Delivering SAS to Excel Jockeys

Sy Truong, Meta-Xceed, Inc., Milpitas, CA
Jing Xu, Providian, Pleasanton CA

ABSTRACT

The ubiquity of Microsoft Excel and Word on desktop computers has made it a default entry point for many users to view and edit their information. Once Excel was included into the MS Office suite in 1993, it became the killer app overtaking other spreadsheet heavy weights such as Lotus 1-2-3. Although Excel has analysis capabilities, it does not have the powerful statistical procedures and depth of SAS. When analyzing certain types of data, such as financial information, Excel is the tool of choice. Its capabilities to easily generate graphs along with the visual pivot table provide powerful methods to view and analyze data. In certain cases, however, Excel is not capable of performing particular tasks which SAS can provide. Some of the topics that this paper will elaborate on include:

- **Connecting SAS to Excel** – The use of TCP port communication allows Excel to connect directly with a SAS session
- **SAS Macro Management –** Managing SAS macros that can be delivered to Excel users
- **Deliver SAS Data –** SAS Datasets can be delivered directly to Excel or Word in optimized smaller blocks
- **Pivot Table and Graph –** SAS data can be formulated and delivered to Excel users in the form of a Pivot Table and/or Graph (Excel Chart)

The union of SAS and Excel enables power users and decision makers, who may not be SAS programmers, to fully explore their business data with the full analytical power of the SAS system.

INTRODUCTION

The advent of the modern spreadsheet has revolutionized how users analyze data such as financial data. It visually displays information in cells similar to how a professor may describe the information on a chalk board. The disadvantage of a chalk board is when a change or mistake is made. Rather than having to erase each item on the board, the cells of a spreadsheet can be updated through a formula. Programming languages have evolved to work with spreadsheets so they can provide a very efficient method for financial analysts to perform analysis interactively. There are many advantages to this approach but there are also some limitations. Some of these limitations include:

- **Change Control –** The changes to the spreadsheet are very interactive so the information stored previous to updates are lost. This can lead to regulatory compliance issues or limit the ability to easily roll back to an earlier version.
- **Security –** Spreadsheets are designed for individuals to work on their own set of data. They cannot easily handle multiple users with different permissions. This combined with the lack of change control makes it difficult to function as a secured system for large sets of data in a large organization with many users.
- **Cell Based Formula –** Formulas defined for a cell in Excel are defined for a particular cell. An example is to sum up all the cells by their identified cell row and column name, =SUM(A1:A3). In this example, the cells A1, A2 and A3 are summed to a specified new cell such as A4. This is limited to cell A4 and is not easily replicated in an array of multiple cells. There have been scripting languages to enhance the capabilities of formulas but it is still limited since formulas are designed as expressions and not designed for complex algorithms which require a full featured programming language.
- **Multiple Users –** Spreadsheets are designed for a single user. This creates limitations if multiple users need to update the same data.
- **Statistical Analysis –** Formulas are designed as an expression to derive at a numeric answer. There is some statistical analysis that can be applied through a spreadsheet but this approach is limited. It is not optimized for more complex statistical modeling such as performing a multi-variate regression analysis.

