THE POWER TO KNOW.

SAS/AF® 9.3 Procedure Guide

SAS® Documentation

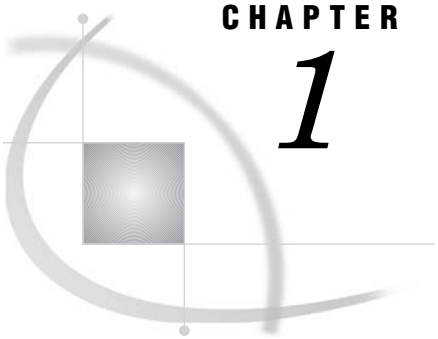
The correct bibliographic citation for this manual is as follows: SAS Institute Inc. 2011. *SAS/AF® 9.3 Procedure Guide*. Cary, NC: SAS Institute Inc.

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

1st electronic book, July 2011

SAS® Publishing provides a complete selection of books and electronic products to help customers use SAS software to its fullest potential. For more information about our e-books, e-learning products, CDs, and hard-copy books, visit the SAS Publishing Web site at **support.sas.com/publishing** or call 1-800-727-3228.

Contents



CHAPTER

1

Introduction to SAS/AF Software

Overview

SAS/AF software provides a set of tools that you can use to develop applications within the SAS System. SAS/AF applications provide interactive user interfaces to all the data access, management, analysis, and presentation features of SAS software.

SAS/AF applications are stored in SAS catalog entries. You use the BUILD procedure (or the BUILD command) in SAS/AF software to create the following types of catalog entries:

- FRAME entries, which provide visual, object-oriented components for developing applications in environments that support a graphical user interface
- PROGRAM and MENU entries, which provide character-based features for developing applications in environments where a graphical user interface is not available
- SCL entries, which store SAS Component Language (SCL) programs
- CBT and HELP entries, which provide information and assistance to application users.

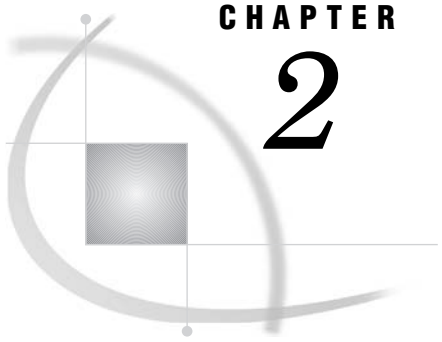
Other utility entry types can also be created to support your applications.

Although SAS/AF software is required for building applications, users do not need to license or install SAS/AF software in order to run the applications. The commands and procedure that launch SAS/AF applications are available in Base SAS software. Catalog entry types that provide a display run in the AF window, which supports its own set of window management commands in addition to any custom commands that your application supports.

Learning More

- For information about the syntax of the BUILD procedure and the BUILD command, see Chapter 2, “The BUILD Procedure,” on page 3.
- For information about the types of catalog entries you can create with the BUILD procedure, see Chapter 3, “SAS/AF Catalog Entry Types,” on page 21.
- For information about running the applications you create with the BUILD procedure, see Chapter 4, “Executing SAS/AF Applications,” on page 53.

- For information about the commands provided in the AF windows in which SAS/AF applications run, see Chapter 5, "AF Window Commands," on page 65.
- For an introduction to developing FRAME entry applications with SAS/AF software, refer to *SAS Guide to Applications Development*.



CHAPTER

2

The BUILD Procedure

Overview

You use the BUILD procedure in SAS/AF software to create applications that can

- display or update the contents of SAS data sets, catalogs, or external files, using either a graphical or character-based user interface
- create SAS statements to be submitted for processing by other SAS procedures
- provide menus for selecting other applications
- provide computer-based training
- provide online Help.

This section provides reference information about the BUILD procedure, including the complete syntax of the statements that are used in the procedure.

Note: You can also open the BUILD procedure windows by using

- the BUILD command
- the CALL BUILD routine in SAS Component Language
- the SAS Explorer window (when you select one of the SAS/AF catalog entry types described in Chapter 3, "SAS/AF Catalog Entry Types," on page 21).

See "BUILD Command" on page 18 for details about the BUILD command. Refer to *SAS Component Language: Reference* for information about using the CALL BUILD routine. Refer to the online Help for Base SAS software for information about using the Explorer window. △

BUILD Procedure Syntax

Note: You can use any number of COMPILE, MERGE, MLINK, PRINT, and SYNC statements with the PROC BUILD statement. The statements execute before any procedure windows are opened. All MERGE statements are executed first, followed by all PRINT statements, all MLINK statements, all SYNC statements, and finally all COMPILE statements.

PROC BUILD <CATALOG=<*libref.*>*catalog-name*<*.entry-name.entry-type*>>
<*options*>;

COMPILE <*options*>;

CROSSREF PROJECT=*libref.catalog-name* | (*catalog-list*) <*options*>;

MERGE CATALOG=*libref.catalog-name* <*options*>;

MLINK <*options*>;

PRINT *items* <*options*>;

SYNC <*options*>;

The PROC BUILD statement is required. The other statements are optional and are used as follows:

To do this	Use this statement
Compile the SAS Component Language code in FRAME, PROGRAM, and SCL entries	COMPILE
Collect information for the Static Analyzer performance analysis tool	CROSSREF
Combine entries from another SAS catalog into the current catalog	MERGE
Generate submenu links between MENU entries	MLINK
Print the contents of catalog entries	PRINT
Synchronize changes in RESOURCE, CLASS, or FRAME entries	SYNC

PROC BUILD Statement

Starts the BUILD procedure and specifies the catalog or catalog entry to open.

Tip: When you specify a four-level name for the CATALOG= argument, a BUILD procedure window opens in which you can edit the specified entry. By default, when you close the BUILD window, the procedure opens an Explorer window showing the contents of the catalog that contains the specified entry. If you want the procedure to end when you close the BUILD window, use the NODIR option in the PROC BUILD statement.

PROC BUILD <BATCH>
<BROWSE>
<CATALOG=<*libref.*>*catalog-name*<*.entry-name*<*.entry-type*>>>
<ENTRYTYPE=*entry-type*>
<NODIR>
<PADCHAR='*character*'>
<RESOURCE=<*libref.catalog-name.*>*resource-name*<.RESOURCE>>
<TESTAF <DEBUG>>
<TEXTLENGTH=*n*>;

Options

You can use the following options in the PROC BUILD statement:

BATCH
executes the BUILD procedure in batch mode rather than interactively. You cannot build individual entries in batch mode, but it is a convenient way to execute PRINT, COMPILE, MERGE, MLINK, and SYNC statements when you do not need to view the catalog or catalog entry. If you use the BATCH option without a PRINT, COMPILE, MERGE, MLINK, or SYNC statement, the specified catalog or catalog entry is not opened.

Note: When you use the BATCH option, all of the other PROC BUILD options except for CATALOG= are ignored. △

BROWSE
opens the specified catalog entry for browsing only. By default, the BUILD procedure tries to open the specified catalog entry for editing. Use the BROWSE option if you only want to view the entry without making changes.

Note: Catalog entries for which you have read-only access are automatically opened for browsing. △

CATALOG=<*libref.*>*catalog-name*<*.entry-name.entry-type*>
CAT=<*libref.*>*catalog-name*<*.entry-name.entry-type*>
C=<*libref.*>*catalog-name*<*.entry-name.entry-type*>
specifies the current catalog or the specific catalog entry to create, edit, or browse. The name that you specify with this argument is interpreted as follows:

- A one-level name identifies a catalog in the default library, WORK.
- A two-level name identifies a catalog in a specified library.
- A three-level name also identifies a catalog in a specified library. The third level is ignored and the first two levels (*libref.catalog-name*) are used.
- A four-level name identifies an entry of a specified type in a specified catalog.

Unless you also use the BATCH option, the procedure opens a window.

- If you supply a one-, two-, or three-level name, an Explorer window opens to display the contents of the specified catalog.
- If you specify a four-level name, the appropriate window opens for the specified entry type. See "BUILD Procedure Windows" on page 20 for information about the corresponding window for each entry type. Refer to Chapter 3, "SAS/AF Catalog Entry Types," on page 21 for information about the uses of the different catalog entry types. If the specified entry does not already exist, a new entry of the specified type is created.

If you omit the CATALOG= argument, a SAS Explorer window opens from which you can select a library and catalog and then create a new entry or select an existing entry to edit or browse.

ENTRYTYPE=*entry-type*
ETYPE=*entry-type*
ET=*entry-type*
specifies the default catalog entry type for the EDIT and BROWSE commands in BUILD procedure windows. The initial default entry type is PROGRAM. Use the ENTRYTYPE= option to change the default entry type to CBT, FRAME, HELP, MENU, or SCL.

NODIR
prevents the Explorer window from opening when a specified catalog entry is closed. By default, the Explorer window opens with the contents of the current catalog displayed when you close the BUILD procedure window. This makes it easy to continue working in the same catalog. Use the NODIR option to prevent this default behavior, in which case the BUILD procedure ends when you close the BUILD procedure window for the entry.

Note: The NODIR option is valid only when you specify a four-level name in the CATALOG= argument. △

PADCHAR='*character*'
specifies the default pad character that is displayed for empty user fields in PROGRAM entry windows. By default, the pad character is the underscore (_). Use the PADCHAR= option to specify a different default pad character for the BUILD session.

Note: The pad character for individual fields can be specified in the ATTR window. △

RESOURCE=<*libref.catalog-name.*>*resource-name*<.RESOURCE>
specifies the RESOURCE entry that is associated with all FRAME entries during the current build session, and also after the current session closes. The resource name is saved in the SAS Registry in the key named Resource under Products\AF\Design Time\Frame. All FRAME entries use the value of this key to specify their default RESOURCE.

The *resource-name* value must be the name of an existing RESOURCE entry. If you omit the *libref.catalog-name* value, the procedure looks for the specified RESOURCE entry in the current catalog, which is identified in the CATALOG= option.

To return the Resource key back to its original value, you can use the RESOURCE command, the RESOURCE= option, or the REGEDIT command.

TESTAF <DEBUG>
executes the specified CBT, FRAME, HELP, MENU, PROGRAM, or SCL entry in a test environment. You can test features such as field validation, the appearance of windows, and flow of control. Statements within submit blocks in PROGRAM entries are not submitted to the SAS session for processing. When you close the entry in the test environment, the Explorer window opens unless you also use the NODIR option.

Note: The TESTAF option is valid only when you specify a four-level name in the CATALOG= argument. △

The DEBUG option activates the SAS Component Language source-level debugger, provided that the SCL program in the specified entry was previously compiled with the DEBUG compile option.

TEXTLENGTH=*n*
TEXTLEN=*n*
specifies the length of the text line in the DISPLAY window for catalog entry types that use the SAS text editor (CBT, HELP, MENU, and PROGRAM). The default value is 78. Use the TEXTLENGTH= option to specify a shorter or longer line length. Valid values for *n* are 1 through 255.

This option is especially useful when you are developing applications that will be used on displays that are wider than the one on which they are being created.

Using the PROC BUILD Statement

The BUILD procedure enables you to create, edit, and manage SAS/AF catalogs and catalog entries. The types of entries you can build depend on your display environment.

- In graphical display environments, FRAME entries provide a graphical user interface and object-oriented programming tools for application development. The information in CLASS and RESOURCE entries defines the objects that are available in the development environment.
- In character-based display environments, SAS/AF applications usually consist of a combination of PROGRAM, CBT, MENU, and HELP entries. The information in EDPARMS, FORM, and KEYS entries supplements the application.

Refer to Chapter 3, "SAS/AF Catalog Entry Types," on page 21 for more information about the different types of catalog entries you can create and edit with the BUILD procedure.

COMPILE Statement

Compiles an application's source programs into stored, executable code.

Note: The compile operation is performed before the procedure opens any windows.

