Paper 035-2008

Crossing the Border: Steps to Writing SAS$^{©}$ Stored Processes

Peter Eberhardt, Fernwood Consulting Group Inc., Toronto, ON

ABSTRACT

Writing a SAS Stored Process requires more that a thorough understanding of the business problem you are trying to solve; it requires a basic understanding of the SAS BI Architecture. This paper will provide a simple introduction to the SAS BI Architecture, and then dive into the workings of the SAS IOM servers to help you tune your code and return the appropriate results to your client. You will learn the nuances of parameter handling at the top of your program, through to discovering the identity of the user submitting the code, and down to the various options to controlling the output you will return. As with all SAS programming, you have many ways you can develop your SAS Stored process; this paper will help you to understand the implications of some of your decisions.

INTRODUCTION

Many of us have written reports or general processing code that has taken on a life of its own in the organization. Initially, you may have fielded requests for the results of the code, made minor changes, run the code, then, pass on the results. In an effort to offload some of your work, perhaps you gave the code to someone in another department to run. Inevitably it would land back in your lap when something went wrong, and the worst thing about it was that changes were made to your original source. To get around the problem of changed code you carefully parameterize the code and turn it into a SAS autocall macro. And this seems to work until users in another division need access to the code, but do not have access to the autocall library. So, a copy is made and you wait for the inevitable “Peter, the report you wrote does not work anymore so can you fix it”. More recently, as your company has rolled out the SAS Business Intelligence (BI) Architecture, you are getting complaints that running your macro is too hard, and “Why can’t I run it from SAS Enterprise Guide (EG) or the Add-in for Microsoft Office (AMO)?”. SAS Stored Processes might be the answer to these issues.

In this paper I will review the basics of the SAS BI Architecture. To write a SAS Stored Process you have to understand more than the business problem and the core SAS code required to solve it. To write a SAS Stored Process you have to understand the SAS BI Architecture, in particular the SAS servers and processes that can be deployed and how a SAS Stored Process uses them. The SAS servers will be examining are:

- The metadata server
- The object spawner
- The stored process server
- The workspace server

Related to the use of the SAS servers and fundamental to some of the choices you will have to make on deploying a SAS Stored Process are:

- Authorization
- Authentication

Before we look at the interaction we will have with the SAS servers, let us look us first examine this entity called a SAS Stored Process.

A SAS STORED PROCESS

A SAS Stored Process consists of 2 major components

- The SAS code that executes
- The metadata that describes the code

Without both of these components we do not have a SAS Stored Process. We will examine both of these components, starting with the metadata. The following section will give a brief overview of the metadata component of the SAS Stored Process. You might have some unanswered questions as you read through this section; however, you should find the answers to these questions in the section on the metadata server.

SAS STORED PROCESS: METADATA

Metadata is a term that is widely used. Although there can be different nuances in how some people use the term, its fundamental meaning is generally considered to be data that describes, or provides information, about other data. For example, a SAS table contains data. If we look at the output of PROC CONTENTS we see the metadata. The data for a column in the table contains the values for each row; the metadata for the column gives information about the column. For example, if we have a table of students with a column called weight, the data are the weights of all of the students in the class. The metadata about the column would be information such as

- The data type (e.g. numeric)
- A label for the column name (e.g. Weight of Student)
- A display format for the data (e.g. 7.2)

An examination of the output of PROC CONTENTS or the SAS dictionary tables will show that SAS maintains a wealth of metadata about its tables. SAS has taken this approach a step further by creating metadata about the SAS programmes we call SAS Stored Processes. As we look at the metadata components remember that some of your questions will be addressed in later sections.

Let's refine our definition of a SAS Stored Process. A SAS Stored Process is a ***centrally managed*** programme that ***runs on a server*** that can ***return results*** to the user. The parts of interest to us are:

- Centrally managed
  - This means we have an administrator to manage our SAS code
  - This means we have a database to manage the metadata about the code
  - This means we have a server somewhere for the single point of management
- Runs on a server
  - This means our source code has to be accessible to a server
  - This means our data must be accessible to a server
  - This means out users must be able to run programmes on a server
- Return results
  - This means we have to be able to communicate between the server and the user

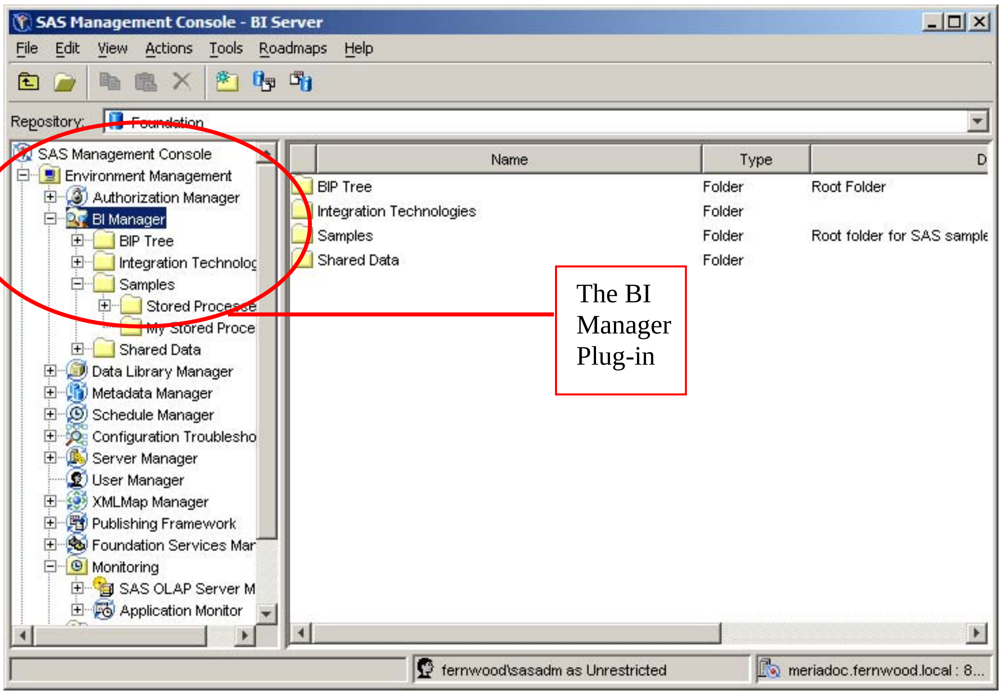
One implication of this is that ***you do not need SAS on your computer to run a*** SAS Stored Process. Another is that, as was stated earlier, you need to define metadata about the SAS Stored Process. The basic metadata required to define a SAS Stored Process are:

1. The name of the SAS Stored Process
2. The SAS server upon which the SAS Stored Process will run
3. The location of the SAS code on the file system
4. The name of the SAS file
5. The input parameters
6. Type of output to be generated

The following screen shots show the metadata being entered through the SAS Management Console (SMC). The process of entering the metadata through the SMC is relatively straight forward exercise so no detail on the screens will be provided. Figure 1 shows the main screen of the SMC; the SAS Stored Processes are normally located in BI Manager folder. In general the location of the SAS Stored Process within the BI Manager is not important. However, if your SAS Stored Process is to be used from within Web Report Studio then the location is important. If your SAS Stored Process is to be used from Web Report Studio then you must save in one of two places in the BI Manager:

