Paper 19

Adding Help and Other User Assistance to SAS/AF® Applications

Marty Tomasi, SAS Institute Inc., Cary, NC

ABSTRACT

With the Nashville Project releases, SAS/AF® software provides several new help-specific features that developers can use to add user assistance to their applications. This paper describes the use of the _help method in the Object class, the help attributes for objects and frames, the toolTipText attribute, and other attributes that are used for displaying different types of help. In addition, this paper looks at the different ways a developer can attach HTML-based help to their application, including a Web-based approach that allows for remote support of applications. Finally, this paper examines ways a developer can extend user assistance beyond simple help systems.

INTRODUCTION

As application usability becomes an increasingly important part of applications development, developers must look for ways to add user assistance features to their applications. User assistance can exist as a typical help system, or it can be a complex performance support tool that includes interaction with the application. New features in SAS/AF software make it easier for developers to add user assistance to their applications.

This paper examines some of the user assistance that you can employ with SAS/AF. It presents

- guidelines for designing an HTML-based help system
- a description of the "help" attributes, the _help method, SAS help commands, and other help considerations specific to SAS such as the expanded HELPLOC option and URL Index (udx) files
- a how-to guide for adding context-sensitive help and tooltips to frames and frame objects
- information on generating documentation for SAS/AF classes and making that documentation available for other developers
- suggestions for adding other types of user assistance
- tips for working with HTML Help files in the Windows environment.

DESIGNING HELP FOR SAS/AF APPLICATIONS

The most common way of adding user assistance to an application is to include an online help system. SAS/AF software supports help in several formats, including HTML-based Web pages and SAS/AF CBT catalog entries. Your applications can also take advantage of context-sensitive help and tooltips.

When you implement a help system for your application, you should consider

- whether or not context-sensitive help is required (so that you can make the necessary changes to frame and component properties)
- what functionality the various help commands provide and how you can use them appropriately
- how your help files are structured and stored
- whether you will distribute your help system with the application or make it available on a Web site
- who your audience is and how they will likely use the application.

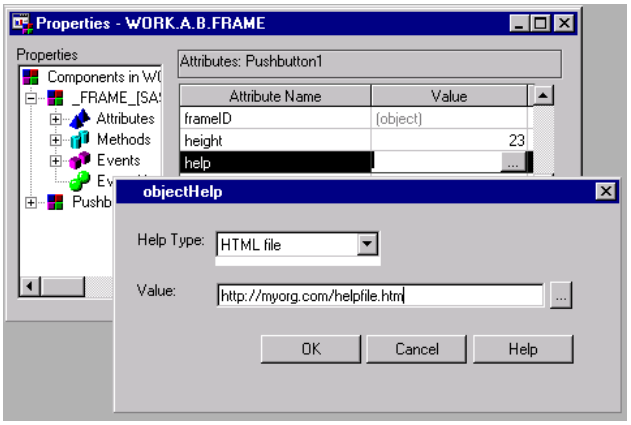
To create a help system for SAS/AF, you generally follow these steps:

1. Design a plan for your topics, create HTML template files, and write the topics.
2. Add the links from the application to help topics by specifying help attributes in SAS/AF for frames and objects on frames.
3. Design and implement your table of contents and index, if needed.
4. Collect all components of your help system (HTML files, graphics, and so forth).
5. Deploy the help by copying the files to the appropriate directory or Web server.
6. Test the help calls from the application.

After you create a help system or help Web site, an important task is updating and managing the information. The same things that annoy you about Web pages — outdated content and broken links — will annoy users of your help. Make sure you schedule time to make changes to your pages and to validate all content and links.

THE HELP ATTRIBUTES

SAS/AF software now includes several help attributes. These attributes can be quickly located by selecting the Help category under the Attributes node of either the Class Editor or the Properties window.



Display 1. *SAS/AF Properties window with Object Help editor*

Help attributes on the Frame class include:

CBTFrameName
: the CBT frame that is displayed when a CBT entry is assigned as the help for the object

help
: the help topic that appears when the user selects help for the window or dialog box

showContextHelp
: a flag indicating whether context-sensitive help is supported by the frame.

