# SAS/AF: Running SCL Outside the Frame

A. Darrell Massengill

## INTRODUCTION

SCL is a powerful programming language that has been part of SAS for many years. People use SCL for many reasons; some people like to use the power of SCL instead of the SAS data step language. Others like the security that SCL gives their code. Many companies have developed large applications and business models using SCL, while other companies develop small utility programs. Whatever the reason for using it, there is a lot of SCL code in use. As you move into the Business Intelligence world or into the world of web applications it is possible to continue to use your SCL programs through a SAS stored process.

This paper gives a very brief overview of SCL and discusses creating batch SCL programs. A batch SCL program can be easily moved into a stored process. The paper covers stored processes and the steps necessary to run your SCL programs as a stored process. Note that this paper cannot cover all of the details of SCL or Stored Processes, but is meant as a basic introduction. References to more detailed information are listed.

## SCL OVERVIEW

Most SAS users have encountered SCL code at some time. Below is a brief reintroduction to SCL.

### WHAT IS SCL?

SCL (SAS Component Language) is a programming language available with SAS/AF. SCL programs are created through the SAS/AF interface and stored in a catalog in .SCL entries. SAS Component Language has statements, functions, CALL routines, operators, expressions, and variables--elements that are common to the Base SAS language and 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.

SAS/AF is required to create and modify an SCL program, but not to run the program. This allows you to create the program on one machine and deploy the catalog on other machines that do not license SAS/AF. The source code can be removed from the catalogs that are deployed so that it cannot be modified; the code remains private and secure.

Additional information can be found in the SAS/AF documentation. See the References section below.

### I THOUGHT SAS/AF WAS DEAD

To quote Mark Twain, “The rumors of my death have been greatly exaggerated”. Several years ago, SAS did encourage users to move toward the newer Java development. However, SAS does recognize that there are many customers that want and need the tools and environment of SAS/AF. There are many users that have used SAS/AF for many years, as well as new users who are just starting to use it. SAS is committed to SAS/AF and this is demonstrated by the fact that SAS 9.2 will have new SAS/AF enhancements. SAS/AF is also back on the SASware ballot. In addition, a new SAS/AF book has been published.

For further information, see Lynn Curley’s paper listed in the Reference Section.

### WHY WOULD I USE SCL?

SCL code is used for many reasons. Some of the key reasons are:

- There are many applications out there already written in SCL.
- It is an easy to use language with a rich set of functions and functionality.
- Programs can run without the source code, offering code security.
- A SAS/AF license is not needed to run the programs although a license is needed to create the programs.
- There is a built-in debugger.

## USING SCL IN BATCH

SCL programs can be run in batch in the same way that other SAS programs can. Batch processing is a convenient way of running repeated processes that crunch numbers, generate reports or update data. The SCL language supports both interactive and non-interactive functionality. One consideration for the creation a batch SCL program is to to avoid the interactive and visual functionality. Another consideration is having the ability to pass information into and and out of an SCL batch program. This makes your SCL code more flexible and able to interface with other programs.

This paper will only give a high level overview of creating a batch program. For more details, see Michael Davis' paper in the References section.

### AVOIDING THE VISUAL

In order to run an SCL program in batch, avoid the visual functionality. This means avoiding the functions that relate to the screen programming such as SAS/FSP, Windows, Cursors, Keys and other similar items. Michael Davis' paper gives a detailed list of functionality to avoid.

### PASSING INFORMATION TO SCL

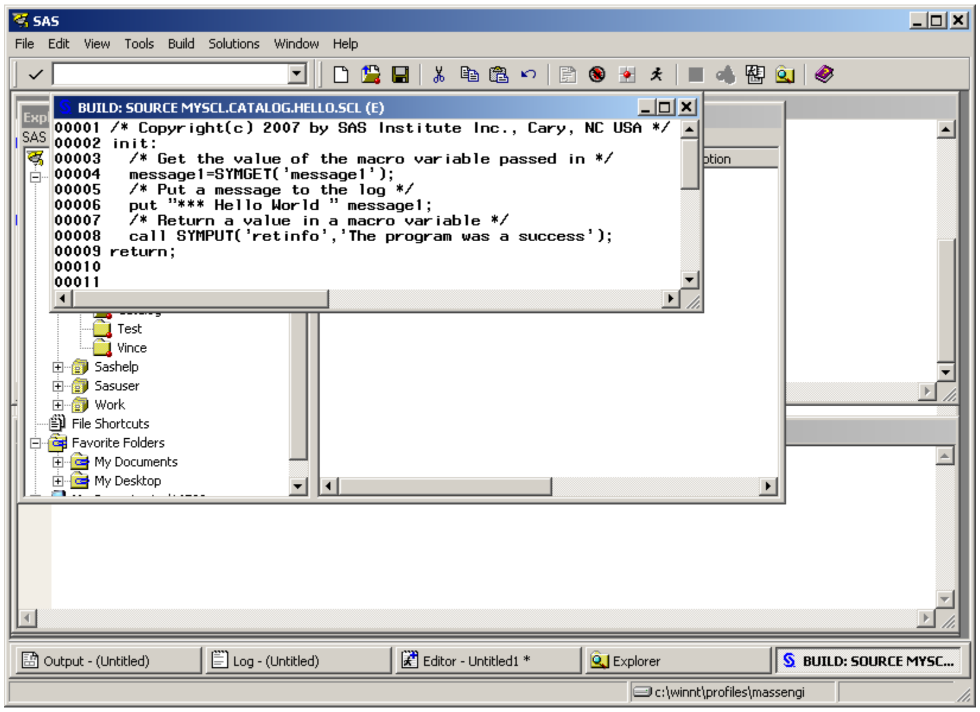
It is often convenient to pass in information to your SCL code so your program is more generic. For example, you may want to pass in the name of a dataset so that the SCL can process different data without you needing to modify the program. One easy way to do this is to pass information in through macro variables and then have the SCL code examine those macro variables. The SYMGET function can be used in SCL to get the value of the macro variable. You can also use macro variables to return information from the SCL program. The SYMPUT function can be used to set the returned information. The example program below uses both SYMGET and SYMPUT.