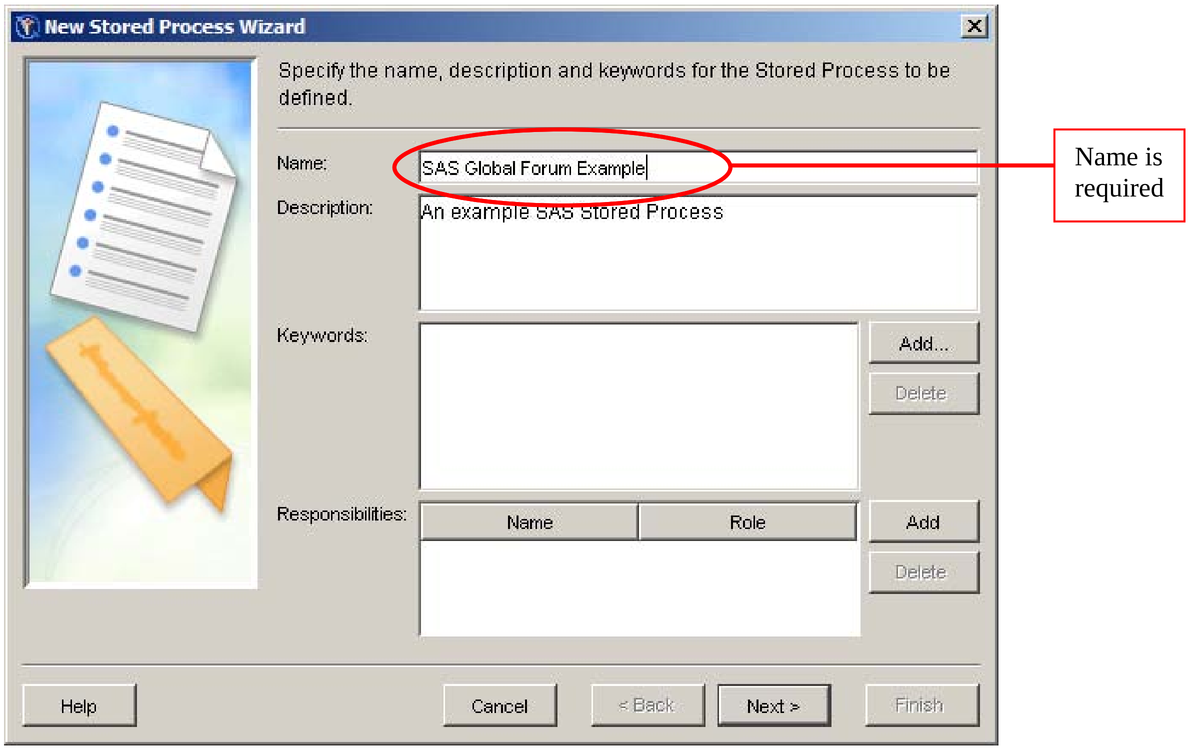
1. BIP Tree … ReportStudio… Shared… Reports
2. BIP Tree … ReportStudio… *userid*… Reports

Figure 2 shows the initial screen when defining a SAS Stored Process; the name of the SAS Stored Process is the only required variable. Although it is not required, it is good practice to enter a description. This becomes more important when the same code file is used in multiple SAS STORED PROCESS definitions. For example, you might have a SAS Stored Process with many parameters, and depending upon how, where, or who is executing the SAS Stored Process you may define different default values and even hide some of the prompts to 'customize' the SAS Stored Process for different uses. Figure 3 shows the definition of the execution environment. Note that the server name, the source code repository and the output type are all drop down boxes, the contents of which are also read from the metadata. Figures 5 and 6 show the details of adding the metadata for a specific parameter. <<<***The authorization screen>>>***



Name	Type	D
BIP Tree	Folder	Root Folder
Integration Technologies	Folder	
Samples	Folder	Root folder for SAS sample
Shared Data	Folder	

FIGURE 1. THE SAS MANGEMENT CONSOLE



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

Name: SAS Global Forum Example

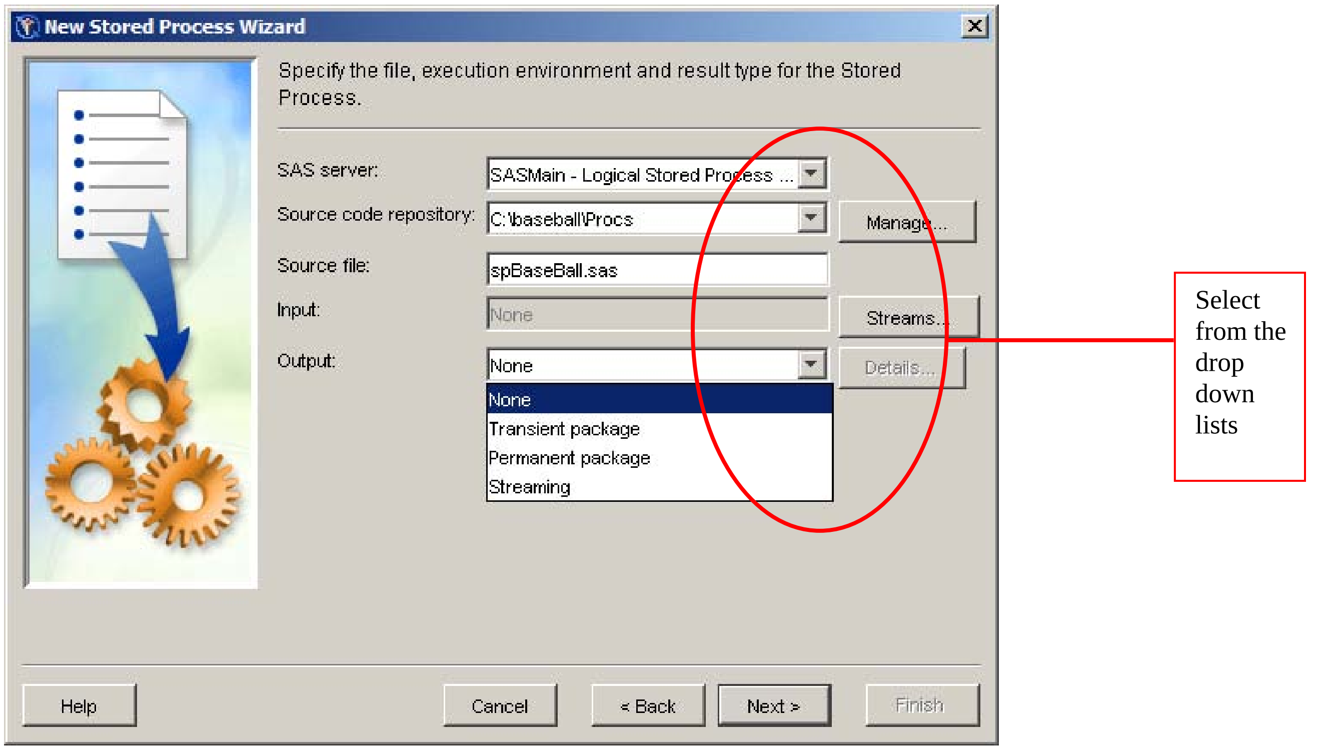
Description: An example SAS Stored Process