Help attributes on the Widget class that are also inherited by the new visual controls (such as the radio box control) include:

CBTFrameName
: the CBT frame that is displayed when a CBT entry is assigned as the help for the object

help
: the help topic that appears when the user selects help for the window or dialog box

helpText
: the help text that appears on the status line when a user selects and holds a control; note that while this attribute can

be set, the functionality it represents is not available on all platforms.

toolTipText
: the text that appears as a tooltip when the cursor is positioned over the control

See “Adding Help to a Frame” and “Adding Context-Sensitive Help” below for more information on working with these attributes.

THE _HELP METHOD

The _help method that is defined in the Object class enables a SAS/AF application to invoke a help topic. The _help method also can be used to invoke help on topics that do not have help attributes assigned, which was a feature that aided development of help files in Release 6.12.

HOW HELP CALLS WORK

When the _help method is invoked on a frame, the following tasks are performed:

- The method checks the frame’s help attribute and opens help in the designated format. (See “Adding Help to a Frame” below for details on setting this attribute.)
- If no value is specified for the help attribute, the _help method creates an ID for the frame using the format *framename_1*. The system then checks to see whether a udx file exists in the HELPLOC path (see below) and whether that udx file contains the *framename_1* mapping. If both conditions are true, then help is displayed by opening the HTML file named framename_1.htm from the directory that has the same name as the frame’s catalog.

When the _help method is invoked for context-sensitive help for an object on a frame, the following tasks are performed:

- The method checks the selected object’s help attribute and opens help in the designated format. (See “Adding Context-Sensitive Help” below for details on setting this attribute.)
- If no value is specified for the help attribute, the _help method searches the udx file for help topics using the following search order:
  1. The _help method searches for instance-specific help on the object. Given a list box named *listbox3* on a frame named *myframe*, the method tries to find help for *myframe_listbox3*.
  2. The _help method searches for default class help for the object’s class. Given a list box that is an instance of the myListBox class, the method tries to find help for *myListBox_1*.
  3. The _help method then searches for help on the “container” that hosts the object, which can be the frame or composite object. For a push button named *myPushButton* that is part of the *compObj* composite, the method tries to find help for *compObj_myPushButton*.

Note that the _help method searches the supplied udx file only if the help attribute is not set. In Release 6.12, this feature enabled a developer to designate a single help call for all instances of a particular class. For example, if you created a subclass of the Command Push Button class called *OKBUTTON*, you could get help for all instances of that widget by providing a map in the index file (that is, the old “.hdx” file) to *okbutton_1*. In Version 7 and Version 8, however, you can add the same functionality by subclassing the Push Button Control, adding a label attribute value of “OK,” and setting the help attribute to a specific URL. Every instance of this subclass uses the same URL for its help.

Simply put, it is recommended that you set the help attribute for frames and objects (other than composites) if you want to provide help.

HELP COMMANDS

SAS/AF software provides several commands that you can use to display help for your application. You can add these help commands to a menu item, or you can assign one of them as the command that is executed by a control such as a push button. You can also attach the commands to icon buttons on a toolbar.

The commands include

HELP
: The HELP command displays help for the active frame or window based on the values of the frame's Help attributes. The help topic or topics are displayed in the designated help browser. On most systems, you can also issue the help command by pressing F1.

  The WINDOWHELP command performs the same function as HELP.

HELPMODE
: When the HELPMODE command is issued, SAS enters object help detection mode (helpmode). In helpmode, the cursor changes to a question mark (?) and SAS waits for the user to select an object on the active frame. When a user selects an object while in helpmode, SAS displays the help that is defined in that object's help attribute. On some systems, you can initiate the helpmode by pressing SHIFT+F1.

WBROWSE "*valid-URL* "
: The WBROWSE command simply displays a valid URL, which can be any page that can be displayed by the associated Web browser.

For example, you could define a PMENU entry and assign it to a frame. You could add a Help menu to the menu bar and assign one of the help commands to provide helpful tips for users. Consider this partial PMENU definition:

```
proc pmenu catalog=company.apps;
  menu main;
  item 'File' menu=filemenu;
  item 'Help' menu=helpmenu;

  Additional code here
  menu helpmenu;
    item 'Contents' selection=a;
    item 'Company Home' selection=b;
    selection a 'help helploc://apps/qt.htm';
    selection b 'wbrowse "http://myorg.com"';
```

A user of the frame that contains this PMENU entry could select **Help>Contents** to view help for the application, or the user could select **Help>Company Home** to view the organization’s intranet.

