Paper 136-30

Moving Data and Analytical Results between SAS® and Microsoft Office

Vincent DelGobbo, SAS Institute Inc., Cary, NC

ABSTRACT

Transferring data between SAS and Microsoft Office can be difficult, especially when SAS is not installed on a Windows platform. This paper discusses using the HTML and XML support in Base SAS software to move data between SAS and Microsoft Office (versions 2002 and later). You can use the techniques described here regardless of the platform on which SAS software is installed.

INTRODUCTION

After reading this paper, you should be able to import most SAS output into a Microsoft Excel or Word document, as well as import Excel workbooks into SAS. The data and sample SAS code that's used in this paper reveal formatting issues that you might encounter when attempting to import HTML and XML output that's generated by the Output Delivery System (ODS) into Microsoft Office documents. The TEMPLATE procedure is used to correct these problems by modifying an ODS style. This paper also explains and provides code for importing Excel workbooks into SAS tables. All code was tested with SAS 9.1 and Microsoft Office XP Service Pack 3.

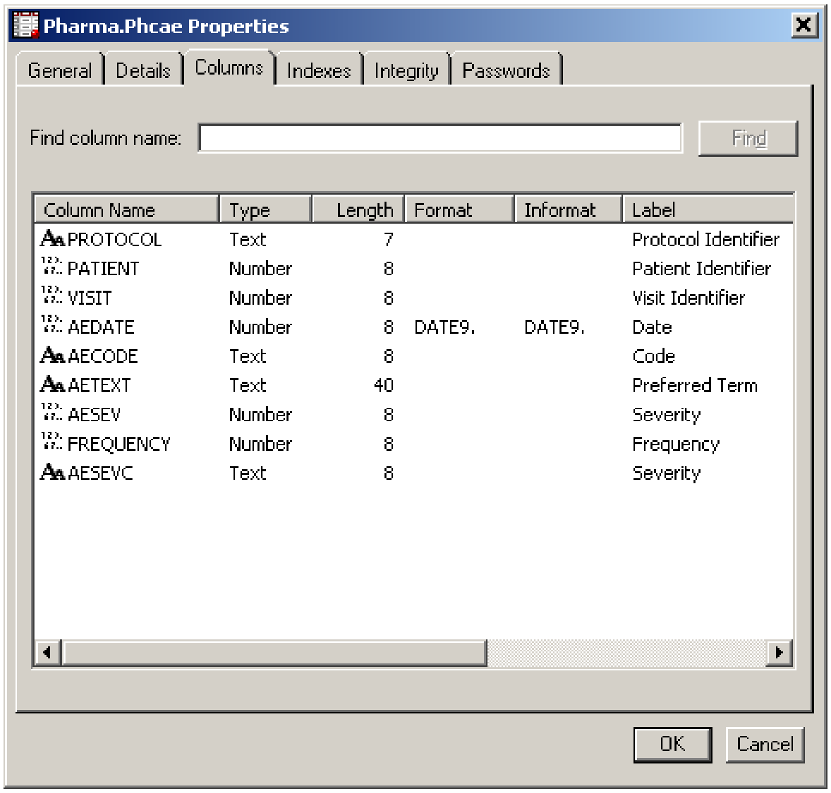
For information about generating other types of output and techniques for using that output in Excel and Word, see "Techniques for SAS Enabling Microsoft Office in a Cross-Platform Environment" in the *Proceedings of the Twenty-Seventh Annual SUGI Conference* (DelGobbo, 2002).

SAMPLE SAS DATA

Figure 1 shows the column properties of the data that we are going to move to Excel and Word. The data is adverse event information for a fictitious drug.

