CHAPTER
3

SAS/AF Catalog Entry Types

Overview

You can use the BUILD procedure in SAS/AF software to create and edit the following types of catalog entries. Other types of entries (for example, SLIST entries containing SCL lists or PMENU entries containing menu definitions) can appear in the catalog along with these entry types, but you cannot edit other types with the BUILD procedure.

Entry Type	Purpose
CBT	Provides sequences of text and responses to user input for tutorials or computer-based training courses. You can also build Help facilities with CBT entries.
CLASS	Stores class definitions for FRAME objects, including the attributes, methods, events, event handlers, and interfaces that are used within a class. See also the RESOURCE entry type.
EDPARMS	Stores default colors, highlighting attributes, and general editing specifications for the SAS text editor that is used to build the displays for CBT, HELP, MENU, and PROGRAM entries.
FORM	Stores printing device, paper, destination, and special print control information. The form information is used when output is routed to a printer.
FRAME	Stores graphical user interfaces for object-oriented applications.
HELP	Provides assistance and instructions for users.
INTRFACE	Stores method definitions that determine whether and how model/view FRAME components can communicate.
KEYS	Associates commands with function keys.
LIST	Stores lists of values that are used to validate user input to fields in PROGRAM entries.
MENU	Provides menus of options that users can select to run other entries.
PROGRAM	Stores the display, field attributes, and the SAS Component Language program code for character-based applications.
RANGE	Stores range definitions that control traffic lighting in FRAME entry objects.
RESOURCE	Stores a collection of classes for FRAME applications.
SCL	Stores a SAS Component Language (SCL) program and its compiled code, but does not include a DISPLAY window.

The EDPARMS, FORM, and KEYS entries are catalog entry types that the BUILD procedure opens as a convenience for application developers. Refer to base SAS documentation for information about the EDPARMS, FORM, and KEYS entries. The remaining SAS/AF entry types are discussed in the following sections.

CBT Entries

CBT entries store interactive user assistance or tutorial applications, which consist of

- □ a display that provides information and questions to users and which accepts user responses
- □ general attributes that control the appearance and behavior of the window in which the CBT entry executes
- □ a child attribute that specifies another entry to which control can be passed when users reach the last frame in the CBT entry.

The following sections describe each of these components of a CBT entry.

CBT Entry Displays

You use the BUILD procedure's DISPLAY window to design the displays for CBT entries. The displays can use any of the text color features and highlighting features that the SAS text editor supports. In addition to static text, the display can include fields in which users can enter or select answers to questions, as well as graphics.

The display for a CBT entry is divided into a sequence of *frames*. Frame boundaries in the display are indicated with either a frame indicator line or a divider line that consists of dash (-) characters across the full width of the DISPLAY window. (You can use the FILL command in the DISPLAY window to create divider lines.) Refer to "Frame Indicator Syntax" on page 25 for information about the syntax of frame indicator lines.

In addition to presenting information to users, frames in CBT entries can pose both fill-in-the-blank and multiple-choice questions. Refer to "Query Frames" on page 24 for details about creating frames that present questions. If the frame contains a question for users, it must begin with a frame indicator line, and it must include one or more feedback indicator lines that determine how the entry responds to user input. Refer to "Feedback Indicator Syntax" on page 28 for information about the syntax of feedback indicator lines.

If a frame does not present a question, users can press ENTER to advance to the next frame in the sequence. If the frame presents a question, users must either attempt to answer the question or use the FORWARD command to skip the question. Users can issue the BACKWARD command to scroll back to previous frames in the sequence. When a user issues an END command to close the CBT entry, the current entry name is stored as the AF checkpoint (unless the CHECKLAST=NO option was specified in the AF command that started the application). Users can issue the SAVE command to save the current frame number and end the current SAS session. When the user opens the CBT entry again, it resumes at the frame that was displayed when the SAVE command was issued.

You can define frames that branch unconditionally to other SAS/AF catalog entries. To define a frame that jumps to another entry, use a divider line to begin the frame, and enter three uppercase P characters in the first three columns of the next line. Follow the `PPP` with the name of the entry to open.

Scrolling Controls

If you design a frame that has more lines than the current window size, only the number of lines that fit in the window are initially displayed. Users must issue a FORWARD command to display the remaining lines of the frame. You can designate a portion of the frame that does not scroll. Enter three caret (^) or NOT (¬) characters in the first three columns of a line to delineate the nonscrolling region of the frame. Any text and fields above the line that contains the ^^^ or ¬¬¬ remain visible as long as the current frame is displayed; FORWARD and BACKWARD commands scroll only the region below the nonscrolling area.

You can define pause indicators in the display to delay the presentation of portions of the text. Enter three at (@) characters in the first three columns of a line to define a pause. Only the text between the beginning of the frame and the first pause indicator (@@@) appears when the frame is initially displayed. When the user presses ENTER, the text from the current pause indicator up to the next pause indicator (or up to the end of the frame, if there are no more pause indicators) is added to the frame, and so on.

You can use the LOCK option in the frame indicator to segment frames. A frame indicator line with the LOCK option ends a sequence of frames. Users cannot press the ENTER key or issue FORWARD or BACKWARD commands to move into or out of a locked frame. Locked frames are displayed only when they are specifically called, such as in a branch from a feedback item in another frame.

Query Frames

CBT entries can pose either fill-in-the-blank or multiple-choice questions. If a frame poses a question, the user cannot press the ENTER key to move to the next frame without attempting to answer the question. However, the FORWARD command can be used to skip the question, unless the field is locked.

You designate the response field for a fill-in-the-blank question with an initial ampersand (&), followed by underscore (_) characters to pad the field to the length required to hold the largest answer value. A response field can be as short as a single & or as long as the width of a display line. The ampersand and pad characters do not appear when the frame is displayed to the user.

Use the CORRECT= option in the frame indicator to specify the correct answer to the fill-in-the-blank question. You can use feedback indicators to define the entry's response to correct or incorrect answers. The feedback indicators can either display messages or branch to other frames or entries.

Designate the response fields for multiple-choice questions with underscore (_) characters. The underscore for each response field should be preceded and followed by a space. You can use up to eight multiple-choice response fields in a frame. When the frame is displayed to users, they can use the TAB key to move the cursor from field to field, and they can either press ENTER or click the mouse to select the desired field.

Each multiple-choice response field should have a corresponding feedback indicator that specifies the entry's response to the selection. The feedback indicators can either display messages or branch to other frames or entries. Use the C option in the feedback indicator to indicate which responses are considered correct.

You can collect information about the user's responses to the questions in the CBT entry. The response statistics are stored in a SAS data set. Refer to the descriptions of the QUIZ= and QUIZ options in "Frame Indicator Options" on page 25 for details.

The AF task creates the following macro variables when a CBT entry is executed:

- &_NQSEEN, which stores the number of questions presented to the user
- &_NQRIGHT, which stores the number of questions that the user answered correctly.

You can use these macro variables in other SAS programs after the CBT entry ends.

In addition to response fields, you can define the following other methods for enabling users to interact with the frames in a CBT entry:

- You can use the SELECT= option in feedback indicator lines to create selection boxes, which are areas of the display in which users can either press ENTER or click the mouse to select the corresponding feedback item.
- You can use the MENU= option in feedback indicator lines to define values that users can enter on the application window's command line to select the corresponding feedback item.

The feedback items for selection boxes and menu choices can either display a message or branch to a specified frame or entry.

Frame Indicator Syntax

The general form of a frame indicator is

? <* | *n*> <*options*>

where *options* can be one or more of the following:

AUTO | AUTO=*n* | NOAUTO

CORRECT=*answer-value* | '*answer-string*' | ?

GRAPH=<(*left-col*, *right-col*, *top-row*, *bottom-row*)> *libref.catalog-name.graphic-entry* </ERASE>

LOCK

NAME=*frame-name*

QUIZ

QUIZ=<*libref.*>*response-data-set*

SOUND | MUSIC=*freq-1 duration-1* <*... freq-n duration-n*> | *note-1 duration-1* <*... note-n duration-n*>

WRONG=<*libref.catalog-name.*>*entry-name.entry-type*

Use the slash character (/) to continue a frame indicator across multiple lines of the display. For example, the following lines comprise a single frame indicator:

```
?2 name=mean /
   correct=42 /
   wrong=review.cbt
```

Frame Indicator Options

You can use the following options in frame indicators:

*
: indicates that the remainder of the line is a comment.

n
: specifies how many attempts the user is given to provide the correct answer to the question in the frame. The value for *n* must be in the range 1 to 8, and it must appear in column 2 of the feedback indicator line.