OTHER HELP ADMINISTRATION PIECES

SAS/AF software relies on two other items to display appropriate help topics: the HELPLOC system option and individual URL index (udx) files.

THE HELPLOC SYSTEM OPTION

SAS/AF software uses the HELPLOC option to identify the path it searches to locate online help files. The default path for SAS online help is a format such as

```
!SASROOT/X11/native_help
```

or

```
!sasroot\core\help
```

With Version 8, you can also add multiple search paths to HELPLOC. To specify the location of the help files for your application, you must add the appropriate path in the HELPLOC system option in the configuration file that is used to start your application.

> **Caution:** Do not remove or modify the default path from the HELPLOC option in your SAS configuration file. Such action prevents SAS software from locating its associated online help.

For complete information on modifying your application’s configuration file to set the HELPLOC option, refer to the *SAS Guide to Applications Development*.

The HELPLOC:// protocol is a mechanism defined by SAS for displaying help files from within your application. The SAS help facility replaces the HELPLOC:// protocol with a path listed in the HELPLOC option. The paths are searched in the order in which they are listed until a valid help topic is found or until there are no further paths to search. For example, if your HELPLOC option contains

```
('!sasroot\core\help'  '!sasuser\classdoc')
```

you could add a path to the directory in which you stored the help for your application:

```
('f:\apps\myapp\help' '!sasroot\core\help'
'!sasuser\classdoc')
```

The SAS help facility will then search through f:\apps\myapp\help to locate a help topic before it searches the SAS online help path (!sasroot\core\help).

You can also specify a path to a valid Web server or a path to a network directory in the HELPLOC option. For example, if your application's configuration file defined HELPLOC as

```
('!sasroot\core\help' 'http://myorg.com/hlp')
```

a help call from a SAS/AF frame that uses the form

```
help helploc://myapp/intro.htm
```

will first search for the requested help topic in the SAS online help path before sending a fully qualified URL to the browser to locate the help topic at http://myorg.com/help/myapp/intro.htm.

URL INDEX (UDX) FILES

URL index files are used by SAS software to map help IDs to HTML file names (or HTML topics within a compiled HTMLHelp file). Every application that uses the _help method to automatically generate help IDs must have an associated udx file. The udx file has the same name as the SAS catalog that contains the frame or class for which help is invoked. This file is stored in a directory identified in the HELPLOC path. For example, if HELPLOC includes a path to f:\apps, help for the sasuser.myapp.myframe.frame entry requires an associated udx file named myapp.udx that is stored in the f:\apps directory.

The URL index file is a text file that contains three columns. The first two columns specify the help ID that is used by SAS/AF (which includes the container name or frame name in the first column and the object name or "1" in the second column). The third column is a unique number that is used to identify the help ID. This number can be any value as long as it is unique to the udx file. For example, consider the following excerpt from a udx file:

```
myframe    1          1000
myframe    listbox1   1001
myclass    1          1002
```

The first entry maps the help for a frame named *myframe*, the second entry maps the help topic for the *listbox1* object on *myframe*, and the third entry maps the help topic for all instances of the class named *myclass*. There must be at least one space separating each column.

URL index files are comparable to the help index (hdx) files that were required in applications built with Release 6.12 of the SAS System.

ADDING HELP TO A FRAME

One of the most useful ways to implement user assistance is to add help to a frame. Your help topic could explain the purpose of the frame, as well as how the user can complete the information presented on the frame.

To add HTML-based help for a frame:

1. Open the Properties window for the frame.
2. Select the _FRAME_ object, then select its Attributes.
3. Select the help attribute, then click the ellipsis (…) button in the Value cell to open the Object Help editor.
4. Specify the type of help. Use http:// and file:// protocols to point to a specific help location. Use helploc:// to point to a file in a directory defined in the HELPLOC path.
5. In the Value field, enter the appropriate HTML file, then click OK to close the editor.

To add CBT-based help for a frame:

1. Open the Properties window for the frame.
2. Select the _FRAME_ object, then select its Attributes.
3. Select the help attribute, then click the ellipsis (…) button in the Value cell to open the Object Help editor.
4. Specify the SAS catalog entry type, then enter the name of the SAS CBT entry in the Value field. Click OK to close the editor.
5. In the Properties window, select the CBTFrameName attribute, then specify the name of the CBT frame that is first displayed.

