Paper 074-2011

Creating an Excel report: A comparison of the different techniques

Romain Miralles, Clinovo, Sunnyvale, CA

ABSTRACT

Many techniques exist to create an Excel file using SAS®. Each of them offers some unique advantages.

LIBNAME statement is a new and useful option available with SAS 9 but old methods like DDE are still very powerful and offer some unique capabilities. Knowing and understanding the different techniques is essential for SAS programmers to quickly and effectively produce a report that will meet the requirement provided by the customer.

This paper will list and describe the different techniques to generate an Excel file from SAS, and provide recommendations on the appropriate method to use when an Excel output must be created.

INTRODUCTION

Programming an Excel report is a common task for most SAS programmers. Customers often want to be able to view, sort, and filter their data, and Excel is usually the tool they master best.

SAS offers a wide range of techniques to create an Excel report, each with its own advantages and limitations. Before starting a new program, the SAS programmer should spend time to determine the most appropriate technique to use, taking into consideration the customer's needs, constraints, and time frame.

In this paper, I will first discuss several techniques used to generate an Excel report using SAS, and highlight some of their advantages and limitations. I will then present a few questions that SAS programmers can ask themselves to determine the best method to use.

PROC EXPORT

DESCRIPTION

PROC EXPORT is the most common way to export a SAS data set to a Microsoft Excel document. Its simple syntax makes it easy to understand. Usually, it is the first method that a SAS programmer will learn to use in order to export data. It is a basic technique that proves useful in many situations.

PROC EXPORT can be used in two different ways:

- First, we can use it to create an Excel file with the data from the data set. With this method, it is not possible to format the output.
- The second solution is to use an Excel template and to populate the file with data from the data set. An Excel document is saved on the computer and the columns are formatted as needed. Then the path and filename of the Excel template are entered in the PROC EXPORT OUTFILE parameter.

SYNTAX

```
PROC EXPORT DATA = sasglf
      DBMS=excel
      OUTFILE = "c:\sas_globalforum.xls"
      REPLACE;
      sheet='page1';
RUN;
```

ADVANTAGES

PROC EXPORT is a good method for easy data manipulation. The Excel file can have one or multiple spreadsheets. It is a good technique if the customer expects a simple listing developed quickly and does not require complex formatting.
The export wizard in SAS is an interface to the EXPORT procedure. We can create the same report with the PROC EXPORT. The advantage is that we can incorporate the SAS code in the program and we do not need to step through all the export wizard windows every time we want to create a file.

LIMITATIONS

Customization of a report appearance is limited even when using an Excel template. PROC EXPORT starts in the default, top, left cell (A1) and fills out the necessary rows and columns. Moreover, PROC EXPORT cannot use the SAS labels as column names in Excel unless you are using SAS 9.2. This limits the use of PROC EXPORT for report programming. Typically, users have no knowledge of the SAS data set structure and SAS variable names are not meaningful to them. In addition, variable names in SAS have restrictions that column labels don't share.

The order of columns in the exported worksheet is the same as the order of variables in the original data set. Variables need to be ordered before using the PROC EXPORT. All the existing variables in the data set will be output using this technique. This method also requires that your SAS installation includes the SAS/ACCESS Interface to PC Files license.

LIBNAME ENGINE

LIBNAME engine is one of the newest methods to transfer information from SAS into Excel. It is available with SAS version 9 and lets you use Excel as a SAS library.
LIBNAME engine allows advanced customization of your output. It does not give full control of Excel like DDE, but it has one major advantage: Excel does not need to be installed on the machine running SAS.

SYNTAX

There are two ways to use LIBNAME for generating an Excel file. It can create a new workbook from scratch or it can write an Excel template.

A. LIBNAME engine without a template

The Excel file should not exist on the drive as the LIBNAME engine cannot overwrite.

```
LIBNAME toexcel EXCEL 'c:\sugi_libname.xls' VER=2002 ;
DATA toexcel.sugi(DBLABEL=YES);
      SET sugi.sugi ;
RUN;
LIBNAME toexcel CLEAR;
```

Ver=2002: To specify the most current version. The default is Excel 97
DBLABEL=YES: Write the variable labels

A data step is used to export the data. We can choose the name of the tab in the Excel output and we can export the variable labels with the option DBLABEL=YES.

