

Choosing the Right Tool from Your SAS® and Microsoft Excel® Tool Belt
Steven First and Jennifer First, Systems Seminar Consultants, Madison, Wisconsin

ABSTRACT

There are over a dozen ways to bring your data into SAS from Microsoft Excel or to push it out from SAS to Excel. Some are simple wizards, and others are more complex programming techniques. How do you know which tool is the best for your application? Which will cause you the least number of headaches? Which will produce the quickest, most accurate results? Which will best satisfy the end user? We will give an overview of import and export techniques for Excel and make recommendations for different application types.

INTRODUCTION

While SAS software provides analytics, reporting, and programming second to none, Excel is probably the most popular software product used today. Because of its ease of use and flexibility, Excel truly runs many parts of business. To use both SAS and Excel gives us an opportunity to allow both software products to do what they do best. As an input source, Excel provides easy navigation and data input which would be very useful to analyze with SAS. As an output destination, Excel is an excellent report distribution and delivery device to reports, graphs, and other output that can show results from SAS in a familiar format.

Some questions to answer are:

- Are the structures similar?
- How do we transfer from SAS files to Excel and vice versa?
- Are there formatting differences?
- What is the best import/export method for your project?

SAS AND EXCEL CAPABILITIES

SAS software is excellent for analytics after data has been captured. SAS is somewhat batch oriented and may require programming skills depending on which products are used. SAS has extensive programming and reporting capabilities, as well as being an excellent data manipulation tool.

Excel is a much more interactive and simple tool. It is very easy to use and learn, and users can begin using Excel almost immediately. Keying data and formulas into cells is straight forward without programming. Excel provides an almost universal format for exchanging and delivering reports and worksheets. While most users don't do much programming in Excel, there is a macro language, as well as interfaces to several programming languages available.

SAS AND EXCEL STRUCTURES

SAS datafiles and Excel worksheets are shaped similarly, in a row and column fashion. Data values are stored in similar manners, and each system handles formatting and even date values in a similar way. SAS has a required dataset descriptor that acts like a data dictionary, defining the columns. Individual columns must contain all numeric or all character values. Excel does not require columns to be named, and its columns can contain a mixture of character, numeric values, or formulas.

The shapes of SAS and Excel data are usually close enough to make transfer from SAS to Excel fairly automatic and easy. However, it can depend on what SAS products, operating system, or special data formats a user has.

The names of SAS files are more restrictive than Excel workbooks. SAS generally doesn't allow special characters in names, where Excel uses the '\$' character in naming workbooks as a default. SAS can reference those names if it uses special naming rules.

SAS AND EXCEL LIBRARIES

SAS stores its data in SAS libraries. SAS libraries are roughly the equivalent of a Windows or Unix directory, with each SAS file being a separate Windows or Unix file. Similar techniques store SAS data on mainframes as well.

The SAS LIBNAME statement normally names a SAS library. Using the idea of a library with members in it, Excel workbooks, which are a single file, can be considered a library, with each range being a separate member inside it. By defining libraries and members this way, it should ease transfer between SAS and Excel.

SAS TO EXCEL

Before we can do any copying, we need to decide whether we want to copy a SAS data file, or a SAS report. We can do both.

SENDING SAS REPORTS TO EXCEL

Sending SAS output to Excel used to require capturing the report in a print file via PROC PRINTTO and then, in a separate process, parsing through the report to create suitable input for Excel. With the advent of ODS, this process has become much simpler. ODS can capture any SAS report and route it to various destinations. One of the simplest files to create is an HTML file. Normally this type of file is used to create web pages, but because Excel does an almost automatic process to convert HTML to Excel format, this process becomes trivial.

Suppose we produce the following report with proc tabulate:

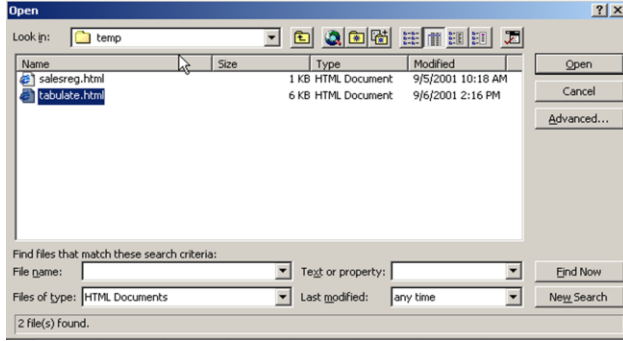
```
proc tabulate data=softsale;
  title 'Softsale Sales and Expenses by Division';
  class state division ;
  var sales expense;
  table state,division*sales*sum
         division*expense*sum;
run;
```

	Division		Division	
	H	S	H	S
	Sales	Sales	Expense	Expense
	Sum	Sum	Sum	Sum
State				
IL	4009.21	743.22	322.12	159.45
MN	3231.75	7732.44	644.55	1339.45
WI	12185.10	8232.11	2786.10	3339.41

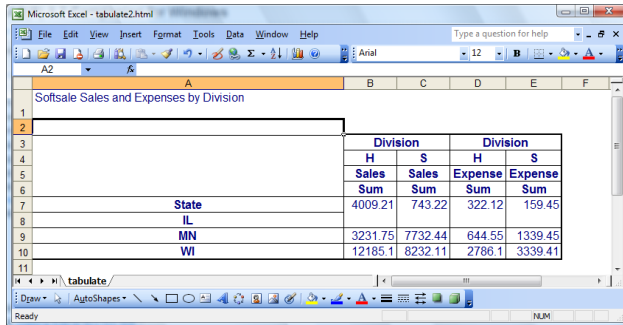
We can easily route the output to HTML with a few ODS statements.

```
ods html body='c:\temp\tabulate.html';
proc tabulate data=softsale;
  . . .
run;
ods html close;
```

To open the file in Excel, we can use File Open, but we do have to ask for the file type of HTML.