Tip: By default, all PROGRAM entries in the current catalog are compiled. You can use the SELECT= option to select individual entries to compile, or use the EXCLUDE= option to prevent certain entries from being compiled.

COMPILE <DEBUG>
<EXCLUDE=*entry-name* | (*entry-list*)> | <SELECT=*entry-name*<.*entry-type*> | (*entry-list*)>;

Options

You can use the following options in the COMPILE statement:

DEBUG
specifies that extra information is collected for the SAS Component Language (SCL) debugger when programs are compiled.

EXCLUDE=*entry-name* | (*entry-list*)
SELECT=*entry-name*<.*entry-type*> | (*entry-list*)
specify entries to exclude from or select for the compile operation. By default, all PROGRAM entries in the current catalog are compiled. Use the EXCLUDE= option to prevent specific PROGRAM entries from being compiled. Use the SELECT= option

to compile only the specified entries. To compile FRAME or SCL entries, you must explicitly specify the entry names, including the entry type, by using the SELECT= option. An error occurs if you specify an entry of any type other than FRAME, PROGRAM, or SCL. If you specify more than one entry name for either of these options, enclose the list in parentheses and separate the names with at least one space.

Each COMPILE statement can include only one EXCLUDE= option or one SELECT= option. However, you can use multiple COMPILE statements in the same PROC BUILD statement.

Using the COMPILE Statement

The SAS Component Language code in FRAME, PROGRAM, and SCL entries must be compiled before the source programs can run. If you do not use the COMPILE command while you are working in the catalog entries, you can use the COMPILE statement to compile the code in existing entries.

If error or warning messages are generated by the compile process, they are written to the SAS log.

COMPILE Statement Example

The following example illustrates a COMPILE statement that compiles three SCL source programs and collects the extra information needed to run the SCL debugger. In this example, the PROGRAM entries named ONE, TWO, and THREE in the catalog EX.APPLIC are compiled in batch mode. No interactive BUILD session is initiated.

```
proc build catalog=ex.applic batch;
   compile debug select=(one two three);
run;
```

CROSSREF Statement

Collects information about the SAS Component Language code in a SAS/AF application for use by the Static Analyzer tool.

Restriction: When you use a CROSSREF statement with the PROC BUILD statement, no other BUILD procedure statements can be used.

Note: When you use the CROSSREF statement, the procedure ends after the analysis data is collected. No BUILD procedure windows are opened, and any options that you specify for the PROC BUILD statement are ignored.

CROSSREF PROJECT=<*libref.*>*catalog-name* | (*catalog-list*)
<DETAIL=ALL | NONE | <*libref.*>*catalog-name* | (*catalog-list)*>
<EXCLUDE=*entry-name.entry-type* | (*entry-list*)> | <SELECT=*entry-name.entry-type* | (*entry-list*)>
<OUTLIB=*libref*>
<SEARCH=<*libref.*>*catalog-name* | (*catalog-list*)>;

Required Argument

You must always supply the following argument with the CROSSREF statement:

PROJECT= <*libref.*>*catalog-name* | (*catalog-list*)
specifies one or more SAS catalogs that comprise the project you want to analyze. If you specify more than one catalog, enclose the list in parentheses and separate the names with spaces.

Options

You can use the following options in the CROSSREF statement:

DETAIL=ALL | NONE | <*libref.*>*catalog-name* | (*catalog-list*)
controls the depth of data collection when functions that refer to other entries (such as CALL DISPLAY, CALL METHOD, or CALL GOTO) are encountered. Specify one of the following values:

ALL
the analysis traverses all links and collects data on all other entries that are referenced in the project. (This is the default behavior.)

NONE
the analysis does not traverse any links nor does it collect data on any entries outside the specified project catalogs.

<*libref.*>*catalog-name* | (*catalog-list*)
the analysis traverses links and collects data on only those entries that reside in the specified catalogs. If you specify more than one catalog name, you must enclose the list in parentheses and separate the names with spaces.

EXCLUDE=*entry-name.entry-type* | (*entry-list*)
SELECT=*entry-name.entry-type* | (*entry-list*)
specifies one or more entries for which no data is collected or for which data is collected. Use the EXCLUDE= option to prevent data from being collected for the specified entries. Data is still collected for the remaining entries in the catalogs specified in the PROJECT= or DETAIL= arguments. Use the SELECT= option to collect data for only the specified entries in the catalogs specified in the PROJECT= or DETAIL= arguments. If you specify more than one catalog entry name, you must enclose the list in parentheses and separate the names with spaces.

The EXCLUDE= and SELECT= options are mutually exclusive. You cannot use both options in the same CROSSREF statement.

OUTLIB=*libref*
specifies the libref for the SAS data library in which the analysis data sets are generated. The new analysis data sets replace any previous analysis data sets in the specified library. If you omit this option, the default is the WORK library.

SEARCH=<*libref.*>*catalog-name* | (*catalog-list*)
specifies one or more SAS catalog names that are used as the catalog search rule. The search rule is applied when a function that refers to another entry (such as CALL DISPLAY, CALL METHOD or CALL GOTO) includes a parameter that has a two-level name in the form *entry-name.entry-type*. The catalogs in the search rule are searched in the specified order for an entry that matches the two-level entry name. When no search rule is in effect, the default is to search only the catalog that contains the entry that includes the function.

If you specify more than one catalog name, you must enclose the list in parentheses and separate the names with spaces.

Using the CROSSREF Statement

The CROSSREF statement performs the data collection phase for the Static Analyzer, one of the performance analysis tools that are provided with SAS/AF software. The

Static Analyzer scans the SCL code for a project and reports usage information for the SCL program elements, including which functions, variables, constants, widgets, SAS macros, and SAS options are used in the project. For purposes of the analysis, you can define a project to include entries in multiple SAS catalogs and SAS data libraries.

After using the CROSSREF statement to collect data, you can view the results by issuing the following command in any SAS window:

SCLPROF STATIC <LIB=*analysis-library*>

Note: Use the LIB= option in the SCLPROF command if you used the OUTLIB= option in the CROSSREF statement to specify a library other than the default. △

For more information about the Static Analyzer, refer to the online Help for the tool.

CROSSREF Statement Examples

The following example illustrates the use of the SELECT= option to restrict data collection to a single entry:

```
proc build;
   crossref project=sashelp.aftools
            select=astopts.frame;
run;
```

The following example illustrates the use of the DETAIL= option to limit the depth of data collection in a large project:

```
proc build;
   crossref project=(sashelp.eis sashelp.eisbol1)
            detail=(sashelp.afclass sashelp.eisbol1
                    sashelp.fsp sashelp.mb)
            search=(sashelp.eisbol1 sashelp.mb
                    sashelp.eis sashelp.fsp
                    sashelp.assist sashelp.calc)
      outlib=sasuser;
run;
```

MERGE Statement

Copies entries from the catalog specified in the MERGE statement into the current catalog.

Note: The merge operation is performed before the procedure opens any windows.

Tip: By default, all entries in the specified catalog are copied to the current catalog (the catalog that was specified in the PROC BUILD statement). You can use the SELECT= option to select individual entries to copy, or use the EXCLUDE= option to prevent certain entries from being copied.

MERGE CATALOG=<*libref.*>*catalog-name*
<ENTRYTYPE=*type*>
<EXCLUDE=*entry-name.entry-type* | (*entry-list*)> | <SELECT=*entry-name.entry-type* | (*entry-list*)>
<NOEDIT>
<NOSOURCE>

<REPLACE>
<UPCASE>;

Required Argument

You must always supply the following argument with the MERGE statement:

CATALOG=<*libref.*>*catalog-name*
CAT=<*libref.*>*catalog-name*
C=<*libref.*>*catalog-name*
specifies the SAS catalog from which entries are copied. If you specify a one-level name, it is assumed to be a catalog name in the default WORK library.

Options

You can use the following options in the MERGE statement:

ENTRYTYPE=*entry-type*
ETYPE=*entry-type*
ET=*entry-type*
specifies the type of entry to copy into the merged catalog. By default, entries of all types are copied (unless you also use the SELECT= or EXCLUDE= option). Use the ENTRYTYPE= option to copy only entries of the specified type.

To copy entries of more than one type, use a separate MERGE statement for each entry type.

EXCLUDE=*entry-name.entry-type* | (*entry-list*)
SELECT=*entry-name.entry-type* | (*entry-list*)
specify entries to exclude from or select for the merge operation. By default, all entries in the specified catalog are copied into the current catalog (unless you also use the ENTRYTYPE= option). Use the EXCLUDE= option to prevent specific entries from being copied. Use the SELECT= option to copy only the specified entries. If you specify more than one entry name for either of these options, enclose the list in parentheses and separate the names with at least one space.

If you use the ENTRYTYPE= option in the same MERGE statement, you can omit the *entry-type* portion of the entry specification for the EXCLUDE= and SELECT= options because only entries of the type specified in the ENTRYTYPE= option can be copied.

Each MERGE statement can include only one EXCLUDE= option or one SELECT= option. However, you can use multiple MERGE statements with the same PROC BUILD statement.

NOEDIT
NOED
specifies that the entries that are merged into the current catalog cannot be edited with the BUILD procedure. This option is useful when you are moving entries from a development catalog into a production catalog and you want to prevent changes to the entries in the production catalog.

NOSOURCE
NOSRC
specifies that the SCL source programs for PROGRAM entries are not copied.

REPLACE
specifies that entries from the catalog specified in the MERGE statement replace like-named entries in the current catalog. By default, existing entries in the current catalog are not replaced.

UPCASE
converts all text to uppercase during the merge.

Using the MERGE Statement

The MERGE statement merges entries from the specified catalog, sorted in alphabetical order by entry type. Unless you use the ENTRYTYPE=, EXCLUDE=, or SELECT= options, the merge operation attempts to copy all entries from the specified catalog to the current catalog (the catalog that was specified in the PROC BUILD statement). However, if any entries in the current catalog have the same names and types as entries in the specified catalog, then the existing entries in the current catalog are not replaced unless you use the REPLACE option.

MERGE Statement Examples

Example 1: Using the SELECT= and ENTRYTYPE= Options

The following example copies only the entry AAA.HELP from the catalog OLD.CAT1 into the catalog NEW.CAT2:

```
proc build catalog=new.cat2;
   merge catalog=old.cat1 entrytype=help select=aaa;
run;
```

Example 2: Using the EXCLUDE= and ENTRYTYPE= Options

The following example copies all the HELP entries except for AAA.HELP and BBB.HELP from the catalog OLD.CAT1 into the catalog NEW.CAT2:

```
proc build catalog=new.cat2;
   merge catalog=old.cat1 entrytype=help exclude=(aaa bbb);
run;
```

Example 3: Using the SELECT= Option without the ENTRYTYPE= Option

The following example copies only the entries A.MENU and B.PROGRAM from the catalog OLD.CAT1 into the catalog NEW.CAT2:

```
proc build catalog=new.cat2;
   merge catalog=old.cat1 select=(a.menu b.program);
run;
```

Example 4: Using the REPLACE Option

The following example copies all entries of type CBT from the catalog EX.FINAL into the catalog EX.COURSES. If a CBT entry in EX.COURSES has the same name as one that is being copied, the procedure replaces it with the copied entry.

```
proc build c=ex.courses;
   merge c=ex.final entrytype=cbt replace;
run;
```

MLINK Statement

Generates menu links for MENU entries that have the Menu-Link attribute in the catalog specified in the PROC BUILD statement.

Note: The menu-linking operation is performed before the procedure opens any windows.

Tip: By default, all MENU entries in the current catalog (the catalog that was specified in the PROC BUILD statement) are checked for selections that have the Menu-Link attribute. You can use the SELECT= option to select individual entries to check, or use the EXCLUDE= option to prevent certain entries from being checked.