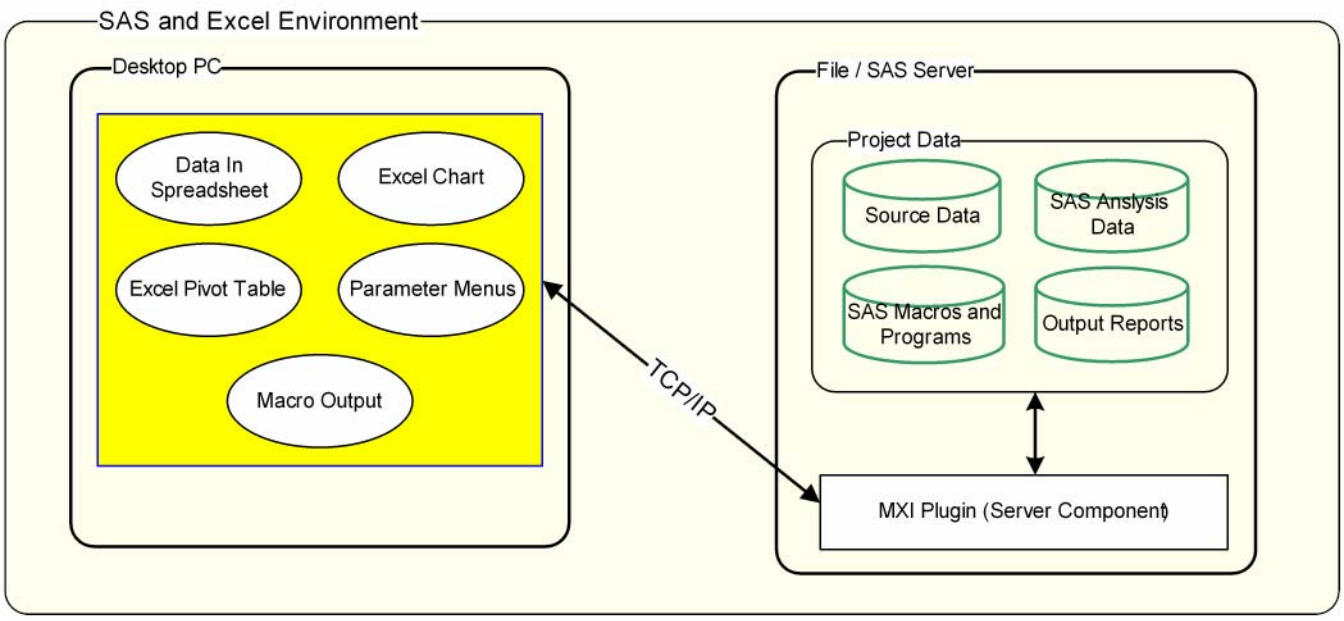
The exploratory and visual aspects of spreadsheets make them very suitable for certain types of financial calculations. However, some of the limitations mentioned above can prevent power users from performing data mining to truly model business conclusions that can be delivered by SAS. SAS has the powerful programming language including a library of statistical procedures which deliver to users functionality beyond the capabilities of spreadsheets. The two tools can function together symbionically to form a complete solution. This paper will explore the integration of SAS and Excel in ways that give SAS programmers methods of delivering to Excel jockeys the power of SAS without requiring them to program SAS.

CONNECTION SAS TO EXCEL

There are several ways to deliver data or connecting information between SAS andExcel. Depending on the requirements, the following solutions can provide the optimal methods of delivery.

Communication Method	Description
Cut and Paste	Text output data can be cut and pasted into Excel through the clip board. If the source data contains the correct delimiter, the pasted values will be populated into a spreadsheet with proper column alignment.
DDE Dynamic Data Exchange	Data and instructions can be sent from SAS to Excel through Dynamic Data Exchange. This can deliver the most up to date data dynamically. It can also include some Excel functions so that the output can be placed onto a particular cell or worksheet.
TCP/IP Socket	Data and instructions can be sent from SAS to Excel through the same protocol that is used on most networks across the Internet.

There are advantages and limitations to each of the three methods mentioned above. This paper will explore the third option of delivering data and analytical computing power of SAS to Excel via TCP/IP socket communication. This process will make it possible to have a SAS macro execute on the server and the results delivered directly to Excel.