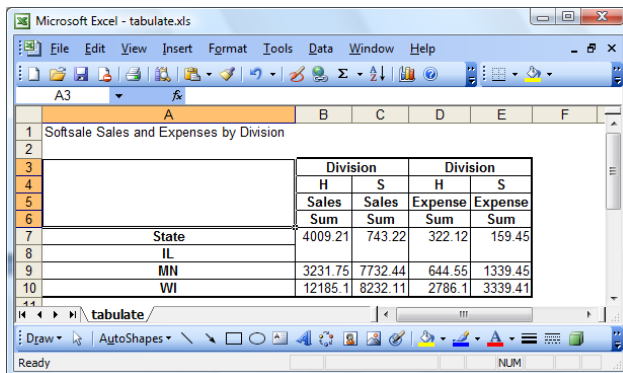


Excel automatically converts the HTML on the way in, and the user may have to specify to save it as an XLS format.



We can also make this process more transparent by continuing to generate HTML but naming our file with .XLS. We are tricking Excel and Windows to think they are getting an Excel file. The file will be converted automatically upon input, and this also allows the user to double click on the file to load Excel. If color and styles aren't critical, we can use style=minimal, which can reduce the file size significantly.

```
options nocenter;
ods html body='c:\temp\tabulate.xls' style=minimal;
proc tabulate data=softsale;
. . .
run;
ods html close;
```



Excel may format things differently than SAS does. The most common examples are leading zeros that don't appear in Excel or text that doesn't span columns in the worksheet the way you would like it to. In our SAS code, we can program a formula to actually write an Excel formula and also control HTML spanning. Alternatively, there may also be Excel formatting commands to handle these problems.

Examples:

```
Account='="00003444" ';  
Title '<td colspan=5> My Title </td>';
```

ODS MARKUP DESTINATIONS

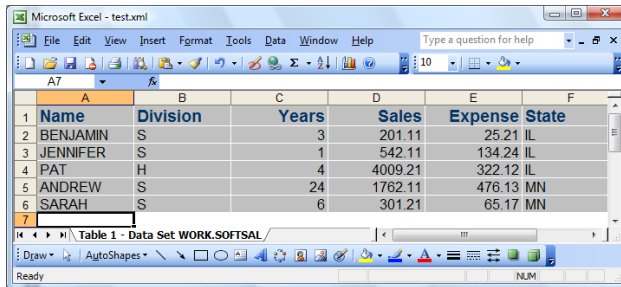
While the method above works well, there are more controlled and flexible ODS families of destinations that provide a wealth of different markup languages. This family of destinations is designed to create markup codes that can be easily input to other software. The ODS statement can specify MARKUP as the destination along with a TAGSET= option to generate specific tagset codes. An alternate form is to specify the TAGSET.tagset as the ODS destination. There are tagsets for XML, CSV, EXCELXP, and many more. One that looks very useful is the EXCELXP tagset which was designed exclusively for input to Excel.

EXCELXP allows you to capture SAS reports as we did above but has many more options to control almost anything in the spreadsheet, such as styles, titling, footnotes, by lines, worksheet names, and much more. To see the entire documentation for the EXCELXP, the program below can be run.

```
ods tagsets.excelxp file='test.xml' options(doc='help');
```

An example of a simple PROC PRINT using EXCELXP follows. Notice that titles don't appear, as they are instead used as print headers. There are options to override this if desired. Also note that SAS will generate the worksheet name.

```
ods tagsets.excelxp file='c:\temp\test.xml';  
proc print data=softsale(obs=5) noobs;  
  var Name Division Years Sales Expense State;  
  title 'Softsale First 5 Obs';  
run;  
ods tagsets.excelxp close;
```



	A	B	C	D	E	F
1	Name	Division	Years	Sales	Expense	State
2	BENJAMIN	S	3	201.11	25.21	IL
3	JENNIFER	S	1	542.11	134.24	IL
4	PAT	H	4	4009.21	322.12	IL
5	ANDREW	S	24	1762.11	476.13	MN
6	SARAH	S	6	301.21	65.17	MN

The example below uses options to set the worksheet name prefix, which SAS will use to create a separate worksheet for each by group. Note also that the titles are now imbedded in the worksheet itself. There are many, many more options that can control the XML generated.

```
ods tagsets.excelxp file='c:\temp\test.xml' options(embedded_titles='Yes' Sheet_Name='State');  
proc print data=softsale(obs=5) noobs;  
  var Name Division Years Sales Expense State;  
  by state;  
  title ;  
run;  
ods tagsets.excelxp close;
```

	A	B	C	D	E	F
1	Softsale First 5 Obs					
2						
3	Name	Division	Years	Sales	Expense	State
4	BENJAMIN	S	3	201.11	25.21	IL
5	JENNIFER	S	1	542.11	134.24	IL
6	PAT	H	4	4009.21	322.12	IL
7	ANDREW	S	24	1762.11	476.13	MN
8	SARAH	S	6	301.21	65.17	MN

COPY DATA FILES TO EXCEL

The examples above all showed copying SAS reports to Excel, but there are also a wealth of ways to copy a SAS datafile to Excel and eliminate the reporting step as required with ODS. Below we have listed some of the methods first requiring only Base SAS or Enterprise Guide.

COPYING AND PASTING FROM SAS TO EXCEL

Copying and pasting all the cells from a SAS dataset in the SAS VIEWTABLE window is not supported, though a single cell can be copied. So, that leaves you with 2 quick options: Use Enterprise Guide to copy and paste or copy and paste from The SAS System Viewer. There is complicated SCL code that you can write for the cut and paste functionality if you'd like, but that is not a quick option, and you might be better using another method.

The SAS System Viewer is a free Windows application that can view SAS data. You can download it from The SAS Institute at: http://www.sas.com/apps/demosdownloads/92_SD_L_sysdep.jsp?packageID=000512. To use this copy and paste option, open The SAS System Viewer and open the data set you want to send to Excel. Select all of the rows and columns you want (or the whole dataset), use the Windows shortcut to copy (Ctrl+C) because there is no edit menu. Then, in Excel, use the Windows shortcut to paste (Ctrl+V). Please note that you should be able to paste all of your data into Excel, but you may not be able to paste in variable names.