B. LIBNAME engine with a template

LIBNAME Engine can be used with an Excel template to create a customized report. The data are output in the named ranges which are a subset of cells defined using Excel option insert->name->define. Multiple named ranges can be defined and used to control the display of data in the spreadsheet.

```
/* The first step is to delete the data from the range to be populated. Without prior
deletion, SAS will not populate the range*/
PROC DATASETS LIB=toexcel;
      DELETE sugi; /* Sugi is the named range defined in excel*/
      RUN;
QUIT;

PROC DATASETS LIB=toexcel;
      DELETE param;   /* 2nd named range defined in excel*/
      RUN;
QUIT;
```

```
DATA toexcel.sugi; /* 2nd output the data in the 'sugi' range defined in Excel*/
      SET sugi ;
RUN;

DATA toexcel.param ;
      SET param ;
RUN;
LIBNAME toexcel CLEAR; /* Disconnect from workbook*/
```

ADVANTAGES

LIBNAME engine offers the advantages of exporting data to Excel in a quick and easy manner, as well as developing elaborate reports. It does not need Excel to be present or activated, and a simple data step can be used to populate the output. The Excel spreadsheet can be manipulated much like a SAS data set and unlike PROC EXPORT (before SAS 9.2), LIBNAME engine can export the variable labels.

LIBNAME engine is a possible solution to program a fancy report. Pre-defined and customized Excel worksheets can be easily used after understanding named ranges in Excel.

LIMITATIONS

Using LIBNAME can at first be confusing. Each data set appears twice when you open the Excel file with LIBNAME:
- One with the expected name. This data set is the named range.
- One with a trailing "$". This data set is the spreadsheet.

It requires some time to understand Excel structure and the use of named ranges. Without a correctly named range in Excel, LIBNAME cannot execute and display an error message in the log.

By default, LIBNAME engine writes the variable names in the first row of a range. When defining the named ranges, users need to remember that the first row will be populated with the variable names. The only solution not to display the headers is to hide the first row of the range in the Excel template. This method also requires that your SAS installation includes the SAS/ACCESS Interface to PC Files license.

EXCELXP TAGSET

DESCRIPTION

ExcelXP tagset is an ODS destination available in SAS version 9.1 that utilizes the Extensible Markup Language (XML). It can be downloaded from the SAS website. Using the ExcelXP Tagset is a powerful method to control formatting of a spreadsheet. Common ODS options can be used as well as many other helpful options specific to the tagset.

ExcelXP tagset can be used to export the results of PROC REPORT, PROC TABULATE, or PROC PRINT. It can display multiple tables per worksheet as well as multiple worksheets.
As any other ODS destination, the option 'style' can be used with the ExcelXP tagset. Many styles are available in SAS and styles can also be customized or created using PROC TEMPLATE.

SYNTAX

```
ODS listing;
ODS results;
ODS listing close; /*Turn off the standard line printer destination*/
ODS noresults;      /*Prevents results from appearing within SAS viewer*/

ODS tagset.ExcelXP
      FILE="c:\sas_globalforum.xls"
      STYLE=sasgl  /*Style to control appearance of output*/


      options     (Embedded_titles = 'yes'  /*ExcelXP options*/
                   Embedded_Footnotes   = 'yes'
                   sheet_name= 'Sasgl'
                   autofilter= 'yes'
                   frozen_headers= '3'
                   autofit_height= 'yes'
                   absolute_column_width= '15,10,10,13');
```

```
TITLE1 "List of SAS conferences";

FOOTNOTE1 "April 4th 2011";

PROC REPORT DATA=sglf NOWD;
     COLUMN conference city state attendees;
     DEFINE conference /CENTER  style(column)=[font_weight=bold] style(header) =
     [background = CX4D7EBF];
     DEFINE city /CENTER    style(header) = [background = CX4D7EBF];
     DEFINE state /CENTER    style(header) = [background = CX4D7EBF];
     DEFINE attendes /CENTER    style(header) = [background = CX4D7EBF];
RUN;


ODS tagset.ExcelXP CLOSE; /* Close and release the xml file so it can be opened with
Excel*/

ODS listing;
ODS results;
```

In this example, we used only a few options but many more are available and documented online.

ADVANTAGES

ExcelXP tagset has an amazing number of options and high flexibility, allowing it to accomplish almost any report. It reduces or eliminates the need for manual formatting, as all formatting and layouts are performed by SAS. There is no need to create a template or edit the Excel workbook. ExcelXP includes a lot of options that control the appearance of the report, and many papers are available online to get a better understanding of all them.

