

Paper 330-2009

## SAS® Stored Processes: Going Beyond the Current Capabilities of the Stored Process Wizard

Joe Flynn, SAS Institute Inc., Cary, NC

### ABSTRACT

As an experienced user of SAS® software, you are likely familiar with SAS stored processes and how to create them. Getting down to basics, a stored process is a SAS program that resides on a server and can be executed as required by a variety of requesting applications. These applications, in turn, can complete a number of different tasks including performing analytic analyses, reporting through the Web, building Web applications, and delivering packages to clients. The SAS® Enterprise Guide® 4.1 Stored Process Wizard has some functionality built in to help you with these tasks. However, SAS Enterprise Guide 4.2 Stored Process Wizard introduces significant improvements to the existing functionality.

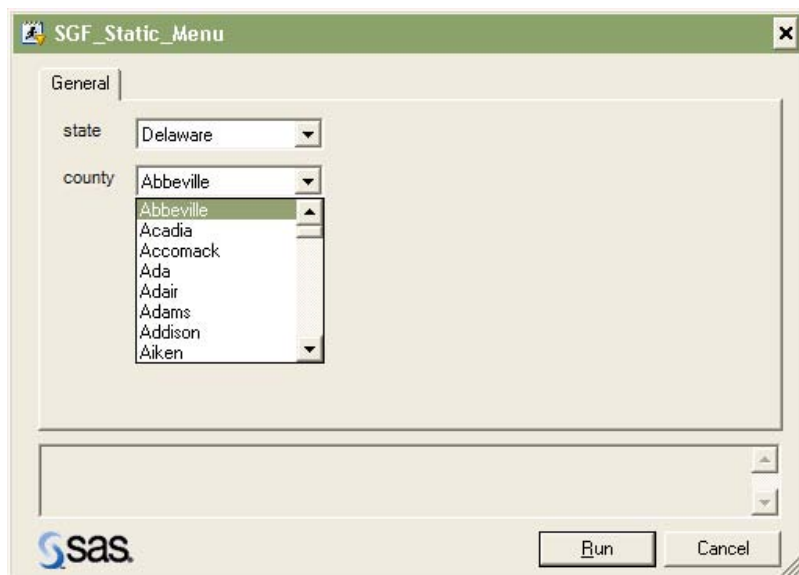
This paper discusses the current capabilities and limitations of using the Stored Process Wizard for user input, as well as the improvements and enhancements introduced with SAS Enterprise Guide 4.2. A practical example is included which addresses the topics of cascading menu prompts and dynamically populated menus.

### OVERVIEW

Many SAS stored processes take advantage of input parameters to add a dynamic aspect to the underlying SAS program. One way to take advantage of input parameters in SAS 9.1.3 is to define them in the Stored Process Manager. When the stored process executes, the user is presented with an input menu that can contain a variety of sub-menus for adding the necessary input. This built-in functionality is very helpful; however, it still leaves some features to be desired.

For example, when using SAS Enterprise Guide 4.1 to create a parameter, input values can be loaded directly from a data set into the metadata. Once loaded into the metadata however, these values become static and are no longer tied to the SAS data set. If the data were to change, these input values would have to be manually updated. This results in undesired maintenance of the stored process. With a stored process, it is more user-friendly to implement sub-setting or cascading menu prompts.

The example in this paper demonstrates the cascading prompts approach using the 2008 Presidential Election voting data for a majority of the counties across the United States. This example provides an excellent scenario in which cascading prompts are useful because there are over 3,000 counties in the United States. Choosing from a county list this large can potentially overwhelm a user. In addition to minimizing the number of choices that are presented, using cascading parameters ensures that a valid choice is made. For example, as shown in Display 1, without sub-setting parameters, a user could potentially select a state and county combination that is invalid.



Display 1. Menu without sub-setting parameters

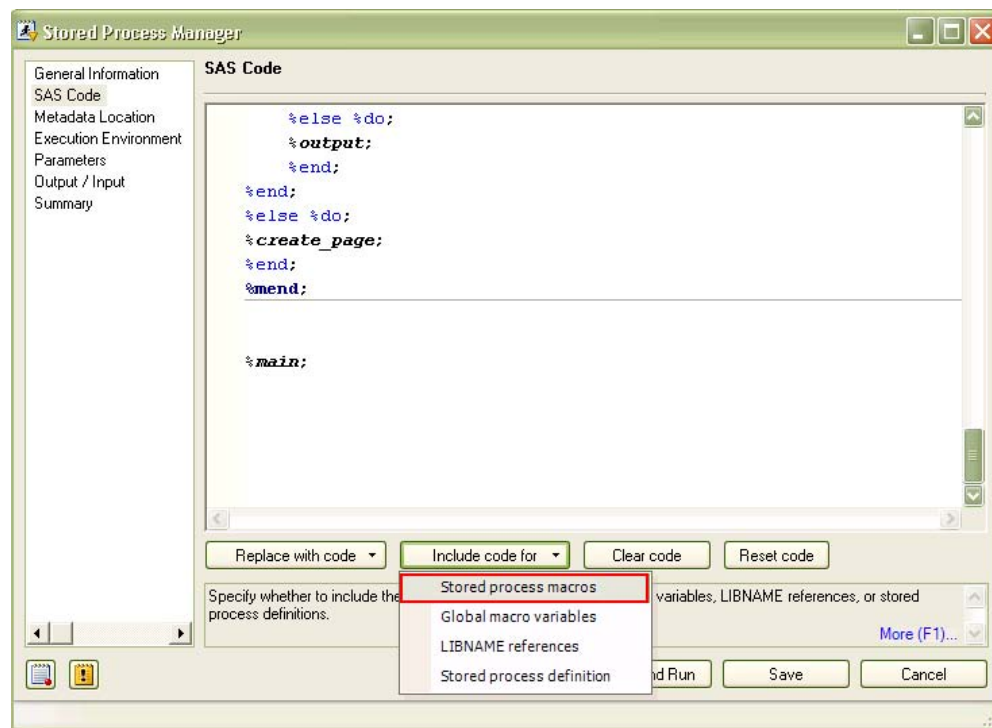
Although sub-setting or cascading menu prompts and dynamically-populated menus are not built into the SAS Enterprise Guide 4.1 Stored Process Wizard, it is still possible to obtain these functionalities. Through the use of SAS Programming Logic, JavaScript, and HTML code, you can generate custom input forms. However, these functionalities are now built into the SAS Enterprise Guide 4.2 Stored Process Wizard, and make it easier to implement cascading menu prompts into all SAS stored processes without the hassle of building a custom Web form. Although these functionalities are built into the SAS Enterprise Guide 4.2 Stored Process Wizard, the ability to build a custom form still has its place, giving the developer complete freedom over the appearance and functionality of the user input form.

