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 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 System windows. The following sections describe the syntax and usage of these commands. Refer to *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 System 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.

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

SAS/AF® Software Procedure Guide, Version 8

ISBN 1–58025–516–7

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

1st printing, October 1999