Paper SAS2174-2018

Insights from a SAS Technical Support Guy: A Deep Dive into the SAS® ODS Excel Destination

Chevell Parker, SAS Institute Inc.

ABSTRACT

SAS is a world leader in data analytics while Microsoft Excel, with over 30 million active users, is a leader when it comes to spreadsheet packages. Excel spreadsheets are heavily used for calculations, information organization, statistical analysis, and graphics. SAS can leverage the power of its world-class analytics and reporting capabilities to produce stylistic and highly functional Excel spreadsheets using the Excel destination in the SAS® Output Delivery (ODS) System.

This paper, relevant to anyone who uses Microsoft Excel, offers insights into the depths of the ODS Excel destination by illustrating how you can customize styles in Microsoft Excel worksheets, and it discusses common layout and reporting questions (including limitations). In addition, the discussion covers useful applications for automating and executing Excel worksheets. After diving deep into this discussion about the ODS Excel destination, you should understand the behavior and capabilities of the destination so that you can create aesthetic and effective Excel worksheets.

INTRODUCTION

The Excel destination is a very popular ODS destination that is different from all the other destinations in that it enables you to generate a fully functional Excel worksheet directly from SAS. With over 50 options in the destination, you can customize your worksheets to handle common tasks such as freezing headers adding filters, and incorporating robust styles. The ODS Excel destination also allows multiple worksheets per workbook; the ability to add tables, graphics, and customized text; and the ability to use SAS formats on fields to convert them to Excel formats. This paper also addresses common layout and reporting questions that, combined with all the functionality mentioned above, enables you to get the most out of using the ODS Excel destination.

CREATING EFFECTIVE STYLES AND EXCEL WORKSHEETS

One of the first things you notice about a worksheet is its style and presentation. A *style* is anything (not limited to fonts, colors, and so on) that enhances the presentation of the output. Not only does the style make the worksheet visually appealing, it also helps you to navigate the worksheet effectively. To create an effective worksheet, you need to ensure that you apply the correct style for the worksheet. The ODS Excel destination generates effective, styled output for Excel worksheets, by default. Using a combination of styles and ODS Excel options, you can effectively create worksheets that enable the users of those worksheets to have a positive experience in navigating the worksheets and understanding the data on the worksheets. The next section provides best practices to follow when you generate worksheets.

BEST PRACTICES TIPS

The following best practices are explained in more detail later in this paper:

- Start output in column B2, which adds whitespace to the worksheet. Whitespace creates balance, acts as a separator between the elements on a page.
- Use subtle shading of alternate table rows to help the eye to follow information across a worksheet page.
- Choose a sans-serif font such as Arial or Calibri (a thinner version of Arial, such as Arial Narrow, renders better on mobile devices). Sans-serif fonts provide better legibility.
- Adhere to using dark text on a light background for easier reading and better printing.
- Use a larger font size for headings and subheadings, and keep the font size of other text large enough to read.
- Manipulate the height and width of cells to prevent your spreadsheet from looking too cramped.

(list continued)

- Use gridlines and borders, as necessary, for readability.
- Use color to enhance your worksheet, but use it sparingly. Too much color can have the opposite effect of what you want.
- Consider adding a background image that is related to the date or a company logo. Logos and background images create a more professional and polished appearance.

APPLYING STYLES TO MICROSOFT EXCEL WORKSHEETS

You can add style to Excel worksheets by using various methods. For example, you can add global styles by using either the SAS® TEMPLATE procedure, the STYLE= override option in various SAS procedures, or cascading style sheets (CSS).

Adding global styles by using a CSS is a more advanced method of adding styles that is beneficial in that you have the most control of the style. Using the STYLE= override option for procedures such as PRINT, REPORT and TABULATE enables you to add style within a table while inline styles (styles that are included in a code line) modify text strings such as labels and titles.

Some of the most popular modifications in worksheets are changes to fonts and font properties (for example, bold, color, and shading), color, and borders in tables and graphics.

The following example demonstrates basic modifications to tables and graphics within worksheets. This PROC TEMPLATE step makes some basic modifications to a table and a graph.

Example 1

```
proc template;
   define style styles.excel_update;
      parent=styles.excel;
      class body / backgroundcolor=_undef_ fontsize=12pt;
      class systemtitle,systemfooter,usertext / fontfamily=Calibri
            color=black backgroundcolor=_undef_;
      class header / fontfamily=Calibri color=black fontsize=10pt;
      class data / fontfamily=Calibri;
      class graph / height=2.5in width=4in;
      class graphdata1 / color=#C8DADA;
      class graphdata2 / color=#5A6351;
      class graphdata3 / color=#69B0A5;
      class graphdata4 / color=#F2F3F2;
   end;
run;

ods excel file="c:\test.xlsx" style=styles.excel_update
          options(embedded_titles="yes" embedded_footnotes="yes"
          sheet_interval="none"
          start_at="2,2");
          title "Profit Analysis for the Year 2018";
          footnote "Detail of Profits based on Products";
   ods text="Profit Detail";

   proc report data=sashelp.orsales(obs=5);
      col Product_line product_category profit total_retail_price;
      compute product_line;
         count+1;
         if mod(count,2) then call define(_row_,"style",
                                          "style={backgroundcolor=#f5f9f6");
      endcomp;
   run;
```

(code continued)

```
    proc sgplot data=sashelp.orsales noautolegend;
       format profit dollar.;
       vbar product_line / stat=sum response=profit group=Product_line
                           dataskin=gloss;
    run;
ods excel close;
```

In This Example

① The BODY style element in the CLASS statement removes the background color of the worksheet and modifies the font size of Excel headings (also known as headers).

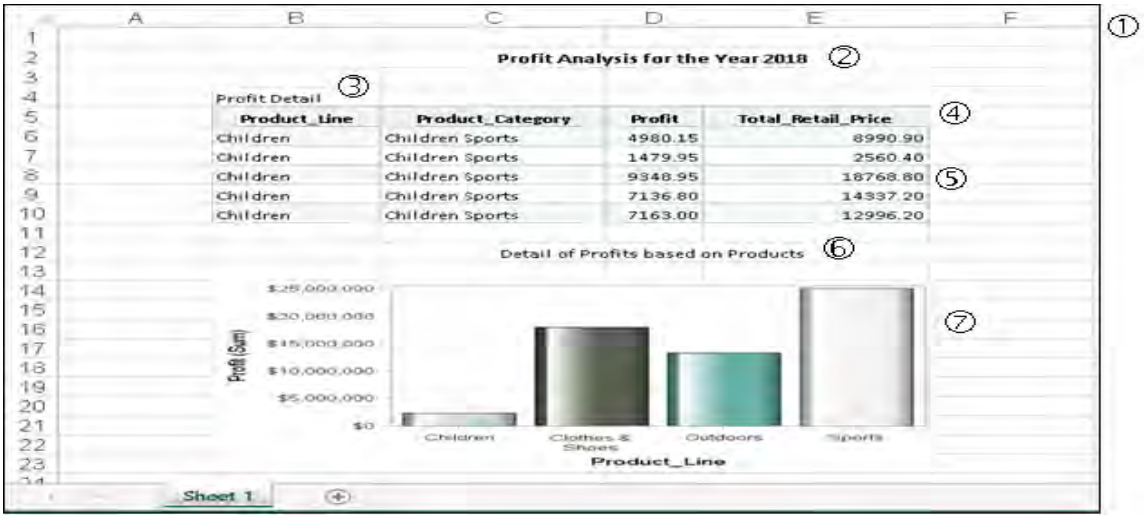
②③⑥ The SYSTEMTITLE, USERTEXT, and SYSTEMFOOTER elements are used to change the font to Calibri, change the foreground color to black, and to remove the background color.

④⑤ The HEADER style element changes the foreground color of the headers to black and changes the font to Calibri. The DATA element changes the font to Calibri.

⑦ The GRAPH style element modifies the height and width of the graph. The GRAPHDATA1-GRAPHDAT4 style elements change the bar colors.

⑧ PROC REPORT generates alternating row shading in the table.

Output



Product_Line	Product_Category	Profit	Total_Retail_Price
Children	Children Sports	4980.15	8990.90
Children	Children Sports	1479.95	2560.40
Children	Children Sports	9348.95	18768.80
Children	Children Sports	7136.80	14337.20
Children	Children Sports	7163.00	12996.20

Figure 1. Applying Style Elements to a Microsoft Excel Worksheet

As listed below, certain SAS styles elements and attributes are not supported by the ODS Excel destination. This list is a sample of the elements and attributes that are more commonly used.

- The PADDING=, CELLSPACING=, RULES=, and FRAMES= attributes are used to control cell padding, cell spacing, and internal and external borders in Excel tables. However, use of these attributes is not supported by the destination.
- The MARGIN= attribute is not supported in the Excel destination's styles, but it is supported in the following SAS system options: TOPMARGIN=, LEFTMARGIN=, RIGHTMARGIN=, and BOTTOMMARGIN=.
- The PREIMAGE= and POSTIMAGE= attributes are also not supported because images are not supported on a per-cell basis. This is also true for the IMAGE method in the Report Writing Interface.
- The HEIGHT= attribute is not supported. Therefore, you must use the ROW_HEIGHT= attribute or the ABSOLUTE_ROW_HEIGHT= suboption to set the height.
- Currently, styles are not supported in the table of contents of an Excel worksheet that is added with the CONTENTS=and INDEX= suboptions.

