ABSTRACT

This paper describes how to create help files for the native help system on your operating system (Windows, OS/2®, or UNIX), and integrate those help files with SAS/AF® or SAS/EIS® applications that you design. This paper illustrates the process of planning a help system, using tools from other vendors to write the topics and compile the help files, and using SAS screen control language (SCL) functions to invoke the help system from within your SAS/AF or SAS/EIS application. This paper is not a complete reference for building native help systems; that task requires more in-depth knowledge than can be addressed here. However, the "References" section at the end of the paper provides a list of sources for more information.

INTRODUCTION

Beginning with Release 6.10, the SAS System under Windows and OS/2 provides the capability to integrate native help with SAS/AF and SAS/EIS applications. In Release 6.11, the help system supplied with the SAS System (traditionally based on SAS CBT entries) was replaced with help files in the native format: WinHelp for Windows, OS/2 help for OS/2, and Helpius (a proprietary help viewer developed by SAS Institute) for the UNIX systems.

With the SAS System now using native help, it is to your advantage to supply native help for any applications you develop using SAS. Not only does native help provide better access to the information you provide (with facilities for searching, annotating, and easy navigation), but it also helps ensure consistency and seamlessness as the SAS/AF and SAS/EIS applications you write interact with the SAS System.

PLANNING YOUR HELP SYSTEM

You can create online help to be accessed at three levels:

Application level

This type of help is typically a user's guide or reference for the entire application. You can provide access to this help using a persistent element of the user interface (such as a menu item). The same help topic appears regardless of where in the application the user invokes the help.

FRAME (dialog or panel) level

This is help for a particular task or FRAME entry, usually invoked using a Help button. This help is context-sensitive; that is, the topic you see depends on the current FRAME entry (and thus, the current task).

Object level

This is help for a particular element on the FRAME entry and is a more granular approach to context-sensitive help. This help can be triggered by a predefined event in the FRAME, such as pressing a certain key combination, or by taking advantage of the HELPMODE within SAS/AF software.

The level of help that you decide to implement depends on the complexity of the application and the sophistication of its intended users. Ideally, you should design your application's interface to be as intuitive as possible to use, minimizing the need for extensive help. Beyond that, you should anticipate when your users will need help, with what tasks they will need help, and how much they will need to know.
BUILDING YOUR HELP FILES

The processes for building WinHelp and Helpplus files are similar. Helpplus is the help viewer provided with the SAS System on UNIX platforms; its design is based on Windows Help. Building OS/2 help uses a different process, though it has some similarities to building WinHelp and Helpplus files.

Building Windows and Helpplus Help Files

Before you can create Windows or Helpplus help files, you need the following tools:

- **The Microsoft help compiler** (Windows only). For WinHelp 3.1 (the standard viewer under Windows 3.1), this file is HCP.EXE. You can find the help compiler, along with some unsupported help tools, on the Microsoft Developer Network (MSDN) CD and Microsoft's FTP site. (See "References" at the end of this paper.)

For WinHelp 4.0 (the standard viewer under Windows 95, Windows NT 3.51, and Windows 3.1 with the Win32s extensions), you need the Microsoft Help Workshop. (The Microsoft Help Workshop consists of HOW.EXE, HCRTF.EXE, HCW.HLP, and HCW.CNT.) You can find the Help Workshop in the Microsoft Win32 SDK. Note that WinHelp 3.1 files are compatible with the WinHelp 4.0 viewer, but WinHelp 4.0 viewers are not backward-compatible to WinHelp 3.1.

Beginning with Release 6.11, the SAS System supports WinHelp 4.0 under all of the Windows environments.

- **The Helpplus help compiler/viewer** (UNIX systems only). This tool is provided with the SAS System. You can find it in /sasroot/sasbase/helpplus, where sasroot is the root path for your SAS System installation.

- **The segmented hotspot editor.** You need this tool only if you want to create hotspot graphics (graphics that jump to different topics depending which part of the graphic you click). You can find SHED.EXE (the Windows version) on either the MSDN CD or the Microsoft Windows SDK. The Helpplus version is supplied with the SAS System in /sasroot/sasbase/shed, where sasroot is the root path for your SAS System installation.

  (Optional) A help authoring tool (such as RoboHelp or ForeHelp) to make your task a bit easier. These tools can assist you in creating help files from start to finish. There are a number of commercially available help authoring tools, plus a few shareware programs.

