Chapter

2

Creating Simple Stored Processes

BASE SAS gives programmers the exponential ability to query and report about data from their desktops; however, this limitation means that a user can access the data from their desktops only. As an organization's reporting needs grow, more individuals need to quickly retrieve and analyze similar information. As a result, a small group with access to the data unintentionally becomes *report gatekeepers*. Other members of the organization have to talk to these gatekeepers for even the simplest piece of data.

Before you convert a SAS program to a stored process, you must consider whether the program is a good candidate for conversion. Although any SAS program can be a stored process, this does not mean that all programs should be a stored process. Programs that require user input, run on user demand, and generate output are typically better candidates than those programs scheduled to run overnight, take a long time to run or require no user modifications.

In Chapter 1, "Getting Started with Stored Processes" you learned how to register a stored process. In this chapter, you will convert an existing SAS program into a stored process and apply different prompt types to increase the scope and flexibility of a program.

2.1 Sample Scenario for Stored Process Conversion

As an example of a report gatekeeper and how a stored process solved the issue, let's consider the following business scenario. There is a SAS program that exports returned hardware parts from an enterprise database to a spreadsheet. Each time a reliability analyst reviews a part with a high return rate, a SAS programmer adds the part number to the where statement, reruns the program, and sends the spreadsheet to the analyst. The analyst retrieves the data and begins the analysis process.

After the program was converted to a stored process, the reliability analyst could access the stored process using the web interface or even the SAS Add-In for Microsoft Office application. Within moments of selecting the desired product from a drop-down list, the analyst is able to start reviewing the results; thus, adding automation to the business process. In addition, the stored process allows the analyst to access the report faster, and the SAS programmer can continue to work on other tasks; thus, increasing the business efficiency.

#5 **SAS programs that require user input, run on demand, or require up to date output immediately are all great candidates for converting to stored processes**

The program in this scenario was an excellent candidate for converting into a stored process. By using a single prompt, the stored process essentially asked the reliability analyst which product to retrieve.

Because the stored process prompts the user, other reliability analysts can also use this same report, except they would have data specific to their specific needs.

2.2 Creating a Stored Process with a Single Prompt

In the first example, you duplicate the previous business scenario by converting an existing program that uses a single prompt into a stored process for your new customer: a large North American furniture company called Friendly Furniture Makers. The company has a team of sales analysts in their marketing department that need to analyze sales trends and forecast future sales so the manufacturing operation knows what to produce.

The company needs to convert a batch program that compares predicted and actual sales for a product to a Web report that the sales analyst team can run on-demand. Currently, the report is generated from an overnight batch process using a program that contains a SAS macro called MakeReport. MakeReport runs four times to create a report for each product line: Desk, Chair, Bed, and Sofa.

#6 **SAS Macro code is easy to convert to a stored process because all of your variables are already present in the code.**

Your assignment is to convert the existing batch program into a stored process that allows the sales analysts to run the stored process from the Web browser. The sales analyst can select the product line from a drop-down list, select Run, and generate the report on demand, as shown in the following figure. This prompt replaces the parameter that is used in the macro code.

