Paper 038-2007

Empowering Your SAS® Business Intelligence End Users via a SAS® Stored Process

David Pope, SAS Institute Inc., Cary, NC

ABSTRACT

The SAS Stored Process presented in this paper is a powerful example of how simple it is to empower your SAS Business Intelligence end users with the power of SAS. The stored process enables an end user to create a SAS data set using SAS® Enterprise Guide, SAS® Add-In for Microsoft Office (MS Excel), or SAS® Web Report Studio; and then add rows, delete rows, or modify rows in that data set.

With just a little imagination, you can easily use the example given in this presentation to show how entire applications can be written using SAS Stored Processes and surfaced via SAS Business Intelligence interfaces.

INTRODUCTION

Stored processes can be used to encapsulate any SAS code no matter how simple or complex it is, so that the code can be easily re-used and shared with others. I like to think of stored processes as glorified SAS macros. The big difference between macros and stored processes is that stored processes can be easily embedded in third-party applications or interfaces but macros cannot. Stored processes are a very powerful way to spread the power of SAS to many people within an organization who don't need to learn how to program in SAS, but they can benefit from the extra intelligence that SAS analytics automatically adds to data analysis and reporting. Stored processes help you leverage the investment that you've made in SAS by spreading the Power to Know® easily throughout your organization.

The stored process discussed in this paper enables an end user to dynamically create a new SAS data set by using SAS Enterprise Guide, SAS Add-In for Microsoft Office, or SAS Web Report Studio, and then update the SAS Metadata Server with the metadata for this newly created data set. In addition, this same stored process enables multiple end users to add rows, delete rows, or modify rows in this single data set by using different interfaces. The following sections show you how to create and run this stored process.

CREATE THE ADD_DEL_MOD STORED PROCESS IN SAS ENTERPRISE GUIDE

1. Use SAS Management Console to create the library named EGTASK, as defined in the following code.
2. Use SAS Enterprise Guide to open a new project and add a new code node.
3. Put the following code in the code node and close the node.

   NOTE: To find the values for the LIBID= and REPID= options, which are used in PROC METALIB in the following program, open SAS Management Console, right-click the EGTASK library and select PROPERTIES. The first 8 characters of the ID property are the values for REPID=; the second 8 characters are the values for LIBID=.

```
/* EGTASK is the library in which the new data set will reside.               */
/* NOTE: For this example, the EGTASK library should also have already been   */
/*       assigned in the SAS Metadata Server by using SAS Management Console. */

libname egtask "C:\SASGF";

/* All macro references on the right-side of the equal sign (=) are names     */
/* of global macro variables that need to be defined as parameters when you   */
/* create the stored process. Those parameters are: ACTION, KEY, COMMODITY,   */
/* WHOLESALE, RETAIL, and DATE.                                               */

%macro wrs_input(maction=&action,knum=&key,com=&commodity,wprice=&wholesale,
                 rprice=&retail,mdate=&date);
```

```
/* The function INITIAL initializes the data set EGTASK.COMMODITY with        */
/* no rows and 5 variables: KEY_NUM, COMMODITY, WHOLESALE_PRICE, RETAIL_PRICE,*/
/* and DATE. The metadata server will also be updated to include metadata in  */
/* the new table EGTASK.COMMODITY.                                            */

%IF &MACTION EQ %STR(initial) %THEN %DO;

data egtask.commodity;
   length key_num 8 commodity $20 wholesale_price retail_price date 8;
   format date mmddyy8.;
   stop;
run;

/* Update the SAS Metadata Server to include the new table so that the table  */
/* can be seen in the EGTASK library by SAS Enterprise Guide and SAS Add-In   */
/* for Microsoft Office.                                                      */

proc metalib;
   omr (libid="BF0002BE"   /* This will be unique to your metadata server.    */
        repid="A5JYLOZI"); /* This will be unique to your metadata server.    */
   select("commodity");
run;

%END;

/* Appends a new row of data to the data set EGTASK.COMMODITY.  */

%IF &MACTION EQ %STR(add) %THEN %DO;

/* The data set TEMP has the same variables as the data set EGTASK.COMMODITY. */
/* TEMP contains 1 row. Data is based on the input provided by the end user.  */

data temp;
   length key_num 8 commodity $20 wholesale_price retail_price date 8;
   key_num=_n_;
   commodity="&com";
   wholesale_price=&wprice;
   retail_price=&rprice;
   date="&mdate"d;
run;

/* Appends the data set TEMP to the end of the data set EGTASK.COMMODITY,    */
/* and assigns the primary key variable KEY_NUM by setting it to the value   */
/* of the automatic variable _N_.                                            */

data egtask.commodity;
   set egtask.commodity temp;
   key_num=_n_;
run;

%END;

%IF &MACTION EQ %STR(delete) %THEN %DO;
```

```
/* Use the primary key that was input by the end user to delete the associated */
/* row in the data set EGTASK.COMMODITY.                                       */

data egtask.commodity;
    set egtask.commodity;
    key_num=_n_;
    if key_num ne &knum then output;
run;

/* After the deletion, re-assign the primary key variable KEY_NUM.  */

data egtask.commodity;
   set egtask.commodity;
   key_num=_n_;
run;

%END;

%IF &MACTION EQ %STR(modify) %THEN %DO;

/* Replace the values in the row of data in EGTASK.COMMODITY with the data     */
/* that is provided by the end user based on the primary key entered.          */
/* NOTE: This section could be changed to use PROC SQL update.                 */

data egtask.commodity;
    set egtask.commodity;
    key_num=_n_;
    if key_num = &knum then do;
       commodity="&commodity";
       wholesale_price=&wprice;
       retail_price=&rprice;
       date="&mdate"d;
    end;
run;

%END;

%IF &MACTION EQ %STR(view) %THEN %DO;

/* VIEW requires no code.  */

%END;

/* After any action is chosen by the end user, this PROC PRINT sends the data  */
/* that is stored in EGTASK.COMMODITY to the end user.                          */

proc print data=egtask.commodity; run;

/* Additional code may be called here. For example, a graph or a forecasting   */
/* based on the updated EGTASK.COMMODITY data set OR other stored processes    */
/* can be created to do other analyses or reports based on the data set        */
/* EGTASK.COMMODITY.                                                           */


%MEND; /* wrs_input */

%wrs_input;
```

