Improving SAS® Processes with Excel®

Steven X. Yan, Ph.D.
Marketing, Zale Corp, Irving, TX 75038

ABSTRACT

SAS and Excel are two of the most useful tools in processing, analyzing, and reporting data, with each of them having its own strength and often a distinct user group. For example, SAS is one of the most powerful tools for data merging, cleansing, coding, and statistical analysis; however, it is not an ideal tool for certain kinds of reporting described in this paper. Excel, on the other hand, is the most widely used business tool for data analysis and reporting and has strong graphing and formatting facilities; however, its capability for processing large data sets and statistical analyses is limited. This paper explores ways to integrate the two applications and take advantage of their strength. Specifically, it will show: 1. How to use Excel to create a user-friendly report specification; 2. How to use SAS to read and parse the specification; and 3. How to write SAS results to Excel for enhanced formatting or graphing.

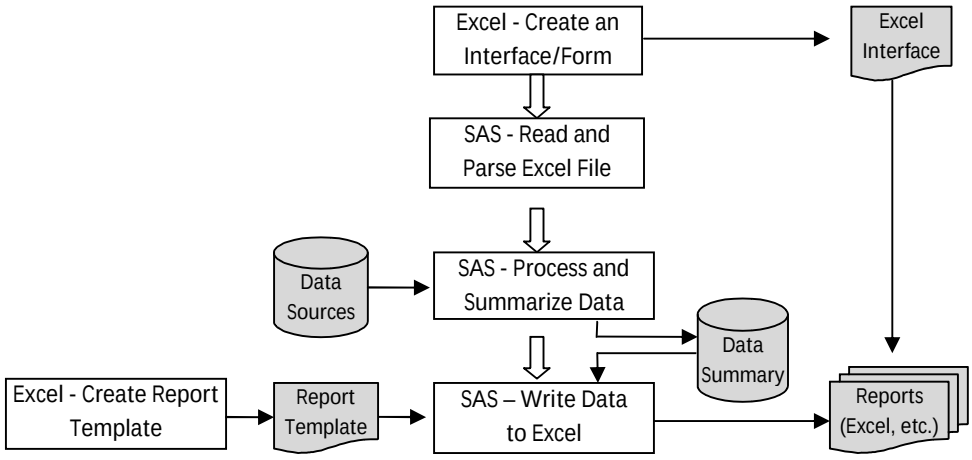
1. INTRODUCTION

Many SAS programmers use the SAS macro utility to extend and customize their applications and reduce repetitive coding. For example, one can assign date parameters to macro variables at the beginning of the program to create sales report for various time periods.

A more advance platform for developing SAS applications is SAS/AF® that can not only pass parameters to the SAS program, but also control the flow of the program execution. Although SAS/AF is a very powerful tool, it requires the installation of this additional component and special training.

In this paper we will create a SAS/AF-like interface using Excel. This approach has several advantages. First, a sophisticated SAS program can be more streamlined and structured with better maintainability. Second, no additional cost or significant training is required to deploy the application. Third, business people with limited or no SAS skills can run the application with ease.

The following is a typical flow of the process.



The development of such applications requires only basic SAS skills in the data step utility, SAS macros, and DDE. Some advanced Excel skills such as Excel macros can greatly enhance the Excel interface but they are not required.

In the following sections we will illustrate this methodology by developing a sales reporting system that compares sales by gender for selected cities, times, or departments. The data used in this example is from a totally simulated department store: SimulMart.

2. DYNAMIC DATA EXCHANGE (DDE) BETWEEN SAS AND EXCEL

First, let's review the DDE technology that is used to exchange data between SAS and Excel. DDE is a mechanism that permits dynamic and continuous data exchanges between two different applications. It has been widely used in SAS applications to read data from and write data to Excel as well as format spreadsheets. The literature on the topic is extensive, both online and offline. See Vyverman (2001, 2002), Watts (2005), Cohen and Shields (2004), Hoffman (2003), and Feng (2005). However, the future of DDE to Excel is not so certain, partly because of its heavy reliance on the Microsoft's X4ML functions (Excel Version 4 Macro Language) that is no longer supported by Microsoft. X4ML functions still work in Excel 2007 and probably future versions as well. Derby (2008) showed that the DDE method was superior to newer methods in some common cases. No matter what will happen, the method presented in this paper can be easily adapted to other platforms.