MLINK <EXCLUDE=*entry-name* | (*entry-list*)> | <SELECT=*entry-name* | (*entry-list*)>
<LEVELS=*n* | _MAX_>
<VERBOSE>;

Options

You can use the following options in the MLINK statement:

EXCLUDE=*entry-name* | (*entry-list*)
SELECT=*entry-name* | (*entry-list*)
specify the MENU entries to exclude from or select for the linking operation. By default, all MENU entries in the current catalog are checked for selections that have the Menu-Link attribute. Use the EXCLUDE option to prevent specific MENU entries from being checked for links. Use the SELECT= option to check only the specified entries for links. If you specify more than one entry for either of these options, enclose the names in parentheses and separate the names with spaces.

LEVELS=*n* | _MAX_
specifies the number of levels of submenus to be linked. By default, only one level of submenu selections is linked. Use the LEVELS= option to specify a different number of levels. Use the _MAX_ option value to link all designated submenus.

VERBOSE
produces additional messages for each menu that has the Menu-Link attribute. These messages identify the name of the MENU entry from which selections are being linked, as well as the level number of the link. By default, messages for each MENU entry report only the start and completion of the MLINK operation, the name of MENU entries that contain no links, and any error conditions that are found.

Using the MLINK Statement

Menu links enable users to access submenu choices directly from an application's higher-level menu. Linked menus are useful to users who have become familiar with a system and who want to bypass intermediate menus to directly invoke choices on secondary menus. For example, suppose a MENU entry includes a selection **3** that opens another MENU entry, and that the second MENU entry includes a selection **SALES** that displays a sales report. If menu links are generated for the first MENU entry, then users can go directly to the sales report by specifying **SALES** in the first menu, without having to open the second menu.

Menu linking is applicable only to MENU entries. In order for a menu to be linked, the Menu-Link attribute must be assigned to it in the ATTR window of the calling

MENU entry. Each time you change menus or submenus, you should re-generate the links to ensure that all linked selections are available from the highest-level menu.

Note: You can also generate menu links by issuing the MLINK command while building the MENU entries. △

PRINT Statement

Prints the contents of entries from the catalog specified in the PROC BUILD statement.

Restriction: The PRINT statement can print the contents of CBT, HELP, LIST, MENU, PROGRAM, SCL, and SOURCE entries. It cannot print the contents of FRAME, CLASS, RANGE, and RESOURCE entries or of any other entry types that are not supported by the BUILD procedure.

Note: The print operation is performed before the procedure opens any windows.

Tip: By default, the procedure attempts to print all entries in the current catalog (the catalog that was specified in the PROC BUILD statement). You can use the SELECT= option to select individual entries to print, or use the EXCLUDE= option to prevent certain entries from being printed.

PRINT *items*
<ENTRYTYPE=*entry-type*>
<EXCLUDE=*entry-name.entry-type* | (*entry-list*)> | <SELECT=*entry-name.entry-type* | (*entry-list*) <XREF>>
<FORM=*form-name*>
<LEFT>
<LINENUM>
<NOPAGEBREAK>
<PRTFILE='*filename*' | *fileref* <APPEND>>;

where *items* must be one or more of the following:
ATTR
DISPLAY<SHOWPAD>
LISTDIR
SOURCE

Required Arguments

You must always supply at least one of the following arguments with the PRINT statement. Any combination of the following arguments can be specified.

ATTR
prints all of the attribute information that is associated with each entry.

DISPLAY <SHOWPAD | SP>
DISP <SHOWPAD | SP>
prints the display portion of each entry.

For PROGRAM entries, user fields in the display are by default represented by their respective field names. Use the SHOWPAD option in conjunction with the

DISPLAY option to represent user fields by their corresponding pad characters instead.

For LIST entries, the values in the list are printed.

LISTDIR
LD
prints a listing of the contents of the catalog specified in the PROC BUILD statement.

SOURCE
SRC
prints the SAS Component Language source code from each PROGRAM, SCL, or SOURCE entry.

Options

You can use the following options in the PRINT statement:

ENTRYTYPE=*entry-type*
ETYPE=*entry-type*
ET=*entry-type*
specifies the type of entry to print. By default, all CBT, HELP, LIST, MENU, PROGRAM, SCL, and SOURCE entries in the current catalog are printed (unless you also use the SELECT= or EXCLUDE= option). Use the ENTRYTYPE= option to print only entries of the specified type.

To print entries of more than one type, use a separate PRINT statement for each entry type.

EXCLUDE=*entry-name.entry-type* | (*entry-list*)
SELECT=*entry-name.entry-type* | (*entry-list*) <XREF>
specify catalog entries to exclude from or select for printing. By default, all CBT, HELP, LIST, MENU, PROGRAM, SCL, and SOURCE entries in the current catalog are printed (unless you also use the ENTRYTYPE= option). Use the EXCLUDE= option to prevent specified entries from being printed. Use the SELECT= option to print only specified entries. If you specify more than one entry for either of these options, enclose the list in parentheses and separate the names with spaces.

If you use the ENTRYTYPE= option in the same PRINT statement, you can omit the *entry-type* portion of the entry specification for the EXCLUDE= or SELECT= options because only entries of the type specified in the ENTRYTYPE= option can be printed.

When you use the SELECT= option to print CBT or MENU entries, you can also use the XREF option to print two cross-reference tables for each MENU or CBT entry. One table lists all entries that are called by that entry, and the other table lists all entries that call the entry.

Each PRINT statement can include only one EXCLUDE= option or one SELECT= option. However, you can use multiple PRINT statements with the same PROC BUILD statement.

FORM=*form-name*
specifies the name of a FORM entry to control the output generated by the PRINT statement. Specify either a one- or three-level name for *form-name*. A one-level name is assumed to be the name of a FORM entry in the current catalog. If the FORM entry is in a different catalog, use a three-level name (*libref.catalog-name.entry-name*). The default is the standard system form, SASHELP.FSP.DEFAULT.FORM.

LEFT
specifies that output produced by the PRINT statement is aligned at the left margin of the page or print file. By default, output is indented four spaces.

LINENUM
prints line numbers (up to 99999) at the beginning of each line of SCL source code.

Note: The LINENUM option is valid only when SOURCE is specified for the *items* argument. △

NOPAGEBREAK
prints entries without page breaks. By default, a page break is generated each time the number of lines per page defined by your FORMS entry is reached, and each page of the program listing has a page header. Blank lines are printed at the end of the program listing until the specified page length is reached. Use the NOPAGEBREAK option to print the program listing without page breaks. Headers are printed only at the top of the entry and every 32,768 lines thereafter.

Note: The NOPAGEBREAK option is valid only when SOURCE is specified for the *items* argument. △

PRTFILE='*filename*' | *fileref* <APPEND>
specifies a file to receive the output of the PRINT statement. By default, output is sent to the default printer or to the printer specified in the form identified in the FORM= option. Use the PRTFILE= option to redirect the output to a file instead. Specify the print file using either

- a fully-qualified filename enclosed in quotes
- a fileref that has been assigned to the file with a FILENAME statement or with an external allocation.

By default, if output is sent to a print file that already exists, the output overwrites the current contents of the file. Add the APPEND option to append printed output to the end of the file if it already exists.

Using the PRINT Statement

You can use the PRINT statement to generate hardcopy documentation of the displays, attributes, and source code in your applications. By default, output is sent to the default printer or to the printer specified in the FORM= option. You can use the PRTFILE= option to route output to a file instead.

PRINT Statement Examples

Example 1: Selecting Entries to Print

The following example shows how to print the DISPLAY window views for two entries, GETFIELD.PROGRAM and GETTYPE.PROGRAM. Pad characters are printed to show the DISPLAY windows as users see them. Because the PRTFILE= option is not specified, the output is sent to the default printer. This example uses batch mode, which is the optimum way to print entries when you do not need to view the catalog or its entries.

```
proc build cat=mylib.mycat batch;
   print display showpad et=program
         select=(getfield gettype);
run;
```

Example 2: Excluding Entries from Printing

The following example shows how to print the SCL source programs for all PROGRAM entries in the MYLIB.MYCAT catalog except GETFIELD.PROGRAM and GETTYPE.PROGRAM. Because the BATCH option is not specified, the Explorer window opens after the printing is performed so that you can work in the BUILD session.

```
proc build cat=mylib.newcat;
   print source etype=program
         exclude=(getfield gettype);
run;
```

Example 3: Printing a Cross Reference

The following example prints a cross reference of entries that call or are called by the MAIN.MENU entry. Output is routed to the file identified by the fileref MENUREF, which you must have previously assigned.

```
proc build cat=mylib.newcat batch;
   print xref select=main.menu
         prtfile=menuref;
run;
```

SYNC Statement

Updates RESOURCE entries to incorporate changes made to one or more CLASS entries contained in the RESOURCE entries, or updates CLASS and FRAME entries to re-establish links from instance variables in the CLASS or FRAME entries to the parent classes.

Note: The synchronization operation is performed before the procedure opens any windows.

Tip: By default, all CLASS, FRAME, and RESOURCE entries in the current catalog (the catalog that was specified in the PROC BUILD statement) are synchronized. You can use the SELECT= option to select individual entries to synchronize, or use the EXCLUDE= option to prevent certain entries from being synchronized.

SYNC <ENTRYTYPE=*entry-type*>
<EXCLUDE=*entry-name.entry-type* | (*entry-list*)> | <SELECT=*entry-name.entry-type* | (*entry-list*) >

ENTRYTYPE=*entry-type*
ET=*entry-type*
specifies the type of entry to synchronize. By default, all CLASS, FRAME, and RESOURCE entries in the current catalog are synchronized (unless you also use the SELECT= or EXCLUDE= option). Use the ENTRYTYPE= option to synchronize only entries of the specified type.

To synchronize entries of more than one type, use a separate SYNC statement for each entry type.

EXCLUDE=*entry-name.entry-type* | (*entry-list*)
SELECT=*entry-name.entry-type* | (*entry-list*)
specify catalog entries to exclude from or select for synchronizing. By default, all CLASS, FRAME, and RESOURCE entries in the current catalog are synchronized (unless you also use the ENTRYTYPE= option). Use the EXCLUDE= option to prevent specified entries from being synchronized. Use the SELECT= option to synchronize only specified entries. If you specify more than one entry for either of these options, enclose the list in parentheses and separate the names with spaces.

If you use the ENTRYTYPE= option in the same SYNC statement, you can omit the *entry-type* portion of the entry specification for the EXCLUDE= or SELECT= options because only entries of the type specified in the ENTRYTYPE= option can be synchronized.

Each SYNC statement can include only one EXCLUDE= option or one SELECT= option. However, you can use multiple SYNC statements with the same PROC BUILD statement.

Using the SYNC Statement

Objects and classes can link their instance variables to their parent classes, enabling you to change instance variables in a class. The change will also affect instances in subclasses of that class. However, it is possible to break the link to the parent class by changing a value to something other than the value specified in the parent, then changing it back. This gives you an instance variable that has the same value as the variable in the parent class. Synchronization restores this link by removing the duplicate item from the object attribute list (for FRAME entries) or from the class instance variables list (for CLASS and RESOURCE entries).

Note: You can also synchronize CLASS, FRAME, and RESOURCE entries by issuing the SYNC command in the BUILD procedure windows while editing the entries. △

BUILD Command Syntax

Tip: The BUILD command provides an easy way to open a BUILD window from any SAS command line.

BUILD <<*libref*.>*catalog-name*<.*entry-name*<.*entry-type*>>>
<RESOURCE=<*libref.catalog-name*.>*resource-name*<.RESOURCE>>

BUILD Command

Initiates a BUILD session.

BUILD <<*libref*.>*catalog-name*<.*entry-name*<.*entry-type*>>>
<RESOURCE=<*libref.catalog-name*.>*resource-name*<.RESOURCE>>

Options

You can specify the following optional arguments with the BUILD command:

<libref.>catalog-name<.entry-name<.entry-type>>
specifies the catalog or catalog entry to open. Argument values are interpreted as follows:

- A one-level name (*catalog-name*) identifies a catalog in the default library, WORK. Remember that the contents of the WORK library are deleted when the SAS session ends.
- A two-level name (*libref.catalog-name*) identifies a catalog in a specified library.