### CREATING THE PROGRAM

This section will take you through the process of creating a simple batch SCL program.

1. Create the SCL code.
   - Before beginning, submit a LIBNAME statement to the location you will create your SCL catalog. For example: LIBNAME myscl 'c:\myscl';
   - Open an explorer window by typing the following in the command window: build myscl.catalog.
   - Open an empty SCL build window by typing the following in the command window: build myscl.catalog.hello.scl.
   - Enter the following text in the SCL build window (see Figure 1):

```
init:
   /* Get the value of the macro variable passed in */
   message1=SYMGET('message1');
   /* Put a message to the log */
   put "*** Hello World " message1;
   /* Return a value in a macro variable */
   call SYMPUT('retinfo','The program was a success');
return;
```



```
/* Copyright(c) 2007 by SAS Institute Inc., Cary, NC USA */
init:
  /* Get the value of the macro variable passed in */
  message1=SYMGET('message1');
  /* Put a message to the log */
  put "*** Hello World " message1;
  /* Return a value in a macro variable */
  call SYMPUT('retinfo','The program was a success');
return;
```

**Figure 1. Creating an SCL Program**

- Close the window and save the SCL Program.
- In the Explorer window, right click on the program name and select ‘compile’ (see Figure 2). Be sure to check for errors in the SAS log. Correct any errors and compile until it compiles cleanly. Then close the Explorer window.

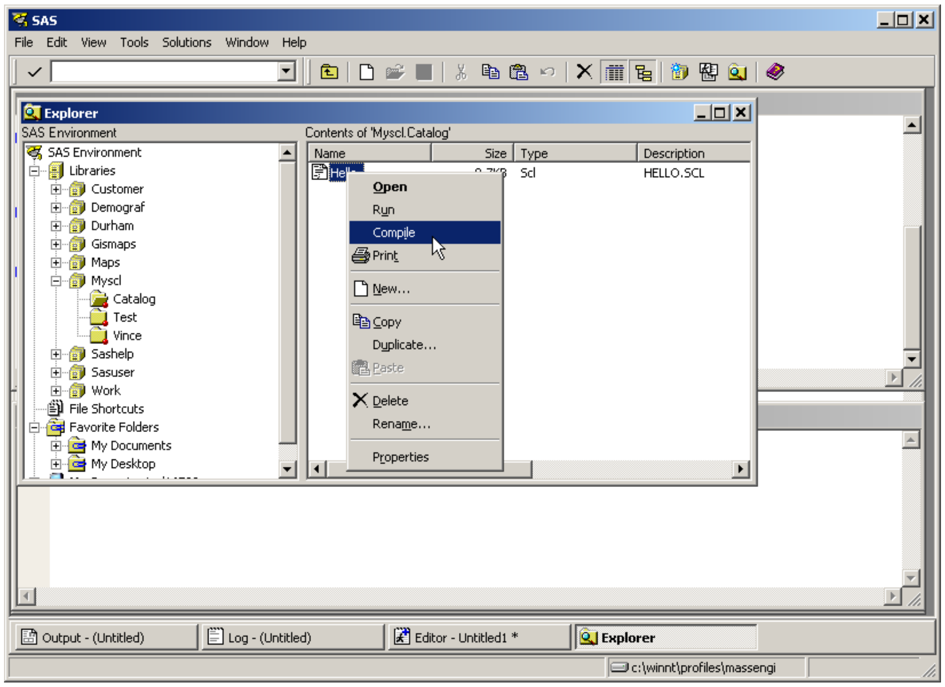


**Figure 2. Compiling the SCL**

2. Create a SAS program to run the SCL.
   - Use DMS, Enterprise Guide or your favorite text editor and create runbatch.sas with the following text:

```
Libname myscl ‘c:\myscl’; /*modify according to your location*/
%let message1=My program ran;
PROC DISPLAY c=myscl.catalog.hello.scl; RUN;
%put &retinfo;
```

   - Save the program to c:\runbatch.sas

3. Execute the program and check the results.
   - Run runbatch.sas as a batch program. Performing this step depends on how you run batch programs on your system.
   - Examine the output in your log. Notice that your value of your input macro variable was displayed and that your returned value was returned correctly. You should see the following:

```
1    libname myscl ‘c:\myscl’;
2    %let message1=My program ran;
3    proc display c=myscl.catalog.hello.scl; run;

*** Hello World  My program ran
NOTE: PROCEDURE DISPLAY used (Total process time):
      real time           x.xx seconds
      cpu time            x.xx seconds


4    %put &retinfo;
The program was a success
```

## RUNNING SCL AS A STORED PROCESS

SCL can be run as a stored process much like it is run as a batch program. The program created above can be used as a stored process. Before running this program as such, we will examine what constitutes a stored process.

## OVERVIEW OF STORED PROCESSES

A stored process is a SAS program that is stored centrally on a server. Most SAS programs can be a stored process. Stored processes enable you to centrally maintain and manage code, give you control over changes, enhance security and application integrity, and ensure that the latest version of code is available for all applications.

Stored processes are like other SAS programs except they have an additional feature that enables customization of the program's execution. This feature enables the invoking application to supply parameters at the time that the stored process is invoked. For example, if you have a stored process that analyzes monthly sales data, you could create a MONTH variable in the stored process. At execution time, the user could supply the parameter as MONTH=MAY to analyze May sales data.

