Paper 023-2007

Converting SAS/IntrNet® Programs to SAS® Stored Processes

Heather Weinstein, SAS Institute Inc., Cary, NC

ABSTRACT

To more fully leverage the capabilities of the SAS®9 Enterprise Intelligence Platform, you can convert existing SAS/IntrNet applications into SAS Stored Processes. Many features are implemented in the SAS Stored Process Server and the SAS Stored Process Web application to minimize the code changes required during a conversion. Existing SAS/IntrNet Application Dispatcher programs can usually be converted to streaming stored processes with minimal or no modifications. This paper explains how to perform such a conversion and discusses some of the differences between Application Dispatcher programs and stored processes.

INTRODUCTION

SAS/IntrNet Application Dispatcher programs execute very much like a stored process Web application. Although Application Dispatcher is only one component of SAS/IntrNet, applications that use Application Dispatcher are the most likely candidates for conversion to stored processes, because these applications execute SAS programs with features that are very similar to stored processes. This paper focuses only on the conversion of Application Dispatcher programs to stored processes.

SAS/IntrNet will continue to be supported, but the stored process framework is a new architecture designed specifically to take advantage of the SAS®9 platform.

You should convert SAS/IntrNet applications to use stored processes if you want to use the following:

- SAS programs in other clients, such as SAS® Enterprise Guide®, SAS® Web Report Studio, or Microsoft Office applications
- the common security model provided by metadata
- the centralized administration provided by metadata integration and SAS® Management Console

Because stored processes and SAS/IntrNet applications are both based on SAS code at the core, you can reuse the majority of your code base. Many features are implemented in the SAS Stored Process Server and SAS Stored Process Web Application to minimize code changes required during a conversion. Existing SAS/IntrNet Application Dispatcher programs can usually be converted to streaming stored processes with minimal or no modifications.

Note: The focus of this paper is on converting SAS/IntrNet programs to stored processes that are intended to run in the SAS Stored Process Web Application. If you want to run these programs in other stored process clients (such as SAS Web Report Studio or the SAS Add-In for Microsoft Office), there might be additional configuration issues. Each client application has its own requirements for stored process behavior.

SAS/INTRNET APPLICATION DISPATCHER OVERVIEW

Application Dispatcher is a component of SAS/IntrNet that enables you to send information from a Web browser to a SAS session for processing, and receive the results on your Web browser. The results can appear as text, HTML, GIF, JPEG, or other formats that are supported by your Web browser.

The Application Broker, which is a component of Application Dispatcher, is written using the Common Gateway Interface (CGI). A Web user can access and analyze data using an Application Dispatcher application by completing an HTML form to select items and specify fields. When the user submits the information, the Application Broker passes the information through the CGI program to a waiting SAS session. SAS software processes the information by using the identified program. Program results return through the CGI to the Web browser and are displayed to the user.

Application Dispatcher exchanges and processes information by using the following components:

- The *input component* runs on the Web server or the client. It typically consists of static or dynamically generated HTML pages that contain URL references or HTML forms. The input component is responsible for selecting what program component to run and what input data and parameters to pass to that program component.

- The *Application Broker* is a CGI program that resides on your Web server (for example, in the cgi-bin or scripts directory). The Application Broker interprets the information received from the input component and passes it to the Application Server.

- The *Application Server* is a SAS session that receives input from the Application Broker. The Application Server accepts information from the Application Broker CGI program and invokes the program component.

- The *program component* is a SAS program invoked within the Application Server. The program performs these tasks:
  1. receives the request from the Application Server
  2. processes the request
  3. returns the results to the Application Broker for delivery to the Web browser and the waiting user

This process is similar to how you execute a typical stored process from the Web. (A Web input form provides parameters to the stored process, which executes a SAS program on a server and returns results.) Because the structure is so similar, it makes it relatively simple to convert most Application Dispatcher programs to stored processes.

For more information about using the Application Dispatcher component of SAS/IntrNet, see *SAS OnlineDoc® 9.13* (SAS Institute Inc. 2006).

SAS STORED PROCESSES OVERVIEW

A stored process is a SAS program that is stored on a server and can be executed as required by requesting applications. Almost any SAS program can be a stored process. You can use stored processes for Web reporting, analytics, building Web applications, delivering packages to clients or to the middle tier, and publishing results to channels or repositories. Stored processes can also access any SAS data source or external file and create new data sets, files, or other data targets supported by SAS.

The stored process concept becomes even more powerful when you consider that these SAS programs can be invoked from multiple client contexts. For example, you might deploy Java applets and Windows-based applications that invoke your stored processes. If your strategy is to use a multi-tiered architecture, you can use Enterprise JavaBeans (EJB) technology, for example, to invoke the same stored processes from an application server.

Stored processes consist of both metadata and physical source code. The metadata reference is stored in the SAS Metadata Server. The source code is typically stored on the stored process server. You can use the BI Manager plug-in for SAS® Management Console to register and manage stored processes. After the stored process is registered in metadata, it can be accessed by other SAS client applications.

Note: Other clients, such as SAS Enterprise Guide and SAS® Data Integration Studio, can also be used to create and register stored processes. This paper shows how to use only the BI Manager plug-in to register stored processes.

The SAS Stored Process Web Application exchanges and processes information in a way that is similar to the Application Dispatcher process. The SAS Stored Process Web Application uses the following components:

- The *custom input form*, or in some cases the *property sheet*, runs on the Web server or the client. It typically consists of static or dynamically generated HTML pages that contain URL references or HTML forms. The input form is responsible for selecting what stored process, or program, to run and what input data and parameters to pass to that program.

- The *SAS Stored Process Web Application* is a Java servlet that resides on the servlet container. It interprets the information received from the input form or property sheet and passes it to the stored process server.

- The *stored process server* is a SAS session that receives input from the SAS Stored Process Web Application. The stored process server accepts information from the client and invokes the stored process.

- The *stored process* is a SAS program that is defined by metadata and is stored on the stored process server. The program performs these tasks:
  1. receives the request from the SAS Stored Process Web Application
  2. processes the request
  3. returns the results to the SAS Stored Process Web application for delivery to the Web browser and the waiting user

COMPATIBILITY FEATURES

The following features describe similarities between Application Dispatcher and stored processes:

- The SAS Stored Process Web Application (a component of the SAS Web Infrastructure Kit) provides the middle-tier equivalent of the Application Broker. The SAS Stored Process Web Application is a Java-based application and requires a servlet container host such as Apache Tomcat, BEA WebLogic, or IBM WebSphere. See the SAS Web Infrastructure Kit installation instructions for other requirements.

- The SAS Stored Process Server (a component of SAS® Integration Technologies) provides the equivalent of the SAS/IntrNet Application Server. The typical stored process server configuration (a load-balanced cluster) is very similar in functionality to a SAS/IntrNet pool service. New servers are started as demand increases to provide a highly scalable system.