- A three-level name (*libref.catalog-name.entry-name*) identifies a PROGRAM entry in a specified catalog.
- A four-level name (*libref.catalog-name.entry-name.entry-type*) identifies an entry of a specified type in a specified catalog.

The form of the argument determines which window the BUILD command opens, as follows:

- If you supply a one- or two-level name, an Explorer window opens to display the contents of the catalog. You can then select an existing entry in the catalog to edit, or you can create a new catalog entry.
- If you supply a three-level name, the DISPLAY window opens for editing the specified PROGRAM entry.
- If you supply a four-level name, the appropriate window opens for the specified entry type.

See "BUILD Procedure Windows" on page 20 for information about the corresponding window for each entry type. If you specify an entry name that does not already exist, a new entry of the specified type is created. Refer to Chapter 3, "SAS/AF Catalog Entry Types," on page 21 for information about the uses of the different catalog entry types.

If you issue a BUILD command with no catalog or entry name argument, a SAS Explorer window opens, from which you can select a library and catalog and then either select an existing catalog entry or create a new catalog entry.

RESOURCE=<*libref.catalog-name.*>*resource-name*<.RESOURCE>
specifies the RESOURCE entry that is associated with all FRAME entries during the current build session, and also after the current session closes. The resource name is saved in the SAS Registry in the key named Resource under Products\AF\Design Time\Frame. All FRAME entries use the value of this key to specify their default RESOURCE.

The *resource-name* value must be the name of an existing RESOURCE entry. If you omit the *libref.catalog-name* value, the procedure looks for the specified RESOURCE entry in the current catalog, which is identified in the CATALOG= option.

To return the Resource key back to its original value, you can use the RESOURCE command, the RESOURCE= option, or the REGEDIT command.

Using the BUILD Command

The BUILD command can be issued from any SAS window. Each BUILD command starts a separate BUILD session, so you can have several BUILD sessions running at the same time, each building different entries in the same or different catalogs. If you attempt to open an entry that is already open for editing in a different BUILD session, it is opened for browsing rather than editing.

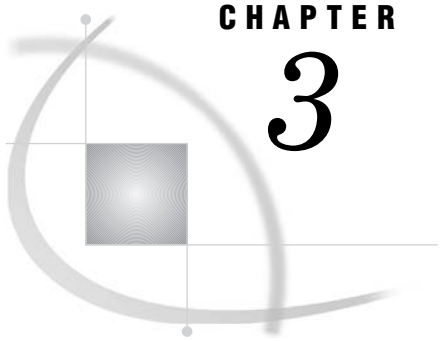
BUILD Procedure Windows

When you specify a catalog entry to create, edit, or browse, the BUILD procedure opens the entry in the appropriate BUILD window for the entry type. The following table shows the corresponding window for each of the catalog entry types that the BUILD procedure supports:

Entry Type	Window Opened
CBT	DISPLAY
CLASS	Class Editor
EDPARMS	EDPARMS
FORM	FORM
FRAME	DISPLAY and Components
HELP	DISPLAY
INTRFACE	Interface Editor
KEYS	KEYS
LIST	LISTATTR and LISTVALUES
MENU	DISPLAY
PROGRAM	DISPLAY
RANGE	RANGE
RESOURCE	Resource Editor
SCL	SOURCE

For more information about the behavior of each of the BUILD procedure windows, including details of the commands you can use in the windows, refer to the online Help for the corresponding window. For information about the entry features that you can define in the BUILD procedure windows, see Chapter 3, "SAS/AF Catalog Entry Types," on page 21.

Note: The EDPARMS, FORM, and KEYS windows are SAS windowing environment windows that the BUILD procedure opens as a convenience for application developers. Refer to the SAS Help System for information about the EDPARMS, FORM, and KEYS windows. △



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.

Table 3.1 SAS/AF Catalog Entry Types

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 the SAS Help System 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 the online Help for SAS/AF software 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 the online Help for SAS/AF software 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 the online Help for SAS/AF software.

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 the online Help for SAS/AF software.

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 the online Help for Base SAS software for more information on global commands for the SAS windowing environment.

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 left to right starting in the upper-left corner and descending.

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 43 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 session.

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.

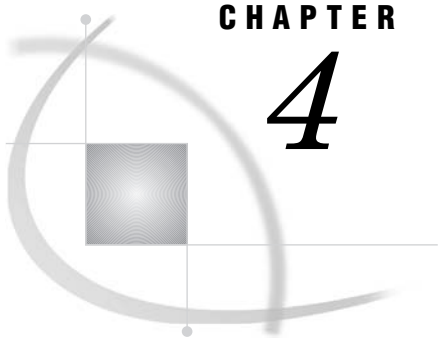
HELP
specifies that the window can be resized or moved but cannot open any other windows.

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.



CHAPTER

4

Executing SAS/AF Applications

Overview

Although you must license SAS/AF software in order to create applications with the BUILD procedure, only Base SAS software is required in order to run the applications that you create. Users can run your SAS/AF applications

- by issuing the global AF or AFAPPLICATION command
- by submitting the DISPLAY procedure in Base SAS software
- by executing the CALL DISPLAY routine in SAS Component Language.

The AF and AFAPPLICATION commands are available in all SAS windows. The following sections describe the syntax and usage of these commands. Refer to *Base SAS Procedures Guide* for information about the DISPLAY procedure. Refer to *SAS Component Language: Reference* for information about the CALL DISPLAY routine.

AF Command

Use the AF command to execute applications that have been created with the BUILD procedure in SAS/AF software.

Note: The AF command can execute entry types that provide a display (CBT, FRAME, HELP, MENU, or PROGRAM), as well as SCL entries that execute without a display. The other SAS/AF catalog entry types are not executable and cannot be executed with the AF command. △

You can issue the AF command from any SAS window. Only one application that is invoked by the AF command can be active at a time. If you issue an AF command while another application is already running, the previous application is closed before the new application is executed. To run additional SAS/AF applications simultaneously, use the AFAPPLICATION command instead. See "AFAPPLICATION Command" on page 59 for details.

Syntax

The general form of the AF command is

AF <AUTORECALL=YES | NO>
<AUTOSAVE=YES | NO>
<AUTOTERM=YES | NO | VERBOSE | NOVERBOSE>
<AWS=YES | NO | KEEP>
<CATALOG=*libref.catalog-name.entry-name.entry-type*>
<CATNAME=*libref.catalog-reference*(*libref.catalog-1 ... libref.catalog-n*)>
<CHECKLAST=YES | NO>
<DEBUG=YES | NO>
<FRAME=*frame-number* | *frame-name*>
<ICON=*icon-number*>
<LABEL=*label*>
<PMENU=YES | NO>
<RESIDENT=*number*>
<RESTART=YES | NO>
<SCLPROF=COVERAGE | TIMER>
<TITLE='*title*'>
<*application-options*>

Requirement

The first time you use the AF command, you must use the CATALOG= argument to identify the application to execute. The name of the entry at which an application starts executing is recorded in the special AF.AFGO entry in your SASUSER.PROFILE catalog. This entry is referred to as the AF checkpoint. Any subsequent AF command that does not include the CATALOG= argument starts at the entry identified in the AF checkpoint. The AF checkpoint is retained across SAS sessions.

Note: If you exit an application from a CBT entry, the name of the CBT entry from which you exit the application is stored as the AF checkpoint rather than the name of the initial entry in the application. This enables you to resume CBT applications at the point where you stopped. Use the CHECKLAST=NO option to override this default behavior and to store the name of the initial entry even when exiting from CBT entries. △

Options

For most of the following AF command options, there is a default action if you do not use the option. You should use an option only when you want to change the default behavior.

AUTORECALL=YES | NO
specifies whether field values that have been saved from a previous invocation of a PROGRAM entry are recalled when the entry is invoked again. By default, field values are not recalled (AUTORECALL=NO). Use AUTORECALL=YES if you want the stored field values to be recalled. Stored values are available only if the entry has previously been closed while the AUTOSAVE=YES option was in effect.

Note: The AUTORECALL= option only affects the behavior of PROGRAM entries. △

AUTOSAVE=YES | NO
specifies whether the values in the fields of PROGRAM entries are saved when you exit from an entry. By default, field values are not saved (AUTOSAVE=NO). Use AUTOSAVE=YES to store all current field values when you close a PROGRAM entry. You can use the AUTORECALL=YES option to recall these values the next time the entry is invoked.

Note: The AUTOSAVE= option only affects the behavior of PROGRAM entries. △

AUTOTERM=YES | NO | VERBOSE | NOVERBOSE
specifies whether the _term method of FRAME entry objects is automatically executed if the objects still exist when the entry ends. By default, any open objects are automatically terminated (AUTOTERM=YES). Use AUTOTERM=NO if you want to prevent the _term method of open FRAME objects from being executed when the entry ends.

By default, no list of open objects is generated (AUTOTERM=NOVERBOSE). Use AUTOTERM=VERBOSE to print to the SAS log a note containing the object list for each object that still exists when the entry ends. This feature works even if automatic object termination is off, and it serves as a debugging aid to identify objects whose _term method has not run.

You cannot combine options in one string. Instead, use a separate AUTOTERM= option with the AF or AFAPPLICATION command. For example:

```
af c=mylib.cat.primary.frame autoterm=verbose autoterm=yes
```

Note: The AUTOTERM= option only affects the behavior of FRAME entries. △

AWS=YES | NO | KEEP
specifies whether the application's windows are confined to a container window called the application workspace (AWS). When an application workspace is used, you can minimize or maximize the entire application by minimizing or maximizing the AWS window.

Note: The AWS= option is ignored if your host environment does not support application workspaces. △

Specify one of the following values for the AWS= option:

YES	opens all windows displayed by the AF application (including windows displayed by the CALL DISPLAY routine) within the single AWS window. This is the default behavior unless it is overridden by a host-specific resource.
NO	opens each entry in an application in its own window rather than in the AWS window.
KEEP	keeps the application workspace open after the last window in the workspace is removed. This prevents the AWS from being deleted during situations such as unconditional branching in CBT entries or using CALL GOTO routines in SCL programs.

In these situations, all the current windows for the application are closed before other windows are opened.

CATALOG=*libref.catalog-name.entry-name.entry-type*
CAT=*libref.catalog-name.entry-name.entry-type*
C=*libref.catalog-name.entry-name.entry-type*
specifies the name and type of the SAS catalog entry at which an application starts executing. You must specify a complete four-level name. You must include this argument the first time you use the AF command.

If you omit the CATALOG= option, the AF command starts the entry that is identified as your AF checkpoint in the SASUSER.PROFILE.AF.AFGO entry. If the application starts executing at a FRAME, PROGRAM, MENU, or HELP entry, then the name of the initial entry is identified in the AF checkpoint. If you exit the application from a CBT entry, then the CBT entry from which you exit is identified as the AF checkpoint.

CATNAME=*libref.catref* (*libref.catalog-1 ... libref.catalog-n*)
logically combines two or more catalogs into one by associating them with a catref (a shortcut name). You can use any valid SAS name for *catref*, but if you use the name of an existing catalog you will not be able to access the contents of that catalog until the catref is cleared. The libref that you specify with the catref must already exist. Enclose the list of catalogs in parentheses, and use blanks to separate the catalog names in the list.

When a program in the SAS/AF application contains a reference to an entry in the *libref.catref* catalog, the AF task searches for the entry in the specified list of catalogs, starting with *catalog-1* and ending with *catalog-n*.

For example, suppose that you open an application with

```
af cat=mylib.mycat.main.frame
   catname=mylib.all (mylib.apps1 mylib.apps2 master.apps)
```

A reference in a program to MYLIB.ALL.TEST.SCL causes the AF task to search for MYLIB.APPS1.TEST.SCL, then for MYLIB.APPS2.TEST.SCL, and finally for MASTER.APPS.TEST.SCL.

