**Paper 022-2007**

# %STPBEGIN: How Enterprise Guide® Almost Removed the L-word from My Relationship With SAS®

Rupinder Dhillon, Dhillon Consulting Inc., Toronto, ON

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

## ABSTRACT

Somewhere in the SAS Enterprise Guide documentation we read how easy it was to create a SAS stored process - write and test your code, add a simple %stpbegin and %stpend and let SAS Enterprise Guide take care of the rest. Presto, a new stored process ready to conquer the enterprise. Although that might work in the Hello World example, our first stored process project proved that applying the same guidelines to a more complex SAS program was much more challenging. In reality, creating stored processes is more involved than adding the basic syntax, input parameters and macro variables. You must also be aware of the differences that exist between SAS Enterprise Guide and PC SAS sessions, the importance of efficient coding, and the distinctions between the Workspace and Stored Process servers. This paper will step through some "gotcha's" that may stump you when converting existing code or creating new stored processes from scratch.

## INTRODUCTION

There is a real siren song to the concept of unadorned access to parameterized SAS programs; the sort of access that provides an intuitive GUI dialogue box prompting the user to select from a drop down list of inputs or simply type in a few values. All that is left is to do is press 'Run' and watch as the results unfold in front of your eyes. In this paper we will discuss how we fell in love with the siren song, how the love turned bitter sweet, and how we kept the love alive by remembering our old flame, SAS. We will start by briefly describing the programs we were tasked with converting to Stored Processes. In order to better understand the SAS Stored Process environment, we will then provide an overview of the SAS Business Intelligence (BI) framework. With this background in place, we will describe some of the challenges we encountered and the lessons we learned along the way.

## THE PROBLEM

Our story starts with complex set of financial forecasting models developed by some very clever mathematicians and statisticians. Our job was to convert the SAS code in these models into more generalized programs where all the 'hard coded' values, and there were lots of them, were converted to macro variables so the model could be more easily and consistently run by the modelers. Centralizing the parameters as global macro variables would provide two benefits. First, it would allow the modelers to change the parameters without having to search through the thousands of lines of SAS code for each of the values they had hard coded. Secondly, it would help to formalize and document the model.

This initial conversion was successful; the model became completely data driven. Using PC SAS to access data on the network, the modelers could turn around forecast requests in well under an hour where previously it took the better part of a day. However, there were a large number of parameters and a very good understanding of the model was required to correctly assign the macro variables before running the model. Moreover, a number of these parameters were set up to control output so the modelers could validate the run, and to allow data over-rides for more complex scenarios.

With the initial conversion in place and the demand for model results increasing, the decision was made to make the model widely available; business analysts across the organization would have access to the model directly, thus freeing the modelers from the task of running models allowing them to return to the task of developing models. The strategic decision of the organization was to move to the SAS BI architecture and to have SAS Enterprise Guide be the tool of choice to access and analyze data, so the decision was made to convert the model into a SAS Stored Process. Before we describe the issues we encountered in converting the model, let's look at the SAS BI architecture.

## SAS BI ARCHITECTURE

The SAS Intelligence Platform architecture is designed to efficiently access large amounts of data, while simultaneously providing timely intelligence to a large number of users. The platform uses an n-tier architecture that enables you to distribute functionality across computer resources, so that the resources that are best suited to the job

perform each type of work.

You can easily scale the architecture to meet the demands of your workload. For a large company, the tiers can be installed across a multitude of machines with different operating systems. For prototyping, demonstrations, or very small enterprises, all of the tiers can be installed on a single machine.
The architecture consists of the following four tiers:

#### DATA SOURCES

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

#### SAS SERVERS

SAS servers perform SAS processing on your enterprise 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 intelligence data and functionality 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 intelligence data and 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.

Now that we've set the stage, we're ready to take a closer look at what's involved in creating your first Stored Process.