  If you omit the *n* option, feedback indicator lines in the frame are ignored, so the value of *n* should always be at least 1 if the frame includes feedback indicator lines.

  If the user fails to enter or select the correct answer to the question within the allotted number of attempts, then by default the correct answer is displayed, and the user is prompted to press ENTER to continue. However, if the frame indicator includes the WRONG= option, then control passes to the specified entry.

AUTO
AUTO=*n*
NOAUTO
: specify the beginning or end of a sequence of frames that are displayed without waiting for user input. Use the AUTO option to display frames as fast as the display device allows. Use AUTO=*n* to specify the rate at which frames are displayed, where *n* is the number of frames per second to display.

  By default, frames from the current sequence are displayed until a query frame is encountered or until the last frame of the sequence is displayed. Use the NOAUTO option to stop the automatic display before the last frame is reached.

CORRECT=*answer-value* | '*answer-string*' | ?

specifies the correct answer when the frame includes a fill-in-the-blank question. The answer can be up to 32 characters long. Enclose the answer string in single or double quotes if it contains embedded blanks.

If you specify CORRECT=?, then any answer that a user enters in the response field is considered correct.

GRAPH=<(*left-col*, *right-col*, *top-row*, *bottom-row*)> *libref.catalog-name.graph-entry* </ERASE>

specifies a graph to be displayed in the frame. The graph must be a catalog entry of type GRSEG created with SAS/GRAPH software. You must specify the *libref.catalog-name* portion of the entry name, even if the entry resides in the same catalog as the CBT entry.

Note: In order for users to see the graph when the CBT entry runs in the application window, SAS/GRAPH software must be licensed at their site, and their display devices must support SAS/GRAPH output. △

By default, the graph is displayed starting on the second row of the display area to leave room for a line of text above. The display must contain enough blank lines so that the graph does not overlay any text. You can specify left and right column values and top and bottom row values to control the position of the graph within the display. The position values must be enclosed in parentheses. The following rules apply:

- the *left-col* value must be greater than 1 and less than the number of columns in the display. (Column 1 is reserved for frame and feedback indicators.)
- the *right-col* value must be greater than 2 and less than the number of columns in the display.
- the *top-row* value can be 1 or greater, but it must be less than the number of rows in the display minus 2.
- the *bottom-row* value must be greater than 1 and equal to or less than the number of rows in the display minus 2.

By default, any new graph that you display overlays any previous graph. Add the /ERASE option to erase any previous graphs before displaying the current graph. To erase the previous graph without displaying a new graph, specify the following:

```
graph=erase.erase.erase/erase
```

LOCK

specifies a frame that is not part of a sequence. Users cannot use the ENTER key or the FORWARD and BACKWARD commands to scroll into or out of locked frames. Locked frames are displayed only when they are explicitly called, such as when they are the target of a branch in a feedback indicator. To exit from a locked frame, a user must either answer a question that branches to a different frame or issue an END command. The END command returns to the CBT frame that called the locked frame.

NAME=*frame-name*

specifies a name for the frame that can be used instead of the frame number when another frame branches to the frame. Using frame names for branch targets is preferable to using frame numbers because a frame's number can change as frames are added or removed.

QUIZ

specifies that information about the user's responses to the question in the current frame is recorded in the SAS data set specified in the QUIZ= option.

QUIZ=*<libref.>response-data-set*

specifies the name of a SAS data set that is used to record information about the user's responses to questions in the frames of the CBT entry. If you omit the *libref* portion of the data set name, the data set is created in the default WORK library.

Note: Use the QUIZ= option in the first frame for which you wish to collect response data, and use the QUIZ option in subsequent frames. △

The tracking data set contains the following variables:

LIBREF	is the libref that contains the catalog where the CBT entry resides.
CATNAME	is the name of the catalog that contains the CBT entry.
OBJNAME	is the name of the CBT entry (or XTESTAFX, if you are testing the entry with the TESTAF command).
FRAME	is the frame number for which response data was recorded.
MATRICES	is the number of attempts that the user is allotted to answer the question.
TRIES	is the number of attempts that the user actually used. *Note:* The number of attempts is not incremented if the frame does not specify a correct answer for the question. △
SCORE	is a number that represents the user's success in answering the question in the frame, as follows:

-1	indicates that the user exhausted all allotted attempts and failed to answer the question correctly.
0	indicates that the user answered the question incorrectly and did not use all the allotted attempts before requesting the correct response.
1	indicates that the user gave the correct response.
2	indicates that the user skipped the frame.

SOUND=*freq-1 duration-1 <... freq-n duration-n>* | *note-1 duration-1*
<... note-n duration-n>
MUSIC=*freq-1 duration-1 <... freq-n duration-n>* | *note-1 duration-1*
<... note-n duration-n>

specifies one or more tones or notes that are played when the frame is displayed, provided the user's display device supports sounds.

You can use either of the following formats to specify the sounds to play:

- □ frequency-duration pairs, where *freq* is the frequency of the tone in cycles per second and *duration* is the duration of the tone in units of 1/100ths of a second.
- □ note-duration pairs, where *note* is a note specification and *duration* is the duration of the note in units of 1/100ths of a second. A note specification consists of the note name from the musical scale (A, B, C, D, E, F, or G) and an octave designation (0-7, corresponding to the octaves on a piano keyboard, starting at the bass end). You can also add a **#** to raise the note by a half tone or a lowercase **b** to lower the note by a half tone. For example, **E6b** specifies an E flat in octave 6.

For a rest (silence), specify either **0** for the frequency or **Z** for the note name.

Note: Users can use the SOUND command in the application window to turn sounds on or off. △

WRONG=<*libref.catalog-name.*>*entry-name.entry-type*
specifies an entry that is displayed when the user fails to give the correct response in the allotted number of attempts.

Feedback Indicator Syntax

The general form of a feedback indicator is

#<*n*<C>> <*branch*> <*options*>

where *options* can be one or more of the following:

FRAME=*frame-number* | *frame-name*

HELP=<*libref.catalog-name.*>*entry-name.entry-type*

MENU=*value*

SELECT=(*left-col, right-col, top-row, bottom-row*)

SOUND | MUSIC=*freq-1 duration-1* <... *freq-n duration-n*> | *note-1 duration-1* <... *note-n duration-n*>

Use the slash character (/) to continue a feedback indicator across multiple lines of the display. For example, the following lines comprise a single feedback indicator:

```
#1 >< fruit.cbt frame=app /
   select=(10,14,3,3) /
   menu=apple
```

If you do not use the *branch* option in the feedback indicator, you can follow the feedback indicator with one or more lines of text. The lines of text that follow the feedback indicator are displayed when users select the corresponding response field or selection box. If the indicator designates a correct response, the first line of text after the indicator is considered the congratulatory message, and the remaining lines are considered the explanatory message. Both are displayed when a user enters a correct response, but only the explanatory lines are displayed if the user fails to give the correct response in the allotted number of attempts or when the user asks to see the correct response without giving the correct response.

Feedback Indicator Options

You can use the following options in feedback indicators:

n
specifies the sequence number of the feedback item. This value must appear in column 2 of the feedback indicator line.
If the frame includes a fill-in-the-blank question, then use the value 1 to provide feedback on user responses to the question. If the frame includes a multiple-choice question, the value of *n* should correspond to the order of the choice field (1 for the first choice, 2 for the second choice, and so on). For feedback indicators that are not associated with response fields (for example, when the SELECT= or MENU= options are used), the value of *n* is not significant.

C
designates a correct response. For multiple-choice questions, you can designate more than one correct response.

branch
: specifies that the corresponding feedback response branches to another entry rather than displaying feedback text. Use one of the following forms for the *branch* specification:

  > *<libref.catalog-name.>entry-name.entry-type*
  : branches to the specified entry and stores the current frame number as the CBT checkpoint. The current frame is displayed the next time the CBT entry is opened.

  >> *<libref.catalog-name.>entry-name.entry-type*
  : branches to the specified entry but does not store a CBT checkpoint. The first frame is displayed the next time the CBT entry is opened.

  >< *<libref.catalog-name.>entry-name.entry-type*
  : branches to the specified entry and returns to the branching frame when the target entry is closed.