Keywords:

Responsibilities:

Name	Role

FIGURE 2: INITIAL DEFINITION OF THE SAS STORED PROCESS



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:\baseball\Procs

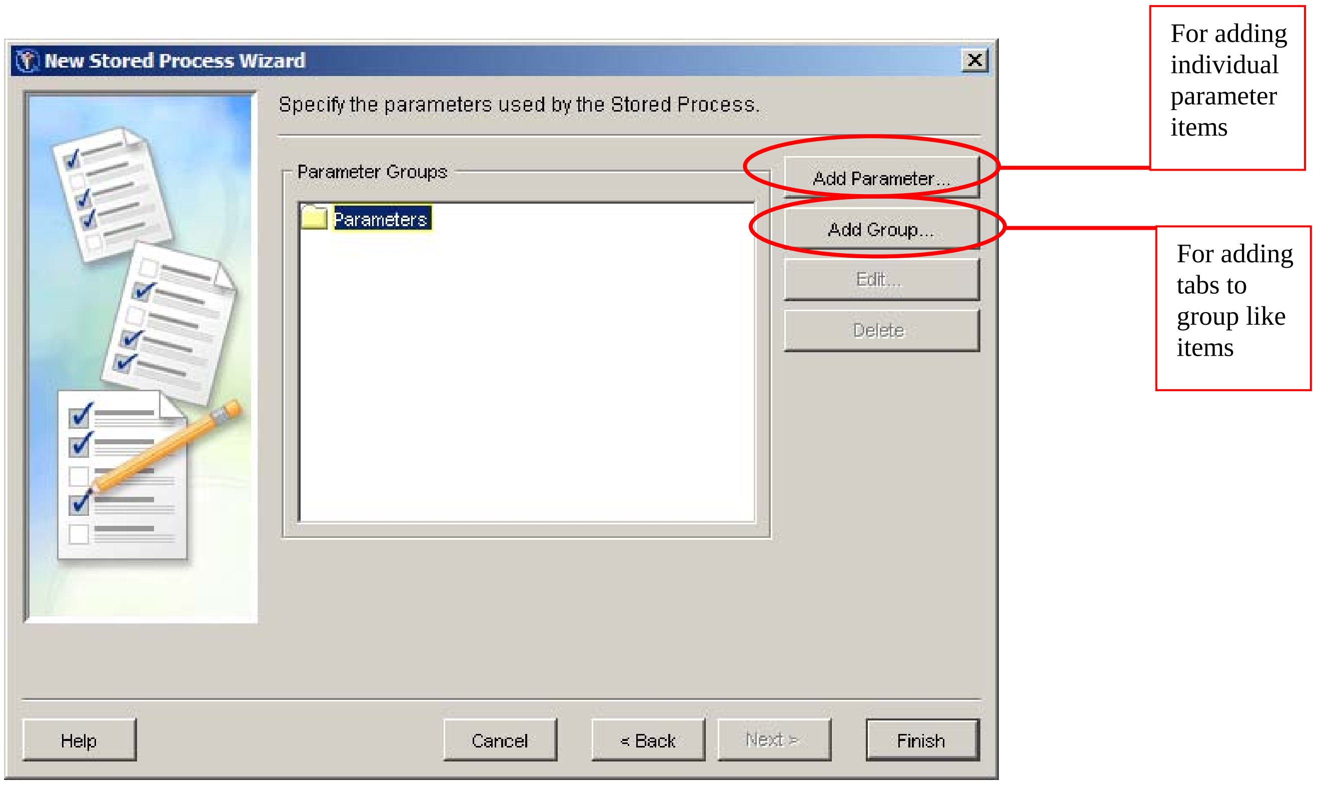
Source file: spBaseBall.sas

Input: None

Output: None

Select from the drop down lists

FIGURE 3: DEFINING THE EXECUTION ENVIRONMENT



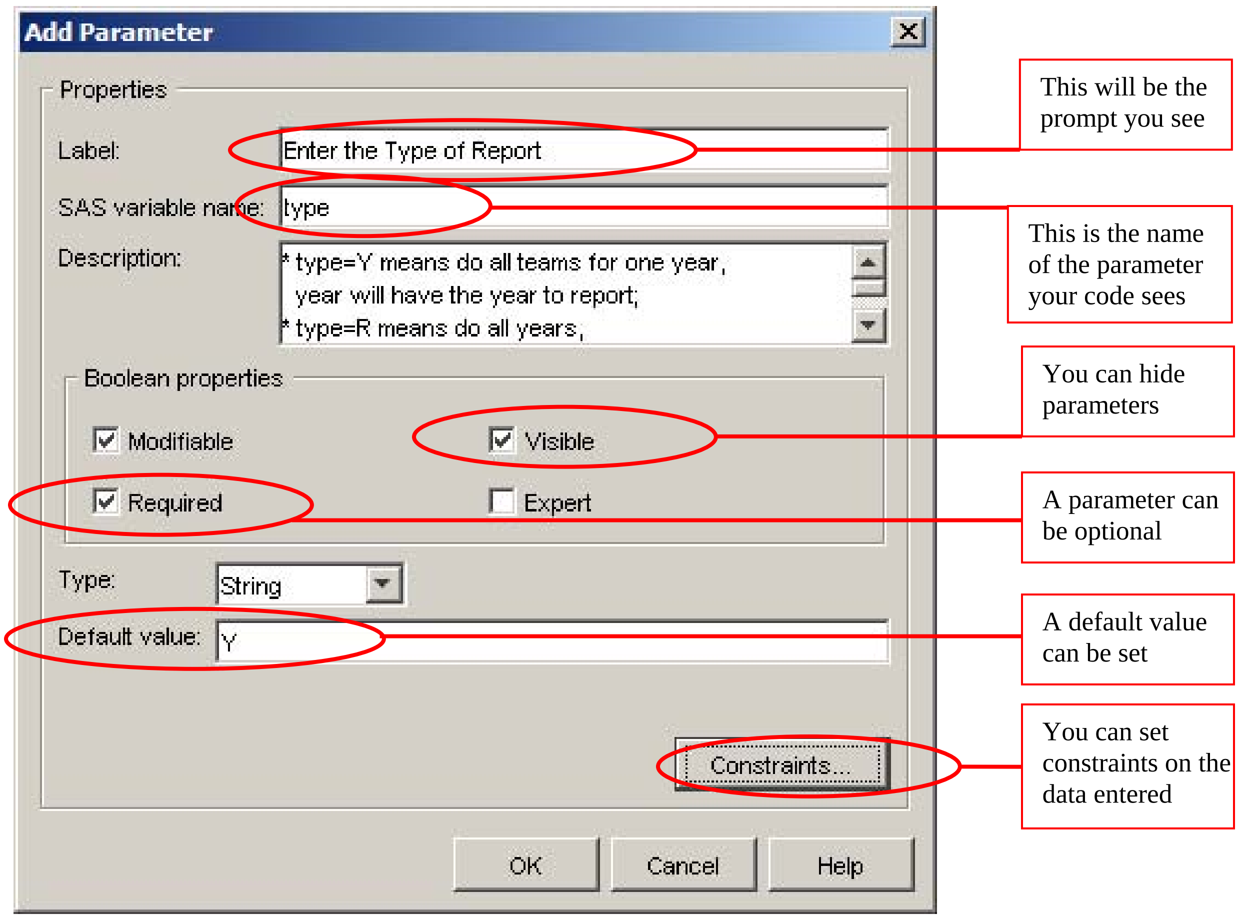
New Stored Process Wizard

Specify the parameters used by the Stored Process.

