Paper 050-2011

A SAS® Programmer's Guide to Stored Processes

Joe Flynn, SAS Institute Inc., Cary, NC, USA

ABSTRACT

If you have SAS programming knowledge and experience, then fully leveraging stored processes in your organization can be a simple task. This paper explains how to develop a stored process by using SAS® Enterprise Guide® 4.3. Topics include:

- defining and registering metadata—permissions; responsibilities; SAS folders; keywords
- writing the SAS code—Output Delivery System (ODS) output; stored process macros; prompts; reporting options
- selecting execution options—STP server; workspace server; result types; source code repositories; operating system access
- defining prompts—the prompt model; output parameters
- determining data sources and targets—input streams; output streams

This paper also includes practical examples that are based on questions submitted to SAS Technical Support.

INTRODUCTION

Stored processes are important in a business intelligence environment. They enable SAS code to be deployed on the SAS server and easily accessed from a variety of client applications. Deploying your code in a centralized location allows easy updates and ensures that everyone is executing the latest revision of the code. This paper covers the basics of creating a stored process from the point of view of a SAS programmer. Some of the frequent problems that are reported to SAS Technical Support are also discussed.

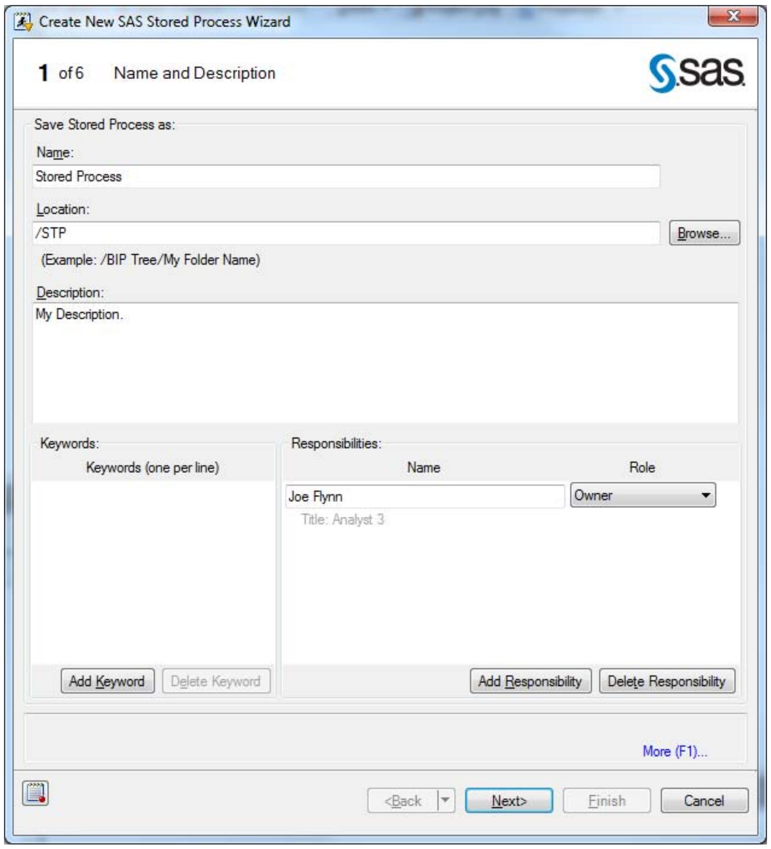
This paper focuses on using SAS Enterprise Guide to create and update your stored processes. Although it is possible to use SAS® Management Console to deploy a stored process, SAS Enterprise Guide handles all of the required steps in one easy-to-follow wizard.

CREATING A STORED PROCESS

As a SAS programmer, you are familiar with the SAS programming language but might not be familiar with the steps for creating a stored process and the related terminology. This section walks you through the SAS Enterprise Guide wizard that enables you to create a stored process. Each step of the process is explained so that you have the knowledge that is required to create a stored process.

STEP 1: NAME AND DESCRIPTION

To invoke the Create New SAS Stored Process wizard in SAS Enterprise Guide select **File ▶ New Stored Process**. There are two main components to a stored process: a metadata object and a SAS file. The first page of the wizard, Name and Description (Display 1), mostly addresses the metadata object that you will create. This metadata object is XML which describes everything about the stored process. This is how the requesting applications determine all of the applicable information, for example, where the SAS code is located, what prompts to display to the user, and on which execution server to run the SAS code.



Save Stored Process as:

Name:
Stored Process

Location:
/STP
(Example: /BIP Tree/My Folder Name)

Description:
My Description.

Keywords:
Keywords (one per line)

Responsibilities:

Name	Role
Joe Flynn Title: Analyst 3	Owner

Display 1. Name and Description Page

In the **Name** field, type the name of the stored process that will be included in the metadata. When users are browsing for your stored process, they will be able to identify it by this name.

The **Location** field identifies the location of the metadata information for your stored process. This location only refers to the location where the metadata will be stored. You will be prompted on a later page of the wizard for the location where you wish to store the physical SAS program on the server. You need to ensure that the location that you choose is one in which you have WriteMetadata permission.

The **Description** field is self-explanatory and enables you to provide a short description about this stored process.

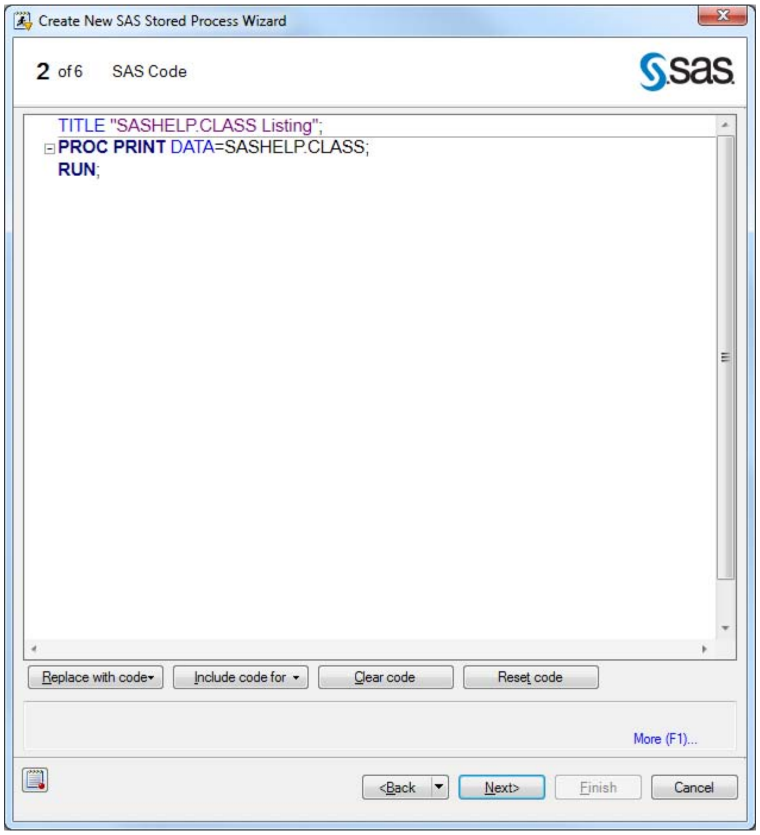
The **Keywords** field can be used to describe the functionality of a stored process. These keywords can be searched on when you are attempting to locate a certain type of stored process.

