

Creating Multi-Sheet Excel Workbooks the Easy Way with SAS®
Vincent DelGobbo, SAS Institute Inc., Cary, NC

ABSTRACT

Transferring SAS® data and analytical results between SAS and Microsoft Excel can be difficult, especially when SAS is not installed on a Windows platform. This paper discusses using the new XML support in Base SAS® 9.1 software to create multi-sheet Microsoft Excel workbooks (versions 2002 and later). You will learn step-by-step techniques for quickly and easily creating attractive multi-sheet Excel workbooks that contain your SAS output. Tips and tricks with the ExcelXP ODS tagset will be divulged. Most importantly, the techniques that are presented in this paper can be used regardless of the platform on which SAS software is installed. You can even use them on a mainframe! The use of SAS server technology is also discussed. Although the title is similar to a paper presented at SUGI 31 (DelGobbo, 2006), this paper contains new and revised material not previously presented by this author.

INTRODUCTION

This paper provides you with step-by-step instructions for using Base SAS 9.1 to create the Excel workbook shown in Figures 1 and 2.

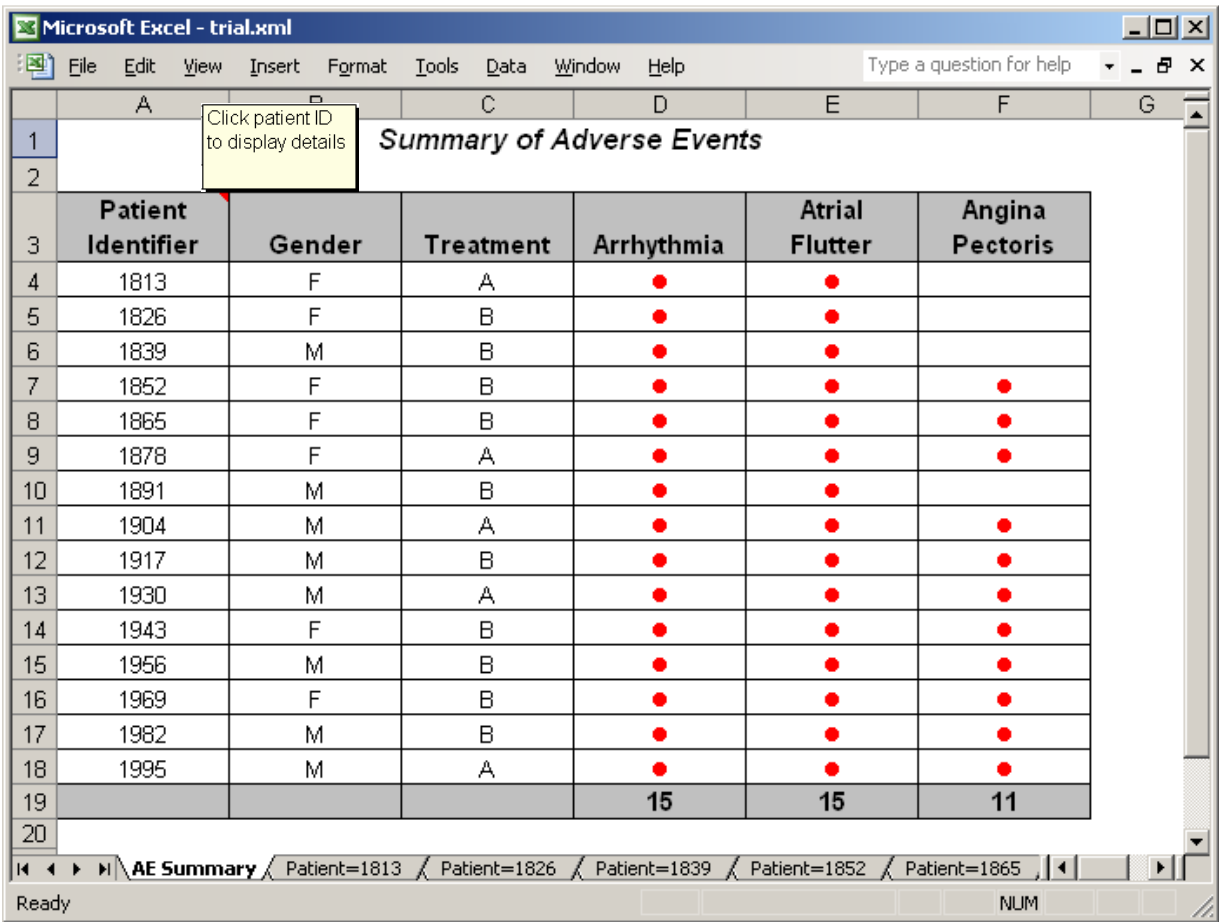


Figure 1. Sample Multi-Sheet Excel Workbook (REPORT Procedure Output)

Microsoft Excel - trial.xml

File Edit View Insert Format Tools Data Window Help

Type a question for help

Patient Visit Log

Patient=1813 Gender=F Treatment=A Dosage (mg)=400

Visit	Visit Date	Systolic BP	Change from Baseline	Diastolic BP	Change from Baseline	Arrhythmia	Atrial Flutter	Angina Pectoris
1	08JAN1991	109		73		•		
2	15JAN1991	149	40	73	0			
3	22JAN1991	141	32	73	0		•	
4	29JAN1991	147	38	73	0			
5	05FEB1991	139	30	74	1	•	•	
6	12FEB1991	149	40	73	0	•	•	

Click to return to AE Summary

Ready NUM

Navigation bar: AE Summary Patient=1813 Patient=1826 Patient=1839 Patient=1852 Patient=1865 Patient=1878 Patient=1891 Patient=

Figure 2. Sample Multi-Sheet Excel Workbook (PRINT Procedure Output)

Figures 1 and 2 show two different worksheets of the same Excel workbook. The data describes the adverse events for a small clinical trial of a fictitious drug. The data and output are modeled after the data and output of the sample SAS software program odsrep4.sas. This sample program ships with Base SAS, and, assuming a typical installation of SAS 9.1 on Windows, can be found in the directory !SASROOT\core\sample\ (for example, C:\Program Files\SAS\SAS 9.1\core\sample\). For assistance accessing the sample library programs, see <http://support.sas.com/faq/003/FAQ00336.html>.