ExcelXP tagset is one of the best ways to create a file with multiple sheets. Unlike other techniques, there is no need to prepare a template or to use a SAS macro. All formatting is performed by SAS, so it becomes very easy to define a style and apply it to all the worksheets.

LIMITATIONS

In order to take advantage of the latest ExcelXP features, the tagset must be downloaded and installed. Computers without the latest version of ExcelXP might not be able to run SAS programs to create reports. Additionally, ExcelXP is still evolving, thus functionalities may change in the future.

Excel does not handle date variables like SAS. Date variables exported from SAS with ExcelXP tagset are interpreted like a text variable. Excel will understand SAS dates only if they are converted to a specific format before using ExcelXP (the paper “The Devil Is in the Details: Styles, Tips, and Tricks That Make Your Microsoft Excel Output Look Great!” gives a great explanation about date format with ExcelXP).

ExcelXP tagset produces an XML file designed for Excel. XML is a great way to store data; however, it is not readable on every computer. Some customers with an older version of Excel might not be able to open the file created by ExcelXP tagset. One of the solutions to avoid this issue is to open the XML file and save it in XLS format. This can be automated using a SAS macro and a VBS script.

ODS CSV

ODS CSV is an option that can be used to create an Excel file. It was experimental in SAS 8 and moved to production in SAS 9. ODS CSV creates a comma-separated value (CSV) text file that can be read and displayed in Excel. The values are enclosed in double quotation marks in the CSV file. Thanks to these quotation marks, commas are allowed as part of the values.

SYNTAX

```
ODS CSV FILE='C:\sasglobal.csv';
   PROC PRINT DATA=sasglobalforum;
   RUN;
```

```
ODS CSV CLOSE;
```

ADVANTAGES

ODS CSV does not actually produce an Excel file. The output is a CSV document that can be imported in Excel. The main advantage of this format comes from its age and its widespread compatibility. CSV files can be opened in any Excel version and almost any spreadsheet and database management system. Additionally, the CSV file size is quite small since the CSV destination does not have any formatting.
These two advantages make ODS CSV a good technique when we need to create a small file that will be read on different versions of Excel.

LIMITATIONS

ODS CSV does not offer control over output appearance, and it creates a simple Excel-readable text file with no formatting. It generates output beginning on row 3 by default. To remove the first two rows, we need to modify the tagset and override the defaults. It offers less control over the titles, footnotes and other styles compared to the other techniques discussed in this paper.

ODS HTML

The ODS HTML BODY is used to create HTML documents. However, it can also be used to create an Excel report by assigning an XLS extension instead of HTML.

SYNTAX

```
ODS HTML BODY = 'C:\sugi.xls';
      PROC FREQ DATA=sugi;
      TABLES var1 var2;
      RUN;
ODS HTML CLOSE;
```

ADVANTAGES

ODS HTML is a destination commonly used for exporting output to Microsoft Excel. It provides an easy way to create a report that can be customized using the predefined styles in SAS.
Not only can we use predefined SAS styles, but we can also define our own style using PROC TEMPLATE or applying cascading style sheets (CSS) to format ODS output. The appearance of the output, color, fonts, borders, sizes as well as many others parameters can thus be defined.

LIMITATIONS

ODS HTML creates a file that Excel opens as a spreadsheet. Although the file extension is XLS, it still remains an HTML document with many tags controlling the formatting. The document created by the markup can become large if the data exported includes many variables and records. It is possible to reduce the size of the file by using the style MINIMAL provided by default with SAS. Other techniques also exist to reduce the size of Excel files.

ODS HTML is still widely used but it has been replaced by the tagset MSoffice2K in SAS 9 which is a more powerful tagset for HTML output that can be read with Excel. Outputs created with ODS HTML are more suitable for the browser rather than Excel.

ODS MSOFFICE2K

The ODS MSOffice2K is an improved version of the ODS HTML available with SAS version 9. It is used in the same way as ODS HTML and produces an HTML file that can be opened by Excel. The main difference is that this tagset was developed especially for creating HTML output compatible with the Office applications.

SYNTAX

```
ODS tagsets.msoffice2k FILE= 'C:\sugi.xls';
   PROC PRINT DATA=sugi;
   RUN;
ODS tagsets.msoffice2k CLOSE;
```

ADVANTAGES

ODS MsOffice2K is a quick and easy method that offers the same benefits as ODS HTML and a better support of embedded images and graphics. It can be used to generate presentation-quality output or simple reports with limited formatting.

