Native Help Technology in SAS/AF® Applications

John Kruth, SAS Institute Inc.

ABSTRACT

Attaching native help to a SAS/AF application can be a complex task. This paper will examine some of the tools and techniques available to both help authors and application developers to simplify this task. Topics will include:

- Details of the expanded _HELP_ method available in Release 6.12
- How the SAS® System calculates help topic identifiers for SAS/AF objects, and how you can plan your help system to answer those identifiers
- How to invoke context-sensitive help for FRAME objects
- How to provide access to your help system from your application's help pmenu

INTRODUCTION

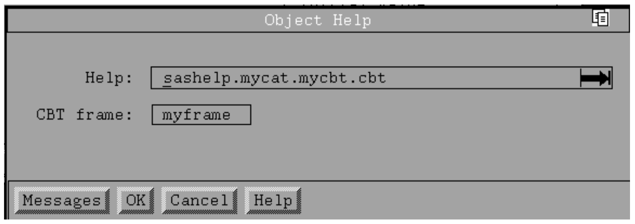
With the introduction of native help to the SAS System in Release 6.11, SAS/AF developers can add native help to their applications. In Release 6.11, determining help IDs for these native help files involved a complex process of deciphering the help attributes that referred to CBTs and translating them into native help information. The enhancements made to the _HELP_ method in Release 6.12 simplify the process of determining help IDs and make it easier for SAS/AF developers to attach help to their applications. In this paper I will describe the changes made to the _HELP_ method and supply SCL code that can be added to SAS/AF applications to invoke help.

THE _HELP_ METHOD

The _HELP_ method enables a SAS/AF developer to call a help topic for an object in a FRAME application. The method determines whether SAS is configured to deliver **native help** or **CBT-based help** and displays the help topic in the appropriate format. This method is defined in the *WIDGET* class, and any object derived from the *WIDGET* class will have the _HELP_ method defined in its method list. Since every object which is displayed on the screen in a FRAME application must be derived from the *WIDGET* class, any visible object can use this method.

Using the _HELP_ Method Before Release 6.12

In previous releases of SAS (including Release 6.11), the _HELP_ method determined which help topic to display based on the help ID that was supplied by the SAS/AF developer. The help ID was specified by setting the *Object Help* attribute for the object using a format that specified a CBT name and a frame in that CBT. For example:


Object Help

Help: sashelp.mycat.mycbt.cbt

CBT frame: myframe

Figure 1 *Help Attribute Screen for a SAS/AF Object*

When the _HELP_ method was called, it would determine whether SAS was configured to deliver native help or CBT-based help. If CBT based help was specified, the CBT indicated in the help ID would be located and the frame indicated would be displayed in the SAS CBT viewer. If native help was specified, the _HELP_ method used the help ID to determine which help file should be opened and which topic within that help file contained the help information for the object.

Native help information was extracted by using the following method:

1. The help file had the same name as the catalog indicated in the CBT portion of the help ID.
2. The ID of the help topic in the help file was created by concatenating the CBT name with the frame name and preceding the CBT name with the letter **C** and the frame name with the letter **F**. For example, the help ID in the figure above specifies that the help for that object is stored in the *mycat.hlp* file and that the topic ID would be *CmycbtFmyframe*.
3. The _HELP_ method invoked the native help viewer appropriate for the platform and displayed the correct help information.

Enhancements to the _HELP_ Method for Version 6.12

Although the Release 6.11 version of the _HELP_ method enabled a developer to use the native help viewer when displaying help, it still required the application developer to supply unique help IDs for each object in the system, set an attribute for each object, and notify the help author of each of these help IDs. In Release 6.12, the _HELP_ method has been enhanced to remove the burden of creating help IDs from the developer and to allow the help author to predict the help IDs that will be needed for any object in a SAS/AF application. In addition, the new _HELP_ method makes it possible for a writer to create *default* help which will be displayed if no other help is available for an object.

The enhanced method generates a unique help ID for each object in a SAS/AF application based on the context of the object within the application. Since each frame in a SAS/AF application can be identified by a unique four level name (for example, sashelp.mycat.myframe.frame), and since each object within a frame must have a unique name, a unique help ID is created by using this information. The help file is identified by the catalog in which the frame is stored. For example, the help information for a frame named *sashelp.mycat.myframe.frame* would be in the *mycat.hlp* file. The help information for each object within that frame would also be in the *mycat.hlp* file.