## A TALE OF TWO SERVERS – STORED PROCESS SERVER VS. WORKSPACE SERVER

One of the first things that you should decide before you get started is whether your Stored Process will run on Workspace server or a Stored Process Server. In our case, we knew that stored processes could be run on either, but without understanding the more intricate differences between the two, we really didn't foresee how running on one versus the other would affect the way in which the Stored Process should be coded and developed. It is therefore important to understand the differences between the two before deciding on one over the other and especially before you start your coding.

Before we jump into some of the differences between the Workspace and Stored Process servers, it is worthwhile to note the mechanics of these two servers in the context of SAS' Integrated Technologies. Both servers are execution servers and allow for distributed processing through various clients – in plain English, both servers allow you to run your SAS programs from different clients (ie.SAS Enterprise Guide, Excel, a web interface) without actually needing SAS on your local computer. You may also hear them commonly referred to as IOM (Integrated Object Model) Servers. Your program of choice, in our case SAS Enterprise Guide, communicates with an IOM Spawner, which in turn connects to either server and allows you to run your SAS code.

#### THE OBJECT SPAWNER

The object spawner controls your link to either of the SAS servers. When you run a stored process or any SAS code, the client submits a request 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 is an important component of this scalable architecture since it essentially acts as a gatekeeper, controlling 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 up a new Stored Process Server versus directing client requests to an existing one. All server connections, and subsequent disconnects are therefore controlled through the Object Spawner which will close and essentially clean up each session once your Stored Process has completed. Here is where the first main difference between the two servers comes in: in the case of the Workspace server, each time the spawner connects, it launches a new server process or SAS session which is dedicated to the SAS process that you are running. In contrast, when running on a Stored Process server you are connecting to a single, continuous SAS session, which you are sharing with other Stored Processes.

**AUTHENTICATION**

The idea of permissions and authentication presents the next major difference between these two servers. We know that one of the basic advantages of the SAS 9 Metadata architecture is the ability to enforce centralized control of data by setting user permissions around data libraries and data access points. Stored Processes also have similar permission settings, which allow you to control who is able to run them. When running Stored Processes on the workspace server, the user's credentials are passed through to the server and are then used to authenticate access permissions. Since credentials are passed at a user level, you are able to put more flexible controls around the data that a person can access, and the stored processes that the person can run. Under the Stored Process server, the user is connecting to a session that is available to numerous clients and users, and is therefore connecting using a general umbrella SAS user account, usually SASSRV. Consequently, this general SASSRV user id must have fairly wide access privileges. This can cause obvious security concerns when it comes to surfacing sensitive data since you lose the ability to restrict access to different groups or users.

**PERMISSIONS**

You are able to control the access that different users have by setting the appropriate *permissions.* Defining permissions allows you to put various restrictions around all of your data and processes such as who can read the data, write to the data source locations, as well as see and change the underlying metadata about your source or process. Furthermore, you are able to put different restrictions on the various data elements. For example, some sensitive data (ie. Employee salaries) should not be readily available to everyone in an organization. You would more than likely want to restrict salary data access to Senior Management while HR should still be able to access other common elements of that data such as the number of active employees. The Metadata architecture allows you to drill down on the levels of the data and set the appropriate permissions on anything that has been defined as a Metadata Object.

In order to facilitate this level of control, your Administrator must define each of your end-user IDs in the Metadata as a 'User'. In addition to defining these individual Users, you can also define common users as a 'Group'. This group will share similar permissions and access and are most likely part of the same division or team within an organization. Once the user IDS are defined in the metadata, you can grant access to these folks so that they may run your Stored Process. As we mentioned earlier, at the moment granting access to your Stored Process to Users and Groups is only possible through the Workspace Server. Future releases may allow the Stored Process Server to authenticate using the User credentials but for now it uses the SASSRV ID.
The following figures depict the process flow involved in running a stored process: (1) The Client Tools will connect to the Metadata Server to authenticate the Users' permissions; (2) Once authenticated, the client will communicate with the Object Spawner; (3) The Object Spawner will then launch the appropriate server session or connect to an existing one.