4. Right-click the code node and select **Create Stored Process**. This opens the Create New SAS Stored Process Wizard.

CREATE THE STORED PROCESS

Work through the Create New Stored Process Wizard to create the stored process.

1. On the General Information page, name the stored process, for example, add_del_mod, and provide a description (a description is optional). Click **Next**.
2. On the SAS Code page, click **Include Code for** and add "LIBNAME references". Close the Messages window and click **Next**.
3. On the Metadata Location page, click **Choose Location** and make this selection:

   ```
   //Foundation/BIP Tree/ReportStudio/Shared/Reports/StoredProcesses
   ```

   Click **OK** and then click **Next**.

4. On the Execution Environment page, click **Modify** and choose an "Execution Server". For this example, I used "SASMain – Logical Stored Process Server". Choose a "Source File path:". You can use the default path that displays, or add a new path. Click **Save** and then click **Next**.
5. On the Parameters page, click **Add** and select "Parameters from SAS Code". Follow these instructions for each scanned parameter. Ignore all other scanned parameters that are not listed below and click **Close**.

```
a. Scanned Parameter action
      i. General Tab
            1. User prompt: Enter_Action
            2. SAS variable name: action
            3. Data type: String
            4. Options: Add Required
     ii. Constraints Tab
            1. Choose: List of values and enter the following
            2. Display as: View - Resolves to: view
            3. Display as: Add - Resolves to: add
            4. Display as: Delete - Resolves to: delete
            5. Display as: Modify - Resolves to: modify
            6. Display as: Initialize - Resolves to: initial
b. Scanned Parameter key
      i. General Tab
            1. User prompt: Enter_Key
            2. SAS variable name: key
            3. Data type: Integer
            4. Default Value: 0.
     ii. Constraints Tab
            1. Not applicable
c. Scanned Parameter commodity
      i. General Tab
            1. User prompt: Enter_Commodity
            2. SAS variable name: commodity
            3. Data type: String
            4. Default Value: none /* ACTUALLY type "none" as default string value. */
            5. Options: Add Required
     ii. Constraints Tab
            1. Not applicable
d. Scanned Parameter wholesale
      i. General Tab
            1. User prompt: Enter_Wholesale_Price
            2. SAS variable name: wholesale
```

```
                3.  Data type: Float
                4.  Default Value: 0
                5.  Options: Add Required
          ii.  Constraints Tab
                1.  Not applicable
    e.  Scanned Parameter retail
           i.  General Tab
                1.  User prompt: Enter_Retail_Price
                2.  SAS variable name: retail
                3.  Data type: Float
                4.  Default Value: 0
                5.  Options: Add Required
          ii.  Constraints Tab
                1.  Not applicable
    f.  Scanned Parameter date
           i.  General Tab
                1.  User prompt: Enter_Date
                2.  SAS variable name: date
                3.  Data type: Date
                4.  Options: Add Required
          ii.  Constraints Tab
                1.  Not applicable
```

6. On the Output Options and Input Streams page, select "Streaming output" and then click **Next**.
7. On the Summary page, read the summary to verify the information and then click **Finish**.