ODS MsOffice2K is geared to work specifically with the Office applications that can read HTML. It is recommended to use this tagset rather than the often-used HTML destinations for generating Excel output.

LIMITATIONS

The output created by the tagset is an HTML document, and as ODS HTML, the file can become quite large if many data are exported to Excel.

This ODS destination does not provide the many options available with the ExcelXP destination and it cannot create multi-sheet reports. However, there is a modified version of this tagset available online (MSOffice2K_x) that adds options to perform many of the common tasks in Excel, including one that generates multi-sheet files.

DDE

DESCRIPTION

Dynamic Data Exchange (DDE) is an old protocol, but it is one of the most powerful methods to integrate SAS and Excel.
DDE is the direct communication between SAS and Excel using a server/client model. Excel acts as a server and SAS as a client. It is the only technique that can use visual basic language, the most powerful feature of Excel, and provides total control over the output.

This technique might seem obscure to people who have no experience with it. There are two ways to use DDE:

- The first approach, and probably the most difficult one, is to execute all the code directly in SAS using X4ML functions to provide instructions to the Excel application. DDE can enable much of the functionality of Excel within SAS.
- The second solution is to create a pre-formatted Excel template and populate it using DDE. With just two macros to open and close the Excel file, it becomes very easy to program a fancy report. This solution is easier and faster. Anyone can prepare a nice spreadsheet using the power of Excel, and populate it with the SAS data.

SYNTAX

```
%LET stufile=C:\sugi;
/*****************************************************************************/
/*                                TO OPEN XLS FILE                           */
/*****************************************************************************/

%MACRO OPENXLS(FOLDER=,IN=);
%LET FIL=;
       DATA _null_;
              LENGTH FILE $300.;
              FILE="'&STUFILE\&folder.&in.'";
              FILE="'"!!TRIM(LEFT(TRANWRD(FILE,"'",'"')))!!"'";
              CALL SYMPUT ("FIL",TRIM(LEFT(FILE)));
       RUN;
options noxwait noxsync;
x &fil.;
filename commands dde "Excel|system";
%MEND OPENXLS;
/*****************************************************************************/
/*     TO SAVE ACTIVE XLS FILE    UNDER SPECIFIC FOLDERS AND QUIT EXCEL          */
/*****************************************************************************/
%MACRO CLOSEXLS(out=,quit=0);
       %LET FIL=;
```

```
        DATA _null_;
              LENGTH FILE $300.;
              FILE="[save.as('&STUFILE.\&out.')]";
              FILE="'"!!TRIM(LEFT(TRANWRD(FILE,"'",'"')))!!"'";
              CALL SYMPUT ("FIL",TRIM(LEFT(FILE)));
        RUN;
        %PUT &FIL=;

        DATA _null_;
              file commands;
              put &fil.;
              put '[CLOSE()]';
              %IF &quit=0 %THEN %DO;
                    put '[QUIT()]';
                    stop;
                    RUN;
                    filename commands clear;
              %END;

              %IF &quit=1 %THEN %DO;
                    RUN;
              %END;

%MEND CLOSEXLS;
%OPENXLS(FOLDER=template\,IN=sugi_2011_dde.xls); /* Open the Excel template*/

/* We use the triplet 'template file name/tab name/range' to define where to output
the data. The template file name is between bracket followed by the tab name and the
range in the format RxCx:RxCx (R for row and C for column). Note that the range need
to be in the language of the excel on your computer*/ FILENAME TAB DDE
"EXCEL|[sugi_2011_dde.xls]sugi!R5C3:R30C10";
FILENAME xlcmds DDE "EXCEL|SYSTEM";


DATA _null_;  /* Data step to output the data in the Excel file */
   SET sugi;
   FILE TAB NOTAB LRECL=7000;

   PUT  conference         '09'x
   city                    '09'x
   state                   '09'x
   attendees               '09'x;
RUN;

/* We can output as many data set as we want. In this example we output a second data
set to another location*/
FILENAME TAB DDE "EXCEL|[ sugi_2011_dde.xls]sugi!R16C1:R18C2";
FILENAME xlcmds DDE "EXCEL|SYSTEM";

DATA _null_;
   SET param;
   FILE TAB NOTAB LRECL=7000;
   PUT  var1               '09'x
        var2               '09'x;
RUN;

DATA _null_; /* Invoke a VB macro 'lastline' embedded in the Excel template */
      FILE xlcmds;
      PUT '[RUN("lastline")]';
RUN;
%CLOSEXLS(OUT=sugi_DDE_&today..xls); Specify the name of the output and close the
file*/
```

ADVANTAGES

DDE is a great technique to create sophisticated reports. It provides full control of Excel and allows you to leverage the powerful capabilities of this reporting tool. Anyone who knows Excel can prepare an Excel template with pre-formatted cells and columns.

