

Exchanging data between SAS® and Microsoft Excel

Yuqing Xiao, Southern Company, Atlanta, GA

ABSTRACT

Transferring data between SAS® and Microsoft Excel has gained popularity over the years. Many methods were developed by different groups of people, from standard data step to most recent SAS ODS tagset. With many options available today, one may wonder 'Which would fit my situation the best?' This paper briefly discusses several options for importing and exporting data from/to Microsoft Excel using SAS and their pros and cons.

INTRODUCTION

Microsoft Office® is the most popular tool used in today's business. Clients often provide data in Excel workbooks and/or like to receive reports, analytical results in Excel format. Some can be simple one sheet data workbooks; some may require multiple sheets with many formatting and tables; some are simple creation of new workbooks; some may desire an update of an existing workbook rather than re-creation of the file. Many methodologies were developed and/or discovered by SAS user groups responding to those needs. Researching and learning them are time consuming. This paper collects a few common ways of importing/exporting data and includes some brief discussion on their usages. Note that all SAS code and discussions are based on SAS 9.1.3 although they are not limited to it.

IMPORT/EXPORT DATA FROM/TO EXCEL

Some of the common ways of Import/Export:

- Data Step
- SAS Import/Export Facility
- SAS/ACCESS® LIBNAME Statement
- SAS/ACCESS Pass-Through Facility
- SAS XML Engine
- Dynamic Data Exchange (DDE)
- ODS
- SAS Add-In to Microsoft Office

DATA STEP

If you have a simple one sheet Excel workbook, import using standard data step might be a good choice. First you need to save the Excel workbook in comma delimited CSV format. Then you can read it in with data step.

Code example to read .CSV file:

```
Filename myfile 'c:\mywork\testfile.csv';
Data sasdat;
  Infile myfile dlm=',' firstobs=2 missover;
  Input var1 var2 $ var3 $ var4;
Run;
```

This method is relatively simple. Almost all SAS users should know how to write a data step to read in text files, and you have the control over variable names, types and/or length. This method works with both Windows SAS and UNIX SAS (additional steps may be needed to FTP CSV file to your UNIX box). Some of the drawbacks are that it doesn't work with multiple sheets, you have to supply the variable names and/or informat specifications (although you can write more sophisticated code to automate the job), data get shifted when there are blank cells, plus you lose additional information that can be stored in XLS format when you convert it.

For export, using PUT statement within the data step can output CSV or XML file to be opened in Excel. It's easy to write CSV file, but limited to plain one sheet workbooks. XML file can have complex structures for more sophisticated workbooks with formulas, pivot tables etc, but writing XML file from scratch is not a daunting task to average SAS users. A trick for doing this is to create the workbook with all the desired features and save in XML format first, then open it with text editor and copy/paste the text into your SAS program, wrap each line in PUT '... '; manually deal with few embedded single quotes and place them in a data step. Custom tweak of the program may be necessary in some cases. The result can be opened in Excel just as if it was generated with Excel.

Code example for output .CSV file:

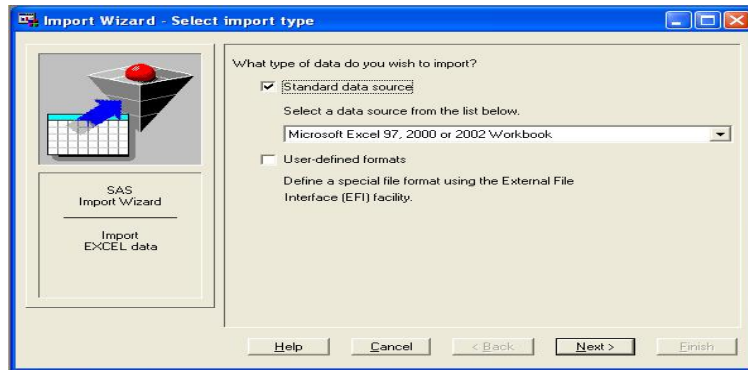
```
Filename myfile 'c:\mywork\testfile.csv';
Data _null_;
  set sasdat;
  file myfile delimiter=',' dsd;
  if _n_=1 then do;
    put 'name1' ',' 'name2' ',' 'name3';
  end;
  put var1 var2 $ var3 $ var4;
Run;
```

Data step is very flexible and can do complex jobs. To convert results from procedures, you need to output them to SAS data sets in order to use this method.