To invoke the help that you specify for a frame, you can either call the HELP command or run the frame's _help method. For example, you might have a push button that you want to use to call help. To set the help command:

1. In the Properties window, select the push button for which you want to add help.
2. Set the value of the commandOnClick attribute to HELP.

As an alternative approach, you could add SCL code to the push button's object label in the frame SCL. For example:

```
pushbutton1:
   _frame_._help();
   /* additional processing here */
return;
```

While the code would perform the same action as a commandOnClick attribute that is set to HELP, this solution enables you to perform more than one action when the button is selected.

ADDING CONTEXT-SENSITIVE HELP

You can also add context-sensitive help that provides assistance for specific components on the frame. Context-sensitive help is invoked whenever the user issues the HELPMODE command (or presses SHIFT+F1 on some systems). The cursor changes to a ? shape that the user can then position over components to view context-sensitive help.

To define help for a component for which you want to display context-sensitive help in HTML:

1. Open the Properties window for the frame.
2. Select the _FRAME_ object, then select its Attributes.
3. Select the showContextHelp attribute and set its value to "Yes."
4. Select the component, then select its help attribute.
5. In the Value cell, click the ellipsis (…) button to open the Object Help editor.
6. Specify the type of help. Use http:// and file:// protocols to point to a specific help location. Use helploc:// to point to a file in a directory defined in the HELPLOC path.
7. In the Value field, enter the appropriate HTML file, then click OK to close the editor.

You can also define context-sensitive help for all instances of a particular class by defining a value for the help attribute on the class itself. For example, if you want to create a standard OK button that you can use on all of your frames:

1. Start the Class Editor and create a new class named *okButton* whose parent is the Push Button Control (sashelp.classes.pushbutton_c.class).

2. In the Attributes section of the Class Editor, select the label attribute and enter *OK* in the Value cell.
3. Select the help attribute, then click the ellipsis (…) button in the Value cell to open the Object Help editor.
4. Enter the appropriate help topic, then click OK.
5. Save the class and exit the Class Editor.

You can then add the new okButton class to the Components window and use it on your frames. If context-sensitive help is enabled for the frame that contains an okButton instance, help is automatically defined.

The HELPMODE command can be issued at the command line, or by a menu selection, or as the result of some action on the frame. Many developers choose to put an icon on the toolbar and issue the HELPMODE command when it is selected.

In addition, developers of SAS/AF applications for the Windows environment can take advantage of the standard context-sensitive window control. If a frame is defined with its type attribute set to "Dialog" and its showContextHelp attribute set to "Yes", then the frame displays a "?" icon next to the Close (X) control in the upper-right of the window. Users can click this icon to start helpmode.

ADDING TOOLTIP HELP

A tooltip (also known as "balloon help") is presented as pop-up text when a user positions the cursor over a component for a particular time interval. Tooltip functionality is only available on certain desktop platforms, so the feature may not be transportable.

To add a tooltip to a component in a frame:

1. Select the component in the build-time frame, then right-click and select Properties Window from the pop-up menu.
2. Select Attributes under the component in the tree, then select the toolTipText attribute.
3. Enter the text you want to display as a tooltip.

Test your frame, then position the cursor over the component to display the tooltip.

You can also programmatically change the toolTipText attribute of a component to reflect a change in the state of your application. For example, a list box may have an attached data set list model. If the value of the model's library attribute changes, you can change the tooltip to presents a different message. For example:

```
listbox1.toolTipText='Tables in the ' ||
         datasetlist1.library || ' Library.';
```

ADDING STATUS-LINE HELP

In Windows, you can add text to display on the SAS status line when a user positions the cursor over a component. To set status line help for a component on a frame:

1. Select the component in the build-time frame, then right-click and select Properties Window from the pop-up menu.
2. Select Attributes under the component in the tree, then select the helpText attribute.
3. Enter the text you want to display on the status line. Note that a warning appears if you set this attribute in an environment that does not support status line help.