The MACROFUN.HLP lists complete description of the X4ML functions and is a very valuable reference. It can be downloaded from the Microsoft's website (Search for MACROFUN on the internet and follow the instructions).

There are two types of DDEs, the triplet and the doublet.

Doublet: `filename myExcel DDE 'Excel|System'.`

Triplet: `filename myExcel DDE 'Excel|[Excel file]sheet!Range'`

The following example illustrates and compares the usage of the two methods for writing data to Excel.

```
*DDE doublet;
filename Excel dde "Excel|system";
data _null_;
   file Excel delimiter='09'x;
   put '[Activate("book1")]';
   put '[Window.Maximize("book1")]';
   put '[Workbook.Activate("sheet1")]';
   array x(10) x01 - x10;
   do i=1 to 100;
      do j=1 to dim(x);
         x(j)=100*i+j;
         end;
      put '[Select("R' i +(-1) 'C1")]';
      put x01 - x10 '09'x;
      end;
   run;
```

```
*DDE triplet;
filename Excel dde "Excel|[book1]sheet2!C1:C10";
data _null_;
   file Excel delimiter='09'x;
   ;
   ;
   ;
   array x(10) x01 - x10;
   do i=1 to 100;
      do j=1 to dim(x);
         x(j)=100*i+j;
         end;
      ;
      put x01 - x10;
      end;
   run;
```

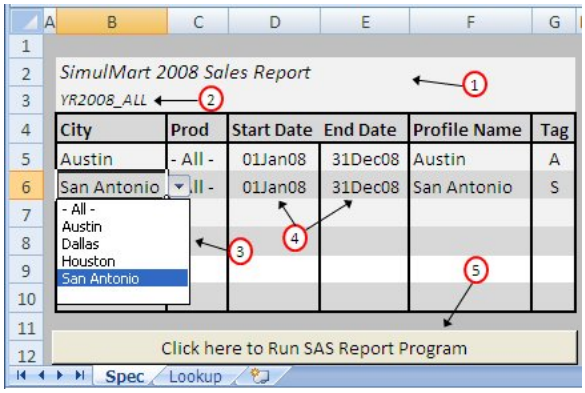
T

a

issue Excel commands directly, and open, manipulate, and close Excel files. One can also move around the spreadsheet freely, write the data, and format the sheet. Note that both methods need to identify the target for the data, a main challenge in customizing Excel report using DDE. This paper will show how to calculate the location in a structured report.

3. DESIGN AND USE OF AN EXCEL INTERFACE

A user-friendly interface is not only the key to the success of the development of the program, but also vital to the success of its application. First, based on the user's request, identify the scope of the report and its parameters. Next, create an Excel sheet that is easy for data input, data validation, and program maintenance and expansion. In addition, the interface should make it easy for SAS to read and parse the information. The following is a simple example of such an interface.



Features of the Excel Interface

1. Full Report Name to appear on the report.
2. File name of the report: Short, descriptive, and no special characters.
3. Dropdown list for value selection using Excel Excel names defined in the Lookup tab.
4. Standard SAS date format.
5. Command button to run an Excel macro that starts a SAS program with the shell() function.

The following SAS code reads and parses the information on the Excel interface. The report description is saved in a dataset instead of a macro variable to allow for any special characters in the description. The variable "region" is made long enough for the report description.

```
*Read the information;
filename Excel dde "Excel|[Report Form.xlsm]Spec!C2:C7" notab;
data reportDesc(keep=region rename=(region=reportDesc))
     reportSpec;
   informat region $100. dept $10. startDate $10. endDate $10. profile $50. tag $10.;
   infile Excel delimiter='09'x dsd pad missover;
   input region dept startDate endDate profile tag;
   if region='' then stop;
   select(_n_);
      when(1);
      when(2) output reportDesc;
      when(3) call symput('reportName',trim(region));
      when(4);
      otherwise output reportSpec;
      end;
   run;

*Parse the information;
data reportSpec;
   set reportSpec end=eof;
   format SASCond $200.;
   if region ne '- All -' then SASCond=trim(SASCond)||' and region="'||trim(region)||'"';
   if dept ne '- All -' then SASCond=trim(SASCond)||' and dept="'||trim(dept)||'"';
   if startDate ne '- All -' then SASCond=trim(SASCond)||' and month>="'||trim(startDate)||'"d';
   if endDate ne '- All -' then SASCond=trim(SASCond)||' and month<="'||trim(endDate)||'"d';
   SASCond=left(SASCond);
   if SASCond=:'and' then SASCond=substr(SASCond,5);
   if eof then call symput('nSpec',compress(put(_n_,2.)));
```

