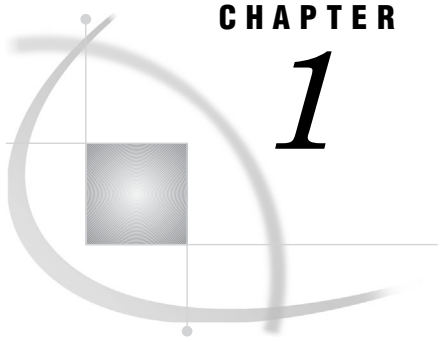


CHAPTER

1

Introduction

Introduction to SCL

SAS Component Language (SCL) is a programming language designed to facilitate the development of interactive applications using the SAS System. For example, you can use SCL with other SAS software to create data entry applications, to display tables and menus, and to generate and submit SAS source code. SCL is the scripting language behind SAS/AF, SAS/FSP, and SAS/EIS software. Applications developed using SCL with SAS/AF software can be run by users who have only licensed Base SAS software. Previous to Version 7, SAS Component Language was known as SAS Screen Control Language.

SCL Elements

SAS Component Language has statements, functions, CALL routines, operators, expressions, and variables—elements that are common to the Base SAS language and to many other programming languages. If you have experience writing DATA or PROC steps in the Base SAS language, the basic elements of SCL are familiar to you. For example, in SCL programs, you can use DO loops, IF-THEN/ELSE statements, comparison operators such as EQ and LT, and SAS macros.

SCL provides additional statements, functions, and other features that make it easy to create object-oriented, interactive applications. For example, SCL provides CLASS and INTERFACE statements that enable you to define classes and interfaces to those classes. You can use dot notation to access an object's attributes directly and to invoke methods instead of using the SEND and NOTIFY routines.

Chapter 2, “The Structure of SCL Programs,” on page 9 describes the organization of an SCL application. Chapter 3, “SCL Fundamentals,” on page 17 describes the basic elements of SCL and the rules that you must follow to put these elements together in a program. Chapter 6, “Controlling Program Flow,” on page 69 describes the statements and other features that you can use to control the flow of your application.

Entering SCL Code

You can enter your SCL code in four ways:

- In the SAS Explorer, select the catalog where you want to store your SCL code, and select **File ► New... ► SCL Program**
- Issue the BUILD command followed by the name of the entry that you want to edit:

  BUILD *libref.catalog.entry*.SCL
- In the Build window for the frame, select **View ► Frame SCL**
- In the Build window for the program screen, select**Tools ► Source Window**

Compiling SCL Programs

You must compile SCL programs before testing or executing an application. The SCL compiler translates your SCL application into an intermediate form that can be executed by the SCL interpreter. In the process of translating your application, the compiler identifies any syntax errors that are present. You should correct these errors and compile the code again until the program compiles without errors. If there are errors in the program, then no intermediate code is produced, and running the entry produces a message stating that the entry has no code.

The compiler produces a list of errors, warnings, and notes in the Log window. A program that compiles with no errors or warnings produces a message like the following in the message line of the Source window:

```
NOTE:
Code generated for SALES.FRAME. Code size=2649.
```

Compiling Your SCL Program Interactively

Note: You must save frames before you can compile them. △

You can compile frames, program screens, and SCL programs by issuing the COMPILE command in the Build window or by selecting **`Compile`** from the pull-down menus. In the Build window for frames or SCL programs, select **Build ► Compile**In the Build window for program screens, select**Run ► Compile**

You can also compile FRAME, PROGRAM, and SCL entries from the SAS Explorer. If your SCL code is associated with a frame or a program screen, then you must compile the FRAME or PROGRAM entry in order for the associated code to be compiled correctly. The SCL code is compiled when you compile the FRAME or PROGRAM entry. If your SCL code is not associated with a FRAME or PROGRAM entry, then you compile the SCL entry. To compile an entry from the Explorer, select the entry, then select **`Compile`** from the pop-up menu.

If you compile a FRAME or PROGRAM entry that does not have any associated SCL code, the SCL compiler displays an error message.

Compiling Your SCL Program in Batch

To compile your SCL application in batch, run PROC BUILD with the COMPILE option:

PROC BUILD CATALOG=*libref.catalog* BATCH;

COMPILE <ENTRYTYPE=*entry-type*>;

RUN;

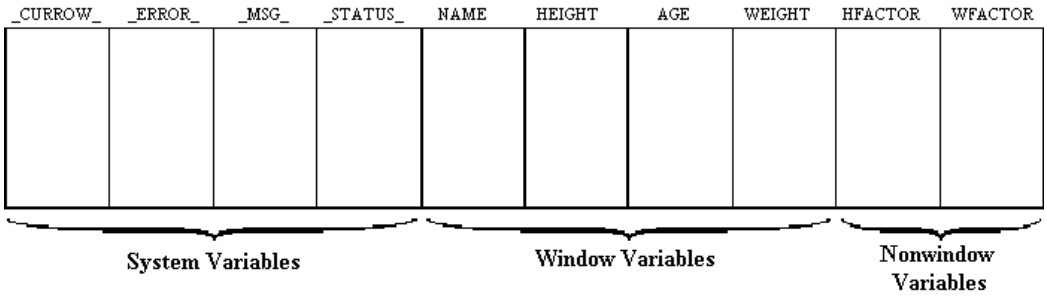
The default entry type is PROGRAM. If you do not specify an *entry-type*, then all PROGRAM, FRAME, and SCL entries in the catalog are compiled. All PROGRAM, FRAME, and SCL entries in an application must be compiled before the application can be executed.

