Paper 044-30

A Microsoft© Access GUI for SAS© Automation

Rubin Nan, PRA International, Lenexa, KS
David Mullins, PRA International, Lenexa, KS

ABSTRACT

This paper describes techniques that use MS-Access Visual Basic tools to create a Graphical User Interface (GUI) to run SAS© and facilitate the passing of parameters to SAS. Using this GUI, users can fulfill any SAS task by embedding SAS codes in the Visual Basic script, and also execute separate SAS programs with or without macro variables. This paper will illustrate the following: how to create front-end components and setup their properties, how to embed SAS codes in the Visual Basic script, and how to use a GUI to run a separate SAS program with or without macro variables.

INTRODUCTION

MS-Access has a complete set of Visual Basic tools. Using these tools, a programmer can design a GUI as a “switch board” to integrate SAS and a MS-Access’ user-friendly database and report system. Suppose some tasks such as routine data analysis, database maintenance (updating, deleting, appending, etc.), figure or table generation, documentation, etc. do not need employees to understand SAS or the underlying database. All of these tasks can be done through a GUI, just by clicking a few buttons. Therefore, it is very efficient and reduces overhead.

Selecting MS-Office embedded Visual Basic tools for designing the GUI has several advantages: 1) MS Office is globally used in business. 2) The ODBC techniques on data exchanges between SAS databases and MS-Access are well-developed. Through the integration of the two software, all tasks such as data manipulation and statistical analysis can be done from an MS-Office GUI; and all SAS-generated output such as documentation, reporting, etc. can be directly exported as MS-compatible format files for either storage or as deliverables to clients.

This paper describes the techniques of GUI implementation using the MS-Office embedded Visual Basic tools to drive SAS sessions to fulfill various tasks. The sending of SAS output back to the GUI involves relatively complicated Visual Basic skills and is beyond the scope of this paper. Also, the techniques described are workable based on our experiences. Other workable - or even better - techniques are possible.

DEVELOPMENT OF A GUI AND AUTOMATION

Visual Basic script can submit SAS code. This can be done with the Visual Basic script language and one command button, without the need to write a separate SAS program for each task of opening a SAS session, running an existing SAS program, assigning a value to a macro variable, etc. (see the following Parts 1, 2, and 3). This coding is simple and has little chance to invite errors.

However, it is not good practice to use the Visual Basic script language to do a complicated SAS task, because the SAS program embedded in the Visual Basic script is not portable. This increases programming complexity and difficulties in error checking during both the SAS and Visual Basic programs’ developmental phase. Therefore, this task is better done as follows:

- A relatively complicated GUI consisting of command buttons, text boxes, list boxes, options, etc., through which users can input parameters for VB variables or macro variable.
- Visual Basic script language that integrates the Visual Basic and SAS tasks.
- A separate SAS program or macro (see the following Part 4).

In the following we describe the development of automation with a GUI for the various tasks to be done:

1. Drive the SAS session to open: Assume a command button was set up with a property name, Run SAS, and SAS.EXE is located at C:\Program Files\SAS\sas.exe. The code in the Visual Basic module will be:

```
Private Sub Run_SAS_Click()

      Dim olesas As Object
      Set olesas = CreateObject("SAS.Application")
      olesas.Visible = True
End Sub
```

2. Drive the SAS program to run: Once the SAS session is open, the program can be submitted from the session. This event can also be done with the GUI. Suppose we need to submit a SAS program, autoexec.sas (as all other SAS programs do in this paper), residing in C:\SAS\ directory , we just need to add one more line of the above VB script to the Visual Basic module as follows:

```
Private Sub Run_sas_Click()

      Dim olesas As Object
      Set olesas = CreateObject("SAS.Application")
      olesas.Submit ("%inc 'C:\SAS\autoexec.SAS '; ")
      olesas.Visible = True

End Sub
```

Please note although we only use one submit function in our example here, multiple submit function statements can be used to run multiple SAS programs, data steps, procedures, or macros. For example, if one SAS program, print.sas, needs to be run after autoexec.sas, another line of the VB script, olesas.submit ("%inc 'C:\SAS\autoexec.SAS '; "), just needs to be added after the first submit statement.

3. Drive SAS statements, DATA steps, or procedures to run: The Visual Basic script can submit SAS code to the SAS session. The following is an example of using the PRINT procedure to print a dataset located at C:\SAS.

```
Private Sub Run_sas_Click()

     Dim olesas As Object
     Set olesas = CreateObject("SAS.Application")
     olesas.Submit("libname dm ‘C:\SAS’; proc print data=dm.demo; run;")
     olesas.Visible = True

End Sub
```

4. Drive macros or programs with macro variables to run: Complicated tasks such as driving macros or programs with macro variables to run need the following:

a. A GUI with the various Visual Basic components.
b. Visual Basic script.
c. A separate SAS program or macros that are both independent of the Visual Basic script. This allows programmers to develop the SAS program independently of the GUI.

To clarify, we use a simple example that uses PROC CONTENTS to get data definition information such as variable name, format, length, and type; and then save the output into a spreadsheet as a report or documentation file.

The dataset could be from a different project, which means it may be located in a different folder. Each project may have its own autoexec.sas file to define its data libraries, macros, page size, line size, font, etc.