USING THE MANY FUNCTIONS OF THE TAGATTR= ATTRIBUTE

The TAGATTR= attribute has many functions within the Excel destination. For example, you can add or override Excel formats, add formulas, rotate text, hide columns, control wrapping, and merge cells. This section dives deep into the following functions of the TAGATTR= attribute:

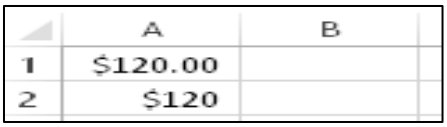
- FORMAT
- FORMULA
- HIDDEN
- MERGEACROSS
- ROTATE
- TYPE
- WRAP

FORMAT

The FORMAT parameter enables you to add or modify Excel formats. When you use a custom Excel format, you can use a combination of characters that act as a template for the display. The most popular use of a custom Excel format is that of using the hash sign (#) and 0 digit selectors. Using a custom format overrides any SAS format that is used and enables you to use advanced formatting. You can also use named formats (CURRENCY, ACCOUNTING, and GENERAL) with the Excel destination. The TAGATTR= attribute overrides any formatting added using SAS formats and the only method of providing formatting using the Report Writing Interface.

The following example illustrates the syntax that you use to add named and custom Excel formats with the TAGATTR= attribute. In the image below, cell A1 displays the value using the named CURRENCY format while cell A2 displays the value using a custom format.

(Named) **TAGATTR**=”FORMAT.CURRENCY”
(Custom) **TAGATTR**=”FORMAT:$#,###”



	A	B
1	$120.00	
2	$120	

FORMULA

You can add formulas to your Excel worksheet by using the FORMULA parameter with the TAGATTR= attribute and using either the R1C1 or theA1 style notation. If you want a formula to be copied across rows, then the R1C1 notation is recommended. The A1 notation is recommended when you have a fixed range of information to use. A formula is often applied along with a custom or named format, as shown in the previous section. When you use a formula and a format in combination, you must separate them by a blank space.

The syntax that is shown below demonstrates how you can apply formulas by using both the R1C1 notation and the A1 notation. In the syntax below, the R1C1 notation adds a formula that subtracts one cell (RC[-2]) from another cell ([RC[-1]). This formula indicates that you want to subtract the value of the column that is two columns from the left of the current cell from the value of the column that is one column from the left of the current cell.

The A1 style notation demonstrates how you can add a sum to the column. In the example below, it is the sum of cells A1, A2, andA3.

(R1C1) **TAGATTR**=”FORMULA:= RC[-1]:RC[-2]”.
(A1) **TAGATTR**=”FORMULA:=SUM(A1:A3)”

ROTATE

To rotate text, which is useful especially if you need to conserve space, you can use the ROTATE parameter with the TAGATTR= attribute. Rotating text might prevent your users from having to scroll to the right of the worksheet to view data. After the text is rotated, you also have to adjust the height to display the text appropriately within the cell. To adjust the height, you need to use the ROW_HEIGHTS= suboption. Specifying a single value for the ROW_HEIGHTS= option modifies the height of the header only. Beginning with the fourth maintenance release of SAS 9.4 (TS1M4), you can specify a unit of measure along with the value. In previous releases, you can only specify an integer as the value.

The next example illustrates how to rotate column headings 90 degrees using the ROW_HEIGHTS= suboption and the TAGATTR= attribute.

Example 2

```
ods excel file="c:\test.xlsx" options(row_heights="55px");
   proc print data=sashelp.class(obs=3)style(header)={tagattr="rotate:90"}
                                                          noobs;
   run;

ods excel close;
```

In This Example

- The ROW_HEIGHTS= suboption sets the header to size of 55 pixels.
- The TAGATTR= option with the ROTATE parameter rotates the headers 90 degrees.

Output

	A	B	C	D	E
1	Na	Se	Ag	He	We
2	Alfred	M	14	69.0	112.5
3	Alice	F	13	56.5	84.0
4	Barbara	F	13	65.3	98.0

Output 1. Header Rotated without Height Adjustment

	A	B	C	D	E
1	Name	Sex	Age	Height	Weight
2	Alfred	M	14	69.0	112.5
3	Alice	F	13	56.5	84.0
4	Barbara	F	13	65.3	98.0

Output 2. Header Rotated after Height Adjustment

TYPE

The TYPE parameter modifies the data type of the cell. The possible values for TYPE are **`String`**, **`Number`**, and **`DateTime`**. At times in an Excel table, a number is stored as text (for example, when you add a currency symbol using the FORMAT procedure). In such cases, the value is denoted by a small green triangle in the upper left corner of the cell. For purposes such as sorting correctly or calculating a summary, you need the number that is stored as text to be stored as a number. You might also have a value that needs to be treated as a text but which is currently stored as a number (for example, a ZIP code that has a leading zero).

If you need to store a value as an Excel date or time value, you should use the **`DateTime`** value for TYPE. If you store such a value as a number, it will not sort as a date.

The following example uses all three values for the TYPE parameter to store data for an Excel table.

Example 3

```
proc format;
   picture cur other="0,000,000" (prefix="€");
run;

data one;
   input Acct $ Total Date mmddyy10.;
   date=date*86400;
   format date datetime. total cur.;
   cards;
1456 100000 02/10/2017
2434 50000 01/01/2018
0123 30000 04/01/2017
0122 20000 05/01/2017
run;
```

(code continued)

```
ods excel file="c:\temp\test.xlsx";

   proc report data=one;
      define Acct /  style(column)={tagattr="type:String"};
      define Total / style(column)={tagattr="type:Number format:€#,###"};
      define date / style(column)={tagattr="type:DateTime
      format:mm/dd/yy;@"};
   run;

ods excel close;
```

In This Example

- The TAGATTR="TYPE:STRING" attribute retains the leading zero in the ACCT column.
- The TAGATTR="TYPE:NUMBER" attribute stores the text string as a number (for the TOTAL column).
- The TAGATTR="TYPE:DATETIME" attribute stores the value of the DATE column as an Excel datetime value.

Output

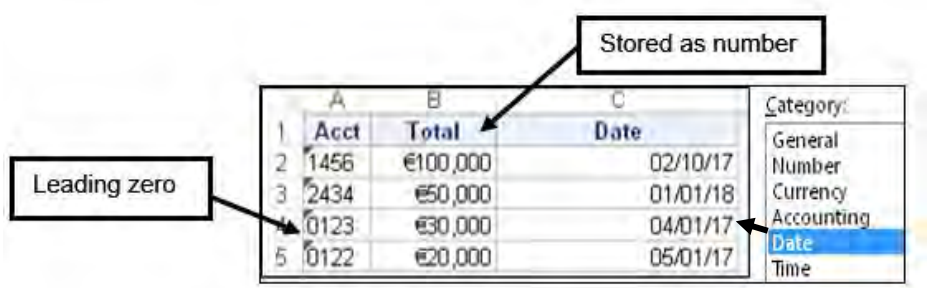


Figure 2. Output That Is Generated After You Use the TAGATTR= Attribute with the TYPE Values of `String`, `Number`, and `DateTime`

WRAP

The WRAP:YES|NO parameter simply selects or de-selects the **Wrap text** feature in Excel, as shown below.

```
proc print data=sashelp.orsales style(header)={tagattr="wrap:no"} label;
```

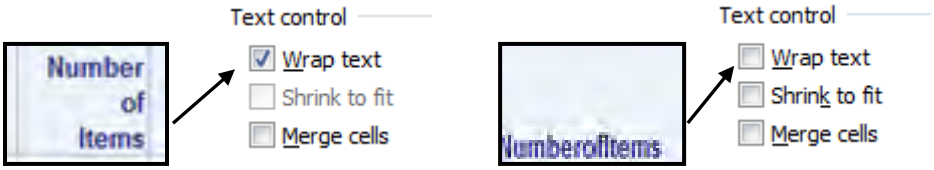


Figure 3. Using the WRAP Parameter to Select or DeSelect the Excel "Wrap text" Feature

MergeAcross

The MERGEACROSS parameter is useful, in limited circumstances, to merge data across columns. This parameter does not have an effect on the table structure. Therefore, merging columns where there is data to the right do not remove text when cells are merged. Some of the useful uses of this parameter currently include spanning text when styles are added and merging columns that do not have data to the right of them.

The next example illustrates two uses the MERGEACROSS parameter.