You can use stored processes for web reporting, analytics, building web applications, delivering packages to clients or the middle tier, and publishing results to channels or repositories. Stored processes can access any SAS data source or external file and can create new data sets, files, or other data targets supported by the SAS System. Stored processes can be run from wide variety of applications including Web Report Studio and Enterprise Guide. Client applications can be created with Java or Microsoft Windows technology.

Converting a SAS program to a stored process is very straight forward. Converting a batch SCL program to a stored process is equally easy. Once the batch SCL program is created, we must create a SAS program that runs your SCL code through Proc DISPLAY Below, we go through the steps to convert our earlier batch SCL program into a stored process. Then we will repeat the process with a generic SAS program named STPRUNEN.

For additional information on stored processes, see the White Paper on SAS Stored Processes and Heather Weinstein’s paper listed in the References section below.

### CREATING A STORED PROCESS

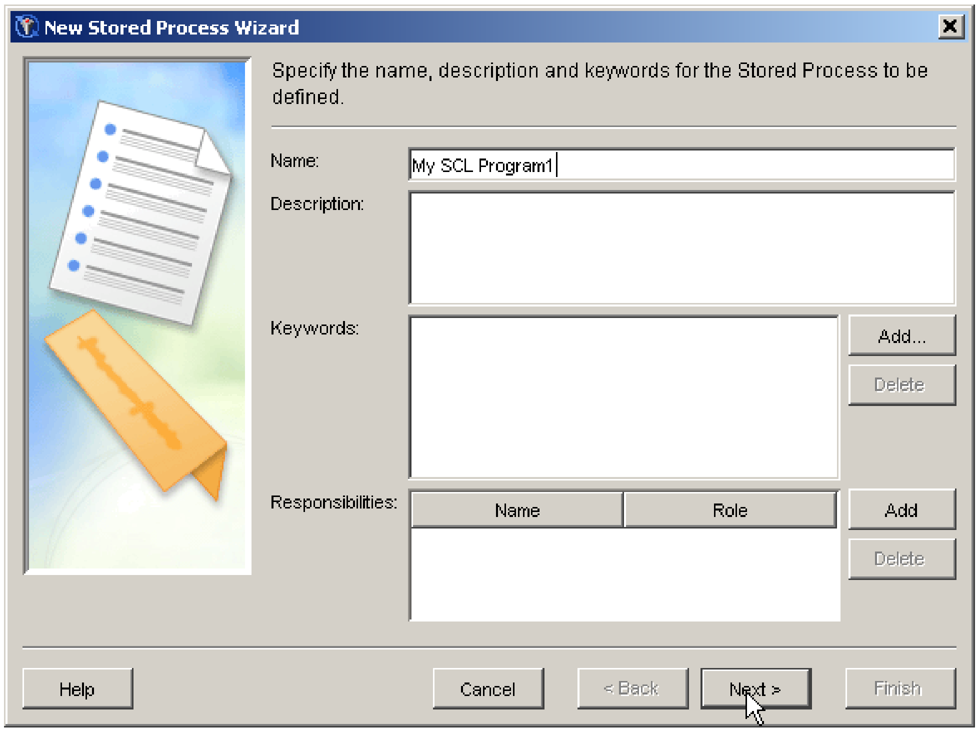
A stored process consists of two parts: physical source code and metadata. You must create the SAS program source and create metadata by registering the program. The source code is stored on the Stored Process server. The metadata is stored on the SAS Metadata server. We have already created our SAS program in the steps above, but we must get it ready to be a stored process. After that is done, we can register the program.

**Prepare the SAS program**

1. The SAS program must be copied to a location on the stored process server. For example: C:\MyStoredProcesses. Copy the runbatch.sas program that we created earlier. You may want to copy the SCL catalog to this same location.
2. Modify the program as needed.
    a. Make sure the program has a LIBNAME statement that points to the location of the SCL catalog.
    b. If the program contains ODS statements, further changes are needed. See Heather Weinstein’s paper for additional information in the Reference section below,

**Register the Stored Process in SAS Management Console**

1. Run SAS Management Console.
2. Select BI Manager in the navigation tree on the left and expand it.
3. Either select an existing folder or create a new folder by Actions-> New Folder
4. Run the Stored Process Wizard by selection Actions -> New Stored Process
    a. Enter the following information in the first screen of the wizard (see Figure 3) and click NEXT.


New Stored Process Wizard

Specify the name, description and keywords for the Stored Process to be defined.

Name: My SCL Program1

Description:

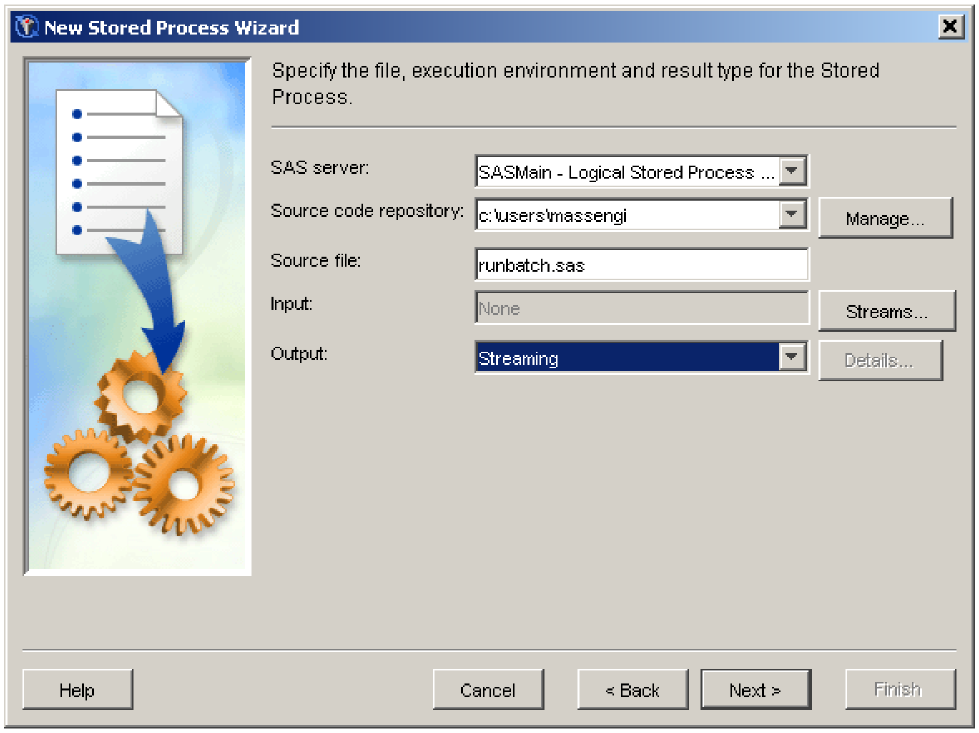
Keywords:

Responsibilities:

| Name | Role |
|---|---|

**Figure 3. Name the stored process**

b. On the next screen, enter the following information (see Figure 4) and click NEXT.
   **SAS server:** SASMain – Logical Stored Process Server
   **Source code repository:** C:\users\massengi (use the directory with your program)
   **Source file:** runbatch.sas
   **Output:** Streaming


New Stored Process Wizard

Specify the file, execution environment and result type for the Stored Process.

SAS server: SASMain - Logical Stored Process ...

Source code repository: c:\users\massengi

Source file: runbatch.sas

Input: None

Output: Streaming

**Figure 4. Specifying the server and program information**

c. The next screen is the Parameters screen (see Figure 5). Do not add any parameters for this program. Click FINISH.

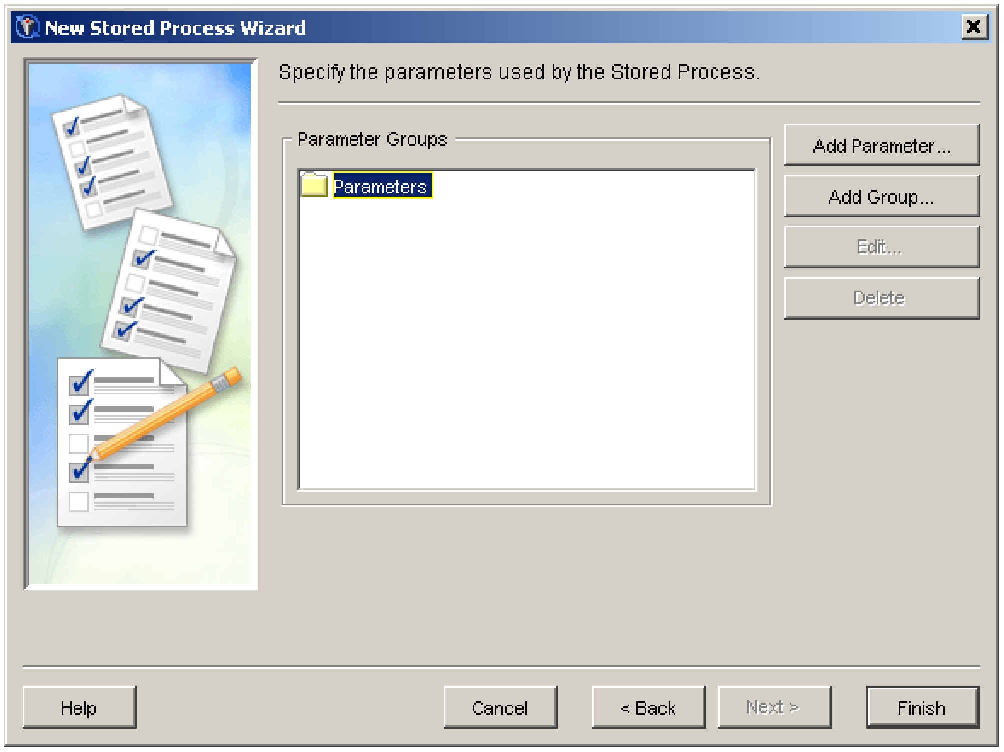
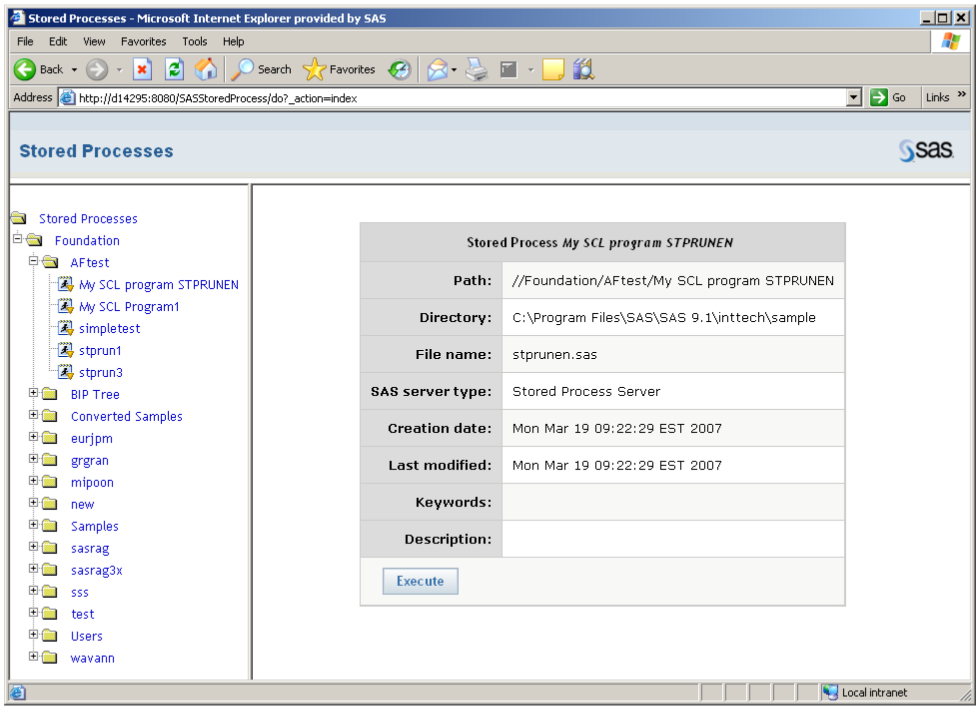