- Streaming output from a stored process is written to the _WEBOUT fileref. The underlying access method has changed, but the functionality is very similar to SAS/IntrNet. ODS, HTML Formatting Tools, DATA step code, or SCL programs can continue to write output to _WEBOUT.

- The Application Server functions (APPSRVSET, APPSSRVGETC, APPSRVGETN, APPSRV_HEADER, APPSRV_SESSION, and APPSRV_UNSAFE) are supported in stored processes except as noted in the "Conversion Considerations" section. In many cases, there are equivalent STPSRV functions that are recommended for new programs.

- The _REPLAY mechanism is supported by the stored process server. The value of the _REPLAY URL has changed, but this does not affect most programs.

- The SAS/IntrNet sessions feature has been implemented by the stored process server. The same SAVE library, session macro variables, and session lifetime management functions are available in stored processes.

CONVERSION CONSIDERATIONS

There are a number of differences in the stored process server environment that might affect existing SAS/IntrNet programs. Use this list as a review checklist for your existing programs.

HTTP HEADERS

- In SAS Stored Processes, HTTP headers cannot be written directly to _WEBOUT using a DATA step PUT statement or SCL FWRITE function. You must use the STPSRV_HEADER (or APPSRV_HEADER) function to set header values. Automatic header generation cannot be disabled with `appsrvset("automatic headers", 0)`.

- The `Location` header record does not currently work with stored processes because the HTTP status code cannot be set, as documented in SAS Technical Support Note SN-V9-018238 (SAS Institute Inc. 2006). While this problem is corrected in SAS 9.2, see the code listed in the support note for a workaround for SAS 9.1.3.

- SAS/IntrNet programs require that HTML Formatting Tools use the RUNMODE=S option, which writes an HTML header directly to _WEBOUT. For stored process programs, you should change the option to RUNMODE=B, or you will get an extra header line in the output.

MACRO VARIABLES

- Unsafe processing is different for stored processes—there is no UNSAFE option. Unsafe characters are quoted instead of removed from the input parameters, so you can safely use the &VAR syntax without worrying about unsafe characters. The following examples work without using the APPSRV_UNSAFE function:

```
%if &MYVAR eq %nrstr(A&P)
%then do something...;
```

Here is another example:

```
data _null_;
file _webout;
put "MYVAR=&MYVAR";
run;
```

APPSRV_UNSAFE works in the stored process server and still returns the complete, unquoted input value. This change might cause subtle behavioral differences if your program relies on the SAS/IntrNet unsafe behavior. For stored processes, you should use the STPSRV_UNQUOTE2 function instead.

- The _REPLAY macro variable does not have the same syntax in stored processes as it did in Application Dispatcher. References to `&_REPLAY` are not recommended for SAS/IntrNet programs, but they can be used in stored processes. The DATA step function `symget('_REPLAY')` does not return a usable URL in a stored process and should be replaced with `"&_REPLAY"`. For example:

```
url = symget('_REPLAY')…;
```

should be changed to

```
url = %str(&_REPLAY)…;
```

However, if you were already using `%str(&_REPLAY)` in SAS/IntrNet, then no change is necessary.

- _SERVICE, _SERVER, and _PORT do not exist for stored processes. You must review any code that uses these macro variables. Usually, they are used to create drill-down URLs or forms. In many cases, this code does not require any change; input values for these variables are ignored.

- In stored processes, _PROGRAM refers to a stored process path and name in the metadata repository folder structure, and not a three- or four-level program name. Any programs that create drill-down links or forms with _PROGRAM must generally be modified to use the stored process path.

CODE DIFFERENCES

- The stored process server cannot directly execute SOURCE, MACRO, or SCL catalog entries. You must use either a “wrapper”.sas source code file or an undocumented sample program called STPRUNEN to execute the catalog entry.

- There is no REQUEST TIMEOUT functionality in stored processes: `appsrvset('request timeout')` is not supported.

- The Application Server functions APPSRV_AUTHCLS, APPSRV_AUTHDS, and APPSRV_AUTHLIB are not supported in stored processes. There are no STPSRV functions that are equivalent to these Application Server functions.

- Stored processes do not support automatic user exit programs such as REQUEST INIT, REQUEST TERM, REQUEST LOGIN, SESSION INIT, SESSION TERM, and SESSION INVSESS. SAS/IntrNet applications that rely on these features require modifications to implement equivalent functionality. In most cases, all relevant stored process source code must be changed to explicitly call user exits when necessary. There is no support for REQUEST LOGIN or SESSION INVSESS functionality in the SAS Stored Process Web Application, although similar functionality can be implemented in a custom Web interface.

- AUTH=HOST functionality is not supported by the stored process server. In Application Dispatcher, this functionality provides the ability for the program to run under the operating system permissions of the client user.

- You cannot upload files with stored processes.

- In stored processes, if you are writing to _WEBOUT using PUT statements while ODS has _WEBOUT open, when you execute the code the PUT statement data might be out of sequence with the data that is generated by ODS. This problem occurs because both your code and ODS are opening the same fileref at the same time. For example, the following code might not always work as expected:

```
ods listing close;
ods html body=_webout path=&_tmpcat
(url=&_replay) Style=Banker;
... other code ...
data _null_;
file _webout;
put '<p align="center"> </p>' ;
put '<p align="center"><b>Test.
If you see this in order, it worked.</b></p>';
run;
... other code ...
ods html close;
```

This code might work in some SAS/IntrNet programs, but it can cause data-order problems even in SAS/IntrNet. This code is more likely to fail in a stored process. This problem can be fixed by inserting your PUT statements before you open ODS, closing ODS while you write directly to the fileref, or using the `ODS HTML TEXT="string"` option to write data. The following code is an example of how you can both close ODS while you write directly to the fileref, and insert your PUT statements before you open ODS:

```
ods html body=_webout (no_bottom_matter)...;
... other code ...
ods html close;

data _null_;
file _webout;
put '<p align="center"> </p>' ;
put '<p align="center"><b>Test.
If you see this in order, it worked.</b></p>';
run;

ods html body=_webout (no_top_matter)...;
... other code ...
ods html close;
```

The following code is an example of how you can use the `ODS HTML TEXT="string"` option to write data:

```
ods listing close;
ods html body=_webout path=&_tmpcat
(url=&_replay) Style=Banker;
... other code ...
ods html text='<p align="center"> </p>' ;
ods html text='<p align="center"><b>Test.
If you see this in order, it worked.</b></p>';
... other code ...
ods html close;
```

OVERVIEW OF CONVERSION STEPS

Conversion of existing programs requires that you complete the following steps:

1. Install and configure the SAS Web Infrastructure Kit, a component of SAS® Integration Technologies that includes the SAS Stored Process Web Application, which is used to emulate the Application Broker.