Example 4

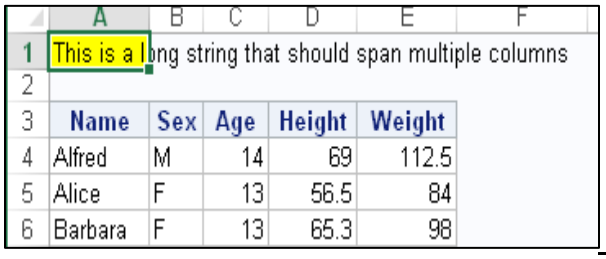
```
ods excel file="c:\temp\test.xlsx" options(sheet_interval="none");

  proc odstext;
     p "This is a long string that should span multiple columns" /
       style={background=yellow tagattr="mergeacross:6"};
  run;

  proc report data=sashelp.class(obs=3);
     column name sex age height weight;
     define name / style(header)={tagattr="mergeacross:2"};
  run;

ods excel close;
```

In This Example

- The first MERGEACROSS parameter in the ODSTEXT procedure to merge cells that contain a long text string.
- The second MERGEACROSS parameter in the DEFINE state merges column A3 (**Name**) and B3 (**Sex**), which retains the header for the column Sex.

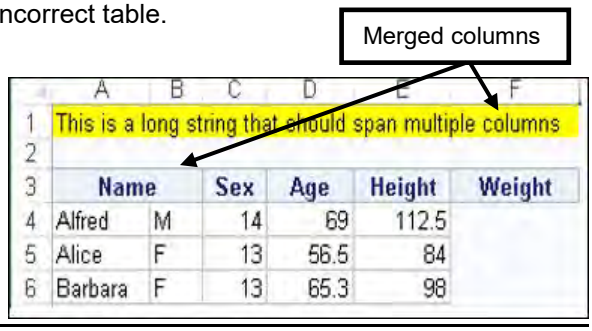
Output

The figure to the left, below, displays the unmerged text string. So when you apply a style to such a string, you see that the style is applied only to the first cell. To ensure that the style is applied to the entire string, you need to merge the cells, as shown in Figure 5. The MERGEACROSS parameter is also applied to the column header, which generates an incorrect table.



	A	B	C	D	E	F
1	This is a long string that should span multiple columns					
2						
3	Name	Sex	Age	Height	Weight	
4	Alfred	M	14	69	112.5	
5	Alice	F	13	56.5	84	
6	Barbara	F	13	65.3	98	

Figure 4. Output without MERGEACROSS Parameter



	A	B	C	D	E	F
1	This is a long string that should span multiple columns					
2						
3	Name		Sex	Age	Height	Weight
4	Alfred	M	14	69	112.5	
5	Alice	F	13	56.5	84	
6	Barbara	F	13	65.3	98	

Figure 5. Output with MERGEACROSS Parameter

HIDDEN

The HIDDEN parameter enables you to hide one or more columns in an Excel worksheet. You can hide a single value or a list of columns that are separated by a comma. The example below hides a single value:

```
tagattr="hidden:3";
```

CREATING ADVANCED FORMATTING FOR WORKSHEETS

This section covers using advanced formatting to enable you to perform tasks that typically you cannot do with the template styles (or, in some cases, with style overrides). Such tasks include the following:

- applying separate styles for each table
- adding alternating row shading for rows or columns
- applying styles to individual columns and rows (for example, adding styles to a summary line in a table)
- adding a style to any procedure

You can create advanced formatting for a worksheet by using cascading style sheets (CSS), which provide the ability to perform tasks that you cannot do with other methods of adding styles. Using a CSS for formatting enables you to take advantage of the CSS3 library, which is part of the World Wide Web Consortium (W3C) standards. With this library, you can use class selectors, type selectors, ID selectors, pseudo selectors, attribute selectors, and combinators to add style to your output. The technology for adding styles via a CSS does not exist in Microsoft Excel. HTML does have that capability. With HTML, you can view the HTML tags that are required to build the style selectors that are part of a CSS. To simulate this same technology with Excel destination, you can use the DOM SAS system option or the DOM option in the ODS Excel statement. This option displays the output as an HTML file. For more information about the DOM option and cascading style sheets, see Parker (2015).

For the next example, you need to create the following CSS file and then include it in the ODS Excel statement that is shown below in Example 5.

CSS File

```
* {font-family:arial}
.body {background:lightblue}
 #idx table th {background-color:royalblue;color:white}
 table tr:nth-child(even) td{background-color:lightgray}
 #idx1 table th {background-color:red;color:white}
 table td:nth-child(5) {color:format('fore');fontweight:bold};
```

This example uses the CSS file above to add styles to tables in the Excel worksheet:

Example 5

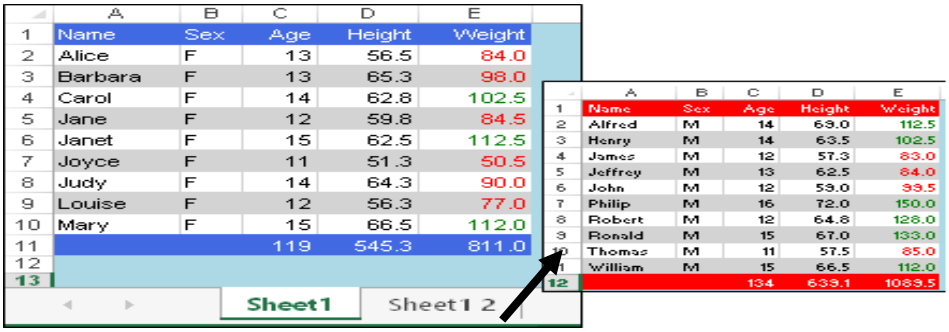
```
proc format;
   value fore 0-<100="red"
              other="green";
run;

ods excel file="c:\test.xlsx" cssstyle="c:\cssex.css";

  proc report data=sashelp.class(where=(sex="F"));
     rbreak after / summarize;
  run;

  proc report data=sashelp.class(where=(sex="M"));
     rbreak after / summarize;
  run;

ods excel close;
```

In This Example

In the SAS example code above, the two PROC REPORT steps create two separate worksheets, while the CSS file adds the following style characteristics:

- The universal style selector (*****) sets the font to Arial for the entire document.
- The .**BODY** class selector sets the background color of the worksheet to royal blue.
- The **#IDX** ID selector sets the background color and the foreground color of the header line for the first worksheet, and the **nth-child** pseudo selector sets alternating colors for the rows.
- The second **#IDX1** ID selector performs the same tasks as the first ID selector, but the changes are for Sheet 2 of the workbook. These selectors change the header line to red (as opposed to blue on the first sheet), and the alternating row colors are maintained as well. T
- PROC FORMAT adds trafficlighting to the WEIGHT column for both worksheets by using the **td** element with the **nth-child** pseudo selector. The trafficlighting style is accessed from the CSS.

Output



	A	B	C	D	E
1	Name	Sex	Age	Height	Weight
2	Alice	F	13	56.5	84.0
3	Barbara	F	13	65.3	98.0
4	Carol	F	14	62.8	102.5
5	Jane	F	12	59.8	84.5
6	Janet	F	15	62.5	112.5
7	Joyce	F	11	51.3	50.5
8	Judy	F	14	64.3	90.0
9	Louise	F	12	56.3	77.0
10	Mary	F	15	66.5	112.0
11			119	545.3	811.0
12					
13					

Sheet1 Sheet1 2

	A	B	C	D	E
1	Name	Sex	Age	Height	Weight
2	Alfred	M	14	69.0	112.5
3	Henry	M	14	63.5	102.5
4	James	M	12	57.3	83.0
5	Jeffrey	M	13	62.5	84.0
6	John	M	12	59.0	99.5
7	Philip	M	16	72.0	150.0
8	Robert	M	12	64.8	128.0
9	Ronald	M	15	67.0	133.0
10	Thomas	M	11	57.5	85.0
11	William	M	15	66.5	112.0
12			134	639.1	1083.5

Figure 6. Creating Advanced Styles Using a CSS

BLENDING STYLES, FORMATTING, AND FORMULAS

Earlier in the paper, you see how to apply styles, add formatting, and include formulas to help enhance your worksheets. Spreadsheets are both a form of expression and a method for computation that enable people to make important decisions and to solve problems. So, spreadsheets need to be accurate, easy to understand, and visually appealing, with easy-to-understand worksheets. You can accomplish all three of these goals by combining the techniques discussed earlier in this paper: adding styles, using formatting, and including formulas for computations.

The next example, which creates a forecast, uses the techniques discussed previously to add style and formatting as well as calculations.