The following sections discuss creating a custom input form, followed by a simple walk-through example of using dynamic input parameters in SAS Enterprise Guide 4.2. The SAS data sets used in the following examples are available for download at the Web address provided in the "Resources" section of this paper.

## CREATING A DYNAMIC INPUT FORM WITH THE SAS® ENTERPRISE GUIDE® 4.1 STORED PROCESS WIZARD

When you create a dynamic input form with SAS Enterprise Guide 4.1, you use SAS *programming* logic, along with PUT statements to build a custom Web form. This method employs the \_WEBOUT fileref to do so, which requires the Execution Server to be set to the **Logical Stored Process Server** and the Output Option to be set to **Streaming**. Also, the stored process that you build using this method is executed in a Web browser. Because this method uses the SAS Stored Process Web application servlet and is not prompting directly from the client, this method will not work for clients such as the SAS® Add-In for Microsoft Office.

Because PUT statements are used to build a Web page rather than ODS (%STPBEGIN and END statements are used with ODS), stored process macros need to be disabled. To disable the stored process macros, click the **Include Code For** menu in the Stored Process Manager, and deselect **Stored process macros** (Display 2).



### Display 2. Disabling stored process macros in the Stored Process Manager

The initial execution of this example returns an HTML form to a Web browser. Updates to parameters occur upon changing the value in the **state** box. When this occurs, the stored process Web application calls the stored process again, passing in a new value for the **state** parameter. The code that runs the process contains three macros, each serving a different purpose. For a better overall understanding of how the three macros work, here is an overview:

The **%MAIN** macro executes first. The purpose of this macro is to determine whether to generate the custom Web form or display the results. This is done by checking the value of the refresh\_submit macro variable. If the value of the refresh\_submit macro variable resolves to 1, the %output macro executes and generates results.

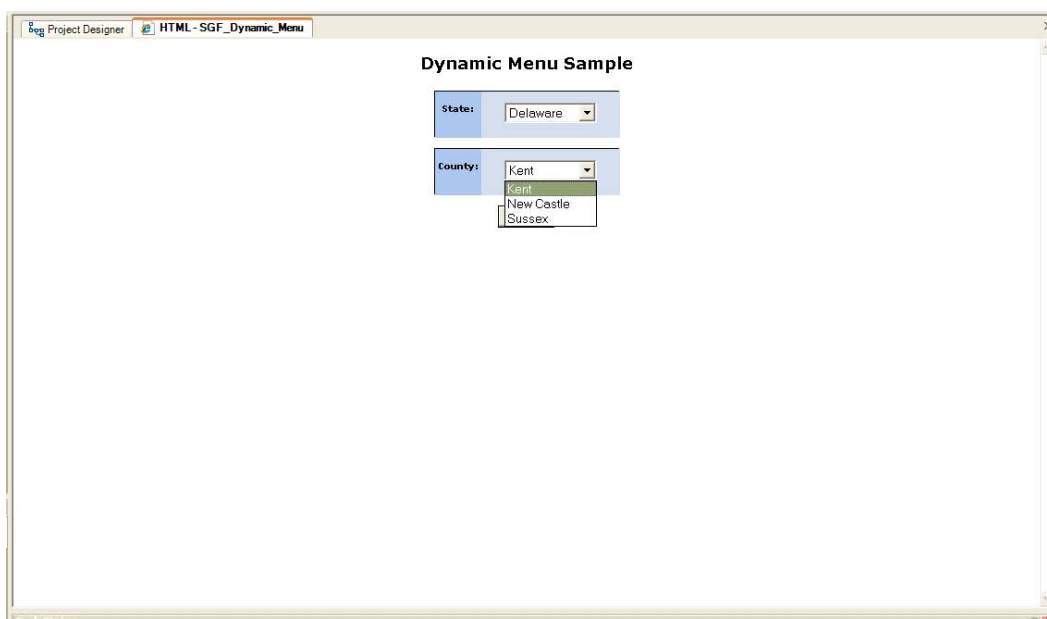
If the macro variable resolves to anything other than 1, the %create\_page macro executes and a new HTML Web form generates.