2. Copy the program to a valid source code repository for a stored process server.

3. Modify the program as required to address the items discussed in the Conversion Considerations section.

4. Register the stored process using the BI Manager plug-in in SAS Management Console.

5. Modify the HTML for any custom input forms. Also, convert any HTML pages that link to your stored process to use the SAS Stored Process Web Application URL syntax.

6. Run the stored process.

EXAMPLE

SAMPLE ENVIRONMENT

The following software makes up the environment for this example:

- The Web server for the SAS/IntrNet portion of the example is Microsoft Internet Information Services (IIS) 6.0.
- Apache Tomcat is the servlet container that is being used for this example. Other valid servlet containers include BEA WebLogic and IBM WebSphere.
- SAS 9.1.3 Service Pack 4.
- SAS Stored Process Server (as opposed to the SAS Workspace Server)
- Windows middle tier
- Windows SAS server tier

ABOUT THE APPLICATION DISPATCHER PROGRAM

THE PROGRAM COMPONENT

The example in this paper uses ODS and the TABULATE procedure to display shoe sales data for a selected region. Here's the SAS code:

```
%global regionname;

ods listing close;
ods html body=_webout;

proc tabulate data = sashelp.shoes format = dollar14.;
title "Shoe Sales for &regionname";
   where (product =: 'Men' or product =: 'Women') & region="&regionname";
   table Subsidiary all,
        (Product='Total Product Sales' all)*Sales=' '*Sum=' ';
   class Subsidiary Product;
   var Sales;
   keylabel All='Grand Total'
            Sum=' ';
run;

ods html close;
```

For the sake of illustration, assume that this SAS code is stored in the location `C:\MySASFiles\intrnet\webtab1.sas`.

THE INPUT COMPONENT

The following HTML is the body for the input component, which is the physical HTML file. For the sake of illustration, assume that this HTML is stored in a file named webtab1.html.

```
<H1>Regional Shoe Sales</H1>
<p>Select a region in order to display shoe sales data for that region by subsidiary
and style.  This sample program uses ODS and the TABULATE procedure.</p>

<HR>

<FORM ACTION="/sasweb/cgi-bin/broker.exe">
<INPUT TYPE="HIDDEN" NAME="_SERVICE" VALUE="default">
<INPUT TYPE="HIDDEN" NAME="_PROGRAM" VALUE="intrnet.webtab1.sas">

<b>Select a region:</b>  <SELECT NAME="regionname">
<OPTION VALUE="Africa">Africa
<OPTION VALUE="Asia">Asia
<OPTION VALUE="Central America/Caribbean">Central America/Caribbean
<OPTION VALUE="Eastern Europe">Eastern Europe
<OPTION VALUE="Middle East">Middle East
<OPTION VALUE="Pacific">Pacific
<OPTION VALUE="South America">South America
```

```
<OPTION VALUE="United States">United States
<OPTION VALUE="Western Europe">Western Europe
</SELECT>

<HR>
<INPUT TYPE="SUBMIT" VALUE="Execute">
<INPUT TYPE="CHECKBOX" NAME="_DEBUG" VALUE="131">Show SAS Log

</FORM>
```

The input component looks like the one shown in Figure 1.

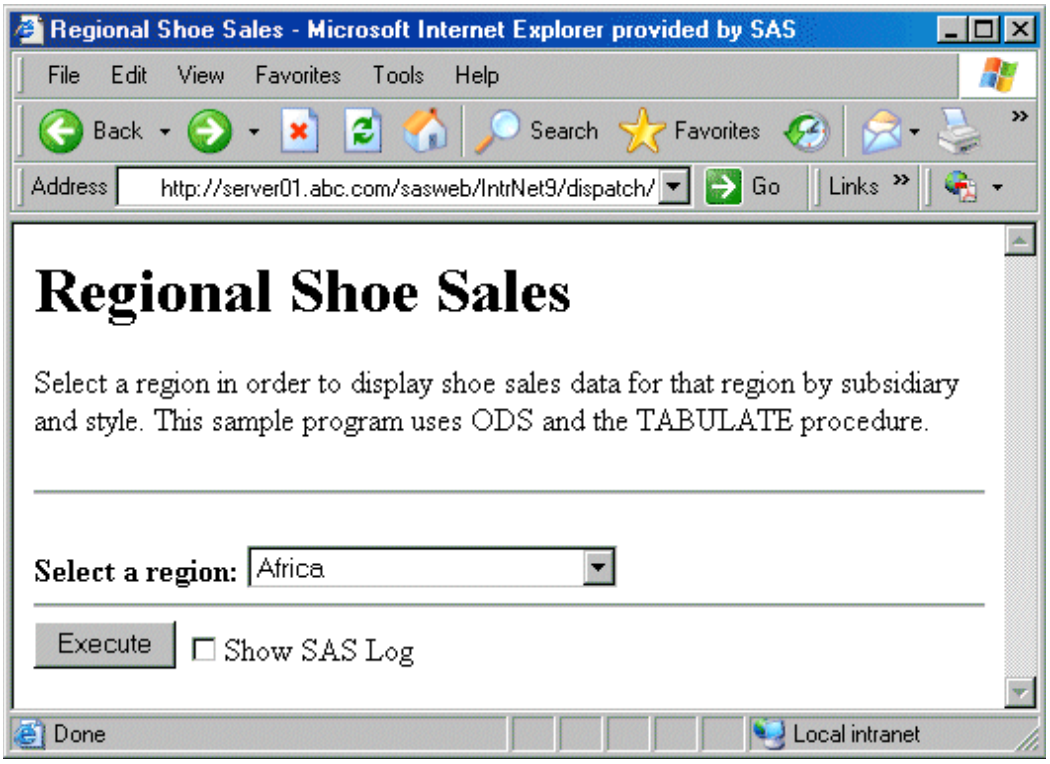
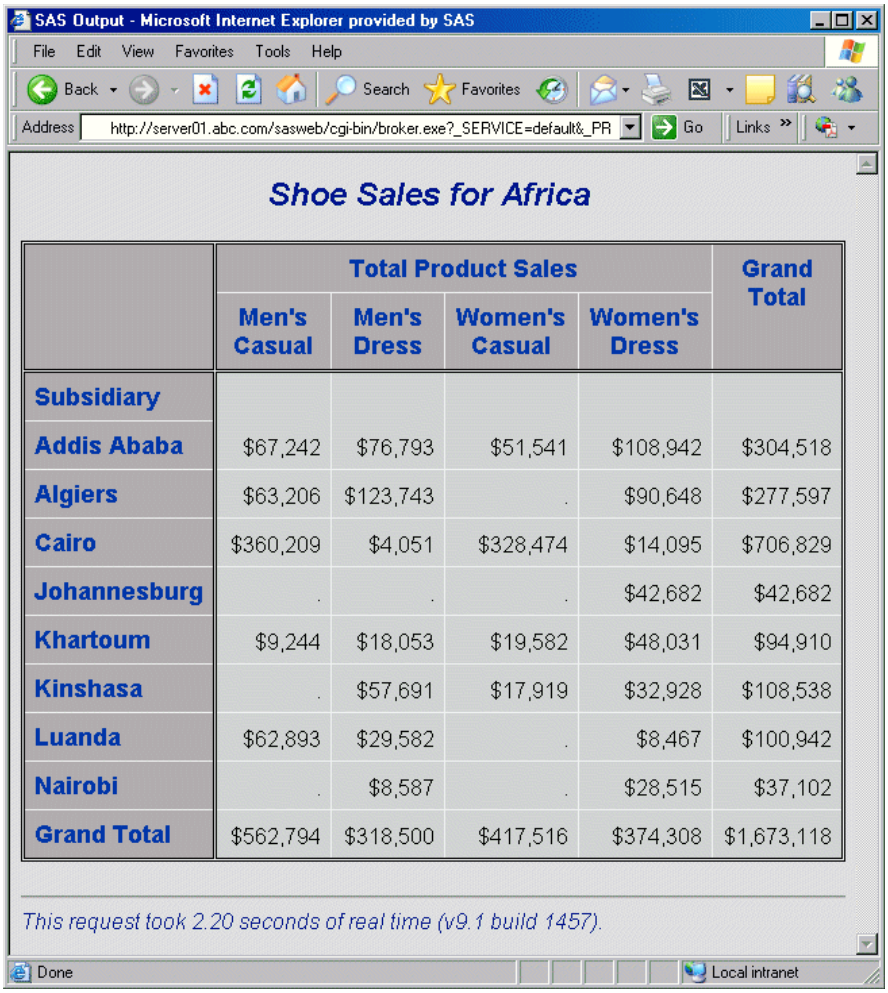