Figure 1:

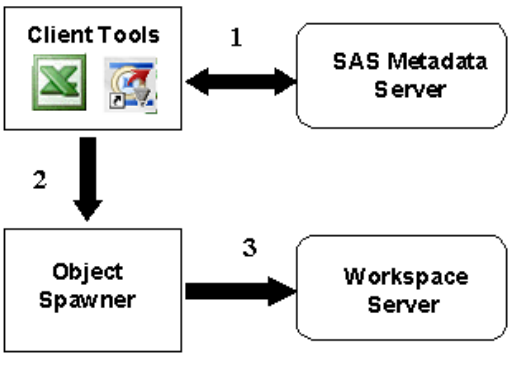
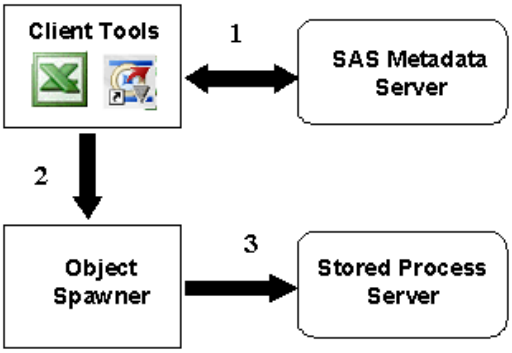



Figure 2:

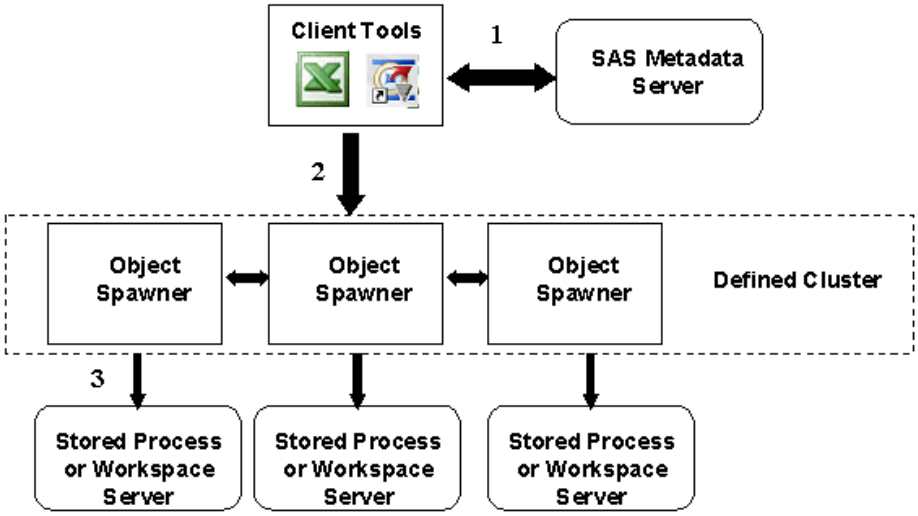



**LOAD MANAGEMENT**

Both the Workspace and Stored Process Servers leverage Load Balancing capabilities to improve the performance of the servers. Load Balancing allows you to share or distribute your SAS processes across all available server machines or server processes and is controlled by the Object Spawner. When configured to run as part of a cluster, the Object Spawner is able to communicate with the other Spawners defined in the cluster to determine which server machine has the most resources available to run your process and then passes along your process to be run. The

difference between the two servers in this context is that the Stored Process Server supports Load Balancing across Server *processes* rather than across server machines. Since these Server Processes can be running on a single machine or across multiple hosts, the Stored Process server can take advantage of Load Balancing even when running on a single machine. Load Balancing on the Workspace Server however, requires that the Servers in the cluster be running on separate machines so load balancing is not available in a single machine environment.