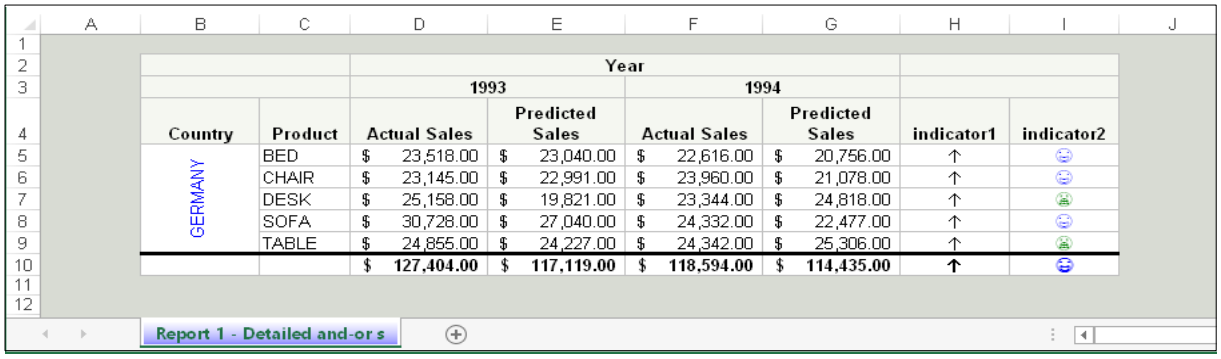
Example 6

```
Ods escapechar="^";
ods excel file="c:\temp\test.xlsx" style=statistical
                options(start_at="2,2" tab_color="blue");
proc report data=sashelp.prdsale(where=(country="GERMANY")) spanrows;
   column  country product year,( actual predict) indicator1 indicator2;
   define year / across;
   define actual / style(column)={tagattr="format:accounting" width=.8in};
   define predict / style(column)={tagattr="format:accounting" width=.8in};
   define country / group style(column)={tagattr="rotate:90" just=center
                                         vjust=middle color=blue};
   define product / group;
   define indicator1 / style(column)={tagattr='
                      formula:=IF(RC[-4] > RC[-3],"á","â")'
                      fontfamily=wingdings just=center};
   define indicator2 /  style={just=center};

   compute indicator2 / length=100 character;
      if _c5_ > _c6_ then indicator2="^{style[color=blue]^{unicode
                                                          1F603}}";
      else indicator2="^{style[color=green] ^{unicode 1F620}}";
   endcomp;
   rbreak after / summarize style(summary)={fontweight=bold
                                            borderstyle=solid
                                            bordertopwidth=5pt
                                            bordertopcolor=black};

run;
ods excel close;
```

In This Example

- The ROTATE parameter is added within the TAGATTR= attribute to rotate the values of the Country column by 90 degrees.
- The FORMAT parameter within the TAGATTR= attribute adds the ACCOUNTING format for ACTUAL and PREDICT variables.
- The INDICATOR1 variable uses a special variable from a character map to add the special characters. (This is a Windows character map that you access by entering **`character map`** in the **Start** menu's search field.) INDICATOR2 uses the UNICODE function to display indicators in the column.
- The TAB_COLOR= and START_AT= options are used along with the STATISTICAL style.

Output



			Year					
			1993		1994			
Country	Product	Actual Sales	Predicted Sales	Actual Sales	Predicted Sales	indicator1	indicator2
GERMANY	BED	$ 23,518.00	$ 23,040.00	$ 22,616.00	$ 20,756.00	↑	
	CHAIR	$ 23,145.00	$ 22,991.00	$ 23,960.00	$ 21,078.00	↑	
	DESK	$ 25,158.00	$ 19,821.00	$ 23,344.00	$ 24,818.00	↑	
	SOFA	$ 30,728.00	$ 27,040.00	$ 24,332.00	$ 22,477.00	↑	
	TABLE	$ 24,855.00	$ 24,227.00	$ 24,342.00	$ 25,306.00	↑	
		$ 127,404.00	**$ 117,119.00**	**$ 118,594.00**	**$ 114,435.00**	**↑**	

Output 3. Generating an Expense Report by Adding Styles and Indicators

ADDRESSING ISSUES THAT OCCUR WHEN YOU USE THE ODS EXCEL DESTINATION

This section addresses behaviors and limitations that you can encounter when you use the Excel destination. The following sections provide an in-depth look at those behaviors and limitations.

WRAPPING TEXT WITH THE ODS EXCEL DESTINATION

How text wraps within a cell when you use the Excel destination is complicated, and it might even seem a bit mysterious. Hopefully, this section can demystify that concept for you.

The ODS Excel destination is a measured destination. A *measured destination* uses an algorithm to determine how text should wrap for the best presentation. When the text wraps within a column, a hard return (carriage-return/line-feed character) is added where the wrapping occurs. This hard return can cause problems during creation of a formula, for example, or it might simply prevent a column from expanding until the hard return is removed. This section discusses several behaviors related to text wrapping with the Excel destination.

To prevent the addition of hard returns during text wrapping, you can use either of the following methods:

- Use the FLOW= suboption.
- Use the ABSOLUTE_COLUMN_WIDTH= option with the FLOW= option.
- Use the WIDTH= attribute with the TAGATTR= attribute.
- Create an output area that is large enough to prevent wrapping.

Using the ODS EXCEL FLOW= Suboption

The FLOW= ODS Excel suboption (available in SAS 9.4 TS1M4 or later) maintains the natural flow of text within a cell and prevents the addition of a hard return. Using the FLOW= ODS suboption enables you to control wrapping selectively for the various parts of the output. Valid values for this option are **`Headers`**, **`Rowheaders`**, **`Data`**, and **`Tables`**. The first three values enable you to control individual parts of a table, whereas the value **`Tables`** enables you to control wrapping for the entire table.

You can add text to a worksheet by using such methods as the ODS TEXT= statement or the ODSTEXT procedure. To control wrapping for such added text, you can use the **`Text`** value in the FLOW= suboption. You can also specify a range to prevent a hard return in text line.

The following output shows a worksheet table that uses hard returns to wrap lines rather than using the FLOW= suboption.

	A	B	C	D	E	F	G	H	I
1	Obs	Year	Quarter	Product Line	Product Category	Product Group	Number of Items	Profit in USD	Total Retail Price in USD
2	1	1999	1999Q1	Children	Children Sports	A-Team, Kids	286	4980.15	8990.90
3	2	1999	1999Q1	Children	Children Sports	Bathing Suits, Kids	98	1479.95	2560.40

Figure 7. Wrapping Text Lines with Hard Returns rather than with the FLOW= Suboption

Rather than using hard returns, which can cause a variety of issues, you can use the FLOW= suboption to wrap the text. Example 7 shows the suboption with the **`Tables`** value that is described earlier.

Example 7

```
ods excel file="c:\temp\test.xlsx" options(flow="tables");

   proc print data=sashelp.orsales label;
   run;

ods excel close;
```

Output

	A	B	C	D	E	F	G	H	I
1	Obs	Year	Quarter	Product Line	Product Category	Product Group	Number of Items	Profit in USD	Total Retail Price in USD
2	1	1999	1999Q1	Children	Children Sports	A-Team, Kids	286	4980.15	8990.90
3	2	1999	1999Q1	Children	Children Sports	Bathing Suits, Kids	98	1479.95	2560.40

Output 4. Using the FLOW= Suboption to Prevent a Hard Return in Text Wrapping

Combining the FLOW= Suboption with Other ODS Excel Options and Attributes

You can use the FLOW= suboption in combination with the ABSOLUTE_COLUMN_ WIDTH= option, the WIDTH= style attribute, or the TAGATTR= attribute with the WRAP parameter to control text wrapping. The benefit of using FLOW= in combination with other Excel options or attributes is that you can determine exactly where you want the wrapping to occur.

The ABSOLUTE_COLUMN_WIDTH= Option

The ABSOLUTE_COLUMN_WIDTH= option enables you to specify widths for one or more columns in a table. Beginning with SAS 9.4 TS1M4 and later, you can now use a unit of measure for the value of the option (for example, pt, px, in, cm, and mm).

If you specify multiple values, they are separated by a comma, as shown below:

If you specify a single value, as shown in the following example, that width applies to all columns.

```
ods excel file="c:\test.xlsx" options(flow="tables"
                                      Absolute_Column_Width="1in,15")
```

The ABSOLUTE_COLUMN_WIDTH= suboption controls where the text wraps inside a cell (similar to functionality that is available in the ExcelXP tagset). If you do not specify the ABSOLUTE_COLUMN_WIDTH= option in conjunction with the FLOW= suboption, any wrapping occurs before the ABSOLUTE_COLUMN_WIDTH= suboption is applied. (Parker 2016)

The WIDTH= Style Attribute

Specifying the WIDTH= attribute with the ODS Excel destination enables you to override the default wrapping algorithm. This attribute is specified in a DEFINE statement, as shown below:

```
define var1 / style={width=30px};
```

If the width that is specified is large enough to display the full value, wrapping does not occur. If the width is not large enough to display the entire value, text wrapping occurs and a hard return is added where the line breaks.

If you specify the WIDTH= attribute in conjunction with the ABSOLUTE_COLUMN_ WIDTH= option, any wrapping is controlled by the value that is specified in the WIDTH= attribute. However, the cell width is still controlled by the ABSOLUTE_COLUMN_WIDTH= option. If you use WIDTH= in conjunction with the FLOW= suboption, then the WIDTH= attribute is used and a natural flow of text is added.

The WRAP Parameter within the TAGATTR= Attribute

When you specify the WRAP=YES|NO parameter in the TAGATTR= attribute, in conjunction with the FLOW= suboption, natural (default) wrapping is used for text in a cell. If you set the WRAP parameter to NO, Microsoft Excel's default text-wrapping option is de-selected. As a result, text might not display correctly and the text height might be incorrect. For an example using the WRAP parameter, see the WRAP example in the section "Using the Many Functions OF THE TAGATTR= Attribute."

Text Wrapping Based on Size of the Output

As mentioned earlier, text wrapping is complicated and goes far beyond what this paper has discussed thus far. By default, the Excel destination attempts to fit output within the size of an 8 ½ X 11 physical piece of paper. When output goes beyond that size, text in the output might wrap This is true for the REPORT procedure which does the best job of preventing text wrap than any of the other procedures. The default output created with PROC PRINT in figure 7 which displays some wrapping of the text would occur with output of the same width generated PROC REPORT.