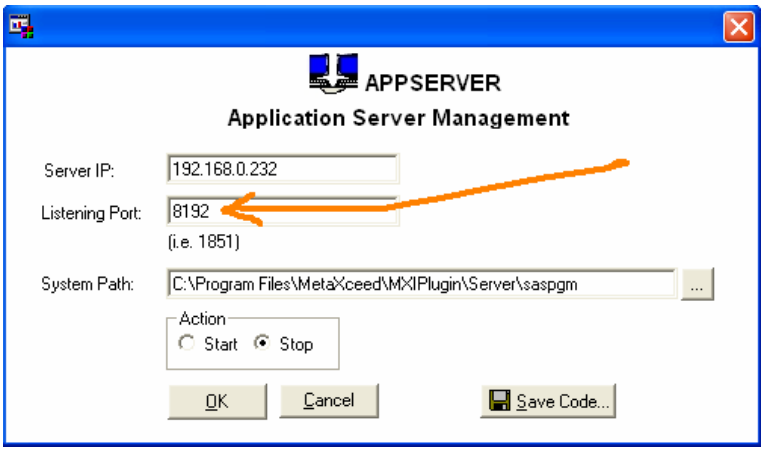
AUTHENTICATION AND SECURE ACCESS

There are multiple layers of security that can be applied to ensure that the data being delivered from SAS to Excel is protected. Some or all of the methods mentioned here can be implemented for optimal security. The security measures include:

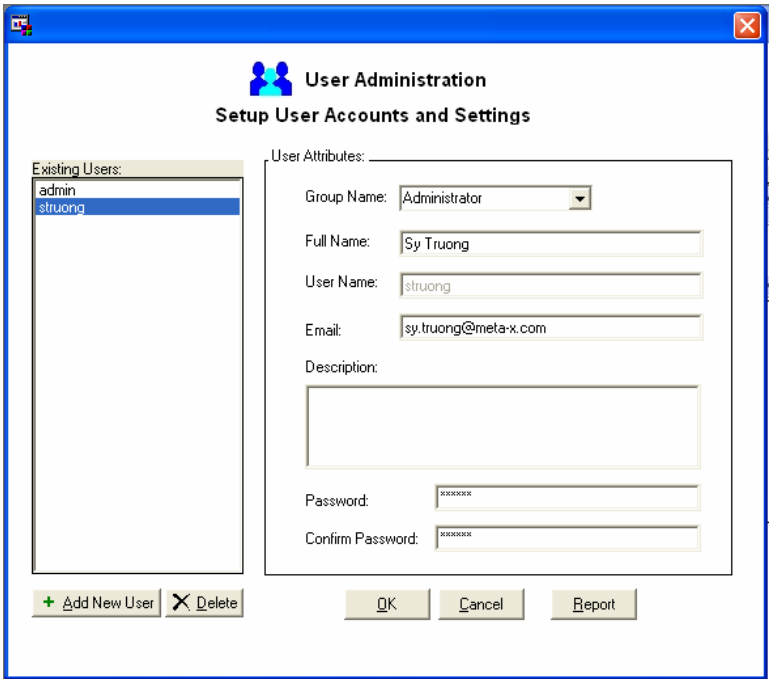
1. **OS Privileges** – The operating system in which the computer system resides acts as the first line of defense. This will authenticate the user who will be able to execute the SAS program and access Excel. This also acts as a layer to prevent users from accessing certain datasets that contain sensitive data.
2. **Data Control** – SAS datasets can contain password protection to prevent unauthorized access. This can be set for read or read write depending on the needs of the particular user.
3. **Application Control** – The application used in this example, MXI Plugin™, also contains a layer of authentication, both on the server side and client side. This will ensure that only specified members that have been granted permissions will have access.

The OS and dataset controls are well understood. The unique approach that is discussed in this paper will demonstrate how the application applies an extra layer to ensure optimal security. The following steps are applied by the application.