SAS IMPORT/EXPORT FACILITY

Another easy way to import/export XLS file is to use SAS Import/Export wizard. For import, go to 'File→ Import data...', choose Microsoft Excel appropriate version from drop down list, choose workbook and worksheet you'd like to read in, name the data set and the wizard will do the rest for you. Follow 'File→Export data ...' for export. You can also save a copy of Import/Export Wizard generated IMPORT/EXPORT procedure code for subsequent use or write your own. Import/Export wizard for XLS format only available for Windows and PROC IMPORT/EXPORT can deal with XLS file in both Windows and UNIX environment but the XLS file has to reside on PC.

SAS IMPORT Wizard:



Code example for PROC IMPORT (Windows):

```
Proc Import datafile= "c:\mywork\testfile.xls"
  out= work.sasdat dbms=excel replace;
  sheet="mydata$";
  getnames=yes;
  mixed=no;
  scantext=yes;
  usedate=yes;
  scantime=yes;
Run;
```

Code example for PROC IMPORT (UNIX):

```
Proc Import datafile= "c:\mywork\testfile.xls"
  out= work.sasdat dbms=excelcs replace;
  server=market;          /* Name of PC files server */
  port=1234;              /* Port number listening on the PC server */
  version='2002';
  sheet=mydata;
  scantext=yes;
  usedate=yes;
  scantime=yes;
  dbsaslabel=none;
  textsize=512;
Run;
```

Code example for PROC EXPORT (UNIX):

```
Proc export data=work.sasdat
  outfile="c:\mywork\testfile.xls" dbms=excelcs replace;
  sheet=mydata;
  version="2002";
  server=market;          /* Server name      */
  port=1234;              /* Port number    */
Run;
```

This is a pretty efficient method. No additional steps required; column names can be reserved as variable names or the reverse; multiple worksheets can be read in or written out in separate steps; blank cells are handled properly. The disadvantages are that SAS/ACCESS to PC files is required for importing/exporting XLS format, you have little control over variable names, types and/or length. There could be cases where column is recognized by the program as numeric and character values are set to missing or vice versa. Plus, the entire worksheet has to be imported, you can't choose columns and/or ranges.

The PROC IMPORT/EXPORT actually imports/exports XLS files through generated SAS/ACCESS code. A more customized approach is to use PROC ACCESS. It's more flexible, can define columns and ranges to be read in, can change variable names, types, formats and more. However, it's only available under Windows and might be a little bit beyond beginner SAS user's scope.

Code example for PROC ACCESS:

```
Proc Access dbms=xls;
  /* create access descriptor */
  create work.dat_view.access;
  path="c:\mywork\testfile.xls"
  worksheet=mydata;
  range='a1..j39';
  getname=yes;
  scantype=5;
  mixed=yes;
  assign=no;
  rename var1 = name1
         var2 = name2;
  format 1      5.0
         2      $4;
  /* create SAS view */
  create work.dat_view.view;
  select var1 var2 var3;
Run;
/* create SAS data set */
Data sasdat;
  set work.dat_view;
Run;
```

Another very similar way of doing this is through the use of .DBF format. It works pretty much like importing/exporting XLS file except that file can be saved under both Windows and UNIX environment.

Code example to read/write .DBF file:

```
Proc Import datafile = "c:\mywork\testfile.dbf"
  out= work.sasdat dbms=dbf replace;
  getdeleted=no;
Run;
Proc export data= work.sasdat
  outfile= "c:\mywork\testfile.dbf" dbms=dbf replace;
Run;
```

A few drawbacks are that separate DBF files may be needed for multi-sheet workbooks, one needs to pay extra attention to hidden columns and/or data filters in Excel because data are lost when saved as DBF file. Also, all the formatting are lost after conversion.

SAS/ACCESS LIBNAME STATEMENT

SAS/ACCESS provides other means of accessing/creating Excel files using then LIBNAME statement. Simply assign a library reference to an Excel workbook and directly read from and/or write to it. For UNIX SAS, use 'pcfiles' option with server name and port name to connect to PC files. Excel workbook can be either a new workbook or an existing one. This method works with multiple worksheets and is available anywhere a library reference is valid, such as data step, SAS SQL and SAS Procedures.

Code example for Import/Export Excel files (Windows):

```
Libname myxls 'c:\mywork\testfile.xls';
/* import data */
Data sasdat;
    set myxls.mydata;
Run;
/* export Excel workbook */
Data myxls.mydata2;
    set sasdat;
Run;
Proc sql;
    create table myxls.summary as select var1, var2, var3
    from sasdat group by var4;
quit;
Libname myxls clear;
```

Code example for Import Excel files (UNIX):

```
Libname myxls pcfiles server=market port=1234 path='c:\mywork\testfile.xls';
Data sasdat;
    set myxls.'mydata$'n;
Run;
Libname myxls clear;
```

SAS/ACCESS PASS-THROUGH FACILITY

An alternative to SAS/ACCESS LIBNAME statement is the Pass-Through facility. It uses SAS/ACCESS to connect to a data source and sends data source specific SQL statements directly to the data source. It enables you to retrieve data or a subset of the data from Excel workbook worksheets and/or named ranges. It's also available in both Windows and UNIX environment.