The worksheet named "AE Summary" (Figure 1) presents information about the patients in the trial, along with a summary of the adverse events that were experienced. The summary row at the bottom of the table displays the number of patients that experienced a given adverse event. Clicking on a patient identifier will display the worksheet containing detailed information for that patient, as indicated by the Excel comment visible as a tool tip. For example, clicking on patient identifier "1813" displays the worksheet "Patient=1813" (see Figure 2).

The columns "Systolic BP" and "Diastolic BP" in the detailed worksheets show the systolic and diastolic blood pressure readings over time, with the values recorded at the first visit being considered as the baseline values. The "Change from Baseline" columns are computed and show the difference between the value recorded on a given day and the baseline value.

The remainder of this paper will guide you through the steps used to create the Excel workbook shown in Figures 1 and 2 using Base SAS 9.1 software. You can download a copy of the code and data used in this paper from the SAS Presents Web site at support.sas.com/saspresents. Find the entry "Creating Multi-Sheet Excel Workbooks the Easy Way with SAS".

The code in this paper was tested using SAS 9.1.3 Service Pack 4 and Microsoft Excel 2002 SP3 software.

REQUIREMENTS

In order to use the techniques in this paper, you must have the following software:

1. Base SAS 9.1.3 or later, on *any* supported operating system and hardware.
2. Microsoft Excel 2002 or later (also referred to as Microsoft Excel XP).
3. An updated version of the SAS ExcelXP ODS tagset.

For more information about obtaining an updated version of the tagset, see "The ExcelXP Tagset" section later in this paper.

OUTPUT DELIVERY SYSTEM (ODS) BASICS

ODS is the part of Base SAS software that enables you to generate different types of output from your procedure code. An ODS *destination* controls the type of output that's generated (HTML, RTF, PDF, etc.). An ODS *style* controls the appearance of the output. In this paper, we use a type of ODS destination, called a *tagset*, that creates XML output that can be opened with Excel. This tagset, named ExcelXP, creates an Excel workbook that has multiple worksheets.

The Excel workbook in Figures 1 and 2 was created using the ExcelXP ODS tagset and the XLSansPrinter ODS style. The ExcelXP tagset creates an XML file that, when opened by Excel, is rendered as a multi-sheet workbook. All formatting and layout are performed by SAS; there is no need to "hand-edit" the Excel workbook. You simply use Excel to open the file created by ODS.

The ODS statements that generate XML that is compatible with versions of Microsoft Excel 2002 and later follow this general form:

```
❶ ods listing close;
❷ ods tagsets.ExcelXP style=style-name file='file-name.xml' ... ;
   * Your SAS code here;
❸ ods tagsets.ExcelXP close;
```

The first ODS statement (❶) closes the LISTING destination, which writes output to a listing file in batch mode, or to the Output window when SAS is run interactively. We only want to generate XML output for use with Excel.

The second ODS statement (❷) uses the ExcelXP tagset to generate the XML output and then stores the output in a file. The STYLE option controls the appearance of the output, such as the font and color scheme. To see a list of ODS styles that are available for use at your site, submit the following SAS code:

```
ods listing;
proc template; list styles; run; quit;
```

The third ODS statement (❸) closes and releases the XML file so that it can be opened with Excel.

Although you can store your output on a local disk (where SAS software is installed), or a network-accessible disk, here are some good reasons to store your SAS output on a Web server:

- The files are available to anyone who has network access.
- The XML files can be accessed by Web-enabled applications besides Excel.
- You can take advantage of Web server authentication and security models.

If you place the files where others can access them over a network, you should set file permissions to prevent accidental alteration.

OPENING ODS OUTPUT WITH EXCEL

To open an ODS-generated file that's stored on a Web server, follow these steps:

1. In Excel, select **File** ➤ **Open**.
2. In the "File name" field, specify the full URL to the file you want to open. For example, **http://web-server/directory/file-name.xml**.
3. Click **Open** to import the XML file.

To open ODS-generated files from a local or network-accessible disk, follow the same steps, except in step 2 you should either navigate to the desired file or type the path and file name in the "File name" field.

Excel will read and convert the XML file to the Excel format. After the conversion, you can perform any Excel function on the data. To save a copy of the file in Excel binary format, select **File ► Save As** and then, from the "Save as type" drop-down list, choose **Microsoft Excel Workbook (*.xls)**.

SETTING UP THE ODS ENVIRONMENT

Our sample code uses a user-defined style named `XLsansPrinter` and a modified version of the `ExcelXP` tagset. Use the `ODS PATH` statement to set the location where this style and tagset will be stored on your system:

```
❶ libname mylib 'some-directory'; * Location to store tagsets and styles;

❷ ods path mylib.tmplmst(update) sashelp.tmplmst(read);
```

The `LIBNAME` statement (❶) specifies where to store the user-defined tagsets and styles. Although you can temporarily store tagsets and styles in the `WORK` library, it's more efficient to create them once, and store them in a permanent library so that you can reference them in other SAS programs.

The `ODS PATH` statement (❷) specifies the locations and the order in which to search for ODS tagsets and styles. Notice that the access mode for `mylib.tmplmst` is specified as "update" and the access mode for `sashelp.tmplmst` is specified as "read". Because ODS searches the `PATH` in the order given and the access mode for `mylib.tmplmst` is "update", `PROC TEMPLATE`, used later in this paper, will store tagsets and styles in a file named `tmplmst.sas7bitm` in the directory that's associated with the `MYLIB` library.

You can think of the `ODS PATH` statement as functioning like the `SAS FMTSEARCH` option. You include the `LIBNAME` and `ODS PATH` statements in each SAS program that needs to use the tagsets and styles that you create.