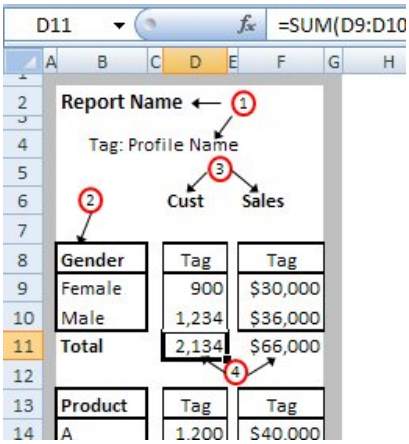
```
run;
```

SAS Data Set: ReportSpec

Region	dept	startDate	endDate	profile	tag	SAS Cond
Austin	- All -	01Jan08	31-Dec-08	Austin	A	region="Austin" and month>="01Jan08"d and month<="31Dec08"d
San Antonio	- All -	01Jan08	31-Dec-08	San Antonio	S	region="San Antonio" and month>= "01Jan08"d and month<="31Dec08"d

4. DESIGN OF A REPORT TEMPLATE

Using a report template takes advantages of Excel's formatting and graphing facilities and reduces coding in both Excel and SAS. A well-designed template can also help SAS programs identify the data content and locations. The template should make it easy for the report modifications and should be macro-free for the tranportability because of network security restrictions. For Excel 2007, files with macros usually have the extension .xlsm and files without macros have .xlsx.



Features of the Report Template

1. Information from the Excel interface. The tag legend will be expanded by SAS automatically for more profiles.
2. Preselected topics/variables. The topic name can be used by SAS to identify its position.
3. Predefined metrics/panels. Each panel will be expanded by SAS automatically for more profiles.
4. Preformatted cells with formulas to reduce SAS coding.

The structure of the above template can be defined using following two macro variables:

```
%let topics = Gender 2 8 |Product 3 13;  /*Triplet: Var nVal StartRow*/
%let panels = Cust D | Sales F;          /*Doublet: Panel Col*/
```

One can also write SAS codes to read the template and identify its structure; however, it may create an unnecessary burden for future code maintence. Panel columns are expressed in letters for direct identification but converted to numbers by SAS for DDE programming.

```
*Separate panel names & positions and convert column letters to numbers;
data _null_;
   format panelName $50. panelCol $10.;
   nPanel=count("&panels",'|')+1;
   do i=1 to nPanel;
      panelName=trim(panelName)||' '||scan(scan("&panels",i,'|'),1);
      col_A=upcase(scan(scan("&panels",i,'|'),2));
      col_N=0;
      do j=1 to length(col_A);
         col_N=col_N*26+rank(substr(col_A,j,1))-rank('A')+1;
         end;
      put i= col_A= col_N=;
      panelCol=trim(panelCol)||' '||compress(put(col_N,2.));
      end;
```

```
   call symput('panelName',trim(left(panelName)));
   call symput('panelCol',trim(left(panelCol)));
   call symput('nPanel',compress(put(nPanel,2.)));
   run;
```

The following code makes a copy of the template and expands and writes the tag legend section. It illustrates that the DDE technique may not be as difficult as it first appears.

```
options noxwait xsync;
data _null_;
   reportName="&projPath\Report\&reportName..xlsx";
   call system("copy "||quote("&projPath\SAS \Report TMPL.xlsx")||" "
                      ||quote(reportName));
   call system("start Excel"||quote(reportName));
   sleep=sleep(2);
   run;

*Write/expand the tag legend section;
Data _null_;
   set reportSpec;
   file Excel;
   if _n_=1 then do;
      put '[Activate("' "&reportName..xlsx" '")]';
      put '[Select("R2C2")]';
      set reportDesc;
      put reportDesc &tab;
      end;
   r=4+_n_-1;
   put '[Select("R' r +(-1) 'C2")]';
   if _n_>1 then put '[Insert(3)]';
   put @6 tag ' - ' profile &tab;
   run;
```

The following code calculates the location of each data segment. The result is saved in a SAS dataset to facilitate DDE coding and debugging. The code uses the array feature to avoid a macro module.

