Paper 150-2012

An Introduction to Creating Multi-Sheet Microsoft Excel Workbooks the Easy Way with SAS®

Vincent DelGobbo, SAS Institute Inc., Cary, NC

ABSTRACT

This paper explains how to use Base SAS® 9 software to create multi-sheet Excel workbooks (for Excel versions 2002 and later). You learn step-by-step techniques for quickly and easily creating attractive multi-sheet Excel workbooks that contain your SAS output using the ExcelXP ODS tagset and ODS styles. 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! Creating and delivering your workbooks on-demand and in real time using SAS server technology is discussed. Although the title is similar to previous papers by this author, this paper contains new and revised material not previously presented.

INTRODUCTION

This paper provides you with step-by-step instructions for using Base SAS 9.1.3 or later to create the Excel workbook shown in Figure 1.

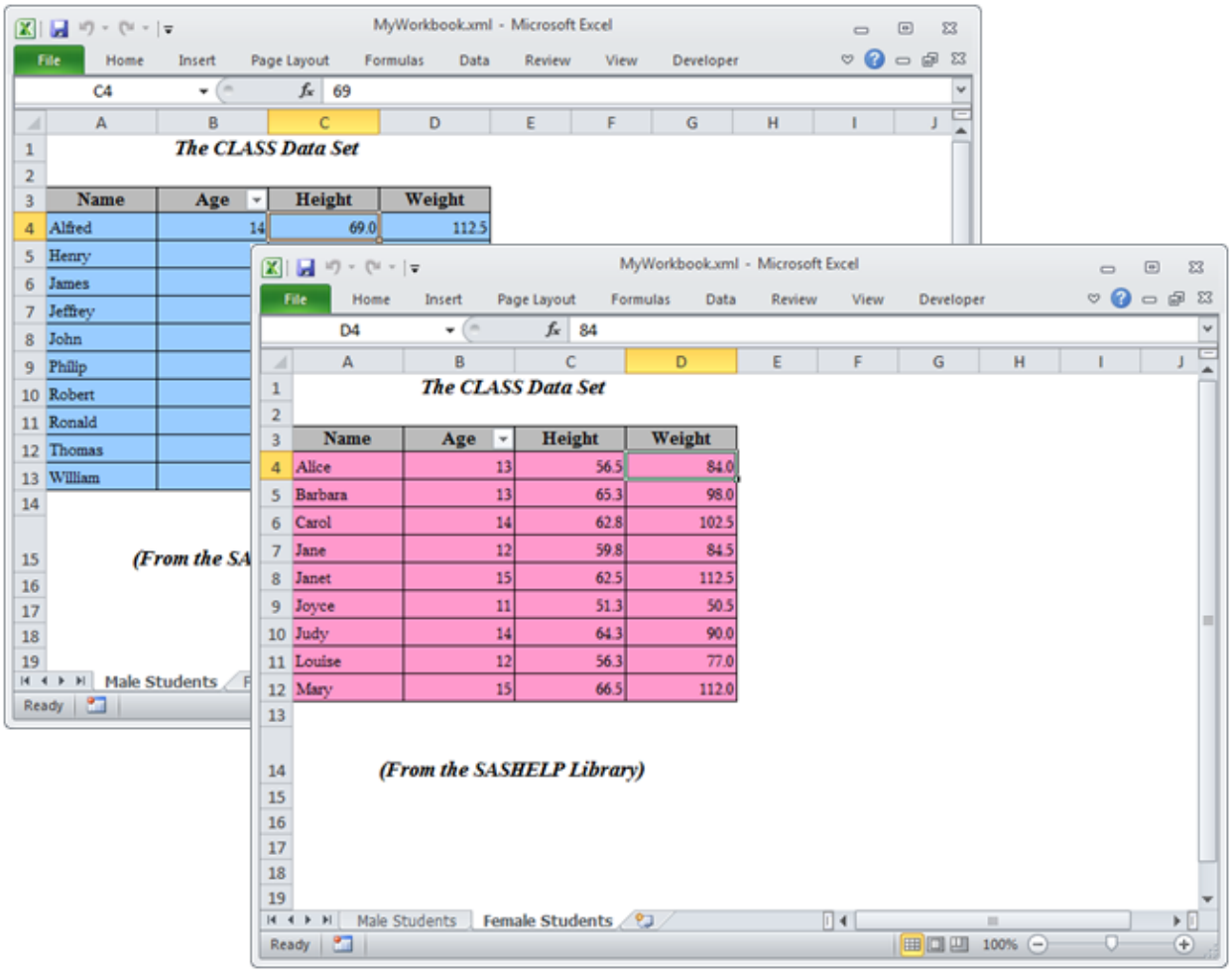

Figure 1. Multi-Sheet Excel Workbook Generated by the ExcelXP ODS Tagset

The workbook contains two worksheets containing fictional demographic data for male and female students. Different background colors are used to further distinguish the male and female data values.

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 "An Introduction to Creating Multi-Sheet Microsoft Excel Workbooks the Easy Way with SAS®".

The code in this paper was tested using SAS® 9.3 and Microsoft Excel 2010 software.

REQUIREMENTS

To use the techniques described in this paper, you must have the following software:

- Base SAS 9.1.3 Service Pack 4 or later, on *any* supported operating system (z/OS, UNIX, etc.) and hardware.
- Microsoft Excel 2002 or later (also referred to as Microsoft Excel XP).
- **An updated version of the SAS ExcelXP ODS tagset.**
  For information about obtaining an updated version of the tagset, see "The ExcelXP Tagset" section later in this paper.

LIMITATIONS

Because the ExcelXP ODS tagset creates files that conform to the Microsoft XML Spreadsheet Specification, you can create multi-sheet Excel workbooks containing the output from almost any SAS procedure. The exception is that the Microsoft XML Spreadsheet Specification does not support images, so the output from SAS/GRAPH® software procedures cannot be used (Microsoft Corporation 2001).

You can use ExcelXP tagset options with all procedure output, but ODS style overrides apply only to the PRINT, REPORT, and TABULATE procedures. Tagset options and style overrides are discussed in the sections "Understanding and Using the ExcelXP Tagset Options" and "Understanding and Using ODS Style Overrides", respectively.

Finally, you cannot use the techniques described in this paper to update existing workbooks; ODS creates the entire document on each execution, and cannot alter existing workbooks.

SAMPLE DATA

The code in this paper uses the CLASS data set, available in the SASHELP library of your SAS installation. Figure 2 presents an abbreviated view of PROC PRINT output viewed in the SAS Display Manager Output window. Note that the height and weight values are formatted to display one decimal place.

Name	Age	Height	Weight
Alfred	14	69.0	112.5
Henry	14	63.5	102.5
James	12	57.3	83.0
...			
Name	Age	Height	Weight
Alice	13	56.5	84.0
Barbara	13	65.3	98.0
Carol	14	62.8	102.5
...			

Figure 2. PROC PRINT Output Viewed in the Output Window

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 is 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 Figure 1 was created using the ExcelXP ODS tagset and the "Printer" ODS style supplied by SAS. 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.

