PharmaSUG 2014 - Paper HT08-SAS

Creating Multi-Sheet Microsoft Excel Workbooks with SAS®: The Basics and Beyond Part 1

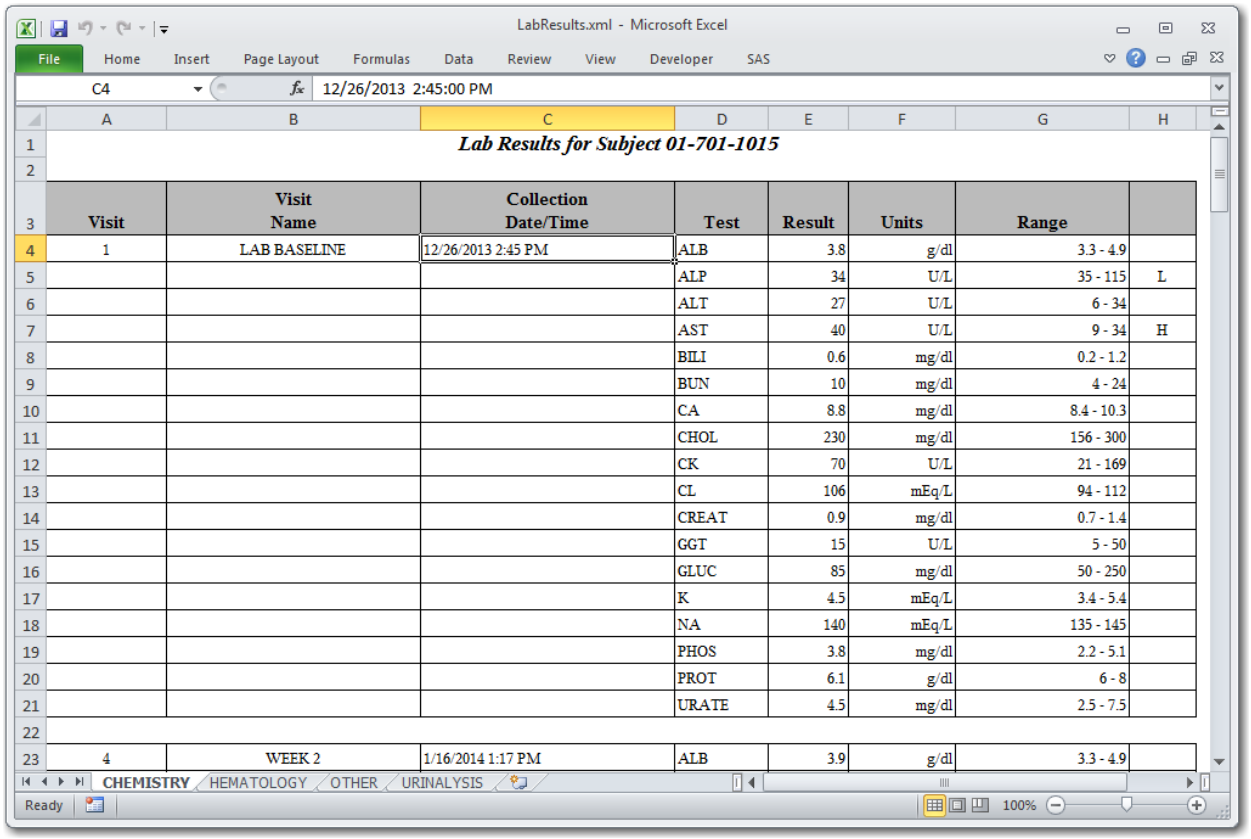
Vincent DelGobbo, SAS Institute Inc., Cary, NC

ABSTRACT

This presentation explains how to use Base SAS®9 software to create multi-sheet Microsoft Excel workbooks. 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. The techniques 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 presentations by this author, this presentation contains new and revised material not previously presented.

INTRODUCTION

This paper explains how to use Base SAS 9.1.3 or later to create the Excel workbook shown in Figure 1.



Visit	Visit Name	Collection Date/Time	Test	Result	Units	Range	
1	LAB BASELINE	12/26/2013 2:45 PM	ALB	3.8	g/dl	3.3 - 4.9	
			ALP	34	U/L	35 - 115	L
			ALT	27	U/L	6 - 34	
			AST	40	U/L	9 - 34	H
			BILI	0.6	mg/dl	0.2 - 1.2	
			BUN	10	mg/dl	4 - 24	
			CA	8.8	mg/dl	8.4 - 10.3	
			CHOL	230	mg/dl	156 - 300	
			CK	70	U/L	21 - 169	
			CL	106	mEq/L	94 - 112	
			CREAT	0.9	mg/dl	0.7 - 1.4	
			GGT	15	U/L	5 - 50	
			GLUC	85	mg/dl	50 - 250	
			K	4.5	mEq/L	3.4 - 5.4	
			NA	140	mEq/L	135 - 145	
			PHOS	3.8	mg/dl	2.2 - 5.1	
			PROT	6.1	g/dl	6 - 8	
			URATE	4.5	mg/dl	2.5 - 7.5	
4	WEEK 2	1/16/2014 1:17 PM	ALB	3.9	g/dl	3.3 - 4.9	

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

The workbook includes four worksheets containing fictional lab results data over time for a single patient. An Excel format, not a SAS format, is used to control the appearance of the datetime 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 "Creating Multi-Sheet Microsoft Excel Workbooks with SAS®: The Basics and Beyond Part 1".

