

Using Excel's Active-X Controls as a GUI for SAS Solicit File Creation

Eric A. Reece, Alliance Data Systems, Gahanna OH

INTRODUCTION

Our solicit file append process was originally done through SAS/Access, but it took considerable amount of time to extract the information. After exploring other options we decided the fastest way to append the consumer name and address information would be through the extraction of the table to a SAS permanent dataset. An index on the consumer_id field was then put on the dataset so we had the ability to reference that in our merge step. We gained significant speed running our solicit file append through this method, but we still lacked a simple user interface that allowed all associates to change variables and would be easy to understand.

In an effort to standardize code and make changing macro variables easier for all users many programmers use %Let and %Window. Although, both of these have their own limitations and are not very user-friendly programmers do not have much choice if they want to avoid the tedium of surfing through code to change variables. Worse yet, if someone that does not understand SAS code is trying to change variables they may delete something accidentally or mistakenly type something wrong into one of the variables.

CHALLENGE

The biggest challenge of a programmer writing programs that non-technical users will execute is designing them to be easy to understand and change variables. If the user is unclear about a variable or setting within the code the results may not be as desired. Another issue found in the commonly used %Let is that the user must put the variable in front of the semi-colon. If for some reason the user accidentally deletes this in the code it will not run at all. %Window is an option that allows for a simple GUI of text inputs although; the biggest limitation is the inability to copy and paste from another source.

A final challenge is automation; what if the programmer wanted the SAS code to run automatically upon opening up a session without any user inputs from within SAS? This is a huge challenge if the project requires flexibility and many changing variables.

SOLUTION

Excel can be used in the place of %Let and %Window to gather user input into a SAS program. Also Excel has many options and is very easy for the user to understand. The programmer also has access to a variety of toolbox controls to achieve the best interface for all users. Ex 1.1 shows what our process entailed using %Let as our macro variable input.

Ex 1.1

```
%Let Project=XX78X3 ;
* Project ID;
%Let TD_Export=S:\Campaigns\Client A\2003\Spring\XX78X3\Textfiles\XX78X3_TD_Export.txt ;
* Top Dog Exported file Path and Filename;
%Let Segload=S:\Campaigns\Client A\2003\Spring\XX78X3\Excel\XX78X3_SEG_LOAD_TEMPLATE.xls ;
* Segload Template Path and Filename;

*****;
* Please use Path only for M,C,P Outfiles
* Leave Macro Blank if no file is to be read out
* eg. S:\2002- Client A\Fall 2002\XXN4X2\Project Execution\Final Files\Vendor;

%Let M_out=S:\Campaigns\Client A\2003\Spring\XX78X3\Final Files\Vendor ;
* Final Mailfile Path only;
%Let C_out= ;
* Final Controlfile Path only;
%Let P_out= ;
* Final PermControlfile Path only;

%let email1=jmarkwood;      * Specify Email Address of Program Manager;
%let email2=ereece;        * Secondary Email contact;
```

This is a very cumbersome way to enter variables into a program. It does not afford much flexibility to the programmer or the user. You also must rely on the user to enter all information correctly without the ability to restrict valid values that can be entered.

The solution to this problem was found by using Excel's capabilities of Active-X controls. We configured a worksheet with three sub sheets. The first sheet labeled 'Input' Ex 1.2 contains the controls that the user will see for selecting the various inputs for the program. The second sheet labeled 'Values' Ex 1.3 stores all of the valid values for list boxes and other controls. Finally, the third sheet labeled 'Data' Ex 1.4 contains all values eventually selected by the user for final input into the program.

Ex 1.2

CMA MAILTAPE INPUT	
Path Options Brand Client A Season ☐ Fall Season ☐ Spring Season Year 2004 Project Number XX51X4	Dump and Freq Options Freq and Dump ☐ Run with MT code ☐ Do not run ☐ Email to ☐ Yes ☐ No ☐ Program Manager Zip File Path Email1 none @alldata.net Email2 none @alldata.net Email3 none @alldata.net Analyst Email none @alldata.net
Layout Options Layout ☐ Standard ☐ Telemarketing EAN ☐ Yes ☐ No	
File Paths and Run Options TD Export Path Mailtape Run Address Dataset Address Dataset updated to: 05/29/2004	
Progress Bar Yes Address Dataset Current Segmentation	

Ex 1.3

Brand	Email1	Email2	Email3	AnalystEmail	TDVersion	Address Dataset	ProgressBar
Client A	none	none	none	none	TopDog 6.2	Current Segmentation	Yes
Client B	anelson	anelson	anelson	erece	TopDog 6.4	Previous Segmentation	No
Client C	bmorrow	bmorrow	bmorrow	jbalint			
Client D	jvitazka	jvitazka	jvitazka	lzimmerman			
Client E	kklaus	kklaus	kklaus	snagaraja			
Client F	ltian	ltian	ltian	ssweeney			
Client G	slang	slang	slang	tstires			
Client H	sschatzke	sschatzke	sschatzke	wcohn			
Client I	tsweeney	tsweeney	tsweeney				
Client J	tmarkwood	tmarkwood	tmarkwood				
Client K	ziosilevich	ziosilevich	ziosilevich				
	bstoddard	bstoddard	bstoddard				
	jmarkwood	jmarkwood	jmarkwood				
	astevenson	astevenson	astevenson				
	mblair	mblair	mblair				