Here are the general ODS statements needed to generate XML output that is compatible with Excel 2002 and later:

```
❶ ods _all_ close;

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

The first ODS statement (❶) closes all destinations that are open, because we want to generate only XML output for use with Excel.

The second ODS statement (❷) uses the ExcelXP tagset to generate the XML output and then store the output in a file. You should use the "xml" extension instead of "xls" or "xlsx", because Excel 2007 and 2010 display a warning if the "xml" extension is not used (Microsoft Corporation 2012b). 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 _all_ close;
ods listing;
proc template; list styles; run; quit;
```

To find the SAS code that generates sample output for the ODS styles available on your system, click the "Full Code" tab in SAS Sample 36900 (SAS Institute Inc. 2009).

The third ODS statement (❸) closes the ExcelXP destination 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 on 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 other than Excel.
- You can take advantage of Web server authentication and security models.

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

OPENING THE OUTPUT WITH EXCEL

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

1. In Excel 2002, 2003, or 2010, select **File → Open**
   In Excel 2007 select **Office Button → Open**.
2. In the **File name** field, specify the full URL for the file that 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 file or type the path and file name in the **File name** field. You can also navigate to the file using Microsoft Windows Explorer, and then double-click the file to open it with Excel.

If you encounter a "problem during load" error when you attempt to open the file with Excel, see "Diagnosing Excel Load Errors" in the appendix.

Excel reads and converts 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 (xls) format using Excel 2002, 2003, or 2010, select **File → Save As** and then, from the **Save as type** drop-down list, select **Microsoft Excel Workbook (*.xls)**. If you're using Excel 2007, click the Microsoft Office Button, and then select **Save As → Excel 97-2003 Workbook**. If you're using Excel 2007 or 2010 and want to save the document in the Microsoft Office Open XML format, choose **Excel Workbook (*.xlsx)** from the **Save as type** drop-down list.

SETTING UP THE ODS ENVIRONMENT

Our sample code uses an updated version of the ExcelXP tagset. The following statements define the location where the tagset is stored on your system:

```
❶ libname mylib 'some-directory'; * Location to store the tagset;

❷ ods path (prepend) mylib.tmplmst(update);
```

The LIBNAME statement (❶) specifies where to store the compiled tagset. Although you can temporarily store the tagset in the WORK library, it is more efficient to compile it once, and then store it in a permanent library so that you can reference it in other SAS programs.

The ODS PATH statement (❷) specifies the locations of, 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 it is first in the search path as a result of the "prepend" option. 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, compiles and stores the tagset in a file named "tmplmst.sas7bitm" in the directory that is associated with the "mylib" library.

Submit this code to display the ODS search path:

```
ods path show;
```

THE EXCELXP TAGSET

Once you have issued the appropriate ODS PATH statement, you can import an updated 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 "An Introduction to Creating Multi-Sheet Microsoft 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 local copy of this file, and then submit the following SAS code to make the tagset available:

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

You need to submit this code only once. The ExcelXP tagset is imported and stored in the directory corresponding to the "mylib" library. All of your future SAS programs can access the tagset by specifying "read" access in the LIBNAME and ODS PATH statements:

```
libname mylib 'some-directory' access=read; * Location to store the tagset;

ods path (prepend) mylib.tmplmst(read);
```

The ExcelXP tagset supports many options that control both the appearance and functionality of the Excel workbook.

To see a listing of the supported options, submit the following 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, a listing of the supported options is included in the download package for this paper.

IMPORTANT NOTE
The version of the ExcelXP tagset that was shipped with Base SAS®9 has undergone many revisions since its initial release. To take advantage of the features discussed in this paper, you must download an updated version of the tagset and install it on your system as described previously. Although this paper was tested with the ExcelXP tagset available on the SAS Presents site mentioned earlier, the latest version of the PROC TEMPLATE code to create the ExcelXP tagset can be found on the ODS Web site (SAS Institute Inc. 2012).

There is currently no notification system for tagset updates. To ensure that you have a recent version, compare the ExcelXP tagset version, displayed in the SAS log whenever the tagset is used, to the version available on the ODS Web site.

A BRIEF ANATOMY OF ODS STYLES

ODS styles control all aspects of the appearance of the output, and Base SAS software ships with more than 50 different styles. A style contains *style elements,* each of which controls a particular part of the output. For example, a style element named "header" controls the appearance of column headings. Style elements consist of collections of *style attributes*, such as the background color and font size.

Use the ODS tagset named "style_popup" when you need to determine the attributes of style elements. The tagset creates an HTML file that, when viewed using the SAS Results window or the Microsoft Internet Explorer Web browser, displays style element information in popup windows (SAS Institute Inc. 2011d).

For example, to see the style attributes of the "header" style element of the "Printer" style, follow these steps:

1. Submit this SAS code:

   ```
   ods results;
   ods _all_ close;
   ods tagsets.style_popup path='output-directory' file='temp.htm' style=Printer;

    *  Your SAS procedure code here;

   ods tagsets.style_popup close;
   ```

2. View the HTML output using the SAS Results window or Microsoft Internet Explorer.

3. Click on a column heading to display a popup window containing the style element name ("header") and style attribute information for that cell (Figure 3).

4. Repeat step 3 for other areas of interest, for example, data cells.

To display the full definition of a style, submit this code:

```
proc template;
  source styles.style-name;
run; quit;
```

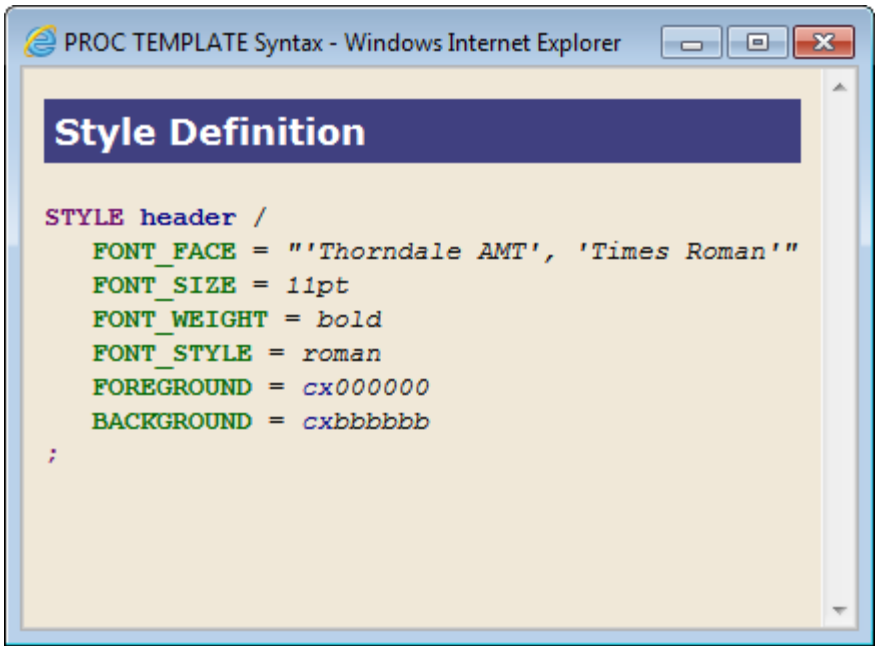

Figure 3. Style Attributes of the "header" Style Element

USING ODS TO CREATE THE MULTI-SHEET EXCEL WORKBOOK

By default, the ExcelXP tagset creates a new worksheet when a SAS procedure creates new tabular output. PROC PRINT is run twice and, because each execution creates one table as output, the workbook contains 2 worksheets.