The **Responsibilities** field can be used to specify the person who is responsible for this stored process. It also provides information about this individual from the metadata.

Click **Next** to go to page 2 of the wizard.

STEP 2: SAS CODE

As a SAS programmer this is where you should feel most comfortable. Page 2 SAS Code is an enhanced editor where you can manually enter SAS code (Display 2).



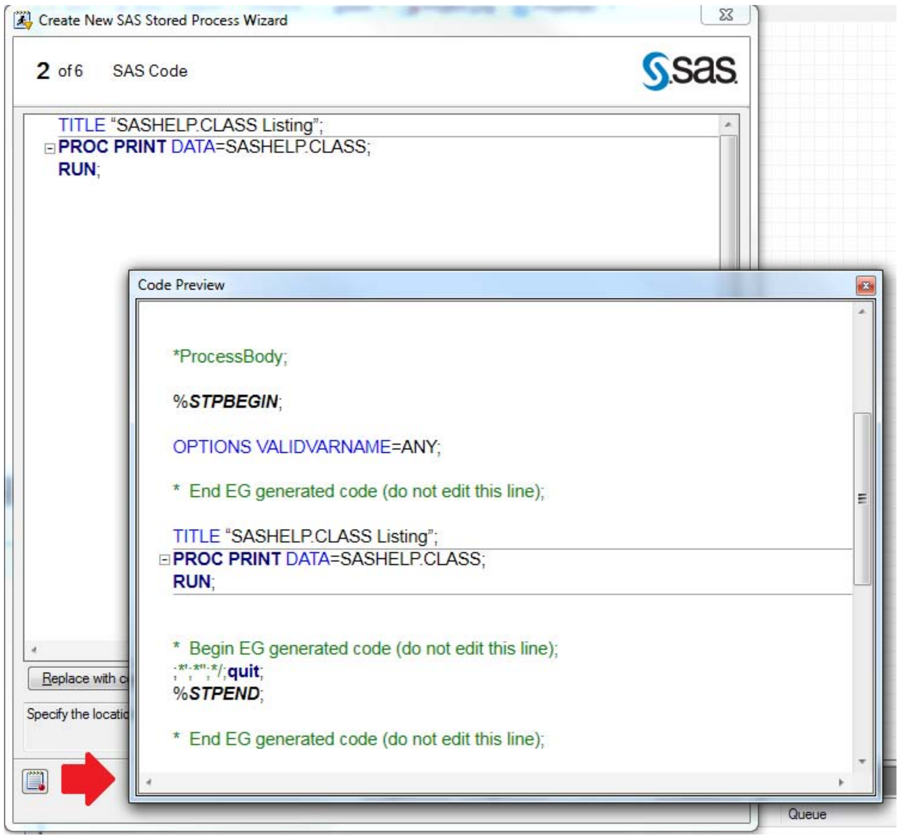
```
TITLE "SASHELP.CLASS Listing";
PROC PRINT DATA=SASHELP.CLASS;
RUN;
```

Display 2. SAS Code Page

You can also click the **Replace with code** button at the bottom of the page to add your SAS code. All of the SAS code that you enter on this page should run as it would in a SAS windowing environment. However, there are a few special considerations when building a stored process, most of which involve ODS output. Take a very simple SAS program which creates a PDF of the SASHELP.CLASS data set:

```
ods pdf file="c:\temp\sh.pdf" style=sasweb;

title "sashelp.class listing";
proc print data=sashelp.class;
run;

ods pdf close;
```

If you run this code on the server, you would only create a physical file and would not return the results back to the client. So, how do you return your results back to the client? There are two ways to accomplish this task. You can use the _WEBOUT fileref as the FILE= option on your normal ODS statement or use stored process macros. However, this discussion will focus on using stored process macros. Think of these stored process macros as a direct replacement for your ODS statements. Therefore, you do not need to put them around your entire program but only around the areas from which you wish to receive output. When creating a stored process, SAS Enterprise Guide automatically appends these macros (%STPBEGIN and %STPEND) around your SAS code. These macros will not show up in your editor window. However, if you click the Code icon () at the bottom left of the wizard, you will see the full SAS code that SAS Enterprise Guide intends to write to the server (Display 3).



```
TITLE "SASHELP.CLASS Listing";
PROC PRINT DATA=SASHELP.CLASS;
RUN;
```

```
*ProcessBody;

%STPBEGIN;

OPTIONS VALIDVARNAME=ANY;

*  End EG generated code (do not edit this line);

TITLE "SASHELP.CLASS Listing";
PROC PRINT DATA=SASHELP.CLASS;
RUN;


*  Begin EG generated code (do not edit this line);
;*';*";*/;quit;
%STPEND;

*  End EG generated code (do not edit this line);
```

Display 3. Full Code Preview

By default, the results of a stored process display in HTML. For example, you execute the following code as a stored process:

```
*ProcessBody;
%stpbegin;

title "sashelp.class listing";
proc print data=sashelp.class;
run;

%stpend;
```

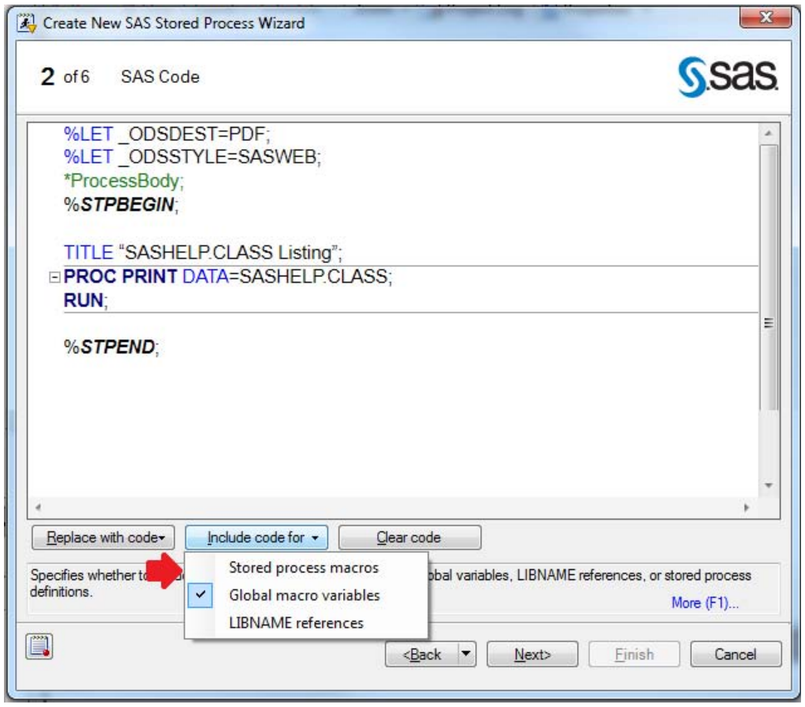
The results that are returned are as if the code was wrapped with ODS HTML statements.

```
ods html file='xxx.html';

title "sashelp.class listing";
proc print data=sashelp.class;
run;

ods html close;
```