**Figure 5. Parameters screen**

d. You have now completed registering the stored process.

**Executing the Stored Process**
Stored Processes can be executed from many different SAS products, including Enterprise Guide, Web Report Studio and SAS Add-In for Microsoft Office. For this example, we will use the Stored Process Web Application Property Sheet (see Figure 6). Run the following URL in the browser replacing 'yourserver' with the name of your server:
http://yourserver:8080/SASStoredProcess/do?_action=index .
Select your stored process and click Execute. Respond to any prompt screens.



Stored Process *My SCL program STPRUNEN*

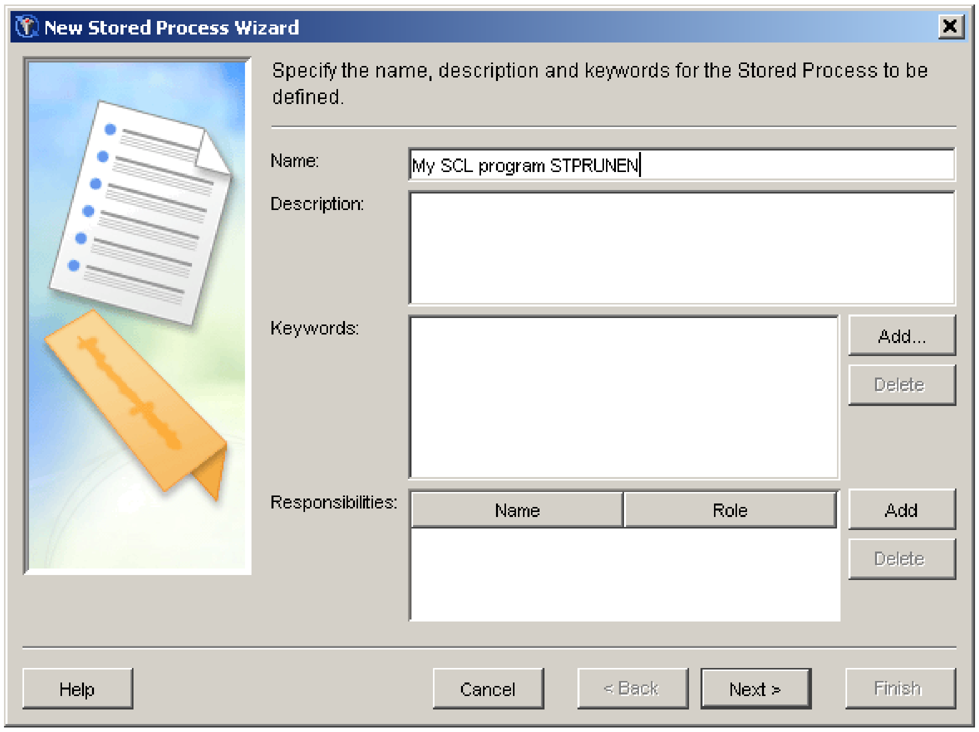
| | |
|---|---|
| **Path:** | //Foundation/AFtest/My SCL program STPRUNEN |
| **Directory:** | C:\Program Files\SAS\SAS 9.1\inttech\sample |
| **File name:** | stprunen.sas |
| **SAS server type:** | Stored Process Server |
| **Creation date:** | Mon Mar 19 09:22:29 EST 2007 |
| **Last modified:** | Mon Mar 19 09:22:29 EST 2007 |
| **Keywords:** | |
| **Description:** | |

**Figure 6. Stored Process Web Application Property Sheet**

**RUNNING STPRUNEN AS A STORED PROCESS**

SAS Integration Technologies includes the STPRUNEN.SAS program that can be used to run SCL code without creating a new .SAS file. With this program, you will need at least one parameter created. _ENTRY is used to pass in the name of the SCL catalog that will be executed. You can register this same program for multiple SCL programs without modifying it each time. You will need to create LIBNAMEs for each catalog location. This can be done by adding all of the LIBNAME statements to stprunen.sas or by adding them to the autoexec.sas file. If they are not defined, the catalogs will not be found. Before beginning, make sure you have completed Steps 1-3 as in the previous example.

1. Run the Stored Process Wizard by selection Actions -> New Stored Process
    a. Enter the following information in the first screen of the wizard (see figure 7) and click NEXT.


New Stored Process Wizard

Specify the name, description and keywords for the Stored Process to be defined.