The topic ID, which indicates which topic within the help file should be displayed, is created by concatenating an underscore

(_) and the character *1* on the end of the frame name. So, the topic ID for *sashelp.mycat.myframe.frame* would be *myframe_1*. This topic ID would be used in an RTF or IPF file to identify the help topic for that frame.

Specific object help (or context-sensitive help) can also be created for each object within a frame. The topic ID for any object in a SAS/AF application is created by concatenating an underscore and the object's name onto the name of the object's container (in this case, the frame). So, if *sashelp.mycat.myframe.frame* contains an object called *mybutton*, the topic ID for that object would be *myframe_mybutton*, and that topic would be in the *mycat.hlp* file.

IMPORTANT: The _HELP_ method only generates these help IDs if there is not a Release 6.11 style help attribute defined for an object. If a 6.11 style help attribute is supplied, the _HELP_ method determines the help file and help ID based upon the methods described earlier in this article. If you want to suppress the 6.11 style help attribute and allow the help ID to be generated by the enhanced _HELP_ method, you must remove the CBT and frame information from the help attribute screen for that object.

Determining an Object's Name

The _HELP_ method uses the object's *LABEL=* attribute to determine the name of the object. When the *NAME=* attribute is set by the application developer, the *LABEL=* attribute is automatically set to the same value as the *NAME=* attribute.

To determine an object's name:

1. Using *build mode*, open the frame that contains the object. You can do this by double clicking on the frame in the BUILD window or by issuing the *BUILD <frame>* command at the command line.
2. Click on the object whose name you want to find.
3. Open the OBJECT ATTRIBUTES window by choosing *Object Attributes* under the *Locals* pmenu choice.
4. The text in the *NAME:* field is the name used by the _HELP_ method.

Names for Widgets Created Programmatically

Developers sometimes create widgets programmatically in an SCL program. Since these widgets are dynamically created while the application is running, by default, the *LABEL=* attribute is not set for these widgets, and object help cannot be created for them. In order to create object help for dynamically created widgets, the developer must set either the *NAME=* attribute or the *LABEL=* attribute for the widgets. This is done by adding these attributes to the *attribute list* that is passed to the *NEW* method that is used to create the widget. See your SAS/AF documentation for information on the *CLASS class* and the *NEW* method.

IMPORTANT: The *NAME=* or *LABEL=* attribute must be set for every dynamically created widget even if the widget is documented using *class* or *default* help.

DEFAULT OR CLASS BASED HELP

Within a SAS/AF application, a specific type of object (or object class) may be used repeatedly in different frames. If these objects are all used in the same way, it may be more efficient to write a single, default help topic, rather than writing separate help topics for each object. The enhanced _HELP_ method provides a simple way to create default help topics. The help ID for this default topic is the name of the object class with an underscore and *1* concatenated on the end of it.

If you have a class called *myclass* and you use multiple instances of the *myclass* class within the *sashelp.mycat* catalog, you can supply help for all of these instances by creating a single help topic called *myclass_1* in the *mycat.hlp* file.

This default or class help will only be called if no specific object help exists. This allows a writer to create default help for every instance of an object class, and to supply different help for an object which is derived from the same class but is used in a different way.

For example, suppose there are two frames, *sashelp.mycat.frame1.frame*, and *sashelp.mycat.frame2.frame*, and each frame has an object within it named *mybutton*. Both of the *mybutton* objects are instances of the *myclass* class. To supply default help for both of the *mybutton* objects, you could create the topic *myclass_1* in the *mycat.hlp* file. In fact, this help topic is not only the default help for these two objects, but it is also the default help for any objects within the *mycat* catalog that are instances of the *myclass* class.

If the *mybutton* object in *frame2.frame* must be documented differently than the majority of the instances of the *myclass* class, you can override the default help for that object by creating *object specific help* with the help ID *frame2_mybutton*. This allows you to supply default help for all instances of the *myclass* class, but, where it is appropriate, also supply object specific help for the instances which must be documented differently.

COMPOSITE WIDGETS

Because composite widgets are usually a group of objects which are stored as a single, composite object, composite widgets actually act as containers for other objects. When a help ID is generated for the objects inside a composite widget, the name of the composite widget is used as the container and therefore, the first portion of the help ID.