You can also copy and paste from EG to Excel. Open your dataset, highlight all of the rows and columns you want, go to your Edit menu and select copy. Then, in Excel, go to your Edit menu and select paste. This should paste your data into an Excel spreadsheet, preserving the rows and columns from your dataset.

WRITING PROGRAMS TO SEND SAS DATA TO EXCEL

A data step can be written to create a comma separated value text file that can be easily imported into Excel. The wide capabilities of the data step make this a very flexible tool. The DSD FILE option automatically inserts a delimiter (default comma) between values and quotes the values if they contain special characters. When this file is opened in Excel, it is automatically parsed into columns. If desired, the Tilde (~) format modifier can be specified if you would like all fields quoted. The ODS CSV destination along with a proc print can also give virtually identical results.

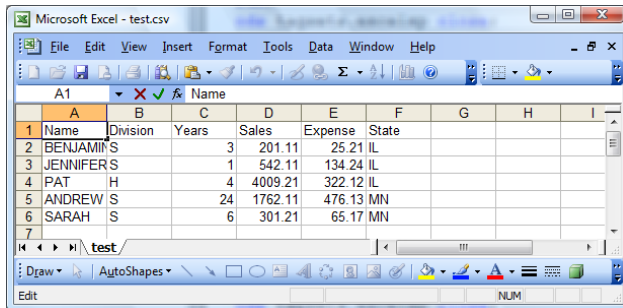
Drawbacks to using the DATA step are that you need to specify the names of the columns in your SAS dataset if you would like a header row and that you would have to alter the data step for each dataset used. The ODS CSV destination along with a proc print can also give virtually identical results.

```
data _null_;
  set softsale(obs=5);
  file 'c:\temp\test.csv' dsd;
  if _n_ =1 then
    put 'Name,Division,Years,Sales,Expense,State';
  put ( _all_ ) (+0);
run;
```

File 'c:\temp\test.csv'

```
Name,Division,Years,Sales,Expense,State
BENJAMIN,S,3,201.11,25.21,IL
JENNIFER,S,1,542.11,134.24,IL
PAT,H,4,4009.21,322.12,IL
ANDREW,S,24,1762.11,476.13,MN
SARAH,S,6,301.21,65.17,MN
```

Here is the Excel view after opening the above file.



USER WRITTEN MACROS TO PRODUCE CSV FILES

There are many user written macros that are widely available to create CSV files from SAS files and produce almost identical results to the file above. The SAS Inc. macro %ds2csv and our own %SSCFLAT macro both read dictionary tables to gather information about the SAS dataset and produce the CSV file with minimum user coding. Options are available to include or exclude header rows, to use column names or labels as the first row, and many others. These macros work well and can be downloaded from SAS or SSC without charge.

DDE

Dynamic Data Exchange (DDE) is a method of dynamically exchanging information between Windows applications.

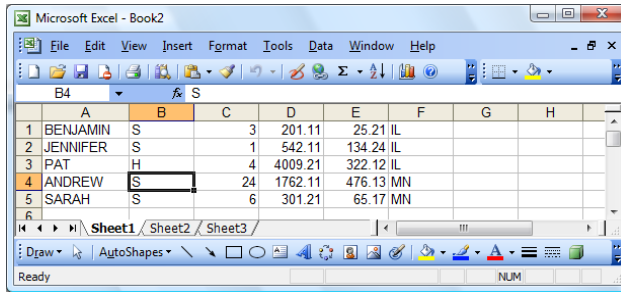
Almost any command or input that you might key into Excel can be done by the data step and DDE, and this technique can produce very powerful logic using the best of the DATA step, along with Excel. DDE has existed for many years and uses a client/server relationship to enable a client application to request information from a server application. SAS is always the client. In this role, SAS requests data from server applications, sends data to server applications, or sends commands to server applications. This method allows the user to harness the power of the data step while interfacing with Excel and other applications.

Excel must have an open worksheet before running the SAS code. A DDE server application can be opened using the X command within SAS code. The XWAIT and XSYNC options must be turned off.

```
options noxwait noxsync;
x 'c:\microsoft office\office\excel.exe';
```

The example below sends the first 5 rows and 7 columns from a SAS session to respective rows and columns in an Excel worksheet.

```
filename excelout dde 'excel|sheet1!r1c1:r5c7';
data _null_;
  set softsale(obs=5);
  file excelout;
  put Name Division Years Sales Expense State;
run;
```



The advantage to using this method is that DDE and SAS are very flexible systems for reading and writing data. Both systems are mature and have many capabilities. The disadvantage is that you must know both languages, and the syntax can be difficult to code correctly.

SAS/ACCESS INTERFACE TO PC FILE FORMATS

If you have this product licensed, transfer of data becomes even easier. The product contains multiple methods for converting data:

- Export and Import wizards to convert data to and from multiple file formats
- PROC EXPORT/IMPORT code can be captured and coded to run the process through SAS
- The Excel Libname engine allows Excel workbooks to be treated as SAS libraries
- PROC SQL pass-through facility can also treat Excel worksheets as SAS datasets
- PROC ACCESS to create access and view descriptors
- PROC DBLOAD to load Excel worksheets

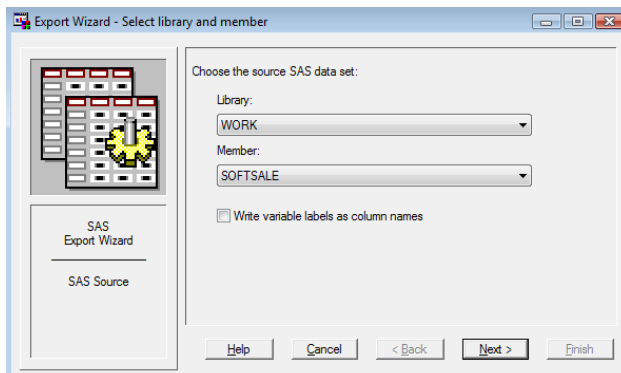
SAS/ACCESS is used to access third party data formats and convert them to and from SAS. Most users would use one of the first four techniques listed above. PROC ACCESS and PROC DBLOAD are generally older products that are included mostly for compatibility to earlier SAS releases. Because ACCESS and DBLOAD procedures are compatible only with SAS 6 procedures, they ignore SAS system options such as VALIDVARNAME=. PROC ACCESS and PROC DBLOAD have other SAS 6 limitations, such as a maximum of 8-byte SAS variable names and a maximum of 200-character value.