ADDING HELP TO DOCUMENT YOUR CLASSES

Several build-time windows in SAS/AF enable you to view class documentation in context with the current active class or property. For example, you can select a class in the Components window, then right-click and select Help on Class from the pop-up menu to view the documentation for that class. The Class Editor, Properties window, and Class Browser provide similar features.

In Version 8, you can make documentation available in the same way for components that you create. SAS/AF includes an experimental documentation utility that enables component developers to generate HTML files that document class, interface, resource, and frame entries. This documentation can provide assistance to developers who use the components that you create. For complete information on the GenDoc utility, refer to the *SAS Guide to Applications Development* or the SAS/AF online help.

The documentation generated by GenDoc is stored in a directory that is specified in the HELPLOC option. This directory path must end with a **classdoc** subdirectory. For example:

```
D:\SAS\classdoc\
```

SAS must be able to write to this directory. Each class that you document is output to an HTML file that has the same name as the class and is stored under subdirectories that designate the class's library and catalog. For example, if you document the class entry *myorg.classes.Account.class*, the output is written to

```
D:\SAS\classdoc\myorg\classes\Account.htm
```

Likewise, the documentation for the *B.intrface* entry stored in the same catalog would be written to

```
D:\SAS\classdoc\myorg\classes\B.intrface.htm
```

You can also create your own HTML files to document your classes without using GenDoc and make them accessible by storing the files in the same directory structure.

When a user of the class selects "Help on Class" from a build-time window such as the Class Editor, SAS searches the HELPLOC path for the appropriate class HTML file. If your organization maintains a library of SAS/AF classes, you can make the documentation for those classes available for all developers. To provide access to the documentation, you could copy the HTML files to a network location or intranet and have developers add that path to their HELPLOC options. For example, if the documentation is available on an intranet, the HELPLOC path might include:

```
http://myorg.com/mis/classdoc
```

OTHER USER ASSISTANCE TIPS

With the support that SAS/AF software now provides for displaying Web pages, user assistance is not limited to what a developer could pack into a CBT entry or a native help file. HTML enables a developer to add user assistance that can include detailed graphics, sound, and even animation. These capabilities make the help system more interactive, enabling users to refine help topic selection, use check boxes and radio buttons to alter views of information, and apply highlights to key points using standard browser events such as onMouseOver.

Since HTML makes it easy to link to virtually anywhere, this technology can be used to expand the assistance that an organization provides to a user of a SAS/AF application. The following list provides possible applications of user assistance:

- **Add remote support for your applications**. By adding help attributes that use the http:// protocol or other features that invoke the WBROSE command, you can direct users from the application to a location that, if appropriate, enables a developer to update help files regardless of where the application is deployed. These help files could contain MAILTO or other email-savvy scripts through which a user could report problems or ask for assistance.
- **Direct users to information assets.** Information that is available on a corporate intranet can be presented to users to help them make decisions about how to proceed. The frame can contain a button that displays a company's policies and procedures. You can add SCL that checks the state of either data or the application itself to determine whether a user would benefit from viewing a report or other Web page. With such a feature, the user assistance extends beyond help and, instead, delivers more performance support.

- **Include "training card" assistance.** You can create training cards or "cue" cards, which are specialized help topics that are delivered based on conditions within a program. They are particularly useful in guiding users through a procedure step-by-step. Whenever the user completes a step, the documentation for the next step can automatically appear. If a user incorrectly performs a step, information can appear that specifically addresses the mistake. Training card help is initiated either from a user selection in a help topic or by actions a user takes in a program.

  For example, you could create a non-visual training card object and add it to a frame. The object has an attribute of type List that you can use to add the names and values of the visual controls on the frame. You could add a method that checks the state of the application by checking the values of the controls on the frame. It then displays the appropriate "training card" topic. For example:

```
useclass sasuser.test.trainingcard.class;
tutor: public method;
  /* check the state of the application  */
  /* by reading the state list attribute */
     /* some processing here  */
  if some-state-value =1 then
  call execcmdi('help helploc://step1.htm"');
  else if some-state-value =1 then
  call execcmdi('help helploc://step2.htm"');
    /* additional condition processing  */
endmethod;
enduseclass;
```

  A method call is added to the object label of each control on the frame. For example:

```
pushbutton1:
   /* other processing here */
   trainingObj.tutor();
return;
```