If you compile a frame in batch, and the compiler cannot find the associated SCL source, the frame will not compile.

The SCL Data Vector

Compiling an SCL program opens a temporary storage area called the SCL data vector (SDV). The SDV holds the values of all window, nonwindow, and system variables that are used in the application.* Areas in the SDV are created automatically for system variables as well as for window variables for all controls in the application window, even if the variables are not used in the SCL program. Figure 1.1 on page 5 shows the SDV for an application that uses the window variables NAME, HEIGHT, AGE, and WEIGHT and the nonwindow variables HFACTOR and WFACTOR.

Figure 1.1 SCL Data Vector for a Typical Application



Testing SCL Applications

After an SCL program compiles successfully, you can test it. For catalog entries that use the TESTAF, AF, or AFA command, you can issue the command from either the Source window or the Display window. You must use the AF or AFA command if a program contains SUBMIT statements, or if you are testing a FRAME entry that is in a library accessed with SAS/SHARE software. It is recommended that you issue the SAVE command before issuing the AF or AFA command.

* The SDV is similar to the program data vector (PDV) that is created by base SAS software when it compiles a DATA step. The program data vector holds the values of the variables that are defined in the DATA step.

SCL programs for FSEDIT and FSVIEW applications run when you return to the FSEDIT or FSVIEW window.

Debugging SCL Programs

SCL includes an interactive debugger for monitoring the execution of SCL programs. The debugger enables you to locate logical errors while a program is executing. To use the debugger, issue the DEBUG ON command before compiling. Then either use the TESTAF command or use the AF or AFA command with the DEBUG=YES option to run the entry under the debugger. To deactivate the SCL debugger, issue the DEBUG OFF command followed by the COMPILE command. For more information about the SCL debugger, see Chapter 14, "The SCL Debugger," on page 753.

Saving SCL Programs

After you create an SCL program, use the SAVE command to save it. You should also use SAVE /ALL to save the entry before issuing a TESTAF command if the entry uses CALL DISPLAY to call itself. Also be sure to save all other open entries before issuing a TESTAF command so that CALL DISPLAY and method calls will execute the most recent versions of your SAS/AF entries.

Optimizing the Performance of SCL Code

You can optimize the performance of SCL programs in your applications with the SCL analysis tools.

The following table lists the available tools and provides information about when you might want to use each tool.

Use the...	*when you want to...*
Coverage Analyzer	monitor an executing application and access a report that shows which lines of SCL code are not executed during testing.
List Diagnostic Utility	monitor an executing application and access reports on the use of SCL lists, including any lists that are left undeleted.
Performance Analyzer	monitor an executing application and identify any bottlenecks in the application's SCL code.
Static Analyzer	access reports that detail SCL code complexity; variable, function, method and I/O usage; and error and warning messages.

To display a menu of the SCL analysis tools, enter **`sclprof`** at the command prompt. For detailed information about using these tools, see the SAS/AF online Help.

Using Other SAS Software Features in SCL Programs

SCL supports most of the features of the Base SAS language. Some Base SAS features are directly supported by SCL. Other Base SAS features have equivalent features in SCL, although there may be small differences in functionality. For example, the IN operator in SCL, returns the index of the element if a match is found rather than a 1 (true).

Chapter 13, "SAS Component Language Dictionary," on page 199 provides entries for elements that have different functionality in SCL programs. The differences are summarized in "Using SAS DATA Step Features in SCL Programs" on page 82.

Although SCL does not provide an equivalent for every command that is available under your operating system, it does provide features that interact directly with SAS software and with host operating systems. For example, you can use the SUBMIT statement to access other features of SAS software, and you can use the SYSTEM function to issue host operating system commands. SCL also supports the SAS macro facility. For more information about these features, see Chapter 7, "Using SCL with Other SAS Software Products," on page 81.

SCL Compatibility Issues

The following are notable compatibility issues between Version 8 (and later) and earlier versions or releases.

ACCESS routine
: The functionality of ACCESS is available through the CALL BUILD routine. Both ACCESS and BUILD now open the Explorer window. Old programs that use ACCESS will still work, although the TYPE and MODE parameters are not supported from the Explorer window. Using BUILD for new programs is recommended because it provides additional functionality.

CALC routine
: The CALL CALC routine is not supported in Version 7.

CARDS statement
: The DATA step statement DATALINES replaces the CARDS statement.

CATALOG function
: The functionality of CATALOG is now available through the CALL BUILD routine. Like CATALOG, BUILD opens the Explorer window when a catalog is specified as the first parameter. Old programs with the CATALOG function will still run, although the SHOWTYPE and PMENU parameters are not supported from the Explorer window. Using BUILD for new programs is recommended because it provides additional functionality.

CATLIST, DIRLIST, and FILELIST, and IVARLIST functions
: Version 7 tables may contain mixed-case names. If any existing application relies on one of these functions to return an uppercased name, you may need to modify the application. See Chapter 13, "SAS Component Language Dictionary," on page 199 for more information.

CATLIST, DIRLIST, FILELIST, and LIBLIST windows
: These windows have been replaced with host selector windows in which the AUTOCLOSE option will now be ignored.