Using DDE, you can call VBA macros embedded in your Excel template. VBA macros are very powerful and enable you to extend Excel in any way you choose. If you need to use an Excel macro for your report, DDE is the best solution.

LIMITATIONS

Both Excel and SAS need to be installed and active for DDE to work. This makes DDE impossible to use in an environment where SAS is installed on a server and Excel on a desktop.
In addition, knowledge of Excel programming is required. DDE uses Excel version 4 macro language, which has been superseded by Visual Basic for Applications (VBA). Finding documentation for this macro language is not always easy.

INTEGRATED OBJECT MODEL

DESCRIPTION

Excel can communicate with SAS using the Integrated Object Model (IOM). SAS integration technologies were introduced in SAS v8, and improved in SAS v9. Excel establishes a connection with a SAS session that can be used to run SAS code. It provides access to SAS data and procedures from Excel and is designed to work in a client-server environment.

SYNTAX

Before using IOM, Excel needs to be configured properly. The following references need to be selected in the visual basic editor using the menu tools->references.

- SASObjectManager 1.1 Type Library
- SAS: Integrated Object Model (IOM)
- Microsoft ActiveX Data Objects 2.8 Library

Below is the code for an Excel VB macro that will run the SAS program sasglforum.sas located in C:\mysaspg. The program creates the data set ‘sasgldata’ in C:\temp. Then the VB macro exports the data set into Excel.

```
Sub iomtest()
Dim swsSAS  As SAS.Workspace
Dim rsSAS   As New ADODB.Recordset
Dim obObjectFactory As New SASObjectManager.ObjectFactory
Dim obServer As New SASObjectManager.ServerDef
Dim cnnIOM  As New ADODB.Connection
Dim xmlInfo As String
Dim fld     As Field
Dim row     As Long
Dim col     As Long


'For this example we use localhost server

obServer.MachineDNSName = "localhost"

'We Create a local SAS workspace
Set swsSAS = obObjectFactory.CreateObjectByServer("Workspace", True,
obServer, "myUserName", "myPassword")

'Run the SAS program at c:\mysaspg\sasglforum.sas
Dim obStoredProcessService As SAS.StoredProcessService
```

```
Set obStoredProcessService = swsSAS.LanguageService.StoredProcessService
obStoredProcessService.Repository = "file:C:\mysaspg"
obStoredProcessService.Execute "sasglforum", Parameters

‘Export the SAS data set to Excel
cnnIOM.Provider = "sas.LocalProvider.1"
cnnIOM.Properties("Data Source") = "c:\temp"
cnnIOM.Open
rsSAS.Open "sasgldata", cnnIOM, adOpenDynamic, adLockPessimistic,
ADODB.adCmdTableDirect
'Select the sheet sasglf and freeze the panes on the 2ndline
   Worksheets("sasglf").Activate
   Range("A2").Select
   ActiveWindow.FreezePanes = True
   If Not (rsSAS.BOF And rsSAS.EOF) Then
      Worksheets("sasglf").Activate
      Range("A1").Select
      rsSAS.MoveFirst
      row = 2
      Do While Not rsSAS.EOF
         ActiveSheet.Cells(row, 1).Select
         For Each fld In rsSAS.Fields
            ActiveCell.Value = fld.Value
            ActiveCell.Next.Select
         Next
         row = row + 1
         rsSAS.MoveNext
      Loop
   Range("A2").Select
   End If
   rsSAS.Close
   Set rsSAS = Nothing
   cnnIOM.Close
   Set cnnIOM = Nothing
   swsSAS.Close
   Set swsSAS = Nothing
   Set swmWM = Nothing
End Sub
```

ADVANTAGES

Integration technologies allow SAS programmers to separate the interface from the engine. The front end can be developed using Excel with all its benefits (VBA, pivot table, etc.), while the back end is developed in SAS. The advantage is that SAS does not need to be installed on the computer on which the application runs.
Additionally, the SAS code is executed directly from the Excel environment. It means that users can refresh the report by running an Excel macro. This macro will execute a SAS program and update the front end.