FRAME=*frame-number* | *frame-name*
: specifies the frame number or frame name to display when the target of the *branch* option or the entry specified in the HELP= option is a CBT entry.

HELP=*<libref.catalog-name.>entry-name.entry-type*
: specifies an entry to open when a user issues the HELP command while the cursor is positioned on the corresponding response field or selection box.

  Note: If you do not specify the HELP= option, the HELP command opens the entry specified in the Help general attribute for the CBT entry. △

MENU=*value*
: specifies a value that users can enter on the application window's command line to select the corresponding feedback response.

SELECT=(*left-col*, *right-col*, *top-row*, *bottom-row*)
: specifies the coordinates of a rectangular region that comprises the selection box for the feedback item. The selection box is highlighted when a user moves the cursor into that region of the display, If a user presses the ENTER key or clicks the mouse while the cursor is within the selection box, the corresponding feedback item is selected. Users can press the TAB key to move between the selection boxes in the current frame.

  Selection boxes should be separated by at least one space.

SOUND=*freq-1 duration-1 <... freq-n duration-n>* | *note-1 duration-1 <... note-n duration-n>*
MUSIC=*freq-1 duration-1 <... freq-n duration-n>* | *note-1 duration-1 <... note-n duration-n>*
: specifies tones or notes that play when the feedback item is selected, provided the user's display device supports sound. Refer to the description of the SOUND= option for frame indicators in "Frame Indicator Options" on page 25 for details about the argument values.

CBT Entry General Attributes

CBT entries also store attributes for the application window in which the entries are displayed to users. See "General Attributes for Application Windows" on page 50 for details about the general attributes you can specify for the application window.

CBT Entry Child Attribute

CBT entries can store a child attribute that specifies the name of another SAS/AF entry that opens if users press ENTER from the last frame of the CBT entry. You specify the child attribute in the ATTR window for the CBT entry. Use the ATTR command in the BUILD procedure's DISPLAY window to open the ATTR window.

Note: The child entry is opened only when a user presses the ENTER key while the final frame of the CBT entry is displayed. Use the Parent general attribute to specify an entry to open when users end some other CBT entry. △

The child attribute consists of the four-level name of the entry to open. If the target entry is in the same catalog as the CBT entry, you only need to specify the name and type of the target entry. If the target entry is stored in a different catalog or in a catalog in a different library, then you must also specify the libref and catalog for the target entry.

CLASS Entries

CLASS entries (also referred to simply as *classes*) store the definitions of components that can be used to build FRAME entries. You use the BUILD procedure's Class Editor window to edit the component definitions in CLASS entries.

Note: Changing the properties of a class changes the properties of all instances of the class and of any subclasses that are derived from the class. △

The class definition in a CLASS entry consists of the following elements:

Description
: is a description for the CLASS entry that is also used as the name for the class when it appears in the Components window for use in building FRAME entries.

Parent Class
: specifies the four-level name of the CLASS entry from which the current class inherits its attributes, methods, events, event handlers, and interfaces. Once you specify the parent class for a new class, you cannot change it.

Meta Class
: specifies the optional four-level name of a CLASS entry of which the current class is an instance. The metaclass enables you to collect information about and modify the behavior of the current class at run time. The metaclass also enables the current class to obtain information about parent classes and child classes.

  By default, all classes are instances of the Class metaclass. See the description of the Class class in *SAS/AF Software: Class Dictionary* for more information about the methods that the default metaclass provides.

Class Properties
: define the appearance and behavior of the class. In the SAS Component Object Model (SCOM), classes have the following properties:

Attributes	define characteristics of the component, such as its name, description, color, label, or size. Each attribute specification consists of a list of metadata that includes the attribute name, value, type, scope, description, and other items that enable functionality.
Methods	define the operations that can be executed by any component you create from the class. Each method specification consists of

	a list of metadata that includes the method name, signature, description, and the name and label of the entry that contains the method implementation. A method's signature is comprised of the method's arguments and their types and order; it uniquely identifies the method to the SCL compiler.
	Note: The code that implements the method is not stored in the CLASS entry itself, but rather in an entry specified in the metadata. The implementation typically consists of a labeled CLASS, USECLASS, or METHOD section in an SCL entry. △
Events	alert applications when there is a change of state, such as when the user clicks the mouse button on a component that was created from the class. Each event specification consists of a list of metadata that includes the event name, description, and items that determine whether the event is enabled and how it is sent.
Event handlers	specify which methods are executed after events occur. Each event handler specification consists of an list of metadata that includes the name of the event that is handled, the name of the object that generates the event, the name of the method to execute in response to the event, and a description.
Interfaces	enable components that you create from the class to indirectly call methods in another component. Each interface specification includes the name of the INTRFACE entry that contains the interface definition. Refer to "INTRFACE Entries" on page 32 for more information.

Refer to *SAS/AF Software: Class Dictionary* for details about the attributes, methods, events, event handlers, and interfaces of the classes that are provided with SAS/AF software.

You can use RESOURCE entries to collect individual classes into libraries. Doing this simplifies the maintenance and deployment of the classes. See "RESOURCE Entries" on page 47 for more information.

For an introduction to using classes and creating your own CLASS entries, refer to *SAS Guide to Applications Development*.

FRAME Entries

FRAME entries store Frame objects plus all the visual objects (or *controls*) and nonvisual objects (or *models*) that comprise a FRAME application. You use the BUILD procedure's DISPLAY window to create and edit FRAME entries.

Frame objects are instances of the Frame class. The Frame class is the foundation of SAS/AF applications that have graphical user interfaces. The Frame class provides windowing capabilities and serves as a container to which you add visual controls and nonvisual components to create a user interface.

The Frame class (and any subclass of it that you create) provides the properties of the window in which your applications run. You specify the frame properties in the Properties window for the FRAME entry. Use the PW command in the DISPLAY window to open the Properties window. For more information about the properties of the Frame class, refer to *SAS/AF Software: Class Dictionary*.

You use the associated Components window to select objects to place in the frame. The FRAME entry can include multiple instances of the same class, and each instance has its own properties. You use the Properties window to edit the properties of the objects that you add to the frame. For more information about the properties of the classes of objects that you can add to FRAME entries, refer to *SAS/AF Software: Class Dictionary*.

Note the distinction between editing the properties of a class in the Class Editor window and editing the properties of an object in the Properties window for a FRAME entry: Editing the properties of a class changes all instances of the class, whereas editing the properties of an object in the FRAME entry changes only the particular instance of the object.

Whenever you create a new FRAME entry, a RESOURCE entry is associated with the FRAME entry. By default, the standard RESOURCE entry that is provided with SAS/AF software (SASHELP.FSP.AFCOMPONENTS.RESOURCE) is used. You can use the RESOURCE= option with the PROC BUILD statement or BUILD command to specify a different initial resource. Once the FRAME entry is created, the associated resource cannot be changed. The specified RESOURCE entry must be available any time you open the FRAME entry in the BUILD environment or execute the entry in the application window.

If the associated resource has an active Frame class, then the new Frame object in the FRAME entry is an instance of that class. If the associated resource has no active Frame class, the default Frame class (SASHELP.FSP.FRAME.CLASS) is used.

For an introduction to building applications with FRAME entries, refer to *SAS Guide to Applications Development*.

HELP Entries

HELP entries store a single frame of text that can be displayed to provide instructions or other assistance to users of your application. You use the BUILD procedure's DISPLAY window to design the displays for HELP entries. The display can use any of the text color features and highlighting features that the SAS text editor supports.

HELP entries can also be used as selection lists for PROGRAM entries. See the discussion of the List attribute for PROGRAM entries in "PROGRAM Entry Field Attributes" on page 37 for details.

HELP entries also store attributes for the application window in which the entries are displayed to users. See "General Attributes for Application Windows" on page 50 for details about the attributes you can specify for the application window.

INTRFACE Entries

INTRFACE entries (also referred to simply as *interfaces*) store abstract method definitions, which define shared methods that FRAME components can use to communicate with each other. You use the BUILD procedure's Interface Editor window to create, edit, or remove method definitions in INTRFACE entries.

Method definitions in INTRFACE entries consist of the method name and, optionally, the method signature. The method signature specifies the name, order, type, and description of the method's arguments. The code that implements the methods is not stored in the INTRFACE entry. Rather, a class that uses the methods defined in the interface to communicate with another class indirectly calls the corresponding methods in the other class.