For adding individual parameter items

For adding tabs to group like items

FIGURE 4: PARAMETER SCREEN BEFORE PARAMETERS ARE ENTERED



Add Parameter

Properties

Label: Enter the Type of Report

SAS variable name: type

Description: * type=Y means do all teams for one year, year will have the year to report; * type=R means do all years;

Boolean properties

☑ Modifiable ☑ Visible

☑ Required ☐ Expert

Type: String

Default value: Y

This will be the prompt you see

This is the name of the parameter your code sees

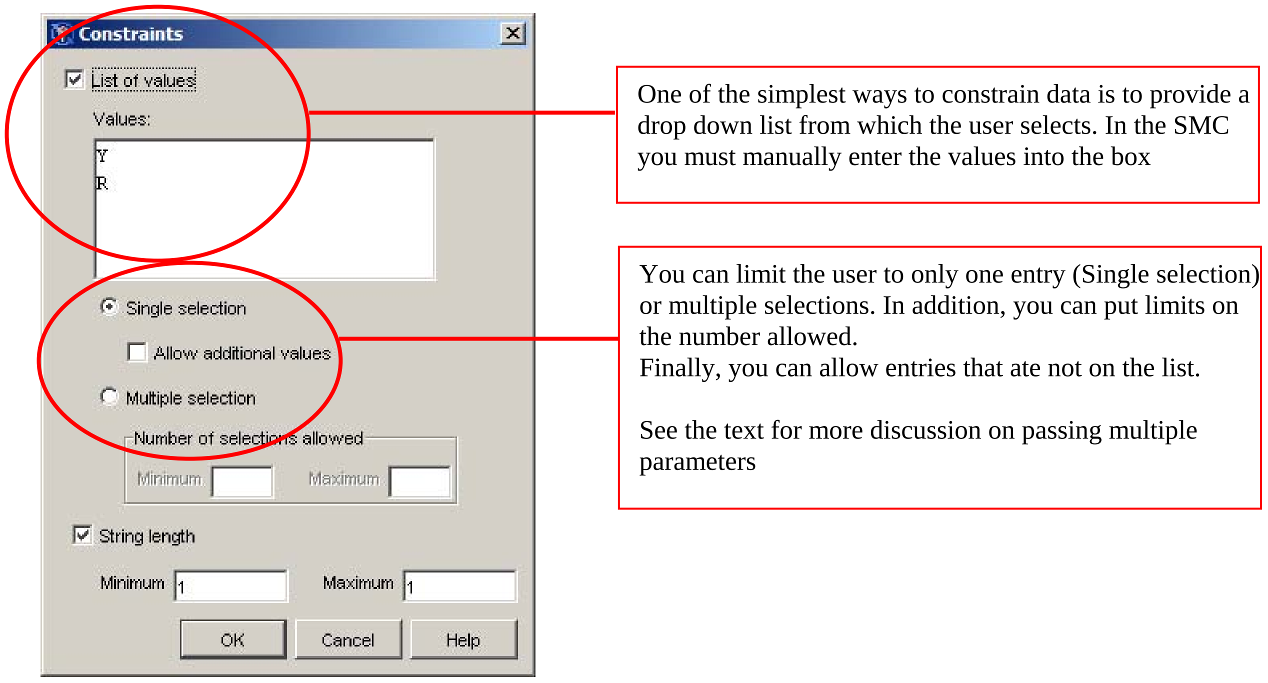
You can hide parameters

A parameter can be optional

A default value can be set

You can set constraints on the data entered

FIGURE 5: PARAMETER VALUES



Constraints

☑ List of values

Values:

Y
R

(●) Single selection

☐ Allow additional values

( ) Multiple selection

Number of selections allowed

Minimum Maximum

☑ String length

Minimum 1 Maximum 1

One of the simplest ways to constrain data is to provide a drop down list from which the user selects. In the SMC you must manually enter the values into the box

You can limit the user to only one entry (Single selection) or multiple selections. In addition, you can put limits on the number allowed.
Finally, you can allow entries that ate not on the list.

See the text for more discussion on passing multiple parameters

FIGURE 6: PARAMETER CONSTRAINTS

The metadata for a SAS Stored Process can also be defined in SAS Enterprise Guide (EG). There are a number of advantages to using EG to define the metadata. The most compelling reason to use EG to define the metadata for a SAS Stored Process is simple: EG is an integrated environment that allows you to write, debug, and test your code, define the metadata for the SAS Stored Process, then execute and test the SAS Stored Process. In addition, if your SAS Stored Process will have a list of values (constraints) you need to enter, the EG interface has a mechanism to load the values from a table.

The main argument against using EG to define the metadata for a SAS Stored Process is management. With the SMC there are usually fewer administrators controlling and managing access to the metadata. If your organization requires more centralized control then the SMC is the tool of choice.

NOTE: SAS9.2

It is expected there will some significant changes to the definition and management of the parameters to a SAS Stored Process in SAS 9.2. Two major complaints about the parameters in a SAS 9.13 SAS Stored Process are:

1. The lists of values in the constraints are static. That is, once entered they can only be changed by some sort of manual intervention. For lists that change often this is a drawback.
2. There is no cross-field checking on a prompt form. That is, you may want to have two date fields and ensure the second date is on or after the first. This sort of validation has to be performed by the SAS Stored Process code.

Indications are that both of these restrictions will be gone in SAS 9.2.

The next section will discuss some of the coding changes you have to add to your repertoire to create a SAS Stored Process.

STORED PROCESS: SAS CODE

Writing SAS code for a SAS Stored Process is much like writing any SAS code.

- Write, debug and test the code
- Replace constants with macro variables and test the code
- Replace any ‘changeable’ variables with macro variables and test the code
- If appropriate, wrap the code in a macro function and test the code

When writing code for a SAS Stored Process we follow the above process, however there are a few things of which we need be aware. The following code snippet shows the basic structure of a SAS Stored Process.

```
%global type;     /* the type of report, Y year, R for ranking              */
%global year;     /* if type=Y, the year to report, for type=R not required */

/*****************************
example assignments would be:
    %let type = Y;
    %let year = 1992;
   OR
    %let type = R;
    %let year = ;
*****************************/

*ProcessBody;

/* sas code that DOES NOT generate ODS output */

%stpBegin;

/* sas code that DOES generate ODS output */

#stpEnd;
```

The four major pieces of interest are:

1. *%global* statements. Declare the SAS Stored Process parameters as global macro variables. The names you declare here must be the same as the names you enter into the metadata as the parameter names.
2. **ProcessBody* comment. This is the trigger for the Workspace Server to assign the values to the macro variables from the prompt form; any initialization of the macro variables before this line will be overwritten. Note that StoredProcess Server assigns the values to the macro variables BEFORE the SAS Stored Process starts to execute, hence this line is not required for SAS Stored Processes that will run on a StoredProcess server. Even though it may not be required it is a good practice to always include it.
3. *%stpBegin*. This macro function initializes the ODS. You include this call only if your SAS Stored Process will be creating ODS output. If you will not be creating ODS output (an output type of *NONE*) or you write directly to *_WEBOUT* then you do not use %stpBegin.
4. *%stpEnd.* This call goes after the section that creates output. This macro function terminates ODS processing and completes the delivery of the output to the client. When used, both *%stpBegin* and *%stpEnd* must be present.