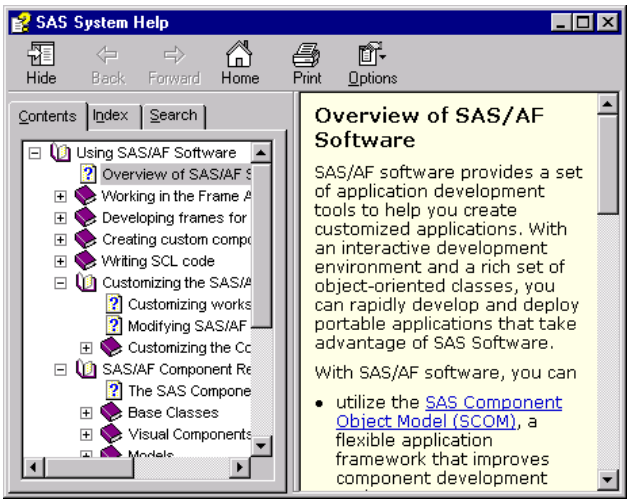
  The actual implementation is left to you to design, based on the needs of your application and its users.

USING WINDOWS-SPECIFIC FEATURES

Many developers who deliver applications on the Windows platform have invested time and resources into creating Windows Help (WinHelp) files. WinHelp was, for many years, the de facto standard for Windows-based help. Microsoft HTML Help is the next-generation of online help for the Windows environment. If you have used WinHelp, you will be familiar with many of the features of HTML Help. Version 7 and Version 8 SAS software on Windows both provide online help via HTML Help files.

Like WinHelp, HTML Help uses a project file to combine topic, contents, index, image, and other source files into one compiled help file. A compiled file consists of a compressed form of HTML that greatly reduces the amount of disk space required for your help system. It supports standard HTML, ActiveX, Java, scripting languages such as JavaScript and JScript, and HTML image formats (.jpeg and .gif files).

HTML Help consists of an online Help Viewer and related help components. The Help Viewer uses the underlying components of Microsoft Internet Explorer to display help content (which means Internet Explorer is currently required to be loaded on systems that make use of HTML Help).



Display 2. *Help for SAS/AF displayed in the HTML Help Viewer*

HTML Help contains the following components:

HTML Help ActiveX control
: a small, modular program used to insert help navigation and secondary window functionality into an HTML file

The HTML Help Viewer
: a fully-functional and customizable three-paned window in which online help topics can appear

The HTML Help Java Applet
: a small, Java-based program that can be used instead of an ActiveX control to insert help navigation into an HTML file

The HTML Help executable program
: the program that displays and runs help when you click a compiled help file

The HTML Help compiler
: the program that compiles project, contents, index, topic, and other files into a compiled help file.

HTML Help is available from Microsoft as a free tool. You are free to redistribute HTML Help files with your application. It is available through several Microsoft developer Web pages, including:

```
www.microsoft.com/workshop/author/htmlhelp/
```

The download includes all components listed above, as well as a help authoring tool called HTML Help Workshop. This tool provides a basic system for creating and managing help projects, HTML topic files, contents files, and index files. HTML Help Workshop also includes a feature that enables you to convert WinHelp projects into HTML Help projects. Other third-party tools such as Blue Sky Software's RoboHTML and ForeFront Software's ForeHTML are additional resources for developing HTML Help files and other HTML-based help.

CREATING AN HTML HELP FILE

Developing a compiled HTML Help file should be a familiar process for developers who created WinHelp files. HTML Help requires a project (.hhp) file, which is a text file that brings together all the elements of a help project. It contains the data that the HTML Help Compiler (hhc.exe) needs to combine topic (.html, .htm), image (.jpeg, .gif, .png), index (.hhk), and contents (.hhc) files into a single compiled help (.chm) file.

To create an HTML Help file for your application:

1. Create the HTML pages that you want to include in the help file. You can include appropriate graphics, scripts, style sheets, and other features of Web-based development. Make sure that you store the files in the appropriate directory (see below).
2. Create the project file. The project file also contains

information about how a compiled help file will appear. Window definitions you create in the project file determine attributes of your help windows, such as size and position. It is recommended that you use a tool such as HTML Help Workshop to create HTML Help project files.