Code example for SAS/ACCESS Pass-Through (Windows):

```
Proc sql;
    connect to excel (path='c:\mywork\testfile.xls');
    create table sasdat as select * from connection to excel
    (select var1, var2, var3 from mydata);
    disconnect from excel;
Quit;
```

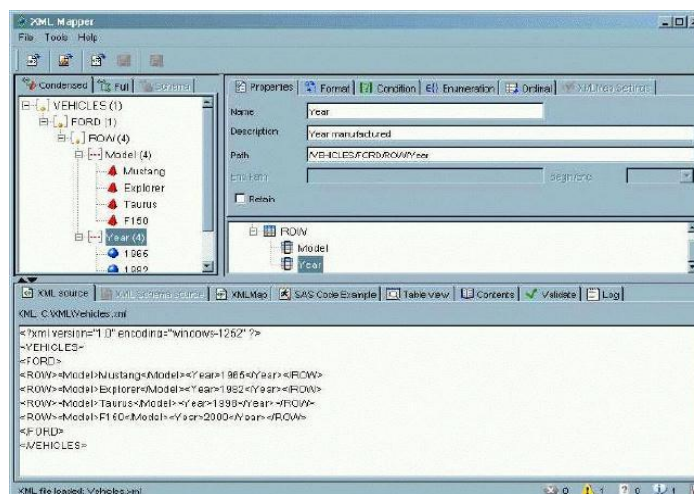
Code example for SAS/ACCESS Pass-Through (UNIX):

```
Proc sql;
    connect to pcfiles (path='c:\mywork\testfile.xls' server=market port=1234);
    create table sasdat as select * from connection to pcfiles
    (select * from mydata);
    disconnect from pcfiles;
Quit;
```

SAS XML ENGINE

Excel can open and save XML documents. SAS XML engine translates data between XML document and SAS data sets. It can read and write XML documents with SAS data step or Procedures. However, without further instructions XML engine reads only files that conform to certain structures. Otherwise, you can create an XMLMap which tells the XML engine how to interpret the XML markup into SAS data sets. A supporting tool called XML Mapper® is available from SAS. It has a graphical interface with three primary panes. Open and display the XML document in its XML primary pane and create XML MAP by drag-and-drop elements from XML primary pane into XMLMAP primary pane, then specify the XML MAP in the LIBNAME statement to translate the XML document.

SAS XML Mapper:



Code example for SAS XML engine:

```
/* assign libref to XML document location and specify XML engine */
Filename mymap 'c:\mywork\myfile.map';
Libname mypath xml 'c:\mywork\myfile.xml' xmlmap=mymap;
/* read in XML document with data step */
data sasdat;
    set mypath.mydata;
run;
/* read in XML document with procedures */
proc copy in=mypath out=sasdat;
    select grades;
run;
/* create XML document with data step */
Data mypath.myfile;
    set sasdat;
Run;
```

XML has many benefits and is becoming very popular in storing data nowadays. Its software- and hardware-independent characteristic makes it very useful in sharing data between different parties and relatively immune to changes in technology. Even though, there are certain disadvantages associated with this method. First, in Excel, saving spreadsheets as XML Data requires user supplied XML schema. Although you can avoid this by choosing 'save as XML spreadsheet', the resulting file structure can be complicate and I haven't had any success building XML Map with this format option. Second, XML engine uses more processing time than other strategies. And last, XML documents can be large. It's not recommended when space or network bandwidth is an issue.

DYNAMIC DATA EXCHANGE (DDE)

DDE is used to dynamically exchange information between Windows applications. Transferring data between Excel spreadsheets and SAS is among DDE's many potential uses. In order to read or write the data, Microsoft Excel workbook must be opened first. It can be done either manually or by SAS code. Then a DDE link is established through FILENAME statement. Text string known as DDE Triplet must be specified and enclosed in quotation marks. Triplet consists of three parts in the form 'application-name|topic ! item', where application is Excel in this case, topic is typically the full path filename which you want to share data and item is the range. An easy trick to determine the triplet is to copy the desired range of cells to the clipboard and then in PC SAS, go to 'Solutions->Accessories->DDE triplet'. The exact triplet will appear in the dialog box. Data can then be read or written with data step. And last, you may want to close the workbook and Excel application.