RUNNING THE ADD_DEL_MOD STORED PROCESS

1. The first time you run this stored process, select "Initialize" on the **Enter_action** drop-down menu, and type any character in the **Enter_commodity** field.
2. Click **Run**.

   If you don't receive any error messages, then congratulations! You have successfully created a new data set named EGTASK.COMMODITY and updated the SAS Metadata Server with the information about this newly created table! Usually, errors now are related to Write permissions to the EGTASK library. To resolve these issues, you might need to change permissions for the OS folder that the EGTASK.COMMODITY data set points too, or change permissions in SAS Management Console for the EGTASK library.

3. Now, run the stored process multiple times from SAS Enterprise Guide, Microsoft Excel via the SAS Add-In for Microsoft Office, and SAS Web Report Studio. You should select different types of action, for example, add several rows from various interfaces.

   Notice that the new table is now selectable from the library EGTASK from both SAS Enterprise Guide and the data sources that are available from SAS Add-In for Microsoft Office in Excel. This means that, as the table is changed, ad hoc analysis of the table is available while the table itself is being updated.

Figures 1, 2, and 3 on the following pages show how to run the ADD_DEL_MOD stored process from three different interfaces: SAS Enterprise Guide, Microsoft Excel using SAS Add-In for Microsoft Office, and SAS Web Report Studio.