The %create\_page macro builds an HTML Web form, which upon submission recalls the stored process. Depending on the method used to refresh the page, different results are generated. There are two different ways to submit the macro from this Web form:

- Change the value of the **State** box (Display 3), as shown in our example. Doing so recalls the stored process and sets the refresh\_submit macro variable to 0. On the next execution the %create\_page macro executes again with a new value for &state.
- Click **Submit**. Doing so recalls the stored process and sets the refresh\_submit macro to 1. The %results macro executes and results are generated.

The %output macro generates the results of this stored process. The first set of stored process macros is wrapped around a DATA step which is used to return text back to the browser, notifying the user that the results will be sent to him or her momentarily by e-mail. The remaining output is written to the WORK directory, compressed, and then sent by e-mail to the specified e-mail address. The compressed attachment includes an SPK file extension. This attachment can be treated much like a ZIP file, and once uncompressed, three PDF files are included.

Display 3 shows the final result. After selecting a State, Delaware in this instance, the stored process automatically refreshes and subsets the values for County. As you can see, this not only forces a correct choice by the user, but it also is much less overwhelming than viewing a list with over 3,000 values.



**Display 3. Final output displayed in SAS® Enterprise Guide®**

The full code that demonstrates how this process works is located in the "Appendix". The comments within the code are helpful to further explain the smaller portions of the code in greater detail.

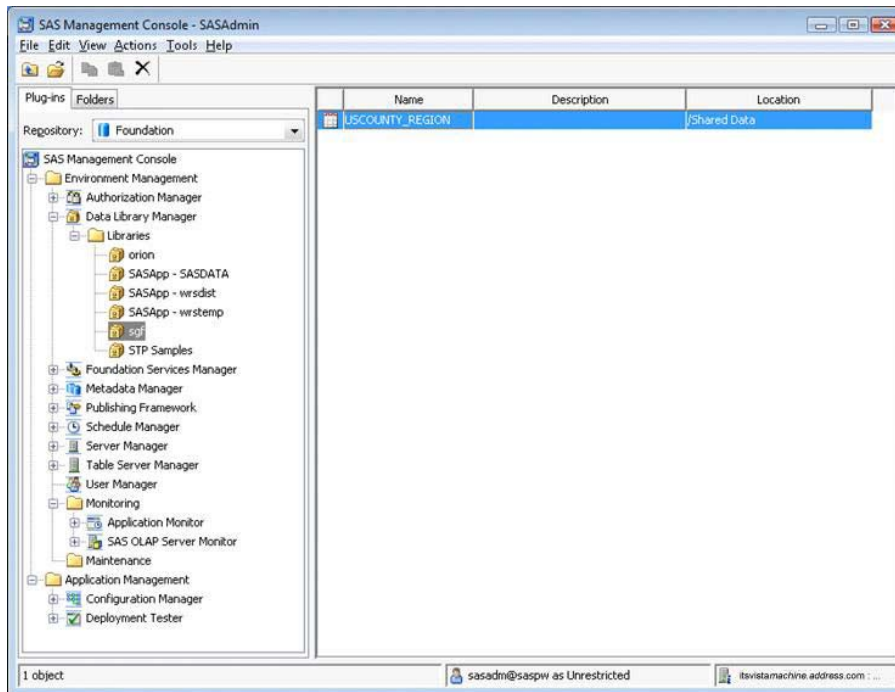
**Note:** Although this stored process is running inside SAS Enterprise Guide, the actual work is being done by the stored process Web application. SAS Enterprise Guide is merely acting as the Web browser.

## CREATING A DYNAMIC INPUT FORM WITH THE SAS® ENTERPRISE GUIDE® 4.2 STORED PROCESS WIZARD

The new and improved SAS Enterprise Guide 4.2 Stored Process Wizard includes built-in functionality for both dynamically-populated menus as well as sub-setting or cascading menu prompts.

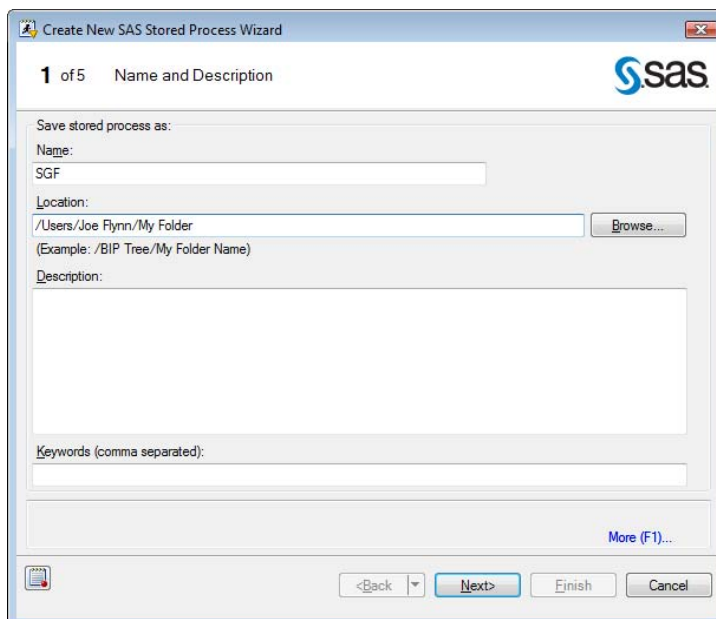
The following steps show how to create cascading menu prompts and dynamically-populated menus in SAS Enterprise Guide 4.2. You can now do this completely inside the Stored Process Wizard, and it requires no extra coding.