For an introduction to using interfaces to implement model/view communications, refer to *SAS Guide to Applications Development*.

LIST Entries

LIST entries store lists of values that are used in conjunction with PROGRAM entries to validate field values and to provide selection lists.

You can specify the following attributes for the list. You use the BUILD procedure's LISTATTR window to set list attributes.

Type
: specifies one of the following types for the values in the list:

CHAR	indicates that the list contains character values. (This is the default.)
NUM	indicates that the list contains numeric values.

The list type should match the type of the PROGRAM entry field that the list is used to validate.

Note: You cannot change the list type once the list is saved. △

Length
: specifies the length of items in the list. Valid values are 1 to 80.

Note: You cannot change the item length once the list is saved. △

Fileref
: specifies a fileref that is associated with a file that is used to populate the list. The fileref must have previously been defined in the SAS session. Each value that is read from the file is appended to the list. The file can contain more than one value per record or line as long as the values are separated by one or more spaces. Values that are longer than the specified item length are truncated.

Note: The fileref is not stored in the LIST entry. The Fileref attribute is blank each time the LISTATTR window opens. △

Pad
: specifies which pad character to use for fields in the LISTVALUES window. The default is the underscore (_) character.

Format
: specifies which format to use for values in the LISTVALUES window and when values from the list are displayed in selection lists.

Just
: specifies how values are aligned in the fields in the LISTVALUES window. The choices are LEFT (default), RIGHT, CENTER, and NONE.

Informat
: specifies the informat that must be used when values are entered in the LISTVALUES window.

Options
: specify one or more of the following characteristics of the list:

SORT
: specifies that the values in the list are sorted in ascending order when the entry is saved. This option is selected by default. Deselect the SORT option if you want to store list values in the order in which they are entered.

CAPS
: specifies that character values in the list are converted to uppercase. This option is selected by default. Deselect the CAPS option if you want to store mixed-case values in the list.

CASE-INSENSITIVE
specifies that the case of character values is ignored when values from the list are used to validate field values. For example, if this option is selected, then the value **RED** in the list matches the value **red** in the PROGRAM entry field that is being validated. If this option is not selected, the values do not match.

Error msg
specifies the message that is displayed when no value in the list matches the value in the PROGRAM entry field that is being validated.

You can use the special indicator **%s** to include the field value in the message, as in the following example:

```
The value %s is not valid for this field.
```

You use the BUILD procedure's LISTVALUES window to enter or edit values in the list. The LISTVALUES window opens automatically when you close the LISTATTR window.

MENU Entries

Menu entries store menu definitions that consist of

- a display that provides instructions to users and accepts user options
- general attributes that control the appearance and behavior of the window in which the MENU entry executes
- selection attributes that define which other SAS/AF entries are opened in response to the options that users specify.

The following sections describe each of these components of a MENU entry.

MENU Entry Displays

Each MENU entry stores a single frame of text that can provide instructions to users about the options that are available in the menu. You use the BUILD procedure's DISPLAY window to design the displays for MENU entries. The text can use any of the color and highlighting features that the SAS text editor supports.

Users select menu options by entering designated option values on the application window's command line. Menus can be linked so that users can access choices on submenus from the command line of the main menu. Refer to the description of the Menu-Link attribute in "MENU Entry Selection Attributes" on page 34 for details.

MENU Entry General Attributes

MENU entries also store general attributes for the application window in which the entries are displayed to users. See "General Attributes for Application Windows" on page 50 for details about the attributes you can specify for the application window.

MENU Entry Selection Attributes

MENU entries support the following attributes for each menu option. You specify these attributes in the ATTR window for the MENU entry. Use the ATTR command in the BUILD procedure's DISPLAY window to open the ATTR window.

Option
: specifies the selection value that users issue in the application window's command line to invoke the option. The selection value

  - □ can be from one to eight characters long
  - □ can consist of a combination of letters, numbers, and underscores
  - □ must not be a command that is valid in the application window. See Chapter 5, "AF Window Commands," on page 65 for a list of commands that the AF window provides. Remember that SAS windowing environment global commands are also valid in the AF window in which MENU entries are displayed. Refer to base SAS documentation for more information on windows environment global commands.

Name
: specifies the name of the catalog entry that the option invokes.

Type
: specifies the type of the catalog entry that the option invokes (CBT, FRAME, HELP, MENU, PROGRAM or SCL).

  Note: For CBT entries, you can append a frame number to open the entry at a specified frame. For example, use **`CBT5`** to open frame 5 of the specified CBT entry. △

Libref
Catalog
: specify the library and catalog that contain the target entry. Enter values for these two attributes only if the target entry is stored in a catalog other than the one that contains the MENU entry.

Menu-Link
: specifies that menu choices in the selected submenu are linked to menus at a higher level in the application's hierarchy of menus. If the submenu is linked, you can specify options from the submenu in the higher-level menu without having to display the lower-level menu.

  Note: The Menu-Link attribute is valid only when the target of the option is another MENU entry. △

  If you assign this option to any menu choices, you must issue the MLINK command (or use the MLINK statement with the BUILD procedure) to generate the linkages for the selected options. Any time you change or add the Menu-Link option for menu choices, you must repeat the linking procedure to reestablish the internal menu linkages.

  By default, only one level of menus is linked. To link all levels of menus, use the MLOPTS LEVEL=_MAX_ option with the MLINK command, or use the LEVELS=_MAX_ option with the MLINK statement.

PROGRAM Entries

PROGRAM entries store SAS/AF applications that consist of

- □ a display that provides instructions and data-entry fields and which accepts user input
- □ general attributes that control the appearance and behavior of the window in which the PROGRAM entry executes
- □ field attributes that define the appearance and behavior of fields

□ a SAS Component Language (SCL) program that controls the application.

The following sections describe each of these components of a PROGRAM entry.

PROGRAM Entry Displays

You use the BUILD procedure's DISPLAY window to design the displays for PROGRAM entries. The displays can use any of the text color features and highlighting features that the SAS text editor supports. In addition to static text, the display can include fields in which users can enter values. The PROGRAM entry's SCL program can also manipulate field values. Refer to "Fields" on page 36 for more information about defining fields in the display.

If your display includes a large amount of text or many fields, you can divide it into units called *frames*. Users can issue FORWARD and BACKWARD commands in the application window to scroll between the frames of the display. To divide the display into frames, enter divider lines consisting of dash (–) characters across the full width of the DISPLAY window. You can also use the FILL command to create divider lines.

You can designate a portion of the display that remains visible and does not scroll. Enter three caret (^) or NOT (¬) characters in the first three columns of a line in the first frame of the display to delineate the nonscrolling region. Any text and fields above the line that contains the ^^^ or ¬¬¬ appear in every frame of the display; FORWARD and BACKWARD commands scroll only the region below the nonscrolling area.

You can also create extended tables in the display. In an extended table, you define one row of fields and use SAS Component Language to dynamically display multiple rows based on the one you define. Refer to "Extended Tables" on page 37 for details.

Fields

Fields in PROGRAM entries accept user input and display information or program output. You designate fields in the display with an initial ampersand (&), followed by an optional name up to eight characters in length. Use underscore characters (_) to pad the field to the length required to hold the largest field value. The field length is determined by the number of columns from the ampersand through the last underscore. A field can be as short as a single & or as long as the width of a display line.

If you omit the field name (or if you create one-character fields that consist of only an ampersand), the field is given the default name FIELD*n*, where *n* is the order of the field on the DISPLAY window, counting from the upper-left corner and descending from left to right.

Each field has a set of attributes that determine its appearance and behavior. Refer to "PROGRAM Entry Field Attributes" on page 37 for information about the field attributes. When you refer to a field in the entry's SCL program, you use the name specified in the field's Alias attribute rather than the field name. By default, the field alias is the same as the field name, but you can change the alias in the entry's ATTR window to give the field a more meaningful name.

Choice Groups

You can join one or more fields into a choice group. Fields that are assigned to choice groups are referred to as stations. Only one station of a choice group can be active at a time. Pressing the ENTER key or clicking the mouse button while the cursor is on one of the fields in the choice group selects the active station. The choice group name can be used as a variable in the entry's SCL program. It returns the value of the selected station.

SAS Component Language provides functions for manipulating choice groups. Refer to *SAS Component Language: Reference* for more information.