**Figure 3: Load Balancing**




**STORED PROCESS FUNCTIONALITY**

The Stored Process server provides the developer with some additional functionality that is not available when using the Workspace Server. One significant feature that is available with the Stored Process server is the ability to input multiple values for a single parameter. For example, an end user is able to select multiple date items from a drop down list that the developer has provided. The Workspace server does technically allow for multiple input values as well but there's a catch. The resulting macro variable that gets passed to your SAS code only contains the value of the last item selected. All other items that were selected are lost. When running on the Stored Process Server, SAS will automatically generate multiple macro variables corresponding to each of the selections, which are then passed along and made available to your SAS code.

The Stored Process server also allows you to generate Streaming Output results (such as HTML or XML) when developing Stored Processes for use through the Web. The Workspace server does not support streaming output.

**AND THE WINNER IS…**

Having looked at the differences between the two servers, the biggest disadvantage of using Stored Process server is that you give up the ability to place strict access controls since you must run your processes under a general User account. Why then, would you want to use a Stored Process server? Despite this limitation, we found that the Stored Process server still holds some major advantages over the Workspace Server:

- The stored process Server is able to handle multiple values for a single parameter when processing User input. The Workspace server only allows for single value parameters and is only able to process the last value that was specified for that parameter.
- The Stored Process server is completely dedicated to running Stored Processes so you do not have to share your session with other resource intensive SAS processes.
- When running on the Stored Process server, you are connecting to an existing Server session. You therefore avoid the time and system overhead associated with starting up a server session.
- The Stored Process server allows for Streaming input when building processes for use in Web environments.
- You can leverage the load balancing capabilities across Stored Process Server machines. The Workspace server can also handle load balancing however, in order to do so, each of the Workspace Servers must be running on a different machine. The Stored Process server on the other hand, can have multiple Stored Process server processes on a single machine and can therefore balance the workload on the same host
- When running Stored Processes on a web environment, the Stored Process server allows you to save the data and results across Stored Process executions by creating *User Sessions.*

- The Stored Process server allows you to access and run Stored Processes through the Add-in for Microsoft Office.

Given the advantages and disadvantages of each server, there is no clear-cut answer as to the best server to use. It is important to decide up front what server you will be using, as it will significantly influence the rest of your development process. Here are some questions that you may want to consider when deciding which server will best suit your needs.

1. *Who will be running your Stored Process? What permissions will these people need?*
   Will you need different access restrictions at various levels of the data? Are all the end users part of a defined group or will they be needing individual access controls?
2. *What kind of data will they be accessing? Where is it?*
   It is important to identify the restrictions that may need to be placed on your data sources. A Stored process which accesses sensitive data should use the workspace server since you are able to put much tighter controls around the data.
3. *How often will your Stored Process be run? How many people will be running it?*
   If your Stored Process is going to be run quite often by lots of users, it might make more sense to run this on Stored Process server since it is strictly dedicated to Stored Processes. You will not have to share your server with other SAS processes that may tax the server. Since you are connecting to an existing Server session, you also avoid the overhead associated with starting up a new Server session each time the Stored Process is run.
4. *How will your users be surfacing your Stored Process? Through SAS Enterprise Guide, the web, Excel?*
   Stored Processes that are meant for web access are better suited to run on the Stored Process server in order to leverage the use of Sessions and Streaming output.
5. *What types of user inputs will you prompt for?*
   If your Stored Process requires that the user be able to select multiple inputs from a drop-down box, you will need to make sure that you select the Stored Process Server. The Workspace server does not currently support this functionality.

## OUR EXPERIENCE

Given the functionality that our Stored Processes would require, we chose to develop our processes to run on the Stored Process server. We needed the end user to be able to select multiple inputs for several of the parameters. We also wanted the Stored Process to be available to run through the Microsoft Office tools as we foresaw that many of the future users would have the Add-in for Excel easily available to them. We were able to forego the stricter controls that the Workspace Server would have provided since the nature of the data and results being returned made it acceptable to have fairly open access under the SASSRV ID. Given these factors, the Stored Process Server made the most sense.