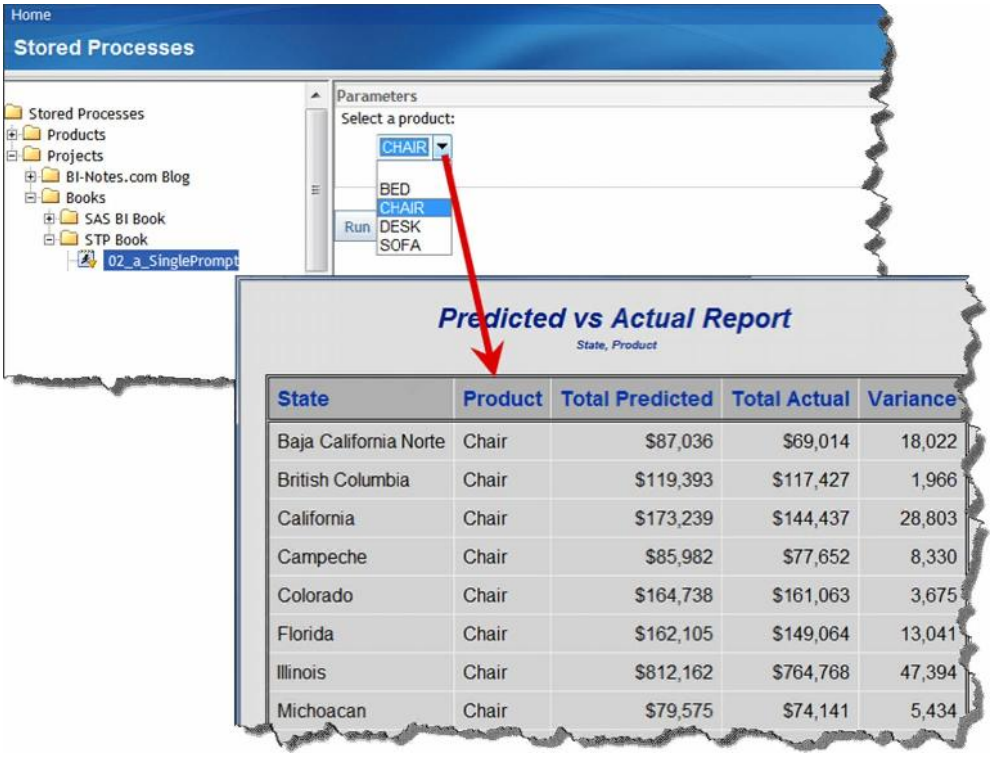

Figure 2-1 Using a single selection prompt

To convert this report, you must register a new stored process and add a single selection prompt for the product to the code. See Section 1.2, "Registering a Stored Process" for specific steps to register the stored process.

2.2.1 Creating a Single Prompt

As you register the stored process, create a text prompt with the name **prodprompt,** which allows a user to select a single item. This prompt contains a static list of the four product lines in our example. Figure 2-2 shows how to set up the prompt and how the resulting prompt appears in the Parameters tab.