Consider the following example, which uses PROC REPORT to generate output that fits into the 8 ½ x 11 size:

Example 8

```
ods excel file="c:\temp\test.xlsx";

   proc report data=sashelp.orsales;
      columns Year Quarter Product_Line Product_Category Product_Group
              Quantity Profit Total_Retail_Price;
   run;

ods excel close;
```

In This Example

In this example, wrapping does not occur because all the text fits within the 8 ½ x 11 space.

Output

	A	B	C	D	E	F	G	H
1	Year	Quarter	Product Line	Product Category	Product Group	Number of Items	Profit in USD	Total Retail Price in USD
2	1999	1999Q1	Children	Children Sports	A-Team, Kids	286	4980.15	8990.90
3	1999	1999Q1	Children	Children Sports	Bathing Suits, Kids	98	1479.95	2560.40

Output 5. Using Sufficient Space to Avoid Text Wrapping

However, if you add an option or anything else that changes the overall size of the table contents, the content does not fit the space and wrapping occurs. In this next example, the NEWVAR variable is included as a computed variable in the code to add a new column.

Example 9

```
ods excel file="c:\temp\test.xlsx";

   proc report data=sashelp.orsales;
      columns Year Quarter Product_Line Product_Category Product_Group
              Quantity Profit Total_Retail_Price newvar;
   run;

ods excel close;
```

In This Example

In this example, the NEWVAR variable is added as a computed variable to create another column. As a result, the table contents no longer fit the allotted space, and the header text is wrapped.

Output

	A	B	C	D	E	F	G	H	I
1	Year	Quarter	Product Line	Product Category	Product Group	Number of Items	Profit in USD	Total Retail Price in USD	newvar
2	1999	1999Q1	Children	Children Sports	A-Team, Kids	286	4980.15	8990.90	.
3	1999	1999Q1	Children	Children Sports	Bathing Suits, Kids	98	1479.95	2560.40	.

Output 6. Increasing the Output Width Causes the Header Text to Wrap

The next example uses the PAPERSIZE= option to increase the area that is used by the Excel destination. This OPTIONS statement would appear before the code in the previous example. The ORIENTATION=LANDSCAPE system option might also enable you to create a large enough area.

Example 10

```
options papersize=(22in 27in);
. . .rest of code example. . .
```

In This Example

By increasing the output area, text wrapping no longer occurs. Be aware that you must include the PAPERSIZE= option in your code before the ODS EXCEL statement.

Output

	A	B	C	D	E	F	G	H	I
1	Year	Quarter	Product Line	Product Category	Product Group	Number of Items	Profit in USD	Total Retail Price in USD	newvar
2	1999	1999Q1	Children	Children Sports	A-Team, Kids	286	4980.15	8990.90	.
3	1999	1999Q1	Children	Children Sports	Bathing Suits, Kids	98	1479.95	2560.40	.

Output 7. Increasing the Output Area with the PAPERSIZE= Option

GENERATING LARGE FILES WITH THE ODS EXCEL DESTINATION

Generating **l**arge files using the Excel destination might require some preparation. The Excel destination retains memory as it collects information about each column until the table is completed. Then the memory is released. As a result, larger tables can cause out-of-memory conditions, depending on the amount of memory that you have allocated for SAS jobs. This section addresses the following topics related to out-of-memory conditions:

- determining how much memory is required
- correcting an out-of-memory condition
- modifying the MEMSIZE= option
- solving memory issues by breaking large tables into smaller ones

Determining How Much Memory That You Need

The term “large file” is a relative concept, and it is different for everyone you ask. However, larger tables require some planning. When an out-of-memory error occurs, you need to determine how much memory is requested. To do that, use the MEMSIZE= option. You can use the following PROC OPTIONS statement to query the current value of the memory size that you have set.

```
proc options option=memsize;
run;
```

If you add the FULLSTIMER system option to this code, you can display the amount of memory that has been used.

Determining an estimate of how much memory is required to run a specific report using the Excel destination takes a very simplistic formula:

required_memory=(columns*rows)*5500

This unofficial formula calculates the requested memory, based on the input data, by multiplying the rows and the columns with a multiplier (5500). This multiplier of 5500 is for SAS 9.4 TS1M5.

Note: This simplistic equation does not account for other issues that might affect memory.

Correcting an Out-of-Memory Condition

When you encounter an out-of-memory condition, your goal is to have enough memory to still complete the job. To successfully run a report that has an out-of-memory condition, use either of the following methods:

- Use the MEMSIZE= system option to increase the memory that is available to SAS.
- Create smaller tables either by creating smaller segments (for example, BY groups) or be breaking up tables.
- Use either the ExcelXP or the MSOffice2K tagset and a post process to convert your file an XLSX file. See SAS Sample 43496 for how to convert your file. (**support.sas.com/kb/43/496.html**)

Modifying Memory Size with the MEMSIZE= System Option

You must add the MEMSIZE= system option at the invocation of SAS, either on the command line or in the SAS configuration file. The following example shows the use of the system option (-MEMSIZE, in this case) on a command line:

```
prompt > sas –dms –memsize 4g
```

To modify the MEMSIZE= system option in the SAS configuration file, you first need to locate the configuration file, which is typically named SASV9.CFG. To locate this file, submit the following PROC OPTIONS statement:

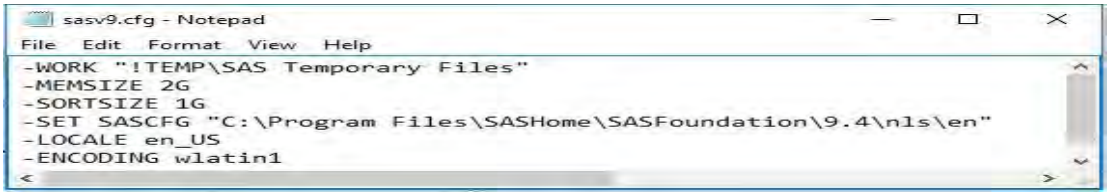
```
proc options option=config;
run;
```

After you locate the configuration file and decide how much memory you need, you can modify the configuration file. If you have a simple, local installation, you should have privileges to modify this file. If not (for example, if you have an instance of SAS on a server), your system administrator needs to modify the file for you. Another option is to make a customized version of this configuration file by saving a copy of the file to your home directory and modifying that copy. In Microsoft Windows operating environments, you can submit the following directive to locate the home directory. You can also use the -CONFIG= command-line option to point to this customized version. See the SAS host companion for your operating system for details. (**support.sas.com/documentation/onlinedoc/base/index.html**)

For Windows: `%USERPROFILE%`

For UNIX/Linux: `$Home`

The configuration file is similar to the following:



Output 8. Adjusting Memory Size in the Configuration File

You can also specify the value MAX as an argument to the -MEMSIZE option, which returns the maximum amount of memory that is available. You can use this technique to see how much memory is used and then set the memory size accordingly.

Resolving Memory Issues by Breaking Up Tables

As mentioned previously, the default behavior of a measured destination (that is, holding data in memory), can cause out-of-memory conditions with larger tables. You can avoid that problem by using the Excel destination to break up large tables into smaller tables so you can export the output to Excel. If you can export the output as BY groups, you can use the SHEET_INTERVAL="BYGROUP" option to break up the table.

You can also break up a table based on the -MEMSIZE setting. The following example uses a macro to query the number of variables and observations as well how much memory is currently available. Based on that query, separate groups can be created, which then create smaller tables. In this example, PROC REPORT is used with the BREAK AFTER statement with the PAGE option to create the containers for each value of the counter group, which is created in the DATA step. A large sample data set is included in this download file: **ftp.sas.com/techsup/download/base/SGF2018-SAS2174.zip**.

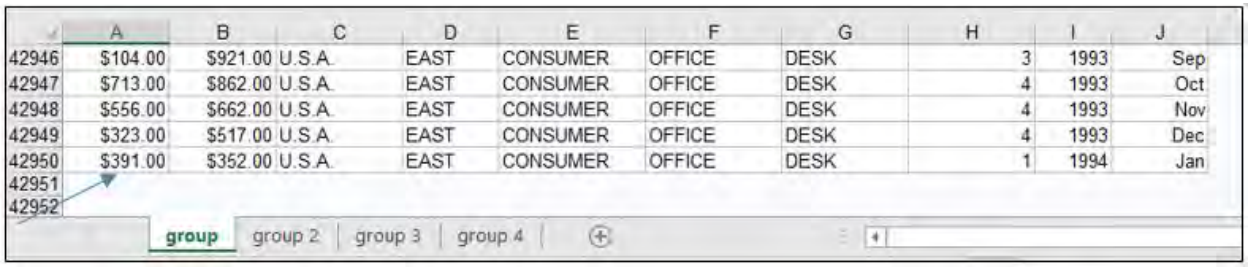
Example 11

```
%macro group(dsn);
      /* Obtain the number of columns and rows in the data sets. */
   %let rc=%sysfunc(open(&dsn,i));
   %let nobs=%sysfunc(attrn(&rc,NOBS));
   %let nvars=%sysfunc(attrn(&rc,NVARS));
   %let close=%sysfunc(close(&rc));

      /* Determine how many rows can be read, based on memory. */
   %let memsize=%sysfunc(getoption(MEMSIZE));
   %let req_memory=%sysevalf(&nobs*&nvars*5500);
   %let memory_delta=%sysevalf(&memsize -&req_memory);
   %let groups=%sysevalf(&memsize/(&nvars*5500));

   %if &memory_delta < 0 %then %put "Groups are created";

   data one;
      set &dsn;
      if mod(_n_,int(&groups))=1 then count+1;
   run;

   ods excel file="c:\temp\test.xlsx" options(sheet_name="group");
      proc report data=one;
         define count / order noprint;
         break after count / page;
      run;

   ods excel close;
%mend;
%group(work.largedata)
```

In This Example

- The GROUP macro calculates the number of groups that are needed, based on available memory, to export the output to the worksheets.
- PROC REPORT uses the PAGE option in the BREAK AFTER statement to generate a worksheet for each unique group.