### SAS ENTERPRISE GUIDE JOINS THE PARTY

As part of our crash course in Stored Processes, this project also presented our first opportunity to use SAS Enterprise Guide as our main SAS Editor tool. We expected that making the switch from PC SAS would be as simple as cutting and pasting the code into a code node and hitting RUN. We were more than a little surprised when the code that had chugged through and worked in PC SAS now had ERROR messages littered all throughout the SAS log and a big red X displayed over our code node, hence the title of this paper. As we worked, diligently, through the various errors and warnings now appearing in our log, it slowly became apparent that SAS Enterprise Guide itself was not the culprit. There were some essential differences between SAS Enterprise Guide and PC SAS that can be summed up in the following underlying ideas:

1. SAS Enterprise Guide is not as forgiving as PC SAS so it's important to follow clean and efficient coding standards, and;
2. You are now running on a Server Environment versus your local machine and there are some basic differences between the two.

Although there were a few instances that could (and would) be blamed on SAS Enterprise Guide 'glitches' or limitations, we found that the majority of the issues that we ran into could be explained, and corrected, in the context of those two aforementioned ideas.

**LESSON 1:**

By default, the 'validvarname' option in SAS Enterprise Guide is set to 'ANY.' Our original code was importing an excel spreadsheet using PROC IMPORT which expected that the validvarname option be set to V7 as it is in PC SAS. Having this option set to 'V7' causes PROC IMPORT to add underscores to column names wherever it comes across a space in the column name. For example, we expected that an Excel column named 'Start Date' would be imported as Start_Date. All subsequent code referred to the variable, start_date. Validvarname set to 'Any' allows for spaces in your variable names so when running our code in SAS Enterprise Guide, the column was imported as 'Start Date' so when we looked for 'start_date', it didn't exist. In this case, we could set the validvarname option back to 'V7' or we could change all references to the variables to read as 'start name'n – we chose to override the option default.

- **Lesson Learned: Check your default options.**

**LESSON 2:**

As is the case with a lot of new software, we did come across some limitations of SAS Enterprise Guide. First off, when running SAS code in SAS Enterprise Guide, the programmer is unable to use traditional methods to stop a data step or the SAS program entirely. Programming methods such as ABORT and END SAS no longer work since these commands force your client to disconnect from the server. When running on your local machine, you can use these commands to stop processing the current data set and continue on with the rest of your SAS program however, in a server environment, your code will bring back an error telling you that you have disconnected from the server. No results will be returned.

- **Lesson Learned: Plan your exit – you can't use ABORT or ABEND in Enterprise Guide.**

**LESSON 3:**

A SAS Enterprise Guide session or job that terminates abnormally can cause some issues on the server. For example, you are running a Stored Process through the Add-in for Excel and, for whatever reason; it appears to have caused your Excel session to hang. Your first instinct is to hit "ctrl-alt-del" to kill that Excel program running on your PC so you can fire up another one and try again. Although you have forced your client to disconnect from the server by closing Excel, it has done so without communicating the appropriate requests to the Object Spawner behind the scenes. The Spawner does not know that the job or Stored Process is supposed to have been stopped so it has not ended your original spawned session on the server. What you're left with is a 'ghost' or 'runaway' session. If you are aware that this has occurred behind the scenes, your administrator can easily terminate the session for you manually. However, given the number of people that could be connecting to these servers, they could potentially become littered with these ghost sessions, leaving a lot of manual clean up to your administrator. You may not be able to eliminate ghost sessions altogether, but you can help minimize the occurrence by using efficient coding standards when developing your Stored Processes or any SAS code.