Refer to the description of the CATNAME statement in *SAS Language Reference: Dictionary* for more information about creating and using catrefs. You can also create and clear catrefs by using the CATNAME command in AF windows. See the description of the CATNAME command in "Window Management Commands" on page 66 for details.

CHECKLAST=YES | NO
CHECK=YES | NO
specifies whether the system stores the name of the CBT entry at which you exit an application. By default, the name of the current entry is stored as the AF checkpoint when you exit an application from a CBT entry (CHECKLAST=YES). The AF checkpoint identifies the application that runs when you issue an AF command without using a CATALOG= argument. Use CHECKLAST=NO to record the name of the initial entry for the application instead, so that the CBT application resumes at the beginning rather than at the entry where you stopped.

Note: The CHECKLAST= option only affects the behavior of CBT entries. △

DEBUG=YES | NO
specifies whether the SCL source-level debugger runs on an application's programs. By default, the debugger is not activated (DEBUG=NO). Use DEBUG=YES to activate the debugger.

Note: In order to analyze programs with the SCL debugger, the programs must be compiled with the compiler's DEBUG ON option. △

The SCL debugger can interactively monitor the execution of SAS/AF applications. It enables you to track down logical errors while the program executes. The debugger displays the source program, specifies which line is executing, and dynamically watches the values of variables. It also enables you to suspend an executing program that is part of a nested series of programs and to execute other programs in the series. For more information about the SCL debugger, see *SAS Component Language: Reference*.

FRAME=*frame-number* | *frame-name*
specifies the starting frame for a CBT application. Identify the starting frame by using either of the following:

- the frame number. To determine the number of a frame in a CBT entry, use the ID command when that frame is displayed.
- the frame name, which is the name specified in the NAME= option on the frame delimiter line. This method is more efficient because the AF command can remain the same even if the number of frames changes.

For testing CBT applications, the FRAME= option provides a convenient way to return directly to a specific frame in the system. It is also useful for indexing a CBT course that contains multiple topics.

Note: The FRAME= option only affects the behavior of CBT entries. △

ICON=*icon-number*
specifies the number of the icon that is used to represent an AWS window when the window is minimized. By default, the SAS/AF icon is displayed.

Note: This option has an effect only if the native windowing system supports application workspaces and you also use the AWS=YES option. △

LABEL=*label*
specifies the name of an SCL program label where execution begins when the AF command is used to execute a stand-alone SCL entry. Execution begins at the specified label and continues until a RETURN statement is reached.

Note: The LABEL= option only affects the behavior of SCL entries. △

PMENU=YES | NO
specifies whether the PMENU facility can be turned off in an application. By default, users can issue the PMENU OFF command to turn off menus in the application (PMENU=NO). Use PMENU=YES if you have customized the menus in your windows and want to ensure that users cannot turn the PMENU facility off for windows in your application.

If your SAS/AF application invokes additional applications by issuing an AF or AFAPPLICATION command with the EXECCMD routine, include the PMENU=YES option in the command if you want the additional applications to have the same behavior.

RESIDENT=*number*
specifies the number of SCL entries that are kept resident in memory after they are read from the catalog. Entries can be read from memory much more quickly than from the catalog, so keeping frequently used SCL entries in memory improves the performance of SAS/AF applications.

When an SCL entry is invoked, the AF task searches resident memory for the entry. If the search is successful, the entry moves to the top of the search list. An SCL entry that is called frequently remains at or near the top of the list and so is found more quickly. If the SCL entry is not found in the search list, it is read from the catalog and inserted at the top of the list. If the maximum number of entries

are already resident, then the last entry in the search list (the least-recently used) is removed to make room for the new entry.

By default, 64 SCL entries are saved in memory. The *number* value is interpreted as follows:

0	No SCL entries are kept in memory. All SCL entries must be read from the catalog each time they are called.
> 0	The specified number of entries are kept in memory.

RESTART=YES
specifies whether CBT entries are restarted from the beginning. By default, a CBT entry starts at the CBT frame that was last accessed (RESTART=NO). Use RESTART=YES if you want to ensure that the CBT entry starts at the first frame.

Note: The RESTART= option only affects the behavior of CBT entries. △

SCLPROF=COVERAGE | TIMER
starts the data collection phase for the specified diagnostic tool.

COVERAGE	starts data collection for the Coverage Analyzer tool. The Coverage Analyzer can uncover gaps in your interactive applications testing by identifying which lines in an application are not executed during a test session. When you end the application that was started by the AF command, the SCL Coverage Analyzer window opens to present the results of the analysis. Refer to the online Help for the Coverage Analyzer for more information on using this diagnostic tool.
TIMER	starts data collection for the Performance Analyzer tool. The Performance Analyzer provides timing and frequency statistics for each entry, label, and function call that is executed in an application's SCL code. It also provides a hierarchical view of the execution sequence for the application. When you end the application that was started by the AF command, the SCL Performance Analyzer window opens to present the results of the analysis. Refer to the online Help for the Performance Analyzer for more information on using this diagnostic tool.

TITLE='*title*'
specifies a title for the AWS window. If the title contains embedded blanks, enclose it in single quotes, as in the following example:

```
af c=corp.fin.inv.program title='Inventory Analysis'
```

Note: The TITLE= option has an effect only if the native windowing system supports application workspaces and you also use the AWS=YES option. △

application-options
are options for the application that is being invoked by the AF command. These options are handled differently, depending on which entry type is being invoked.

- For FRAME, PROGRAM, or SCL entries, values are passed to the entry in the SCL list _CMDLIST_, which is a sublist of the local environment list. See "Passing Options to an Application" on page 61 for details.
- For CBT entries, values specified following the AF command options cause the AF task to search the entry's initial frame for a feedback indicator line that has a matching MENU=*value* option. If a match is found, the CBT frame specified in the line's FRAME= option is displayed.

For example, the following command causes the AF task to search the first frame of the entry USING.CBT for a feedback indicator line that contains the option MENU=HELLO:

```
af c=company.sales.using.cbt hello
```

If a matching option is found, the AF task branches to the CBT frame that is specified in the line's FRAME= option. If no matching MENU= option is found, then the **HELLO** argument is ignored.

- For MENU entries, values that follow the AF command options are matched with that menu's selection options.

  For example, the following command causes the AF task to search the MAIN.MENU entry for a selection option named HELLO:

  ```
  af c=company.sales.main.menu hello
  ```

  If the specified option exists, the AF task branches to the entry that is invoked by the selection option. If HELLO is not a valid selection, the AF task displays the following message:

  ```
  Warning: no HELLO selection for this menu
  ```

  You can also specify an option number on the primary menu or on submenus. To specify option numbers for submenus, separate the option numbers by periods. For example, you can specify 3.2 to display the entry called by choice 2 from the submenu that is called by choice 3 of the main menu.
- For HELP entries, values that follow the AF command options are ignored.

Using the AF Command

For entry types that provide a display (CBT, FRAME, HELP, MENU, and PROGRAM), the AF command opens the specified entry in an AF window. You can use SAS windowing environment commands while executing applications in the AF window, including commands that move, resize, and change the windows. However, window sizes and colors cannot be saved. Window properties are determined permanently when an application is designed. See Chapter 5, "AF Window Commands," on page 65 for information about other commands that are supported in the AF window.

You can also use the AF command to execute SCL entries. In this case, the SAS Component Language code in the entry is executed without opening a display window.

AFAPPLICATION Command

The AFAPPLICATION command is similar to the AF command, but it enables you to run multiple SAS/AF applications simultaneously. Each application that is executed with the AFAPPLICATION command runs as a separate task.

Syntax

The AFAPPLICATION command uses the same syntax as the AF command:

AFAPPLICATION <AUTORECALL=YES | NO>
<AUTOSAVE=YES | NO>
<AUTOTERM=YES | NO | VERBOSE | NOVERBOSE>

<AWS=YES | NO | KEEP>
<CATALOG=*libref.catalog-name.entry-name.entry-type*>
<CATNAME=*libref.catalog-reference*(*libref.catalog-1 ... libref.catalog-n*)>
<CHECKLAST=YES | NO>
<DEBUG=YES | NO>
<FRAME=*frame-number* | *frame-name*>
<ICON=*icon-number*>
<LABEL=*label*>
<PMENU=YES | NO>
<RESIDENT=*number*>
<RESTART=YES | NO>
<SCLPROF=COVERAGE | TIMER>
<TITLE=*title*>
<*application-options*>

Note: The AFAPPLICATION command can be abbreviated as AFAPPL or AFA. △

The AFAPPLICATION command options are the same as for the AF command. See "Options" on page 54 for details.

Comparison with the AF Command

The AFAPPLICATION command is similar to the AF command in the following ways:

- The same command options can be used.
- The checkpoint entry identified in SASUSER.PROFILE.AF.AFGO runs if the CATALOG= option is omitted.
- Application-specific options that you supply with the command are passed to the application for further processing via the _CMDLIST_ sublist of the local environment list.

The AFAPPLICATION command differs from the AF command in that it does not update the AF checkpoint. That is, the name of the initial entry in the application that is executed by the AFAPPLICATION command (or the name and frame of the CBT entry, if the user is exiting from a CBT entry) is not recorded in the SASUSER.PROFILE.AF.AFGO entry as it is for the AF command.

Sharing Data Between Multiple Applications

When you run a SAS/AF application with the AFAPPLICATION command, remember that other applications may be running at the same time. Those other applications may try to access the same SAS data sets or members of SAS catalogs as your application. For example, suppose your application uses the FILLIST function to read an SLIST catalog member. Before your application updates the list and writes it back to the SLIST entry, another application that has update access to the catalog can read from and write to the same SLIST entry. This situation can create problems with data integrity.

If your application needs to share data with another application, you should consider

- opening your data sets with member-level locking to prevent other applications from opening the data sets at the same time
- accessing your catalog and data files through SAS/SHARE software so that other applications can have simultaneous update access

- locking your catalog entries and other SAS data files by using the SCL LOCK function.

Passing Options to an Application

The AF and AFAPPLICATION commands can pass option values to your FRAME, PROGRAM, and SCL applications. The general form for application options is

<option-name=>option-value <... <option-name-n=>option-value-n>

For example, if you design an application that requires a list of observation numbers as input, you can invoke the application with the following AF command:

```
af c=master.apps.obs.scl obs=17 23 19 47
```

The AF task stores the specified options and their values in a special list called _CMDLIST_. This list is a sublist of the SCL local environment list that is created when a SAS/AF application is invoked.

If the option is specified in the form *name=value* (for example, **OBS=17**), then both the name and the value are stored in the list; otherwise just the value (for example, 23 or 19) is stored. In this case *value* is a number. However, it also can be one of the following:

- an unquoted text string such as **YES** or **NO**
- a quoted text string such as **'Apply SAS Software'**
- a hexadecimal string such as **'534153'x**
- a date, time, or datetime literal such as **'19Jun1991'D**.

Note: If you want several words to be treated as one argument, you must enclose them in quotes. △

You can use the list manipulation functions in SAS Component Language to extract the option values and to use them in your applications. Refer to *SAS Component Language: Reference* for information about SCL list functions.