Name: My SCL program STPRUNEN

Description:

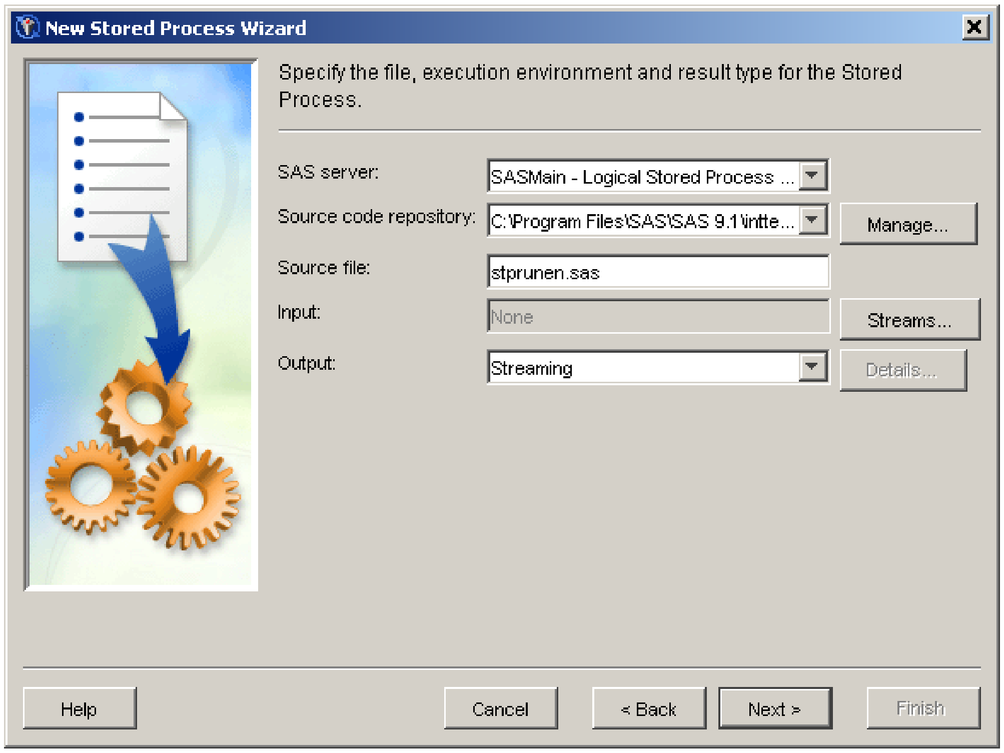
Keywords:

Responsibilities:

| Name | Role |
|---|---|

**Figure 7. Name the stored process**

b. On the next screen, enter the following information (see figure 8) and then click NEXT. Note that this program is in the source code repository at C:\Program Files\SAS\SAS 9.1\inttech\sample. Stprunen.sas is installed with Integration Technologies and does not need to be moved to this location. You do need to create a LIBNAME to the location of your SCL catalog. You can modify stprunen.sas to have a list of all LIBNAMEs that you will run from this program or you can add the list of LIBNAMEs to your AUTOEXEC file.


New Stored Process Wizard

Specify the file, execution environment and result type for the Stored Process.

SAS server: SASMain - Logical Stored Process ...

Source code repository: C:\Program Files\SAS\SAS 9.1\intte...

Source file: stprunen.sas

Input: None

Output: Streaming

**Figure 8. Specifying the server and program information**

c. On the Parameters page, you need to add three parameters. The first is for the SCL program name, *_ENTRY,* that is passed into the stprunen program. The second is for *message1* that passes your message into your SCL program. The final parameter is for *retinfo* for returning information back from your SCL program. Click on the Add Parameters button (see Figure 9).

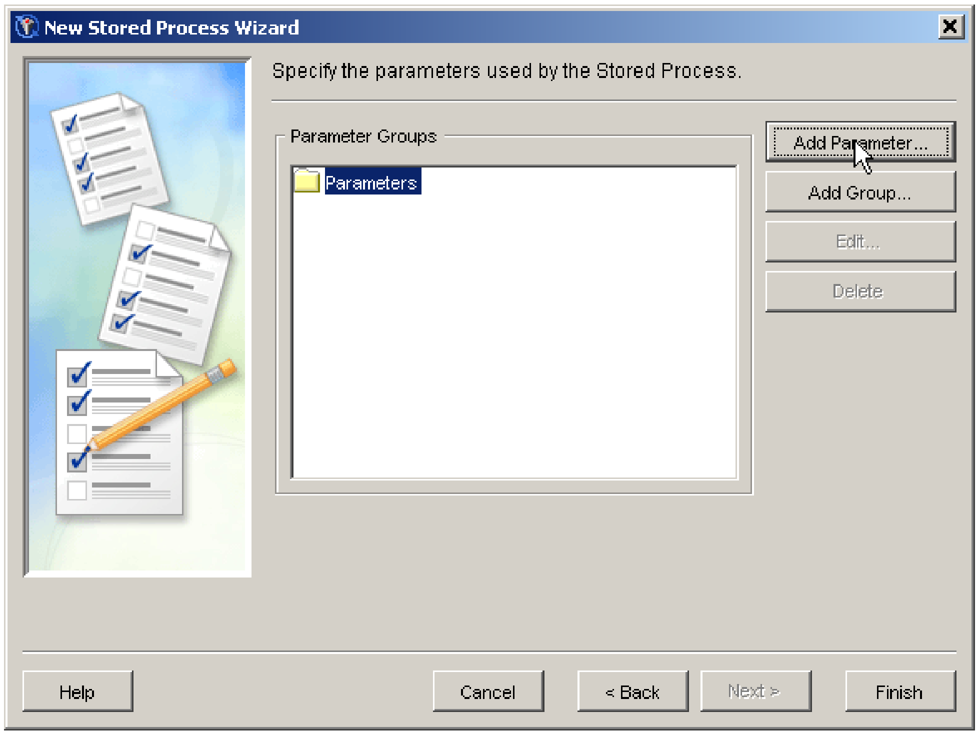
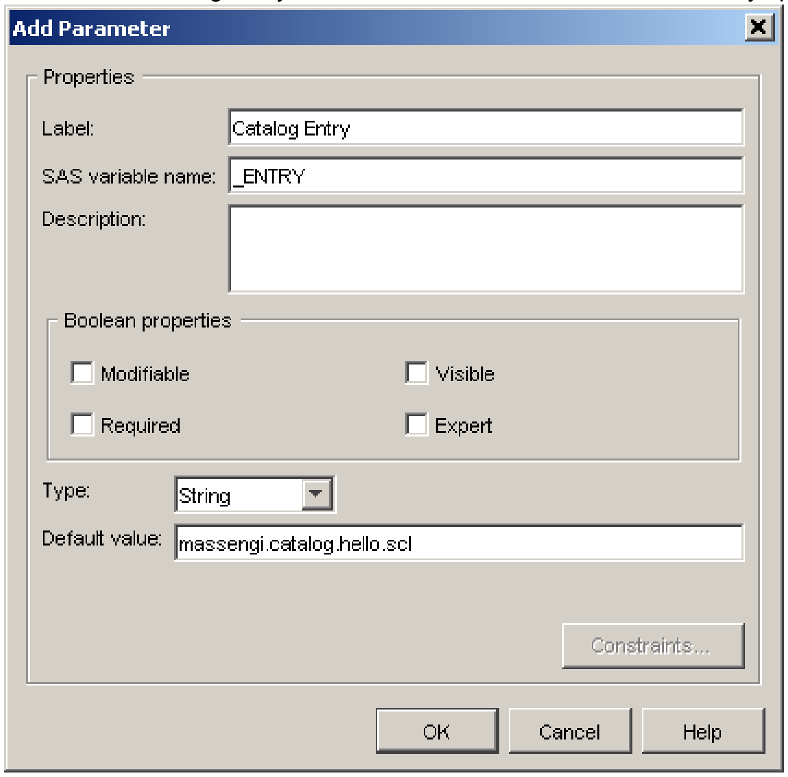