THE EXCELXP TAGSET

Once you have issued the appropriate `ODS PATH` statement, you can import the modified version of the `ExcelXP` tagset and use it in your SAS programs. The version of the tagset used in this paper can be found in the download package on the SAS Presents Web site at support.sas.com/saspresents. Find the entry "Creating Multi-Sheet Excel Workbooks the Easy Way with SAS". The download package contains a file named "ExcelXP.sas", which contains the SAS code for creating the `ExcelXP` tagset. Save a copy of this file, and submit the following SAS code to make the tagset available:

```
%include 'ExcelXP.sas'; * Specify path to the file, if necessary;
```

You should have to perform this step only once. The `ExcelXP` tagset will be imported and stored in the directory corresponding to the `MYLIB` library. All of your future SAS programs can access the tagset by specifying the correct `LIBNAME` and `ODS PATH` statements (see "Setting up the ODS Environment" above).

The `ExcelXP` tagset supports a number of different options that control both the appearance and functionality of the Excel workbook. To see a listing of the supported options, submit this SAS code:

```
filename temp temp;
ods tagsets.ExcelXP file=temp options(doc='help');
ods tagsets.ExcelXP close;
```

The tagset information is printed to the SAS Log. For your convenience, this information is included in the download package for this paper.

IMPORTANT NOTE

The version of the `ExcelXP` tagset shipped with Base SAS 9.1 has undergone many revisions since its initial release. In order to take advantages of the features discussed in this paper, you must download a recent copy of the tagset and install it on your system as described above. The version of the tagset used in this paper can be found in the

download package on the SAS Presents Web site at (support.sas.com/saspresents). The most recent version of the tagset is available from the ODS Web site ("ODS MARKUP Resources", SAS Institute Inc.).

A BRIEF ANATOMY OF ODS STYLES

ODS styles control all aspects of the appearance of the output, and Base SAS software ships with over 40 different styles. A style is composed of *style elements*, each of which controls a particular part of the output. For example, all styles contain a style element named "header" that controls the appearance of column headings. Style elements consist of collections of *style attributes*, such as the background color and font size. The definition of the style element named "header" might look like this:

```
style header /  
  font_face   = "Arial, Helvetica"  
  font_size   = 11pt  
  font_weight = bold  
  foreground  = black  
  background  = #bbbbbb;
```

As you can see, this style element contains the style attributes "font_face", "font_size", and so on. The TEMPLATE procedure can be used to modify an existing style or to create a completely new style from scratch. The next section describes using the TEMPLATE procedure to modify an existing style.

THE XLSANSPRINTER STYLE

Although the appearance of the workbook shown in Figures 1 and 2 closely resembles what is generated using the standard ODS style named sansPrinter, the workbook was created using a custom ODS style named XlsansPrinter, which is based on the standard sansPrinter style.

The TEMPLATE procedure was used to make minor changes to the sansPrinter style, and the resulting style was saved using the name XlsansPrinter, to indicate that this style is intended to be used with Microsoft Excel. The complete code for creating this style can be found in the appendix of this paper, in the section "Code for Creating the XlsansPrinter Style".

The justification attribute for the standard style elements "header", "rowheader", and "data" has been set to center, to ensure that all data in the workbook are centered within their cells. A new style element named "data_bullet" was created based on the newly redefined "data" style element. The "data_bullet" style element contains a style attribute named "tagattr" that sets the Excel format, and a style attribute to change the font to Wingdings. The style element "header_patientcomment" is used to add a comment to one of the header cells in the workbook. The "data_bullet" and "header_patientcomment" style elements will be used in the section "Changing the Appearance Using ODS Style Overrides and Excel Formats".

Run the code in the appendix to create the XlsansPrinter style on your system. You should have to perform this step only once. The XlsansPrinter style will be stored in the directory corresponding to the MYLIB library. All of your future SAS programs can access the style by specifying the correct LIBNAME and ODS PATH statements (see the "Setting up the ODS Environment" section).

Because an in-depth discussion of creating and using ODS styles is beyond the scope of this paper, see the chapter about the TEMPLATE procedure in the ODS documentation (SAS Institute Inc. 2004).

USING ODS TO CREATE THE MULTI-SHEET EXCEL WORKBOOK

By default, the ExcelXP tagset will create a new worksheet each time a SAS procedure creates new tabular output. Our sample code executes the REPORT procedure to create the first worksheet. Subsequent worksheets are created by the PRINT procedure, with a new worksheet being created for each distinct BY group. The resulting workbook contains a total of 16 worksheets.

SAS CODE TO CREATE THE EXCEL WORKBOOK

Here is a listing of the *basic* SAS code used to create the Excel workbook:

```
options center;

ods listing close;
❶ ods tagsets.ExcelXP path='output-directory' file='trial.xml' style=XLSansPrinter;

    * Create a summary table to use as the summary/table of contents;

❷ proc means data=sample.trial sum noprint;
    by patient;
    var arrhythmia flutteratr angina;
    id sex drug;
    output out=trialsummary(drop=_type_ _freq_) sum=;
run; quit;

    * Create the summary/table of contents worksheet;

title 'Summary of Adverse Events';
footnote;

❸ proc report nowindows data=trialsummary split='*';

    columns patient sex drug arrhythmia flutteratr angina;

    define patient / display;
    define sex--drug / display;
    define arrhythmia--angina / analysis n;

    rbreak after / summarize;
run; quit;

    * Create the detailed worksheets for each patient;

title 'Patient Visit Log';
footnote 'Click to return to AE Summary';

options missing=' ';

❹ proc print data=sample.trial noobs label split='*';
    by patient sex drug dosage;
    pageby patient;
    id visit visitdate;
    var systolic diastolic;
    var arrhythmia flutteratr angina;
    label patient = 'Patient'
           sex    = '    Gender'
           drug   = '    Treatment'
           dosage = '    Dosage (mg)';
run; quit;

options missing='.';

ods tagsets.ExcelXP close;
```

As you can see (❶), the ExcelXP tagset is used to generate the output, and the XLSansPrinter style is used to control the appearance aspects of the output. The MEANS procedure (❷) is used to create a summary table, and PROC REPORT (❸) is run to create the first worksheet based on this summarized data. The PRINT procedure (❹)

is run with a BY statement to create the remaining patient detail worksheets. The SAS table "trial" is included in the download package for this paper.

Figures 3 and 4 show the result of executing the SAS code above and opening the file "trial.xml" using Excel.