Values That Are Automatically Placed in the Command List

The library, catalog name, entry name, and entry type values are automatically added to the command list. (If you omit the CATALOG= option in the AF command, the application name is retrieved from the SASUSER.PROFILE.AF.AFGO catalog entry.) Therefore, the command list is always at least four items long. For example, suppose you issue the following command:

```
af c=training.sas.intro.program
```

For this command, the _CMDLIST_ list contains the following values:

```
_CMDLIST_=(LIBNAME='TRAINING'
           CATALOG='SAS'
           NAME='INTRO'
           TYPE='PROGRAM'
           )
```

Differences in Storing AF Command and Application-specific Options

The rules for storing the values of application-specific options in the _CMDLIST_ list are somewhat different than for AF command options, as explained in the following table:

Values	For AF command options	For application-specific options
Options repeated in the same AF command	are ignored except for the last occurrence, which is stored in the command list	are all stored in the command list
Abbreviations specified for YES and NO values	are stored in the command list as YES and NO (the abbreviations are expanded and converted to uppercase)	are stored in the command list exactly as specified
Values specified in lowercase	are converted to uppercase unless quoted	are stored in the command list in lowercase

For example, suppose you issue the following AF command:

```
af c=a.b.c.program check=y check=n
```

For this command, the _CMDLIST_ list contains the following values:

```
_CMDLIST_=(LIBNAME='A'
           CATALOG='B'
           NAME='C'
           TYPE='PROGRAM'
           CHECKLAST='NO'
           )
```

Note: Notice that the CHECKLAST= option appears only once in the command list, reflecting the last occurrence of the CHECK= option in the AF command. (The short form of the option name is expanded to its full form.) △

However, suppose you enter the following command:

```
af c=a.b.c  name='David S.' Obs=17 23 19 term=y term=n
```

For this command, the _CMDLIST_ list contains the following values:

```
_CMDLIST_=(LIBNAME='A'
           CATALOG='B'
           NAME='C'
           TYPE='PROGRAM'
           NAME='David S.'
           OBS=17
           23
           19
           TERM='y'
           TERM='n'
           )
```

Note: The application-specific NAME= option does not conflict with the NAME= option generated by the AF command that contains the current entry name. △

Using Command Macros

If your application accepts options, you can design a command macro to invoke the application. For example, suppose you create an entry named FINANCE.REPORTS.GENRPT.SCL that accepts the following options:

TITLE="*title-text*"

DATE=*SAS-date-value*

You can create the following command macro to invoke the application:

```
%macro genrpt(title="Financial Report",date=0)/cmd;
   afapp c=finance.reports.genrpt.scl title=&title date=&date
%mend genrpt;
```

Then, users can invoke the application with the GENRPT command, provided the macro is loaded in the current SAS session and the CMDMAC system option is specified. If a user issues the **genrpt** command with no arguments, then the SCL entry is executed with the default title and date. However, a user can specify a different title and date. For example, a user can issue the following command:

```
genrpt date='24Jul1999'D title="Personnel Report"
```

The SAS macro facility changes that command into the following command:

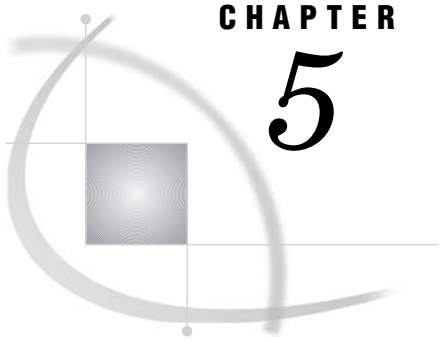
```
afapp c=finance.reports.genrpt.scl
      title="Personnel Report" date='24Jul1999'D
```

Suppressing SAS Windows When a SAS/AF Application Opens

You can use the SAS system option INITCMD to execute a SAS/AF application when a SAS session starts and to open the AF window without opening any intervening SAS windows such as the PROGRAM EDITOR, LOG, or OUTPUT windows. The INITCMD system option must either be used in conjunction with the command that starts the SAS session or be specified in the SAS configuration file.

When you invoke your application with the INITCMD system option, the SAS session automatically ends when the application ends.

Refer to *SAS Language Reference: Dictionary* for more information about the INITCMD system option.



CHAPTER

5

AF Window Commands

Overview

The following commands are available in the AF window in which SAS/AF applications execute. The behavior of the window depends on which type of catalog entry is being displayed. Some commands are not available for all catalog entry types. Exceptions are noted in the command descriptions.

Note: For FRAME and PROGRAM entries, application developers can provide additional commands that are specific to the entry.
△

Window Management	*=menu-option-list*	MUSIC
	AFSYS	PREVIEW
	CANCEL	QCAN
	CATNAME	QEND
	CLEAR	QSTACK
	DEBUG	RECALL
	DUP	RETURN|=
	END	SAVE
	FIND	SOUND
	HELP	TYPE
	ID	WREGION
	KEYFIELD	WSIZE
	LOCATE	
Scrolling	BACKWARD	LEFT
	BOTTOM	RIGHT
	FORWARD	TOP
	HSCROLL	VSCROLL
Printing	FONT	PRTFILE
	FORMNAME	SPRINT

In addition to the commands shown in this list, all of the SAS windowing environment global commands are also available in the AF window. Refer to the online Help for Base SAS software for descriptions of the global commands.

Note: Application developers can use SCL programs in FRAME and PROGRAM entries to block the execution of some or all of these commands, or to provide different processing of the commands. △

Window Management Commands

=*menu-options*
executes the entry called by the specified options for the first MENU entry in the current execution stack. To specify submenu options in the *menu-options* value, separate the options with periods.

For example, if the application window currently displays a PROGRAM entry that was opened from a MENU entry, the following command attempts to open the entry that is selected by the option value 5 in the original MENU entry:

```
=5
```

AFSYS *action*
enables you to review and manage resident entries and to control the automatic termination of FRAME objects that still exist when a FRAME entry ends. This command is useful for testing applications.

The AFSYS command uses the following *action* arguments:

AUTOTERM ON | OFF
controls whether the _term method of a FRAME entry object is automatically executed if the object still exists when a FRAME entry ends. By default, any open objects are automatically terminated (AUTOTERM ON). Use AUTOTERM OFF if you want to prevent the _term method of open FRAME objects from being executed when the FRAME entry ends.

AUTOTERM VERBOSE | NOVERBOSE>
controls whether a list of open objects is sent to the Log window. By default, no list is generated (AUTOTERM NOVERBOSE). Use AUTOTERM VERBOSE to send the object list for each object that still exists when the FRAME entry ends.

SHOW ACTIVE
sends a list of all entries that are active for the current application to the Log window.

SHOW INACTIVE
sends a list of all entries that are resident but inactive for the current application to the Log window.

RESIDENT ON | OFF
controls whether entries that have the RESIDENT attribute are retained in memory when they are inactive. The default is to retain entries (RESIDENT ON). Specify OFF to prevent inactive entries from being retained in memory even if they have the RESIDENT attribute.

PURGE *n* | ALL>
removes inactive entries from memory. Specify ALL to remove all inactive entries. Specify an entry number to remove a particular inactive entry. Use the AFSYS SHOW INACTIVE command to view the entry numbers for inactive entries.

To specify a combination of actions, use separate AFSYS commands, as shown in the following example:

```
afsys autoterm verbose; afsys autoterm on
```

CANCEL
closes the current entry and returns control to the calling entry, or to the parent entry if one was specified in the current entry's general attributes.

For FRAME and PROGRAM entries, the CANCEL command performs the following tasks before closing the entry:

- sets the SCL _STATUS_ variable to **C**
- runs the TERM section of the entry's SCL program (and, for FRAME entries, the _term method for the frame and all components).

Any pending SAS statements in the PREVIEW buffer are not submitted to the SAS session for processing.

CATNAME <CLEAR | LIST> <*libref.*>*catalog-reference* (<*libref.*>*catalog-1* ... <*libref.*>*catalog-n*)
defines a catalog reference (catref) that provides a logical combination of the specified catalogs. You can use any valid SAS name for *catalog-reference*, but if you use the name of an existing catalog you will not be able to access the contents of that catalog until the catref is cleared. The libref that you specify with the catref must already exist. Enclose the list of catalogs in parentheses, and use blanks to separate the catalog names in the list.

When a program in the SAS/AF application contains a reference to an entry in the catref catalog, the AF task searches for the entry in the specified list of catalogs, starting with *catalog-1* and ending with *catalog-n*.

For example, suppose that you issue the following command:

```
catname mylib.all (mylib.apps1 mylib.apps2 master.apps)
```

A reference in a program to MYLIB.ALL.TEST.SCL causes the AF task to search for MYLIB.APPS1.TEST.SCL, then for MYLIB.APPS2.TEST.SCL, and finally for MASTER.APPS.TEST.SCL.

Catref assignments remain in effect until they are cleared. Use the CLEAR option with the CATNAME command to clear a specified catref. Use the LIST option with the CATNAME command to list the catalogs in the catref.

CLEAR
clears values from all unprotected data entry areas of the application window.

Note: The CLEAR command is only supported for PROGRAM entries. △

DEBUG <ON | OFF>
turns the SAS Component Language source-level debugger on or off. If you use the DEBUG command without an argument, the debugger is turned on. You cannot turn the debugger off while a debugger command is active.

Note: In order to use the debugger, the SCL code in the entry must have been compiled with the DEBUG compile option turned on. △

The SCL debugger is a tool for identifying and correcting problems in SAS Component Language programs. Refer to *SAS Component Language: Reference* for information on using the SCL debugger.

DUP
copies the value stored by the SELECT command to the current field. The cursor must be positioned on the desired field when you issue the command, so the DUP command is easier to use if you assign it to a function key.

Note: The DUP command is only supported for PROGRAM entries. △

END
closes the current entry and returns control to the calling entry, or to the parent entry if one was specified in the current entry's general attributes.

For FRAME and PROGRAM entries, the END command performs the following actions before closing the entry:

- sets the SCL _STATUS_ variable to **E**
- verifies that modified fields contain valid values
- executes the MAIN section of the SCL program if any fields have been modified and contain valid values
- verifies that all required fields contain values
- executes the TERM section of the SCL program (and, for FRAME entries, the _term method for the frame and all components)
- submits the contents of the PREVIEW buffer, including any submit blocks in the TERM section.

FIND *search-string* NEXT | FIRST | LAST | PREV | ALL> <PREFIX | SUFFIX | WORD> <CASE | ICASE>
scrolls the line that contains the specified value to the top of the window for MENU and HELP entries.

For PROGRAM entries that contain extended tables, the FIND command finds an occurrence of the specified value in the field that is identified with the

KEYFIELD command. If the string contains embedded blanks, it must be enclosed in quotes.

You can modify the behavior of the FIND command by adding any one of the following options:

ALL	reports the total number of occurrences of the string in the extended table on the window's message line and moves the cursor to the first occurrence.
FIRST	moves the cursor to the first occurrence of the string in the extended table .
LAST	moves the cursor to the last occurrence of the string in the extended table.
NEXT	moves the cursor to the next occurrence of the string in the extended table.
PREV	moves the cursor to the previous occurrence of the string in the extended table.

The default behavior is NEXT.

By default, the FIND command locates any occurrence of the specified string, even where the string is embedded in other strings. You can use any one of the following options to change the command's behavior:

PREFIX	causes the search string to match the text string only when the text string occurs at the beginning of a word.
SUFFIX	causes the search string to match the text string only when the text string occurs at the end of a word.
WORD	causes the search string to match the text string only when the text string is a distinct word.

By default, the FIND command is case-sensitive. You can use the following options to change the command's behavior:

CASE	causes the search string to match the text string only if the case is the same.
ICASE	causes the search string to match the text string regardless of case.

After you issue a FIND command, you can use the RFIND command to repeat the search for the next occurrence of the string, or use the BFIND command to repeat the search for the previous occurrence.

Note: The FIND command is only supported for HELP, MENU, and PROGRAM entries. △

HELP
opens the entry specified in the current entry's Help attribute.

ID <ON | OFF>
displays the four-level name of the current entry on the window's message line for FRAME, HELP, MENU, or PROGRAM entries. For FRAME entries, the four-level name of the associated SCL entry is also displayed.

For CBT entries, the ID command shows the current frame number and frame name in the window title. Once turned on, the ID information is displayed until you turn it off. If you use the ID command without the ON or OFF option, it acts as a toggle.

Note: The ON and OFF options are only supported for CBT entries. △

KEYFIELD
identifies the field that is searched by subsequent FIND and LOCATE commands. The cursor must be positioned on the desired field when you issue the command, so the KEYFIELD command is easier to use if you assign it to a function key.