Figure 1. Application Dispatcher Input Component

You can select a region from the list and click **Execute** to display a table of sales data for that region. When you click **Execute**, Application Dispatcher executes the program and sends the results back to the Web browser. The results look like the program output shown in Figure 2.



Shoe Sales for Africa

	Total Product Sales				Grand Total
	Men's Casual	Men's Dress	Women's Casual	Women's Dress	
Subsidiary					
Addis Ababa	$67,242	$76,793	$51,541	$108,942	$304,518
Algiers	$63,206	$123,743	.	$90,648	$277,597
Cairo	$360,209	$4,051	$328,474	$14,095	$706,829
Johannesburg	.	.	.	$42,682	$42,682
Khartoum	$9,244	$18,053	$19,582	$48,031	$94,910
Kinshasa	.	$57,691	$17,919	$32,928	$108,538
Luanda	$62,893	$29,582	.	$8,467	$100,942
Nairobi	.	$8,587	.	$28,515	$37,102
Grand Total	$562,794	$318,500	$417,516	$374,308	$1,673,118

This request took 2.20 seconds of real time (v9.1 build 1457).

Figure 2. Application Dispatcher Program Output

The HTML form created the following URL for the results page, based on the default selections in the input form:

```
http://myserver/sasweb/cgi-bin/broker.exe?
_SERVICE=default&_PROGRAM=intrnet.webtab1.sas&regionname=Africa
```

The URL is typically built for you by the Web browser using fields in an HTML form. The HTML page uses the following FORM tag to submit the program to the Application Broker:

```
<FORM ACTION="/sasweb/cgi-bin/broker.exe">
```

The following hidden fields are used to create name/value pairs to complete the required syntax:

```
<INPUT TYPE="HIDDEN" NAME="_SERVICE" VALUE="default">
<INPUT TYPE="HIDDEN" NAME="_PROGRAM" VALUE="intrnet.webtab1.sas">
```

Notice that the value for _PROGRAM is set to `intrnet.webtab1.sas`. This first level indicates that the program is stored in a directory identified by the intrnet fileref. The next two levels (`webtab1.sas`) provide the name of the program that is executed.

Note: There are several samples that ship with Application Dispatcher that you can use to practice the conversion to stored processes. (Some of these samples have already been converted to stored process samples, and are installed with SAS Integration Technologies.) You can execute the Application Dispatcher samples from the following URL:

```
http://myserver/sasweb/IntrNet9/dispatch/samples.html
```

CONVERTING THE APPLICATION DISPATCHER PROGRAM TO A STORED PROCESS

STEP 1: COPY THE SOURCE PROGRAM

To preserve the functionality of the original SAS/IntrNet example, copy the SAS program to a new location before modifying it. Copy the webtab1.sas program from `C:\MySASFiles\intrnet` to a location on the stored process server, such as `C:\MySASFiles\storedprocesses`.

STEP 2: MODIFY THE PROGRAM AS NEEDED

1. Open the new copy of webtab1.sas to check for conversion considerations.

```
%global regionname;

ods listing close;
ods html body=_webout;

* PROC TABULATE code here;

ods html close;
```

2. Note that the program is already writing to _WEBOUT, so %STPBEGIN and %STPEND are not needed. However, replacing the ODS HTML statement with %STPBEGIN and %STPEND makes it easier to run the stored process from various clients. This replacement also enables you to run the code from a client like the SAS Stored Process Web Application and specify different values for _ODSDEST. For example, you can change the values of _ODSDEST and generate output as PDF, RTF, or PostScript, without making any SAS code changes.

   For this example, delete the two ODS statements at the beginning of the code, and replace them with the following statements:

```
*ProcessBody;
%stpbegin;
```

   Note: The *ProcessBody; statement is not necessary for this program because this example does not use a workspace server, but add it to your program as a best practice. Fewer changes would be necessary if you ever did want to run this stored process from a workspace server.

   Replace the ODS statement at the end of the code with the following statement:

```
%stpend;
```

   Check the list of conversion considerations. No further code changes are necessary for this program to run as a stored process. The stored process now consists of the following code:

```
%global regionname;

*ProcessBody;
%stpbegin;

* PROC TABULATE code here;

%stpend;
```

3. Save the changes and close webtab1.sas.

STEP 3: REGISTER THE STORED PROCESS IN SAS MANAGEMENT CONSOLE

Note: Before you can register a stored process, a server must be defined for the stored process to run on. Converted SAS/IntrNet programs generally should be registered on a stored process server; workspace servers do not support streaming output or a number of other SAS/IntrNet compatibility features. If a stored process server is not already defined, then you can use the Server Manager in SAS Management Console to define a server. For more information about how to define a server, see the Help for the Server Manager.

To register a new stored process, complete the following steps:

1. From the SAS Management Console navigation tree, select the folder under BI Manager in which you would like to create the new stored process. For this example, create a `/Converted Samples` folder under BI Manager.

   To create a new folder under BI Manager, navigate to where you want to put the new folder. Select **Actions** ➧ **New Folder**. The New Folder Wizard appears.