USING THE EXPORT WIZARD

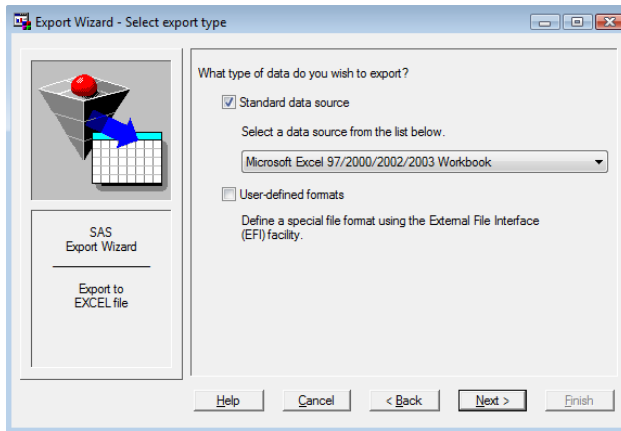
The SAS Export Wizard is a point-and-click interface that guides you through the process of reading data from a SAS data set and writing it to an external file format. External file formats can include many other formats besides Excel.

Like all wizards, we just follow instructions and click. If we need help, we can click the Help button.

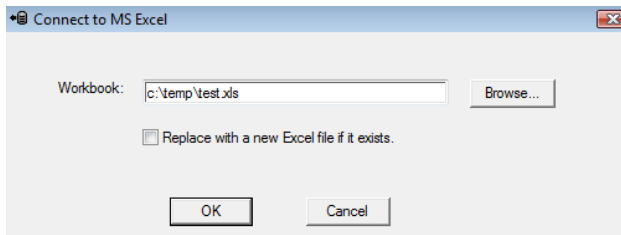
1. Open the Export Wizard from the SAS windowing environment by selecting File then Export Data. Select Library. The member window opens, and we select Library and Member in the Export Wizard.



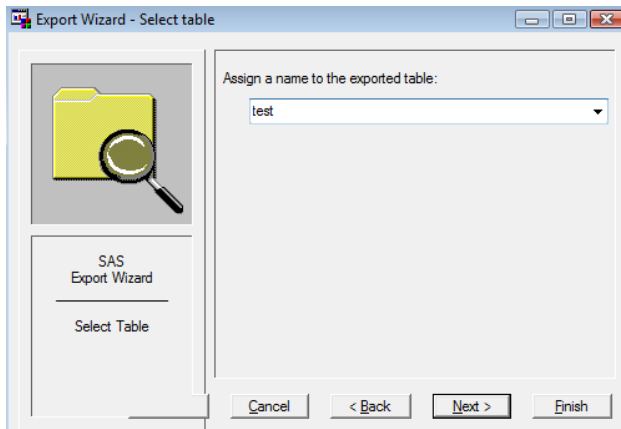
2. Select the input SAS data set from which you want to export data. You can select from library and member drop-down lists or type respective entries and then click the Next button. The Select export type window opens. Select Export Type in the Export Wizard.



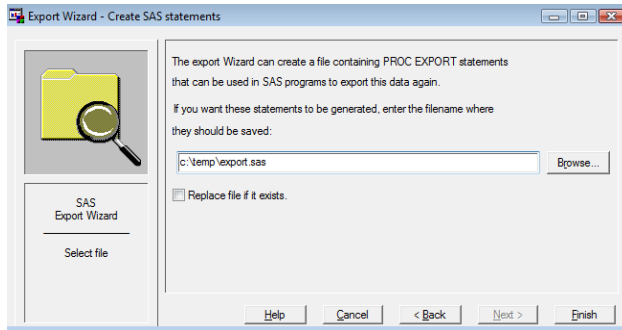
3. From the drop-down list, you can select the type of files to create and click Next. The Connect to MS Excel dialog box opens.



4. You assign the output file by typing or browsing and click OK. The Select Table window opens.

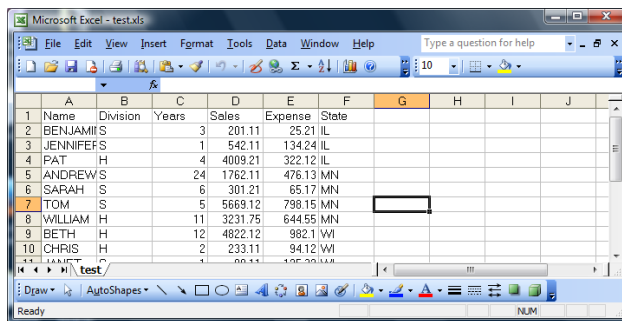


5. Select a name from the drop-down list or enter a new name. The Create SAS statements window opens.



- At this point, you can save the generated PROC EXPORT code. To do this, enter a filename or select a file from the drop-down list, select the Replace file if it exists check box (optional), and click the Finish button.

Here is the resulting worksheet:



PROC EXPORT

If you prefer to code rather than use wizards, PROC EXPORT code that was either captured from the wizard or coded from scratch gives repeatability and many more options. The EXPORT procedure reads data from a SAS data set and writes it to an external data source, including Excel and others. PROC EXPORT exports the data by either generating DATA step code, generating SAS/ACCESS code, or using a translation engine. Here are the required statements and arguments saved by the wizard in our previous example:

```
PROC EXPORT DATA= WORK.SOFTSALE                                /* name of SAS ds in          */
            OUTFILE= "c:\temp\test.xls"                        /* worksheet file going out   */
            DBMS=EXCEL REPLACE;                                /* build Excel, overlay if existing */
            RANGE="test";                                       /* range (table name) to create */
RUN;
```

Options are available to use SAS labels for column headers, to specify output sheet names, and other specific options. The SAS documentation is very thorough and thus won't be repeated here.

The advantage to PROC EXPORT is that it is a straightforward conversion that is repeatable. The disadvantage is that coding is required, a copy of the data is made, and conversion issues may surface.