Selection Lists

If you specify the List attribute for a field, users can select values for the field from a selection list. The selection list of valid field values is displayed when a user enters a designated prompt character in the first column of the field. The default prompt character is the question mark (?), but you can use the PROGRAM entry's Prompt character general attribute to specify a different prompt character for your application.

Note: If the List attribute specifies a range of values rather than a list of valid values (or if it specifies an entry that contains a list of valid values), then entering the prompt character does not display a selection list. Rather, it displays a dialog window that explains the range of valid values. △

SAS Component Language provides a variety of functions for displaying selection lists. Refer to *SAS Component Language: Reference* for more information.

Extended Tables

You can use PROGRAM entries to display tables of values called *extended tables*. The extended table can be static, with a fixed number of rows, or dynamic, in which rows can be added or deleted. The values in the rows of the extended table can come from a SAS data set, from an array in the SCL program, or from an external file. In addition to displaying information, extended tables can be used as custom selection lists in your applications.

To create an extended table, you must

- □ select the EXTENDED TABLE general attribute in the PROGRAM entry's GATTR window.
- □ create the fields that define a row of the extended table in the PROGRAM entry's DISPLAY window.

  Note: If you define a nonscrolling region to serve as a table heading, the fields for the extended table must appear below the ^^^ or ─┬┬┐ characters that delineate the nonscrolling region. △
- □ add the SCL code to support the extended table in the PROGRAM entry's SOURCE window. Refer to *SAS Component Language: Reference* for more information on the SCL elements that support extended tables.

PROGRAM Entry General Attributes

PROGRAM entries also store attributes for the application window in which the entries are displayed to users. See "General Attributes for Application Windows" on page 50 for details about the attributes you can specify for the application window.

PROGRAM Entry Field Attributes

Each field that you define in the PROGRAM entry's display has the following attributes. You specify these field attributes in the ATTR window for the PROGRAM entry. Use the ATTR command in the BUILD procedure's DISPLAY window to open the ATTR window.

Alias	specifies the name by which you refer to the field in the entry's SCL program. Each field in a PROGRAM entry must have a unique alias. By default, the alias is the same as the field name.
Choice Group	specifies a choice group to which the field belongs. The choice group name cannot be the same as an existing alias. The fields in a choice

group are called stations, and the choice group variable takes the value of the active station. Only one station in a choice group can be active at a time. Refer to “Choice Groups” on page 36 for more information on creating choice groups.

Pad

specifies the character that is used to fill blank fields in the application window. By default, fields are padded with underscore (_) characters. You can use the Pad attribute to change the pad character for an individual field. To change the default pad character for all fields, use the PADCHAR= option with the PROC BUILD statement, or issue the PDCHAR command in the DISPLAY window.

Type

specifies the type of validation that is performed to verify the values that users enter in the field. By default, fields are assigned one of the following types:

- ACTION, if the field is one character in length
- CHAR, if the field length is greater than one character.

Refer to “Field Types” on page 42 for details about these and other field types that you can specify.

When a user enters a value in a field, the AF task evaluates whether the value meets the conditions of the specified type. If the value is determined to be invalid,

- an error message is displayed on the application window’s message line
- the cursor is positioned on the field
- the field is highlighted, using the color and highlighting attribute specified in the Error color and Error attr attributes.

Users cannot use the END command to exit from the application window until the field contains a valid value. They must use the CANCEL command to exit without correcting the invalid value.

Protect

controls whether field values can be changed and whether the TAB key moves the cursor to the field.

YES	specifies that users cannot change values in the field and that the TAB key does not move the cursor to the field. However, the entry’s SCL program can still change the field value.
NO	specifies that users can change values in the field and that the TAB key moves the cursor to the field. This is the default behavior.
INITIAL	specifies that users cannot change values in the field, but that the TAB key moves the cursor to the field. If a user attempts to change the field value, the field reverts to the value specified in the Initial attribute when the user presses ENTER. This attribute is typically used for fields that participate in choice groups or that are assigned a push button Type attribute.

CAUTION:

If you set the Protect attribute to YES, do not select the REQUIRED option. Making a protected field required means that users cannot use the END command to exit from the application because attempting to end the application while a required field is blank results in an error. △

Format	specifies a format that controls how values that are entered in the field are displayed. You can specify any standard SAS format or a user-defined format. *Note:* If you assign a format to a field, you should also assign a compatible informat. △
Just	specifies how values that are entered in the field are aligned after the user presses ENTER.

LEFT	aligns values with the left margin of the field. This is the default behavior.
RIGHT	aligns values with the right margin of the field.
CENTER	centers values in the field width.
NONE	displays character values as they are entered. Numeric values are aligned with the right margin of the field.

Informat	specifies an informat that controls how values that are entered into the field are interpreted. You can specify any standard SAS informat or a user-defined informat. *Note:* If you assign an informat to a field, you should also assign a compatible format. △
Error color	specifies a color that is applied to the field when a user enters an invalid value. The following standard SAS color values are supported:

BLACK	CYAN	MAGENTA	RED
BLUE	GRAY	ORANGE	WHITE
BROWN	GREEN	PINK	YELLOW

Error attr	specifies a highlighting attribute that is applied to the field when a user enters an invalid value. The following attribute values are allowed, although some attributes may not be supported on some display devices: REVERSE, HIGHLIGHT, UNDERLINE, BLINKING, or NONE (the default).
Help	specifies the name and type of an entry that provides help information about the field. The specified entry is displayed when a user issues the HELP command while the cursor is positioned on the field.

Valid help entry types are CBT, HELP, MENU, and PROGRAM. Because only two-level names can be entered, the specified entry must reside in the same catalog as the current PROGRAM entry. For CBT entries, you can specify a frame number by appending the number to the entry type. For example, specify **`INFO`** and **`CBT5`** to open the entry INFO.CBT with frame 5 displayed.

If the Help attribute is not specified for the field, the entry specified in the Help general attribute for the PROGRAM entry is displayed when a user requests help for the field.

List

determines the values that are valid for a field, provided the Type attribute permits a List attribute. Depending on the Type attribute, the List attribute can contain one or more of the following:

- a list of values separated by spaces.
- a range of values.

  Use the less than (**<**) character to indicate that the List attribute value is a range. If you specify one value after the **<**, the range consists of all values greater than or equal to the specified value. For example, **`< 100`** indicates all values greater than or equal to 100. If you specify a pair of values following the **<**, the range includes all values between and including the specified values. For example, the following range specification matches all values between 10 and 100:

  ```
  < 10 100
  ```

  The following range specification matches all uppercase values between A and Z:

  ```
  < A Z
  ```

- the name of a LIST entry that contains a list of valid values.

  Use the form **=***libref.catalog-name.entry-name* to specify the name of the LIST entry. Note that an equal sign (=) is added as a prefix. You can omit the *libref.catalog-name* portion if the LIST entry is in the same catalog as the current PROGRAM entry. Refer to "LIST Entries" on page 33 for more information about LIST entries.

- the name of a data set or the name of one or more fields that contain the names of SAS data sets.

  For types such as ONEVAR, VARLIST, and VARSTMT that verify the names of variables in a data set, the value of the List attribute determines which data set is searched for the variables that are specified in the field.

  Use the form ******libref.data-set-name* to specify the data set name. Note that an asterisk (*) is added as a prefix.

  If you specify one or more field names that contain data set names, the corresponding fields should be defined as INPUT type to ensure that the data sets named in the fields exist. By default, the field values are valid only if specified variables exist in all the data sets named in the variable. To specify that the field values are valid if the specified variables exist in one or more of the data sets, add an at (@) character as the first character in the List attribute value.

You can also use the List attribute to specify a selection list for the field. In this case, the List attribute has the following form:

\\<*prompt*> <*num-sel*> =*entry-name*<C | F | L> | @*link-name*\\

where

prompt

specifies the prompt character that displays the selection list when a user enters it as the first character in the field. If you omit the *prompt* argument, the default is the question mark (?).

num-sel
: specifies how many items users can select from the list. If you omit the *num-sel* argument, the default is 1.

=*entry-name* <C | F | L>
: specifies a LIST, HELP, or CBT entry that provides the items for the selection list.

  If you specify a HELP entry, each line of text in the entry's display becomes an item in the selection list. You can add one of the following options to specify which portion of the selected line is returned:

  C
  : indicates that the word at the cursor position in the selected line is returned. This option is valid only when one selection is allowed.