	A	B	C	D	E	F
	Patient Identifier	Gender	Treatment	Arrhythmia	Atrial Flutter	Angina Pectoris
1	1813	F	A	1	1	0
2	1826	F	B	1	1	0
3	1839	M	B	1	1	0
4	1852	F	B	1	1	1
5	1865	F	B	1	1	1
6	1878	F	A	1	1	1
7	1891	M	B	1	1	0
8	1904	M	A	1	1	1
9	1917	M	B	1	1	1
10	1930	M	A	1	1	1
11	1943	F	B	1	1	1
12	1956	M	B	1	1	1
13	1969	F	B	1	1	1
14	1982	M	B	1	1	1
15	1995	M	A	1	1	1
16				15	15	11
17						
18						
19						
20						
21						

Figure 3. ODS ExcelXP-Generated Workbook (REPORT Procedure Output)

Notice that Figure 3 does not match Figure 1. The problems exhibited in Figure 3 are:

1. Zeroes and ones appear as data in the adverse event columns, instead of blanks and red bullets.
2. The background color for the summary row at the bottom of the window should be gray, and the values should appear as bold, non-italic text.
3. The Excel comment is missing from the "Patient Identifier" column.
4. The title line is missing.
5. A default worksheet name was used.
6. The values in the "Patient Identifier" columns are not "clickable".
7. Some columns are narrower than they should be, causing the column headings to be obscured.

Similarly, Figure 4 does not match Figure 2. The problems are:

1. Blanks and ones appear as data in the adverse event columns, instead of blanks and red bullets.
2. The title and footnote lines are missing.
3. A default worksheet name is used.
4. The "Change from Baseline" columns are missing.

We will now make changes to the preceding SAS code to correct these problems.

Visit	Visit Date	Systolic BP	Diastolic BP	Arrhythmia	Atrial Flutter	Angina Pectoris
1	08JAN1991	109	73	1		
2	15JAN1991	149	73			
3	22JAN1991	141	73		1	
4	29JAN1991	147	73			
5	05FEB1991	139	74	1	1	
6	12FEB1991	149	73	1	1	

Figure 4. ODS ExcelXP-Generated Workbook (PRINT Procedure Output)

CHANGING THE APPEARANCE USING ODS STYLE OVERRIDES AND EXCEL FORMATS

As mentioned earlier, ODS styles control the appearance of the output. If the default style elements do not provide you with the desired appearance, you can use ODS style overrides to change the style elements used for the various portions of your output.

Style overrides are supported by the PRINT, TABULATE, and REPORT procedures. Style overrides are best used for changing a small number of features because, if overused, they can result in excessive output file size. Style overrides can be specified in several ways, with the two most common being:

- ❶ `style=[style-attribute-name1=value1 style-attribute-name2=value2 ...]`
- ❷ `style=style-element-name`

The *style-attribute-name* is the name of a style attribute that you want to override. The *style-element-name* is the name of the style element that you want to use.

The first format (❶) uses individual style attributes, and while it is the most commonly used format, it is also an inefficient way of applying the style override. Here is a style override that can be used to display red bullet characters for positive numbers in your output:

```
style=[tagattr='format:[Red]\l;_l' font_face=Wingdings]
```

Although useful, style overrides of this type should be avoided when possible because the additional XML data that is output can make an XML file dramatically larger than if it had been generated without the style overrides. The XML files generated by the ExcelXP tagset are already large because they comply with the Microsoft XML Spreadsheet Specification ("XML Spreadsheet Reference", Microsoft Corporation). Inefficient use of style overrides will result in files that need more disk space and take longer to open in Excel.

The second format (❷) is the most efficient, as it has the smallest impact on output file size. You simply specify the name of the style element you want ODS to use. Using this format involves creating a new or modified style element, setting the style attributes within the element, and then using the style element name in your style override.

In the case of the XLSansPrinter style, the style element named "data_bullet" was created:

```
style data_bullet from data /
  tagattr='format:[Red]\l;_l'
  font_face=Wingdings;
```

This style element has the same attributes as the "data" style element from which it was based upon, except that the font Wingdings is used and an Excel format is applied to the data in the cells via the "tagattr" attribute. The preceding Excel format contains three sections, separated by semicolons. The first section, [Red]\l, is applied to numeric values that are greater than zero. Excel will display these values as a red lowercase letter "l". Since the Wingdings font is being used, the lowercase letter "l" appears as a bullet character. Tables listing the characters in the Wingdings font can be found in the appendix of this paper, in the section "The Wingdings Font for Microsoft Windows".

The second section of the Excel format is applied to numeric values that are less than zero. As you can see, no format is defined for this section. The definition was omitted because none of the data in the SAS output contains values that are negative. The third section of the format, _l is applied to values that are equal to zero. Zeroes and blanks will be converted to a blank character, the width of a lowercase letter "l". To find out more about Excel formats, refer to a book on Excel or type "create a custom number format" into the Excel help system.

The XLSansPrinter contains other new and modified style elements. Refer to the appendix for a full listing of the code for this style.

Now that the XLSansPrinter style contains all the new and modified style elements needed, it is time to apply the style elements using style overrides. The modified REPORT procedure code is:

```
proc report nowindows data=trialsummary split='*' style(summary)=header;

  columns patient sex drug arrhythmia flutteratr angina;

  define patient / display style(header)=header_patientcomment;
  define sex--drug / display;
  define arrhythmia--angina / analysis n style(column)=data_bullet;

  rbreak after / summarize;
run; quit;
```

These style overrides will cause the appearance of the data shown in Figure 3 to match what is shown in Figure 1. The summary line will have a gray background with bold text, and the values in the adverse events columns will be displayed as a blank (for 0's) or a red bullet (for 1's). Additionally, when hovering over the column heading "Patient Identifier", "Click patient ID to display details" will be displayed as a tool tip.