LIBNAME ACCESS TO EXCEL

Probably the most simple of all techniques, the SAS/ACCESS LIBNAME statement, along with the PCFILES engine, extends the SAS global LIBNAME statement to support assigning a libref to Microsoft Excel and other data sources. This allows you to reference an Excel workbook as a SAS library and worksheets within the workbook as SAS data files. While naming conventions and internal storage are different between SAS and Excel, default actions and options generally allow us to read and write Excel data as easily as referencing a SAS dataset.

Before treating Excel as a SAS dataset, it may be worth examining differences between the two systems. One major difference is that while an Excel workbook is similar to a SAS library, equating SAS tables to worksheets can be

problematic, as Excel doesn't require the data to begin in the first rows and columns. Thus another structure is needed in Excel if it is to act like a SAS table. That structure is a subset of a worksheet called a *named range*.

The SAS documentation deals with this topic as follows:

"SAS/ACCESS Interface to PC Files treats an Excel workbook as a database and a range as a table. A range name has to be defined in the Excel file before it can be used by SAS. A worksheet is treated as a special range. A worksheet name appended with a \$ character is treated as a range.

For example, Sheet1 is a sheet name in an Excel file. SAS treats Sheet1\$ as a valid range name and uses it to refer to the whole worksheet. You need to use SAS n-literal when referring to the sheet name, for example, 'Sheet1\$h. The first row of data in a range is normally treated as a column name and is used for SAS variables names".

WHAT CAN THE LIBNAME EXCEL ENGINE DO AND NOT DO?

The SAS libname engine can:

- Create new workbooks
- Create a new worksheet within a named range and write data to that range
- Write data to an empty existing named range
- Append data to worksheet data or a named range
- Read data from existing worksheets and ranges
- Delete all the data in a worksheet or range
- Do all the above without Excel installed.

The SAS libname Excel engine cannot:

- Rename worksheets in a workbook
- Delete worksheets or workbooks
- Change or apply formatting
- Delete cells with formulas
- Write formulas into a cell.

It should be noted that the SAS x command and other techniques can be used to delete workbooks.

WRITING TO A WORKSHEET

While in most cases we will be using SAS to read worksheets (covered later), we certainly can write to worksheets via the LIBNAME engine as well. In the program below, we assign the libref, write out a worksheet named "test", and SAS will create a named range called "test\$". We can treat test like any other SAS dataset. Note that by clearing the libref at the end of the program, the workbook is freed to be opened by Excel.

```
libname myxls excel path="c:\temp\test.xls";
data myxls.test;
  set softsale;
run;
proc print data=myxls.test;run;
libname myxls clear;
```

If the program is run again, now the worksheet exists and the step would fail, unless we precede the code with a delete statement.

```
options noxwait;
x 'del c:\temp\test.xls' noxwait;
libname myxls excel path="c:\temp\test.xls";

data myxls.test;
  set softsale;
```

```
run;
proc print data=myxls.test;run;
```

WRITING TO A WORKBOOK WITH PROC SQL

PROC SQL can also write to a workbook with the CREATE TABLE clause.

```
options noxwait;
x 'del c:\temp\test.xls' noxwait;
libname myxls excel path="c:\temp\test.xls";

proc sql;
  create table myxls.test as
    select * from softsale;
quit;

libname myxls clear;
```

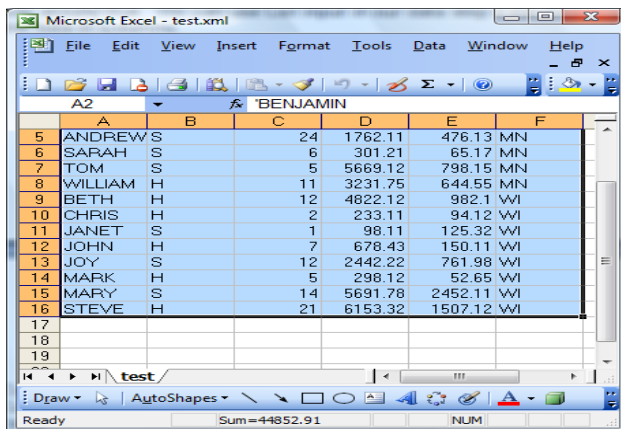
IMPORTING EXCEL DATA INTO BASE SAS

SAS can also read data from Excel with a variety of methods. In many cases, there are almost identical (but opposite) processes that were discussed when writing to Excel earlier. This will be a briefer discussion than before because there is some much that would just be duplication.

COPY AND PASTE TO SAS

Again COPY and Paste is about as simple as it gets, and you can copy cells from Excel and paste them into a data step after a datalines statement. At first glance, Paste does not line the columns up, and at this time we don't know any way around that. We can use List input in our data step, if the data is conducive to that, or we could insert spaces to line up the data in columns.

In the screen below, you would highlight the desired cells and issue the edit copy command (Ctrl C).



	A	B	C	D	E	F
5	ANDREWS		24	1762.11	476.13	MN
6	SARAH	S	6	301.21	65.17	MN
7	TOM	S	5	5669.12	798.15	MN
8	WILLIAM	H	11	3231.75	644.55	MN
9	BETH	H	12	4822.12	982.1	WI
10	CHRIS	H	2	233.11	94.12	WI
11	JANET	S	1	98.11	125.32	WI
12	JOHN	H	7	678.43	150.11	WI
13	JOY	S	12	2442.22	761.98	WI
14	MARK	H	5	298.12	52.65	WI
15	MARY	S	14	5691.78	2452.11	WI
16	STEVE	H	21	6153.32	1507.12	WI
17						
18						
19						

After typing in our datastep and pasting the data after DATALINES, we are ready to build our SAS dataset.