In the code above the calls to %stpBegin and %stpEnd are both terminated with a semi-colon(;); the semi-colon is required. For more information on %stpBegin and %stpEnd see the SAS 9.1.3 Integrations Technology Developers Guide at http://support.sas.com/rnd/itech/doc9/dev_guide/stprocess/stpmacro.html .

Before we look more into SAS code for dealing with parameters and options, let's turn our attention back to the SAS BI Architecture. .

SAS BI ARCHITECTURE

The SAS BI platform was designed to bring together some of the key components of Business Intelligence:

- Data storage – the ability store and easily access large amounts of data
- Data integration – the ability to access data from disparate sources
- Advanced analytics – mining the past to help see the future

Of course, these have been core strengths of SAS for years. The SAS BI platform extends the core strength of SAS through:

- A common metadata layer
- A scalable n-tier architecture

We have already seen parts of the common metadata layer when we looked at creating a SAS Stored Process. Let's look briefly at the different tiers. For a small organization or a proof of concept exercise all of these tiers can exist on one computer. For large organizations, each tier can be distributed across different computers using one or more operating systems across the tier.

DATA SOURCES

Data sources store your enterprise data. All of your existing data assets can be used, whether your data are stored in relational database management systems, SAS tables, or ERP system tables.

SAS SERVERS

SAS servers perform SAS processing on your data. Several types of SAS servers are available to handle different workload types and processing intensities. The software distributes processing loads among server resources so that multiple client requests for information can be met without delay.

MIDDLE TIER

The middle tier enables data and results to be surfaced to users via a Web browser. This tier provides Web-based interfaces for report creation and information distribution, while passing analysis and processing requests to the SAS servers.

CLIENT TIER

The client tier provides users with desktop access to data and analytic functionality through easy-to-use interfaces. For most information consumers, reporting and analysis tasks can be performed with just a Web browser. For more advanced design and analysis tasks, SAS client software is installed on users' desktops.

.

Since the topic of this paper is SAS Stored Processes, it is time to turn out attention to the servers upon which the SAS Stored Processes run.

SAS SERVERS

In version 8, SAS introduced Integration Technologies (IT) and the Integrated Object Model (IOM). This was the beginning of the goal to separate the interface from the engine. The IOM meant that SAS sessions could be instantiated, accessed, and shut down from within a variety of programming environments. This separation of the interface from the engine became manifest in SAS v9 with the introduction of java based tools such as Data Integration Studio (DI Studio) and the SAS Management Console (SMC). In the Windows desktop world Microsoft .NET environments were used to create SAS Enterprise Guide (EG) and the SAS Add-in for Microsoft Office (AMO). SAS also made available two IOM execution servers:

- Workspace server
- Stored Process server

Although both are fundamentally SAS sessions, not only are they different than your normal desktop display manager SAS session, but also they are different than each other. Before we look at these two servers we will look at how we 'talk' to the servers.

THE OBJECT SPAWNER

The object spawner is the gatekeeper/traffic cop controlling access to the SAS servers. When you submit a request for execution services, for example submit a SAS Stored Process or run code from within EG, the request goes to the Spawner which will then either start up a Workspace server session or connect you to an existing Stored Process server session. The Object Spawner plays a pivotal role in this scalable architecture since it essentially controls how many sessions are started or how many processes are running on a single server session. The Object Spawner can also be configured to know when to start a new Stored Process Server versus directing client requests to an existing one. Thus all server connections, and subsequent disconnects, are controlled through the Object Spawner.

Although the Object Spawner may start and stop servers and direct your code to the appropriate server, there is a crucial difference in how the servers are started.

- A new Workspace server session is started for each request. This session is dedicated to the client submitting the request and is shut down when the processing is complete
- One of the existing Stored Process server sessions is used when a request is submitted. When the processing is complete the session goes back into the pool, waiting to process the next request. When there are no available sessions in the pool the Object Spawner may start new sessions, or it may hold the request until a session is available; this is controlled through the Object Spawner's configuration.

This difference has important security issues that we will look at next, specifically Authentication and Authorization.

AUTHENTICATION

Authentication occurs when your credentials are passed to the authentication provider (typically the operating system) for validation. In simple terms, authentication involves first determining whether you can run processes on the computer, and then if you can, what resources you are allowed to access. Essentially, authentication happens outside the SAS platform,

AUTHORIZATION

Unlike authentication, authorization happens within the SAS platform. A crucial role of the metadata server is to verify user or client process is authorized to use the resources it is requesting. The figure below illustrates the flow between the client and the servers. To run a SAS Stored Process the client , for example EG, first queries the metadata server to determine which SAS Stored Processes you are allowed to execute (1). After you select a SAS Stored process, again the metadata is queried to get the metadata on the SAS Stored Process (1); this includes the location of the source, the server upon which it will run, the input parameters and the type of output it will return. Armed with this, EG would then pass this on to the Object Spawner (2) which would then start a new Workspace server session or direct it to an existing Stored Process server (3), depending upon the execution server defined in the metadata.

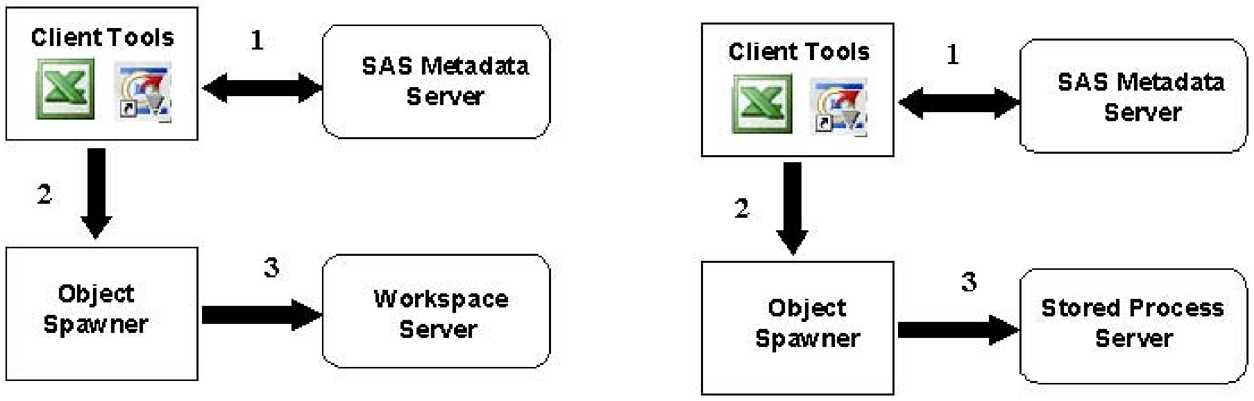


FIGURE 7: EXECUTION FLOW

Since the two flows above look the same, let's look a little deeper into the two servers to see why and when we would choose one over the other. There are differences in the way the two servers handle parameters, in the types of results they can return, and the type of client that can access them. Before we talk about these issues we will look at the issue that is paramount in many organizations – security.