Code example to read .XLS file through DDE:

```
/* invoke Excel and open workbook */
Options noxwait noxsync;
x '"c:\program files\microsoft office\office11\excel.exe"';
Data _null_;
    rc = sleep(5);
Run;
```

```

Filename ddecmd dde 'excel|system';
Data _null_;
    file ddecmd;
    put '[FILE-OPEN("c:\mywork\testfile.xls")]';
Run;
/* read desired rows and columns from Excel file into SAS */
Filename myfile dde 'excel|c:\mywork\[testfile.xls]mydata!r2c1:r10c3';
Data sasdat;
    infile myfile notab dlm='09'x dsd missover;
    informat var1 5. var2 $4. var3 $20.;
    input var1 var2 $ var3 $;
Run;
/* close workbook and quit Excel */
Data _null_;
    file xlin;
    put '[FILE-CLOSE("c:\mywork\testfile.xls ")]';
    put '[QUIT()]';
Run;

```

Code example to write .XLS file through DDE:

```

/* invoke Excel and open new workbook */
Options noxwait noxsync;
x "c:\program files\microsoft office\office11\excel.exe";
Data _null_;
    rc = sleep(5);
Run;
/* save blank spreadsheet to desired location */
Filename ddecmd dde 'excel|system';
Data _null_;
    file ddecmd;
    put '[SAVE.AS("c:\mywork\testfile.xls")]';
Run;
/* define fileref using DDE access method */
Filename myfile dde 'excel|c:\mywork\[testfile.xls]sheet1!r1c1:r&rows.c3' notab;
/* write to Excel workbook */
Data _null_;
    file myfile;
    set work.sasdat;
    if _n_=1 then put 'name1' '09'x 'name2' '09'x 'name3';
    put var1 '09'x var2 '09'x var3;
Run;
/* save workbook and quit Excel */
Filename ddecmd dde 'excel|system';
Data _null_;
    file ddecmd;
    put '[SAVE()]';
    put '[QUIT()]';
Run;

```

DDE gives you the flexibility of specifying the desired ranges for read/ write and control over variable types and length. In addition, you can issue Excel commands to customize the spreadsheet, like set format, font, color, header/footer or run macros, etc. You may consider this method when creating highly customized spreadsheets or updating existing spreadsheets. It also works with older versions of SAS. The shortcomings are that it's not available for UNIX and Excel must be running for it to work. Plus, you have to know Excel commands to do formatting.

ODS

ODS stands for Output Delivery System. It's a very powerful tool which can generate almost limitless types of output with highly customized formatting. ODS organizes output from data steps or procedures into a series of objects and sends all or part of them to user specified destinations. Choice of ODS destinations determines the type of output to be generated, such as HTML, XML, PDF etc. By default, LISTING destination is open and all others are closed. To create multi-sheet Excel workbooks, specify tagsets.ExcelXP ODS destination and wrap the procedure and/or data

step between ODS open and close statements. Appearance of the output is controlled by ODS styles. You can create your own style definition or modify an existing one through PROC TEMPLATE. With custom style definitions, SAS ODS can generate very sophisticated Excel workbook with formulas, highlighting, wrapped text and much more.

Sample code for creating multi-sheet spreadsheet with ODS:

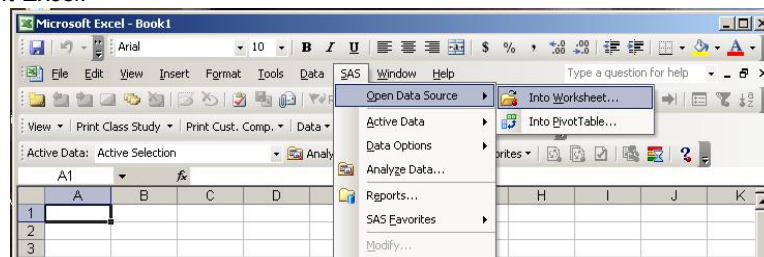
```
/* close ods listing destination */
Ods listing close;
/* open tagsets destination and send output to xml file */
Ods tagsets.ExcelXP path='c:\mywork' file='testfile.xml' style=mystyle;
Ods tagsets.ExcelXP options(sheet_name='mydata');
Data _null_;
  set sasdat;
  file print ods=(variables=(var1 var2 var3));
  put _ods_;
Run;
Ods tagsets.ExcelXP options(sheet_name='summary');
Ods Exclude Moments TestForLocation;
Proc univariate data=sasdat;
  By var2;
  var var1;
Run;
/* close tagsets destination and re-open listing destination */
Ods tagsets.ExcelXP close;
Ods listing;
```

ODS is an excellent way of exporting SAS datasets to Excel workbooks and can make fancy looking Excel workbooks without opening and running Excel. It can also output CSV or HTML files to be opened in Excel. One possible drawback is that exploring many style options for customized formatting can be time consuming and may not be your best choice when moving data is the central focus.