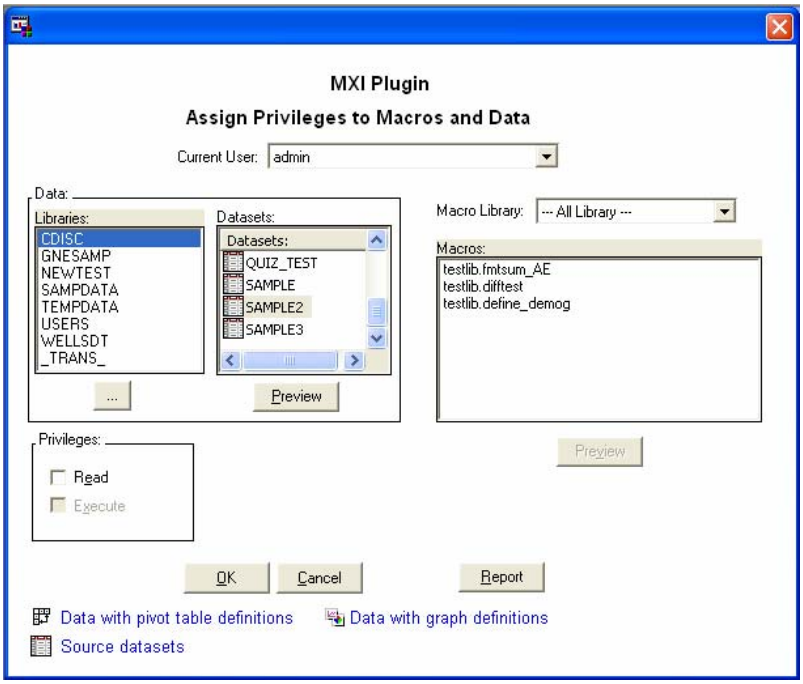
STEP 1: **SOCKET PORT** – The application server is configured to be available only on a specified port. This information is not published so it is more difficult for unintentional users to access the data through this port.



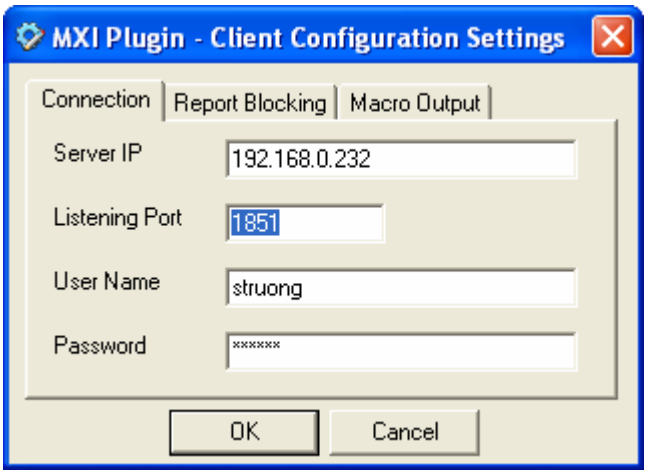
STEP 2: **ACCESS CONTROL LIST**– The application server will have the ability to create user accounts. This is needed to control the access and limit access to specified data and SAS macro programs.



STEP 3: **PRIVILEGE ASSIGNMENTS –** All SAS datasets and SAS macro programs will have user access assigned. This will be controlled at the level of each user. For SAS datasets, the user will have the ability to assign read permissions. For SAS macro programs, the user can have either read or execute privileges granted.



STEP 4: **CLIENT AUTHENTIFICAITON –** The client application in this case is a plug-in application to MS Excel. A dialog box is presented to the user with a way of authenticating to the server. This will authenticate the user and includes the server and socket port number. This will prevent accidental access on default port numbers which other applications use.



STEP 5: **TRANSPORT ENCRYPTION –** The data being sent from the SAS server to the Excel client will be encrypted with a password. This will prevent access to the contents of the data if the data packets are intercepted during delivery.

The five steps demonstrate methods that ensure that the data being delivered from the server to the client is secured. The multiple layers create redundancy to prevent accidental or malicious intent to alter or capture sensitive data.

SAS MACRO MANAGEMENT

SAS macro is a programming language that is an extension of the SAS base programming language. It is a superset of the base language in that you can apply all the same data steps and SAS procedures to fully utilize the power of SAS. In addition, it provides a way of modularizing repetitive portions of your SAS program. It is recommended that when a particular task or programming code segment is repeated more than three times, it should be placed into a macro. This way, it can be repeated in a consistent manner. This provides SAS programmers with a method for automating algorithms while allowing other programmers to apply the same logic without having to completely understand the intricacies of that code segment.