Output



	A	B	C	D	E	F	G	H	I	J
42946	$104.00	$921.00	U.S.A.	EAST	CONSUMER	OFFICE	DESK	3	1993	Sep
42947	$713.00	$862.00	U.S.A.	EAST	CONSUMER	OFFICE	DESK	4	1993	Oct
42948	$556.00	$662.00	U.S.A.	EAST	CONSUMER	OFFICE	DESK	4	1993	Nov
42949	$323.00	$517.00	U.S.A.	EAST	CONSUMER	OFFICE	DESK	4	1993	Dec
42950	$391.00	$352.00	U.S.A.	EAST	CONSUMER	OFFICE	DESK	1	1994	Jan
42951										
42952										

Output 9. Using the PAGE Option to Create Separate Groups for Exporting Output to Microsoft Excel

MANAGING EXCEL WORKBOOKS

In managing Excel workbooks and worksheets, you might like to protect worksheets so that users can only read the worksheets (that is, they cannot make changes or recommendations to the file). You might also want to be able to write to an existing worksheet or to write multiple tables or graphs to a worksheet. This section addresses how to accomplish these tasks and discusses some limitations that exist.

Writing to an Existing Excel Workbook

The Excel destination cannot update workbooks or worksheets in a workbook because it generates an entirely new workbook each time you execute the destination, However, you can use other methods with the Excel destination to update an existing workbook.

For example, you can use PROC REPORT and the Excel destination to perform a lookup of data.

Example 12

```
options nolabel;

proc template;
   define style lookup;
      parent=styles.excel;
      class body / background=_undef_ textalign=center;
    end;
run;

ods excel file="c:\Codes.xlsx" options(sheet_name="Report") style=lookup;

   proc report data=sashelp.demographics(obs=100);
      column CONT ID ISO Country;
      define cont / display;
      define id / display;
      define iso / display;
      define country / computed
                       style(column)={tagattr="formula:=VLOOKUP(RC[-1],
                       'Data'!$C$1:$D$100,2,FALSE)" just=left};
   run;

ods excel close;
```

(code continued)

```
proc export data=sashelp.demographics outfile="c:\Codes.xlsx"
   dbms=xlsx replace;
   sheet="Data";
run;
```

In This Example

- The ODS Excel destination with PROC REPORT adds the VLOOKUP function to the computed column Country.
- PROC EXPORT is used to add a data sheet to the existing workbook created with ODS EXCEL.
- PROC TEMPLATE was used to provide style information for the added worksheet created from PROC EXPORT.

Output

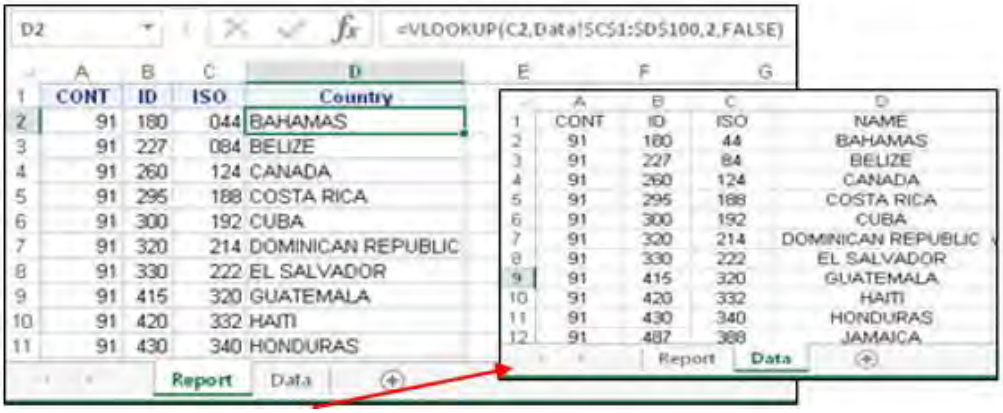


Figure 6. Using PROC REPORT and the VLOOKUP Function to Update an Existing Worksheet

Under Windows, you can also use the %EXCEL_ENHANCE macro to update a worksheet. This macro enables you to copy generated worksheets to an existing workbook. The macro accomplishes this task by building Visual Basic code from the macro that copies a worksheet from one workbook to another.

Protecting Excel Worksheets

The ability to protect worksheets, using the Excel destination's new PROTECT_SHEETS= suboption was made available in SAS 9.4 TS1M4. This suboption enables you to create a worksheet that cannot be edited (Read-only). This type of protection is useful when you want to send a document out for review but you do not want reviewers to be able to modify the document. You can also toggle this option if you want to apply protection to selected worksheets within a workbook.

Example 13 illustrates how you use the PROTECT_SHEETS= suboption to apply protection to all the worksheets in the TEMP.XLSX workbook.

Example 13

```
ods excel file="c:\temp.xlsx" options(protect_sheets="on");

   proc print data=sashelp.orsales;
   run;

ods excel close;
```

ADDING AND POSITIONING IMAGES, GRAPHICS, AND TABLES

One way to enhance your worksheets is to add graphics and images to a worksheet. The Excel destination can add graphic images by using the SAS Output Delivery System's (ODS) Graph Template Language (GTL), the ODS Statistical Graphics (or, ODS Graphics) procedures, and SAS/GRAPH® software procedures. The following discussion covers topics that will help you to create graphic images and tables. The discussion also covers default behaviors and limitations.

Adding Multiple Tables or Graphs on a Page

People often want to show a graph and a table on the same worksheet for easy reference. You can add multiple tables or graphics on each worksheet using the SHEET_INTERVAL= "NOW" option. This new value, available since SAS 9.4 TS1M5, enables you to add a worksheet on demand.

The following example creates a table and a graph on the same worksheet before advancing to a new worksheet, where another table and graph are added. If you use SAS 9.4 TS1M4 and earlier. you can advance the worksheet by using the workaround in SAS Note 57766, "Add multiple tables and graphs to each worksheet using the ODS Destination for Excel." (**support.sas.com/kb/57/766.html**).

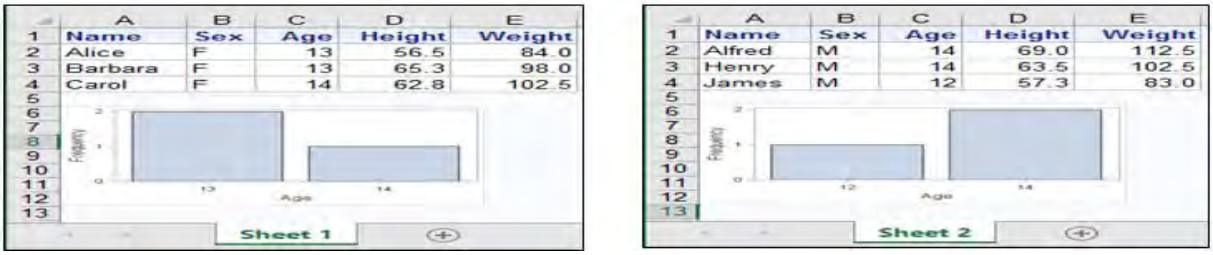
Example 14

```
ods graphics / height=3in width=3in;
ods excel file="c:\test.xlsx" options(sheet_interval="none");

   proc print data=sashelp.class(where=(sex="F"));
   run;

   proc sgplot data=sashelp.class(where=(sex="F"));
      vbar age;
   run;

ods excel options(sheet_interval="now");

   proc print data=sashelp.class(where=(sex="M"));
   run;

   proc sgplot data=sashelp.class(where=(sex="M"));
      vbar age;
   run;

ods excel close;
```

In This Example

- The SHEET_INTERVAL="NONE" suboption enables the graph and table to remain on the same worksheet.
- The SHEET_INTERVAL="NOW" suboption advances to a new worksheet, where a new table and graph are added.

Output



Output 10. Using the SHEET_INTERVAL= Suboption to Add a Table and a Graph on the Same Worksheet

Positioning Tables and Graphics on the Worksheet

You have a few tools available to help you position tables and graphics on a worksheet. For example, you can use the Excel destination's START_AT= suboption (see Example 1), although this option is limited to specifying a one-time starting position on the worksheet.