A single style override is needed to display red bullets in the PROC PRINT output:

```
var arrhythmia flutteratr angina / style(data)=data_bullet;
```

GETTING THE TITLE AND FOOTNOTE INTO THE WORKBOOKS: AN INTRODUCTION TO TAGSET OPTIONS

As mentioned earlier, the ExcelXP tagset supports several options that control both the appearance and functionality of the Excel workbook. Many of these tagset options are simply tied straight into existing Excel options or features. Tagset options are specified on the ODS statement using the "options" keyword:

```
ods tagsets.ExcelXP options(option-name1='value1' option-name2='value2' ...) ... ;
```

It is important to note that tagset options remain in effect until they are either turned off or set to another value. Since multiple ODS statements are allowed, it is good practice, in terms of functionality and code readability, to

explicitly reset tagset options to their default value when you are finished using them. For example:

```
ods tagsets.ExcelXP options(option-name='some-value') ... ;
* Your SAS code here;
ods tagsets.ExcelXP options(option-name='default-value') ... ;
* Other SAS code here;
```

By default, SAS titles and footnotes appear as Excel print headers and print footers, which are displayed only when the Excel document is printed. To include the titles and footnotes in the worksheets, use the EMBEDDED_TITLES and EMBEDDED_FOOTNOTES options:

```
options center;

ods listing close;
ods tagsets.ExcelXP path='output-directory' file='trial.xml' style=XLSansPrinter;

* PROC MEANS code;

ods tagsets.ExcelXP options(embedded_titles='yes' embedded_footnotes='yes');

title 'Summary of Adverse Events';
footnote;

* PROC REPORT code;

ods tagsets.ExcelXP options(embedded_titles='yes' embedded_footnotes='yes');

title 'Patient Visit Log';
footnote 'Click to return to AE Summary';

* PROC PRINT code;

ods tagsets.ExcelXP close;
```

ASSIGNING CUSTOM WORKSHEET NAMES

ODS generates a unique name for each worksheet, as required by Microsoft Excel. There are, however, several tagset options that you can use to alter the names of the worksheets. The SHEET_NAME option is used to explicitly set the worksheet name, and will be used to set the first worksheet name to "AE Summary". For the PRINT procedure output, we will use a combination of the SHEET_NAME and SHEET_INTERVAL options, which will cause the first BY variable and value to be used in the worksheet name.

The following code will cause the worksheet names to match those shown in Figures 1 and 2.

```
options center;

ods listing close;
ods tagsets.ExcelXP path='output-directory' file='trial.xml' style=XLSansPrinter;

* PROC MEANS code;

ods tagsets.ExcelXP options(embedded_titles='yes' embedded_footnotes='yes'
                             sheet_name='AE Summary');

title 'Summary of Adverse Events';
footnote;

* PROC REPORT code;
```

```
ods tagsets.ExcelXP options(embedded_titles='yes' embedded_footnotes='yes'
                             sheet_name='none' sheet_interval='bygroup');

title 'Patient Visit Log';
footnote 'Click to return to AE Summary';

* PROC PRINT code;

ods tagsets.ExcelXP close;
```

ADDING "DRILL DOWN" TO THE SUMMARY WORKSHEET

The action of clicking on an item in order to display more detailed information about that item is commonly referred to as "drill down". In our case, we want detailed patient information to be displayed when we click on a patient identifier in the summary worksheet. This can be implemented by adding a COMPUTE block to the existing REPORT procedure code:

```
proc report nowindows data=trialsummary ... ;
  columns ... ;
  define ... ;
  rbreak after / summarize;

  *,
  *   Build an Excel link to cell A1 of the various detailed worksheets
  *   ('#Patient=1813'!A1, '#Patient=1826'!A1, etc.).
  *,
  compute patient;
    urlstring = "#'Patient=" || strip(put(patient, 4.)) || "'!A1";
    call define(_col_, 'URL', urlstring);
  endcomp;

run; quit;
```

To link to the cells of an Excel workbook, the general format of the link is:

file-name#'worksheet-name'!cell-reference

When linking to a cell in the same workbook, as we are, it is best to omit the file name. This will prevent errors if the file is renamed from "trial.xml" to some other name. As you can see from Figure 1, the worksheet names we want to drill down to follow the format "Patient=*patient-identifier*". The COMPUTE block creates a variable according to this pattern, and associates it as a drill down link to the corresponding patient identifier. Thus, all patient identifier values are now clickable, and will link to the worksheet containing the detailed information for the corresponding patient.

In Figure 2 the text of the footnote is "Click to return to AE Summary", indicating that clicking on the footnote text will display the "AE Summary" worksheet. To make the text of the footnote clickable, add a LINK attribute to the FOOTNOTE statement before the PRINT procedure code:

```
footnote link="#'AE Summary'!A1" 'Click to return to AE Summary';
```

The ExcelXP tagset uses the value supplied in the LINK attribute to create a link associated with the footnote text. When the text is clicked on, Excel will navigate to cell A1 of the "AE Summary" worksheet.

ADDING BASELINE DIFFERENCES USING EXCEL FORMULAS

To populate the patient detail worksheets with blood pressure change from baseline values, we must change the PRINT procedure code. We could use DATA step code to compute the values, but doing so would result in "static" data in the workbook. If the systolic and diastolic values are updated in the workbook, the change from baseline value would not update itself. Instead, we will use DATA step code to create columns that contain Excel formulas

that will automatically calculate the differences. By doing so, we create "dynamic" data that will update itself if the blood pressure readings are updated in the Excel workbook.

Excel formulas begin with an equal sign ("="), and usually contain references to other cells in a worksheet. There are two different methods for referencing cells: the A1 style and the R1C1 style. The A1 style, used earlier to add drill down links to the workbook, is probably the most widely used format. However, the Microsoft XML Spreadsheet Specification requires that, when using XML to build the spreadsheets, R1C1-style references be used because "they are significantly easier to parse and generate than A1-style formulas" ("XML Spreadsheet Reference", Microsoft Corporation). Table 1 provides some examples of formulas that use R1C1-style references.

Formula	Explanation of the Computation Performed
=R[-2]C	Reference the value in the cell 2 rows up and in the same column.
=R[2]C[2]*100	Multiply the value in the cell 2 rows down and 2 columns to the right by 100.
=SUM(RC[-5]:RC[-1])	Sum the values in the cells 5 columns to the left in the same row through 1 column to the left in the same row.
=RC[-1]-R[-4]C[-1]	Subtract the value in the cell 4 rows up and 1 column to the left from the value in the cell 1 column to the left.