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).

LIMITATIONS

Because the ExcelXP ODS tagset creates files that conform to the Microsoft XML Spreadsheet Specification, you can create multi-sheet Excel workbooks that contain the output from almost any SAS procedure. The exception is that the Microsoft XML Spreadsheet Specification does not support images, so the output from graphics 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.

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

Table 1 presents an abbreviated list of column properties for the LabResults SAS table that is used to create the Excel workbook shown in Figure 1. An asterisk (*) is used as a split character in some variable labels to control text wrapping in the column headings, and the values in the VISIT_DATETIME column are SAS datetime values.

Column Name	Column Label	Column Type	Typical Values
LBCAT	Category for Lab Test	Character	CHEMISTRY, HEMATOLOGY, OTHER, URINALYSIS
VISITNUM	Visit	Numeric	1, 4, 8, 10, 13
VISIT	Visit*Name	Character	LAB BASELINE, WEEK 2, WEEK 4, WEEK 26
VISIT_DATETIME	Collection*Date/Time	Numeric	1703688300, 1707828960, 1719920700
LBTESTCD	Test	Character	BILI, CHOL, EOS, WBC
LBORRES	Result	Character	0.05, 0.9, 1.013, 8.8, 15, 140
LBORRESU	Units	Character	%, MILL/uL, NO UNITS, g/dL, pg/ml
RANGE	Range	Character	1.006 - 1.03, 21 - 169, 156 - 300, not available
LBNRIND	<blank>	Character	A, H, L, N

Table 1. Column Properties and Representative Data Values for the LabResults SAS Table

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 2014b). 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 enter the path and filename 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.

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.

UPDATING THE EXCELXP TAGSET

The version of the ExcelXP tagset that is shipped with Base SAS is periodically updated. 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 (SAS Institute Inc. 2014a).

Submit this code to display the tagset version number in the SAS log:

```
filename temp temp;

ods tagsets.ExcelXP file=temp;
ods tagsets.ExcelXP close;

filename temp clear;
```

All the code in this paper uses an up-to-date version of the ExcelXP tagset. If you're using a tagset that's more than 2 or 3 versions old, consider upgrading by following the steps below. Otherwise, continue to the next section.

The first step in upgrading your tagset is to define the location where the tagset will be stored on your system with these two statements:

```
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`, the compiled tagset is stored 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;
```

Once you have issued the appropriate ODS PATH statement (❷), 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 "Creating Multi-Sheet Microsoft Excel Workbooks with SAS®: The Basics and Beyond Part 1". The download package contains a file named ExcelXP.sas that contains the SAS code for compiling 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 compiled and stored in the directory corresponding to the MYLIB library. To give all of your future SAS programs access to the tagset, specify "read" access in the LIBNAME and in the ODS PATH statements:

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

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

See SAS Usage Note 32394 for additional information about updating tagsets (SAS Institute Inc. 2014b).