1. Before creating the stored process, ensure that all data sets that are used in dynamic prompting are registered metadata. You can register metadata using SAS® Management Console. (You can also use the new "Update Metadata Library" tool that is included in SAS Enterprise Guide 4.2. A discussion of this tool, however, is beyond the scope of this paper.) This example uses the SAS Management Console to create the library called SGF and the registered table, USCOUNTY\_REGION (Display 4).



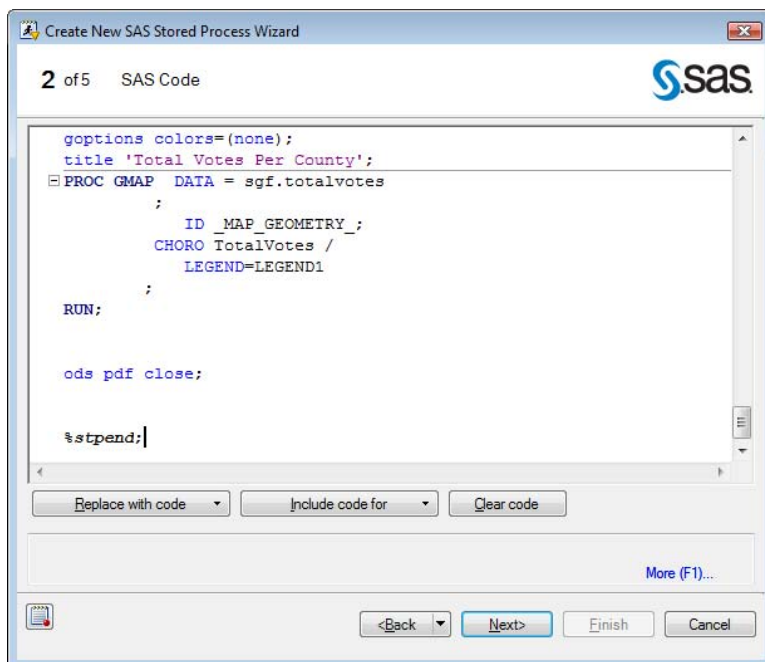
**Display 4. Registered table in SAS® Management Console**

2. Open SAS Enterprise Guide and select **File ► New ► Stored Process**. This opens the new SAS 9.2 Stored Process Wizard. Notice that the wizard now has only five steps. Certain steps have been combined to create a more logical flow in the wizard. However, that the wizard can have six steps if the SAS code uses a library.
3. On page 1 of the wizard, decide the name and metadata location of the stored process. You can also provide a description and any necessary keywords (Display 5).



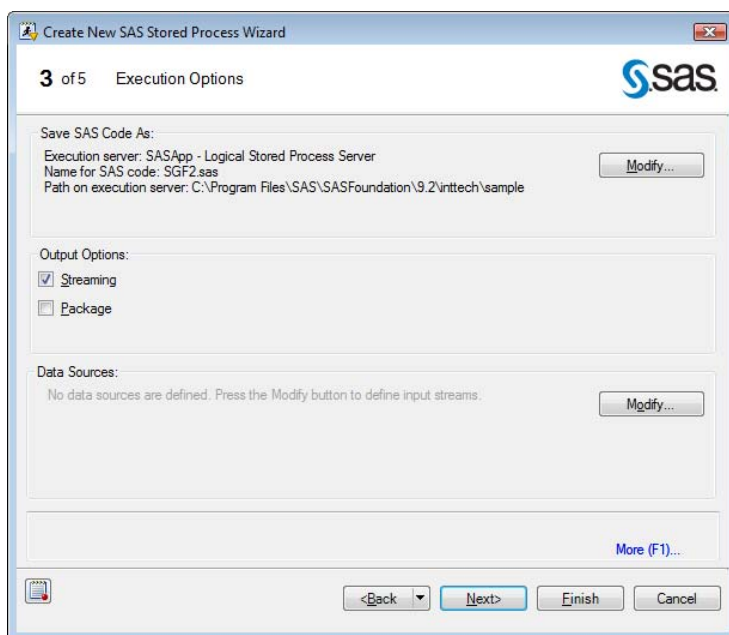
**Display 5. The Name and Description page of the SAS Stored Process Wizard**

4. On page 2, enter the stored process code (Display 6). The code in Display 6 is contained within the output macro from the example code in the “Appendix”.