LIMITATIONS

In order to use this method, SAS programmers need to understand SAS integrated object model and to have a good knowledge of VBA as most of the code is written directly in Excel. Unfortunately, not everyone has the skills to use this method. It requires a lot of time to understand integration technologies before being able to use IOM.

HOW TO CHOOSE THE BEST METHOD

Each method presented in this paper has some advantages and limitations when it comes to creating an Excel report. Choosing the best method to create a specific document is one of the responsibilities of a SAS programmer. As a programmer, we need to take the time to identify the solution that will best fit our needs. There are a few questions that we can ask ourselves in order to determine which technique to use.

WHAT IS THE REPORT FOR?

The first step is to review and understand the requirements. The goal of the project can be to simply export a SAS data set to Excel or to develop a very advanced report like a dashboard using SAS data. For exporting a data set to Excel, a PROC EXPORT, ODS CSV or LIBNAME engine would be enough in most cases. However, if the objective of the project is to develop a very advanced Excel report, the best technique to use would probably be DDE, IOM, LIBNAME engine, ExcelXP or ODS MSOffice2K.

WHO IS MY CUSTOMER? HOW MANY PEOPLE WILL USE THE REPORT?

This is an important question. Depending on the customer, we will spend more or less time creating the Excel report. For example if the customer is a client, I will spend more time developing a nice report to try to impress him. DDE, ExcelXP, ODS MSOffice2K, or LIBNAME engine would be the first choice.

On the other hand, if I need to create a report for myself or colleague who is also a SAS programmer, I will spend less time on customizing the report and choose the fastest method to generate the file: a simple PROC EXPORT, ODS MSOffice2K, or ODS CSV.

IS IT A ONE-TIME REPORT?

If the report is intended to be used only once, I will spend less time on it than a report that will be run multiple times and used by many people.
For a one-time report, the method used would also depend on the customer. A PROC EXPORT, LIBNAME engine, or ODS MSOffice2K will be good for most cases. If I still need to format the output, I would probably choose ExcelXP or MSOffice2K tagset, and use a pre-defined style and some simple options to improve the output.

HOW IS THE REPORT SENT TO THE CUSTOMERS? HOW BIG WILL MY FILE BE?

The size of the data set exported to Excel, as well as the SAS installation, are two factors that will help in choosing the best method. If SAS is executed on a server without Excel installed, we will not be able to use DDE for a customized report. LIBNAME engine will be a better solution.

Sometimes we need to export a very large data set to Excel and then send the file to the customersby email. In that case, ODS CSV can be a good solution even if customization is limited. Using ODS MSOffice2K or ExcelXP tagset would generate a large file that we might not be able to send by email.

In the table below, we used the different techniques described in this paper with a data set of 60000 rows and 27 columns. As expected, ODS CSV produces the smallest file.

SAS techniques	Output size
ODS CSV	12 mb
Proc Export	18 mb
LIBNAME engine	18 mb
DDE	22 mb
ODS HTML	57 mb
ODS MSoffice2k	97 mb
ExcelXp	143 mb

DO I NEED A MULTI-SHEET REPORT?

If I need a multi-sheet report I can use different techniques. For multi-tabs without formatting, PROC EXPORT or LIBNAME engine are sufficient. If the output needs to be formatted, different techniques can be used. A template can be prepared in different tabs and populated with LIBNAME or DDE. However, using a template is not a good idea if the report has many tabs as it would take too much time to prepare each sheet.

We can also use a SAS macro to create multi-sheet reports. We create multiple Excel files with SAS and then use the SAS macro to combine these individual Excel files into one worksheet. This kind of macro has been described previously in several other papers.
We can also download and use the modified tagset MSOffice2K_x. This tagset adds options to the original MSOffice2K tagset. One of the new options allows users to add multiple worksheets per Excel file.

However, the ideal solution to export SAS data to Excel workbooks that contain multiple worksheets is ExcelXP tagset. By using styles and the numerous options available with the tagset, we can create a custom template that will be applied to all the tabs. The tagset will create the Excel file with the tabs and style defined in the SAS code.