The advantage to this method is that Copy and PASTE are extremely simple. The disadvantage is that we still have to code a data step, and this would only work for small files.

```

data softsale;
  input Name $ Division $ Years Sales Expense State $;
  datalines;
BENJAMIN      S      3      201.11 25.21  IL
JENNIFER      S      1      542.11 134.24 IL
PAT           H      4      4009.21 322.12 IL
ANDREW        S      24      1762.11 476.13 MN
SARAH         S      6      301.21 65.17  MN
TOM           S      5      5669.12 798.15 MN
WILLIAM       H      11      3231.75 644.55 MN
BETH          H      12      4822.12 982.1  WI
CHRIS         H      2      233.11 94.12  WI
JANET         S      1      98.11  125.32 WI
JOHN          H      7      678.43 150.11 WI
JOY           S      12      2442.22 761.98 WI
MARK          H      5      298.12 52.65  WI
MARY          S      14      5691.78 2452.11 WI
STEVE         H      21      6153.32 1507.12 WI
;
run;
proc print data=softsale;
run;

```

READING DIRECTLY FROM THE CLIPBOARD

The CLIPBRD access method (specified on FILENAME), allows the program to read directly from the Windows Clipboard, and as such, you can skip the paste step.

```

filename myexcel clipbrd;
data softsale;
infile myexcel firstobs=2;
  input Name $ Division $ Years Sales Expense State $;
run;
proc print data=softsale;
run;

```

READING A CSV FILE WITH SAS

EXCEL can easily create a CSV file with the File Save As command, choosing a file type of CSV. In the file created below, notice that the first line still contains the headers which probably aren't wanted.

```

Name,Division,Years,Sales,Expense,State
BENJAMIN,S,3,201.11,25.21,IL
JENNIFER,S,1,542.11,134.24,IL
PAT,H,4,4009.21,322.12,IL
ANDREW,S,24,1762.11,476.13,MN
SARAH,S,6,301.21,65.17,MN
TOM,S,5,5669.12,798.15,MN
WILLIAM,H,11,3231.75,644.55,MN
BETH,H,12,4822.12,982.1,WI
CHRIS,H,2,233.11,94.12,WI
JANET,S,1,98.11,125.32,WI
JOHN,H,7,678.43,150.11,WI
JOY,S,12,2442.22,761.98,WI
MARK,H,5,298.12,52.65,WI
MARY,S,14,5691.78,2452.11,WI
STEVE,H,21,6153.32,1507.12,WI

```

The DSD INFILE option will search for the commas to separate the fields upon input. The FIRSTOBS= option can be set to 2, which starts reading with record 2, ignoring the header line.

The advantages to this method are that DSD does most of the work for reading our CSV file, and only Base SAS is needed. The disadvantages are that we have to code and need to specify column names, types, and sometimes lengths.

```
data softsale;
  infile 'c:\temp\test.csv' dsd firstobs=2;
  input Name $ Division $ Years Sales Expense State $;
run;
proc print data=softsale;
run;
```

USING DDE AND XML TO READ EXCEL DATA WITHIN SAS

DDE is supported for reading data into the DATA step and is very similar to what was shown earlier. In fact, a worksheet can be read and written to in the same step. Again, this allows for very specialized and detailed control from the data step.

XML can also be produced from Excel and processed by SAS via an XML libname engine. While possible, XML input is beyond the scope of this paper.

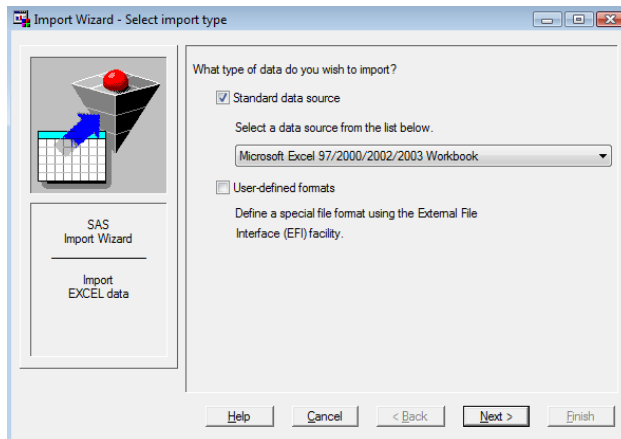
SAS ACCESS FOR PC FILE FORMATS

If you have SAS ACCESS for PC File Formats installed, reading Excel becomes much easier. You can use Import wizards, PROC IMPORT, LIBNAME, and even PROC SQL pass-through to read data.

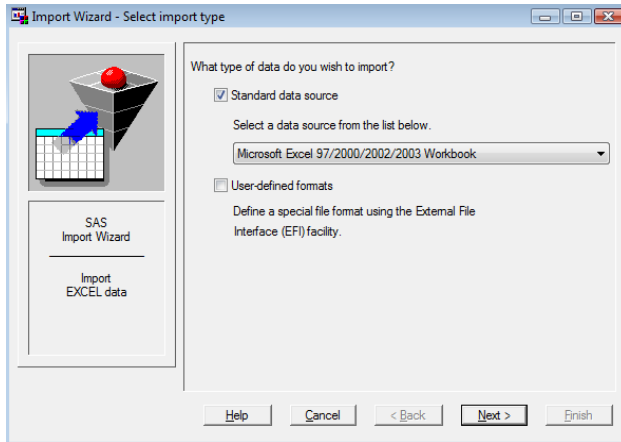
PROC IMPORT WIZARDS

Wizards are available to convert Excel worksheets to SAS files, and they are very similar to the ones shown earlier with PROC EXPORT.

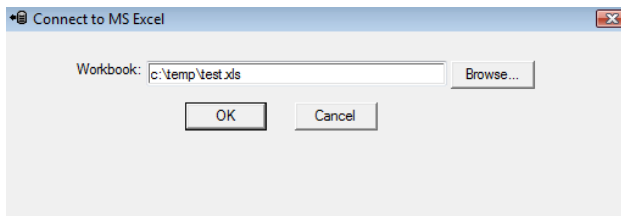
1. To start the process, use the File Import Data menu.