To change the type of results that are returned to the user, you can use a reserved macro variable _ODSDEST. Also if you intend to use the SASWEB style, you can set this style using the _ODSSTYLE macro variable. These values must be set before you invoke the %STPBEGIN macro. Remember, however, that SAS Enterprise Guide is wrapping your user-written code with the %STPBEGIN and %STPEND macros (see Display 3). This leaves you with two options for setting the macro variables: use a prompt, which is discuss later, or manually code the macros %STPBEGIN and %STPEND in your editor window and disable them in SAS Enterprise Guide. So for example your program would now look like this:

```
%let _odsdest=pdf;
%let _odsstyle=sasweb;
*processbody;
%stpbegin;

title "sashelp.class listing";
proc print data=sashelp.class;
run;

%stpend;
```

To ensure that these macros are disabled to prevent duplicates, click the **Include code for** button and deselect **Stored process macros** as shown in Display 4.



```
%LET _ODSDEST=PDF;
%LET _ODSSTYLE=SASWEB;
*ProcessBody;
%STPBEGIN;

TITLE "SASHELP.CLASS Listing";
PROC PRINT DATA=SASHELP.CLASS;
RUN;

%STPEND;
```

Display 4. Disabling the %STPBEGIN and %STPEND Macros

Reserved macro variables exist for any ODS options that you might want to set in your output. A list of these variables is available in the "Using Reserved Macro Variables" section of the *SAS® 9.2 Stored Processes: Developers Guide* (SAS Institute Inc. 2009).

Next you need to account for any prompts that you want to include in the stored process. When thinking about prompts in a stored process, think of them as a macro variable which is declared at the very top of the program. To keep this example simple, assume that you intend to prompt the user for a single age value in which to subset the SASHELP.CLASS data set. Assuming that the value will be set with a %LET AGE= statement at the top of the code, you simply need to add the WHERE clause to the PROC PRINT statement and specify the &AGE parameter, which is defined later:

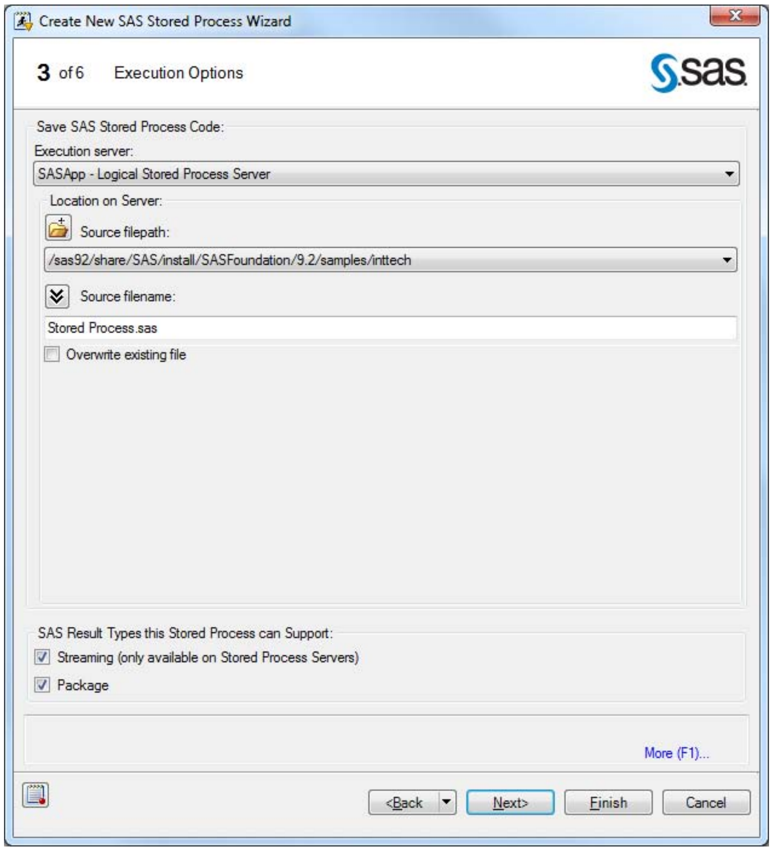
```
%let _odsdest=pdf;
%let _odsstyle=sasweb;
*processbody;
%stpbegin;

title "sashelp.class listing";
proc print data=sashelp.class;
where age = &age;
run;

%stpend;
```

Make sure you do not have a %LET declaring any of your prompt values directly in the code, or the values will be overridden. You might have noticed in the code that a `*ProcessBody;` statement is added. This is only necessary if you are running on the SAS Workspace Server. This statement only serves as a flag for where to insert your prompt values into the SAS code.

Click **Next** to go to page 3 of the wizard.

STEP 3: EXECUTION OPTIONS

The next page of the wizard addresses execution options (Display 5).



Save SAS Stored Process Code:
Execution server: SASApp - Logical Stored Process Server
Location on Server:
Source filepath: /sas92/share/SAS/install/SASFoundation/9.2/samples/inttech
Source filename: Stored Process.sas
☐ Overwrite existing file

SAS Result Types this Stored Process can Support:
☑ Streaming (only available on Stored Process Servers)
☑ Package

Display 5. Execution Options

In the **Save SAS Stored Process Code** field, there are a few options that you need to specify. First, you must choose an Execution server from the list. You can decide between executing on the Logical Stored Process Server or the Logical Workspace Server. There are many advantages to using the Logical Stored Process Server including reduced execution times and the availability of streaming output. Special considerations need to be taken, however, if you intend to use the Logical Stored Process Server. All SAS sessions that are executed run under a group account called SAS General Servers (by default the user ID that is associated with this group is SASSRV). You need to ensure the SASSRV account has access to all library locations that are accessed in your SAS program.

In the **Location on Server** field, you must choose a Source filepath in which to store your SAS file on the server file system. These locations are defined as *source code repositories* which are basically directories on the file system. Additional paths can be added using SAS Enterprise Guide or SAS Management Console. If your execution server is set to the Logical Stored Process Server, you must ensure that the SASSRV account has Write access to the directory that is specified, as this account will be used to write your SAS program to the file system. Alternatively, if the Logical Workspace Server is selected, the identity that is logged into SAS Enterprise Guide will be used to write the source file.