USING ODS TO CREATE THE MULTI-SHEET EXCEL WORKBOOK

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

```
ods _all_ close;

❶ ods tagsets.ExcelXP file='LabResults.xml' style=Printer;

title 'Lab Results for Subject 01-701-1015';
footnote;

❷ proc report data=sample.LabResults nowd split='*';
  by lbcat;

  column visitnum  visit     visit_datetime  lbtestcd  lborres
         lborresu  lbornrlo  lbornrhi        range     lbnrind;

  define visitnum       / order    center 'Visit';
  define visit          / order    center 'Visit*Name';
  define visit_datetime / order    left   'Collection*Date/Time'
                                   format=e8601dt.;
  define lbtestcd       / display  left   'Test';
❸ define lborres        / display  right  'Result';
  define lborresu       / display  right  'Units';
  define lbornrlo       / noprint;
  define lbornrhi       / noprint;
  define range          / computed right  'Range';
  define lbnrind        / display  center '';

❹ compute range / char length=13;
    if (compress(lbornrlo) ne '' and compress(lbornrhi) ne '')
      then range = strip(lbornrlo) || ' - ' || strip(lbornrhi);
      else range = 'not available';
  endcomp;

❺ compute lbnrind;
    if (upcase(lbnrind) eq 'N') then lbnrind = '';
  endcomp;
```

```
❻   compute after visitnum;
      line '';
    endcomp;
  run; quit;

❼ 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. By default, the ExcelXP tagset creates a new worksheet when a SAS procedure creates new tabular output. PROC REPORT (❷) is run with a BY statement and creates four tables, one for each distinct value of the LBCAT variable. Each table is created in a separate worksheet.

The COLUMN statement specifies the order to display the columns, and the column roles, justification, labels, and formats are specified in the DEFINE statements (❸). The LBORNRLO and LBORNRHI columns are not displayed in the output, but are used to compute the value of the RANGE column.

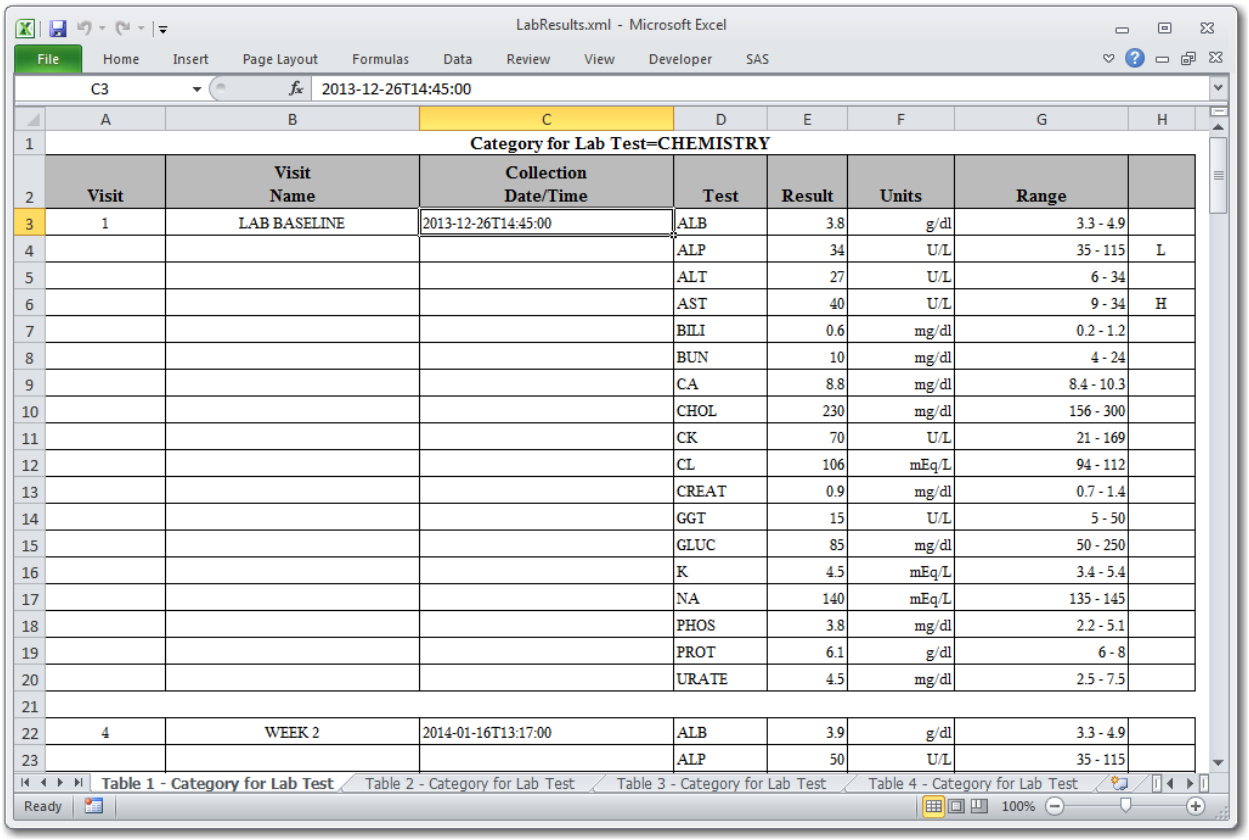
The first COMPUTE block (❹) constructs the range if both low and high values are available for the test, otherwise "not available" is used for the value. The second block (❺) suppresses the display of the LBNRIND value for tests with normal results, and the third block (❻) inserts a blank line between office visits.

The last ODS statement (❼) closes the ExcelXP destination and releases the XML file so that it can be opened with Excel.

Figure 2 displays the results of executing the basic SAS code, and then opening the resulting LabResults.xml file with Excel. Notice that Figure 2 does not match Figure 1. The following problems are exhibited in Figure 2:

1. Unattractive, default worksheet names are used.
2. Title text is missing.
3. Standard BY line text ("Category for Lab Test=CHEMISTRY") precedes the table.
4. The collection date and time values are displayed differently.

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".



	A	B	C	D	E	F	G	H
1	Category for Lab Test=CHEMISTRY							
2	Visit	Visit Name	Collection Date/Time	Test	Result	Units	Range	
3	1	LAB BASELINE	2013-12-26T14:45:00	ALB	3.8	g/dl	3.3 - 4.9	
4				ALP	34	U/L	35 - 115	L
5				ALT	27	U/L	6 - 34	
6				AST	40	U/L	9 - 34	H
7				BILI	0.6	mg/dl	0.2 - 1.2	
8				BUN	10	mg/dl	4 - 24	
9				CA	8.8	mg/dl	8.4 - 10.3	
10				CHOL	230	mg/dl	156 - 300	
11				CK	70	U/L	21 - 169	
12				CL	106	mEq/L	94 - 112	
13				CREAT	0.9	mg/dl	0.7 - 1.4	
14				GGT	15	U/L	5 - 50	
15				GLUC	85	mg/dl	50 - 250	
16				K	4.5	mEq/L	3.4 - 5.4	
17				NA	140	mEq/L	135 - 145	
18				PHOS	3.8	mg/dl	2.2 - 5.1	
19				PROT	6.1	g/dl	6 - 8	
20				URATE	4.5	mg/dl	2.5 - 7.5	
21								
22	4	WEEK 2	2014-01-16T13:17:00	ALB	3.9	g/dl	3.3 - 4.9	
23				ALP	50	U/L	35 - 115	

Figure 2. Initial ODS ExcelXP Tagset-Generated Workbook

UNDERSTANDING AND USING THE EXCELXP TAGSET OPTIONS

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 file='file-name.xml' style=style-name ... ;
  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;
ods tagsets.ExcelXP close;
```

