments 4–14, 4–15, 4–16
  Contents topic B–2
  conventions 4–15
  declaring help macros 4–11
  error messages B–2
  location 4–15
  overview 4–6
  placement of sections 4–16
  sections 4–7
  syntax 4–15

Help projects
  compiling 4–18

Help source files 3–1

Help statements
  bmc 3–18
  bml 3–18
  bmr 3–18

Help strings 1–2
  coding in the AppBuilder 1–5
  database fields 1–3
  dialog boxes 1–2
  field-level widgets 1–4
  in code 1–4

Help systems 1–8
  consistency 2–15
  context-sensitive help 1–9
  creating from printed manuals 2–13
  designing 2–13
  designing structure 2–13
  input files 4–19
  organizing with manual 2–13
  planning 2–13
  planning access methods 2–13
  styles 2–14
  testing 4–22

Help topic files 4–19
  compressing 4–7, B–1
  managing 3–4
  pictured in Word 3–3

Help topics 1–8
  aliases 4–9
  browse order 3–14
  browse sequences 3–12
  build tags 3–14
  colors 2–15
  consistency 2–15
  context IDs 4–13
  creating 3–4
  designing structure 2–13
  fonts 2–15, B–2
  graphics. *See* Graphics
  hypertext links. *See* Hypertext links
  inserting control codes 3–6
  keywords 3–12
  links. *See* Hypertext links
  non-scrolling regions 3–15
  segmented hypergraphic bitmaps 3–27
  style guidelines 2–14
  substitution 4–9
  topic IDs 3–10
  topic titles 3–10
  using white space 2–15

Help Topics button 2–8, 2–9

Help Topics dialog box 2–2
  calling from the application 1–17, 1–18

Help windows 2–6
  Back button 2–9
  Version dialog box B–2

Help Workshop 4–4, 4–16

Help-on-Help file. *See* How to Use

Hexadecimals, in MAP section
4–14, 4–15

History dialog box 3–6

HLP option B–2

Hotspot Editor 3–27

Hotspots 3–27

How to Use Help file
calling from the application 1–20
SYSTEM-HELP statement 1–13, 1–20

Hypertext links 2–10
cross-file links 3–24
destination topic identifier 3–20
link text 3–20
segmented hypergraphic bitmaps 3–27
viewing with Word 3–20

I

Icons. *See* Graphics

Images. *See* Graphics

Include files 4–14, 4–15

INCLUDE section 4–8

#include statement (C) 4–14

Index tab 1–13, 2–4, 3–6

INDEX-SEPARATORS option B–2

Input focus 1–22

Italic typeface
as typographical convention xi

J

Jumps 2–11, 3–21
appearance 2–11
coding example 3–22
creating 3–21
cross-file links 3–24
graphics 3–16
hotspots 3–27
jumping to another window 3–24
pictured 2–10

K

KEY option
SYSTEM-HELP statement 1–11

Key-phrase tables B–3

Keystrokes xii

Keywords 3–6, 3–12
alternate keyword tables B–3
help topics 3–10
inserting 3–7
MULTIKEY option B–3
multiple keywords 3–12
syntax 3–12

KLinks 3–25, 3–26

L

LCID option B–2

Link text 3–20

M

Macros 3–28
CONFIG section 4–8, 4–11

MACROS section 3–28, 4–8

Main window 2–7
  Bookmark menu 2–8
  buttons 2–8
  Edit menu 2–8
  File menu 2–8
  Help menu 2–8
  menus 2–8
  navigation 2–8

Manual
  file locations A–1
  organization of x
  syntax notation xii

MAP section 4–8, 4–12
  comments 4–14, 4–15
  context-sensitive help 4–13
  examples 4–14
  hexadecimals 4–14, 4–15
  mapping methods 4–14
  placement 4–16