3. Add a table of contents and/or index as necessary. Again, it is recommended that you use an HTML Help authoring tool to add these features.
4. Compile your project using the hhc.exe compiler provided by Microsoft.
5. Test your help system to make sure everything appears the way you want it.

Certain requirements exist when you create an HTML Help file for a SAS/AF application. First, all HTML files that are called by the application must exist in a directory that is named the same as the compiled HTML Help (.chm) file and is a subdirectory of the path containing the project (.hhp) file. For example, the following files could comprise an HTML Help-based help system for an application that exists in the file *myapp.chm*:

```
myapp.hhp
myapp.hhc
myapp.hhk
myapp\main.htm
myapp\edit.htm
myapp\sales_1.htm
myapp\signon.htm
```

Second, you must follow specific naming conventions for the files and directories that comprise your help project if you *do not* add a specific helploc:// value to the help attribute of a frame or object. In other words, if your application relies on the _help method to generate help IDs, you must follow these guidelines for naming your files:

- The HTML files must use the same format as defined above for help calls that require .udx files. For example, help for a frame named *sales* that has **no** defined help attribute must have an HTML file named *sales_1.htm*.
- HTML files must be compiled from a directory that has the same name as the SAS catalog for which you want to add help. For example, to add help for a frame in the SAS catalog entry *myorg.myapp.sales.frame*, you would add HTML files to a directory named *myapp*.
- The resulting compiled HTML Help (.chm) file must have the same name as the SAS catalog containing the frame(s).
- If your application uses IDs from the _help method and makes use of frames from different catalogs, you must create a separate HTML Help (.chm) file for each catalog.

CALLING A COMPILED HTML HELP FILE

To invoke a compiled HTML Help file from a SAS/AF application, you can use one of the following:

- Specify the helploc:// protocol for the help attribute on the appropriate frame or object. Set the value using the form

  ```
  helploc://helpfile/htmltopic.htm
  ```

  For example, if the .chm file is named *myapp.chm* and the topic you want to call is in *howto.htm*, the value would be set to

  ```
  helploc://myapp/howto.htm
  ```

  Note that you must place the .chm file in a path specified by the HELPLOC option.
- If you do not set the help attribute and instead rely on the _help method functionality, you need to include the .udx file in the same directory in which you place the appropriate .chm file. The _help method locates the topic in the .udx file, then opens the topic with the generated ID as the file name.

The Windows version of SAS will find the help topics in a .chm file if a directory and HTML file of the same name do not exist in the same or other HELPLOC path. In other environments such as Unix, the same help call simply looks for an HTML file the designated directory. This feature enables you to develop your SAS/AF application with one form of a help call that works on all supported platforms.

CONCLUSION

In a best-case scenario, the user interfaces that you design for your applications will be intuitive and easy to use. In fact, your goal should be to reduce the amount of help that is required by an application. SAS/AF software helps you reach this goal by adding improved visual controls, which enable you to create more intuitive interfaces. When those interfaces *do* require help, the new object properties enable you to specify help while you are designing the application.

SAS/AF software demonstrates another goal of user interfaces: to provide help in context with a particular task. In the build-time environment, SAS/AF developers can view help on a specific class (or class property) from within certain windows. The class documentation tool (GenDoc) enables component developers to provide the same kind of information for their “users” – that is, other developers.

As help and user assistance play a more significant role in the usability of applications, it is important that users find that they can get assistance from more than just a separate window that floats above or behind the software application. Most importantly, perhaps, is the need to consider user assistance as an integral part of applications development, not just as part of the same development cycle, but, in some cases, as part of the same code as the software application itself.

REFERENCES

Kruth, John (1997), “Native Help Technology in SAS/AF Applications,” *Proceedings of the Twenty Second Annual SAS Users Group International Conference,* 22, 1520-1523.

SAS Institute Inc. (1999), *SAS Guide to Applications Development, First Edition,* Cary, NC: SAS Institute Inc.

ACKNOWLEDGMENTS

I would like to thank Jeff Diamond and David Henderson of SAS Institute who were instrumental in implementing the new help features in SAS software. I would also like to thank Chris Barrett and Joy Lashley, who reviewed this paper and offered excellent comments to help make it more usable.

CONTACT INFORMATION

Marty Tomasi
SAS Institute Inc.
SAS Campus Drive
Cary, NC 27513
sasmdt@wnt.sas.com