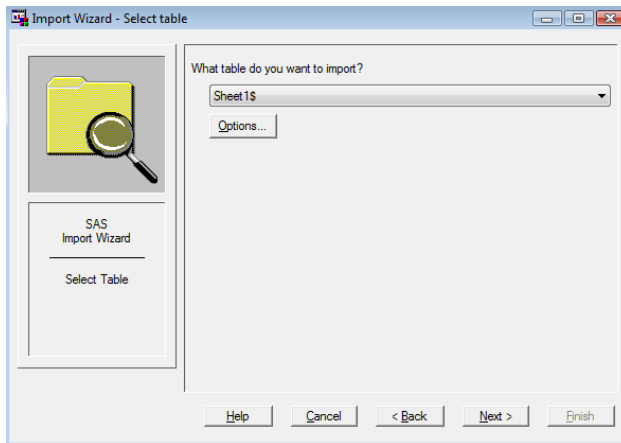
2. You can select an import type of Excel and press Next.



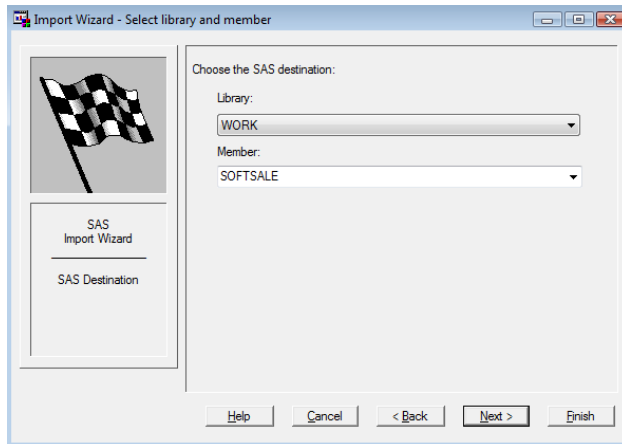
3. You can then fill in the workbook name, or browse to find it, press OK.



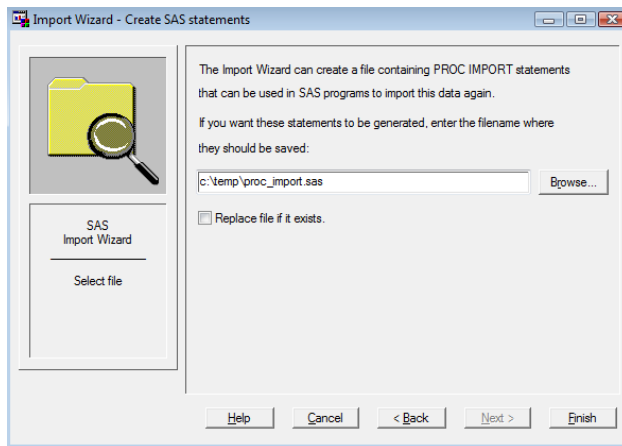
4. Then choose the sheet you would like to import.



5. Name your output SAS dataset.



6. You can capture the generated SAS PROC IMPORT for rerun, then Finish.



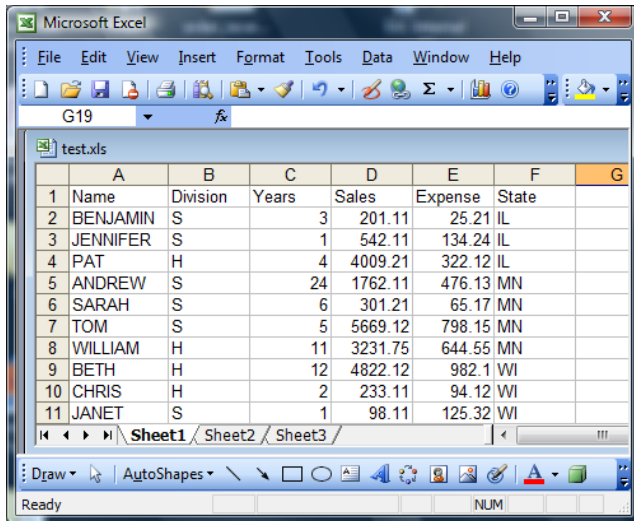
The data file is built, and the SAS code can be re-run at any time.

```
PROC IMPORT OUT= WORK.SOFTSALE
            DATAFILE= "c:\temp\test.xls"
            DBMS=EXCEL REPLACE;
    RANGE="Sheet1$";
    GETNAMES=YES;
    MIXED=NO;
    SCANTEXT=YES;
    USEDATE=YES;
    SCANTIME=YES;
RUN;
```

READING EXCEL WITH LIBNAME

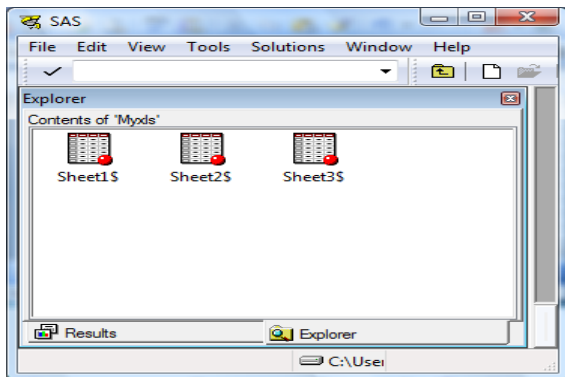
If you have an Excel workbook, LIBNAME makes the workbook appear as a SAS library, and each “named range” appears as a table. LIBNAME has many options to control the way data is read, and this makes it virtually transparent to the SAS user.

This is probably the easiest way to access Excel from SAS. The disadvantages are that SAS/ACCESS must be licensed, and you have to understand a bit about Excel ranges.



New workbooks normally don't have named ranges to start, so SAS will have to assign some. LIBNAME automatically sets up a range name as the sheet name with a \$ appended to it, so that each sheet can act as a SAS dataset. You can verify that by using the SAS explorer window or Proc Contents.

```
libname myxls excel path="c:\temp\test.xls";
```



You can now use the name ranges as you would with any SAS dataset; however, because of the \$ in the name, you have to use special name literal coding by enclosing it in quotes and following the name with an N. If you prefer, you could create a named range, of your choosing, in the Excel worksheet and use that instead of the ranges SAS sets up.

```
proc print data=myxls.'sheet1$'n;
run;
```

THE PROC SQL PASS-THROUGH FACILITY

PROC SQL Pass-Through allows the user to access third party databases via a sub query with a minimum of SAS interference. This is probably easiest when using Excel named ranges. For example: Create a SAS table from the named range "sdata" in our workbook.

```
proc sql dquote=ansi;
connect to excel (path="c:\temp\test.xls");
create table work.softsale as
select * from connection to excel
(select * from sdata);
disconnect from excel;
quit;
```


EXPORTING AND IMPORTING EXCEL DATA INTO ENTERPRISE GUIDE

Exporting data from EG to Excel is very simple. Go to your process flow, right click on the dataset you want to export, and select Send To, then Excel. It will send it to a new workbook.