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

```
ods _all_ close;

❶ ods tagsets.ExcelXP path='output-directory' file='MyWorkbook.xml'
    style=Printer;

  title 'The CLASS Data Set';
  footnote '(From the SASHELP Library)';

❷ proc print data=sashelp.class noobs;
    where (sex eq 'M');
    var name age height weight;
  run; quit;

❸ proc print data=sashelp.class noobs;
    where (sex eq 'F');
    var name age height weight;
  run; quit;

❹ ods tagsets.ExcelXP close;
```

As you can see in the ODS statement (❶), the ExcelXP tagset generates the output, and the "Printer" style controls the appearance of the output. The PRINT procedure is run once to create a worksheet containing data for the male students, and then again to create a worksheet for female students (❷ and ❸). The last ODS statement (❹) closes the ExcelXP destination and releases the XML file so that it can be opened with Excel.

Figure 4 displays the results of executing the basic SAS code, and opening the resulting "MyWorkbook.xml" file with Excel.

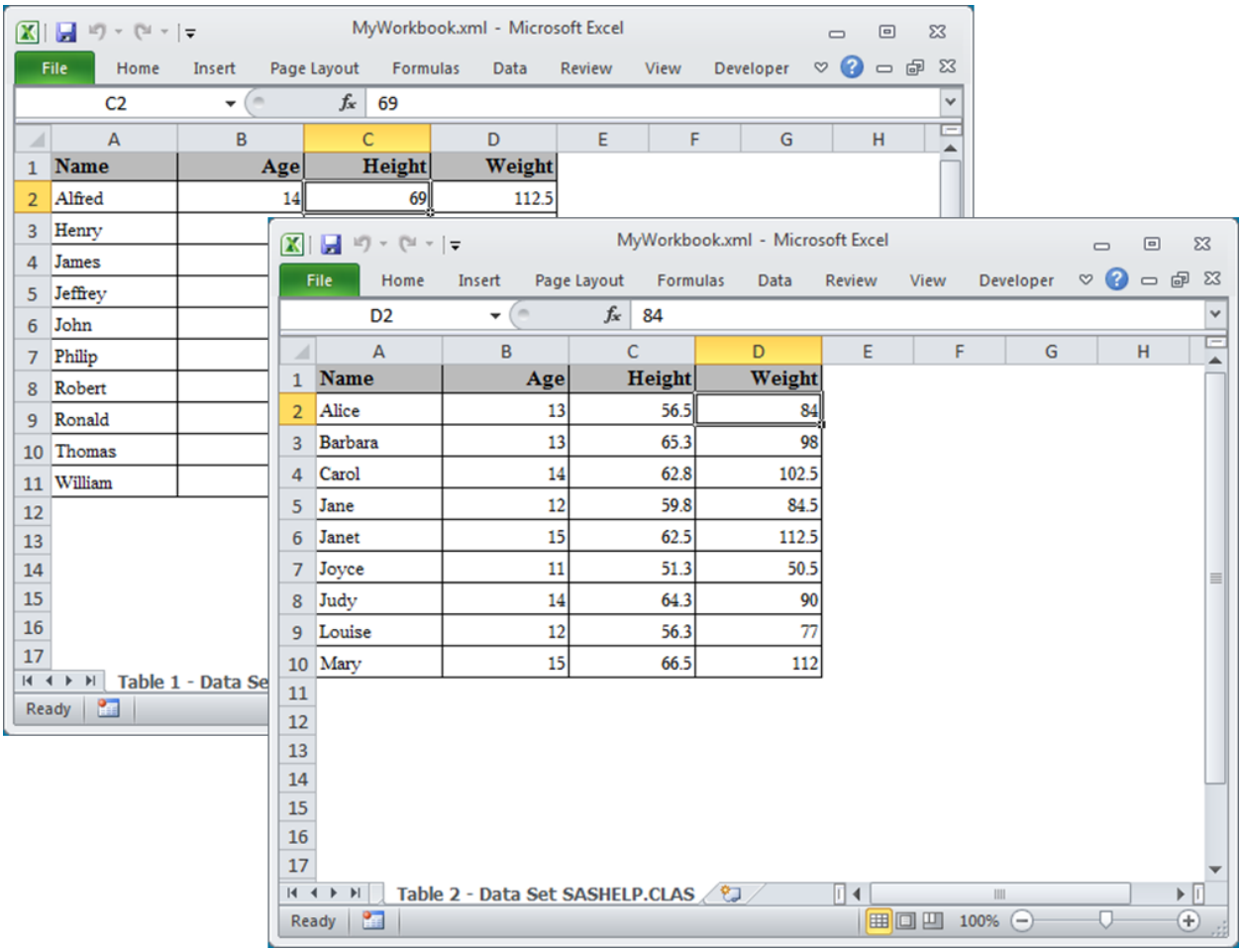

Figure 4. Initial ExcelXP ODS Tagset-Generated Workbook

Notice that Figure 4 does not match Figure 1. The following problems are exhibited in Figure 4:

1. Unattractive, default worksheet names are used.
2. Title and footnote text is missing.
3. Column B does not contain an Excel AutoFilter.
4. The column heading text is not centered.
5. The background color for all data cells is white.
6. Not all of the data values for height and weight are displayed with 1 decimal place.

We can now change the basic SAS code to correct these problems. The complete SAS code used to create the workbook shown in Figure 1 is listed in the section "The Final SAS Code".

UNDERSTANDING AND USING THE EXCELXP TAGSET OPTIONS

As mentioned above, the ExcelXP tagset supports many options that control both the appearance and functionality of the Excel workbook. Many of these tagset options are simply tied directly into existing Excel options or features. Tagset options are specified in an ODS statement using the OPTIONS keyword:

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

Note that the value that you specify for a tagset option remains in effect until the ExcelXP destination is closed, or the option is set to another value. Because 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');
  *  Some SAS procedure code here;
ods tagsets.ExcelXP options(option-name='default-value');
  *  Other SAS procedure code here;
```

When specifying *additional* ODS statements as shown above, do not specify the `file`, `style`, or any other keyword or option that is supported by ODS. Those options should be specified only on the initial ODS statement.

Tagset options are supported for **all** SAS procedure output, unlike ODS style overrides, which are supported only by the PRINT, REPORT, and TABULATE procedures.

SPECIFYING WORKSHEET NAMES

ODS generates a unique name for each worksheet, as required by Excel. Figure 4 shows the worksheet names that result from running the initial SAS code. There are, however, several tagset options that you can use to alter the names of the worksheets.

Use the SHEET_NAME option to explicitly specify a worksheet name. Recall that tagset options remain in effect until the ExcelXP destination is closed. We specify the option twice because we want a different name for each worksheet.