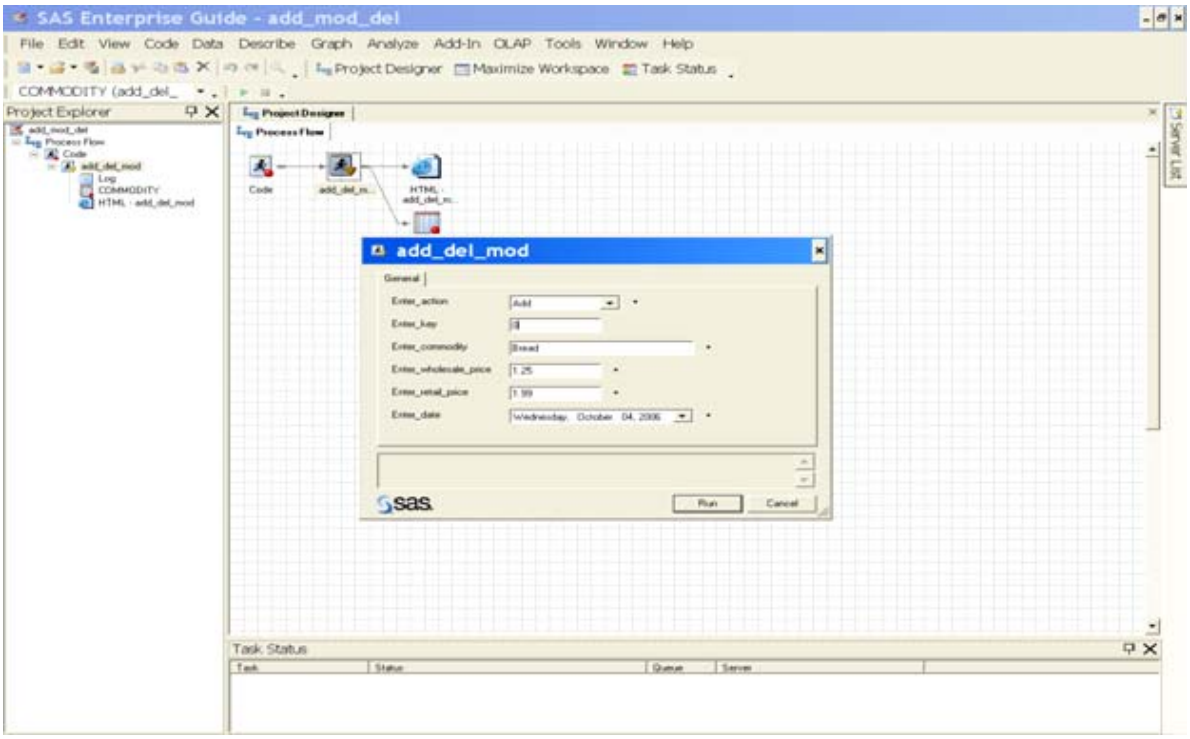

Figure 1. Running the Add_Del_Mod Stored Process from SAS Enterprise Guide

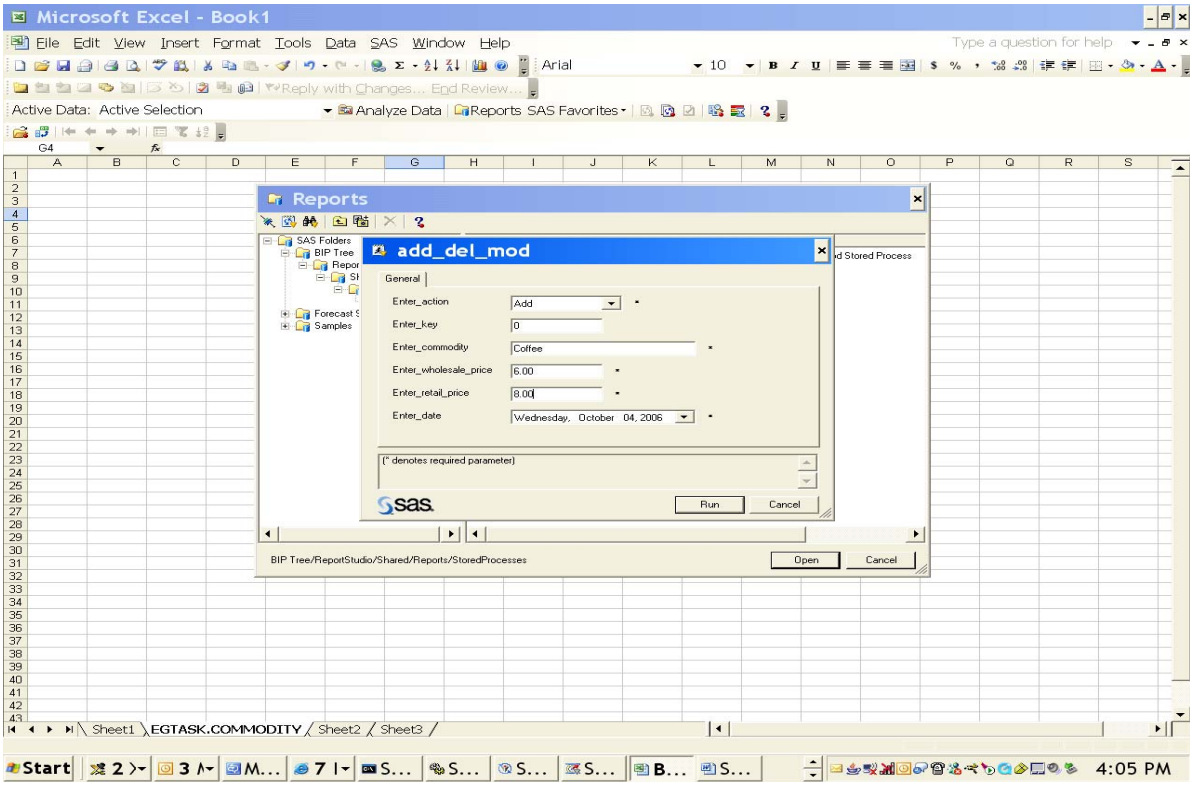

Figure 2. Running the Add_Del_Mod Stored Process from Microsoft Excel Using SAS Add-In for Microsoft Office