If you want to import data into Enterprise Guide, you can copy and paste to overwrite a dataset that exists in EG. In our experience, the data you are pasting from Excel needs to be the same dimensions as the dataset. Otherwise we have always gotten an error. When you paste, you may also get an error message saying it must change the dataset from read only before pasting data in.

A much better option for getting Excel data into EG is to follow these import steps. On the File menu, select Open → Data. Select Local Computer or SAS Servers. Navigate to the location of the data set. Highlight the Excel data set name and click Open. Check the boxes next to the tables you want to open and click Open. Then select View The File As Is to open the entire table and use default column headings, data types, and formats.

You can also control the way that your data is imported instead of using SAS's defaults by using the Import Data Wizard. To do this, instead of selecting View the file as is, select Open the file as a SAS data set. This allows you to choose rows and columns to import, the row to use for column headings, and names, types, labels, formats and informats for variables.

To control the region that you are importing from Excel into Enterprise Guide, on the Selection Pane of the Import Data Wizard, highlight Region to Import, and set options. You can choose a line to use as column headers, choose the starting line, and choose the ending line.

If you want to change the column properties while you are importing, on the Selection Pane, highlight Column Options. On the Columns list, highlight the name of a column. In the Column Properties box, you can change the name, label, type, length, informat, format, and include in output (determines whether the variable will be included in the resulting SAS data set).

Once you have completed your import options, click on the Run button to complete the process. When it is complete, the data table is open in the Work Area. Icons representing the original Excel table, the Import Data Task, and the resulting SAS data set appear in the Process Flow and Project Explorer. The data table can be used in tasks and queries.

SAS ADD-IN FOR MICROSOFT OFFICE

SAS Add-In for Microsoft Office is part of SAS's comprehensive Enterprise Intelligence Platform. It allows business users to use SAS's analytic and reporting capabilities within the familiar environment of Microsoft Office.

SAS Add-In for Microsoft Office is installed as an integrated menu option within Microsoft Excel. It includes two SAS toolbars: SAS Data Analysis and SAS Analysis Tools. There is also a "SAS Favorites" menu so that the user can organize their common analytic and reporting tasks.

ACCESSING AND MANAGING DATA WITHIN SAS ADD-IN FOR MICROSOFT OFFICE

SAS Add-In for Microsoft Office can access your SAS data sources or a data source available on your server (including Oracle, Teradata, and DB2). It can access very large data sources, more than the traditional 65,536 row data size limit of Excel 2003, analyze the data, and return results back to Excel. For these very large data sources, the user can browse several thousand rows at a time, but since it performs its processing on the SAS Server, it bypasses the data access limitations of Microsoft Office for its analysis.

Users can refresh data automatically or run queries and apply filters to ensure that they are analyzing the most current information. Users can also switch dynamically between multiple data sources, so they can run multiple tasks with different data sources from within one worksheet or document.

Because the data SAS Add-In for Microsoft Office uses references the business metadata, analysts can access and utilize enterprise data from within the comfort of Office, while maintaining consistency, without a dependence on IT.

ANALYSIS AND REPORTING WITHIN SAS ADD-IN FOR MICROSOFT OFFICE

Users can write programs that execute SAS analytics or analysis, display dates, or produce charts. These "SAS Stored Processes" can be easily accessed by end users in Excel via a graphical user interface. SAS Add-In for Microsoft Office also has many dialogs that guide users through common SAS analytical tasks. Both Stored Processes and the dialogs produce output in Microsoft Excel. Tabular results can be delivered as raw data (CSV) or

HTML into Microsoft Excel, for further manipulation within Excel. Graphics results can be delivered directly into Excel, with many options to change the appearance and type of graph within Excel.

Results can be refreshed from within Microsoft Office documents with a click of the mouse. Users can refresh everything within their document or spreadsheet, which will rerun all the stored processes, update the data being browsed and replace the results using the newest data, or selectively refresh the results of one or more stored processes within the document.

CONCLUSION

SAS provides a variety of methods to both read and write Excel information from the SAS system for almost any batch or interactive application. The method you choose will depend on what products you have access to, characteristics of your source data, output requirements, data storage limitations, and programming skills. If your current method doesn't create your desired output or limits you in other ways, explore the various other methods available for transferring data and reports between SAS and Excel.

REFERENCES

"More Tips and Tricks for Creating Multi-Sheet Microsoft Excel Workbooks the Easy Way with SAS", DeIGobbo, SGF 2009

"De-Mystifying the SAS Libname Engine in Microsoft Excel: A Practical Guide", Choate and Martell, Sugi 31

ACKNOWLEDGMENTS

The authors would like to acknowledge Joy First for editing this paper. We would also like to acknowledge our clients who have exposed us to such a variety of source data and output requests, who have given us the ability to work with many of the different methods of working with SAS and Excel together.

RECOMMENDED READING

SAS 9.2 Documentation.

CONTACT INFORMATION

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

Name:	Steven First Jennifer First
Enterprise:	Systems Seminar Consultants
Address:	2997 Yarmouth Greenway Drive
City, State ZIP:	Madison, WI 53711
Work Phone:	608 278-9964
Fax:	608 278-0065
E-mail:	sfirst@sys-seminar.com jfirst@sys-seminar.com
Web:	www.sys-seminar.com

SAS and all other SAS Institute Inc. product or service names are registered trademarks or trademarks of SAS Institute Inc. in the USA and other countries. ® indicates USA registration.

Other brand and product names are trademarks of their respective companies.