To do such a task, we need to design a GUI (shown as Figure 1) with various components, to set up the component’s properties (shown in Table 1), to write the Visual Basic code (shown as attached Visual Basic code), and the SAS code (shown as attached SAS program, print.sas).

The GUI components include a common button to drive the SAS session to run, a text box, and two list boxes (text boxes are also doable) to allow users to respectively input 1) the folder where the autoexec.sas, print.sas, and the .XLS document file are located, 2) the data library name, and 3) the dataset name.

The SAS code can be a SAS program with macro variables or just a SAS macro. We use one text box and two list boxes as examples to input parameters (macro variable values). These can be replaced by each other. However, a list box can contain multiple, predetermined parameter values; whereas a text box can only contain one parameter value for each input, and this parameter value can be set up as default.

5. Automation control: The script language for controlling the SAS session from the GUI is available in Visual Basic. The script includes checking whether the SAS session is busy, toggling the SAS session between visible and invisible, setting the main SAS window title of the SAS session, ending the SAS session, checking errors, etc. The attached example Visual Basic code shows just part of the automation control. Others can be seen in detail in Reference #2.

Figure 1 – GUI design example to run the print.sas program, located at C:\sas, to print the dataset name, variable length, type, format, and label as a file into MS-Excel; using one or all dataset(s) located at the data library, Master or Analysis.

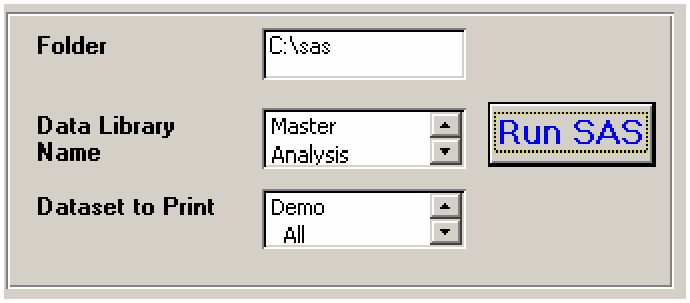

Table 1 – GUI component's property setup

Label	Component	Component Name	Default Value or Caption
Folder	Text Box	Txtfolder	C:\sas
Data Library Name	List Box	Lstlib	Master, Analysis
Dataset to Print	List Box	Lstdsn	Demo, _All_
	Command Button	Run SAS	Run SAS

Visual Basic Code:

```
Private Sub Run_SAS_Click()

     'Error checking starts
     On Error GoTo Err_Run_sas_Click
     Dim olesas As Object

     Dim txtfolder As String
     Dim lstlib As String
     Dim lstdsn As String

     folder = txtfolder.Value
     lib = lstlib.Value
     dsn = lstdsn.Value

     Set olesas = CreateObject("SAS.Application")
     olesas.Submit (" %let dsn=" & dsn & ";  %let lib=" & lib & "; ")

     olesas.Submit (" %let folder=" & folder & ";  %inc '&folder\print.sas'; ")

    'Set up the SAS© session running visible during run time

     olesas.Visible = True

    'The following two lines of code is for ending the SAS© session
     olesas.Quit
     Set olesas = Nothing

     Exit_Run_sas_Click:
     Exit Sub
```

```
        ‘The following three lines of code provides the error message
        ‘if there is any errors

        Err_Run_sas_Click:
           MsgBox ("Please check folder, autoexec.sas, and print.sas")
           Resume Exit_Run_sas_Click
     End Sub
```

SAS© Program, print.sas:

```
    %inc “&folder\autoexec.sas”;
     Proc Contents data=&lib..&dsn (keep=memname name type label format) noprint;
     run;

     Proc Export data= &dsn outfile = "&folder\&dsn.xls"  dbms=excel2000 replace;
     Run;
```

CONCLUSION

The passing of parameters from a user-designed MS-Access GUI to SAS can be implemented. By using such a GUI, the SAS session and its run-time activities can be set up to take place behind the scenes. Users do not necessarily need to be knowledgeable of SAS to do data manipulation, statistical analysis, database maintenance, figure or table generation, documentation, or delivery of clients’ reports through an Intranet or the Internet.

This can improve efficiency, reducing the skill requirements to do such tasks and increasing the utilization of SAS programmers.

REFERENCE

1. McGowan, K. (2002). Using Visual Basic to Customize a Set of SAS Reports. Proceedings of the Twenty-Seventh Annual SAS Users Group International Conference. Paper P164-27.

2. _________ (2002). Example of Automating SAS© with OLE. http://jeff-lab.queensu.sas.ca/stat/sas/sasman/sashtml/win/z2ples.htm.

3. Holland, R.H. (1997). Accessing SAS© Data without using SAS© Code. http://www.hollandnumerics.co.uk/bcarda/bcarda.htm.

CONTACT INFORMATION

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

Rubin Nan
PRA International, Inc.
16400 College Boulevard
Lenexa, Kansas 66219
Work phone: (913)-577-2882
Fax: (913) 599-0344
Email: NanRubin@praintl.com

David Mullins
PRA International, Inc.
16400 College Boulevard
Lenexa, Kansas 66219
Work phone: (913)-577-2946
Fax: (913) 599-0344
Email: MullinsDavid@Praintl.com