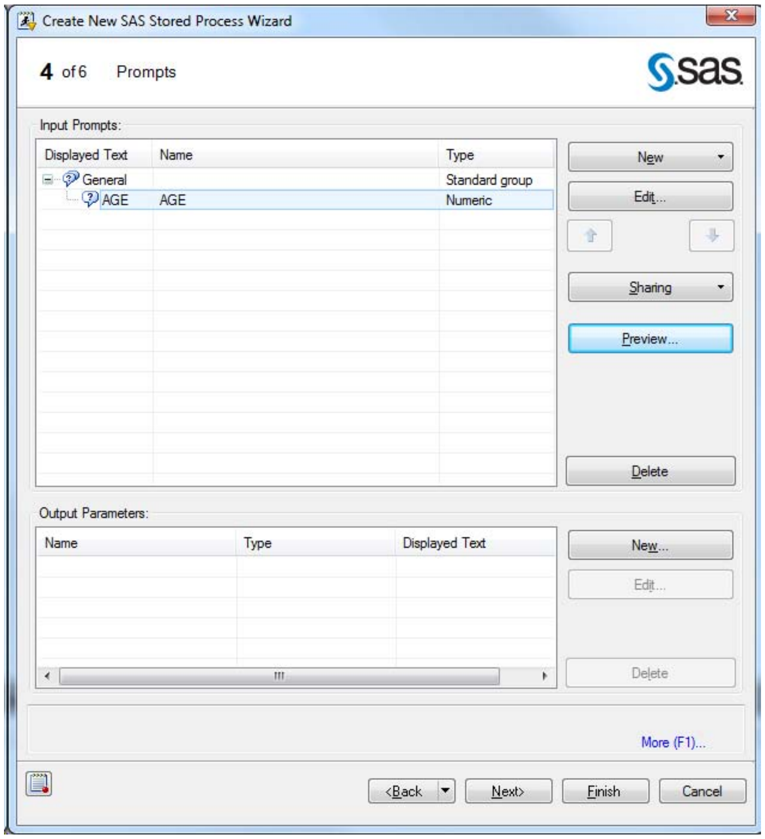
Finally, in the **Source filename** field, you must enter a name, which will determine the name of the physical SAS file on the server file system, for example, StoredProcess.sas.

Next, in the **SAS Result Types this Stored Process can Support** field, you must choose your output options. You can choose between Streaming and Package. *Streaming* output enables you to make use of the _WEBOUT fileref, which is discussed later. This will enable you to build Web applications and stream results directly back to the client. *Packaged* output generates all output on the server first. After completion, the output is compressed and sent back to the client. If you are using the _WEBOUT fileref then make sure you only select **Streaming**. If you are using the stored process macros then you can select both output types simultaneously, which will enable the requesting application to decide which result type to use.

Click **Next** to go to page 4 of the wizard.

STEP 4: PROMPTS

The next page of the wizard addresses prompts (Display 6).

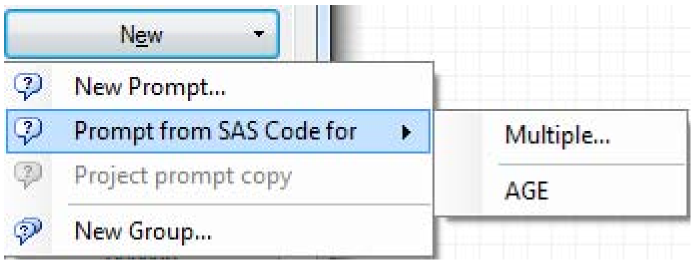


Displayed Text	Name	Type
General		Standard group
AGE	AGE	Numeric

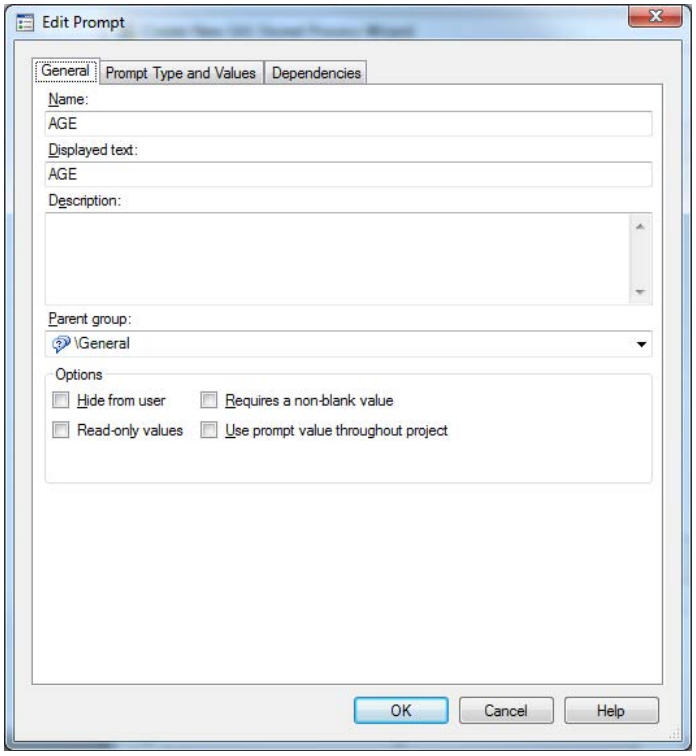
Name	Type	Displayed Text

Display 6. Prompts

In this step, you set up any prompt values for your stored process. As you recall in the simple example, you are subsetting your SASHELP.CLASS data set by an age value. To add a prompt, click **New**, select **Prompt from SAS code for,** and select **AGE**.



SAS Enterprise Guide determines what prompts you intend to add to the stored process by scanning the code for any unassigned macro variables. On the **General** tab of the initial prompt page, the **Name** and **Displayed text** fields are prepopulated with your AGE value (Display 7).


Edit Prompt

General | Prompt Type and Values | Dependencies

Name: AGE

Displayed text: AGE

Description:

Parent group: \General

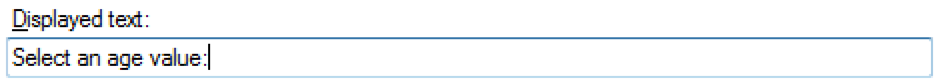
Options

- Hide from user
- Requires a non-blank value
- Read-only values
- Use prompt value throughout project

Display 7. The General Tab Showing the Name and Displayed Text Fields

The **Name** field corresponds to the name of the macro variable in your SAS code. In this example, this must remain AGE in order for your stored process to function properly.

The **Displayed text** field however determines the text that is displayed for the prompt. Setting this value to something like **Select an age value** would be more fitting.

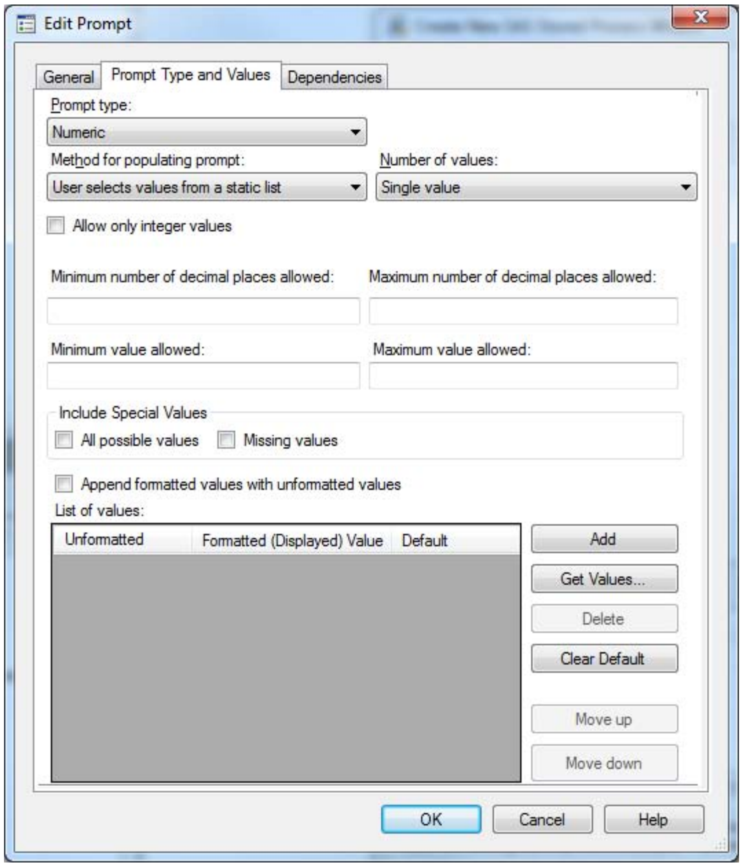

Displayed text: Select an age value:

In the **Description** field, you can specify additional details about your prompt.