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 in the initial ODS statement.

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;

filename temp clear;
```

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.

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

USING BY GROUP VALUES IN THE WORKSHEET NAMES

ODS generates a unique name for each worksheet, as required by Excel. Figure 2 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.

The SHEET_INTERVAL option controls the interval at which SAS output is placed into worksheets, and SHEET_LABEL is used to specify the prefix for the worksheet names. When used together, the current value of the first BY variable is used in the worksheet name. The following code causes the worksheet names to match those shown in Figure 1:

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

title '...'; footnote;

ods tagsets.ExcelXP options(sheet_interval='bygroup'
                            sheet_label=' ');

proc report data=sample.LabResults nowd split='*';
  by lbcat;
  column ... ;
  define ... ;
  compute ... ;
run; quit;

ods tagsets.ExcelXP close;
```

The blank space between the quotation marks for the SHEET_LABEL option suppresses the printing of a prefix. If you want a particular text string to precede the BY variable value, specify that text between the quotation marks.

Because tagset options remain in effect until their value is changed or the destination is closed, the SHEET_INTERVAL, SHEET_LABEL, and any other options specified in this ODS statement affect all four worksheets,

INCLUDING TITLE 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 3.

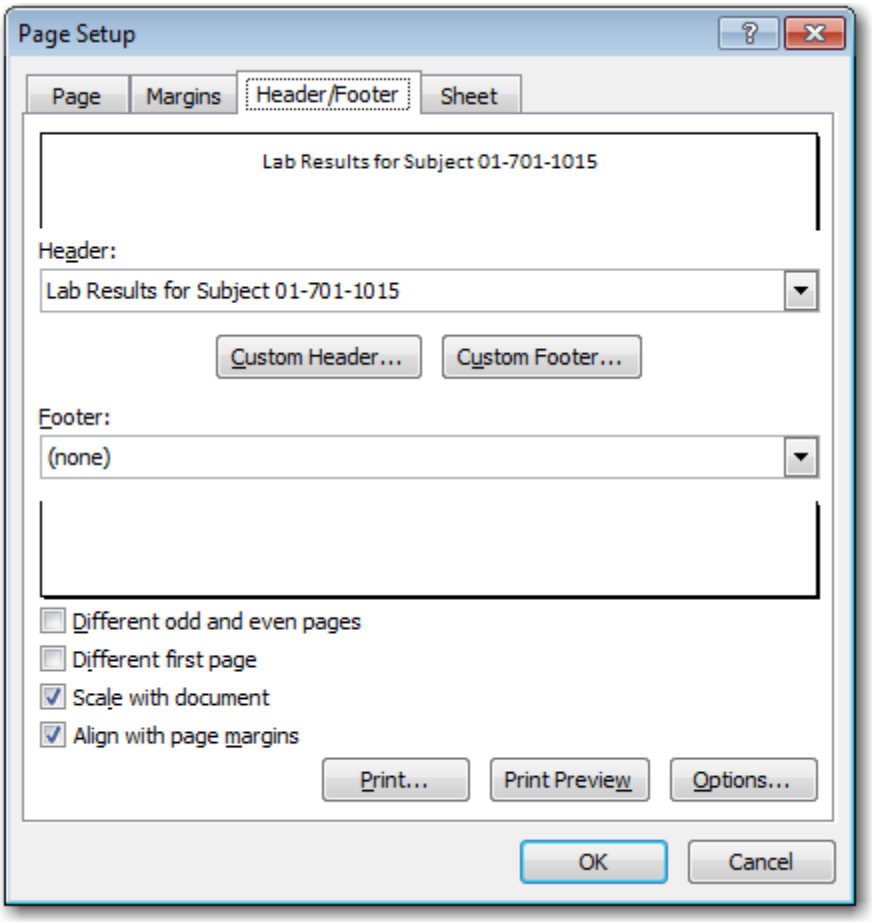


Figure 3. Excel Page Setup Dialog Box Showing Title Text

To include title text on-screen, in the worksheet body, use the EMBEDDED_TITLES option:

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

title '...'; footnote;

ods tagsets.ExcelXP options(sheet_interval='bygroup'
                            sheet_label=' '
                            embedded_titles='yes');

proc report data=sample.LabResults nowd split='*';
  by lbcat;
  column ... ;
  define ... ;
  compute ... ;
run; quit;

ods tagsets.ExcelXP close;
```

A partial view of the resulting output is shown in Figure 4.

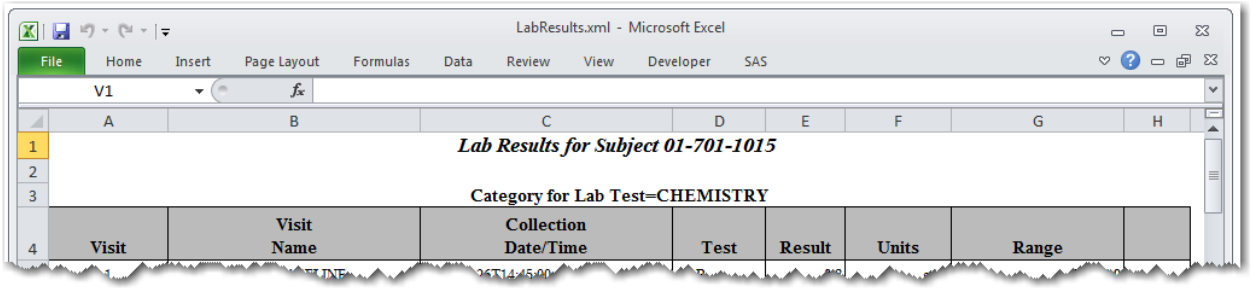


Figure 4. ODS ExcelXP Tagset-Generated Workbook with TITLE Statement Text in the Document

SUPPRESSING THE BY LINE TEXT

BY line text appears in the worksheets because the REPORT procedure is executed with a BY statement. However, this text is redundant because the BY value is displayed in the worksheet name. To omit the BY line text, specify the SUPPRESS_BYLINES option in the ODS statement:

```
ods tagsets.ExcelXP options(sheet_interval='bygroup'
                            sheet_label=' '
                            embedded_titles='yes'
                            suppress_bylines='yes');
```

Do not use the NOBYLINE *system* option, because this disables BY group processing in the ExcelXP tagset, even though the SHEET_INTERVAL tagset option is set to "bygroup".

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 5 shows the locations that are pertinent to the REPORT procedure output (SAS Institute Inc. 2008). 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 REPORT code alters 3 attributes of the COLUMN location for the MYVAR variable:

```
define myvar / style(column)=[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. Because of these drawbacks, inline style overrides 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 reuse. 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. 2012).

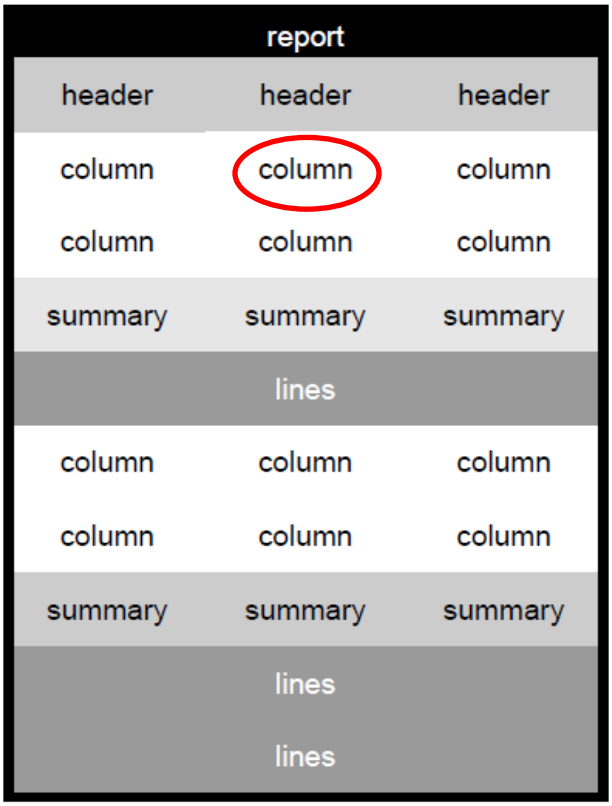


Figure 5. Style Locations for the REPORT Procedure

CHANGING THE EXCEL DATA TYPE

SAS and Microsoft Excel use different date and time systems (Microsoft Corporation 2014a, SAS Institute Inc. 2013). Consequently, you often encounter problems when Excel reads SAS output containing datetime values.

One way to correct this behavior is to write SAS code to convert numeric SAS datetime values to numeric Excel datetime values, but this approach is problematic because you must alter your original SAS data. While you can create a new SAS view or table that contains the new datetime values, this is a vector for errors and becomes inefficient as your data grows.

A better solution, one that does not require you to alter the underlying data, is to use a combination of SAS and Excel formats. First you specify a SAS format in the procedure code, and then you specify an Excel format using a style override. The SAS format controls what is physically written into the XML file, and the Excel format changes the way the value is displayed.