2. Select **Actions** ➧ **New Stored Process**. The New Stored Process Wizard appears.

3. In the New Stored Process Wizard, complete the following steps:

   a. Enter the following information on the first page of the wizard (Figure 3):

      Name: Regional Shoe Sales
      Description: Converted from SAS/IntrNet program.

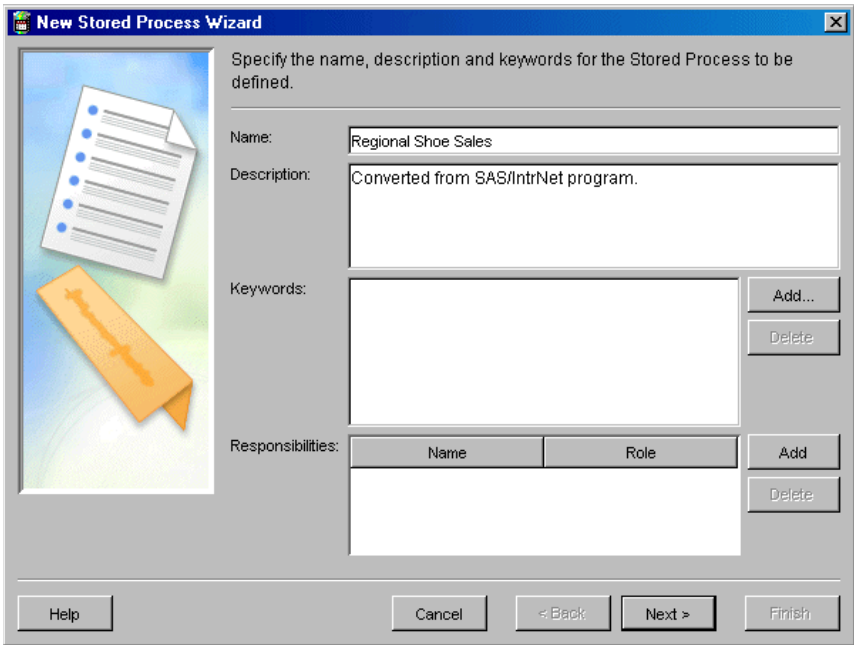


Figure 3. New Stored Process Wizard – Name Specification

   b. Click **Next**.

   c. On the next page of the wizard, specify the following information (Figure 4):

      SAS server: SASMain – Logical Stored Process Server
      Source code repository: C:\MySASFiles\storedprocesses (A source code repository is a location on the application server that contains stored process source code. Click **Manage** if you need to add a new source code repository to the list. For more information about the source code repository, see the BI Manager Help.)

      Source file: webtab1.sas
      Output: Streaming

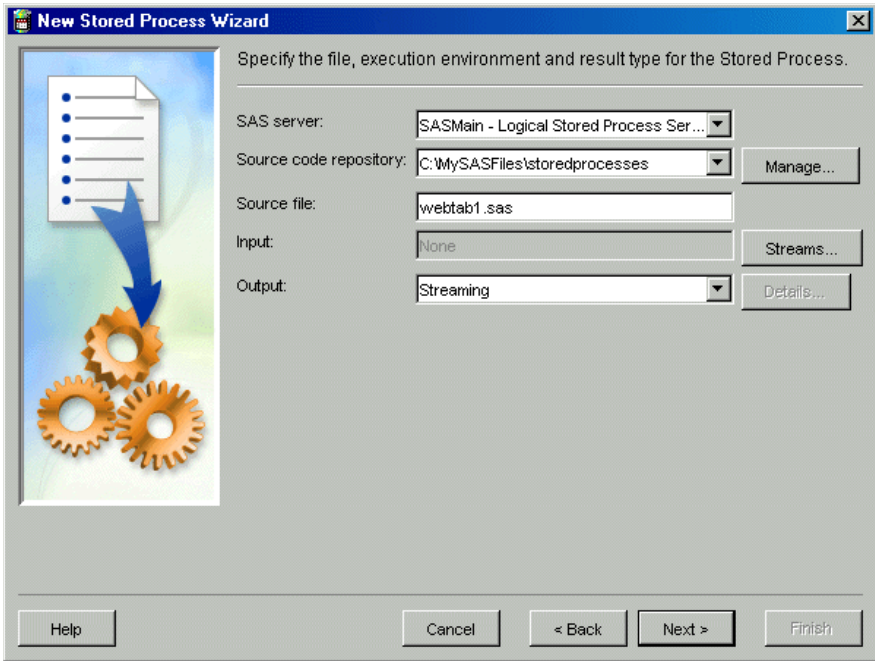


Figure 4. New Stored Process Wizard – Execution Details

d. Click **Next**.

e. The next page of the wizard is where you add parameters. Parameters are optional unless you plan to execute the stored process in other clients that need the metadata information in order to build a property sheet, or if you want to take advantage of the dynamic property page that is built by the SAS Stored Process Web Application (Figure 8). Parameters are also useful if you want to restrict input values or types of input. Do not define any parameters right now.

f. Click **Finish** to register the new stored process.

Note: After you have registered the stored process, use the Stored Process Properties dialog box to control access to the stored process. For more information, see the BI Manager Help.

STEP 4: CREATE A NEW JSP PAGE AND MODIFY THE HTML

To preserve the functionality of the original SAS/IntrNet example, copy the HTML file to a new location before modifying it.

Note: This example shows you how to use the input component from the Application Dispatcher program as a custom input form for the stored process. You will be able to use the _PROGRAM variable along with _ACTION=FORM in the URL to display the custom input form for the stored process. However, copying the HTML file is optional—you can run stored processes without a custom input form.

1. If you want this Web page to be used as the default input form in the SAS Stored Process Web Application, then copy the webtab1.html file to a location under the Apache Tomcat folder, such as `C:\Tomcat4.1\webapps\SASStoredProcess\input\Converted_Samples`. (This location is correct only if the new stored process is registered in the `/Converted Samples` folder in the metadata repository. Otherwise, the path will be different.)

Note: You could also choose to copy the HTML file to a new directory under the IIS Web Server, or to a new directory under the Apache Tomcat folder with the SAS Stored Process Web Application. However, if you decide to do this, you should be aware that appending _ACTION=FORM to the URL to find the custom input form will not work.