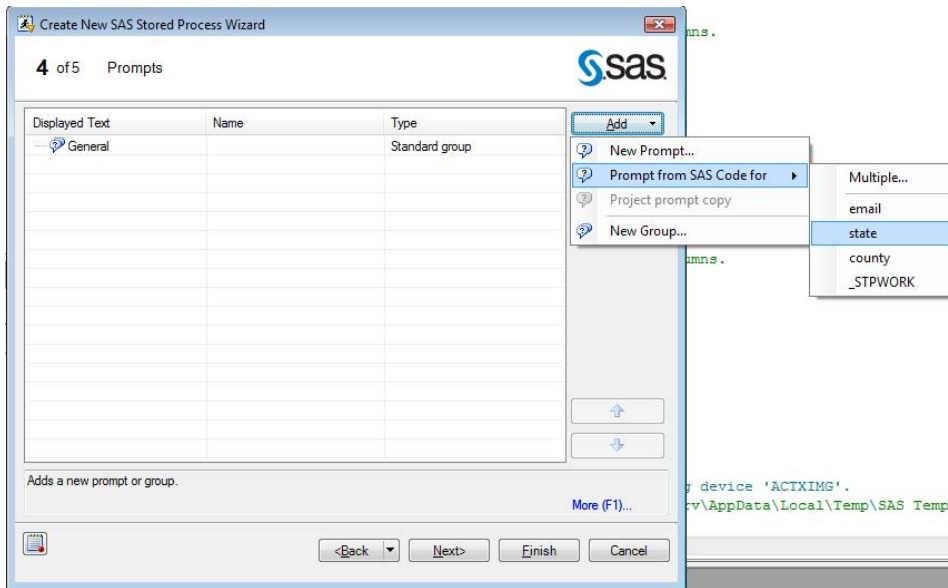
**Display 6. The SAS Code page of the SAS Stored Process Wizard**

5. The next step in the wizard prompts for execution options (Display 7). Output options changed quite a bit from the SAS Enterprise Guide 4.1 Stored Process Wizard to the SAS Enterprise Guide 4.2 Stored Process Wizard. With the 4.1 version of the Stored Process Wizard, you have to declare the type of result that your stored process would generate. With the 4.2 version of the Stored Process Wizard, you are now able to specify multiple forms of results (stream and package) which your stored process can generate. Generally, if you are using ODS (%STPBEGIN and END), then check both **Streaming** and **Package Output Options**. This enables the stored process to run using a large variety of clients. If you are using the \_webout file reference, then only check the **Streaming Output Option**.



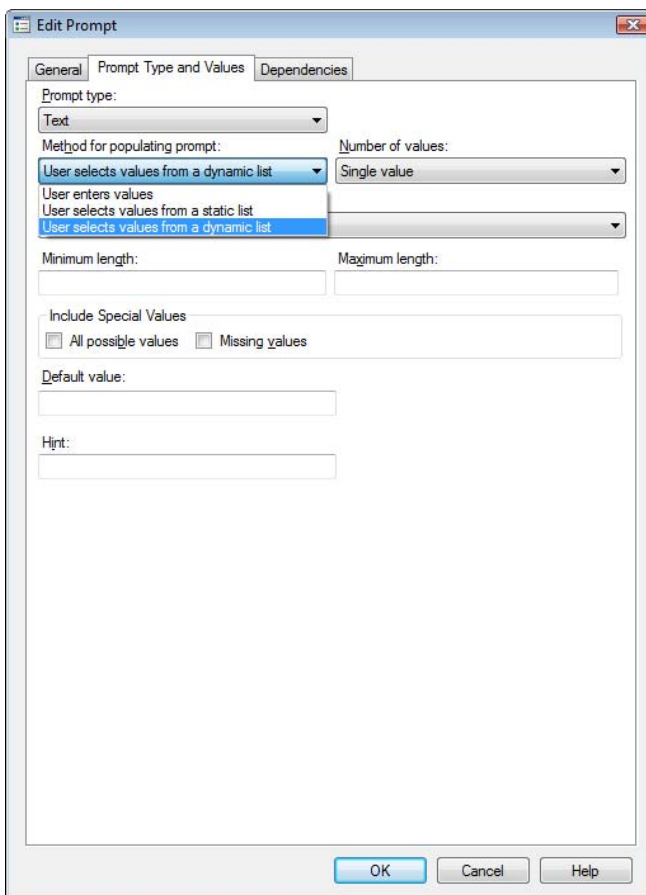
**Display 7. The Execution Options page of the SAS Stored Process Wizard**

6. On the Prompts page, click **Add ► Prompt from SAS Code** and select the desired variable from the menu, which in this example is **state** (Display 8).



**Display 8. The Prompts page of the SAS Stored Process Wizard**

7. On the Edit Prompt page, click the **Prompt Type and Values** tab. Under the **Method for populating prompt** menu, select **User selects values from a dynamic list** (Display 9). Using this method relates the values that are displayed for this prompt directly to a data set. This step eliminates the previous need to manually update parameters as the data changes.



**Display 9. The Edit Prompt page of the SAS Stored Process Wizard**

8. After selecting **User selects values from a dynamic list** in step 7, the contents of the menu change. A **Data source** box appears. Click **Browse** (Display 10) to locate and open the desired table in the metadata. In this example, locate and open the USCOUNTY\_REGION table.

The screenshot shows the 'Edit Prompt' dialog box with the 'General' tab selected. The 'Prompt type' is 'Text'. The 'Method for populating prompt' is 'User selects values from a dynamic list'. The 'Number of values' is 'Single value'. The 'Data source' field is empty, and the 'Browse...' button is highlighted with a red box. Other fields include 'Unformatted Values' (Column:), 'Formatted (Displayed) Values' (Column: and Format:), and 'Sort order' (Default sort order and Default value:).