```
ods tagsets.ExcelXP path='...' file='...' style=Printer;

title '...'; footnote '...';

ods tagsets.ExcelXP options(sheet_name='Male Students');

proc print data=sashelp.class noobs;
  where (sex eq 'M');
  var name age height weight;
run; quit;

ods tagsets.ExcelXP options(sheet_name='Female Students');

proc print data=sashelp.class noobs;
  where (sex eq 'F');
  var name age height weight;
run; quit;

ods tagsets.ExcelXP close;
```

Refer to this author's earlier papers for techniques used to automatically generate worksheet names from BY group values (DelGobbo 2009, 2010, 2011)

INCLUDING TITLE AND FOOTNOTE TEXT IN THE WORKSHEET BODY

By default, SAS titles and footnotes appear as Excel print headers and print footers, respectively, which are displayed when the Excel document is printed. You can confirm this by viewing the Excel "Header/Footer" tab in the "Page Setup" dialog box, shown in Figure 5.

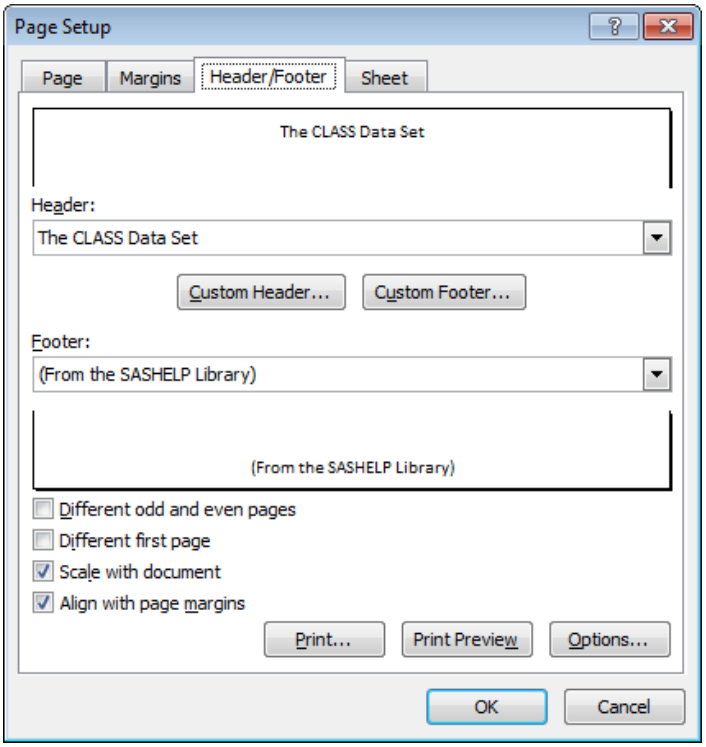

Figure 5. Excel Page Setup Dialog Box Showing Title and Footnote Text

To include title and footnote text on-screen, in the worksheet body, use the EMBEDDED_TITLES and EMBEDDED_FOOTNOTES options:

```
*  Set some "global" tagset options that affect all worksheets;

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

ods tagsets.ExcelXP options(sheet_name='Male Students');

proc print data=sashelp.class noobs; where (sex eq 'M'); ... ; run; quit;

ods tagsets.ExcelXP options(sheet_name='Female Students');

proc print data=sashelp.class noobs; where (sex eq 'F'); ... ; run; quit;
```

Because tagset options remain in effect until their value is changed or the destination is closed, the EMBEDDED_TITLES and EMBEDDED_FOOTNOTES options affect both worksheets.

SPECIFYING EXCEL PRINT HEADERS AND FOOTERS

Although the name of each worksheet includes the gender presented in the data, this information is not shown when the workbook is printed. You can add Excel print headers and footers to display this information. Excel allows you to define custom print headers and footers from the "Header/Footer" tab in the "Page Setup" dialog box (Figure 5). To use Excel to manually insert the worksheet name in the header, click the "Custom Header" button to open the "Header" dialog box shown in Figure 6. Then click the "Insert Sheet Name" button.

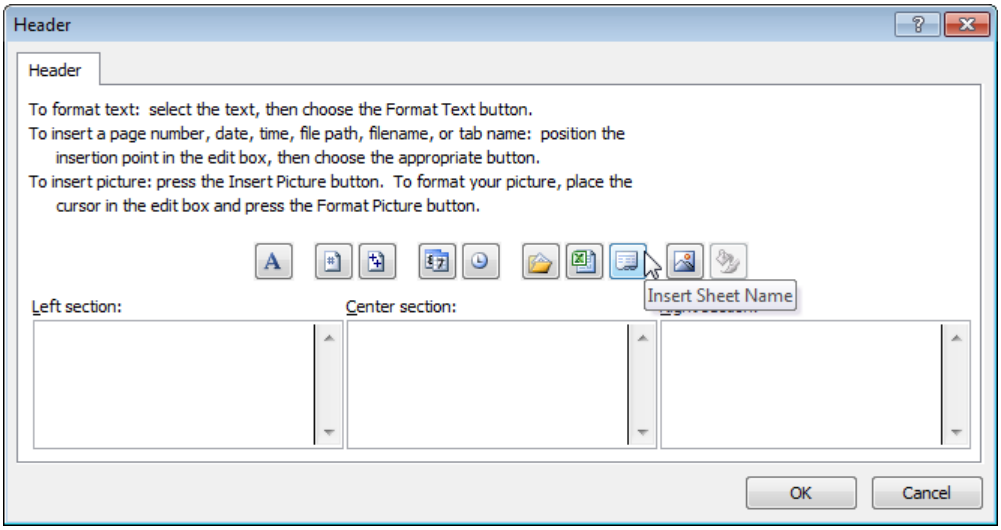

Figure 6. Excel Header Dialog Box

You can also control the header and footer from SAS code using the PRINT_HEADER and PRINT_FOOTER tagset options. In addition to plain text, you can specify Excel control sequences (Table 1) that provide the same functionality as the buttons shown in Figure 6. For example, use `&A` to insert the worksheet name into the header.

Control Sequence	Function
&C	Center text
&A	Insert sheet name
&R	Right-justify text
&P	Insert page number
&N	Insert number of pages
&D	Insert date
&T	Insert time
&F	Insert file name
&Z	Insert file path
&L	Left-justify text
&U*your-text*&U	Single underline *your-text*
&E*your-text*&E	Double underline *your-text*
&S*your-text*&S	Strikethrough *your-text*
&X*your-text* &X	Superscript *your-text*
&Y*your-text* &Y	Subscript *your-text*
&&	Insert literal &

Table 1. Excel Print Header and Footer Control Sequences

We use plain text and several control sequences to add useful information to our header and footer:

```
*  Set some "global" tagset options that affect all worksheets;