Because Excel expects datetime values to be represented using the ISO 8601 format (yyyy-mm-ddThh:mm:ss), SAS datetime values should be formatted using the IS8601DT format. However, open the LabResults.xml file using a text editor and you will see that the datetime cells are specified to be **String** type:

```
<Cell ss:StyleID="data__l" ss:Index="3">
  <Data ss:Type="String">2013-12-26T14:45:00</Data>
</Cell>
```

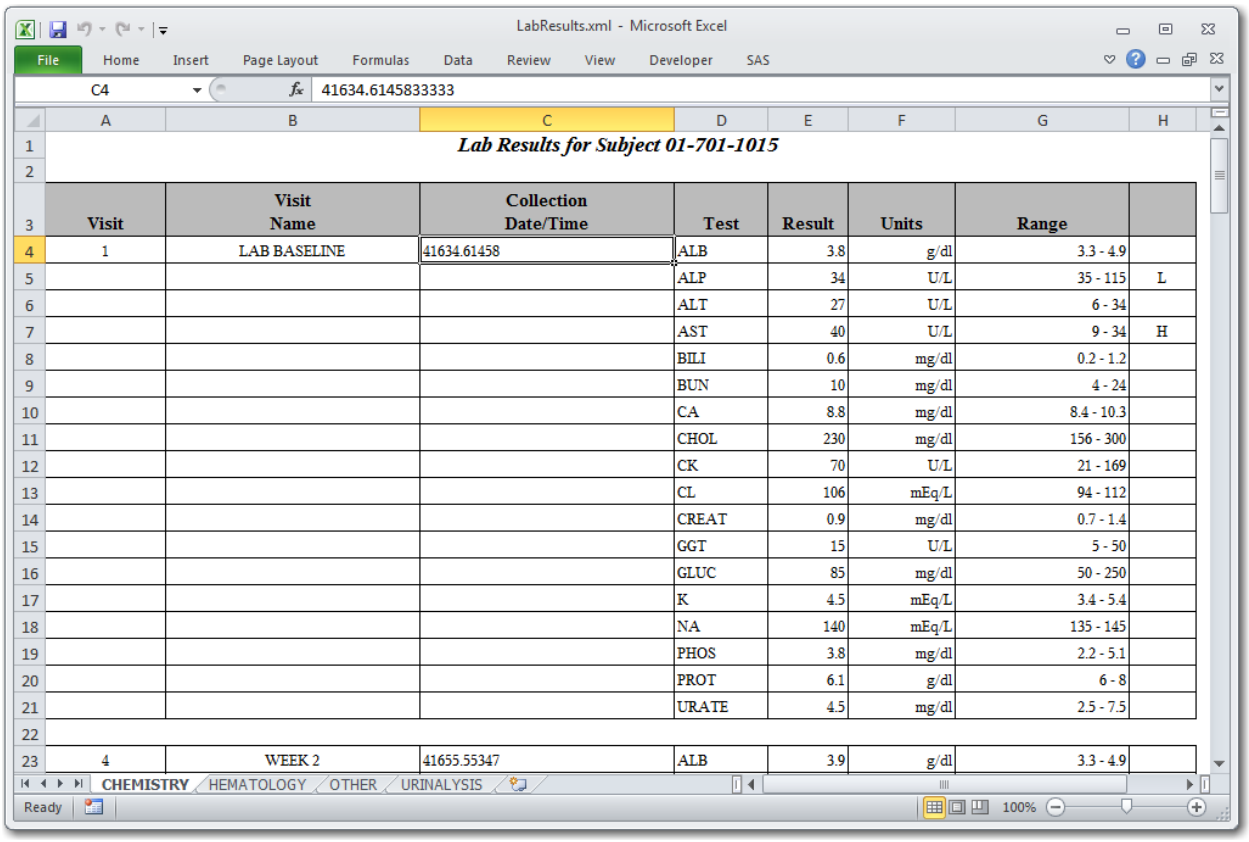
Because Excel does not recognize the value as a datetime value, you cannot apply Excel formats or perform mathematical operations on the data. To fix this problem, use the TAGATTR attribute in a style override to specify that the value is an Excel **DateTime** type:

```
proc report data=sample.LabResults nowd split='*';
  by lbcat;

  column ...;

  define ...;
  define visit_datetime / order left 'Collection*Date/Time' format=e8601dt.
    style(column)=[tagattr='type:DateTime'];
  define ...;

  compute ... ;
run; quit;
```

The value specified in the style override, **`DateTime`**, is case-sensitive. Datetime values now have the correct type specified in the LabResults.xml file:

```
<Cell ss:StyleID="data__l" ss:Index="3">
  <Data ss:Type="DateTime">2013-12-26T14:45:00</Data>
</Cell>
```

The unformatted Excel datetime values are shown in Figure 6.