  F
  : indicates that the first word on the selected line is returned. This is the default behavior.

  L
  : indicates that the entire selected line is returned.

@link-name
: specifies the name of a linked field that determines which type of format or informat information is displayed. This form of the List attribute is valid only for fields of type FMT or INFMT. If the first character in the linked field is **C** or **$**, then help on character formats or informats is displayed. Otherwise, help on numeric formats or informats is displayed. Only the names of formats or informats of the corresponding type can be entered in the field.

Initial
: specifies a character string (up to 56 characters) or a numeric value that is displayed in the field when the PROGRAM entry is initially displayed to a user in the application window. If the Initial attribute is blank, the field is represented with the specified pad character or with a default pad character.

Replace
: specifies a character string (up to 56 characters) that is used as a replacement string when values are substituted in SUBMIT blocks in the PROGRAM entry's SCL program. Refer to the description of the REPLACE statement in *SAS Component Language: Reference* for more information about replacement strings.

Options
: control the following characteristics of the field:

  CAPS
  : specifies that characters entered into the field are converted to uppercase when the user presses ENTER. This attribute is selected by default. Deselect this attribute if you want users to be able to enter mixed-case values in the field.

  CURSOR
  : specifies that the cursor is positioned on the field when the application window opens. By default,

	the CURSOR attribute is selected for the first field created in the DISPLAY window and is deselected for the remaining fields. Only one field should have the CURSOR attribute selected. If more than one field has the CURSOR attribute selected, the cursor is initially positioned on the first field that has the attribute selected.
REQUIRED	specifies that a user must enter a valid value in the field before the application window can be closed with the END command. ***CAUTION:*** **If you select the REQUIRED option, do not set the Protect attribute to YES.** Protecting a required field means that users cannot use the END command to exit from the application because attempting to end the application while a required field is blank results in an error. △
AUTOSKIP	specifies that the cursor moves automatically to the next unprotected field when user input fills the current field's last position. By default, the AUTOSKIP attribute is selected for all fields. Deselect the AUTOSKIP attribute if you want the cursor to remain on the current field when a user enters values that fill the field.
NOPROMPT	specifies that the prompt character is ignored for the field. By default, when a user enters a designated prompt character in the first position of the field, the AF task displays either a selection list of valid values or information about the range of valid values for the field. (The behavior depends on the value of the List attribute.) Select the NOPROMPT option to disable this behavior and treat the prompt character like a regular text character. The prompt character for the entry is specified in the entry's Prompt character general attribute. The default prompt character is the question mark (?).
NON-DISPLAY	specifies that values that users enter in the field are not displayed in the application window. Users can still tab to a nondisplayed field (provided the field is not protected), and field values are still validated. This attribute is useful when the field is used for entering passwords or other values that should be kept hidden.

Field Types

The Type field in PROGRAM entries can take one of the following attribute values:

ACTION
: specifies that the value that a user enters in the field is converted to a predefined character or character string. For ACTION fields for which no List attribute is

specified, a value entered in the field is automatically converted to uppercase **X**. If a List attribute is specified, then any value entered in the field is automatically converted to the value that is specified as the field's List attribute. This enables you to define the value for the field as a single character (for example, *) or as a word (for example, **YES**).

ATTR
: verifies that the field value is the name of a text highlighting attribute. Valid values are REVERSE, HIGHLIGHT, UNDERLINE, BLINKING, or NONE, or the corresponding one-character abbreviations (R, H, U, B, or N).

CHAR
: verifies that the field value is standard SAS character data.

CHARLST
: verifies that the field contains a list of standard SAS character data values. Values in the list are separated by spaces.

COLOR
: verifies that the field value is one of the standard SAS color names:

BLACK	CYAN	MAGENTA	RED
BLUE	GRAY	ORANGE	WHITE
BROWN	GREEN	PINK	YELLOW

DSNAME
: verifies that the field value is a valid one- or two-level SAS data set name.

Note: The DSNAME type tests only whether the specified name is valid, not whether the specified data set exists. Use the INPUT type to verify that the specified data set exists, or use the OUTPUT type to verify that the specified data set does not exist. △

FILENAME
: verifies that the field value is a fileref that has been previously defined in the current SAS session.

FIXED
: verifies that the field value is an integer numeric value.

FIXEDLST
: verifies that the field contains a list of integer numeric values. Values in the list are separated by spaces.

FMT
FMTC
FMTN
: verify that the field value is a valid format name. Use FMTC to verify that the field value is the name of a character format, or use FMTN to verify that it is the name of a numeric format. Use FMT to verify that the field value is the name of either a character format or a numeric format.

Note: Do not specify a value for the List attribute when you specify FMT, FMTC, or FMTN for the Type attribute. The field value is validated against the list of all standard SAS formats and user-defined formats. △

INFMT
INFMTC
INFMTN
: verify that the field value is a valid informat name. Use INFMTC to verify that the field value is the name of a character informat, or use INFMTN to verify that

it is the name of a numeric informat. Use INFMT to verify that the field value is the name of either a character informat or a numeric informat.

Note: Do not specify a value for the List attribute when you specify INFMT, INFMTC, or INFMTN for the Type attribute. The field value is validated against the list of all standard SAS informats and user-defined informats. △

INPUT
: verifies that the field value is the name of an existing SAS data set.

INPUTALL
: verifies that the field value is either a list of names of existing SAS data sets or the special SAS designation _ALL_.

Note: The default pad character for this type of field should be a character other than the default underscore. It should also be a character that is not likely to be used in a data set name. △

LIBNAME
: verifies that the field value is a libref that has previously been defined in the current SAS session.

NAME
: verifies that the field value is a valid SAS name.

NUM
: verifies that the field value is a standard SAS numeric value.

Note: Use the FIXED type if you want to restrict the field to integer values. △

NUMLST
: verifies that the field contains a list of standard SAS numeric values. Values in the list are separated by spaces.

Note: Use the FIXEDLST type if you want to restrict the field to a list of integer values. △

ONEVAR
ONEVARC
ONEVARN
: verify that the field value is the name of one variable from a SAS data set. Use ONEVARC to verify that the field value is the name of a character variable, or use ONEVARN to verify that it is the name of a numeric variable. Use ONEVAR to verify that the field name is the name of either a character variable or a numeric variable.

The List attribute specifies which data set to search for the variable. The List attribute can specify either the name of the data set or the names of fields that contain the name of the data set. If the List attribute is not specified, the PROGRAM entry's Lookup data set general attribute is used. The Lookup data set general attribute contains the name of a field that contains the name of the data set to search.

OUTPUT
: verifies that the field value is a valid data set name and that the specified data set does not currently exist.

Note: The OUTPUT field type is typically used for names of data sets that are created for application output. This type of validation ensures that the output data set does not overwrite an existing data set. △

PUSHBTNC
PUSHBTNN
: display the field as a push button that users can click (or move the cursor to and press ENTER) to cause an action to occur. Use the Initial attribute to specify the

value that appears as the button label. Use PUSHBTNC for fields that return character values, or use PUSHBTNN for fields that return numeric values.

SHORT
: verifies that the field value is an integer number in the range of -32767 to 32767.

VARLIST
VARLISTC
VARLISTN
: verify that the field contains a list of variable names that appear in a data set. Use VARLISTC to verify that the field contains the names of character variables, or use VARLISTN to verify that it contains the names of numeric variables. Use VARLIST to verify that the field contains the names of either character variables or numeric variables. These types ignore any other characters in the field, such as arithmetic operators, and verify only the field's variable names.

  The List attribute specifies which data set to search for the variable. The List attribute can specify either the name of the data set or the names of fields that contain the name of the data set. If the List attribute is not specified, the PROGRAM entry's Lookup data set general attribute is used. The Lookup data set general attribute contains the name of a field that contains the name of the data set to search.

VARSTMT
VARSTMTC
VARSTMTN
: verify that the field contains a list of variable names that appear in a data set. Use VARSTMTC to verify that the field contains the names of character variables, or use VARSTMTN to verify that it contains the names of numeric variables. Use VARSTMT to verify that the field contains the names of either character or numeric variables. These types allows only variable names to be entered into the field. Use VARLIST (or VARLISTC or VARLISTN) to allow the field to contain other characters in addition to the variable names.