Note: The KEYFIELD command is only supported for PROGRAM entries. △

LOCATE <:> *search-string*
finds an occurrence of the specified value in the extended table field that is identified with the KEYFIELD command. If the string contains embedded blanks, special characters, or lowercase letters, then it must be enclosed in quotes.

The LOCATE command always begins matching from the first character of the field value. By default, it searches only for an exact match of the entire field value. To match only the first part of a field value, add a colon (:) before the search string.

After you issue a LOCATE command, you can use the RLOCATE command to repeat the search for the next occurrence of the string, or use the BLOCATE command to repeat the search for the previous occurrence.

Note: The LOCATE command is only supported for PROGRAM entries that display extended tables. △

MUSIC
See the SOUND command.

PREVIEW
opens the PREVIEW window, in which you can view and edit any SAS statements that are waiting to be submitted for processing when the application ends. The SAS statements are generated by SUBMIT blocks in SAS Component Language programs. Refer to *SAS Component Language: Reference* for more information on SUBMIT blocks.

QCAN
closes all open entries for the current application and returns control to the SAS session.

For FRAME and PROGRAM entries, the QCAN command performs the following steps before closing the entry:

- sets the SCL _STATUS_ variable to **C**
- runs the TERM section of the entry's SCL program (and, for FRAME entries, the _term method for the frame and all components).

Any pending SAS statements in the PREVIEW buffer are not submitted to the SAS session for processing.

Note: The difference between the QCAN command and the CANCEL command is that the QCAN command always returns control to the SAS session, not to the calling entry or the parent entry. △

QEND
closes all open entries for the current application and returns control to the SAS session.

For FRAME and PROGRAM entries, the QEND command performs the following actions before returning control to the SAS session:

- sets the SCL _STATUS_ variable to **E**
- executes the MAIN section of the SCL program if a field has been modified
- executes the TERM section of the SCL program (and, for FRAME entries, the _term method for the frame and all components)
- submits the contents of the PREVIEW buffer, including any submit blocks in the TERM section

Note: The difference between the QEND command and the END command is that the QEND command always returns control to the SAS session, not to the calling entry or the parent entry. △

QSTACK
prints a report of the application's execution stack in the Log window. The stack contains the active entry that is currently running and all the inactive entries that have been opened since you started executing the application. The listing shows two parts of the execution stack:

N-Stack	lists entries that can be viewed with the NEXT command.
S-Stack	lists all the entries in the execution stack. This stack lists interim MENU entries that have been called but are inactive and cannot be displayed with the NEXT command.

RECALL
recalls any field values that have been saved with the SAVE command (or with the SAVESCREEN function in an SCL program or with the AUTOSAVE=YES option in the AF command).

Note: The RECALL command is only supported for PROGRAM entries. △

RETURN
=
opens the entry at which the application started executing.

Note: When the RETURN command is issued in a window that was opened by a CALL GOTO routine in SCL, the result depends on the action option specified in the routine. △

SAVE
saves the current values of a PROGRAM entry's fields. You can recall the saved values later by using the RECALL command. The values are stored in an entry named *program-name*.AFPGM in your SASUSER.PROFILE catalog, where *program-name* is the name of the PROGRAM entry.

For CBT entries, the SAVE command records the current entry name and frame number, then ends the application and the SAS session. You can issue the AF command in a later SAS session to resume the CBT entry at the frame from which you issued the SAVE command.

Note: The SAVE command is only supported for CBT and PROGRAM entries. △

SOUND <ON | OFF>
MUSIC <ON | OFF>
controls whether sounds are produced for SOUND= or MUSIC= options in frames of CBT entries (provided your display device is capable of producing sounds). Use SOUND OFF if you do not want the hear the sounds that are specified in the CBT frame. If you issue a SOUND command without an ON or OFF option, it acts as a toggle, turning sound off if it was on or on if it was off.

Note: The SOUND command is only supported for CBT entries. △

TYPE <*entry-type*>
specifies the entry type that is assumed when the type is not explicitly specified in a command that must ordinarily be followed by an entry name in the form *entry-name.entry-type*. Use the TYPE command without arguments to display the current default type on the window's message line.

WREGION <*start-row start-column rows columns*>
WREGION <<TOP | BOTTOM | LEFT | RIGHT> <DEVICE>>
WREGION CLEAR
specifies the position and size of the next window of a multi-window session.

The *start-row* and *start-column* values specify the position of the upper left corner of the window, and the *rows* and *columns* values specify the window size.

Note: You can use the WSIZE command to determine the current window specifications before you issue a WREGION command. △

Alternatively, you can specify TOP, BOTTOM, LEFT, or RIGHT to cause the next window to occupy the corresponding half of the current window. For example, WREGION TOP specifies that the next window occupies the top half of the current window. Add the DEVICE option to indicate that the specification is relative to the device display rather than to the current window. For example, WREGION TOP DEVICE specifies that the next window occupies the top half of the device's display. This option is not supported in some display environments.

The CLEAR option clears any previous WREGION setting and returns the window to the size specified in the entry's general attributes. If the size is not specified in the general attributes, the window size returns to the default size.

WSIZE
displays the size and position specifications of the current window on the window's message line.

Scrolling Commands

BACKWARD <*n* | HALF | PAGE | MAX>
scrolls toward the top of the HELP, MENU, or PROGRAM entry's display text if the text contains more lines than the current window size. If you omit the scroll increment, the window is scrolled by a default increment that can be set with the VSCROLL command.

For CBT entries, the BACKWARD command displays the previous frame in the current sequence.

Note: The BACKWARD command is not supported for FRAME entries. △

BOTTOM
scrolls the window contents so that the last line of the display text appears in the window.

Note: The BOTTOM command is only supported for HELP, MENU, and PROGRAM entries. △

FORWARD <*n* | HALF | PAGE | MAX>
scrolls toward the bottom of the HELP, MENU, or PROGRAM entry's display text if the text contains more lines than the current window size. If you omit the scroll increment, the window is scrolled by a default increment that can be set with the VSCROLL command.

For CBT entries, the FORWARD command displays the next frame in the current sequence.

Note: The FORWARD command is not supported for FRAME entries. △

HSCROLL <*n* | HALF | PAGE>
sets the scroll amount for LEFT or RIGHT commands that do not specify a scroll increment. The default horizontal scroll increment is HALF.

Note: The HSCROLL command is only supported for HELP, MENU, and PROGRAM entries. △

LEFT <*n* | HALF | PAGE>
scrolls the window contents to the left by the specified increment. If you omit the scroll increment, the window is scrolled by a default increment that can be set with the HSCROLL command.

Note: The LEFT command is not supported for FRAME entries. △

RIGHT <*n* | HALF | PAGE>
scrolls the window contents to the right by the specified increment. If you omit the scroll increment, the window is scrolled by a default increment that can be set with the HSCROLL command.

Note: The RIGHT command is not supported for FRAME entries. △

TOP
scrolls the window contents so that the first line of display text appears at the top of the window.

Note: The TOP command is only supported for HELP, MENU, and PROGRAM entries. △

VSCROLL <*n* | HALF | PAGE | MAX>
sets the scroll amount for BACKWARD or FORWARD commands that do not specify a scroll increment. The default vertical scroll increment is PAGE.

Note: The VSCROLL command is only supported for HELP, MENU, and PROGRAM entries. △

Printing Commands

FONT
opens the FONT window and displays the font-control information from the current form. The listing shows which color and highlighting attributes the form interprets as signals to change printing characteristics. The FONT window is for browsing only; to change the font information you must open the FORM window and edit the FORM entry. See the description of the FORM window in Base SAS documentation for more information.

Note: The FONT window is not opened if no font information is defined in the current form. Instead, a message is displayed, indicating that the form contains no font information. △

FORMNAME <*form-name*>
FORMNAME CLEAR
specifies the default FORM entry, which contains instructions for printing images that are captured with the SPRINT command. The SAS System's default form is DEFAULT.FORM. You can use the FORMNAME command to specify a different default. See Base SAS documentation for more information about forms.

When a form is required, SAS first looks for the specified FORM entry in the current catalog. If the form is not there, SAS next looks in your personal PROFILE catalog (SASUSER.PROFILE, or WORK.PROFILE if the SASUSER library is not defined) and then, finally, in the SASHELP.FSP catalog.

Use the FORMNAME CLEAR command to resume using DEFAULT.FORM as the default FORM entry name.

Use the FORMNAME command with no arguments to display the current default form on the window's message line.

PRTFILE <*fileref* | '*actual-filename*' <APPEND | REPLACE>>
PRTFILE CLEAR

specifies a file to which output from the SPRINT command is sent. By default, output is sent to the printer destination that is specified in the current form. You can use the PRTFILE command to route the output to a file instead.

To identify the target file, you can use either a previously assigned fileref or the actual filename. If you specify a filename, it must be enclosed in quotes.

With the filename or fileref, you can also specify either the APPEND or REPLACE option to determine how output is handled when the file already exists. The default is REPLACE, which causes output that is sent to an existing file to overwrite the current contents of the file. To append the new output to any existing contents, use the APPEND option instead.

Use the PRTFILE CLEAR command to cancel the previous PRTFILE command and to route output to the printer again.

Use the PRTFILE command with no arguments to display the name of the current print file on the window's message line.

SPRINT <FILE=*fileref* | '*actual-filename*'> <FORM=*form-name*> <NOBORDER>
SPRINT FREE

captures the contents of the current window (except for the command and message lines). Unless you use the NOBORDER option, the window borders are also included in the capture.

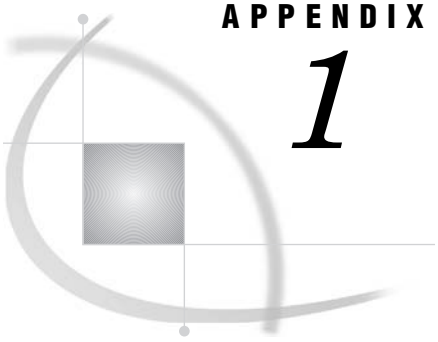
Output characteristics are determined by an associated FORM entry. You can use a FORMNAME command before issuing a SPRINT command to change the default FORM entry for all captures. Use the FORM= option with the SPRINT command to select a form other than the default for an individual capture.

By default, output is sent to the printer destination that is specified in the associated form. Use the PRTFILE command before issuing a SPRINT command if you want to send all output to a file instead of to the printer. To route an individual capture to a file instead of to the printer, or to route an individual capture to a file other than the one specified in the PRTFILE command, use the FILE= option with the SPRINT command. Changing output destinations within an application automatically frees the previous print file or print queue.

Note: Once you have sent SPRINT output to a file, any additional output that you send to that file must use the same FORM entry. △

The print queue or print file that the SPRINT command uses is freed when you end the application from which you captured contents. To free the print queue or print file before ending the application, use the SPRINT FREE command.

You can use the SPRINT command to capture information from several windows in a single print file or print queue. In this case, the print file or print queue is not freed until you end the application that sent output to the file or queue.



APPENDIX

1

Recommended Reading

Recommended Reading

Here is the recommended reading list for this title:

- *SAS Language Reference: Concepts*
- *SAS Language Reference: Dictionary*
- *Base SAS Procedures Guide*
- *SAS Component Language: Reference*
- *SAS Guide to Applications Development*

For a complete list of SAS publications, go to **support.sas.com/bookstore**. If you have questions about which titles you need, please contact a SAS Publishing Sales Representative at:

SAS Publishing Sales
SAS Campus Drive
Cary, NC 27513
Telephone: 1-800-727-3228
Fax: 1-919-531-9439
E-mail: **sasbook@sas.com**
Web address: **support.sas.com/bookstore**

Customers outside the United States and Canada, please contact your local SAS office for assistance.