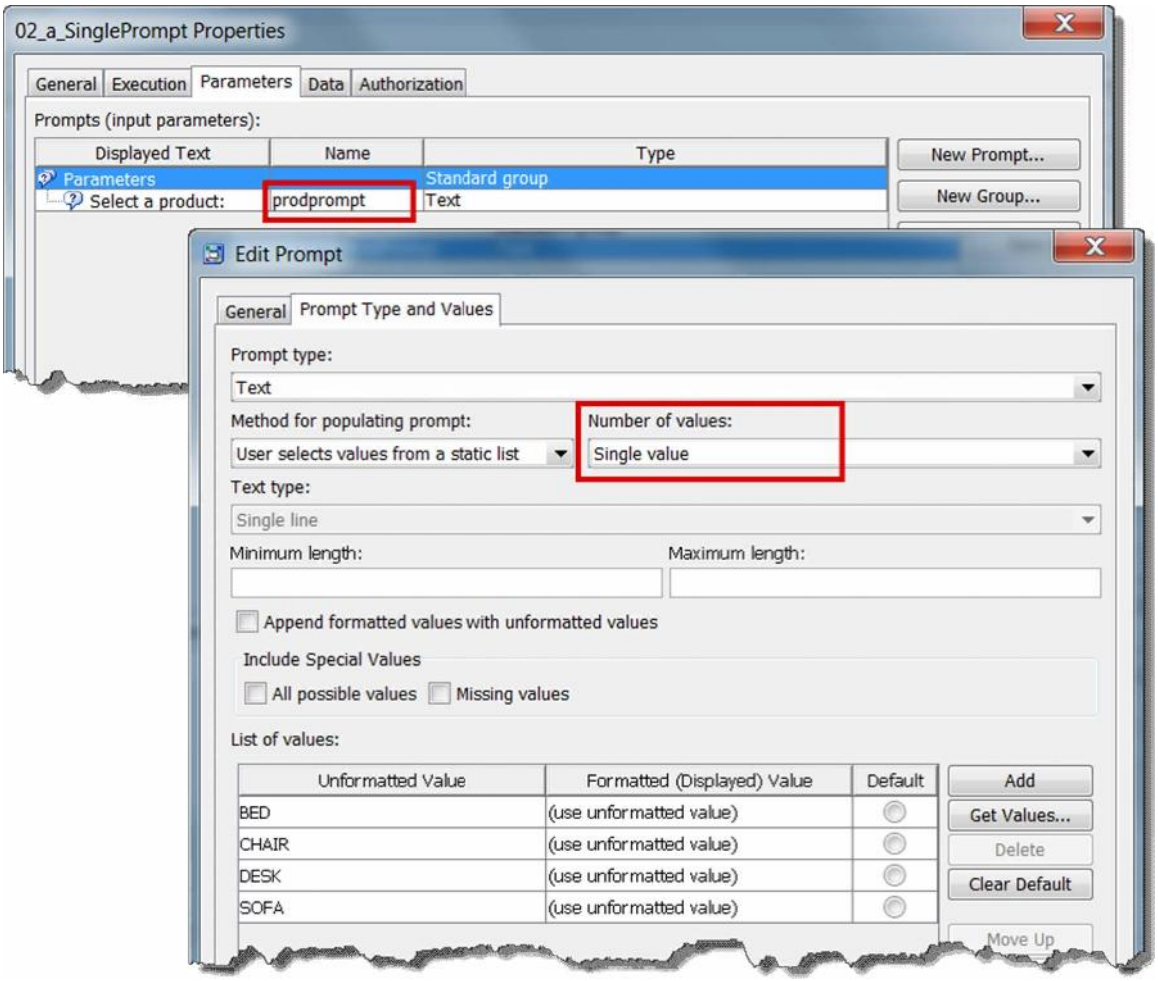

Figure 2-2 Creating a product prompt

2.2.2 Updating the Code

To modify the existing code, you must replace the current macro with the %STPBEGIN/%STPEND macros. The following code shows how to convert the existing legacy code to a stored process.

#7 **As your stored process code becomes more complex, ensure that your metadata descriptions and code comments explain what the code is doing and why.**

Example 2-A Single Prompt	Required Modifications
~~%macro MakeReport(prodprompt);~~	Remove the macro statement. The code can run outside of the macro. The newly created prompt replaces the need for the macro parameter.
`libname booksamp meta library="STP Book Sample Data";`	Referencing the library via the META engine ensures that user credentials are used in authorizing access.

`proc sql;`
`create table work.qresult as`
`select distinct`
`  state label="State"`
`  ,propcase(product) as Product`
`  label="Product"`
`  ,sum(predict) as predictsum`
`  label="Total Predicted"`
`  format=dollar12.`
`  ,sum(actual) as actualsum`
`  label="Total Actual"`
`  format=dollar12.`
`  ,sum(predict)-sum(actual) as diff`
`  label="Variance"`
`  format=negparen10.`
`from booksamp.prdsal2011`
	Stored processes use the same BASE SAS code as other processes. No changes are required in the PROC SQL, which generates a temporary dataset that summarizes three variables by state and product.

`where upcase(product)` `  = "&prodprompt"`	Because you are converting this macro to a stored process, no code changes are required. As shown in Figure 2-1, the prompt name is prodprompt and is used here inside the double quotation marks. Use double quotation marks to ensure that the macro variable resolves correctly. If you use single quotes, SAS does not convert the &prodprompt macro variable, and the stored process will not return any results.

`group by state, product`
`order by state;`
`quit;`	

~~ods html file="//path/rpt_&prodprompt.xls";~~ `%stpbegin;`	The stored process returns the output to the Web browser so that you can remove the ODS HTML statements. The %STPBEGIN macro manages the output.

Example 2-A Single Prompt	Required Modifications

`title1 "Predicted vs Actual Report";`
`title2 height=1 "State, Product";`
`proc print data=work.qresult label noobs;`
`run;`
	PROC PRINT creates the final output from the temporary dataset.

~~`ods html close;`~~ ~~`%mend;`~~ **`%stpend;`**	Replace the macro end variable with the stored process end variable.
~~`%MakeReport(BED);`~~ ~~`%MakeReport(CHAIR);`~~ ~~`%MakeReport(DESK);`~~ ~~`%MakeReport(SOFA);`~~	Delete the macro statements because the user executes the code when calling the stored process using the prompt to select the desired report.

#8 **After creating a stored process, test it by selecting several of the prompt choices to ensure that the user will get expected results.**