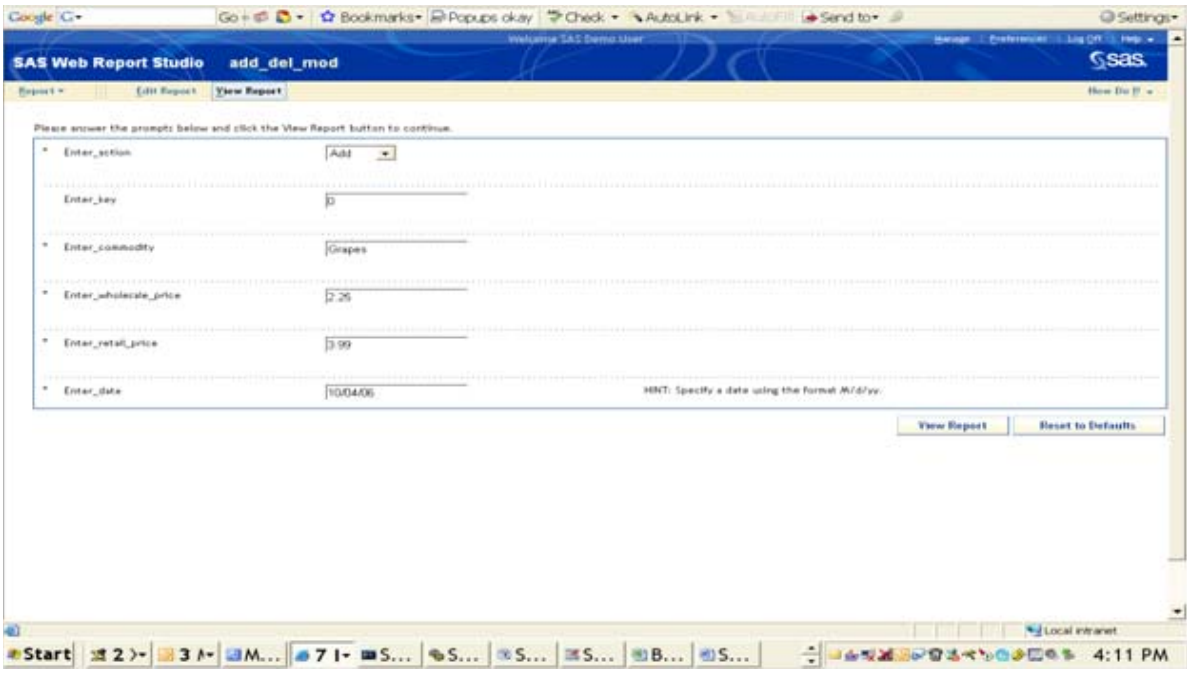

Figure 3. Running the Add_Del_Mod Stored Process from SAS Web Report Studio

CONCLUSION

The SAS Stored Process presented in this paper shows how easy it is to share the power of SAS with your business intelligence end users. Because a stored process can be any SAS program, what you can share with end users is limited only by the restrictions that you, as a programmer, add to your stored processes. Throughout my career as a SAS user, I've found, time-and-time again, that SAS can do anything I want it to do and do it in many different ways. Stored processes continue this tradition and SAS Business Intelligence solutions provide an easy framework for you to share your stored processes with end users.

You can easily take this example and expand it to enable end users to add observations to a data set from Excel and SAS Web Report Studio, and then use that data set to provide a forecast. Another powerful example is to provide Excel and SAS Web Report Studio users with the ability to take a new data source and run it through a champion SAS Enterprise Miner scoring model that returns the results immediately to the end user. Take a moment and think about just how powerful these examples are. With SAS Stored Processes, Excel users, or Web browser users can score data with a SAS Enterprise Miner model, or see forecasts based on newly added data without involving anyone else.

RECOMMENDED READING

To gain a more complete understanding of SAS Stored Processes and SAS Business Intelligence, the author recommends the following publications and Web sites.

SAS Institute Inc. 2006. "SAS® 9.1.3 Integration Technologies: Developer's Guide". Available at support.sas.com/rnd/itech/library/toc_devguide.html.

SAS Institute Inc. 2007. SAS® Integration Technologies. "SAS Integration Technologies: Technical Papers and Presentations". Available at support.sas.com/rnd/itech/papers/index.html.

SAS Institute Inc. 2007. SAS® Integration Technologies. "Welcome to the SAS Integration Technologies Web Site". Available at support.sas.com/rnd/itech/intro.html.

SAS Institute Inc. 2007. Technologies/Business Intelligence. "Business Intelligence". Available at www.sas.com/technologies/bi/index.html.

SAS Institute Inc. 2005. "The SAS® Enterprise Intelligence Platform: SAS® Business Intelligence". Available at www.sas.com/ctx/whitepapers/whitepapers_frame.jsp?code=239.

CONTACT INFORMATION

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

David Pope
SAS Institute Inc.
SAS Campus Drive
Cary, NC 27513
(919) 531-4480
E-mail: David. Pope@sas.com
www.sas.com