The ingredients for WinHelp and Helpplus files are:

- rich text format (RTF) files, which you can create with an ASCII editor or with an authoring tool that supports RTF (such as Microsoft Word or ForeFront's ForeHelp)
- help project file (HPU), which you can create with an ASCII editor or a help authoring tool
- contents file (CNT), which you can create with an ASCII editor or a help authoring tool. (This is for WinHelp and only if you want to take advantage of the WinHelp 4.0 contents tab.)
- Windows bitmaps or metafiles or X pixmaps (if you want graphics)
- segmented hotspot graphics, which you can create with Microsoft's SHED program or the shed program that comes with Helpplus (if you want hotspot graphics).

In these RTF-based help systems, the basic unit of information is the topic. The topic is defined by certain RTF codes that have special meaning to the help compiler. Here is a summary of these codes:

- **title (RTF code: footnote {})**
  The text that identifies the topic to the reader during keyword searches. Note that this text does not automatically appear in the displayed topic.
- **context ID (RTF code: footnote #)**
  The internal identifier used to connect links to the topic.
- **keywords (RTF code: footnote X)**
  Words or strings of words that the viewer uses during keyword searches (optional)
- **browse sequence (RTF code: footnote +)**
  Part of a sequence of topics that associates the current topic with a previous topic and the next topic (optional)
- **non-scrolling region (RTF code: keep)**
  An area at the beginning of the topic that remains stationary as the user scrolls through the topic.

Here is a simple RTF file containing a typical help topic:

```rtf
\rtf1\ansi\deff12
\backslashatalt\pard\plain\s1\sa40\keep\n\footnote $ Accessing Patient Records$
\footnote # \footnote Opening Patient Records$
\footnote K Accessing Patient Records$
\plain \fs16
\plain \fs28 \b Accessing Patient Records\par \pard \plain \fs18 \b To open a patient's medical record\par \pard \fs18 \e Open the patient's medical record by clicking on the file cabinet image. A list of patient names appears. \par
```

You can see that in addition to the topic footnotes, this code makes use of several other RTF formatting codes. For complete information about the RTF formatting codes and their effect within WinHelp topics, see the references cited at the end of this paper.

The help project file (HPJ file) is the master file for the help project. It specifies which viewer options to set and which RTF files to include. It also defines the numbers that your SAS application needs to use to access a particular help topic. Here is an overview of the main sections of the HPJ file:

- **[ALIAS] Section**
  Lets you assign one or more alias identifiers to a particular help topic. This is useful for standardizing links between help files or assigning placeholders for topics that are not complete or have been removed.

- **[CONFIG] Section**
  Sets up the initial environment of the help viewer by running any startup macros you specify.

- **[FILES] Section**
  Points to the RTF files that make up the content of the help file.

- **[MAP] Section**
  Provides the context numbers needed to map WinHelp API calls (from a software application) to the appropriate topics. When you use the HOSTHELP SAS function (described later) to invoke help, you specify one of these mapped numbers to display a topic.
Applications Development

[OPTIONS] Section
Contains everything from compression options to the help project title to the copyright statement.

[WINDOWS] Section
Defines the different types of window configurations you use in your help project.

Here is an example of a simple HPJ file:

```
[OPTIONS]
ERRORLOG=IMAGEDMO.ERK
COMPRESS=NO
CONTENTS=Contents
TITLE=Help for Medical File Cabinet
COPYRIGHT=Copyright 1996 SAS Institute Inc

[WINDOWS]
main=.,,(192,192,192),0

[FILES]
IMAGEDMO.RTF

[MAP]
#define Opening_Patient_Records 2001
#define open_file_folder 1002
#define file_cabinet 1001
```

If you use an authoring tool (such as ForeHelp) to create your help files, the help project file and RTF files are created automatically. Without an authoring tool, you must use an ASCII editor to enter these HPJ and RTF codes manually.

Using Helpplus

The Helpplus application serves as both the help compiler and help viewer for native help for the SAS System under UNIX platforms.

To use Helpplus to compile a help file, invoke the program with this basic form:

```
helpplus -c -in filename.hpj -o <filename.hlp>
```

The options have these meanings:
- `-c` invoke Helpplus as a compiler
- `-in` specify the input file (HPJ) to compile
- `-o` specify the output file (HLP) to create (optional)

To use Helpplus to view a compiled help file, invoke the program using this basic form:

```
helpplus helpfile
```

where `helpfile.hlp` is the help file that you want to view.