There are a couple of ways in which SAS macros are delivered to other users. The main method is by creating a macro library. In this approach, each macro is placed into a program where the name of the program is the same as the name of the macro, and then placed in specified folders. For example, if you were to have code that calculates the age of an individual, the program can be named calc_age.sas and the corresponding macro is named %calc_age. The macro can therefore be accessible to any program if the path of the program cal_age.sas is defined in the AUTOPATH as a SAS option. Similar to other programming languages, an include statement can also be used to include the external program calc_age.sas into your current program such as in:

```
*** include the macro to calculate age ***;
%include "c:\mypath\calc_age.sas";
```

There are therefore two steps in the process of delivering a SAS macro. These include:

STEP 1: **SETUP MACRO –** Start out by creating a SAS library through the AUTOPATH option and inform the users of this so that they can apply the steps to enable access.

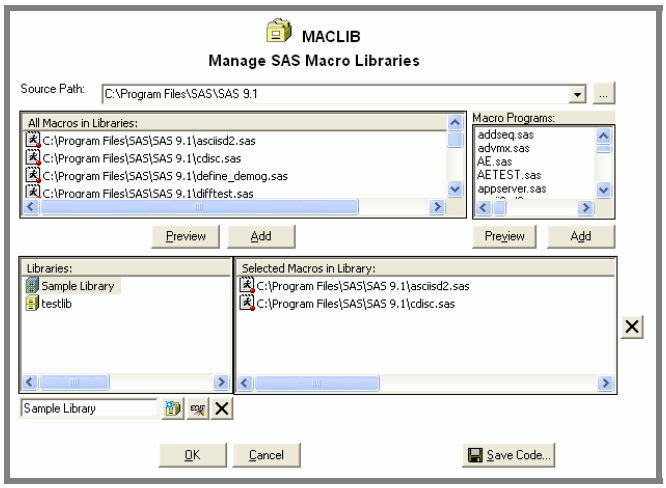
STEP 2: **USING MACRO –** The user would then set up the macro auto path in their autoexec.sas or through applying the include statement to gain access. The user would then call the macro through the name such as in the call:

```
*** Apply the calculation of age ***;
data mydata;
   set sourcelib.data;

   %calc_age;
run;
```

The above traditional steps for managing the SAS macros are designed for SAS programmers. This requires the user to understand the concepts of the macro language and know how to access the macros programmatically through proper syntax. This is appropriate for SAS programmers but can be prohibitive if delivered to non SAS programmers such as Excel jockeys. An alternative and more suitable method is described in the following steps.

STEP 1: **MACRO LIBRARY MANAGEMENT –** A graphical user interface is provided for the macro developer to manage and organize the macros in preparation for delivery.



The concept of a library is similar to the SAS auto path macro library in that a series of macros can be placed into this directory for delivery. In addition to having the path associated, a user friendly library name can be assigned such as "Sample Library". This approach can provide more flexibility since programs from different directory paths can be associated to one library. The method for adding macros to a library in this approach is applied through a graphical user interface. This makes the process more user friendly. However, since this step is generally applied by SAS developers, an additional batch approach is also available. For experienced users, the batch method can be more efficient if many macros are applied at once. In this example, a macro can be added to a macro library through a macro.

```
/*-----------------------------------------------------*
* Program: maclib.sas
* Path: C:\Program Files\SAS\SAS 9.1
* Description: Manage macro libraries for MXI Plugin
* By: Sy Truong, %maclib, 02/18/2007, 12:06:03 pm
*-----------------------------------------------------*/
%maclib(macro = ,
        library = Old Name,
        action = delete);

%maclib(macro = ,
        library = Sample Library,
        action = add);
```

CONVERTING MACRO PARAMETERS

Macro parameter is the main method for modularizing your SAS logic. This allows for you to parameterize and change specific values such as input data or variable names within your SAS logic in order for it to perform many different tasks. For example, a macro can be named "getparam" which stands for "get parameters" as shown below:

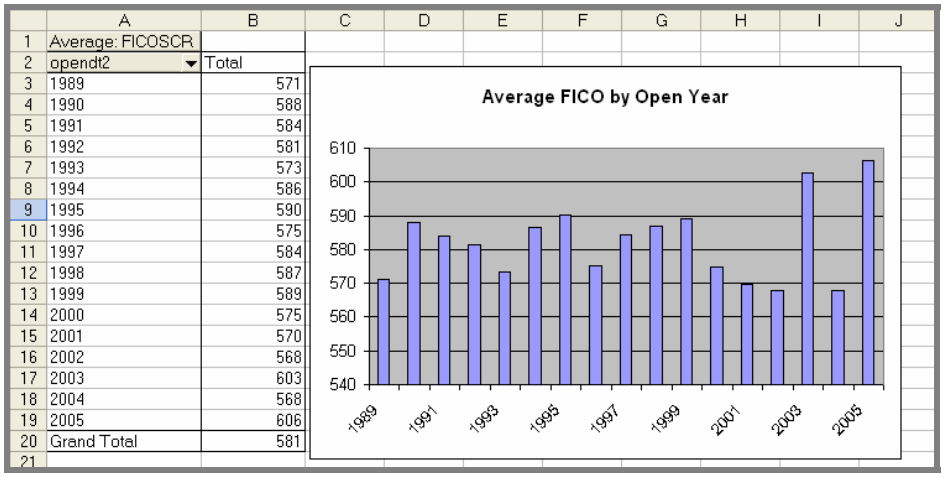
```
%getparam(macro = C:\saspgm\pivottab.sas,
          desc = ,
          param =  datname var type,
          require = ,
          type =  text text radio buttons,
          vallist =  none none SAMPDATA.AE.DOMAIN,
          defval = mydata|id 6 of maps.afghanis|AE);
```

In this example, the macro %getparam macro contains parameters such as MACRO, DESC and PARAM. The values associated with each parameter are then fed into the associated macro code to perform tasks specified by the user request. This fundamental construct of the macro language acts as an interface between the user and the macro itself.

The same parameters can be delivered to users in the form of selection lists and radio buttons that Excel users are more accustomed to. MXI Plugin can set this up so that the user experience is completely within Excel providing the familiar user interface, while delivering powerful analysis on the SAS server.

PIVOT TABLE AND GRAPH

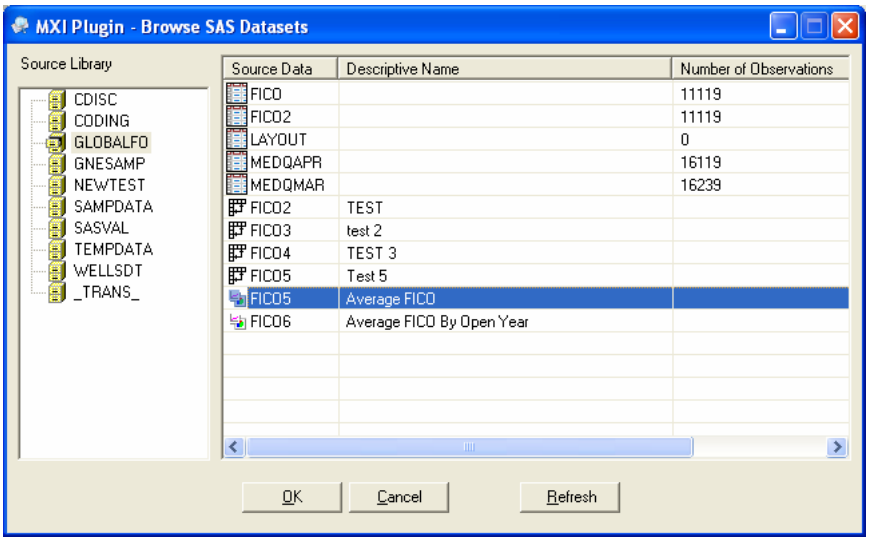
Pivot tables and graphs or charts are powerful tools to explore and display information. This can be demonstrated in an example where credit score information, recorded as a FICO score, is collected for customers. The financial institution commonly asks the question if the latest average FICO score is high or low. This information is summarized across all customers by year. An example result of performing this analysis is shown here:



	A	B
1	Average: FICOSCR	
2	opendt2	Total
3	1989	571
4	1990	588
5	1991	584
6	1992	581
7	1993	573
8	1994	586
9	1995	590
10	1996	575
11	1997	584
12	1998	587
13	1999	589
14	2000	575
15	2001	570
16	2002	568
17	2003	603
18	2004	568
19	2005	606
20	Grand Total	581
21		

The pivot table is shown on the left two columns of the spreadsheet. The user can pull down the account open date "OPENDT2" column and select specific years. The graph to the right will then show the selected pivot table data points in a graph. The pivot table adds flexibility in selecting the data segment of interest. The accompanying graph provides a visual presentation of the same information to enable users to spot trends. The above information can be prepared by the following steps.

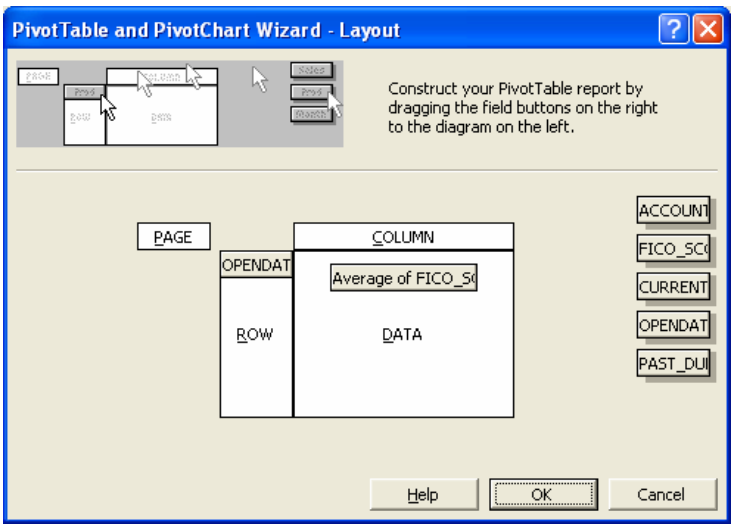
STEP 1: **DELIVER THE DATA –** The data can be delivered from SAS directly into Excel. This is accomplished through the MXI Plugin menu within Excel or through the Excel toolbar. The data selection is then presented through the data libraries that are established on the server.



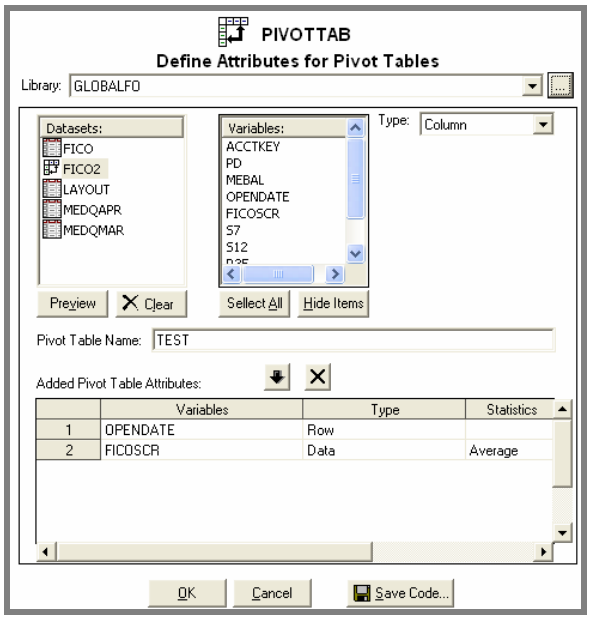
Once the user has been properly authenticated, the data can be delivered to users as SAS datasets, pivot tables or graphs. If

the user decides to select the SAS datasets, the raw data will be delivered to Excel as a spreadsheet. In this case, the user would continue to the next step to establish the pivot table.

STEP 2: **SETUP PIVOT TABLE –** If the user selects the source data from the list of SAS datasets, the data will be presented as a spreadsheet. In this case, the user can then use Excel to set up the pivot table.