The Excel destination does not support the ODS Layout facility, so the destination is also limited in that it does not enable you to position tables and graphics side by side. However, you can accomplish this task outside of the destination (but that is rendered by the destination) by using either of the following

methods:

- Use the SAS/GRAPH® procedure GREPLAY.
- Use the Tagsets.MSOffice2K_X tagset.
- Use the EXCEL_ENHANCE macro. (**ftp.sas.com/techsup/download/base/SGF2018-SAS2174.zip**)

The following example uses the EXCEL_ENHANCE macro to post process the Excel worksheet and add tables side by side. **Note:** The example code generates VBScript language. Therefore, this specific code works only under Windows.

Example 15.

```
ods noresults;
ods listing gpath="c:\temp\";
ods excel file="c:\temp\test.xlsx" options(sheet_name="sheet1"
                          sheet_interval="none" embedded_titles="yes");
ods graphics / height=2in width=2in imagename="graphtest";
ods excel exclude sgplot(persist);

proc sgplot data=sashelp.prdsale;
   vbar product / response=actual;
run;

proc sgplot data=sashelp.prdsale;
   vbar country / response=actual;
run;

title "Sales by Product";
proc report data=sashelp.prdsale;
   column product actual ;
   define product / group;
run;

title "Sales by Country";
proc report data=sashelp.prdsale;
   column country actual ;
   define country / group;
run;

ods excel close;

%include "c:\enhance_macro.sas";
%Excel_enhance(open_workbook=c:\temp\test.xlsx,
               move_table=%str(a10:b15!g1),
               insert_image=%str(c:\temp\graphtest.png#sheet1!d3,
                                 c:\temp\graphtest1.png#sheet1!j3),
               create_workbook=c:\test\temp_update.xlsx);
```

In This Example

- The OPEN_WORKBOOK= parameter in the %EXCEL_ENHANCE macro opens the workbook to the point where the tables are to be positioned.
- The MOVE_TABLE= parameter in the macro specifies a range of cells to copy, followed by the location (after the exclamation point [!]) to which to copy the cells.
- The CREATE_WORKBOOK= parameter creates a new workbook. This feature is useful when you want to maintain your original worksheet.

Output

Prior to modification

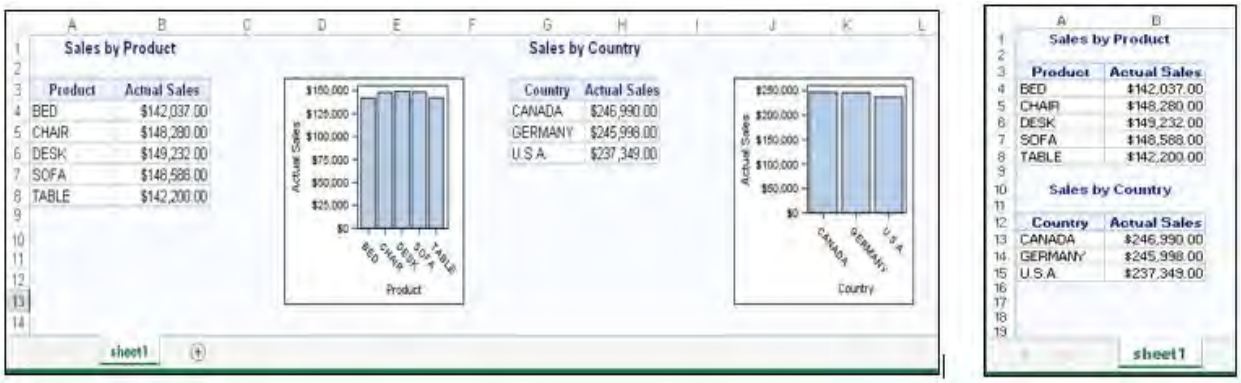


Figure 7. Using OPEN_WORKBOOK= Parameter and the %EXCEL_ENHANCE Macro to Position Tables

Valid Graphics Formats

The ODS Excel destination supports three image formats: EMF, JPG, and PNG. The default file format is PNG. Specifying images in a format other than PNG results in an informational warning. For example, when you use a release of SAS® Enterprise Guide® that is earlier than 7.1.3, you might see the following error:

```
WARNING: Invalid device 'ACTIVCEX' for EXCEL destination.
         Using default device 'PNG'.
```

The warning occurs because some earlier releases of SAS Enterprise Guide use the ActiveX device. The warning does not occur with SAS Enterprise Guide 7.1.3.

Image Scaling

Regardless of whether you use ODS Graphics or SAS/GRAPH statistical procedures, you can modify the height and width of images. For ODS Graphics, you can set the HEIGHT and WIDTH= options on the ODS GRAPHICS statement. For SAS/GRAPH statistical procedures, you can set height and width with the HSIZE= and VSIZE= options in the GOPTIONS statement.

When you view a graphic image in Excel, you might notice that the image is not the exact size that you specify. In releases earlier than SAS 9.4 TS1M5, output is scaled about 15% smaller. However, this is not the case in SAS 9.4 TS1M5. In TS1M5, images might not appear the exact size that you specify because the software must make the image fit within the columns and rows in the worksheet.

Adding a Logo to an Excel Worksheet

The Excel destination currently supports the addition of only background images (using the BACKGROUNDIMAGE= attribute). Adding image types such as a logo is not supported because they need to be included on a per-cell basis. However, you can use a few other methods to include images that are included on a per-cell basis.

- You can use the GOPTIONS statement with the IBACK= option along with the GSLIDE procedure to add a logo to cell A1.
- You can use the %EXCEL_ENHANCE macro (in Windows environments only).
- You can also add an image by using the Apache POI Library to post process the file. Apache POI enables you to read, write, and modify Microsoft Office files, including Excel. The use of Apache POI to modify an Excel file is discussed later in the section "Using Open Source with the Excel Destination."

The following example uses the GOPTIONS statement with the IBACK= option and PROC GLSIDE to add a logo. This method does not allow you to position the image.

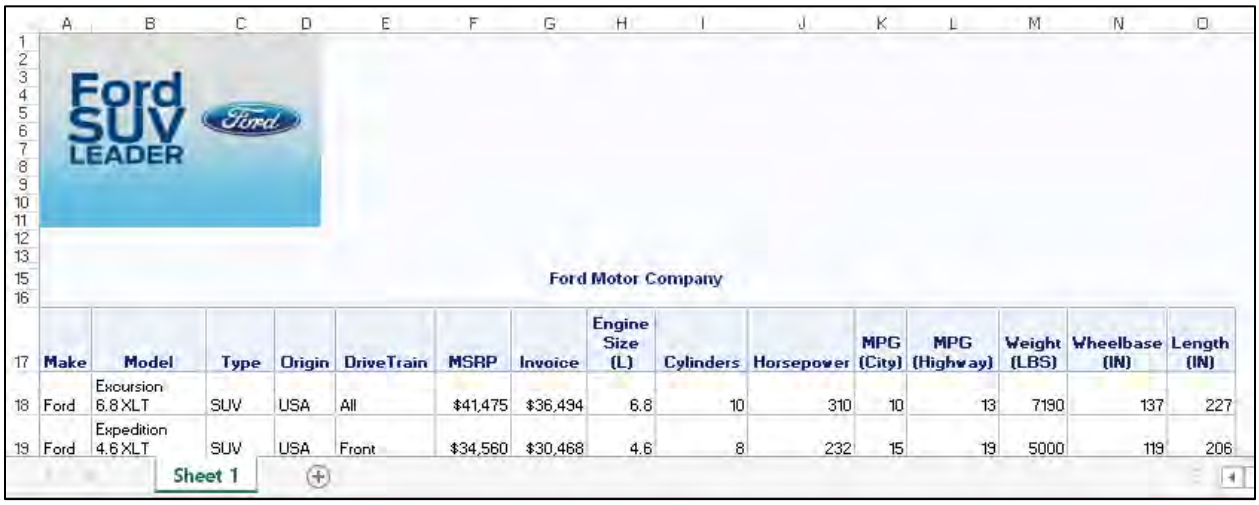
Example 16

```
ods excel file="c:\temp.xlsx" options(sheet_interval="none"
                                      embedded_titles="yes");
   title;
   goptions iback="c:\ford.jpg" imagestyle=fit;

   proc gslide;
   run;

   proc report data=sashelp.cars(where=(make="Ford"));
      title "Ford Motor Company";
   run;

ods excel close;
```

In This Example

- The GOPTIONS statement with the IBACK= option adds an image to cell A1 in the worksheet.
- PROC GSLIDE exports the output to the worksheet.