The **Parent group** field allows prompts to be grouped into different sections. This can enable a more logical way to present prompts to a user. For this example, the default **\General** will suffice.

The **Options** field provides a few self-explanatory options for this prompt, none of which currently apply to our current example.

Selecting the **Prompt Type and Values** tab provides additional options for this prompt (Display 8).



Prompt type: Numeric

Method for populating prompt: User selects values from a static list

Number of values: Single value

Allow only integer values

Minimum number of decimal places allowed:

Maximum number of decimal places allowed:

Minimum value allowed:

Maximum value allowed:

Include Special Values

All possible values

Missing values

Append formatted values with unformatted values

List of values:

Unformatted	Formatted (Displayed) Value	Default

Display 8. The Prompt Type and Values Tab

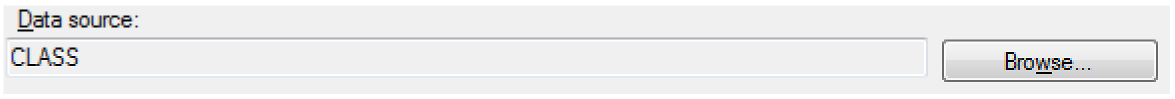
For this example you intend to prompt a user for an age. Therefore, in the **Prompt type** field select **Numeric**.

In the **Method for populating prompt** field, there are three options: **User enters a value**, **User selects values from a static list**, and **User selects values from a dynamic list**. To use a dynamic list the SAS table must be registered in the metadata. For this example, you have not done that, and changes in the age variable in the SASHELP.CLASS data set are not expected; therefore, selecting **User selects from a static list** will suffice.

To keep the example simple, you have only coded for a single value. Therefore, chose the default option **Single value** in the **Number of values** field.

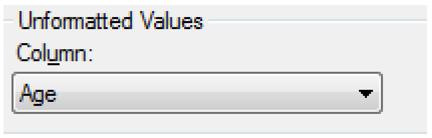
You do not want to add any additional constraints so skip to the **List of Values** field. In this section you can manually add each value by clicking the **Add** button, or you can load all of the values directly from your SASHELP.CLASS data set. To do so click **Get Values.…**.

In the **Get Values** dialog box, click **Browse..** and locate and select the SASHELP.CLASS data set.



Data source: CLASS

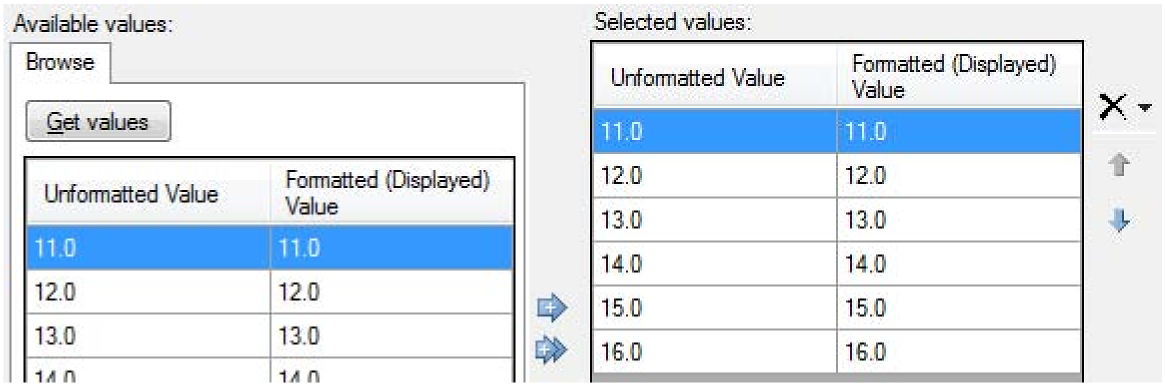
Under **Unformatted Values Column**, select **Age**.



Unformatted Values

Column: Age

You are not applying any formats to these values; therefore, you can skip to the **Available values** field. Here you must click **Get values** and the distinct age values will be loaded from the SASHELP.CLASS data set. Click the double arrow to move all values from the **Available values** field to the **Selected values** field.

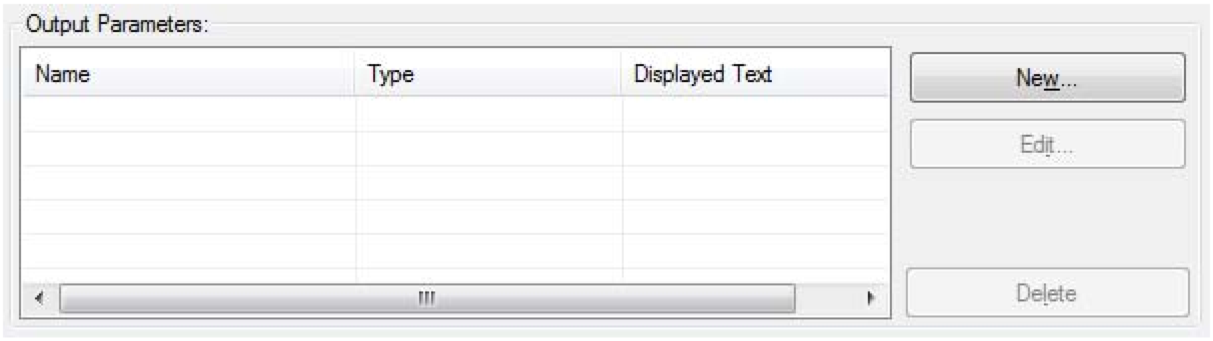


Unformatted Value	Formatted (Displayed) Value
11.0	11.0
12.0	12.0
13.0	13.0
14.0	14.0

Unformatted Value	Formatted (Displayed) Value
11.0	11.0
12.0	12.0
13.0	13.0
14.0	14.0
15.0	15.0
16.0	16.0

Click **OK** to return to the **Prompt Type and Values** tab. Now the **List of values** should be populated. Because you chose a static prompt, dependencies will not be available and, therefore, this concludes the setup of your Age prompt.

In addition to input prompts, output parameters are also available.