Table 1. Sample Excel Formulas Using R1C1-Style References

The last row of the table contains the type of formula we need to calculate the baseline blood pressure differences ("Change from Baseline"). Recall that the two baseline blood pressure values are stored in the first row of the systolic and diastolic data (see Figure 2). This baseline value is followed by subsequent blood pressure values. In order to calculate the change from baseline, we must count upwards from the current value, find the baseline value, and then subtract it from the current value. To accomplish this, add the following DATA step code before the PROC PRINT code:

```
data trial; set sample.trial;

length diastolicdiff systolicdiff $20;

if (visit ne 1) then do;
    systolicdiff = '=RC[-1]-R[-' || compress(put(visit-1, best.)) || ']C[-1]';
    diastolicdiff = systolicdiff;
end;

label systolicdiff = 'Change*from*Baseline'
      diastolicdiff = 'Change*from*Baseline';
run;
```

The columns SYSTOLICDIFF and DIASTOLICDIFF contain the Excel formula used to calculate the baseline differences. You will also need to modify the PRINT procedure code to use this table and the SYSTOLICDIFF and DIASTOLICDIFF columns:

```
proc print data=trial noobs label split='*';
    ... ;
    var systolic systolicdiff diastolic diastolicdiff;
    ... ;
run; quit;
```

ADJUSTING COLUMN WIDTHS FOR THE REPORT AND PRINT PROCEDURE OUTPUT

The ExcelXP tagset takes into account several aspects of the SAS output when determining the column width:

- Character variable length (LENGTH statement/attribute).
- Number of characters in the column label.
- Number of characters in the data value.
- Font size and weight.
- Fudge Factor (an arbitrary value used to fine tune column widths).

Some procedures, such as REPORT and TABULATE, do not provide the tagset with enough information to compute a column width. That is why the columns in Figure 3 are narrower than they need to be, causing some column headings to be obscured. We plan to fix this problem in a future release, but for now, this can be corrected by adding the ABSOLUTE_COLUMN_WIDTH tagset option to the ODS statement preceding the REPORT procedure code:

```
ods tagsets.ExcelXP options(embedded_titles='yes' embedded_footnotes='yes'  
                             sheet_name='AE Summary'  
                             absolute_column_width='9');
```

The value specified for this option is the width of columns, in characters. This is only an approximation, because the font size, font weight and a fudge factor are also used in the calculation. For example, the longest column heading in Figure 3 is the word "Identifier", which contains 10 characters. However, by trial-and-error it was determined that specifying "9" for ABSOLUTE_COLUMN_WIDTH provided an optimal column width.

In our example, the tagset will use "9" for a width of all columns in the output. If you want to specify explicit values for each column, specify a comma-separated list of values, with one value for each column.

Currently, the baseline difference columns in the patient detail worksheets are wider than they need to be because the tagset is using the number of characters in the data value (the Excel formula) to compute the width. This problem can also be corrected by specifying the ABSOLUTE_COLUMN_WIDTH option. It was determined that "9" was also a good value to use to correct the column widths in this output. Because tagset options stay in effect until they are changed, it is not necessary to specify the ABSOLUTE_COLUMN_WIDTH option for the PRINT procedure output. However, explicitly specifying this option helps in the readability and debugging of the SAS code. With this change implemented, the resulting ODS statement would be:

```
ods tagsets.ExcelXP options(embedded_titles='yes' embedded_footnotes='yes'  
                             sheet_name='none' sheet_interval='bygroup'  
                             absolute_column_width='9');
```

SETTING THE PRINT ORIENTATION AND EXCEL "FIT TO PAGE" OPTION

By default, the print orientation is set to portrait mode. This will work well for the "AE Summary" worksheet because it is narrow. However, it is better to use landscape mode for the patient detail worksheets. The print orientation can be changed using the ORIENTATION tagset option.

To ensure that Excel prints a worksheet on a single page, use the ExcelXP tagset option FITTOPAGE. When Excel encounters this option it reduces the size of the worksheet so that it is printed on one page.

Add these two options to the ODS statement that precedes the PRINT procedure output to ensure that each worksheet fits on a single printed page:

```
ods tagsets.ExcelXP options(embedded_titles='yes' embedded_footnotes='yes'  
                             sheet_name='none' sheet_interval='bygroup'  
                             absolute_column_width='9'  
                             orientation='landscape' fittopage='yes');
```

THE FINAL SAS CODE

With all of the modifications in place, the code should appear as follows:

```
options center;

ods listing close;

ods tagsets.ExcelXP path='output-directory' file='trial.xml' style=XLSansPrinter;

* Create a summary table to use as the summary/table of contents;

proc means data=sample.trial sum noprint;
  by patient;
  var arrhythmia flutteratr angina;
  id sex drug;
  output out=trialsummary(drop=_type_ _freq_) sum=;
run; quit;

* Create the summary/table of contents worksheet;

ods tagsets.ExcelXP options(embedded_titles='yes' embedded_footnotes='yes'
                           sheet_name='AE Summary'
                           absolute_column_width='9');

title 'Summary of Adverse Events';
footnote;

proc report nowindows data=trialsummary split='*' style(summary)=header;

  columns patient sex drug arrhythmia flutteratr angina;

  define patient / display style(header)=header_patientcomment ;
  define sex--drug / display;
  define arrhythmia--angina / analysis n style(column)=data_bullet ;

  rbreak after / summarize;

  *;
  * Build an Excel link to cell A1 of the various detailed worksheets
  * ('#Patient=1813'!A1, '#Patient=1826'!A1, etc.).
  *;

  compute patient ;
    urlstring = "#Patient=" || strip(put(patient, 4.)) || "!A1";
    call define(_col_, 'URL', urlstring);
  endcomp;

run; quit;

*
* Add two columns to original data to contain Excel formulas to
* calculate the BP baseline differences.
*

data trial; set sample.trial;

length diastolicdiff systolicdiff $20;

if (visit ne 1) then do;
```

```

        systolicdiff = 'RC[-1]-R[-1] || compress(put(visit-1, best.)) || 'C[-1]';
        diastolicdiff = systolicdiff;
    end;

    label systolicdiff = 'Change*from*Baseline'
          diastolicdiff = 'Change*from*Baseline';
run;

* Create the detailed worksheets for each patient;

ods tagsets.ExcelXP options(embedded_titles='yes' embedded_footnotes='yes'
                           sheet_name='none' sheet_interval='bygroup'
                           absolute_column_width='9'
                           orientation='landscape' fittopage='yes');

title 'Patient Visit Log';
footnote link="#'AE Summary'!A1" 'Click to return to AE Summary';

options missing=' ';

proc print data=trial noobs label split='*';
  by patient sex drug dosage;
  pageby patient;
  id visit visitdate;
  var systolic systolicdiff diastolic diastolicdiff;
  var arrhythmia flutteratr angina / style(data)=data_bullet ;
  label patient = 'Patient'
        sex      = '    Gender'
        drug     = '    Treatment'
        dosage   = '    Dosage (mg)';
run; quit;

options missing='.';

ods tagsets.ExcelXP close;