Visit	Visit Name	Collection Date/Time	Test	Result	Units	Range	
1	LAB BASELINE	41634.61458	ALB	3.8	g/dl	3.3 - 4.9	
			ALP	34	U/L	35 - 115	L
			ALT	27	U/L	6 - 34	
			AST	40	U/L	9 - 34	H
			BILI	0.6	mg/dl	0.2 - 1.2	
			BUN	10	mg/dl	4 - 24	
			CA	8.8	mg/dl	8.4 - 10.3	
			CHOL	230	mg/dl	156 - 300	
			CK	70	U/L	21 - 169	
			CL	106	mEq/L	94 - 112	
			CREAT	0.9	mg/dl	0.7 - 1.4	
			GGT	15	U/L	5 - 50	
			GLUC	85	mg/dl	50 - 250	
			K	4.5	mEq/L	3.4 - 5.4	
			NA	140	mEq/L	135 - 145	
			PHOS	3.8	mg/dl	2.2 - 5.1	
			PROT	6.1	g/dl	6 - 8	
			URATE	4.5	mg/dl	2.5 - 7.5	
4	WEEK 2	41655.55347	ALB	3.9	g/dl	3.3 - 4.9	

Figure 6. ODS ExcelXP Tagset-Generated Workbook with Excel Datetime Values

APPLYING AN EXCEL FORMAT TO THE DATETIME VALUES

Use Excel formats, not SAS formats, to control the appearance of display values in Excel. Table 2 shows how the Excel datetime value for January 16, 2014 at 1:17 pm is displayed in Excel using different formats.

Excel Format	Display Value
m/d/yyyy h:mm AM/PM	1/16/2014 1:17 PM
mm/dd/yy hh:mm AM/PM	01/16/14 01:17 PM
ddmmmyyy:hh:mm:ss	16Jan2014:13:17:00
mm/dd/yy	01/16/14
hh:mm AM/PM	01:17 PM
m/d/e	1/16/2014
yyyy-mm-ddThh:mm:ss	2014-01-16T13:17:00
mmm d, yyyy \@ h:mm AM/PM	Jan 16, 2014 @ 1:17 PM
mmmm d, yyyy \@ h:mm a/p	January 16, 2014 @ 1:17 p
ddd mmmmm d, yyyy	Thu J 16, 2014
dddd mmm d, yyyy	Thursday January 16, 2014
General	41655.55347
#,###.0000	41,655.5535