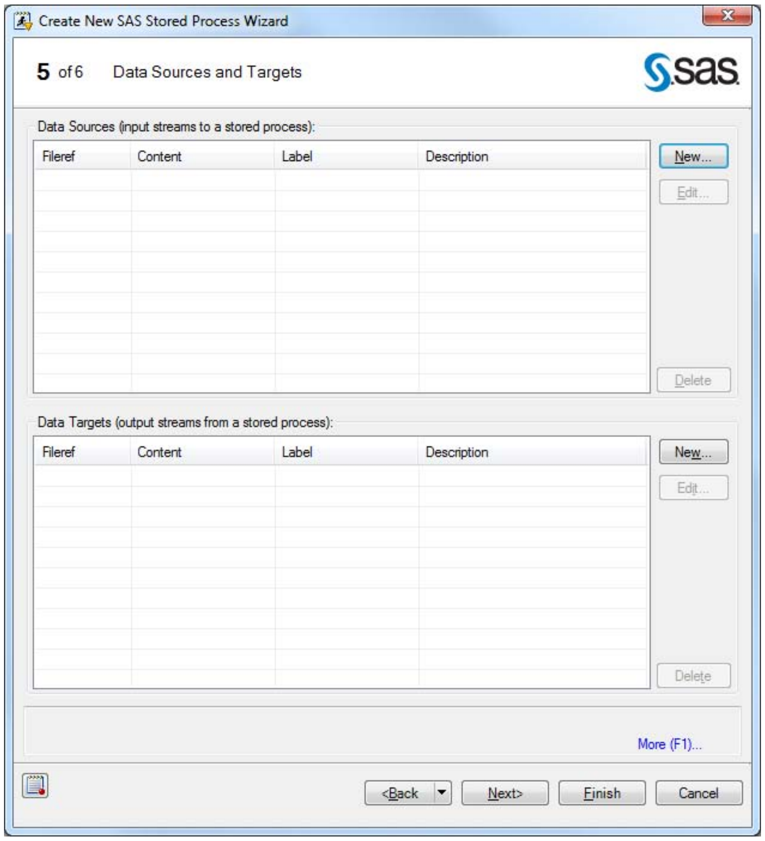


These enable stored processes to return SAS macro variables to the requesting client. Output parameters are generally used in conjunction with SAS BI Web Services, and are not included in this example.

Click **Next** to go to page 5 of the wizard.

STEP 5: DATA SOURCES AND PROMPTS

The next page of the wizard addresses data sources and targets (Display 9).


Create New SAS Stored Process Wizard
5 of 6 Data Sources and Targets
sas
Data Sources (input streams to a stored process):
Fileref
Content
Label
Description
New...
Edit...
Delete
Data Targets (output streams from a stored process):
Fileref
Content
Label
Description
New...
Edit...
Delete
More (F1)...
<Back
Next>
Finish
Cancel

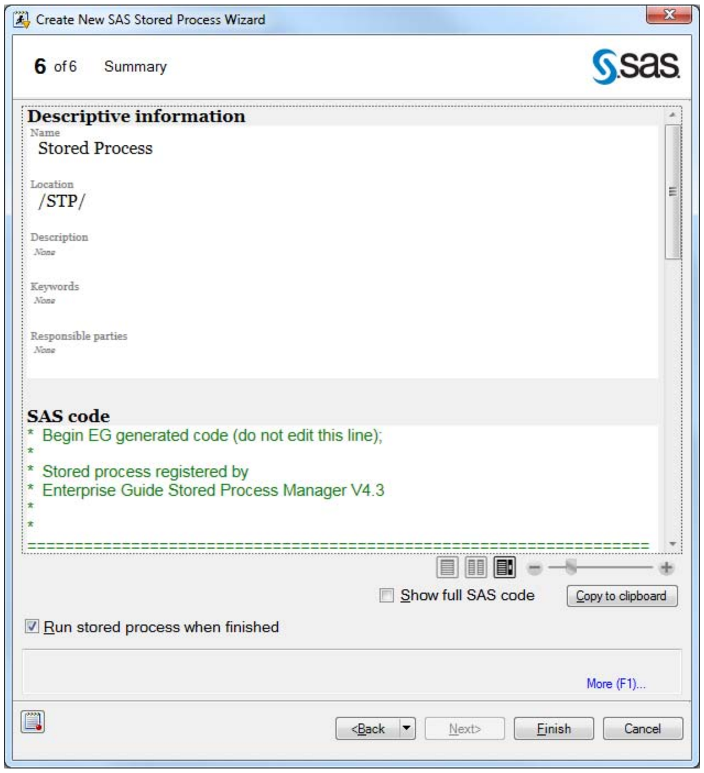
Display 9. Data Sources and Targets

Data sources are used to stream data between the clients. Currently SAS Enterprise Guide 4.3 does not support the execution of stored processes with data sources. However, it will enable you to configure them. In the "Additional Topics" section of this paper, data sources are used to stream data from an Excel spreadsheet into a stored process. For this example, however, leave the **Data Sources** and **Data Targets** fields blank.

Click **Next** to go to page 6 of the wizard.

STEP 6: SUMMARY

This section provides a summary of all of the information that you previously provided to the wizard (Display 10).



Descriptive information

Name
Stored Process

Location
/STP/

Description
None

Keywords
None

Responsible parties
None

SAS code

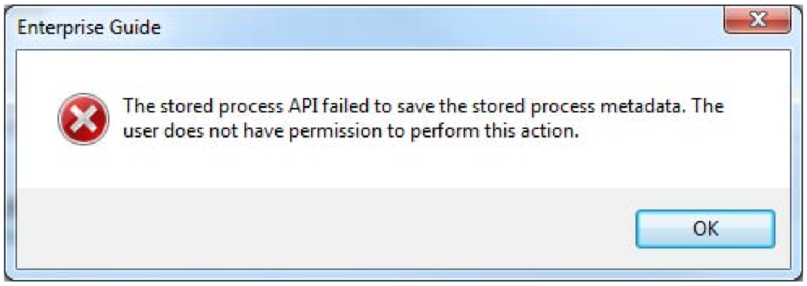
```
*  Begin EG generated code (do not edit this line);
*
*  Stored process registered by
*  Enterprise Guide Stored Process Manager V4.3
*
*
=================================================================
```

Display 9. Summary

When you click **Finish**, SAS Enterprise Guide will complete these tasks:

1. write the metadata object that is associated with this stored process
2. write the physical .SAS file to the source code repository

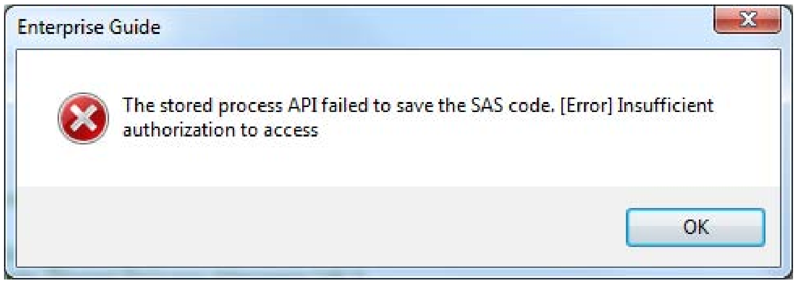
The following error commonly occurs at this step:



The stored process API failed to save the stored process metadata. The user does not have permission to perform this action.

This error is caused by not having the WriteMetadata permission for the SAS folder location that was provided on Page 1 of the wizard.

Here is another common error that you might encounter:



The stored process API failed to save the SAS code. [Error] Insufficient authorization to access

This error is caused by not having Write access to the source file directory in the operating system. If the SAS Logical Stored Process Server is being used, ensure that the SASSRV account can write to the location that is specified. If the SAS Logical Workspace Server is being used, ensure that the user who is logged on to SAS Enterprise Guide has sufficient Write access to the location that is specified.