ods tagsets.ExcelXP options(embedded_titles='yes'
                            embedded_footnotes='yes'
                            print_header='&C&A&RPage &P of &N'
                            print_footer='&RPrinted &D at &T');
```

The resulting print headers and footers are shown in Figure 7.

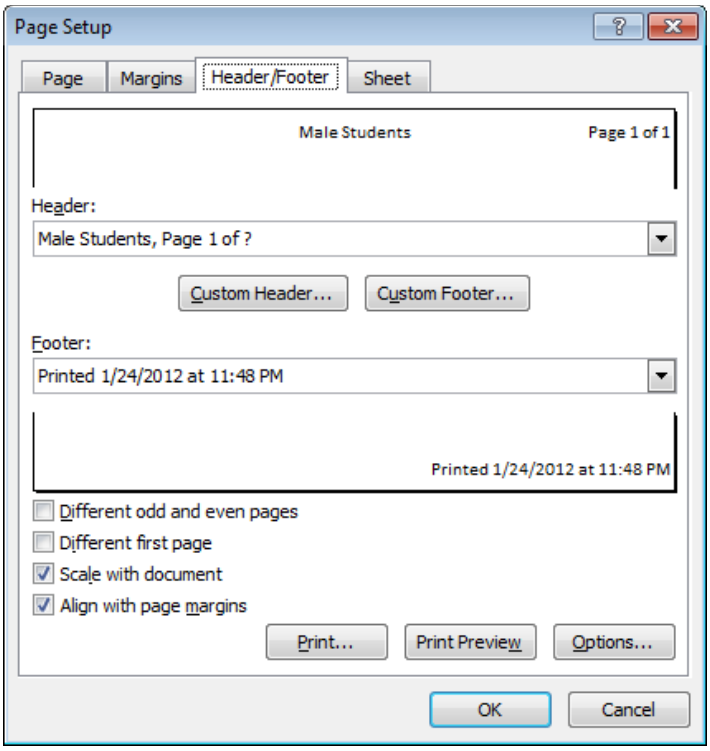

Figure 7. Excel Page Setup Dialog Box Showing Print Header and Footer Text

ADDING AN EXCEL AUTOFILTER TO THE AGE COLUMN

An Excel AutoFilter enables you to filter, or subset, the data that is being displayed in a worksheet. A column of data containing an AutoFilter is indicated by an arrow button in the header cell of that column. For example, the "Age" column in Figure 1 contains an AutoFilter.

Suppose that you want to view only records for 15-year old students. You click the AutoFilter button on the "Age" column, and then select "15", as illustrated in Figure 8. All records in the worksheet that do not have "15" as a value in the "Age" column are hidden. The data is still present in the worksheet, but it is not displayed due to the filtering.

Use the AUTOFILTER tagset option to add an Excel AutoFilter to the second column of your workbook:

```
*  Set some "global" tagset options that affect all worksheets;

ods tagsets.ExcelXP options(embedded_titles='yes'
                            embedded_footnotes='yes'
                            print_header='&C&A&RPage &P of &N'
                            print_footer='&RPrinted &D at &T'
                            autofilter='2');
```

You can control which columns have AutoFilters by specifying 'all', 'none', or a range of contiguous columns ('2-4', for example).

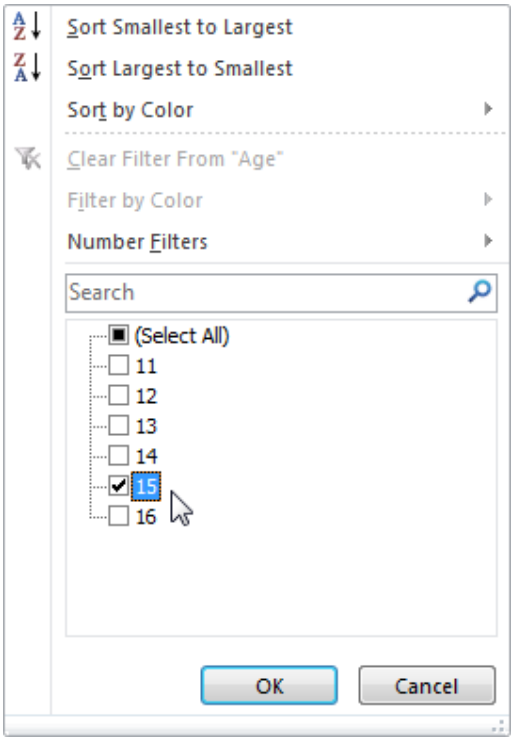

Figure 8. Column 2 AutoFilter Menu

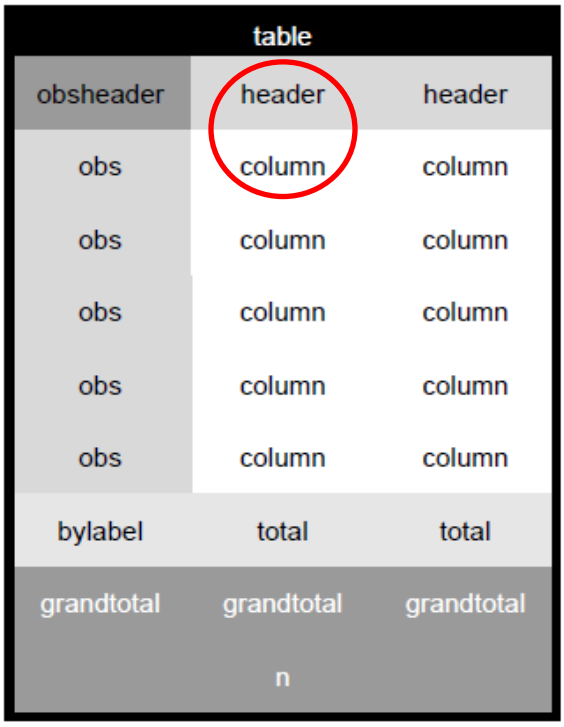

Figure 9. Style Locations for the PRINT Procedure

UNDERSTANDING AND USING ODS STYLE OVERRIDES

You can alter the attributes or style elements used by specific parts of your SAS output by using style overrides. These specific parts of your SAS output are called *locations*. Figure 9 shows the locations that are pertinent to the PRINT procedure output (SAS Institute Inc. 2008). The "Header" location controls the appearance of column headings and the "Column" location controls the appearance of data cells.

Style overrides are supported by the PRINT, REPORT, and TABULATE procedures, and can be specified in several ways, the two most common formats being:

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

The first format (❶) uses individual style attributes defined inline. For example, the following PROC PRINT code alters 3 attributes of the "Header" location for the "myvar" variable:

```
var myvar / style(Header)=[background=yellow font_size=10pt just=left];
```

While this is the most commonly used format, it has some disadvantages. To use the same style override for different variables, you must apply it in multiple places, making your SAS code harder to read and maintain. And, if you want to use the style overrides in other SAS programs, you must copy the list of attribute name/value pairs to the new code. Style overrides of this type should be used sparingly.

The second format (❷) overcomes these problems by referencing a style element. Using this format involves creating a new style element, setting the style attributes within the element, and then using the *style element* name in your style override. This results in code that is easier to read, maintain, and re-use. Earlier papers by this author provide a detailed discussion of this topic (DelGobbo 2008, 2009, 2010, 2011).

Refer to the ODS documentation for a full listing of style attributes (SAS Institute Inc. 2011b).

CENTERING THE HEADING TEXT

Although we could use the inline format of the STYLE option to center the column heading text by specifying the style override for each variable, it is easier to use a single style override on the procedure statement to change the justification of *all* column headings:

```
ods tagsets.ExcelXP options(sheet_name='Male Students');

proc print data=sashelp.class noobs style(Header)=[just=center];
  where (sex eq 'M');
  var name age height weight;
run; quit;

ods tagsets.ExcelXP options(sheet_name='Female Students');

proc print data=sashelp.class noobs style(Header)=[just=center];
  where (sex eq 'F');
  var name age height weight;
run; quit;
```

CHANGING BACKGROUND COLORS

Excel versions 2002 and 2003 have a limited color palette. Figure 10 shows the default colors. If you plan to view your workbooks using one of these versions of Excel, choose colors that are listed in the default palette. Otherwise Excel maps the unsupported color to a color that is supported. The color values used in the style overrides are indicated by a star (✸) in Figure 10.

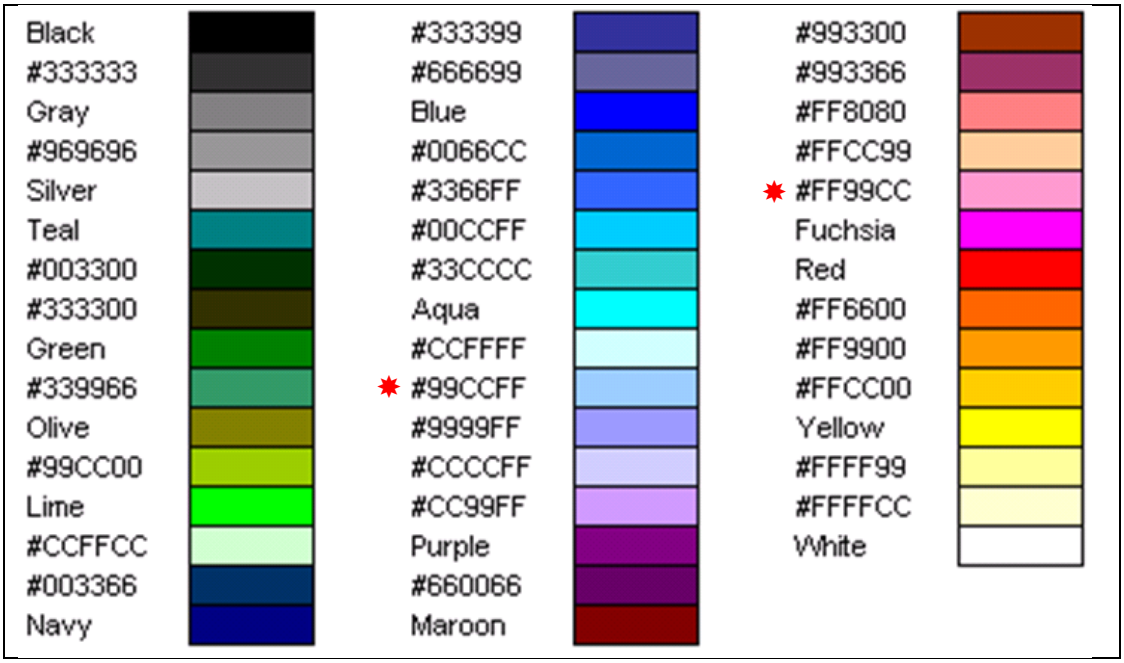

Figure 10. Default Colors Supported by Excel Versions 2002 and 2003

However, specifying colors is not foolproof. Because each Excel user can customize the color palette, colors are not guaranteed to exist in all instances of Excel.

Note that Excel versions 2007 and 2010 do not have this restricted color palette.

To apply one ODS style override to the "name" and "age" columns, and later, a different style override the "height" and "weight" columns, we need to split the single VAR statements into 2 separate statements. Style overrides are applied to the "Column" location of each variable to change the background color of the data cells.

```
ods tagsets.ExcelXP options(sheet_name='Male Students');

proc print data=sashelp.class noobs style(Header)=[just=center];
  where (sex eq 'M');
  var name age      / style(Column)=[background=#99ccff];
  var height weight / style(Column)=[background=#99ccff];
run; quit;

ods tagsets.ExcelXP options(sheet_name='Female Students');

proc print data=sashelp.class noobs style(Header)=[just=center];
  where (sex eq 'F');
  var name age      / style(Column)=[background=#ff99cc];
  var height weight / style(Column)=[background=#ff99cc];
run; quit;
```

SPECIFYING AN EXCEL NUMBER FORMAT FOR HEIGHT AND WEIGHT

Our Excel workbook now closely resembles Figure 1, except that one decimal place is not always used for the height and weight data. You could sometimes have success using SAS formats to control the Excel display values, but it is usually better to instead use Excel formats.

The pound sign (#) in an Excel format is used to represent a numeric digit, excluding insignificant zeros, and a zero (0) displays a numeric digit including insignificant zeros. Use zeros in Excel formats when you want to retain leading or trailing zeros. Figure 11 shows the general structure of Excel formats (Microsoft Corporation 2012a).

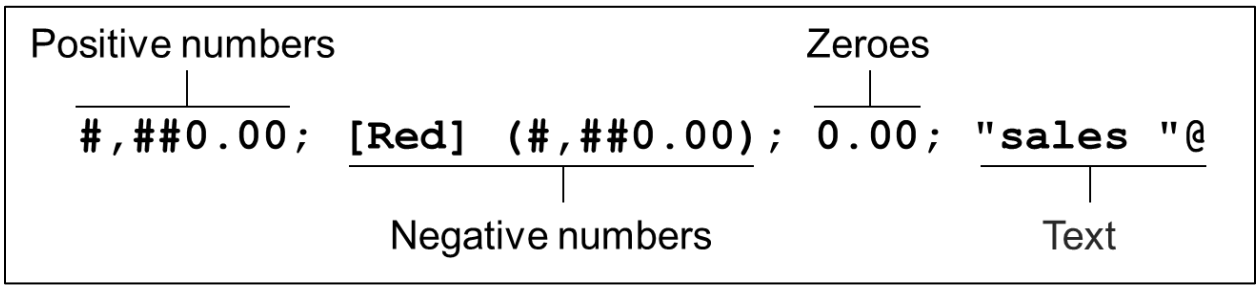

Figure 11. Structure of Excel Number Formats

Table 2 shows the results of applying the Excel format shown in Figure 11.

Raw Value	Formatted Value	Comment
.5	0.50	Leading & Trailing Zeros
5	5.00	Leading & Trailing Zeros
123	123.00	Trailing Zeros
1234	1,234.00	Trailing Zeros, Thousands Separator
-1234	(1,234.00)	Leading & Trailing Zeros, Thousands Separator, Red ()
0	0.00	Special Zero Handling
Data	sales data	Special Text Handling

Table 2. Results of Applying Excel Format Shown in Figure 11

Because we are working only with positive values, we use the Excel format `#.0` with the ODS "tagattr" attribute to specify an Excel format. Be sure to quote the entire attribute value and include the "format:" keyword:

```
ods tagsets.ExcelXP options(sheet_name='Male Students');

proc print data=sashelp.class noobs style(Header)=[just=center];
  where (sex eq 'M');
  var name age      / style(Column)=[background=#99ccff];
  var height weight / style(Column)=[background=#99ccff tagattr='format:#.0'];
run; quit;

ods tagsets.ExcelXP options(sheet_name='Female Students');

proc print data=sashelp.class noobs style(Header)=[just=center];
  where (sex eq 'F');
  var name age      / style(Column)=[background=#ff99cc];
  var height weight / style(Column)=[background=#ff99cc tagattr='format:#.0'];
run; quit;
```

With all the code modifications in place, the resulting workbook matches the output shown in Figure 1.

THE FINAL SAS CODE

The final SAS code to create the output of Figure 1 follows:

```
ods _all_ close;

ods tagsets.ExcelXP path='output-directory' file='MyWorkbook.xml'
  style=Printer;

title 'The CLASS Data Set';
footnote '(From the SASHELP Library)';