2. Modify the HTML page to call the stored process instead of the SAS/IntrNet program.

   a. Open webtab1.html in an HTML editor.

   b. Change the value of the ACTION attribute in the FORM tag to `http://myserver:8080/SASStoredProcess/do`.

   c. Remove the hidden field for _SERVICE.

   d. Change the value of the hidden field for _PROGRAM to the metadata location and name of the stored process: `/Converted Samples/Regional Shoe Sales`.

   e. You can leave the _DEBUG check box with a value of `131`. This is equivalent to the value `LOG,TIME,FIELDS`.

   f. You might also choose to change the text that appears on the Web page. If you want to use images, you need to move them to a location in the current directory or change the tag to point back to the old directory.

   g. The body of the file now contains the following HTML:

```
<H1>Regional Shoe Sales</H1>
<p>Select a region in order to display shoe sales data for that region by
subsidiary and style.  This sample program uses ODS and the TABULATE
procedure.</p>

<HR>

<FORM ACTION="http://myserver:8080/SASStoredProcess/do">
<INPUT TYPE="HIDDEN" NAME="_PROGRAM" VALUE="/Converted Samples/Regional
Shoe Sales">

<b>Select a region:</b>  <SELECT NAME="regionname">
<OPTION VALUE="Africa">Africa
<OPTION VALUE="Asia">Asia
<OPTION VALUE="Central America/Caribbean">Central America/Caribbean
<OPTION VALUE="Eastern Europe">Eastern Europe
<OPTION VALUE="Middle East">Middle East
<OPTION VALUE="Pacific">Pacific
<OPTION VALUE="South America">South America
<OPTION VALUE="United States">United States
<OPTION VALUE="Western Europe">Western Europe
</SELECT>

<HR>
<INPUT TYPE="SUBMIT" VALUE="Execute">
<INPUT TYPE="CHECKBOX" NAME="_DEBUG" VALUE="131">Show SAS Log

</FORM>
```

   h. Save the file as Regional Shoe Sales.jsp, and close it.

Note: If this JSP file is located somewhere other than in the SAS Stored Process Web Application directory, then you need to specify the complete URL to the stored process servlet, as follows, in the ACTION attribute in the FORM tag: `http://myserver:8080/SASStoredProcess/do`. Otherwise, this URL can be a relative link, as follows: `/SASStoredProcess/do`. If you do place the JSP file under the same directory as the SAS Stored Process Web Application, then you need to be careful to preserve the application if you later upgrade or redeploy the SAS Stored Process Web Application.

You should also convert any HTML pages that link to your stored process to use the SASStoredProcess URL syntax. For example, you might use the following URL to link to the Hello World sample program using the Application Broker:

```
http://myserver/cgi-bin/broker?
_service=default&_program=sample.webhello.sas
```

The URL specifies your Application Server, an absolute path to the Application Broker, and the query string (followed by the question mark character). The query string contains the name/value pair data that is input to the application. Each name is separated from the following value by an equal sign (=). Multiple name/value pairs are separated by ampersands (&).The Web page that executes an Application Dispatcher program must pass the _SERVICE and _PROGRAM variables. In this example, the _SERVICE=DEFAULT pair specifies the service that handles this request, and the _PROGRAM=SAMPLE.WEBHELLO.SAS pair specifies the library, name, and type of request program to be executed.

For the SAS Stored Process Web Application, the previous URL would need to be changed. You might use the following URL if you want to run the program from the SAS Stored Process Web Application:

```
http://myserver:8080/SASStoredProcess/do?
_program =/Samples/Stored+Processes/Sample:+Hello+World
```

The URL specifies your stored process server, an absolute path to the SAS Stored Process Web Application (instead of the Application Broker), and the query string. Notice that `/cgi-bin/broker?` has been replaced with the stored process Web application equivalent: `/SASStoredProcess/do?`. The _SERVICE name/value pair is not used with stored processes, and _PROGRAM is the reserved input parameter that specifies the metadata location and the name of the stored process to be executed.

There are special rules for the formatting of name/value pairs in a URL. Special characters (most punctuation characters, including spaces) in a value must be URL encoded. Spaces can be encoded as a plus sign (+) or `%20`. Other characters are encoded using the %*nn* convention, where *nn* is the hexadecimal representation of the character in the ASCII character set. In the previous example, the value `/Samples/Stored+Processes/Sample:+Hello+World` actually identifies the stored process named `Sample: Hello World`. The space in the name is encoded as a plus sign (+). If your parameter values might contain special characters, it is important that they are URL encoded.

STEP 5: EXECUTE THE STORED PROCESS USING THE NEW JSP PAGE

1. You can use _ACTION=FORM in the URL in order to display the custom input form. For example, type the following URL in a Web browser:

   ```
   http://myserver:8080/SASStoredProcess/do?
   _program=/Converted+Samples/Regional+Shoe+Sales&_action=form
   ```

   Your Web browser is forwarded to the following URL, which displays the modified custom input form (similar to Figure 1):

   ```
   http://myserver:8080/SASStoredProcess/input/Converted_Samples/
   Regional_Shoe_Sales.jsp?_program=/Converted Samples/Regional Shoe Sales
   ```

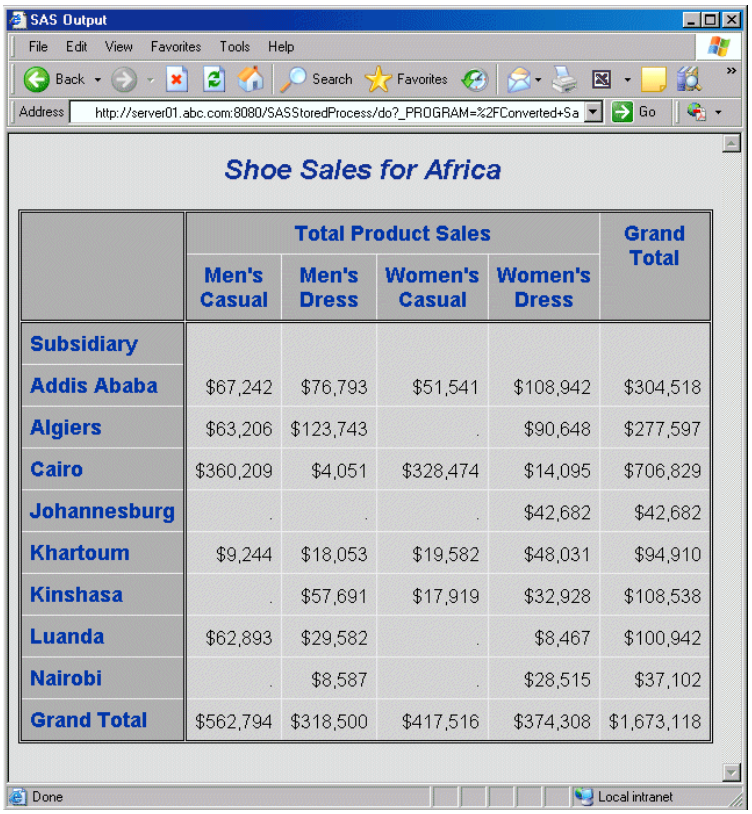
   Note: Be sure to start Apache Tomcat first.

2. Select the default region (Africa) and click **Execute**.

   The JSP page executes the stored process using the following generated URL:

   ```
   http://myserver:8080/SASStoredProcess/do?
   _PROGRAM=/Converted Samples/Regional Shoe Sales&regionname=Africa
   ```

The results look just like the results from the Application Dispatcher program (Figure 5).