Glossary

answer field
in CBT entries, the space in which users respond to either fill-in-the-blank or multiple-choice questions.

application workspace (AWS)
a window that contains other windows (child windows) or from which other windows can be invoked, but which is not contained within any parent window that is part of the same software application. See also child window.

argument
in syntax descriptions, any word that follows the keyword in a SAS statement. See also parameter.

attributes
the characteristics that describe and control the appearance and function of windows and window elements. See also field attributes, general attributes.

branching
the process of calling another program or window. CBT entries support two types of branching: conditional and unconditional. See also conditional branching, unconditional branching.

button
a component of a graphical user interface. The button is programmed to execute a command, to open a window, or to perform some other function when a user selects it.

catalog entry
See SAS catalog entry.

checkpoint
a SAS catalog entry that identifies the first or last entry from the last SAS/AF application that was executed with the AF command. See also SAS catalog entry.

child entry
in CBT entries, a secondary, or sublevel, CBT entry that is displayed when a user completes the current CBT entry. See also parent entry.

choice group
in PROGRAM entries, fields that have the same Choice group attribute. These fields, called stations, display values from which users can make a single selection.

class
in object-oriented programming, the template or model for an object. A class includes data that describes the object's characteristics (instance variables) and the operations (methods) that the object can perform.

comment indicator
in CBT entries, an asterisk that marks the beginning of commented text. The asterisk can be placed either on the frame indicator line or on the feedback indicator line.

conditional branching
in CBT entries, branching in which a response from a user determines which program or window is called. See also branching, unconditional branching.

critical success factor (CSF)
a graphical object that shows the relative position of a specific numeric value within a range of values. Many attributes of the object, such as its general shape, the type of arrow indicator, and how or whether the range is segmented, can be customized.

delimiter line
in SAS/AF PROGRAM entries, a line that contains logical NOT signs in columns 1-3. This line separates the scrollable area from the fixed, nonscrollable area in the PROGRAM entry's DISPLAY window. The nonscrollable area appears above the delimiter line. SAS/AF software recognizes the caret (^) and the negation character (¬) as NOT signs.

display
the area of a computer monitor in which the graphical user interface of a software application is visible to a user of that application.

entry type
a characteristic of a SAS catalog entry that identifies the catalog entry's structure and attributes to SAS. When you create an entry, SAS automatically assigns the entry type as part of the name.

execution stack
a last-in, first-out stack that lists the current and inactive entries that were called during the execution of a SAS/AF, FSEDIT, or FSVIEW application.

extended table
a window (in a PROGRAM entry) or a window element (in a FRAME entry) that displays values in a tabular format by repeating a set of fields (or other objects, in a FRAME entry). The number of rows that are displayed is determined by the SAS Component Language program or by an attribute of the extended table object in the FRAME entry. Extended tables are either static or dynamic. For static extended tables, the number of rows is fixed. For dynamic extended tables, the number of rows can vary.

feedback indicator line
in CBT entries, a line that the software displays on which the user enters the answer to a question. This line is designated with a pound sign (#) in column 1 that indicates feedback.

field
a window area in which users can view, enter, or modify a value.

field attributes
a set of characteristics that describe and control how the contents of a field are treated when values are entered or displayed in the field.

field validation
the process of checking user-entered values either against attributes that have been specified for a field or against conditions that have been specified in a SAS Component Language program.

frame
a nonscrollable area in a window. Programmers create frames in windows by using boundary markers. If there are no boundary markers, users execute scroll commands to see all of the contents of a window when those contents are larger than the window itself.

frame indicator line
in CBT entries, a text line that separates frames. This line is indicated either by a question mark in column 1 or by a row of dashes across the width of the DISPLAY window.

general attributes
the set of characteristics that are assigned to a SAS catalog entry (such as a PROGRAM entry or a FRAME entry) that displays a window.

global command
a command that is valid in all windows for a particular SAS software product.

KEYS entry
a type of catalog entry that contains function key settings for interactive windowing procedures.

libref (library reference)
a name that is temporarily associated with a SAS data library. The complete name of a SAS file consists of two words, separated by a period. The libref, which is the first word, indicates the library. The second word is the name of the specific SAS file. For example, in VLIB.NEWBDAY, the libref VLIB tells SAS which library contains the file NEWBDAY. You assign a libref with a LIBNAME statement or with an operating system command.

line continuation indicator
in CBT entries, a forward slash that is positioned as the last nonblank character on the frame indicator line or on the feedback indicator line. The line continuation indicator appends the following line to the line on which the indicator appears.

linked action field
in PROGRAM entries, a field with the Type attribute ACTION, which is associated with a station of a choice group. Linked action fields enable users to make a choice by selecting either the field itself or the associated station. See also check box, choice group.

linking
in MENU entries, the process of associating a primary menu with a series of secondary MENU entries so that users can access submenu choices directly from the primary menu.

logical row
a row of fields or objects that is repeated in an extended table. See also extended table.

lookup data set
a data set that contains information that is used for additional validation of input values in fields that can use a lookup data set.

member
a SAS file in a SAS library.

member name
a name that is assigned to a SAS file in a SAS library. See also member type.

member type
a SAS name that identifies the type of information that is stored in a SAS file. Member types include ACCESS, DATA, CATALOG, ITEMSTOR, MDDB, PROGRAM, and VIEW.

member-level access
a type of access to a SAS data library that permits only one user to use a member (such as a SAS data set) at a time. See also record-level access.

menu
a window object that presents choices to users. In SAS software, menus include menu bars, pull-down menus, block menus, and selection lists.

menu bar
the primary list of items at the top of a window, which represent the actions or classes of actions that can be executed. Selecting an item executes an action, opens a pull-down menu, or opens a dialog box that requests additional information. See also pop-up menu, pull-down menu.

pad character
a special character that is displayed for each character of an empty input field. By default, the pad character is an underscore (_).

parameter
(1) in SAS/AF and SAS/FSP applications, a window characteristic that can be controlled by the user. (2) in SAS Component Language (SCL), a value that is passed from one entry in an application to another. For example, in SAS/AF applications, parameters are passed between entries by using the CALL DISPLAY and ENTRY statements. (3) a unit of command syntax other than the keyword. For example, NAME=, TYPE=, and COLOR= are typical command parameters that can be either optional or required.

parent entry
the CBT entry that called the current CBT entry.

pause indicator
in CBT entries, a line that indicates that a frame should pause. Additional information is displayed when a user presses the ENTER key. This line is designated by three at signs (@) in columns 1-3.

PMENU facility
a menuing system in SAS that is used instead of the command line as a way to execute commands. The PMENU facility consists of a menu bar, pull-down menus, and dialog boxes.

pop-up menu
a menu that appears when it is requested. These menus are context-specific, depending on which window is active and on the cursor location. See also pull-down menu.

protected field
a field to which users cannot tab and in which they cannot alter values.

pull-down menu
the list of menu items or choices that appears when you choose an item from a menu bar or from another menu. See also PMENU facility.

record-level access
a type of access to a SAS data set or other file that permits more than one user to access the SAS data set or file at a time. Only one user can use a single observation or record of the file at a time, but other users can access other observations or records in the same file. See also member-level access.

SAS catalog
a SAS file that stores many different kinds of information in smaller units called catalog entries. A single SAS catalog can contain several different types of catalog entries. See also SAS catalog entry.

SAS catalog entry
a separate storage unit within a SAS catalog. Each entry has an entry type that identifies its purpose to SAS. Some catalog entries contain system information such as key definitions. Other catalog entries contain application information such as window definitions, Help windows, formats, informats, macros, or graphics output. See also entry type.

SAS Component Language (SCL)
See SCL (SAS Component Language).

SCL (SAS Component Language)
a programming language that is provided with SAS/AF and SAS/FSP software. You can use SCL for developing interactive applications that manipulate SAS data sets and external files; for displaying tables, menus, and selection lists; for generating SAS source code and submitting it to SAS for execution; and for generating code for execution by the host command processor.

selection list
a list of items in a window, from which users can make one or more selections. Sources for selection lists are LIST entries, special SCL functions, and extended tables.

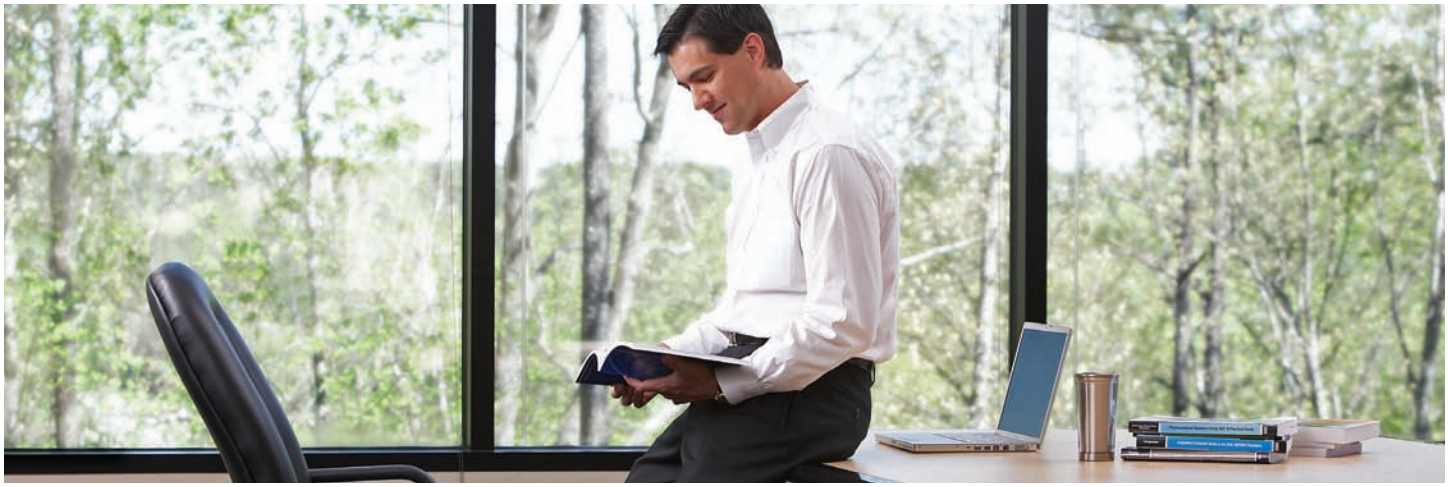
station
a field in a choice group. Only one station can be active at a particular time. See also choice group.

stream
a series of information about the entries and windows that are created when a user invokes a SAS/AF, FSEDIT, or FSVIEW application. Within the stream, SAS creates stacks to keep track of which entries have been called and which windows are currently open. Stacks use a last-in, first-out hierarchy.

tracking data set
a special data set that is created to save information about user responses to questions in CBT entries.

unconditional branching
in CBT entries, unconditionally changing the flow of control to a statement that does not immediately follow the current statement. See also branching, conditional branching.

SAS® Publishing Delivers!



SAS Publishing provides you with a wide range of resources to help you develop your SAS software expertise.
Visit us online at **support.sas.com/bookstore**.

SAS® PRESS

SAS Press titles deliver expert advice from SAS® users worldwide. Written by experienced SAS professionals, SAS Press books deliver real-world insights on a broad range of topics for all skill levels.

support.sas.com/saspress

SAS® DOCUMENTATION

We produce a full range of primary documentation:

- Online help built into the software
- Tutorials integrated into the product
- Reference documentation delivered in HTML and PDF formats—free on the Web
- Hard-copy books

support.sas.com/documentation

SAS® PUBLISHING NEWS

Subscribe to SAS Publishing News to receive up-to-date information via e-mail about all new SAS titles, product news, special offers and promotions, and Web site features.

support.sas.com/spn

SOCIAL MEDIA: JOIN THE CONVERSATION!

Connect with SAS Publishing through social media. Visit our Web site for links to our pages on Facebook, Twitter, and LinkedIn. Learn about our blogs, author podcasts, and RSS feeds, too.

support.sas.com/socialmedia

sas® | THE POWER TO KNOW®