Ex 1.4

Brand	Fall Season	Year cd	Layout Standard	EAN NO	F and D	Email	Email1	Email2	Email3	AnalystEmail	Project no
Client A	FALSE	2004	TRUE	TRUE	FALSE	FALSE	none	none	none	none	AFN8X3

The Excel will take care of the input of the users, but we are still left to get the data into our SAS program. To accomplish this we can use a simple Proc Import command to reference our Excel sheet. Remember to change the datasheet that you will be referring to within Excel.

Ex 1.5

```
proc import
  datafile= "&Input_Sheet." dbms=excel97 replace out= input_var;
  sheet="Data";
  getnames=yes;
run;
```

Once all values have been read in with the Proc Import we need to assign them to macro variables. To accomplish this we use a series of Call Symput statements. Ex 1.6 shows our variables being assigned to a Brand Div macro depending on which was selected by the user within Excel.

Ex 1.6

```
data check;
set input_var;

* Brand Macro *;
if brand="Client A" then Call Symput("Brand_div", "CA");
  else if brand = "Client B" then Call Symput("Brand_div", "CB");
  else if brand="Client C" then Call Symput("Brand_div", "CC");
  else if brand="Client D" then Call Symput("Brand_div", "CD");
  else if brand="Client E" then Call Symput("Brand_div", "CE");
  else if brand="Client F" then Call Symput("Brand_div", "CF");
  else if brand="Client G" then Call Symput("Brand_div", "CG");
  else if brand="Client H" then Call Symput("Brand_div", "CH");
  else if brand="Client I" then Call Symput("Brand_div", "CI");
  else if brand="Client J" then Call Symput("Brand_div", "CJ");
  else if brand="Client K" then Call Symput("Brand_div", "CK");
```

At this point we have solved the issue of user input and a GUI with Excel. It offers more choices for interfacing and is much more robust and user-friendly than %Window or %Let. However, we are still left with one macro that the user must input into SAS that references the Excel sheet (shown as macro Input Sheet above in Ex 1.5).

To solve this issue we must have a system of folders set up that is organized to be the same every time we have a new solicit file project. Utilizing SAS' own internal stored variables it recognizes the pathname at which the code was opened. As long as our Excel file with our macro inputs is within the same folder structure we can make SAS do the work for us when we open up the code within our Editor. Ex 1.7.

Ex 1.7

```
proc sql outobs=1 noprint;
  select xpath into :this_pgm
    from sashelp.vextfl
   where fileref eqt '#LN'
   order by fileref desc
  ;
quit;

data something;
  file = symget("this_pgm");
  length path $150.;
  path = substr( file, 1, index( file, 'Code\' ) - 1 ) ||
  "Excel\Segload_Template_1.11.xls";
  call symput( "Segload", path);
run;

%put &this_pgm.;
```

BENEFITS

Some of the benefits realized from creating a GUI from Excel are as follows

- Training associates

- For non-technical users it was much easier for them to change values through Excel than through a %Let statement within SAS. The interface is much easier to look at and understand so it allowed us to eliminate many errors that were occurring from associates changing values within the SAS program.
- Minimize errors
 - %Let statements do not eliminate non-valid values from being entered into a program. Also the user could potentially delete a semi-colon from the macro statement, which would cause an error. Using Excel all of these errors can be eliminated. Valid values are the only values that will show up in list boxes, radio buttons and checkboxes. The user also does not need to worry about changing anything in the SAS code since it is completely automated based on the values from Excel.
- Faster turnaround times
 - Associates found it much easier to enter information into a user-friendly environment. Also, minimizing the errors allowed us to create our solicit files much faster which in-turn allowed us to spend more time on training and developing other processes.

CONCLUSION

Using Excel as the source for macro variable input is a viable alternative to the more traditional %Let or %Window functions. It is very easy to understand for a non-technical user and allows for complicated SAS programs to be run simply and efficiently. Excel only allows particular values to be selected so the user does not have to 'guess' what the value should be. Finally, using SAS internal variables to find the Excel document and read the appropriate variables affords the user the ability to open the code and press run without having to explicitly state a pathname for a file. All of these things make Excel a great way to allow users to input values into a productionalized SAS process.

AUTHOR CONTACT INFORMATION

Eric A. Reece
Alliance Data Systems
800 Techcenter Dr.
Gahanna, OH 43230
Work: (614) 729-4388
Email: ereece@alldata.net