Shoe Sales for Africa

	Total Product Sales				Grand Total
	Men's Casual	Men's Dress	Women's Casual	Women's Dress	
Subsidiary					
Addis Ababa	$67,242	$76,793	$51,541	$108,942	$304,518
Algiers	$63,206	$123,743	.	$90,648	$277,597
Cairo	$360,209	$4,051	$328,474	$14,095	$706,829
Johannesburg	.	.	.	$42,682	$42,682
Khartoum	$9,244	$18,053	$19,582	$48,031	$94,910
Kinshasa	.	$57,691	$17,919	$32,928	$108,538
Luanda	$62,893	$29,582	.	$8,467	$100,942
Nairobi	.	$8,587	.	$28,515	$37,102
Grand Total	$562,794	$318,500	$417,516	$374,308	$1,673,118

Figure 5. Stored Process Results

ADDING A PARAMETER TO THE STORED PROCESS DEFINITION

STEP 1: MODIFY THE STORED PROCESS METADATA DEFINITION

Parameter definitions are not required if you are converting a SAS/IntrNet program to a stored process. If there are macro variables in the program that are used to substitute parameter values in the program, you can define them as parameters to the stored process. If you define the value as a parameter, it means that other clients can use the metadata to create a property sheet that prompts for the parameter, or you can take advantage of the dynamic property page that is built by the SAS Stored Process Web application. If you do not define the parameter, it means that the program must use defaults in the code if you want to execute the stored process in other clients. If you intend to use the stored process in other clients, you should define parameters in the metadata.

In webtab1.html and webtab1.sas, the REGIONNAME macro variable is substituted into the PROC TABULATE code. Because the HTML form uses a drop-down list, you can count on a valid value always being passed to the program from that Web page. If you want to make sure this stored process runs correctly in other clients (or if you want to use the dynamic property page that was built by the SAS Stored Process Web Application), then you need to define a parameter that returns a macro variable named REGIONNAME with a valid list of regions.

To add the REGIONNAME parameter, complete the following steps:

1. In BI Manager, open the Stored Process Properties dialog box for the Regional Shoe Sales stored process.

2. On the Parameters tab, click **Add Parameter**.

3. In the Add Parameter dialog box, specify the following information (Figure 6):

   Label: Select a region
   SAS variable name: regionname
   Boolean properties: Modifiable, Required, and Visible
   Type: String

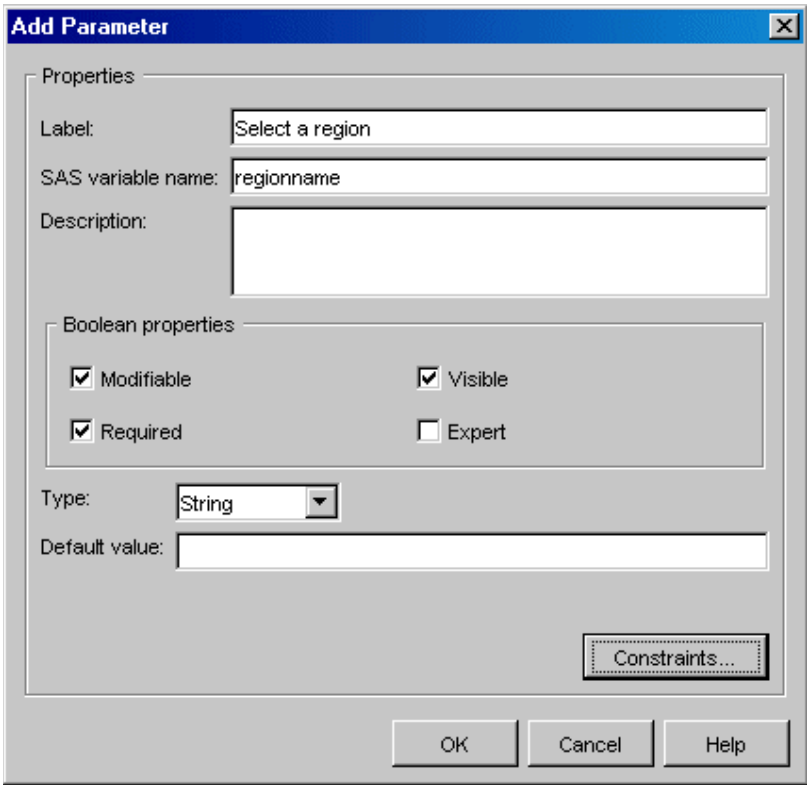


Figure 6. Add Parameter Dialog Box

4. Click **Constraints**. In the Constraints dialog box, specify the following information (Figure 7):

List of values
Values: Africa
Asia
Central America/Caribbean
Eastern Europe
Middle East
Pacific
South America
United States
Western Europe
Single selection

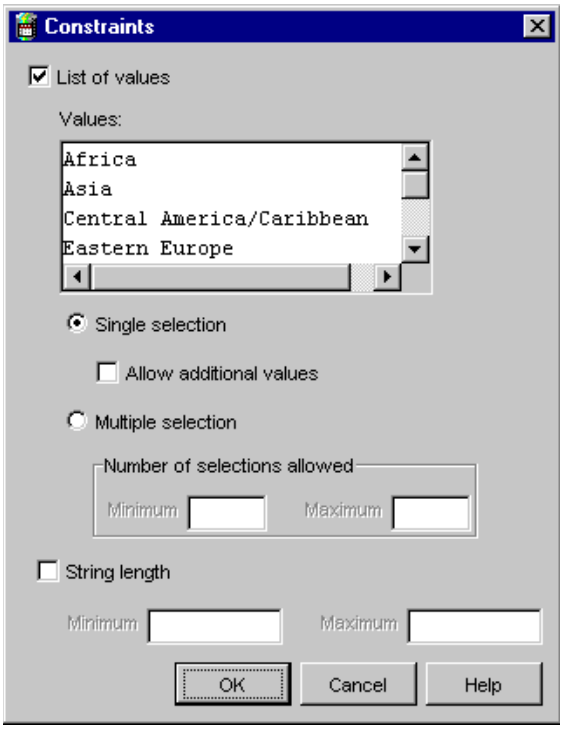


Figure 7. Constraints Dialog Box

5. Click **OK** in the Constraints dialog box, and then click **OK** in the Add Parameter dialog box.

STEP 2: EXECUTE THE STORED PROCESS USING THE PROPERTY SHEET

To view the parameter that you added to the stored process metadata definition, execute the stored process using the SAS Stored Process Web Application property sheet (Figure 8) instead of the custom input form. The property sheet uses the parameter that you defined in the New Stored Process Wizard when you registered the stored process metadata. To access the property sheet for this stored process, type the following URL in a Web browser:

```
http://myserver:8080/SASStoredProcess/do?
_PROGRAM=/Converted Samples/Regional Shoe Sales&_action=properties
```