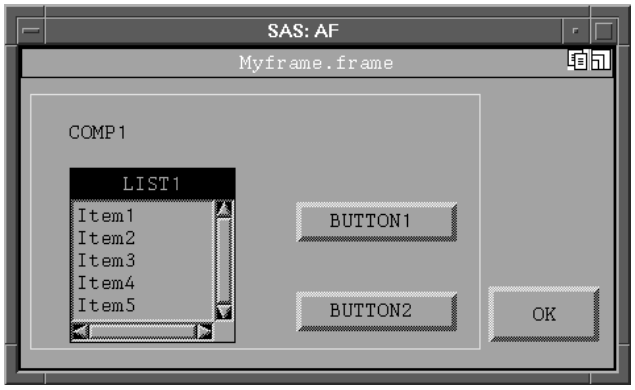


Figure 2 *Example Frame MYFRAME.FRAME*

The above frame is called *myframe.frame* and it contains two widgets; one named *OK* and another, which is a composite widget, named *COMP1*. *COMP1* is an instance of a composite class called *mycomp.class*. The *mycomp.class* class is made up of 3 objects, *LIST1*, *BUTTON1*, and *BUTTON2*.

The help IDs for each of the objects within this frame would be generated as follows :

- For the entire frame, the help ID is *MYFRAME_1*.
- For the *OK* object in the frame, the ID is *MYFRAME_OK* .
- For the entire composite widget (*COMP1*), the ID is *MYFRAME_COMP1*.
- The objects within the composite widget, *COMP1*, would be identified with the help IDs *COMP1_LIST1*, *COMP1_BUTTON1*, and *COMP1_BUTTON2*.

Notice that when we are generating the help IDs for the objects within the composite widget, the container portion of the help ID is the name of the composite widget and not the name of the frame.

DEFAULT HELP AND COMPOSITE WIDGETS

Just as it is possible to create default help for other classes within SAS/AF, default help can be created for composite classes and for the objects inside the composite classes. A composite class is treated like any other class in SAS/AF. To create a default help topic for the entire composite class, concatenate an underscore and a *1* onto the name of the class. So, for the *COMP1* object in the frame above which is an instance of the class *mycomp.class*, the default help ID for the entire composite widget is *MYCOMP_1*.

The objects in the composite may also have default documentation. When creating default help for the objects in a composite widget, use the composite class name as the container (or the first part of the ID) and use the name of the object in the composite as the second part of the ID. For the three objects in the *COMP1* composite object, the IDs for the default help are *MYCOMP_LIST1*, *MYCOMP_BUTTON1*, and *MYCOMP_BUTTON2*. Notice that when you are writing default help, the **class name** (MYCOMP) is the container, but when you are writing instance specific help, the **instance name** (COMP1) is the container. Again, the default or class help will only be used if there is no instance help available for an object.

OTHER FEATURES OF THE _HELP_ METHOD

Search Order

When help is requested for an object within a frame, the _HELP_ method searches for the help IDs for an object by using the following search order:

1. The _HELP_ method searches for instance-specific help on the object.
2. The _HELP_ method then searches for default or class help for the class that is associated with that object.

   NOTE : If the object is a part of a composite widget, class help is determined in two steps. First, the _HELP_ method looks for class help in the context of the composite widget. If that is not found, it looks for class help for the object itself.
3. The _HELP_ method then searches for help on the object that contains the object for which help was originally requested. This may be the frame that the object is in or the composite widget that this object is part of.

This hierarchical search path makes it possible for an author to document only the specific items in the frame that need context-sensitive help. As long as each frame is documented, the user will get some help even if the specific object that help was requested for does not have help written for it.

NOTE : This search order only applies if there is no CBT and frame attribute specified for the object. See the note in the section above describing the enhancements to the _HELP_ method in Release 6.12.

Unique Catalog Names

Because the _HELP_ method uses the catalog name to identify the help file, the catalog names used for the applications must be unique in order for the help system to work correctly. There cannot be two files called *base.hlp* in the help directory. A BASE catalog cannot exist in a library other than the SASHELP library. For the _HELP_ method to work correctly, all of the catalog names within the SAS System must be unique regardless of whether or not they are stored in different libraries.

Popups vs. Topic Windows

Help for a specific instance of an object is delivered in a popup window on systems where popups are available. Help for a frame and default help for classes is delivered in a full help viewer topic window. If popups are not available on the host platform, all help information is delivered in a full viewer window. To simplify this, if the help ID ends with *_1*, the help information is delivered as a full topic window. All other help is delivered in a popup window.

CALLING HELP FROM A SAS/AF APPLICATION

Now that we have an understanding of how the _HELP_ method generates help IDs for objects in a SAS/AF application and how help authors must use this information when writing their help files, let's examine what the SAS/AF developer must do to enable help to be called from within an application.