**Figure 9. Parameters screen**

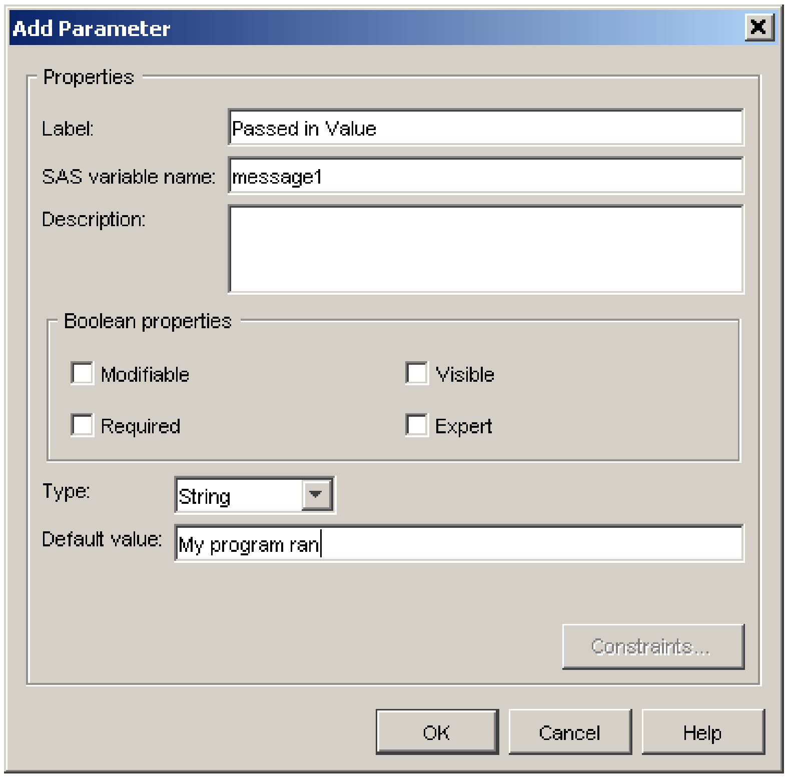
d. First, add the *_ENTRY* parameter as specified in Figure 10. This parameter is used by stprunen.sas to pass in different program names without modifying the program each time. You set the parameter to the value of the SCL catalog that you will run. Make sure all values are as they appear in Figure 10. Click OK when done.



Label: Catalog Entry
SAS variable name: _ENTRY
Description:
☐ Modifiable ☐ Visible
☐ Required ☐ Expert
Type: String
Default value: massengi.catalog.hello.scl

**Figure 10. Add _ENTRY parameter**

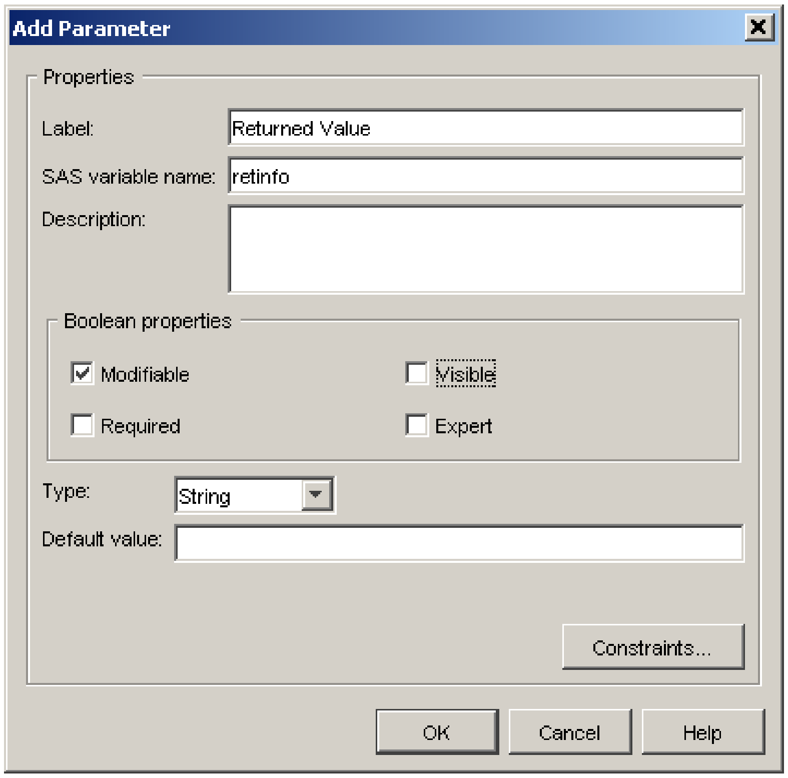
e. Next add the *message1* parameter as specified in Figure 11. This is the value of the parameter in your SCL program that allows you to pass a message in that will be displayed beside the text "Hello World". You will set your message here. Click OK when done.