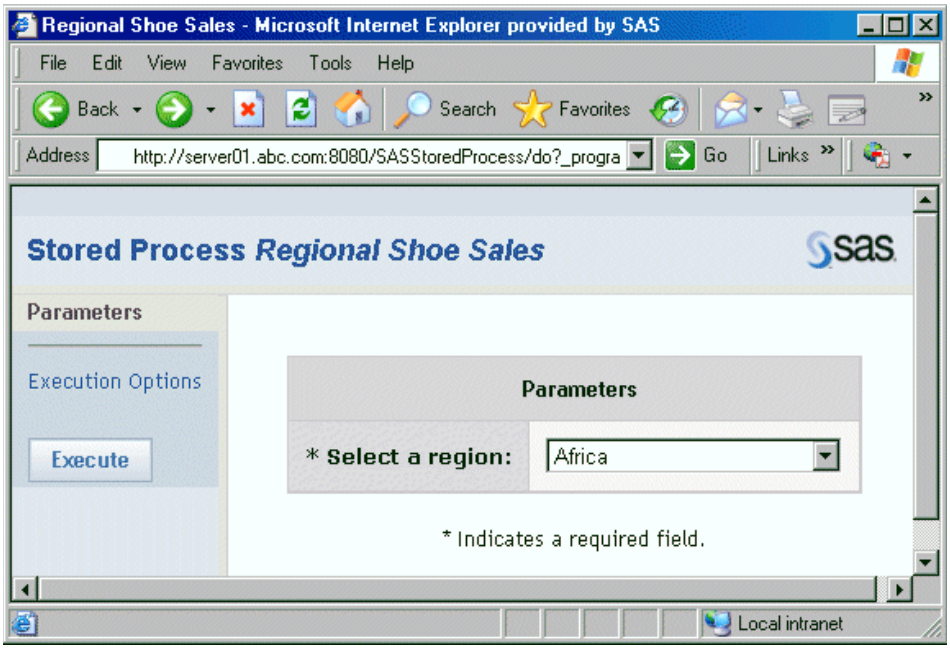


Figure 8. SAS Stored Process Web Application – Property Sheet

Select the default region (Africa) and click **Execute**. You see the same results (the table of shoe sales for Africa that was shown in Figure 5) displayed in a separate Web browser window.

Note: You can click **Execution Options** in the property sheet if you want to change options like the output type, the ODS style for the output, or whether to display the SAS log with the output.

EXECUTING CATALOG ENTRIES

If you are converting SAS/IntrNet programs that use catalog entries, then you need to use either a "wrapper".sas source code file or an undocumented sample program called STPRUNEN to execute the catalog entry. As mentioned in the "Conversion Considerations" section, the stored process server cannot directly execute SOURCE, MACRO, or SCL catalog entries.

You can use a wrapper program like the following to execute SOURCE catalog entries:

```
libname mylib 'sas-data-library';  /* this library could be pre-assigned */
filename fileref1 catalog 'mylib.catalog.program.source';
%include fileref1;
```

The wrapper program for MACRO catalog entries could be something like the following:

```
libname mysas 'SAS-data-library';  /* this library could be pre-assigned */
filename mymacros catalog 'mysas.mycat';
options sasautos=mymacros mautosource;
%macroname;
```

These two sample programs show only the minimum code that is necessary to execute catalog entries from a SAS program. This might be enough in some cases, but you might want to use some other SAS/IntrNet features by including macro variables such as _PGMLIB, _PGMCAT, _PGM, _PGMTYPE, and _APSLIST. These macro variables are included in the STPRUNEN sample program, which can be used to execute SCL and SOURCE catalog entries.

Some functionality of the STPRUNEN sample program is not fully documented or tested. Starting with SAS 9.1.3, stprunen.sas is installed with the SAS Integration Technologies samples in the `!SASROOT\inttech\sample` directory (for Windows). On UNIX, it is in `!SASROOT/samples/inttech/stprunen.sas`.

This program can be registered as a stored process and used to execute SCL or SOURCE catalog entries. It cannot be used to execute MACRO catalog entries. This program might be useful for converting existing SAS/IntrNet programs to SAS Stored Processes. It is generally useful only for stored processes that stream output to _WEBOUT.

To use the STPRUNEN sample program, complete the following steps:

1. Edit the stprunen.sas file and add LIBNAME statements for any libraries that contain the catalog entries you want to execute. If you prefer, you can pre-assign these libraries in the server's SAS configuration file or autoexec file.

2. Use the New Stored Process Wizard in SAS Management Console to create a separate stored process for each catalog entry that you want to execute.

   a. After you specify the name of the stored process, specify the following information on the Execution Details page of the wizard:

      SAS Server: SASMain – Logical Stored Process Server
      Source repository: C:\Program Files\SAS\SAS 9.1\inttech\sample
      Source file: stprunen.sas
      Output: Streaming

   Note: Because this program requires streaming output, you must use the stored process server.

b. On the Parameters page of the New Stored Process Wizard, add a new parameter and specify the following information for that parameter:

Label: Catalog Entry
SAS variable name: _ENTRY
Boolean properties: Deselect Modifiable and Visible.
Default value: Set it to the four-level catalog entry name (for example, MYLIB.MYCAT.MYENTRY.SCL or MYLIB.MYCAT.MYENTRY.SOURCE).

The STPRUNEN sample program requires that the SAS variable name of _ENTRY define the catalog entry that you want to execute. The libref (the first level of the four-level name) must be assigned, either in the STPRUNEN sample program or by some other method.

The stored process now executes the SCL or SOURCE entry.

CONCLUSION

Stored processes offer exciting new possibilities for your SAS/IntrNet programs, as they allow you to integrate SAS intelligence across the SAS platform. By converting your SAS/IntrNet applications to stored processes, you can continue to reap the benefits of your existing SAS investments and extend their usage across the enterprise. This paper gives you tools and considerations to streamline your conversion process.

RECOMMENDED READING

SAS Institute Inc. 2006. *SAS® 9.1.3 Integration Technologies: Developer's Guide, Fifth Edition.* Cary, NC: SAS Institute Inc. Available at support.sas.com/rnd/itech/updates/913/dev_guide_sp4_2.pdf.

SAS Institute Inc. 2006. "SAS/IntrNet 9.1: Application Dispatcher." *SAS OnlineDoc® 9.1.3.* SAS Institute Inc. Cary, NC. Available at support.sas.com/onlinedoc/913/docMainpage.jsp.

SAS Institute Inc. 2006. "STPSRV_HEADER(Location,...) fails to redirect Web client." SAS Institute Inc. Cary, NC. Available at support.sas.com/techsup/unotes/SN/018/018238.html.

CONTACT INFORMATION

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

Heather Weinstein
SAS Institute Inc.
SAS Campus Drive
Cary, NC 27513
E-mail: Heather.Weinstein@sas.com