**Display 10. Clicking Browse on the Edit Prompt page to open a desired table**

9. After opening the desired table, in this example, the USCOUNTY\_REGION table, in both the **Unformatted Values** and **Formatted (Displayed) Values** boxes, select the value for sub-setting the date. In this example, select **State** as the **Column:** value (Display 11).

The screenshot shows the 'Edit Prompt' dialog box with the 'Data source' field set to 'USCOUNTY\_REGION'. The 'Unformatted Values' 'Column:' dropdown is set to 'State' and is highlighted with a red box. The 'Formatted (Displayed) Values' 'Column:' dropdown is also set to 'State' and is highlighted with a red box. The 'Format' field is set to 'Default format'. Other fields include 'Sort order' (Default sort order and Default value:).

**Display 11. Selecting Column values on the Edit Prompt page**

10. Click **OK** and you will return to the Editing Prompt page. Verify that the parameter you selected has been added as the **Column** value. Complete step 9 for any other values, in this example, the **County** variable.

11. Click the **Dependencies** tab (Display 12).

The screenshot shows the 'Edit Prompt' dialog box with the 'Dependencies' tab selected. The 'Data source' is set to 'USCOUNTY\_REGION'. Under 'Unformatted Values', the 'Column' is 'County'. Under 'Formatted (Displayed) Values', the 'Column' is 'County' and the 'Format' is 'Default format'. There are checkboxes for 'Append formatted values with unformatted values', 'Include Special Values', and 'All possible values'. There are also dropdowns for 'Sort order' and 'Default value', and a 'Select...' button. At the bottom, there are 'OK', 'Cancel', and 'Help' buttons.

**Display 12. Selecting the Dependencies tab on the Editing Prompt page**

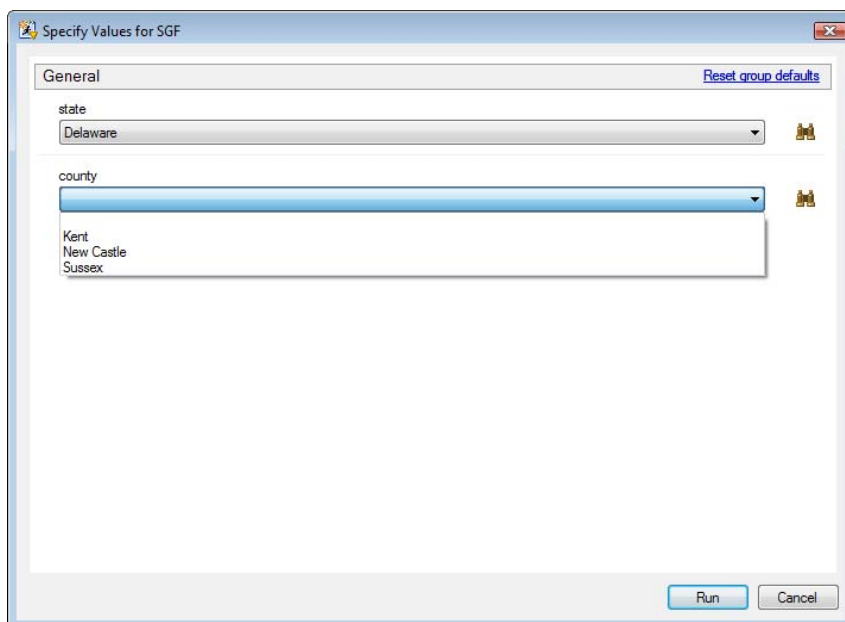
12. On the **Dependencies** tab, click **Add**. A dependency appears. By default this dependency matches the value of a previously added parameter to the value of the equivalent column in the relevant metadata table. In this example, the default dependency **State** matches to the value of the **State Column** in the USCOUNTY\_REGION table (Display 13). This process acts much like a WHERE statement, filtering the results that are returned for your **County** variable. Click **OK**, and verify that the desired parameter in the example, the **State** parameter, has been added.

The screenshot shows the 'Edit Prompt' dialog box with the 'Dependencies' tab selected. A table lists dependencies for the 'state' prompt. The table has two columns: 'Prompt' and 'Description of Dependency'. The first row shows 'state' as the prompt and 'USCOUNTY\_REGION.State Contains Value of 'state'' as the description. There are 'Add' and 'Delete' buttons next to the table. Below the table, there are dropdowns for 'Column' (State), 'Operator' (Contains), and 'Value of' (state). At the bottom, there are 'OK', 'Cancel', and 'Help' buttons. The 'Add' button is highlighted with a red box.

**Display 13. Description of Dependency for the State value on the Editing Prompt page**



13. On page 5 of the wizard, click **Finish** and run the stored process.
14. On the Specify Values page, the wizard prompts you for input for the values that you selected in the previous steps. In the example, select **Delaware** as the **state** value (Display 14). When selecting a value for **county**, you should be presented with only three choices. This functionality is an improvement over the functionality with the SAS Enterprise Guide 4.1 Stored Process Wizard because this prompting is coming from the containing application, rather than from a back-end SAS code or servlet action.