Label: Passed in Value
SAS variable name: message1
Description:
Modifiable [ ]
Visible [ ]
Required [ ]
Expert [ ]
Type: String
Default value: My program ran

**Figure 11. Add message1 parameter**

f. Finally, add the *retinfo* parameter as in Figure 12. This parameter is for the information returned from your SCL code. This information can be displayed or used in other programs. Click OK when done.



Add Parameter

Properties

Label: Returned Value

SAS variable name: retinfo

Description:

Boolean properties

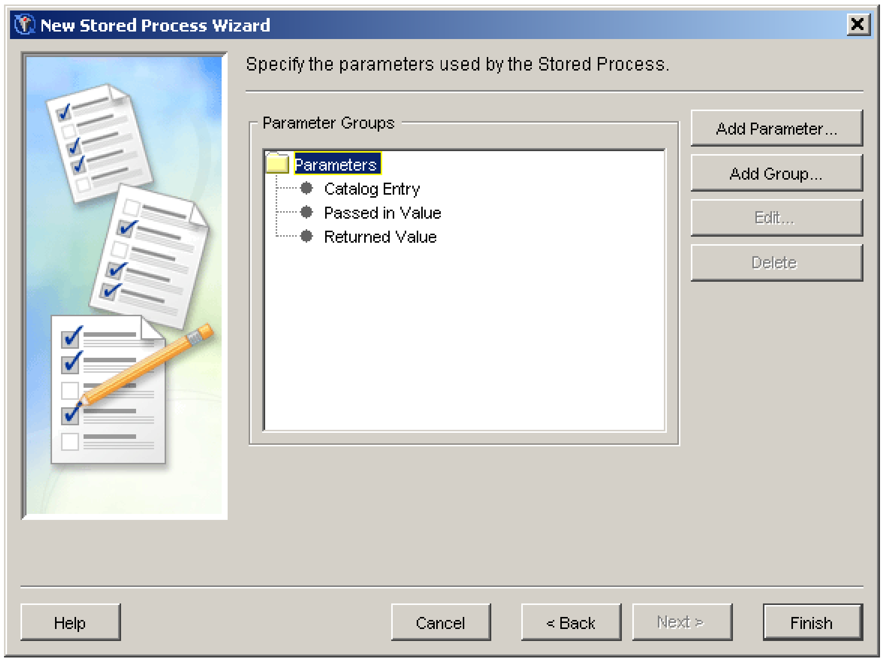
- [✓] Modifiable
- [ ] Visible
- [ ] Required
- [ ] Expert

Type: String

Default value:

**Figure 12. Add retinfo parameter**

g. You have now added all the necessary parameters. If a new Add Parameters screen appears, click CANCEL. You will be returned to the Parameter screen (Figure 13). Click FINISH to complete the registration process.



New Stored Process Wizard

Specify the parameters used by the Stored Process.

Parameter Groups

- Parameters
  - Catalog Entry
  - Passed in Value
  - Returned Value

**Figure 13. Parameter Screen**

h. You can now execute this stored process as you did with the previous program above.

## CONCLUSION

By performing the steps enumerated in this paper, you can see that turning a SAS program or an SCL program into a stored process is an easy and straight forward process. The power and flexibility of SCL programs is even more advantageous now that you understand how to convert them into a stored process to be accessed from anywhere in your organization.

## REFERENCES

Curley, Lynn. 2006. “SAS/AF® Rises Again: Enhancements in SAS® 9.2”. SUGI 31 Proceedings. Cary, NC: SAS Institute Inc. Available: http://support.sas.com/events/sasglobalforum/previous/online.html

Davis, Michael. 1998. “SCL for the Rest of Us: Nonvisual Uses of Screen Control Language”. SUGI 23 Proceedings. Cary, NC: SAS Institute Inc. Available: http://support.sas.com/events/sasglobalforum/previous/online.html

SAS/AF Online Documentation: http://support.sas.com/documentation/onlinedoc/af/

SAS White Paper: SAS Stored Processes: An Introduction and Overview. Available from: http://www.sas.com/whitepapers/index.html

Stored Process Documentation. SAS 9.1.3 Integration Technologies. Available from: http://support.sas.com/rnd/itech/doc9/dev_guide/stprocess/index.html

Weinstein, Heather. 2007. “Converting SAS/IntrNet Programs to SAS Stored Processes”. SAS Global Forum 2007 Proceedings. Cary, NC: SAS Institute Inc. Available: http://support.sas.com/events/sasglobalforum/previous/online.html

## CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the author at:

Darrell Massengill
SAS Institute Inc.
SAS Campus Dr.
Cary, NC 27511
E-mail: Darrell.Massengill@sas.com