For more information about using Helpplus, view the Helpplus Programmer's Reference, which is shipped with the SAS System as a help file in `sasroot/mathdoc/hipdoc.hlp`, where `sasroot` is the root path for your SAS System installation.

Differences Between WinHelp and Helpplus Source Files

WinHelp and Helpplus recognize the same basic set of RTF codes to build help topics, but there are some differences that you should keep in mind if you intend to create one set of RTF files to use on both Windows and UNIX.

help project file (HPJ) format
Helpplus recognizes all of the options and syntax available in a WinHelp 3.1 HPJ file, but not all those in a WinHelp 4.0 HPJ file. If you create your help file for WinHelp 4.0 using Microsoft Help Workshop, you will need to construct a different HPJ file for use with the Helpplus-based help.

filename format
Because the UNIX and Windows file systems are different (Windows uses the DOS convention of 8-dot-3 filenames and is case insensitive; UNIX supports longer filenames and is case sensitive), most likely you will have to maintain separate HPJ files anyway.

graphics format
Windows Help supports the use of Windows bitmaps (BMP files) and Windows metafiles (WMF files), and Helpplus supports the UNIX-based X pixmaps (XPM files). Not only must you create a set of graphics for each system, you must also make sure that the RTF codes that refer to the graphics use the proper filenames for the system you are on. Also, any segmented hostpost graphics you use must have been created using the ShED program for the appropriate system.

font size
Typical Windows help systems use an 8- to 10-point font size for the body text. But on a typical X-Windows system with a large, high-resolution display, this font size will be much too small. You can use the MAPFONTSIZE option in the [OPTIONS] section of the Helpplus HPJ file to resolve this difference without having to change all references to the font size within your RTF files. For example, the options

```
MAPFONTSIZE 10:14
MAPFONTSIZE 14:18
```

will increase all instances of 10-point and 14-point text to 14-point and 18-point, respectively.

Note: The MAPFONTSIZE option is a WinHelp 4.0 feature of the HPJ file that Helpplus supports.

complex RTF coding
Even though the Helpplus compiler supports the same basic RTF codes that WinHelp supports, you may find that Helpplus is less tolerant when it comes to deciphering complicated RTF files. (For an example of a ridiculously complicated RTF file, use Microsoft Word to save a document as RTF and then view the resulting file in an ASCII editor.) Helpplus reliably supports any valid RTF file that you might create with an ASCII editor (manually editing the codes yourself) and has also been shown to support most files generated by ForeFront's ForeHelp.

Building OS/2 Help Files

Before you can build an OS/2 help file, you need the OS/2 help compiler (IPFC.EXE) and its associated files. The OS/2 help compiler is provided with the OS/2 2.1 Software Development Kit.

OS/2 help files come in two flavors: INF files (for online documentation) and HLP files (for context-sensitive help). The main difference between the file types is that the INF files can be viewed independently of an application by using the OS/2 VIEW utility, but they cannot be invoked as context-sensitive help from within an application. HLP files cannot be viewed other than by invoking them programmatically from within an application. If you are creating context sensitive help for use with SAS/AF applications, use the HLP format.

You can use the same source files to create either type of help file, and the OS/2 help compiler creates HLP files by default. To create an INF file instead, you specify a /INF switch on the IPFC command line. Even if your final help file will be in HLP format, you might want to format the file as INF during development to make it easier to view the file independent of your SAS/AF application.
The ingredients for OS/2 help files are:

- Information Presentation Facility (IPF) files, which you can create with an ASCII editor
- OS/2 bitmap files (if you want graphics)

IPF is a tag-based markup language. The main components of an IPF help topic are:

- **heading tag (h1 through h6)**
  - Indicates the title of the topic, the level of the topic in the help file hierarchy, the reference identifier (coded with the id= attribute), and the resource identifier (coded with the res= attribute). The reference identifier (a unique alphanumeric identifier) is used to create links within the OS/2 help file to other help topics. The resource identifier (a unique numeric identifier) is used to invoke the topic as context sensitive help from an application. The resource identifier is the number you will need to use in your SCL code when invoking the help topic from your SAS/AF application.

- **topic text with formatting tags**
  - Contains the content of the topic, plus various formatting tags that control how the topic appears when displayed.