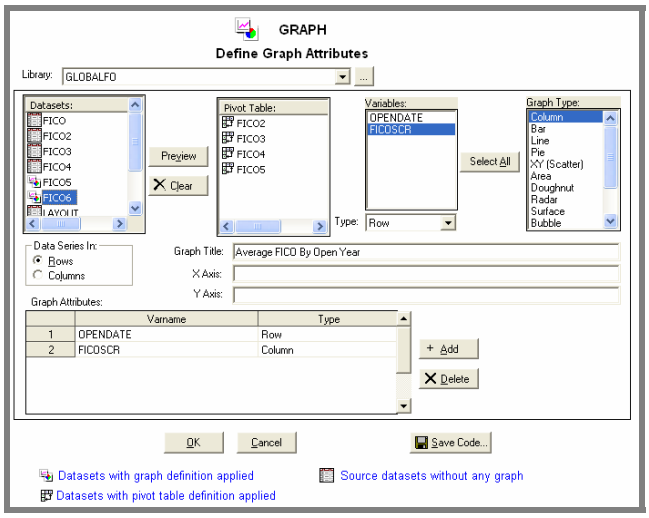


The Excel Layout tool allows the user to select the row such as the open date in this example. The column of the pivot table can also be assigned as seen in this example where the average FICO score is selected. This does require a little work within Excel for the user to then establish the pivot table. Alternatively, you can set up the same pivot table and have it delivered directly to Excel by using the MXI Plugin server pivot table setup tool.



In this case, the open date and FICO score variables can be selected in a similar manner. The difference is that this is being established beforehand so that the user can select the pivot table directly as a sample data source. This setup occurs on the SAS server, but once delivered to the client, it will appear as a native Excel pivot table.

STEP 3: **SETUP GRAPH/CHART –** The last step in this analysis is to create a graph or chart from the selected pivot table. The user has the choice of selecting data from the raw source or from the pivot table. This information is then used to create the graph within Excel. An alternative method is to have this process set up on the server. This adds greater flexibility since the graphs can be preprocessed on the server and delivered directly into Excel as an Excel native chart, or it can be generated on the client Excel spreadsheet.



In this example, the source to the graph can be either the source data or a pivot table. You would then choose the column and row variables. The titles and types of graphs can be selected in a similar way that the wizard in Excel provides. The difference is that you are preparing to have this delivered completely on the server prior to its delivery to the user on Excel.

The above steps show how you can deliver SAS data to Excel either as a spreadsheet, pivot table or graph. You can leave it up to the user to then create their own pivot table and graph within Excel, or have this prepared ahead of time so that the user can have the most updated data on the server delivered in the form of a pivot table or graph. This empowers Excel users to access SAS data with visual display formats which are native to Excel.

CONCLUSION

MS Excel has become the default standard for data analysis on desktop client computers. SAS has established itself as a powerful analytical data mining tool on servers. This paper presents a unique approach using MXI Plugin which marries the two tools enabling SAS data sets to be delivered directly to Excel. The delivery method uses the standard TCP/IP as the underling protocol for communication. There are multiple layers of security that are put into place to prevent accidental or intentional data corruption. The powerful analytical logic of SAS is delivered to Excel in multiple ways including: SAS macro, SAS data, pivot table and graph. This information is delivered in a user friendly manner since users can select parameters of a macro in a form of pull down lists or check boxes. The SAS data is presented as spreadsheets and pivot tables and graphs can be delivered in Excel formatted pivot tables and charts. This empowers Excel users with many options to access the power of SAS without having to program SAS. This approach provides Excel jockeys with an interface that is familiar to Excel users while delivering the full power of the SAS system.

REFERENCES

ABOUT THE AUTHOR

Sy Truong is President of MXI (Meta-Xceed, Inc.) They may be contacted at:

Sy Truong
Meta-Xceed, Inc. (MXI)
1751 McCarthy Blvd.
Milpitas, CA 95035
(408) 955-9333
sy.truong@meta-x.com

Jing Xu
Washington Mutual Card Services
Pleasanton, CA
4940 Johnson DR
Pleasanton, CA 94588
(925) 738-5430
jing.xu@wamu.net