MAPFONTSIZE option B–2

Menus 1–17

Messages
  displaying descriptions xvii

Monospaced typeface
  as typographical convention xii

MRB files. *See* Graphics

MULTIKEY option B–3

Multimedia files 4–10

MULTIPLE-KEY option
  SYSTEM-HELP statement 1–12

Multiple-resolution bitmaps 3–17
  See also Graphics

N

Navigating help topics
  jumps 2–11
  popups 2–12

Non-scrolling regions 3–15, 3–16

NOTES option B–3

O

OLDKEYPHRASE option B–3

OPTIONS section 4–9, 4–12, B–1
  BMROOT option 4–8, 4–15, B–1
  BUILD option B–1
  COMPRESS option B–1
  CONTENTS option B–2
  COPYRIGHT option B–2
  ERRORLOG option B–2
  FORCEFONT option B–2
  MAPFONTSIZE option B–2
  OLDKEYPHRASE option B–3
  REPORT option B–3
  ROOT option 4–10, 4–15
  TITLE option B–3

P

PARTIAL-KEY option
  SYSTEM-HELP statement 1–12

Placement by reference 3–17

Popups 2–12
  appearance 2–12, 3–20
  coding example 3–23
  creating 3–22
  cross-file links 3–24
  graphics 3–16

hotspots 3–27
jumping to another window 3–24
pictured 2–12
SYSTEM-HELP statement 1–11
windows 1–11, 3–16

Porting help systems 3–14, 3–17, 3–23, B–1, B–3

POSITION MAXIMIZE option
SYSTEM-HELP statement 1–12

POSITION X Y WIDTH HEIGHT option
SYSTEM-HELP statement 1–12

Procedures
examples of xvi

Project file sections 4–7
ALIAS 4–9
BAGGAGE 4–10
BITMAPS 4–10
BUILDTAGS 4–10
CONFIG 4–11
FILES 4–12
MAP 4–12
OPTIONS 4–12, B–1
WINDOWS 4–13

Project files. *See* Help project files

PROMPT-FOR statement 1–4

Q

QUIT option
SYSTEM-HELP statement 1–13

Quitting help from application 1–13

R

Referencing graphics 3–17

REPLACE option B–3

REPORT option B–3

ROOT option 4–10, 4–15, B–3

S

Schema definition 1–3

Secondary windows 2–8
buttons 2–9
window redirection 3–24
WINDOWS section 4–13

Sections. *See* Project file sections

Segmented hypergraphic bitmaps. *See* Graphics

Semicolons 3–6

Sequence names 3–7, 3–14

Sequence numbers 3–14

SET statement 1–4

SET-CONTENTS option
SYSTEM-HELP statement 1–11

SHED 3–27

SHG files,. *See* Graphics

Sort order
browse sequences 3–14

Status area 1–2

STATUS-AREA attribute 1–2

Syntax notation xii

System Help Editor. *See* SHED

SYSTEM-HELP statement 1–8
FINDER option 1–17
HELP parameter 1–20
popups 1–11
QUIT parameter 1–13

T

Testing help systems 4–22

TEXT option
  SYSTEM-HELP statement 1–12

TITLE option B–3

TMPDIR option B–3

ToolTips 1–5
  defining in the AppBuilder 1–7

Topic IDs 1–8, 3–6, 3–8, 3–10, 4–13
  aliases 4–9
  allowed characters 3–10
  assigning 3–10
  errors 4–9
  examples in MAP section 4–14
  header files 4–14, 4–15
  include files 4–14, 4–15
  inserting 3–7
  MAP section 4–13, 4–15
  substitution 4–9
  syntax 3–10

Topic titles 3–6, 3–10
  inserting 3–7

Triggers 1–17, 1–22

Typographical conventions xi

U

underlined text
  dotted 2–12
  solid 2–11

UPDATE statement 1–4

V

Version dialog box B–2

VIEW-AS phrase 1–6

W

Widget hierarchy 1–22

Widgets 1–2

Window redirection 3–24

Window status area 1–2

Window types 2–6

WINDOW-NAME option
  SYSTEM-HELP statement 1–10

Windows bitmaps. *See* Graphics

Windows metafiles. *See* Graphics

WINDOWS section 4–9, 4–13

WMF files. *See* Graphics