ADDITIONAL TOPICS

This section of the paper addresses common problems that customers report to SAS Technical Support when they are creating stored processes.

DATE VALUES

Although date values are available in the Add A New Prompt dialog, the values are passed as text strings to macro variables. For example, imagine that you pass a date variable DATE into a stored process. A common mistake is to directly use this DATE variable like this:

```
proc sql;
select * from work.foo
where datefield < &date;
quit;
```

This will result in an error because the DATE variable is passed to the stored process with syntax that is similar to the following:

```
%let date = 01jan2010;
```

In this situation, you need to tell SAS that this value is in fact a date value, so this is the correct syntax to use:

```
proc sql;
select * from work.foo
where datefield < "&date"d;
quit;
```

MULTI-VALUED PROMPTS

Consider the previous example and imagine that the user can select multiple age values. This will require a bit more coding, because in SAS, each value that is passed to the stored process is contained in a new macro variable. For example, you select two ages 12 and 13. These values are passed to the stored process like this:

AGE=12
AGE0=2
AGE1=12
AGE2=13

As you can see, Age0 acts as an index for the number of values that are passed in for the AGE parameter. A numeric suffix is added to the end of AGE and incremented for each value that is chosen by the user. To make this into a usable format for your WHERE clause, you must do some manipulation. The following example includes a numGen macro, which takes one parameter **`parm`**. This is the name of the prompt value for which you wish to create an IN clause, which in this case is age. The macro examines the Age0 value to determine the number of values that are passed into the stored process. If this value is greater than or equal to two, a list will be generated for use within the IN clause of the WHERE statement. If only one value is passed into the stored process, it will merely return the single value. The following code includes comments to help further understand the code.

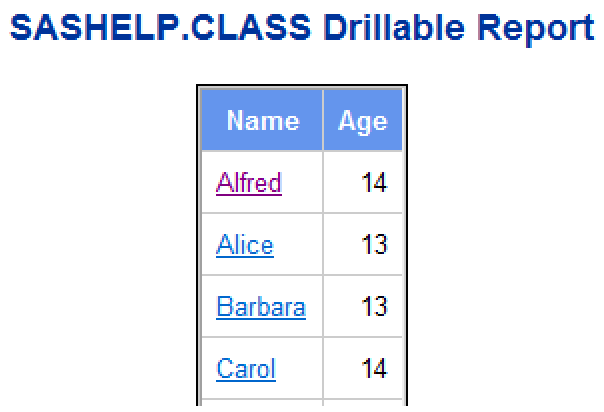
```
%macro numGen(parm=);
            %global &parm.0;
   /* Evaluate the Age0 parameter. If it is greater than or equal to 2 a list will be
      generated for your IN clause. */
        %if %eval(&&&parm.0 ge 2 ) %then %do;
   /* Create a comma separated list of all values that are passed into the stored process. */
                        %do i = 1 %to &&&parm.0;
                                        %if &i = 1 %then %do;
                                                &&&parm&i
                                        %end;
                                        %else %do;
                                                , &&&parm&i
                                        %end;
                        %end;
        %end;
   /* If Age0 is less than 2, only one parameter was specified, so put the single value out. */
        %else %do;
                &&&parm
        %end;
%mend;

%stpbegin;
proc print data=sashelp.class;
   /* Call the numGen macro and pass in the value of your parameter, which in this case is
      age. */
where age in (%numGen(parm=age));
run;
%stpend;
```

DRILLABLE REPORTS

"How do I create a drillable report?" This is a common question to SAS Technical Support. This task can be accomplished with two stored processes and some REPORT procedure code. One feature of a stored process is the ability to call it directly from a URL. This requires a stored process Web application. You will need to know the following information before you build your first stored process:

1. The URL to your stored process Web application, for example, **`http://your.web.server:8080/SASStoredProcess/do`**.
2. The location of the SAS folder for your second stored process, for example, **`/STP/Report2`**.

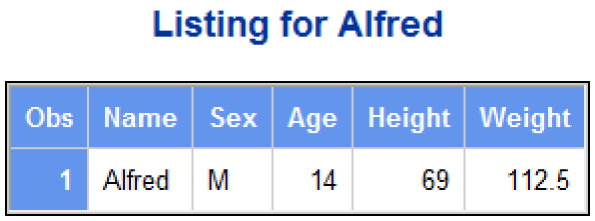
Imagine that you have two stored processes in the **`/STP`** SAS folder location in metadata called Report1 and Report2. You want Report 1 to provide a listing of the members of the SASHELP.CLASS data set as well as the members' age. Your report looks like this:



SASHELP.CLASS Drillable Report

Name	Age
Alfred	14
Alice	13
Barbara	13
Carol	14

You want the members' names to actually be links, which pass the value of **Name** to Report2, which is a subset of the SASHELP.CLASS data set by the name. Report 2 looks like this:



Listing for Alfred

Obs	Name	Sex	Age	Height	Weight
1	Alfred	M	14	69	112.5

The code for Report1 looks like this:

```
title "SASHELP.CLASS Drillable Report";
proc report nowd data=sashelp.class headline;
    columns Name Age;
    define Name / width = 20;
    define Age / width=6;
    compute Name;
       urlstring =
'http://d72206.na.sas.com:8080/SASStoredProcess/do?&_program=/STP/Report2&name=' ||
strip(name);
call define(_col_, 'URL', urlstring);
    endcomp;
  run;
```

Notice that the URLSTRING= field provides the URL to the stored process Web application, the SAS folder location for Report2, and a name parameter that is being passed to Report2. The Report2 code looks like this:

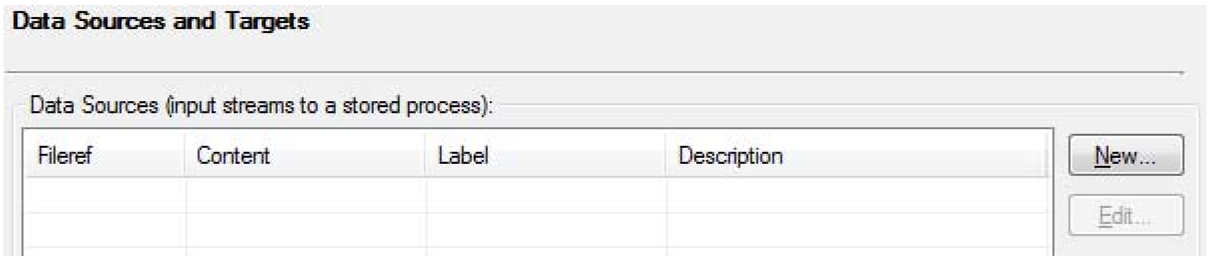
```
title "Listing for &name";
proc print data=sashelp.class;
where name = "&name";
run;
```