```

SAS SERVER TECHNOLOGY

If you have licensed SAS/IntrNet software, you can incorporate dynamically generated SAS output into Excel by using the Application Dispatcher. You can also perform similar tasks with the Stored Process Server, which is new for SAS 9.1.

The Application Dispatcher and the Stored Process Server enable you to execute SAS programs from a Web browser or any other client that can open an HTTP connection to either of these SAS servers (which can run on *any* platform where SAS is licensed). The SAS programs that you execute from the browser can contain any combination of DATA step, PROC, MACRO, or SCL code. Thus, all the code that's been shown up to this point can be executed by using either the Application Dispatcher or the Stored Process Server.

Program execution is typically initiated by accessing a URL that points to the SAS server program. Parameters, if any, are passed to the program as name/value pairs in the URL. The SAS server takes these name/value pairs and constructs SAS MACRO variables that are available to the SAS program.

Figure 5 shows a Web page that can be used to deliver SAS output directly to Excel, using the Web browser as the client.

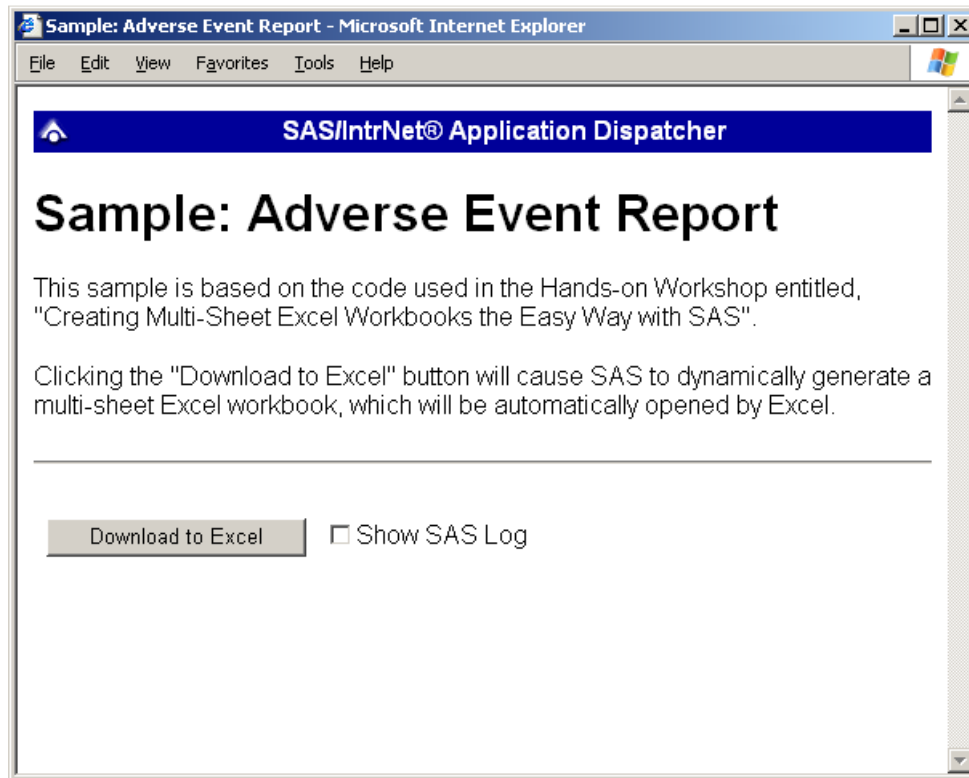


Figure 5. Web Page to Drive a SAS/IntrNet Application

Clicking the "Download to Excel" button would result in the execution of a slightly modified version of the REPORT and PRINT procedure code that we have been working with. The modifications are as follows:

```
%let RV=%sysfunc(appsrv_header(Content-type,application/vnd.ms-excel));
%let RV=%sysfunc(appsrv_header(Content-disposition,attachment; filename=
"Trial.xls")); * Ignore line wrapping;

options center;

ods listing close;

ods tagsets.ExcelXP file=_webout style=XLsansPrinter;
* Remainder of "final" SAS code;
ods tagsets.ExcelXP close;
```

The first APPSRV_HEADER function sets a special MIME header that causes the SAS output to be opened by Excel, instead of being rendered by the Web browser. This statement is required.

The second APPSRV_HEADER function sets a special MIME header that populates the file name field in the "File Download" dialog box. As you can see from Figure 6, the file name appears as "Trial.xls". This header may cause problems with some versions of Excel, so be sure to test you applications before deploying them in a production environment. This statement is optional.

The special, reserved FILEREF _WEBOUT is automatically assigned by the SAS server, and is always used to direct output from the SAS server to the client. Modify your existing ODS statement to direct the output to this FILEREF, instead of an external disk file.

You can click "Open" to directly open your SAS output using Excel, or click "Save" to save a copy for later use.

This is just one example of how you can dynamically deliver SAS output to Excel. For more detailed information and other examples, refer to the SAS/IntrNet Application Dispatcher and/or Stored Process documentation, as well as this author's earlier papers (DeIGobbo, 2002, 2003, 2004).