In general, IPF is much more readable than RTF. Here is a sample IPF file:

```plaintext
:uservdoc.
:docprof toc=1234.
:title Help for Medical File Cabinet
:h1 Accomplishing Common Tasks

:h2 id=open res=2001 Accessing Patients
:p.
To open a patient's medical record
:ul.
:li Open the patient file cabinet
by clicking on the file cabinet image.
A list of patient names appears.
:li In the :hp2 Select a Patient:ehp2.
:hp2 list, scroll through the list of patient names and click on the one whose file you want
:access.
:li Click on the file folder image.
The patient's file opens.
:eul.

:uservdoc.
```

Complete usage and reference information about IPF is provided in the IPFC2O.INF file (which you can open using the OS/2 VIEW program) included in the OS/2 Software Development Kit with the IPF compiler.

**FORMATTING YOUR HELP TOPICS**

Depending on the level of help you decide to provide, your help topics may vary in length from one sentence to several pages. Object-level help topics are typically very small, while application-level help topics, which might be organized into tasks, are larger.

As you write the content of your help file, keep the following style issues in mind:

- **Topic length**
  - A help topic should be one or (at most) two screens long, where a screen is 15-20 lines. Minimize topic length by breaking information into smaller categories, using multicolored lists, and funneling subordinate information to pop-up (definition) and secondary windows.

- **Pop-up hot spots**
  - Avoid "green-screen" syndrome. Don't place too many pop-up links (which are indicated with green text and underlines) on a single screen. For example, link terms to their definitions only once during the topic.

- **Link placement**
  - Avoid using links in running text that replace the contents of the current window. Instead, include a See Also section with the relevant links at the end of the topic.

- **Font selection**
  - Use a font that the user will have installed. For Windows help, Microsoft-supplied fonts are a good bet.

- **Font style**
  - Avoid using italics (they are hard to read) and use bold sparingly. Don't mix font sizes in running text.

- **Color**
  - Avoid using lots of colors to highlight text. It's distracting (and misleading). Also, some users are not able to distinguish colors very well; do not rely on color to convey meaning.

- **Headings**
  - Use headings that convey as much information as possible about the topic's contents. Or, be sure to associate meaningful keywords with the topic. Be consistent in heading size and formatting. Use non-scrolling regions to set off your heading.

- **Contents Topic**
  - When working on a help file that will ship with other files, be sure to assign distinctive titles to the initial (contents) topics so users don't get lost in hyperspace.

- **Artwork**
  - Include artwork when necessary. Consider the following:
    - If your software product is running as the user accesses help, a picture of the product: screen might be redundant.
    - Users tend to want to click on graphics. Anticipate the reactions of your users.
    - The more graphics, the larger the resulting help file.

- **Examples**
  - When including large examples (especially coding examples), place them in a jumped link rather than a pop-up. Users might like to copy the examples to the clipboard or print them. Remember that the SAS System lets the user submit code directly from the clipboard.

**HOOKING UP YOUR HELP SYSTEM**

After you create the help file, you need to configure your SAS/AF or SAS/EIS application to invoke the help topics. The SAS System provides a few mechanisms to do this; the method you choose depends on the type of help file you are providing.

**Using the SAS Help Command**

You can invoke a help topic that you created by using a portable SAS help command that resembles a CBT invocation:

```
HELP C=SASHELP.helpfile.:bname.CBT FRAME=framename
```

The file (named helpfile.hlp) must reside in the location pointed to by the SAS HELPLOC system option. Also, you must have another text file in the same location with the name helpfile.hdx, which contains entries for each topic you want to access.
The entries have the following format:

cbname framename context_number

where:

cbname is the name of the CBT entry that you specify in the HELP command. Note that you actually don’t need to create a CBT entry; you only need to make sure that this name matches the cbname in the command.

framename is the name of the CBT frame that you specify in the HELP command. Again, you actually don’t need to create a frame; you only need to make sure that this name matches the framename in the command.

context_number identifies the topic to invoke (as defined by the [MAP] section in the help project file for WinHelp or Helpplus, or the res= attribute in OS/2).

The advantage of using this method is that the HELP command is completely portable, even to those platforms that do not support native help. You can provide SAS-based CBT entries as an alternative on those platforms, such as MVS. This method might also be a good choice when you want to replace a CBT-based help system with a native help file, but don’t want to change the SAS code that invokes the help.

This scheme has two restrictions:

- You must have a SAS catalog named helpfile in the SASHELP library, even if the catalog contains no entries. The SAS System first checks for the existence of the catalog before it invokes native help; if the catalog does not exist, SAS writes an error message to the log.
- You must end your SAS session and restart before the entries in the helpfile.hdx take effect. SAS loads the contents of these files upon invocation.