Table 2. Excel Formats and Corresponding Display Values

You can also use the TAGATTR attribute to apply an Excel format to the data:

```
proc report data=sample.LabResults nowd split='*';
  by lbcat;

  column ...;

  define ...;
  define visit_datetime / order left 'Collection*Date/Time' format=e8601dt.
    style(column)=[tagattr='type:DateTime format:m/d/yyyy h:mm AM/PM'];
  define ...;

  compute ... ;
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 file='LabResults.xml' style=Printer;

title 'Lab Results for Subject 01-701-1015';
footnote;

ods tagsets.ExcelXP options(sheet_interval='bygroup'
                            sheet_label=' '
                            embedded_titles='yes'
                            suppress_bylines='yes');

proc report data=sample.LabResults nowd split='*';
  by lbcat;

  column visitnum  visit     visit_datetime  lbtestcd  lborres
         lborresu  lbornrlo  lbornrhi        range     lbnrind;

  define visitnum       / order    center 'Visit';
  define visit          / order    center 'Visit*Name';
  define visit_datetime / order    left   'Collection*Date/Time'
    format=e8601dt.
    style(column)=[tagattr='format:m/d/yyyy h:mm AM/PM type:DateTime'];
  define lbtestcd       / display  left   'Test';
  define lborres        / display  right  'Result';
  define lborresu       / display  right  'Units';
  define lbornrlo       / noprint;
  define lbornrhi       / noprint;
  define range          / computed right  'Range';
  define lbnrind        / display  center '';
```

```
  compute range / char length=13;
    if (compress(lbornrlo) ne '' and compress(lbornrhi) ne '')
      then range = strip(lbornrlo) || ' - ' || strip(lbornrhi);
      else range = 'not available';
  endcomp;

  compute lbnrind;
    if (upcase(lbnrind) eq 'N') then lbnrind = '';
  endcomp;

  compute after visitnum;
    line '';
  endcomp;
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 7 shows a Web page that can deliver SAS output directly to Excel, using a Web browser as the client.

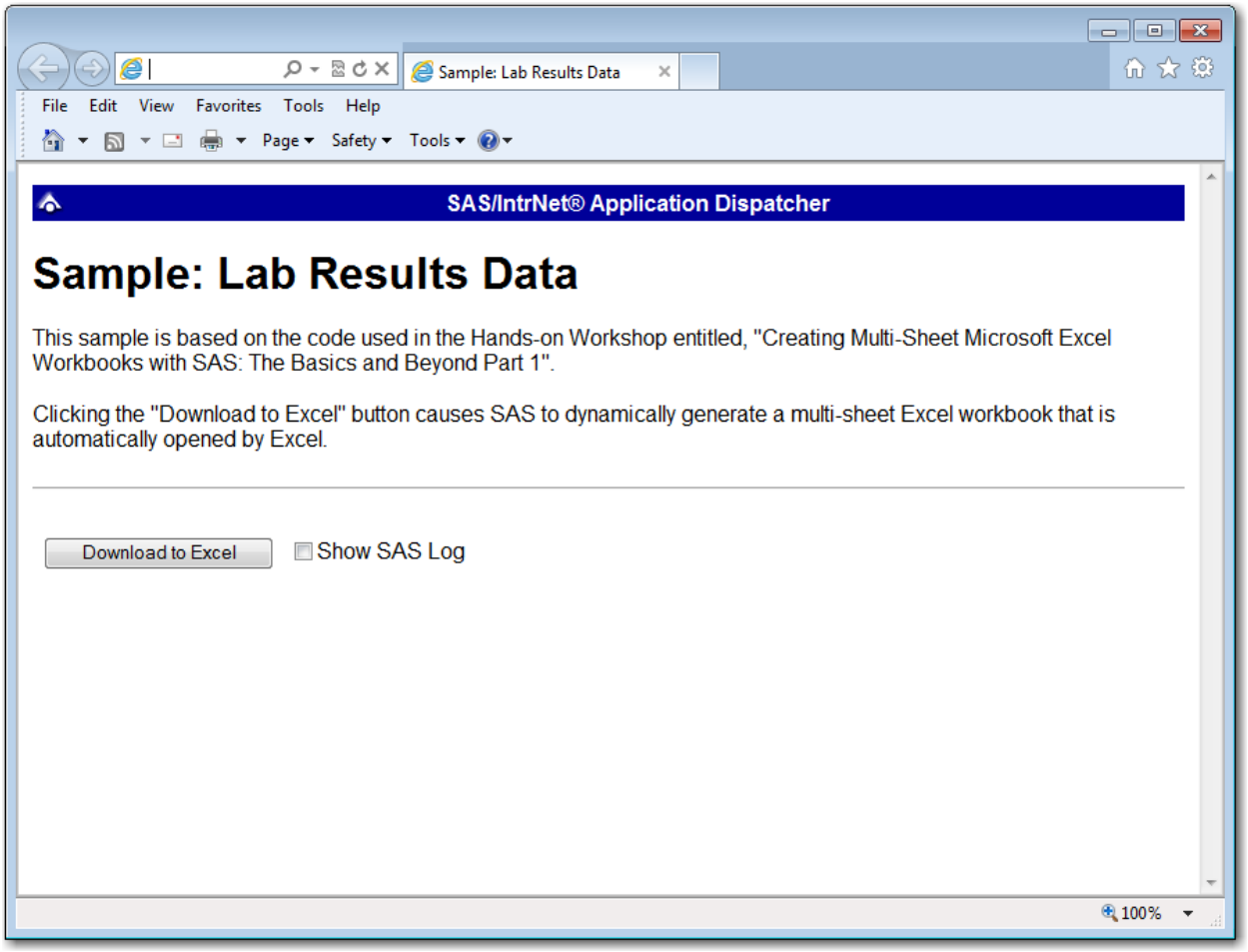


Figure 7. Web Page to Drive a SAS/IntrNet Application

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=
"LabResults.xml"));  * Ignore line wrapping;

ods _all_ 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 filename to be displayed in the File Download dialog box. As you can see in Figure 8, the filename appears as **LabResults.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 8. 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 you are presented with the File Download dialog box (Figure 8). You can then 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, 2011b), as well as this author's earlier papers (DelGobbo 2002, 2003, 2004).

CONCLUSION

The SAS 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

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://msdn.microsoft.com/en-us/library/aa140066(office.10).aspx.

Microsoft Corporation. 2014a. "About dates and date systems". Available at http://office.microsoft.com/en-us/excel-help/about-dates-and-date-systems-HP005200674.aspx.

Microsoft Corporation. 2014b. "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 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. 2012. *SAS® 9.3 Output Delivery System: User's Guide, Second Edition.* Cary, NC: SAS Institute Inc. Available at http://support.sas.com/documentation/cdl/en/odsug/65308/HTML/default/viewer.htm#n19a4b40swc766n18qczor47r08f.htm.

SAS Institute Inc. 2013. *SAS® 9.4 Language Reference: Concepts, Second Edition.* Cary, NC: SAS Institute Inc. Available at http://support.sas.com/documentation/cdl/en/lrcon/67227/HTML/default/viewer.htm#p1wj0wt2ebe2a0n1lv4lem9hdc0v.htm.

SAS Institute Inc. 2014a. "ODS MARKUP Resources". Available at http://support.sas.com/rnd/base/ods/odsmarkup.

SAS Institute Inc. 2014b. "Usage Note 32394: Installing and Storing Updated Tagsets for ODS MARKUP". Available at http://support.sas.com/kb/32/394.html.

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. "An Introduction Creating Multi-Sheet Microsoft Excel Workbooks the Easy Way with SAS®". *Proceedings of the SAS Global Forum 2012 Conference*. Cary, NC: SAS Institute Inc. Available at http://support.sas.com/resources/papers/proceedings12/150-2012.pdf.

DelGobbo, Vincent. 2013. "Some Techniques for Integrating SAS Output with Microsoft Excel Using Base SAS®". *Proceedings of the SAS Global Forum 2013 Conference*. Cary, NC: SAS Institute Inc. Available at http://support.sas.com/resources/papers/proceedings13/143-2013.pdf.

DelGobbo, Vincent. 2014. "Vince DelGobbo's ExcelXP Tagset Paper Index". Available at http://support.sas.com/community/events/sastalks/presentations/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.
100 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.