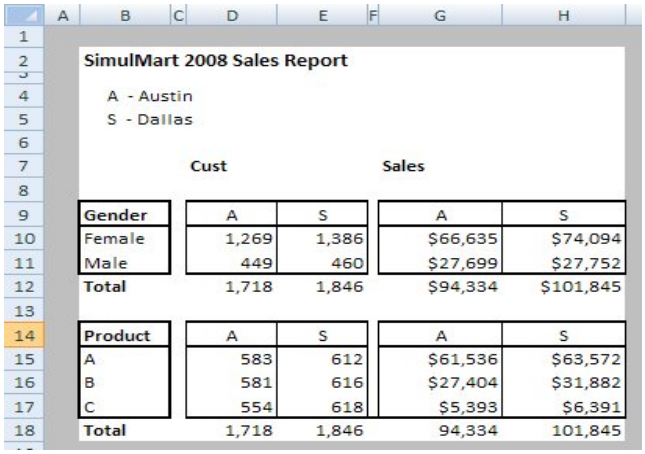
```
data segPos(keep=topic nValue row &panelName);
   format topic $20. nValue 2. row 2.;
   array Panel(&nPanel) &panelName;
   nTopic=count("&topics",'|')+1;
   do t=1 to ntopic;
      temp=scan("&topics",t,'|');
      topic=scan(temp,1,' ');
      nValue=input(scan(temp,2,' '),2.);
      row=input(scan(temp,3,' '),2.)+&nspec-1;
      do p=1 to &nPanel;
         panel(p)=input(scan("&panelCol",p),2.)+(p-1)*(&nSpec-1);
         end;
      output;
      end;
   call symput('nTopic',compress(put(nTopic,8.)));
   run;
```

SAS Data Set: segPos

Topic	nValue	row	Cust	Sales
Gender	2	9	4	7
Product	3	14	4	7

5. DATA SUMMARIZATION AND EXPLORTATION

Data can be easily summarized and written to Excel with the help of the SAS datasets *reportSpec* and *segPos created earlier*. The following is the report from the simulated data.



SimulMart 2008 Sales Report

A - Austin
S - Dallas

	Cust		Sales	
Gender	A	S	A	S
Female	1,269	1,386	$66,635	$74,094
Male	449	460	$27,699	$27,752
Total	1,718	1,846	$94,334	$101,845

Product	A	S	A	S
A	583	612	$61,536	$63,572
B	581	616	$27,404	$31,882
C	554	618	$5,393	$6,391
Total	1,718	1,846	94,334	101,845

REFERENCES

Cohen, J., and Shields, J. (2004) "SAS® /DDE: How We Make Our Customers Happy and Still Use SAS® ", *NESUG 17, Hands-On Workshops*.

Derby, N. (2008) "Revisiting DDE: An Updated Macro for Exporting SAS® Data into Customer-Formatted Excel® Spreadsheets", *SAS® Global Forum*, Paper 259-2008.

Hoffman, J. (2003) "Using DDE to Communicate between SAS® and Excel®". SCSUG 13.

Vyverman, K. (2002) "Creating Customer Excel® Workbooks from Base SAS® with Dynamic Data Exchange: A Complete Walkthrough". *The Twenty-Seventh Annual SAS® Users Group International Conference*, Paper 190-2002.

Watts, P. (2005) "Using single-purpose SAS® macros to format Excel® spreadsheets with DDE". *The Thirtieth SAS® Users Group International Conference*, Paper 089-30.

Fend. Y. (2005) "Generating Customer Report Tables: Using SAS® with DDE and VBA". *The Thirtieth SAS® Users Group International Conference*, Posters.

ACKNOWLEDGMENTS

I wish to express my gratitude to Rick Chamber, Sr. Director of Marketing, Zale Corp, for his unwavering support for the development and implementation of the methodology in our marketing programs. I would also like to thank Phil Nousak for his assistance preparing this paper. He has provided valuable suggestions, both technical and linguistic.

CONTACT INFORMATION

Your comments are greatly appreciated. Contact the author:

Steven X. Yan

Email: STEVEN_X_YAN@HOTMAIL.COM
Phone: 972-580-5558 (Work)
972-523-9806 (Cell)