- **Lesson Learned: Clean up after yourself - killing an EXCEL or SAS Enterprise Guide session does not kill your SAS session on the server.**

**LESSON 4:**

Another symptom of our coding problems was the fact that SAS Enterprise Guide was running very slowly and would sometimes hang altogether. Our code appeared to be running but nothing was actually happening or our SAS log would show that the code had been processed yet the task status would still show that the code was running and no results were being returned. This was a lot tougher to debug since there were no error messages being returned to steer us towards the root of the problem. What we were experiencing in this case was a client memory leak. Given the number of macros and conditional logic loops we were churning though as part of the model, it was important to go back though our code and make sure we were using proper and efficient coding standards. The culprit turned out to be a looping ODS Dataset command with no corresponding ODS CLOSE statement. This was something that we were getting away with in PC SAS but presented a debugging challenge in SAS Enterprise Guide.

- **Lesson Learned: Don't get sloppy – Maintain clean and concise coding standards.**

**LESSON 5:**

By default, the Stored Process wizard will scan your program for all macro variables and prompt you to convert them to input parameters. We knew that not all of our existing macro variables would become input parameters but this did prompt us to evaluate which macro variables we wanted the end user to be able to modify. The original programs were developed for the modelers' use and therefore had numerous modifiable macro variables. As the modelers ran different scenarios and gauged the need for tweaking, they needed to be able to change the macro variable values on the fly. At first glance, we assumed that these macro variables would naturally become the user input parameters. We quickly realized that this would result in the user having to provide extraneous inputs for each run. Instead, we

removed some of the functionality that was specific to the modelers and limited the modifiable options to those that made the most sense to the true end users.

- **Lesson Learned: Know your end user – limit your parameter list to the essentials.**

## CONCLUSION:

Armed with our new understanding of SAS Enterprise Guide, running on Server environments versus locally, as well as the differences between the Stored Process Server and the Workspace server, we were able to tackle the rest of our coding issues. We were able to develop a series of easy to use, self-contained Stored Processes that allowed more people to leverage the work that the Modelers had been doing. We have since shared our 'learnings' with others from our team who have also started converting some of their existing SAS programs into Stored Processes. Although our experience was frustrating and tedious at times, there is a lot of satisfaction that comes with knowing that this initial project has served as a springboard for wider Stored Process usage. People throughout the organization have begun to recognize the benefits of using Stored Processes and are excited about the potential of streamlining and controlling existing procedures. In the meantime, we'll continue to explore possible usages, push the limits and capabilities, and share any 'gotchas' that might pop up along the way.

## REFERENCES


SAS OnlineDoc 9.1.3 for the Web

"Creating and Using SAS® Stored Processes with SAS Enterprise Guide®"
Rossland, Eric & Richardson, Kari. SUGI 30 Proceedings: Paper 135-50
(http://www2.sas.com/proceedings/sugi30/135-30.pdf)


## RECOMMENDED READING


SAS Whitepaper: SAS® Stored Processes: An Introduction and Overview
(www.sas.com)

"SAS® Enterprise Guide® Stored Processes, Part 1: The Information Consumer's View"
Fecht, Marje & Bennet, Peter R. SUGI 31 Proceedings: Paper 257-31
(http://www2.sas.com/proceedings/sugi31/257-31.pdf)

"SAS® Enterprise Guide® Stored Processes, Part 2: Creation and Deployment "
Fecht, Marje & Bennet, Peter R. SUGI 31 Proceedings: Paper 258-31
(http://www2.sas.com/proceedings/sugi31/258-31.pdf)


## CONTACT INFORMATION

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


Rupinder Dhillon
Dhillon Consulting Inc.
50 Portland St. Unit 303
Toronto, ON M5V 2M7
Voice: 416 220 9191
Email: rupinder@dhillonconsulting.com

Peter Eberhardt
Fernwood Consulting Group Inc.
288 Laird Dr.
Toronto, ON M4G 3X5
Canada
Voice: 416 429 5705
Email: peter@fernwood.ca