*  Set some "global" tagset options that affect all worksheets;

ods tagsets.ExcelXP options(embedded_titles='yes'
                            embedded_footnotes='yes'
                            print_header='&C&A&RPage &P of &N'
                            print_footer='&RPrinted &D at &T'
                            autofilter='2');

ods tagsets.ExcelXP options(sheet_name='Male Students');

proc print data=sashelp.class noobs style(Header)=[just=center];
  where (sex eq 'M');
  var name age      / style(Column)=[background=#99ccff];
  var height weight / style(Column)=[background=#99ccff tagattr='format:#.0'];
run; quit;

ods tagsets.ExcelXP options(sheet_name='Female Students');

proc print data=sashelp.class noobs style(Header)=[just=center];
  where (sex eq 'F');
  var name age      / style(Column)=[background=#ff99cc];
  var height weight / style(Column)=[background=#ff99cc tagattr='format:#.0'];
run; quit;

ods tagsets.ExcelXP close;
```

SAS SERVER TECHNOLOGY

You can incorporate dynamically-generated SAS output into Excel using the Application Dispatcher or the SAS® Stored Process Server. The Application Dispatcher is part of SAS/IntrNet® software. The SAS Stored Process Server is available starting with SAS® 9 as part of SAS® Integration Technologies, and is included with server offerings that leverage the SAS Business Analytics infrastructure (for example, SAS® BI Server and SAS® Enterprise BI Server).

These products enable you to execute SAS programs from a Web browser or any other client that can open an HTTP connection to the Application Dispatcher or the SAS Stored Process Server. Both of these products can run on any platform where SAS is licensed. SAS software does not need to be installed on the client machine.

The SAS programs that you execute from the browser can contain any combination of DATA Step, procedure, macro, or SCL code. Thus, all of the code that has been shown up to this point can be executed by both the Application Dispatcher and the SAS Stored Process Server.

Program execution is typically initiated by accessing a URL that points to the SAS server program. Parameters 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 12 shows a Web page that can deliver SAS output directly to Excel, using a Web browser as the client.

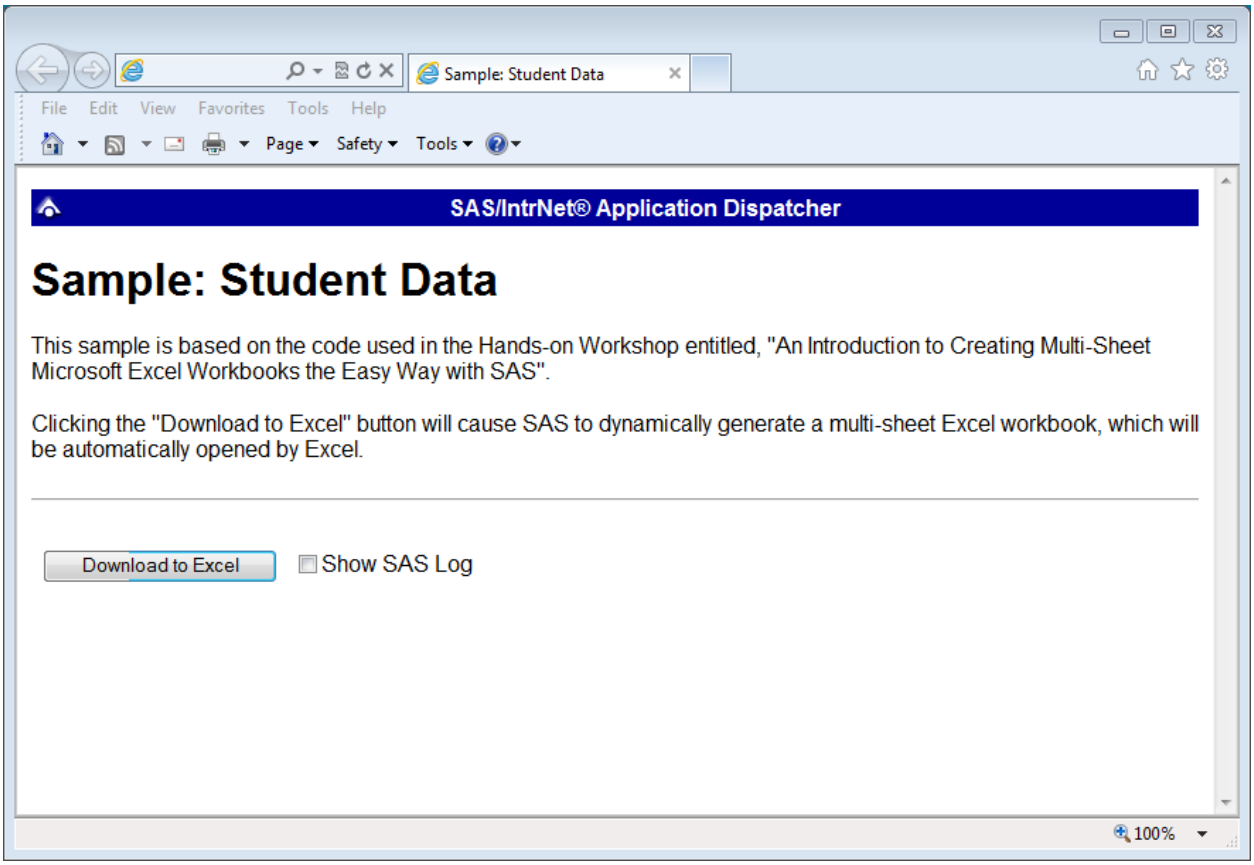

Figure 12. Web Page to Drive a SAS/IntrNet Application

Clicking **Download to Excel** executes a slightly modified version of the SAS code that we have been working on. 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=
"StudentData.xml"));   * Ignore line wrapping;

ods listing close;
```

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

The first APPSRV_HEADER function sets a 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 MIME header that causes the file name to be displayed in the "File Download" dialog box. As you can see in Figure 13, the file name appears as "StudentData.xml". This header might cause problems with some versions of Excel, so be sure to test your applications before deploying them in a production environment. This statement is optional.

Figure 13. File Download Dialog Box

The reserved fileref _WEBOUT is automatically defined 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 to an external disk file.

When you click the **Download to Excel** button on the Web page and are presented with the "File Download" dialog box (Figure 13), you can click **Open** to immediately 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, see the SAS/IntrNet Application Dispatcher and SAS Stored Process documentation (SAS Institute Inc. 2011a, 2011c), as well as this author's earlier papers (DelGobbo 2002, 2003, 2004).

CONCLUSION

The SAS® 9 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 a tagset that complies with the Microsoft XML Spreadsheet Specification, you can customize the output to achieve your design goals.