Output



Make	Model	Type	Origin	DriveTrain	MSRP	Invoice	Engine Size (L)	Cylinders	Horsepower	MPG (City)	MPG (Highway)	Weight (LBS)	Wheelbase (IN)	Length (IN)
Ford	Excursion 6.8 XLT	SUV	USA	All	$41,475	$36,494	6.8	10	310	10	13	7190	137	227
Ford	Expedition 4.6 XLT	SUV	USA	Front	$34,560	$30,468	4.6	8	232	15	19	5000	119	206

Output 11. Using PROC GLIDE with the GOPTIONS Statement to Add a Logo

USING OPEN SOURCE WITH THE ODS EXCEL DESTINATION

At times, you might need functionality that currently does not exist with the ODS Excel destination to add a graphic image to a worksheet. For example, you might want to add a logo in a specified position (under both UNIX and Windows) or add a pivot table or other functionality which currently does not exist with the ODS Excel destination. To do that, you can modify an existing XLSX file with the Apache POI API. *Apache POI* is a Java API that enables you to manipulate Microsoft Office Open XML (OOXML) files. You can use POI to read, write, and modify Excel files as well as other file formats. The Excel destination generates OOXML files, specifically XLSX files, which are the focus of this paper.

The next example uses a call to the POI API, so you need to you need to acquire certain JAR files and set various CLASS paths in order to use it. You can download the CLASS file SGF_IMAGE (used in the example) from **ftp.sas.com/techsup/download/base/SGF2018-SAS2174.zip**. You can also compile the SGF_IMAGE.JAVA file and use it to generate the CLASS file. To download the example Java code, click here. For assistance in setting up the POI Library, see **https://poi.apache.org/.** A good

Apache POI tutorial is available at **www.tutorialspoint.com/apache_poi/apache_poi_java_excel.htm**.

The following example passes parameters (in this case, an Excel workbook, an image, and a saved file) to the Java CLASS file SGF_IMAGE:

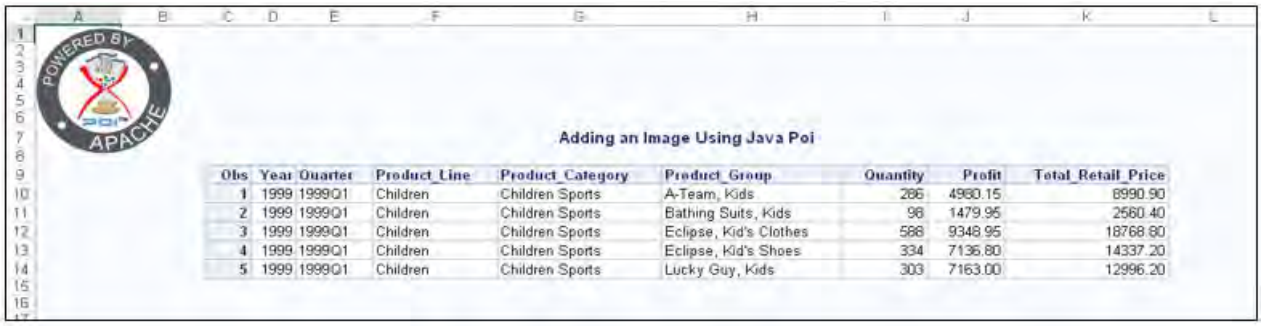
Example 18

```
ods excel file="c:\temp\addimage.xlsx" options(start_at="3,7"
                                               sheet_name="Java_image"
                                               embedded_titles="yes");

   proc print data=sashelp.orsales(obs=5);
      title "Adding an Image Using Java Poi";
   run;

ods excel close;
dm 'x "Java SGF_IMAGE c:\temp\addimage.xlsx c:\temp\apache.png
     c:\temp\image_added.xlsx";';
```

In This Example

- The Excel destination adds the XLSX file.
- The X statement passes the Java CLASS file and the parameters (for example, class name, the workbook to update, the image file, and the saved workbook) to a command line.
- The workbook, an image and saved files are passed to the SGF_IMAGE CLASS file. The CLASS file then updates the worksheet with the image and exports to a saved file (in this example, image_added.xlsx.

Output



Obs	Year	Quarter	Product_Line	Product_Category	Product_Group	Quantity	Profit	Total_Retail_Price
1	1999	1999Q1	Children	Children Sports	A-Team, Kids	286	4980.15	8990.90
2	1999	1999Q1	Children	Children Sports	Bathing Suits, Kids	98	1479.95	2560.40
3	1999	1999Q1	Children	Children Sports	Eclipse, Kid's Clothes	588	9348.95	18768.80
4	1999	1999Q1	Children	Children Sports	Eclipse, Kid's Shoes	334	7136.80	14337.20
5	1999	1999Q1	Children	Children Sports	Lucky Guy, Kids	303	7163.00	12996.20

Output 12. Logo Added by Using the Java POI API

AUTOMATING THE GENERATION AND SENDING OF SAS® OUTPUT

You can automate the process of generating output and emailing it by creating Excel output and sending email from that output in a single step. However, there are some methods you can use to accomplish the same task. For example, you cane the FILENAME statement with the EMAIL access method to send email after you create the output using the destination. You can also use the Excel destination with other access methods (for example, WEBDAV). In addition, you can use a SAS® stored process with the Excel destination.

When you automate Excel output, you might want to perform tasks such as generating email (with the Excel destination) and then emailing it as an Excel file, all in the same step. You can perform this task by using the EMAIL access method in the FILENAME statement in conjunction with the Excel destination. Because the destination uses the Office Open XML format, you need to use the correct Multipurpose Internet Mail Extension (MIME) type when you email the Excel document so that the mail server can handle the file correctly. Failing to do this might generate an error saying that you have an unrecognized format.

The following example uses the Excel destination to create a file, and it uses the EMAIL access method to automate the sending of the file.

Example 19.

```
  Ods noresults;

  data one;
     input Email $18. Units Amount;
     cards;
  john.doe@att.com  1000  50000
  john.doe@att.com  2000 100000
  bob.smith@att.com 2000 100000
  bob.smith@att.com 4000 200000
  jim.jones@att.com 5000 250000
  ken.lee@att.com   3000 150000
  ;
  run;

%macro send(email);

ods excel file="c:\temp\%sysfunc(scan(&email,1,'@')).xlsx";
   proc print data=one(where=(email="&email")) noobs;
   sum units amount;
   run;
ods excel close;

filename mailer email to="&email" subject="Report dated &sysdate"
                  attach=("c:\temp\%sysfunc(scan(&email,1,'@')).xlsx"
                  content-type="application/vnd.openxmlformats-
                  officedocument.spreadsheetml.sheet");

  data _null_;
     file mailer;
     put "Please find the budget numbers for %sysfunc(date(),date.)";
  run;

%mend;

proc sort data=one out=final;
   by email;
run;

data _null_;
   set final;
   by email;
   if first.email then call execute('%send('||email||')');
run;
```

In This Example

- The DATA step uses FIRST. Logic to pass each unique value of the email address to the macros as a parameter with the help of the CALL EXECUTE statement.
- The macro creates an Excel workbook for each unique email address and subsets the output based on the email address.
- The FILENAME statement contains the EMAIL access method, which enables the file to be mailed automatically.

CONCLUSION

As you can see, the ODS Excel destination is a very powerful destination. It enables you to add styles in a variety of ways (for example, by using PROC TEMPLATE and style overrides). The Excel destination

can also perform advanced formatting with the help of cascading style sheets. This paper also addressed common issues that occur with text wrapping, memory management, and image insertion. The discussion also covered using the ODS Excel destination with email. After taking this deep dive with a Technical Support Guy, you should be able to use the techniques that are described in this paper to create your own effective worksheets.

REFERENCES

Tutorials Point. 2018. "Apache POI Tutorial." Hyderabad, India. Available at **www.tutorialspoint.com/apache_poi/index.htm**.

Parker, Chevell. 2016. "A Ringside Seat: The ODS Excel Destination versus the ODS ExcelXP Tagset." *Proceedings of the SAS Global Forum 2016 Conference.* Cary, NC: SAS Institute Inc. Available at **www.sas.com/content/dam/SAS/support/en/technical-papers/SAS5642-2016.pdf**.

Parker, Chevell. 2015. "Staying Relevant in a Competitive World: Using the SAS Output Delivery System to Enhance, Customize, and Render Reports." *Proceedings of the SAS Global Forum 2015 Conference*. Cary, NC: SAS Institute Inc. Available at **support.sas.com/resources/papers/proceedings15/SAS1880-2015.pdf**.

Smith, Kevin D. 2013. "Cascading Style Sheets: Breaking Out of the Box of ODS Styles." Proceedings of the SAS Global Forum 2013 Conference. Cary, NC: SAS Institute Inc. Available at **support.sas.com/resources/papers/proceedings13/365-2013.pdf**.

RESOURCES

Eslinger, Jane. 2017. "The REPORT Procedure and ODS Destination for Microsoft Excel: The Smarter, Faster Way to Create First-Rate Reports." Proceedings of the SAS Global Forum 2017 Conference. Cary, NC: SAS Institute Inc. Available at **www.sas.com/content/dam/SAS/support/en/technical-papers/SAS0235-2017.pdf**.

SAS Institute Inc. 2016. SAS Note 57766, "Add multiple tables and graphs to each worksheet using the ODS Destination for Excel." Available at **support.sas.com/kb/57/766.html**.

CONTACT INFORMATION

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

Chevell Parker
SAS Institute Inc.
SAS Campus Drive
Cary, NC 27513

Email: **support@sas.com**
Web: **support.sas.com**