SECURITY

THE WORKSPACE SERVER

The Workspace server is a single user server. This means that every time the Object Spawner needs a Workspace server it starts a new server process; there are exceptions to this, but they are beyond the scope of this paper. One advantage of starting a new server process dedicated to a single user is that the credentials of the user submitting the request can be sent to the operating system for authentication. In other words, all of the security features of the host system are in effect.

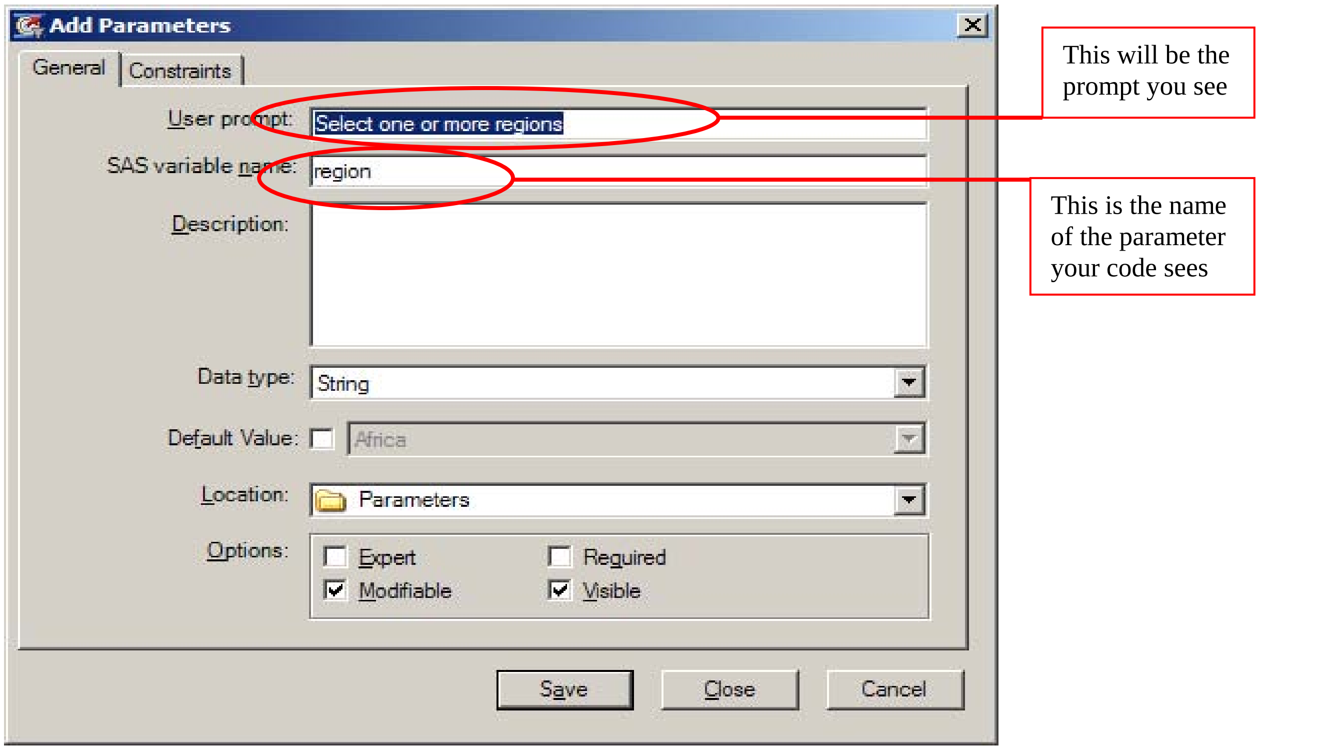
THESTORED PROCESS SERVER

The Stored Process server is a scalable multi-user server. This means a server process is started, then processes requests from any users that needs it services. Usually there are several (3 by default) server processes started and the Object Spawner balances the workload among them. The key here is that the server is ***multi-user***; that means it does not run under the credentials of any of the users using it services, rather it runs under the credentials of a generic user, the SASSRV user by default. For this scheme to work, the SASSRV account must very broad access rights. This means any one user of a Stored Process server has potential access to resources well outside the resources his or her credential would allow. People in security might say few features of the host security system are in effect.

This difference in security may not be as stark as it appears; that is, there can be ways to mitigate the apparent security 'problem'. Although the SASSRV account has wide open access, the users running the SAS Stored Process have been authorized to access and submit the SAS Stored Process though their access privileges in the metadata. If the same due diligence is applied to the metadata privileges as is applied to the host privileges then security issues are minimized, but not eliminated. If you work in an environment where security is paramount, then you will have to look at only using SAS Stored Processes that use the Workspace server

PARAMETERS

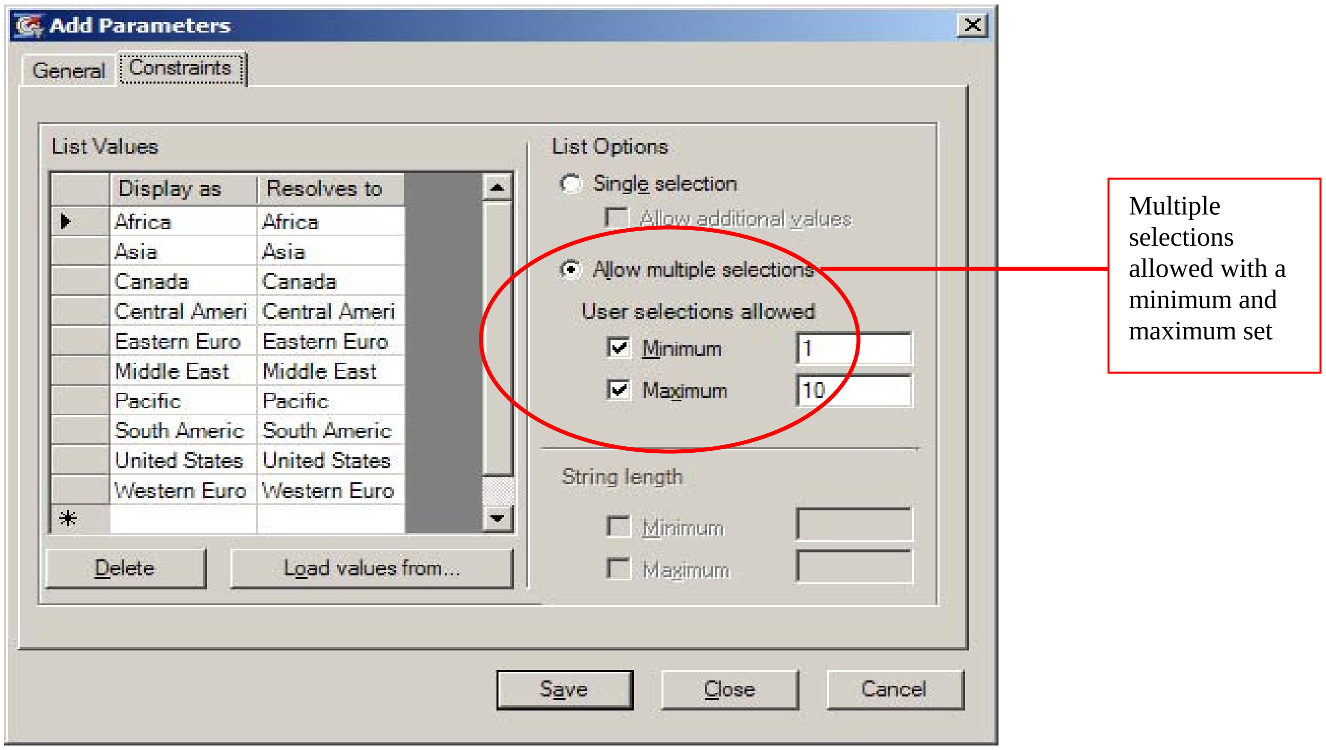
Earlier we showed how parameters can be added to the metadata of a SAS Stored Process. In particular we talked about creating a drop-down list of values from which the user can choose one or more values. When we defined the parameter we provided a parameter name which gets passed into our SAS Stored Process as a macro variable; in Figure 8 below the parameter name is ***region***.