Context-Sensitive or Object Help

The *helpmode* command has been created to allow SAS/AF developers to put the SAS system into context-sensitive help mode. When the *helpmode* command is issued at the command line, the cursor changes to a *?* and the SAS System prepares to deliver object help on the next item that is clicked upon. When an item is chosen while in *helpmode*, the _HELP_ method is called for that object.

The *helpmode* command may be issued at the command line, or by a menu choice, or as the result of some action like a push button being clicked. Many developers choose to put a context-sensitive help icon on a toolbar and issue the *helpmode* command when it is selected.

Help Buttons

Most applications have frames with a help button. To use a help push button to deliver help on an entire frame, send a message to the frame to call the _HELP_ method.

For example, when the push button is clicked, issue the following statement in the SCL code:

```
call send(_frame_, '_HELP_');
```

This statement displays the help information that was written for the frame that contains the push button.

At times, the help button is not intended to deliver help on an entire frame, but only on a specific object within the frame (like the active page in a tabbed dialog box). In those cases, the developer must maintain a variable that contains the widget ID of the object on which help should be delivered (eg. the current page). When the help button is clicked, the SCL code should call the _HELP_ method for that widget.

For example, if the variable that contains the widget ID of the object is *CURR_OBJ*, the SCL code for the help button should be:

```
            call send(CURR_OBJ, '_HELP_');
```

Using *call send* to invoke the *_HELP_* method allows you to deliver help on any object in SAS/AF.

Opening a Specific Help Topic

You may wish to bring up a specific help topic that is not associated as object help. For example, a menu item might be used to bring up a table of contents topic for some online reference documentation. To call a specific topic, you must use a Release 6.11 style help call.

To bring up a specific topic, first be sure that the topic ID for the topic conforms to the Release 6.11 style. The ID must be in the format:

```
            CcbtnameFfname
```

Once you know the topic ID that you wish to display, the following type of command must be issued :

```
     help c=sashelp.helpfile.cbtname.document
          frame=fname
```

Where *helpfile* is the name of the help file that contains the topic and *cbtname* and *fname* correspond to the first and second parts of the topic ID.

Finally, be sure that you have a help index file (see the section below) which contains an entry for your help ID.

PRACTICALITIES

Help Index (hdx) Files

Help index (or hdx) files are used by the SAS System to map textual context IDs to numerical context IDs. Every native help file that is connected to a SAS/AF application must have an hdx file associated with it. The hdx file must have the same name as the help file, but it must have the extension **.hdx** instead of the .hlp extension.

When you build your native help file, you can associate the textual context IDs with numeric IDs. When using WinHelp or Helplus (a proprietary help viewer developed by SAS Institute for UNIX platforms) this is done via the *help project file*. In OS/2, this is done by including a *res ID* for the topics. Details on this mapping process can be found in the documentation for the viewer or help compiler that you are using.

The help index files have a three column format. The first two columns are the two portions of the help ID (the container and object name). The third column is the numerical ID that is mapped to the textual context ID in your help file. The following is an example of an hdx file:

```
frame       1         1000
frame       object    1001
class       1         1002
class       object    1003
```

The first entry indicates that the *frame_1* context ID is mapped to the numeric ID *1000*. The second entry shows the *frame_object* context ID mapped to *1001*. The spacing is not important in the hdx files, but there must be at least one space between the columns.

An hdx file should only contain entries for objects that are documented in the help file. If default help (or class help) is used for an object, the hdx file should only have an entry for the ID for the class help and not the ID for the instance help.

For more information about help files and hdx files and how they should be created and formatted, see the article entitled *Developing Native Help for SAS/AF Applications* in the SUGI 21 proceedings.

Helploc

For the help system to find your help information, your help files **and** hdx files must be available in the help location specified by the *helploc* option in SAS. You can create your help files and store them in the directory specified by helploc, or the *helploc* may be changed when you start a SAS session. To reset the helploc option, issue the *-helploc <path>* command line option. For more information see *SAS Software : Changes and Enhancements, Release 6.11*.

ACKNOWLEDGEMENTS

I would like to thank Elizabeth Brownrigg, Jeff Diamond, and Philip Shelton of SAS Institute who reviewed this paper. I would also like to thank Jeff Polzin, Deva Kumar, and Jeff Diamond for thier help in implementing these features in Release 6.12 of the SAS System. And finally, I would like to thank Chris Hemedinger for his expertise and assistance in understanding native help files and how they are created.

John Kruth, SAS Institute Inc., SAS Campus Dr., Cary NC 27513, (919)677-8000 x5841, sasjgk@unx.sas.com