WILL I USE AN EXCEL TEMPLATE FOR MY REPORT?

We sometimes need to use an Excel template designed by an Excel expert or by ourselves. In that case, the two best methods are LIBNAME engine and DDE. PROC EXPORT can be used with an Excel template but formatting options are too limited.
The syntax is quite different between LIBNAME and DDE. With the LIBNAME engine, we need to use named ranges in Excel and make sure the data will fit in the template. For DDE, only the column names need to be set up in the Excel template. It doesn't require named ranges, and data will be output in the rows and columns defined in the SAS code.
The biggest advantage of DDE over LIBNAME engine is the possibility to call and execute VBA macros.

WHICH EXCEL VERSION WILL BE USED TO READ THE REPORT?

Multiple versions of Excel exist on the market. The latest version is able to read an XML file. Unfortunately, some companies/customers are still using an old version of Excel like Excel 97 or 2000 that cannot read an XML document. It is one of the parameters that needs to be taken into consideration before sending a report to a customer. The techniques described above generate different types of output which are not necessarily compatible with all versions of Excel.

SAS techniques	Output format	Compatibility
ExcelXP	XML	2002+
MsOffice2k	HTML	2000+
ODS HTML	HTML	97+
ODS CSV	CSV	97+
LIBNAME engine	XLS	97+
DDE	XLS	97+
PROC EXPORT	XLS	97+

CONCLUSION

SAS is a powerful tool that provides programmers with a wide range of solutions and options for exporting data to Excel. Although several methods exist to accomplish the same result, they might not necessarily be appropriate for the task at hand. While some techniques offer better customization of reports, others provide simpler, more efficient syntax. By having a limited knowledge of these methods, programmers may not be taking advantage of the full capabilities of SAS. SAS programmers may thus find it beneficial to be familiar with multiple techniques in order to choose the most effective approach for their project.

REFERENCES

Ying Feng, SUGI 2005, Generating Custom Report Tables: Using SAS with DDE and VBA
http://www2.sas.com/proceedings/sugi30/161-30.pdf

Paul A. Choate, Carol A. Martell, SUGI 2006, De-Mystifying the SAS® LIBNAME Engine in Microsoft Excel: A Practical Guide http://www2.sas.com/proceedings/sugi31/024-31.pdf

Andrews, NESUG2008, Printable Spreadsheets Made Easy: Utilizing the SAS® Excel XP Tagset, Rick
http://www.nesug.org/proceedings/nesug08/ap/ap06.pdf

Vincent DelGobbo, pharmasug 2007, Creating Multi-Sheet Excel Workbooks the Easy Way with SAS
http://www.lexjansen.com/pharmasug/2005/posters/po33.pdf

Ralph Winters, NESUG2004, Excellent Ways of Exporting SAS Data to Excel
http://www.nesug.org/proceedings/nesug04/io/io09.pdf

Steven First, Jennifer First, SUGI2010, Choosing the Right Tool from Your SAS® and Microsoft Excel® Tool Belt
http://support.sas.com/resources/papers/proceedings10/144-2010.pdf

Vincent DelGobbo, SGF 2009, More Tips and Tricks for Creating Multi-Sheet Microsoft Excel Workbooks the Easy Way with SAS http://support.sas.com/resources/papers/proceedings09/152-2009.pdf

The Devil Is in the Details: Styles, Tips, and Tricks That Make Your Microsoft Excel Output Look Great!, Eric Gebhart, SUGI2008
http://www2.sas.com/proceedings/forum2008/036-2008.pdf

SAS® Proc Report and ODS ExcelXP Tagsets to Produce Customized Excel Output Without DDE, Mira Shapiro,SESUG2010
http://analytics.ncsu.edu/sesug/2010/RIV04.Shapiro.pdf

Make Bill Gates and Dr. Goodnight Run Your SAS Code: Using VBA, ADO and IOM to Make Excel and SAS Play Nice, Ted Conway,SUGI2005
http://www2.sas.com/proceedings/sugi30/157-30.pdf

Chevell Parker,SAS 9.1 MS OFFICE integration
http://support.sas.com/rnd/base/ods/templateFAQ/office91.pdf

CONTACT INFORMATION

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

Romain Miralles
Clinovo.
1208 E. Arques Avenue, suite 114
Sunnyvale, CA 94085
E-mail : mrom34@gmail.com
Web : http://www.clinovo.com