For example, suppose you have help topics in a help file named SASAPP.HLP, and you want to invoke the topic mapped to the context number 1001 within the help file. You would need to:

1. Create an empty SAS catalog within the SASHELP library named SASAPP.
2. Copy the SASAPP.HLP help file to the path pointed to by the HELPLOC system option.
3. Create a file named SASAPP.HDX (in the path pointed to by the HELPLOC system option) that contains an entry such as:

   myhelp mytopic 1001

4. Restart your SAS session, and then invoke help with this command:

   help c=sashelp.sasapp.myhelp.cbt frame=mytopic

Using the HELPREGENERATE System Option

The HELPREGENERATE system option (available only on Windows and OS/2) is undocumented. Therefore, you should note that support for the option is subject to change without notice. However, you can use this command to provide easy access to your custom help file from the Help menu within SAS.

The syntax of the option is:

-HELPREGENERATE "menu text" "helpfile.hlp"
  
For example, to add the SAS online documentation help file to your Help menu (in the SAS System under Windows), specify this option when you start SAS:

-HELPREGENERATE "SAS Online Doc"
  "core\winhelp\sasdoc.nlm"
  "Open SAS online documentation"

The menu item appears in the Help menu after the Sample Programs item (before the item separator above Utility Application).

The help filename should be specified as a path relative to $sasroot (for example, core\winhelp\helpfile.hlp). You can also specify absolute paths.

Any help text you optionally specify will appear in the message area in the status bar at the bottom of the SAS window when you select the menu item.

Using the HOSTHELP Function

The HOSTHELP function, introduced in Release 6.11 of the SAS System, invokes the native help system to display the topic you specify. When you use the HOSTHELP function, you specify a help method (or mode), the name of a help file, and additional information, such as a topic identifier, depending on the method you specified.

The HOSTHELP function is a portable function; that is, it exists on all supported Release 6.11 platforms. However, the methods that the function supports vary depending on the operating system. For more information about the HOSTHELP function, see SAS Software: Changes and Enhancements, Release 6.11. The following methods are supported on Windows, UNIX, and OS/2:

HELP_CONTENTS
displays the Contents topic of the help file, as shown in Display 1. The Contents topic is the main topic or table of contents for the help file.

HELP_CONTEXT
displays a specific topic that you identify, as shown in Display 2.

In addition, Windows and UNIX also support these methods:

HELP_CONTEXTPOPUP
displays, in a popup window, a specific topic that you identify, as shown in Display 3.

HELP_KEY and HELP_PARTIALKEY
displays the topic or list of topics associated with a specified keyword.

The Contents topic, topic identifiers, and keywords are elements of a help file that you establish when you create the file.

You can override the _HELP_ method in your SAS/AF applications to call the HOSTHELP function rather than the CBT help. When doing this, the tricky part is invoking the HOSTHELP function at the right point to provide the type of help your users need, when they need it.

The following example is one scenario that works, but you might find other procedures that fit your needs better.

1. In each object in the FRAME for which you want to provide help, add an instance variable (or local variable) named helpid. Assign a unique integer value to this variable in each widget. This integer value should be the same as the context number you have mapped to the object’s help topic (either in the [MAP] section of the HPJ file in WinHelp or HelpPlus or using the res= attribute of the topic heading in OS/2).
2. Create the SCL code to replace the _HELP_ method. It should look similar to the following:

```sas
length _self_ 8;
/* Create list for widget properties */
prop=makealist();
/* Create list for _LOCALS_ list */
locals=makealist();
/* Get properties */
call send(_self_, '_GET_PROPERTIES_', prop);
/* Get _LOCALS_ */
locals=getitem(prop, '_LOCALS_');
/* Get helpid */
helpid=getitem(locals, 'helpid');
/* build help filename */
/* We are assuming the help file resides */
/* in the location pointed to by HELPLOC */
/* system option */
/* Note that HELPLOC can contain multiple */
/* paths on UNIX systems, so you might need */
/* to hard code the help file path instead */
helploc=optget('HELPOBJ');
helpfile=helploc || 'imagedmo.hlp';
/* Invoke HOSTHELP using the helpid as */
/* an argument */
rc=hosthelp('HELP_CONTEXTPOPUP', helpfile, helpid);
return;
```

Note: The HELP_CONTEXTPOPUP method for the HOSTHELP function is supported under Windows and Helpplus, but not OS/2. You would need to use the HELP_CONTEXT method for OS/2.

Compile this code and save it in its own SCL entry named HOSTHELP.SCL.