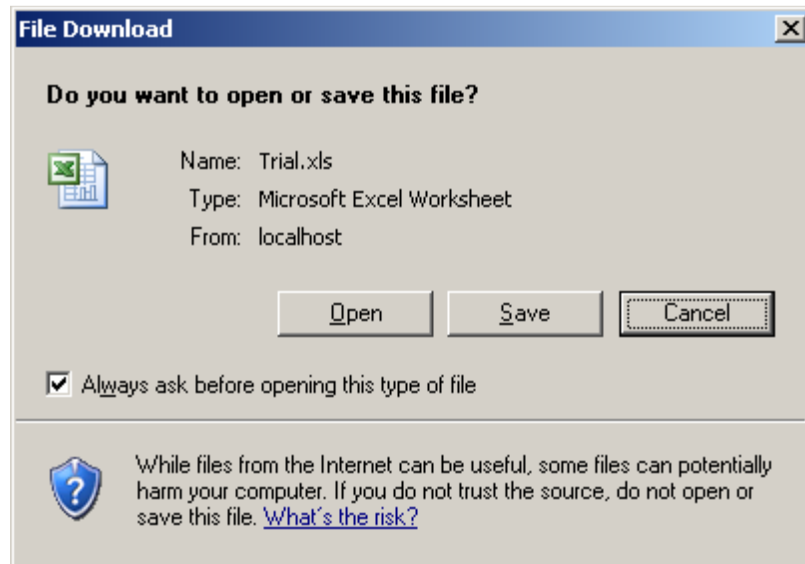


Figure 6. File Download Dialog Box

CONCLUSION

The SAS 9.1 ExcelXP ODS tagset provides an easy way to export your SAS data to Excel workbooks that contain multiple worksheets. By using ODS styles, style overrides, and tagset options, you can customize the output to achieve your design goals. The tagset complies with the Microsoft XML Spreadsheet Specification.

APPENDIX

CODE FOR CREATING THE XLSANSPRINTER STYLE

```
proc template;
  define style styles.XLsansPrinter;
    parent = styles.sansPrinter;

    /* Redefine characteristics of some of the standard style elements */

    style header from header /
      just = center;

    style rowheader from rowheader /
      just = center;

    style data from data /
      just = center;

    /* Assign an Excel format and different font - used with some "data" cells */

    style data_bullet from data /
      tagattr='format:[Red]\l;;_l'
      font_face=Wingdings;

    /* Assign an Excel comment - used with one "header" cell */
```

```

style header_patientcomment from header /
  flyover = 'Click patient ID to display details';

end;
run; quit;

```

THE WINGDINGS FONT FOR MICROSOFT WINDOWS

Keyboard Character	Wingdings	
	Character	Shift-Character
a		
b		
c		
d		
e		
f		
g		
h		
i		
j		
k		
l		
m		
n		
o		
p		
q		
r		
s		
t		
u		
v		
w		
x		
y		
z		

Keyboard Character	Wingdings Character
1	
2	
3	
4	
5	
6	
7	
8	
9	
0	
`	
~	
!	
@	
#	
\$	
%	
^	
&	
*	
(	

Keyboard Character	Wingdings Character
)	
-	
_	
=	
+	
[	
{	
]	
}	
\	
;	
:	
'	
"	
,	
<	
.	
>	
/	
?	

REFERENCES

- DelGobbo, V. 2002. "Techniques for SAS® Enabling Microsoft Office in a Cross-Platform Environment". *Proceedings of the Twenty-Seventh Annual SAS Users Group International Conference*, 27. CD-ROM. Paper 174. Available <http://www2.sas.com/proceedings/sugi27/p174-27.pdf>.
- DelGobbo, V. 2003. "A Beginner's Guide to Incorporating SAS® Output in Microsoft Office Applications". *Proceedings of the Twenty-Eighth Annual SAS Users Group International Conference*, 28. CD-ROM. Paper 52. Available <http://www2.sas.com/proceedings/sugi28/052-28.pdf>.
- DelGobbo, V. 2004. "From SAS® to Excel via XML". Available <http://support.sas.com/saspresents>.

DelGobbo, V. 2006. "Creating AND Importing Multi-Sheet Excel Workbooks the Easy Way with SAS® ". *Proceedings of the Thirty-First Annual SAS Users Group International Conference*, 31. CD-ROM. Paper 115. Available <http://www2.sas.com/proceedings/sugi31/115-31.pdf>.

Microsoft Corporation. "XML Spreadsheet Reference". Available [http://msdn2.microsoft.com/en-us/library/aa140066\(office.10\).aspx](http://msdn2.microsoft.com/en-us/library/aa140066(office.10).aspx).

SAS Institute Inc. 2004. "Chapter 9: TEMPLATE Procedure: Creating a Style Definition". *SAS 9.1 Output Delivery System, User's Guide*. Cary, NC: SAS Institute Inc.

SAS Institute Inc. "ODS MARKUP Resources". Available <http://support.sas.com/rnd/base/topics/odsmarkup/>.

ACKNOWLEDGMENTS

The author would like to thank Chris Barrett of SAS Institute Inc. and Nancy Brucken of i3 Statprobe for their valuable contributions to this paper.

RECOMMENDED READING

SAS Institute Inc. "Application Dispatcher". Available <http://support.sas.com/rnd/web/intrnet/dispatch.html>.

SAS Institute Inc. "The PRINT Procedure". *Base SAS® 9.1.3 Procedures Guide*. Cary, NC: SAS Institute Inc. Available http://support.sas.com/documentation/onlinedoc/91pdf/sasdoc_913/base_proc_8417.pdf.

SAS Institute Inc. "The REPORT Procedure". *Base SAS® 9.1.3 Procedures Guide*. Cary, NC: SAS Institute Inc. Available http://support.sas.com/documentation/onlinedoc/91pdf/sasdoc_913/base_proc_8417.pdf.

SAS Institute Inc. "SAS Stored Processes". Available http://support.sas.com/rnd/itech/doc9/dev_guide/stprocess/index.html.

CONTACT INFORMATION

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

Vincent DelGobbo
SAS Institute Inc.
SAS Campus Drive
Cary, NC 27513
Phone: (919) 677-8000

sasvcd@unx.SAS.com

If your registered in-house or local SAS users group would like to request this presentation as your annual SAS presentation (as a seminar, talk, or workshop) at an upcoming meeting, please submit an online User Group Request Form (support.sas.com/usergroups/namerica/lug-form.html) at least eight weeks in advance.

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.