  The List attribute specifies the data set to search for the variable. The List attribute can specify either the name of the data set or the names of fields that contain the name of the data set. If the List attribute is not specified, the PROGRAM entry's Lookup data set general attribute is used. The Lookup data set general attribute contains the name of a field that contains the name of the data set to search.

PROGRAM Entry SCL Programs

PROGRAM entries can store a SAS Component Language (SCL) program that can manipulate field values and control the behavior of the entry. You use the BUILD procedure's SOURCE window to edit the PROGRAM entry's SCL program. You can issue the SOURCE command in the DISPLAY window to open the SOURCE window for the entry. Refer to *SAS Component Language: Reference* for details about the statements and functions that you can use in SCL programs.

Before the SCL code in an PROGRAM entry can be executed, it must be compiled. To compile the program, issue the COMPILE command in either the SOURCE window or the DISPLAY window for the entry. You can also use the COMPILE statement with the PROC BUILD statement to compile the contents of existing PROGRAM entries.

When the source code in the SCL entry is compiled, the SCL compiler writes any error or warning messages to the SAS log. If the SCL program is compiled successfully, the compiled code is added to the PROGRAM entry along with the source code.

When you invoke a PROGRAM entry, the AF task executes the statements in the program's INIT section. When you modify a field value, statements in the program's

MAIN section are executed. If the program contains labeled sections whose labels match the names of the modified fields, then those sections are also executed before the MAIN section. Statements in the TERM section of the program are executed when you end the PROGRAM entry. Refer to *SAS Component Language: Reference* for more information on SAS Component Language processing.

RANGE Entries

RANGE entries are utility entries that store range definitions. Range definitions are used in conjunction with FRAME entries to control traffic lighting for Text Entry objects and Critical Success Factor objects. You use the BUILD procedure's RANGE window to edit the range definition in a RANGE entry.

A range definition can consist of up to 24 segments. Each segment defines the range of values that match that segment, as well as the color and highlighting attributes that are applied to values that match the segment. For each segment, you can define the following attributes:

Lower value

specifies a numeric minimum value for the range segment. This value can be omitted for the first segment, implying that all values lower than or equal to the Upper value match the first segment.

Note: If a statistic has been defined for the segment, you cannot modify the Lower value attribute. △

If the Lower value of the current segment is the same as the Upper value of the preceding segment, then the segments are contiguous, and the specified Lower value is not inclusive in the current segment. That is, a value must be greater than the Lower value in order to match the current segment. Values that are equal to the Lower value match the preceding segment. If the Lower value of the current segment is not the same as the Upper value of the preceding segment, then the segments are noncontiguous. In this case, values that are equal to the Lower value do match the current segment.

Upper value

specifies a numeric maximum value for the range segment. This value can be omitted for the last segment in the range definition, implying that all values greater than the Lower value (and equal to the Lower value, if the segment is noncontiguous) match the last segment.

Note: If a statistic has been defined for the segment, you cannot modify the Upper value attribute. △

Color

specifies the color for the segment. In the RANGE window, you can select the down arrow control in the **`Color`** field to obtain a list of valid colors, or select the right arrow control to define a custom color. You must specify a color for each segment that has range values. The RANGE window includes a **`Color scale`** bar that shows which colors have been selected for the defined segments.

Attribute

specifies a highlighting attribute for the segment. In the RANGE window, you can select the down arrow control in the **`Attribute`** field to obtain a list of valid display attributes. The default is NONE.

Statistics data set
: specifies a data set and variable to be used in computing the statistics for the lower and upper values of the range segment, as an alternative to specifying range values for the Lower value and Upper value attributes.

  By default, the statistics are computed and stored when the RANGE entry is built. However, you can choose to refresh the statistics when the RANGE entry is used.

Formats
: specifies an informat and a format for the range segment. The informat is used to convert character values to numeric values for comparison purposes. The format is used to display values in Critical Success Factor (CSF) objects.

RESOURCE Entries

RESOURCE entries (also referred to simply as *resources*) store a collection of classes that are used for building FRAME entries. You use the BUILD procedure's Resource Editor window to add classes to or remove classes from RESOURCE entries.

When you add a class to a RESOURCE entry, the class definition is copied from the corresponding CLASS entry into the RESOURCE entry. The RESOURCE entry stores a complete, static copy of its classes. If you change the name, location, description, or any of the properties of a class that is added to a resource, you must synchronize the resource to update the copy of the class in the RESOURCE entry. You use the SYNC command in the Resource Editor window (or the SYNC statement with a PROC BUILD statement) to synchronize a RESOURCE entry.

Note: If you open a class for editing from within the Resource Editor window, the resource is automatically synchronized when you save your changes to the class. △

The BUILD procedure's Components window displays the classes that are available when you are building a FRAME entry. Classes can appear in the Components window even if they are not part of a resource, but you can collect a set of classes into a resource and then add the single resource to the Components window to make all the classes in the resource available for use in the FRAME entry. The Components window displays the RESOURCE entry's description as the name of the resource.

The RESOURCE entry stores the display status of each class. The display status determines whether the class appears in the Components window when the resource is added to that window. Visual classes that can be dropped onto a frame should have their status option set to Display. If you do not want the class to appear in the Components window, you can use the **`Toggle Display Status`** control in the Resource Editor window to turn off the display status. Although a resource should contain all the classes that an application uses, only those components that can be dropped onto a frame should be set to display in the Components window.

You can use resources to organize and maintain class libraries for developing SAS/AF applications. For example, you could use a personal RESOURCE entry to store classes that you develop, and a separate RESOURCE entry to store classes that are developed at your site, in addition to the standard RESOURCE entry that is provided with SAS/AF software (SASHELP.FSP.AFCOMPONENTS.RESOURCE).

Note: Although resources are helpful organizational aids, there are performance drawbacks to using multiple resources. If a FRAME entry has only one resource to load, the initialization stage is generally faster than when multiple resources must be loaded. △

Whenever you create a new FRAME entry, a RESOURCE entry is associated with the FRAME entry. By default, the standard RESOURCE entry provided with SAS/AF software (SASHELP.FSP.AFCOMPONENTS.RESOURCE) is used. You can use the RESOURCE= statement with the PROC BUILD statement or the RESOURCE= option with the BUILD command to specify a different initial resource. Once the FRAME entry is created, the associated resource cannot be changed. The associated RESOURCE entry must be available any time you open the FRAME entry in the BUILD environment or execute the entry in the application window.

If the associated resource has an active Frame class, then the new FRAME entry is an instance of that class. When you add a subclass of the Frame class to the resource, you have the option of making it the active Frame class for the resource. A resource can contain multiple Frame classes, but only one can be active at a time. You can use the **`Active`** control in the Resource Editor window to select the active Frame class. If the associated resource has no active Frame class, the default Frame class (SASHELP.FSP.FRAME.CLASS) is used.

Refer to *SAS Guide to Applications Development* for an introduction to working with RESOURCE entries.

SCL Entries

SCL entries store SAS Component Language (SCL) code, but they do not provide a display. The SCL programs for FRAME entries are stored in associated SCL entries. You can also use SCL entries to store method definitions and SCL programs that perform tasks that do not require any user interaction. You use the BUILD procedure's SOURCE window to edit the SCL code in SCL entries. Refer to *SAS Component Language: Reference* for more information on the SAS Component Language elements that you can use in SCL programs.

Before the SCL code in an SCL entry can be executed, it must be compiled. To compile the program, issue the COMPILE command in the SOURCE window (or in the DISPLAY window if the code is associated with a FRAME entry). You can also use the COMPILE statement with the PROC BUILD statement to compile the contents of existing SCL entries.

When the source code in an SCL entry is compiled, the SCL compiler writes any error or warning messages to the SAS log. If no errors are encountered, a message similar to the following is displayed:

```
Code generated for MYPROG. Code size=1276
```

However, if there are warning messages, you see a message similar to the following:

```
Code generated (with messages) for MYPROG. Code size=1276
```

If there are errors in your program, you see the following message:

```
ERROR: Compile error(s) detected. No code generated.
```

When you save an SCL entry after its program is compiled successfully, the compiled code is saved to the entry along with the source code. At this point, you can execute the SCL entry as described in "Calling SCL Entries from Other SAS/AF Programs" on page 49.