3. In the init: label for each frame, enter the following statement for each widget for which you have help:

```sas
call notify('object-name',
    '_set_instance_method',
    '_HELP', 'libref.catname.hosthelp.scl');
```

If you want to override the _HELP_ method for all widgets within a frame, include the following code within the init: label:

```sas
/* load an instance of the widget class */
wClass=loadclass('sashelp.fsp.widget');
/* get an SCL list with the IDs of all */
/* the widget instances */
call send(wClass, '_get_instances_',
    widgets, 'yes');
/* override the _HELP_ method for each widget */
do w=1 to listlen(widgets);
    call sendgetitem(widgets, w,
        '_set_instance_method',
        '_HELP',
        'libref.catname.hosthelp.scl');
end;
/* delete the list of widget IDs */
rc=dellist(widgets);
```

4. Include a widget (a pushbutton, for example) on the frame that, when activated, will toggle HELPMODE for the frame. The code for the widget might look similar to the following:

```sas
helpmode:
    call notify('(', '_IS_HELP_MODE_', enabled);
    if enabled then
        call notify('(', '_CLEAR_HELP_MODE_');
    else
        call notify('(', '_SET_HELP_MODE_');
    return;
```

If your application uses a tool bar object, you can add a tool bar button that invokes HELPMODE.

When the user toggles HELPMODE, the pointer changes to an arrow with a question mark. When the user clicks on an object in the FRAME, the object's associated help topic is displayed.

CONCLUSION

Native help, especially for Windows applications, is a growing industry. Users of Windows programs expect comprehensive and well-designed help. Even Microsoft, which in the past provided only unsupported tools and documentation about developing native help, has turned around and developed new tools and created significant support for application help developers.

Creating native help files is not an easy task to learn, but as the SAS System continues to provide native help for all of its components, SAS users will expect native help with their SAS/AF and SAS/EIS applications.

REFERENCES

Vendor-Supplied Information and Tools

Microsoft Development Network (MSDN) CD
Get the Windows Help Authoring Tools (WHAT) from the “Unsupported Tools” section of the CD. These tools are also available on Microsoft's FTP server, ftp.microsoft.com.

Microsoft Win32 Software Development Kit
Get the HCW.EXE, HCRTF.EXE, HCW.HLP, and HCW.CNT files from this SDK. The HCW.HLP help file (Help Author's Guide) contains complete reference information for WinHelp development.

IBM® OS/2 2.1 Software Development Kit
Get the tools provided in the IPFC folder and the IPFC20.INF file from the BOOK folder. VIEW the IPFC20.INF file for complete information about the OS/2 Information Presentation Facility.

"Helpplus Programming Reference"
This help file is shipped with the SAS System under the UNIX environments. To view the file, run:
```
sasroot/sasexe/helpplus sasroot/nathelp/hprdoc
```
where sasroot is the root path for your SAS System installation.
Applications Development

Books
Microsoft Press (1995), Microsoft Windows 95 Help Authoring Kit:
Microsoft Press, ISBN 1-55615-892-0
This book comes with a CD that contains all of the Microsoft help compiler tools provided with the Microsoft Win32 SDK.
A popular book on Windows help development, from start to finish.

Internet Resources
WINHLPL-Listserv
An internet mailing list dedicated to Windows help issues.
Participants range from novice help developers to advanced programmers. No question is too basic—all seem to get good responses. To subscribe, send e-mail to
listserv@Admin.HumberC.on.ca.ca with this message:
sub winhlpl Your Name
comp.os.ms-windows.programmer.winhelp USENET newsgroup
Basically the same charter as the WINHLPL-Listserv with many of the same participants.

World Wide Web Pages
Several sites on the WWW devoted to help development issues. Here are a few:
• The WinHelp Universe:
  http://www.primenet.com/-wai/universe.html
• WinHelp-L Web Page:
  http://www.humberc.on.ca/-byrnes/winhelp.html
• Windows Help Authoring Resource Page:
  http://www.metainfo.com/knowware/helpme.html

ACKNOWLEDGMENTS
Phil Herold, John Kruth, and Wanda Verneaut of SAS Institute reviewed this paper. Linda Fineman of SAS Institute provided a sample SAS/AF application to test and illustrate the processes described in this paper.
SAS, SAS/AF, and SAS/EIS are registered trademarks of SAS Institute Inc. in the USA and other countries. IBM and OS/2 are registered trademarks or trademarks of International Business Machines Corporation. © indicates USA registration.
Other brand and product names are registered trademarks or trademarks of their respective companies.