Add Parameters
General
Constraints
This will be the prompt you see
This is the name of the parameter your code sees
Save
Close
Cancel

User prompt: Select one or more regions
SAS variable name: region
Description:
Data type: String
Default Value: ☐ Africa
Location: Parameters
Options: ☐ Expert ☐ Required ☑ Modifiable ☑ Visible

FIGURE 8: PARAMETER DEFINITION(FROM EG)

On the constraints form in Figure 9 the list is a multi-selection list with a minimum and maximum set.



Display as	Resolves to
Africa	Africa
Asia	Asia
Canada	Canada
Central Ameri	Central Ameri
Eastern Euro	Eastern Euro
Middle East	Middle East
Pacific	Pacific
South Americ	South Americ
United States	United States
Western Euro	Western Euro

○ Single selection
☐ Allow additional values
● Allow multiple selections
User selections allowed
☑ Minimum 1
☑ Maximum 10
String length
☐ Minimum
☐ Maximum

FIGURE 9: PARAMETER CONSTRAINTS (FROM EG)

When we run the SAS Stored Process we can select multiple values from the list; notice the message on the form has picked up information from the metadata about the minimum and maximum number of selections we are allowed. Since only one parameter name was provided (***region***), how do all of the selections get passed into the SAS Stored Process?

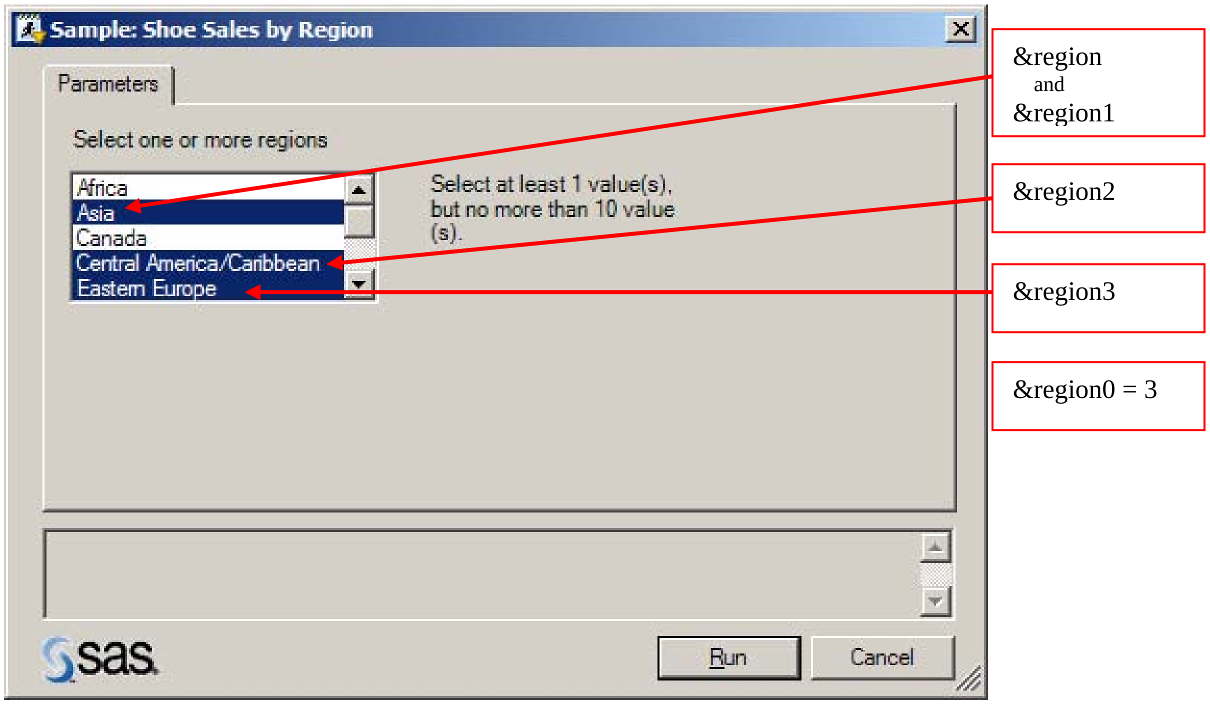

Sample: Shoe Sales by Region
Parameters
Select one or more regions
Africa
Asia
Canada
Central America/Caribbean
Eastern Europe
Select at least 1 value(s), but no more than 10 value (s).
Run
Cancel
&region and &region1
&region2
&region3
&region0 = 3

FIGURE 10: SELECTING VALUES

To differentiate the selections a numeric suffix is added to the macro name. The first one is &region1, the second &region2 and so on. In addition, the base macro name (&region) is also available with the value of the first selection. Finally one more macro variable with the suffix zero (&region0) is passed set to the value of the number of parameters set; from Figure 10 the value of &region0 would be 3. This is summarized as follows:

- &region0 = 3
- &region = Asia
- &region1 = Asia
- &region2 = Central America/Caribbean
- &region3 = Eastern Europe

Like all rules, there is an exception. If only 1 value is selected only the base name (&region) is passed in. Within your code you will have to have logic to deal not only with multiple values but also with only one. The following code snippet shows a simple method to impose some consistency.

```
%if %symexist(region0)
  %then
     %do; %* create global macro variables &region0 and &region1;
       %global region0 region1;
       %let region0 = 0;
       %let region1 = &region;
     %end;
%do i = 1 %to &region0;
    %* some statements using &&region&i;
%end;
```

If you need to use multiple choice parameters in this manner you have limited options. The Stored Process server supports these multiple choice parameters but the Workspace server does not. If the interface allows you to select multiple values, only the last value in the selected list is available in your programme. And again there is an exception to this rule. If you are using the Workspace server and EG 4.1 you can use multi-valued parameters. The following table summarizes this.