The image shows a dialog box titled "Specify Values for SGF". It has a "General" tab and a "Reset group defaults" link. The "state" dropdown menu is set to "Delaware". The "county" dropdown menu is open, showing a list of three counties: "Kent", "New Castle", and "Sussex". At the bottom of the dialog, there are "Run" and "Cancel" buttons.

**Display 14. Specifying values for the input data**

## CONCLUSION

The enhancements included in SAS Enterprise Guide 4.2 Stored Process Wizard (which is available with SAS 9.2) enable you to create truly dynamic reports with minimal effort. This is done through the use of cascading menu prompts and dynamically-populated menus. Although these functionalities are now built into the software, manually creating dynamic Web forms can still be useful and gives full control over both the appearance and functionality of the user-input form. While it might require extra work, the input form can be customized to any situation. Nonetheless, both of these methods are excellent ways in which stored process developers can build user-friendly reports for decision makers.

## APPENDIX

```

%macro create_page;

    /* Dynamically obtain a list of distinct counties and states from the */
    /* sgf.uscounty_region table. */
proc sql noprint;
select distinct(county) into :county_list separated by "-"
from sgf.uscounty_region
where state="&State";

select distinct(state) into :state_list separated by "-"
from sgf.uscounty_region;
quit;

data _null_;
    /* Specify the _WEBOUT file ref, which will stream PUT statements back */
    /* to the browser. */
file _webout;
put '<html>';

    /* Set up a simple CSS. */
put '<style type="text/css">';
put 'select {height: 30px; width: 100px}';
put 'body {';
put 'font:normal 10px verdana;';
put '}'';

put '.box {';
put 'width:200px;';
put 'position:relative;';
put 'border:1px solid #000;';
put 'height:50px;';
put '}'';

put '.left {';
put 'position:absolute;';
put 'left:0px;';
put 'background-color:#AEC5EF;';
put 'width:50px;';
put 'height:50px;';
put '}'';

put '.right {';
put 'position:absolute;';
put 'left:50px;';
put 'background-color:#D4DDEF;';
put 'width:150px;';
put 'height:50px;';
put '}'';
put '</style>';

    /* Create a javascript function to differentiate between the submit button */
    /* being clicked and the State being changed. Both force a refresh on the */
    /* page, therefore the value of refresh_submit is used during next execution */
    /* of this stored process to determine which method was used to refresh the */
    /* page. This will determine whether to generate another Web page, or to */
    /* display the results. */
put '<script type="text/javascript" language="JavaScript">';
put 'function submitFunction(num) {';
put 'if (num==1) { document.my_form.refresh_submit.value=1; }';
put 'document.my_form.submit()';
put '}'';
put '</script>';

put '<h3> <center> Dynamic Menu Sample </center> </h3>';
put '<body>';
put '<center>';

```

```

/* Create an HTML Form; Action= must correspond to the address of the */
/* SAS stored process Web application. */
put '<form name="my_form"';
put '    method="post"';
put '    action="http://your.webserver:8080/SASStoredProcess/do">';

/* Required hidden value, this needs to correspond to the BIP tree location */
/* of the stored process which will be called; in this case the same STP is */
/* being called. */
put '<input type="hidden"';
put '    name="_program"';
put '    value="/Samples/Stored Processes/SGF_Dynamic_Menu">';

/* Set the default value of the Refresh_Submit variable to 0. */
put '<input type="hidden"';
put '    name="refresh_submit"';
put '    value="0">';

/* Initialize i to 1 for use in a do loop below. */
%let i = 1;

/* Sets up the table defined in the CSS. */
put '<div class="box">';
put '<div class="left">';
put '<br><b>State: </label>';
put '</div>';
put '<div class="right">';

/* Set up an HTML drop-down box. When the value is changed, the submitFunction */
/* is called and passes in a value of 0. */
put '<br><select name="state" onChange="submitFunction(0);">'

/* If a state was previously selected, display it as the first option in */
/* the list. */
%if %symexist(state) %then %do;
put "<option value='&state'>&state</option>";
%end;
/* Iterate through the values contained in the &state_list and create a value */
/* for each.*/
%do %while(%qscan(&state_list,&i,-) ne);
    %let n&i = %scan(&state_list,&i,-);
    put %unquote(%str('%<option value="&n&i">&n&i</option>%'));
    %let i = %eval(&i+1);
%end;
put '</select>';
put '</div>';
put '</div> <br />';

put '<div class="box">';
put '<div class="left">';
/* Very similar to the previous drop-down box, except nothing is executed when */
/* the value changes. */
put '<br><b>County:</b>';
put '</div>';
put '<div class="right">';
%let i = 1;
put '<br><select name="county">';
%if %symexist(county_list) %then %do;
%do %while(%qscan(&county_list,&i,-) ne);
    %let n&i = %scan(&county_list,&i,-);
    put %unquote(%str('%<option value="&n&i">&n&i</option>%'));
    %let i = %eval(&i+1);
%end;
%end;
put '</select>';
put '</div>';
put '</div>';