SAS ADD-IN TO MICROSOFT OFFICE

SAS Add-In for Microsoft Office is my new favorite method of sharing data between SAS and Microsoft Office products. It's provided with SAS BI server or SAS Enterprise BI server and can be distributed among the entire organization. After installing it, you can open SAS dataset in Excel, view and/or edit data, analyze the data and do reporting. Data in Excel spreadsheet can be copied back to a SAS server and saved as SAS data set. To open a SAS data set, find SAS menu in the toolbar, select 'SAS→Open Data Source →Into Worksheet', choose your SAS server and SAS data set and click open. Then you will be prompted to select variables to be brought into the spreadsheet. To copy a workbook to a SAS data set, select the desired range of data, go to 'SAS→Active Data→Copy to SAS Server', use browser to find the location where you'd like to save the data, name the data set then save.

SAS Menu in Microsoft Excel:



There are a few things you need to be aware of. First, SAS Add-In for Microsoft Office must connect to an environment running SAS BI Server or SAS Enterprise BI Server. Second, the data folder must be registered with SAS BI Server to be seen in browser windows. And note you may want to change the default number of displayed records at 'Options' window under 'SAS' menu item.

CONCLUSION

So far, we explored several common ways of exchanging data between two applications. They all have some pros and cons and may be suitable for different situations. The table below summarizes some of their characteristics and may help your method selection. Yet, even more methods exist. They can be custom-made macros like %sas2xls,

%sas2csv.sas and %xlxp2sas, data conversion software, or solutions with other languages like XML, VSTO and VBA etc.

Summary of Data Exchanging Methods:

	Available with PC SAS	Available with UNIX SAS	Import capable	Export capable	Work with Multi-sheet workbook	Excel Formatting available	Work with procedures	Other SAS packages required
Data Step	●	●	●	●	●	●		
Import/Export Wizard	●		●	●	●			SAS/ACCESS
Import/Export procedure	●	●	●	●	●			SAS/ACCESS
Proc Access	●		●	●	●			SAS/ACCESS
SAS/ACCESS LIBNAME	●	●	●	●	●		●	SAS/ACCESS
SAS/ACCESS Pass-Through	●	●	●		●			SAS/ACCESS
SAS XML Engine	●	●	●	●	●		●	
DDE	●		●	●	●	●		
ODS	●	●		●	●	●	●	
SAS Add-In	●		●	●	●			SAS BI

REFERENCES

1. SAS OnlineDOC, SAS Institute Inc.
<<http://support.sas.com/91doc/docMainpage.jsp>>
2. "Importing Excel files to SAS Datasets."
<http://archive.chesapeakebay.net/cims/importing_excellb.pdf>
3. Curtis A. Smith. "Importing Excel Files Into SAS Using DDE"
<http://www.lexjansen.com/wuss/2007/ApplicationsDevelopment/APP_SmithC_ImportingExcelFiles.pdf>
4. Lawrence Helbers, Alex Vinokurov. "SAS® Output to Excel: DDE and Beyond." NESUG 2002 Conference.
5. Ralph Winters. "Excellent Ways of Exporting SAS® Data to Excel." NESUG 2004 Conference.
6. Vincent DelGobbo. "Creating AND Importing Multi-Sheet Excel Workbooks the Easy Way with SAS®." SUGI 2006 Conference, paper 115-31.
7. Mark Terjeson. "How to create an excel pivot table using SAS" Online posting. 7 Apr. 2005. SAS-L
<<http://www.listserv.uga.edu/cgi-bin/wa?A2=ind0504A&L=sas-l&P=R30054>>
8. Alan Churchill. "Convert SAS datasets to Excel" Online posting. 23 Jan. 2006.
<<http://savian.blogspot.com/search?q=sas+to+excel>>

ACKNOWLEDGMENTS

I would like to thank my colleagues Bob Bolen, Chaoying Hsieh and Dian Cunningham for their thorough review and valuable comments.

CONTACT INFORMATION

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

Yuqing Xiao
Southern Company Marketing Services
Bin 10206
241 Ralph McGill Blvd. NE
Atlanta, GA 30308
Work Phone: (404) 506-4619
E-mail: yxiao@southernco.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.