Note: You should always compile an entry before you save it. If you save an SCL entry without compiling it, you cannot execute it. SAS/AF software provides a warning message indicating that the entry has been saved without intermediate code. △

Calling SCL Entries from Other SAS/AF Programs

You can execute SCL code that has been compiled and stored in an SCL entry in the following ways:

- □ by using the AF or AFAPPLICATION commands to execute the SCL entry
- □ by using the CALL DISPLAY routine in an SCL program to execute the SCL entry
- □ by using the CALL METHOD routine in an SCL program to execute individual sections of code in the SCL entry.

The following sections describe the use of the CALL DISPLAY and CALL METHOD routines in SAS Component Language. Refer to Chapter 4, "Executing SAS/AF Applications," on page 53 for information on using the AF and AFAPPLICATION commands.

Using CALL DISPLAY to Execute SCL Entries

When you use the CALL DISPLAY routine to invoke an SCL entry, the AF task executes statements in the following order before returning to the calling program:

1. ENTRY statement
2. INIT section
3. MAIN section
4. TERM section

The program in the SCL entry must have at least one of the special labels INIT, MAIN, or TERM.

You can pass parameters in the CALL DISPLAY statement and receive them via an ENTRY statement in the SCL code.

Refer to *SAS Component Language: Reference* for more information on the CALL DISPLAY routine.

Using CALL METHOD to Execute SCL Routines

You can create modular SCL routines that you can invoke from any SAS/AF application. Each module can have its own parameter list. The parameter list is analogous to an ENTRY statement. You can store several different modules in a single SCL entry.

You identify each module in the SCL source code with the METHOD statement plus an associated label. You invoke each module with the CALL METHOD routine, specifying the entry name and the label, plus any parameters to pass. For example, the following code invokes the module labeled FUNC1 in the entry MYFUNC.SCL in the current search path:

```
call method("MYFUNC", "FUNC1", 10, 30, x);
```

CAUTION:
A program halts if you attempt to invoke a routine that does not exist in the SCL program. For example, if your application executes the METHOD call in the previous example and the label FUNC1 does not exist in MYFUNC.SCL, your program halts. △

Refer to *SAS Component Language: Reference* for more information on the CALL METHOD routine.

General Attributes for Application Windows

In addition to other information, CBT, HELP, MENU, and PROGRAM entries also store attributes that control the appearance and behavior of the application window in which the entries are displayed to users.

You can specify the following general attributes for CBT, HELP, MENU, and PROGRAM entries. You define these general attributes in the GATTR window for the entry. Use the GATTR command in the BUILD procedure's DISPLAY window to open the GATTR window.

- *Window Attributes*

  Name
  : specifies a window name that is displayed in the window's title bar when users execute the entry.

  Start row, col
  : specify the default position of the upper-left corner of the application window on the user's display (or within the AWS window, if application workspaces are used). The default for all entry types is row 1 and column 1. However, the numbers for the default are not displayed in these fields, and the specification for the starting position is ignored in some windowing environments.

  Number of rows, cols
  : specify the default window height (in rows) and width (in columns). No default window size values are displayed in these fields in the GATTR window. The default size depends on your host windowing environment.

  Note: You can issue the SETWSZ command in the DISPLAY window to change the window size while you are building the entry. △

  Banner
  : specifies the appearance of the entry's command line.

COMMAND	provides a command line with the prompt **`Command===>`**.
SELECT	provides a command line with the prompt **`Select Option===>`** (typically used for MENU entries).
NONE	disables the command line and the message line as well. *Note:* Messages are not displayed in windows for which you select the NONE option. In PROGRAM entries, you can use SCL to display messages in a field. △

  Note: If you specify a PMENU entry in the Command menu attribute and the PMENU facility is active, then the specified menu is displayed instead of a command line. △

- *General Attributes*

  Help
  : specifies the name and type of an entry to display when users issue the HELP command while the current entry is executing. If you omit the entry type, the default type is CBT.

  Note: Because you can specify only a one- or two-level name, the entry that you specify must reside in the same catalog as the current entry. △

  Keys
  : specifies the name of a KEYS entry that defines function key settings for the application window. The default is to use the DMKEYS.KEYS function key

entry. You can use the BUILD procedure to create custom function key definitions for your applications.

Note: Because you can specify only a one-level name, the KEYS entry that you specify must reside in the same catalog as the current entry. △

Lookup data set

specifies the alias of a field that identifies the data set that is used to validate fields in a PROGRAM entry when no data set is specified in the field's List attributes. The Type attribute of the field that you specify should be set to **`Input`** and should contain the name of the desired SAS data set, but the AF task does not verify that these conditions are met.

Note: The Lookup data set attribute is applicable only to PROGRAM entries. △

Command menu

specifies the name of a PMENU entry that contains menu definitions for the application window.

Note: Because you can specify only a one-level name, the PMENU entry that you specify must reside in the same catalog as the current entry. △

Prompt character

specifies a character that a user can type in a field to get assistance or information about valid values for the field. A user must type the specified character in the first position of the field in order for it to be considered a prompt. The default prompt character is **`?`** (question mark).

The response to the prompt character depends on whether the List attribute is defined for the field.

- □ For fields to which a LIST attribute is assigned, a selection window opens, from which users can choose from the field values defined by the List attribute.
- □ For fields to which no LIST attribute is assigned, a dialog box opens that provides information about the type of values (character or numeric) that can be entered in the field.

You can prevent the prompting behavior for specific fields by specifying the NOPROMPT field attribute for the corresponding fields.

The Prompt character attribute applies only to PROGRAM entries.

System options

control the following window behaviors:

NO EXIT

prevents users from switching to other windows. If a user moves the cursor out of the application window and presses ENTER, the cursor returns to the application window.

EXTENDED TABLE

specifies that the scrollable portion of a PROGRAM entry's display is used as an extended table. See "Extended Tables" on page 37 for more information about extended tables.

RESIDENT

retains the entry's program in memory after it is first loaded. Keeping frequently used entries resident in memory can improve the performance of your applications by eliminating the wait for the entries to be loaded. However, making a large number of entries resident can cause your session to run out of memory.

Note: You can use the AFSYS command in the application window to determine which entries are currently resident in memory. △

□ *Parent Attributes*

Name, Type, Libref, Catalog
: specify the name of an entry to which control is passed when a user issues an END or CANCEL command. By default, control returns to the calling entry, except for CBT entries, which return control to the SAS System.

To pass control to another entry in the same catalog as the current entry, you only need to specify values for the Name and Type attributes. If the target entry is in another catalog or another library, specify values for the Libref and Catalog attributes as well.

□ *Window Type Attributes*

STANDARD
: specifies that the window can be resized and moved and can open other windows without passing permanent control to them.

DIALOG
: specifies that the window cannot be resized or moved. Select this attribute if you want to prevent users from issuing ZOOM, GROW, MOVE, or SHRINK commands in the application window. Dialog windows can open other windows and can pass temporary control to them.

If you assign a PMENU entry to the window, the menu selections are displayed as a row of buttons at the bottom of the window.

DIALOG BOX
: specifies that the window cannot be resized or moved and cannot open any other windows. Use the DIALOG BOX attribute for the last window that can be opened in a hierarchy.

If you assign a PMENU entry to the window, the menu selections are displayed as a row of buttons at the bottom of the window.

LIST
: specifies that the window can be resized and moved and can open other windows without passing permanent control to them. Assign this attribute to windows that are used as selection lists.

If you assign a PMENU entry to the window, the menu selections are displayed as a row of buttons at the bottom of the window.

HELP
: specifies that the window can be resized or moved but cannot open any other windows.

If you assign a PMENU entry to the window, the menu selections are displayed as a row of buttons at the bottom of the window.

□ *Scroll Bar Attributes*

HORIZONTAL
: controls whether a horizontal scroll bar is displayed along the bottom window border when scroll bars are turned on. Horizontal scroll bars are useful when the entry's display area is wider than the current window size.

VERTICAL
: controls whether a vertical scroll bar is displayed along the right window border when scroll bars are turned on. Vertical scroll bars are useful when the entry's display area is taller than the current window size.

The correct bibliographic citation for this manual is as follows: SAS Institute Inc., *SAS/AF® Software Procedure Guide, Version 8*, Cary, NC: SAS Institute Inc., 1999.

SAS/AF® Software Procedure Guide, Version 8

ISBN 1–58025–516–7

SAS Institute Inc., SAS Campus Drive, Cary, North Carolina 27513.

1st printing, October 1999