A RANGE OF EXCEL DATA AS INPUT

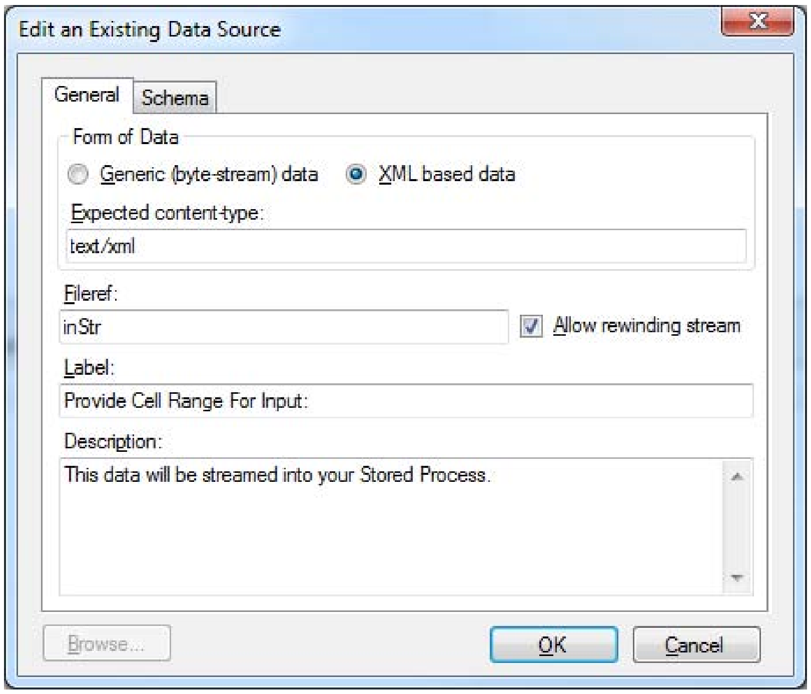
The ability to provide a range of Excel data as input to a stored process is made possible through input streams. To make use of this feature create a stored process with the following code:

```
   /* The input stream will be the fileref that you designate from the add input datasource
      menu. */
libname instr xml;

data work.report_input;
        /* &_webin_sasname is a reserved macro variable for uploading data. */
     set instr.&_webin_sasname;
run;

proc print;
run;
```

On page 5 of the Create New SAS Stored Process wizard next to the **Data Sources (input streams to a stored process)** field, click **New** to create a new data source input stream.



On the Edit an Existing Data Source page on the **General** tab, select **XML based data** under **Form of Data**. This applies to the format of Excel.


Edit an Existing Data Source

General | Schema

Form of Data
( ) Generic (byte-stream) data (●) XML based data

Expected content-type: text/xml

Fileref: inStr [✓] Allow rewinding stream

Label: Provide Cell Range For Input:

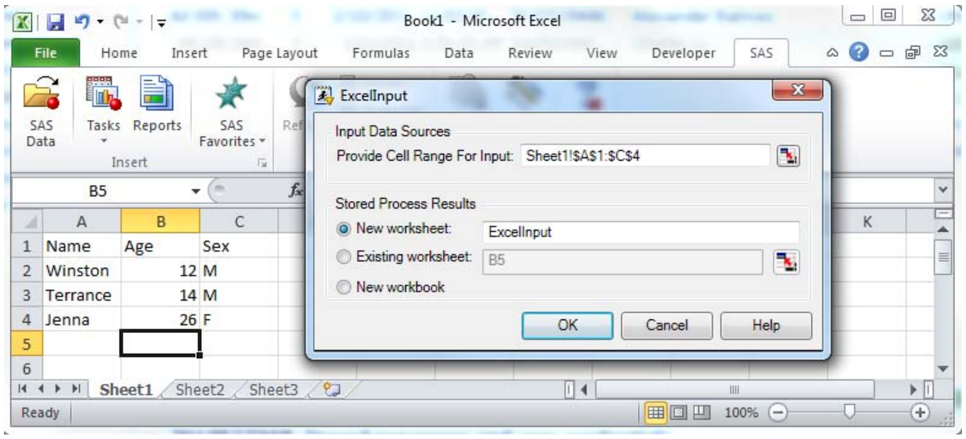
Description: This data will be streamed into your Stored Process.

Selecting this box should automatically change the **Expected content-type** field to **text/xml**.

Next input the fileref that is used in the SAS code, which in this example is inStr and select **Allow rewinding stream**. This will enable the stored process to make multiple passes through the data and will be required for reading in Excel data.

The information in the **Label** field specifies what is presented to the user. You can provided additional details about the data source in the **Description** field.

When you execute the stored process from Excel, you are prompted to add the input data source, which can be a range of cells.



ExcelInput

Input Data Sources

Provide Cell Range For Input: Sheet1!$A$1:$C$4

Stored Process Results

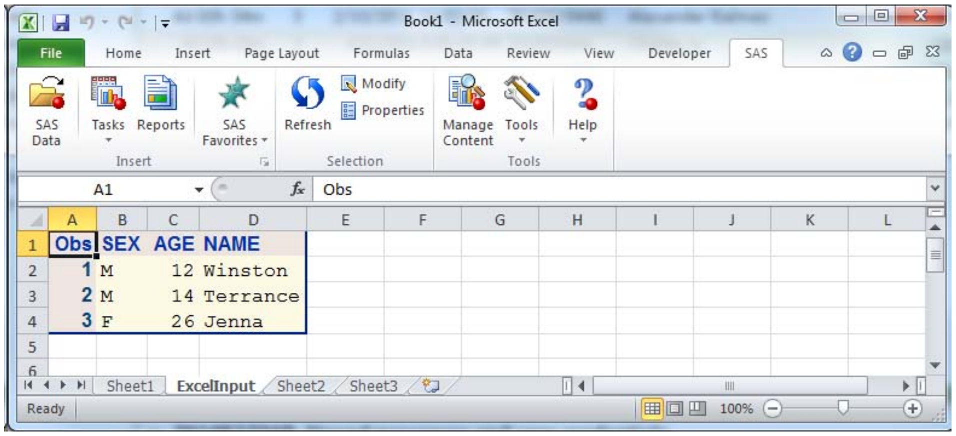
(●) New worksheet: ExcelInput

( ) Existing worksheet: B5

( ) New workbook

	A	B	C
1	Name	Age	Sex
2	Winston	12	M
3	Terrance	14	M
4	Jenna	26	F

Click **OK** and the stored process streams the designated cells to the stored process, writes them to the WORK.REPORT_INPUT data set, and prints this data to a new worksheet called ExcelInput.



	A	B	C	D
1	Obs	SEX	AGE	NAME
2	1	M	12	Winston
3	2	M	14	Terrance
4	3	F	26	Jenna

CONCLUSION

In conclusion, with solid knowledge of the SAS programming language leveraging stored processes at your site should be an easy task. SAS Enterprise Guide 4.3 makes this task easier by providing an easy-to-follow wizard which creates the required metadata and source code. Understanding stored process macros and their role in output is a key concept which will enable you to convert existing programs directly into stored processes.

RECOMMENDED READING

SAS Institute Inc. 2009. "Using Reserved Macro Variables." *SAS® 9.2 Stored Processes: Developer's Guide*. Cary, NC:SAS Institute Inc. Available at support.sas.com/documentation/cdl/en/stpug/61271/PDF/default/stpug.pdf.

CONTACT INFORMATION

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

Joe Flynn
SAS Institute Inc.
SAS Campus Drive
Cary, NC 27513
E-mail: support@sas.com
Web: support.sas.com