Server	**EG 4.1 Client**	**All Other Clients**
Workspace Server	YES	NO
Stored Process Server	YES	YES

RESULT TYPES

Since a SAS Stored Process is a SAS programme it can produce virtually any kind of output. Output that has to be delivered back to the client is known as a ***result type***; there are 4 result types available:

1. None
   - No output is returned to client
   - The default for new SAS Stored Processes
   - Used with SAS Stored Processes that are linked to Information Maps
   - Used when the SAS Stored Processes is created output that stays on the server
2. Streaming
   - Typically used in web applications
   - Only available on the Stored Process server
3. Transient Result Package
   - A container with heterogeneous output
   - Exists only while the client is connected to the server
4. Permanent Result Package
   - Saved to the file system or a webDAV server

If the client application requires streaming output then you have to use the Stored Process server. If your SAS Stored Process will be linked to an Information Map it must use a Workspace server.

SERVER REVIEW

There are a number of points you have to consider when deciding which server will execute your SAS Stored Process. In some cases the decision is made for you. If your SAS Stored Process will be linked to an Information Map then you must use a Workspace server. If your client is a web application that requires streaming output then you must use a Stored Process server. Some points to consider are:

- ***How important is security?*** If you need to run in closed down environment then you will probably have to use a Workspace server.
- ***How many users will be running the SAS Stored Process?*** If the SAS Stored Processes will be used by hundreds of people on a regular basis then you will probably want to use a Stored Process server since it scales better. The Object Spawner can start and stop servers to balance the workload.
- ***How complex is the SAS Stored Processes?*** For complex and long running SAS Stored Processes you might want to use the Workspace server. For long running processes the overhead of starting and stopping a Workspace server is negligible. One the other hand, very short running queries may be better off on a Stored Process server to avoid the overhead of starting and shutting down servers.

MORE SAS CODE

There are a number of reserved automatic macro variables available to you with your SAS Stored Process. For a complete list of these variables see http://support.sas.com/rnd/itech/doc9/dev_guide/stprocess/reserved.html. Some of these macro variables are created by specific client applications submitting the SAS Stored Process while others are available in all SAS Stored processes. In this section I will touch on only a few that I have found useful.

Macro Variable	Used By	Description
_METAPERSON	All	Specifies the Person metadata name that is associated with

		the _METAUSER login variable. The value of this variable can be UNKNOWN. This variable cannot be modified by the client. Useful for creating audits of whom is running the SAS Stored Process.
_METAUSER	All	Specifies the login username that is used to connect to the metadata server. This variable cannot be modified by the client. Useful for creating audits of whom is running the SAS Stored Process.
_GOPTIONS	% stpBegin/ % stpEnd	Sets any SAS/GRAPH option documented in "Graphics Options and Device Parameters Dictionary" in the SAS/GRAPH Reference in SAS Help and Documentation. You must specify the option name and its value in the syntax used for the GOPTIONS statement. For example, set _GOPTIONS to ftext=Swiss htext=2 to specify the Swiss text font with a height of 2.
_ODSDEST	% stpBegin/ % stpEnd	Specifies the ODS destination. The default ODS destination is HTML if _ODSDEST is not specified.
_ODSOPTIONS	% stpBegin/ % stpEnd	Specifies options to be appended to the ODS statement. Do not use this macro to override options defined by a specific macro variable. For example, do not specify ENCODING=value in this variable because it conflicts with _ODSENCODING. Note: NOGTITLE and NOGFOOTNOTE are appended to the ODS statement as default options. You can override this behavior by specifying GTITLE or GFOOTNOTE for _ODSOPTIONS
_ODSSTYLE	% stpBegin/ % stpEnd	Sets the ODS STYLE= option. You can specify any ODS style that is valid on your system.

The _METAPERSON and _METAUSER macro variables are useful when you want to create an audit log of when and who ran the SAS Stored Process. In addition they can be part of TITLE or FOOTNOTE statements as a documenting feature,

The _GOPTIONS and the _ODS* macro variables are used to control output; they are only used if you are using %stpBegin and %stpEnd. If you want to change the default values for these variables then they must be set BEFORE the call to %stpBegin.

DEBUGGING YOUR SAS STORED PROCESS

Writing SAS code for a SAS Stored Process is much like writing any SAS code.

- Write, debug and test the code
- Replace constants with macro variables and test the code
- Replace any 'changeable' variables with macro variables and test the code
- If appropriate, wrap the code in a macro function and test the code

In general if you do this you will not have problems with your code when it runs as a SAS Stored Process. However, if you do you development and testing in a Display Manager SAS session you might run into some problems keep in mind that a SAS server is not exactly the same as desktop SAS. In my experience I have found that server environment is less forgiving when my programming becomes sloppy. Doing the development, debugging and testing in EG can help since EG is using a Workspace server to run the code. When you run into problems executing a SAS Stored Process you need to determine where the problem might be. The problem could be in:

- The server environment
- The client environment
- The metadata definition
- The SAS programme being executed

As with all SAS programming errors, you want to look at the log. Since there are many possible sources for an error, there are many possible logs you can look at. There are two basic categories of logs:

1. Server logs
   - Normally you will have to get a SAS administrator involved to make the logs available to you
2. Client logs
   - EG has both task logs and a project log. You should examine both

SERVER LOGS

All the SAS servers can generate logs. In a production environment you want to minimize the amount of logging so you do not adversely impact performance. In addition, even with minimal logging the log files should be archived and deleted on a regular basis to help avoid space problems on the server. Changing the server log options must be done by a SAS administrator. Changing the logging options usually means stopping and restarting the object spawner; this cannot be done lightly since it will impact all users.

Altering the configuration and logging options is a task that needs to be done by a SAS administrator. As such it is well beyond the scope of this paper. You need to be aware that detailed logging from the servers can be available should you require it.

CONCLUSION

Understanding how to write SAS Stored Processes requires understanding the SAS Business Intelligence framework. In the paper we reviewed the basic definition and requirements of a SAS Stored Process and specifically looked at how to create the metadata component. From there we looked at the two SAS servers that can be used to execute a SAS Stored Process. Some of the features of the two servers were compared with particular emphasis on the security strengths and limitations of the two servers. We touched on some of the automatic macro variables that are available while the SAS Stored Process is executing. Finally we briefly talked about the possible sources of problems and places to look when your SAS Stored Process does not execute.

REFERENCES

Dhillon, Rupinder and Peter Eberhardt "%STPBEGIN: How Enterprise Guide® Almost Removed the L-word from My Relationship With SAS®"
Proceedings of SAS Global Forum, Orlando FL 2007

Eberhardt, Peter "Rev Up Your Spreadsheets with some V8 Power"
Proceedings of SAS User Group International, Orlando FL 2002

SAS Institute Inc. 2006. *SAS® 9.1.3 IntegrationTechnologies: Developer's Guide, Fifth Edition*. Cary, NC: SAS Institute Inc.

CONTACT INFORMATION (HEADER 1)

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

Peter Eberhardt
Fernwood Consulting Group Inc.
288 Laird Dr
Toronto, ON M4G 3X5
Canada
E-mail:peter@fernwood.ca
Web:www.fernwood.ca