APPENDIX

DIAGNOSING EXCEL LOAD ERRORS

When you open a malformed XML file using Excel, you are presented with an error dialog box containing a message like this:

```
This file cannot be opened because of errors. Errors are listed in: C:\Documents
and Settings\userid\Local Settings\Temporary Internet
Files\Content.MSO\404A54C0.log.
```

On most systems, the log file created by Excel is stored in a hidden directory, so you cannot navigate to it using Microsoft Windows Explorer. You can, however, navigate to the directory from a command prompt (cmd.exe) by entering this text and pressing the ENTER key (ignore the line wrapping):

```
%SYSTEMROOT%\explorer.exe /e, "%USERPROFILE%\Local Settings\Temporary Internet
Files\Content.MSO"
```

Malformed XML is sometimes created because of an invalid value specified for the "tagattr" attribute. For example, Excel formats have limited color support (Microsoft 2012a), and specifying an unsupported color causes the problem:

```
var height / style(Column)={tagattr='format:[Orange]#.0'};
```

The Excel log file contains the following information, indicating that there is an invalid value for the "Format" attribute of the "NumberFormat" tag:

```
XML ERROR in Style
REASON:    Bad Value
FILE:      C:\HOW\DelGobbo\MyWorkbook.xml
GROUP:     Style
TAG:       NumberFormat
ATTRIB:    Format
VALUE:     [Orange]#.0
```

Misspelling a style element name or specifying a style element that is not part of the style definition can cause malformed XML. For example, if the "data_bold" style element exists in the user-defined "MyStyle" style, but you misspell it as "date_bold":

```
ods tagsets.ExcelXP file='MyWorkbook.xml' style=MyStyle ...;

...

define MyVar / display order style(Column)=date_bold;

...

ods tagsets.ExcelXP close;
```

Here is an example of the second case, where the "data_bold" style element is spelled correctly, but it is not defined in the "Printer" style that is supplied by SAS. The style element exists in the user-defined "XLPrinter" style, but you forgot to change the style name from "Printer" to "XLPrinter":

```
ods tagsets.ExcelXP file='MyWorkbook.xml' style=Printer ...;

...

define MyVar / display order style(Column)=data_bold;

...

ods tagsets.ExcelXP close;
```

In both cases, the Excel log file indicates that "unknown" is an invalid value for the "StyleID" attribute of the "Cell" tag:

```
XML ERROR in Table
REASON: Bad Value
FILE: C:\HOW\DelGobbo\MyWorkbook.xml
GROUP:  Row
TAG:  Cell
ATTRIB: StyleID
VALUE:  unknown
```

If you open the file "MyWorkbook.xml" using a text editor, you will see instances of the incorrect value being used:

```
<Cell ss:StyleID="unknown" ... >
```

VISUAL BASIC CODE TO CONVERT XML TO NATIVE EXCEL FORMATS

The author is developing a Visual Basic script that converts ExcelXP-generated files to native Excel xls or xlsx formats. Contact the author if you would like a copy of this experimental code.

REFERENCES

DelGobbo, Vincent. 2002. "Techniques for SAS® Enabling Microsoft® Office in a Cross-Platform Environment". *Proceedings of the Twenty-Seventh Annual SAS Users Group International Conference*. Cary, NC: SAS Institute Inc. Available at http://www2.sas.com/proceedings/sugi27/p174-27.pdf.

DelGobbo, Vincent. 2003. "A Beginner's Guide to Incorporating SAS® Output in Microsoft® Office Applications". *Proceedings of the Twenty-Eighth Annual SAS Users Group International Conference*. Cary, NC: SAS Institute Inc. Available at http://www2.sas.com/proceedings/sugi28/052-28.pdf.

DelGobbo, Vincent. 2004. "From SAS® to Excel via XML". Available at http://support.sas.com/rnd/papers/sugi29/ExcelXML.pdf.

DelGobbo, Vincent. 2008. "Tips and Tricks for Creating Multi-Sheet Microsoft Excel Workbooks the Easy Way with SAS®". *Proceedings of the SAS Global Forum 2008 Conference*. Cary, NC: SAS Institute Inc. Available at http://www2.sas.com/proceedings/forum2008/192-2008.pdf.

DelGobbo, Vincent. 2009. "More Tips and Tricks for Creating Multi-Sheet Microsoft Excel Workbooks the Easy Way with SAS®". *Proceedings of the SAS Global Forum 2009 Conference*. Cary, NC: SAS Institute Inc. Available at http://support.sas.com/resources/papers/proceedings09/152-2009.pdf.

DelGobbo, Vincent. 2010. "Traffic Lighting Your Multi-Sheet Microsoft Excel Workbooks the Easy Way with SAS®". *Proceedings of the SAS Global Forum 2010 Conference*. Cary, NC: SAS Institute Inc. Available at http://support.sas.com/resources/papers/proceedings10/153-2010.pdf.

DelGobbo, Vincent. 2011. "Creating Stylish Multi-Sheet Microsoft Excel Workbooks the Easy Way with SAS®". *Proceedings of the SAS Global Forum 2011 Conference*. Cary, NC: SAS Institute Inc. Available at http://support.sas.com/resources/papers/proceedings11/170-2011.pdf.

Microsoft Corporation. 2001. "XML Spreadsheet Reference". Available at http://msdn2.microsoft.com/en-us/library/aa140066(office.10).aspx.

Microsoft Corporation. 2012a. "Create of delete a custom number format". Available at http://office.microsoft.com/assistance/hfws.aspx?AssetID=HP051995001033.

Microsoft Corporation. 2012b. "When you open a file in Excel 2007, you receive a warning that the file format differs from the format that the file name extension specifies". Available at http://support.microsoft.com/kb/948615.

SAS Institute Inc. 2008. "SAS® 9 Reporting Procedure Styles Tip Sheet". Available at http://support.sas.com/rnd/base/ods/scratch/reporting-styles-tips.pdf.

SAS Institute Inc. 2009. "Sample 36900: Instructions for viewing all of the style templates that are shipped with the SAS® System". Available at http://support.sas.com/kb/36/900.html.

SAS Institute Inc. 2011a. *SAS/IntrNet® 9.3: Application Dispatcher.* Cary, NC: SAS Institute Inc. Available at http://support.sas.com/documentation/cdl/en/dispatch/62562/HTML/default/viewer.htm#p06h82ux8glu1pn16k9dxw8tjpyz.htm

SAS Institute Inc. 2011b. *SAS® 9.3 Output Delivery System User: User's Guide.* Cary, NC: SAS Institute Inc. Available at http://support.sas.com/documentation/cdl/en/odsug/62755/HTML/default/viewer.htm#p15bfjqgegjal0n1j3ze57yaqljr.htm.

SAS Institute Inc. 2011c. *SAS® 9.3 Stored Processes: Developer's Guide.* Cary, NC: SAS Institute Inc. Available at http://support.sas.com/documentation/cdl/en/stpug/62758/HTML/default/viewer.htm#titlepage.htm

SAS Institute Inc. 2011d. "Uses of the Results Window". *SAS® 9.3 Language Reference: Concepts*. Cary, NC: SAS Institute Inc. Available at http://support.sas.com/documentation/cdl/en/lrcon/62753/HTML/default/viewer.htm#n1039zk8bk9aton1fmbm7z2wji3k.htm.

SAS Institute Inc. 2012. "ODS MARKUP Resources".
Available at http://support.sas.com/rnd/base/topics/odsmarkup/.

ACKNOWLEDGEMENTS

The author would like to thank Chris Barrett of SAS Institute Inc. for his valuable contributions to this paper.

RECOMMENDED READING

DelGobbo, Vincent. 2012. "Vince DelGobbo's ExcelXP Tagset Paper Index".
Available at http://www.sas.com/events/cm/867226/ExcelXPPaperIndex.pdf.

SAS Institute Inc. 2012. "Quick Reference for the TAGSETS.EXCELXP Tagset".
Available at http://support.sas.com/rnd/base/ods/odsmarkup/excelxp_help.html

CONTACT INFORMATION

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

Vincent DelGobbo
SAS Institute Inc.
SAS Campus Drive
Cary, NC 27513

sasvcd@SAS.com

http://www.sas.com/reg/gen/corp/867226?page=Resources

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 (from support.sas.com/sasusersupport/usergroups/support) 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.