```

```

    /* When clicking the Submit button, execute the submitFunction, passing in */
    /* a value of 1. This will in turn set the value of refresh_submit to 1, */
    /* and recall the stored process. The next execution of the stored process */
    /* will then print out the results. */
put '<br><INPUT type=submit value=Submit onClick="submitFunction(1);">';
put '</form>';
put '</body>';
put '</html>';
%mend;

%macro output;
    /* Return a message to the browser. */
%stpbegin;
data _null_;
file print;
put "Results Will be Emailed Momentarily..";
run;
%stpend;

%GLOBAL _result;
%GLOBAL _email_address;
%GLOBAL _FROM;
%GLOBAL _SUBJECT;
%GLOBAL _odsstyle;
%GLOBAL _archive_path;
%GLOBAL _odsdest;

    /* Set up reserved macro variables for Package_To_Email. */

*ProcessBody;
%let _result=PACKAGE_TO_EMAIL;
%let _email_address=joe.flynn@sas.com;
%let _FROM=joe.flynn@sas.com;
%let _SUBJECT=Package To Email Test;
%let _odsstyle=sasweb;
%let _archive_path=tempfile;
%let _odsdest=none;

%stpbegin;
proc sql;
create table work.sub_totalvotes as
select * from sgf.totalvotes
where state="&state";

create table work.sub_statewinner as
select * from sgf.statewinner
where state="&state";

create table work.sub_uscounties as
select * from sgf.uscounties
where state="&state" and county="&county";

quit;

    /* Create the County PDF file in the WORK directory. */
    /* This will be compressed and attached to the e-mail. */
ods pdf file="&_STPWORK/County.pdf";
goptions colors=(CX990033 CX0066FF CXB3CC99);
Title "Total Vote Distribution";
title2 "For &County County " ;
PROC GCHART DATA =work.sub_uscounties
;
    PIE      candidate /
    SUMVAR=Total
    TYPE=SUM
    NOLEGEND
    SLICE=OUTSIDE
    PERCENT=INSIDE
    VALUE=OUTSIDE
    OTHER=4
    OTHERLABEL="Other"

```

```

        COUTLINE=BLACK
        NOHEADING
        ;
        RUN;
goptions colors=(none);
Title "Absentee Vote Count";
Title2 "For &County County";
PROC GCHART DATA=work.sub_uscounties
;
    VBAR
    candidate
    /
    SUMVAR=Absentee
    CLIPREF
FRAME    TYPE=SUM
        COUTLINE=BLACK
        RAXIS=AXIS1
        MAXIS=AXIS2
;
run;
quit;

Title " ";
Title2 " ";
proc print data=work.sub_uscounties noobs;
var State County Candidate Day_Of_Election Absentee Provisional Total;
sum Total;
run;

ods pdf close;

    /* Create the State PDF file in the WORK directory. */
    /* This will be compressed and attached to the e-mail. */
ods pdf file="&_STPWORK/State.pdf";
goptions colors=(CX990033 CX0066FF CXB3CC99);
title 'Winner by County';
title2 "For &State";
PROC GMAP DATA = work.sub_statewinner
;
    ID _MAP_GEOMETRY_;
    CHORO Candidate /
    LEGEND=LEGEND1
;
RUN;

goptions colors=(none);
title 'Total Votes Per County';
title2 "For &State";
PROC GMAP DATA = work.sub_totalvotes
;
    ID _MAP_GEOMETRY_;
    CHORO TotalVotes /
    LEGEND=LEGEND1
;
RUN;
ods pdf close;

    /* Create the National PDF file in the WORK directory. */
    /* This will be compressed and attached to the e-mail. */
ods pdf file="&_STPWORK/National.pdf";
goptions colors=(CX990033 CX0066FF CXB3CC99);
title 'Winner by County';
PROC GMAP DATA = sgf.statewinner
;
    ID _MAP_GEOMETRY_;
    CHORO Candidate /
    LEGEND=LEGEND1
;
RUN;

```

```

goptions colors=(none);
title 'Total Votes Per County';
PROC GMAP DATA = sgf.totalvotes
;
    ID _MAP_GEOMETRY_;
    CHORO TotalVotes /
    LEGEND=LEGEND1
;
RUN;

ods pdf close;
%mend;

%macro main;
    /* Check if the refresh_submit macro variable exists. */
    /* This value is set by the create_page macro. */
    /* If the value resolves to 1, the Submit button from the Web form was */
    /* clicked. */
    /* If the value resolves to 0, the value of state was changed. */
%if %symexist(refresh_submit) %then %do;
    %if &refresh_submit ne 1 %then %do;
        %create_page;
    %end;
    %else %do;
        %output;
    %end;
%end;
%else %do;
%create_page;
%end;
%mend;

%main;

```

## RESOURCES

You can download the SAS data sets that were used in the examples in this paper at the following location:

Flynn, J. 2009. "SAS® Stored Processes: Going Beyond the Current Capabilities of the Stored Process Wizard". Technical Papers and Presentations Web site. Available at [support.sas.com/rnd/papers/index.html](http://support.sas.com/rnd/papers/index.html).

## CONTACT INFORMATION

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

Joe Flynn  
SAS Institute Inc.  
SAS Campus Drive  
Cary, NC 27613  
Work Phone: 919-677-8000  
Fax: 919-531-9449  
E-mail: [support@sas.com](mailto:support@sas.com)  
Web: [support.sas.com/ts](http://support.sas.com/ts)

SAS and all other SAS Institute Inc. product or service names are registered trademarks or trademarks of SAS Institute Inc. in the USA and other countries. © indicates USA registration. Other brand and product names are trademarks of their respective companies. Copyright © 2009 SAS Institute Inc., Cary, NC, USA. All rights reserved.