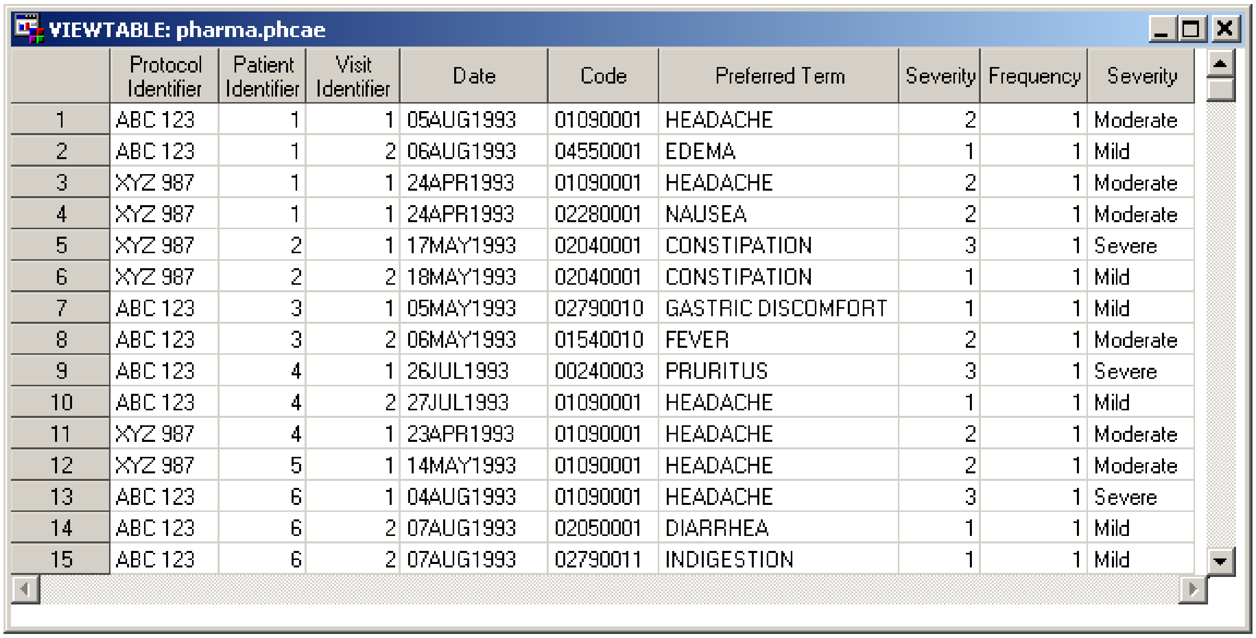
Notice that labels were applied to each column in order to create a more attractive table when imported into the Microsoft Office applications.

The two columns AESEV and AESEVC represent the severity of the adverse event. As illustrated in Figure 2, the same data is represented in two ways: the AESEV column contains a numeric value for the severity, and the AESEVC column contains a character value for the severity. Later in this paper, the SAS code uses both columns.



Column Name	Type	Length	Format	Informat	Label
PROTOCOL	Text	7			Protocol Identifier
PATIENT	Number	8			Patient Identifier
VISIT	Number	8			Visit Identifier
AEDATE	Number	8	DATE9.	DATE9.	Date
AECODE	Text	8			Code
AETEXT	Text	40			Preferred Term
AESEV	Number	8			Severity
FREQUENCY	Number	8			Frequency
AESEVC	Text	8			Severity

Figure 1. Columns in the SAS Adverse Event Table



VIEWTABLE: pharma.phcae

	Protocol Identifier	Patient Identifier	Visit Identifier	Date	Code	Preferred Term	Severity	Frequency	Severity
1	ABC 123	1	1	05AUG1993	01090001	HEADACHE	2	1	Moderate
2	ABC 123	1	2	06AUG1993	04550001	EDEMA	1	1	Mild
3	XYZ 987	1	1	24APR1993	01090001	HEADACHE	2	1	Moderate
4	XYZ 987	1	1	24APR1993	02280001	NAUSEA	2	1	Moderate
5	XYZ 987	2	1	17MAY1993	02040001	CONSTIPATION	3	1	Severe
6	XYZ 987	2	2	18MAY1993	02040001	CONSTIPATION	1	1	Mild
7	ABC 123	3	1	05MAY1993	02790010	GASTRIC DISCOMFORT	1	1	Mild
8	ABC 123	3	2	06MAY1993	01540010	FEVER	2	1	Moderate
9	ABC 123	4	1	26JUL1993	00240003	PRURITUS	3	1	Severe
10	ABC 123	4	2	27JUL1993	01090001	HEADACHE	1	1	Mild
11	XYZ 987	4	1	23APR1993	01090001	HEADACHE	2	1	Moderate
12	XYZ 987	5	1	14MAY1993	01090001	HEADACHE	2	1	Moderate
13	ABC 123	6	1	04AUG1993	01090001	HEADACHE	3	1	Severe
14	ABC 123	6	2	07AUG1993	02050001	DIARRHEA	1	1	Mild
15	ABC 123	6	2	07AUG1993	02790011	INDIGESTION	1	1	Mild

Figure 2. Partial View of the SAS Adverse Event Table

Notice that the Code column contains values that have leading zeros. By using an ODS style override, you can retain the leading zeros when importing the Code data into Excel. Microsoft Word retains leading zeros by default.

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 *tagset* controls the type of output that's generated (HTML, RTF, PDF, etc.); an ODS *style* controls the appearance of the output. In this paper, we use a tagset that creates HTML output that can be opened with Excel and Word, and a tagset that creates XML that can be opened by Excel. The latter tagset creates an Excel workbook that has multiple worksheets.

Extensible Markup Language (XML) defines and formats data for easy exchange. XML is similar to HTML in that it uses *tags*. Unlike HTML tags that control how data is rendered, XML tags describe the structure and meaning of data but do not control how it is rendered.

The ODS statements that generate content that's compatible with Microsoft Office follow this general form:

```
❶ ods listing close;
❷ ods TagsetName style=StyleName file= ... ;
     *  your SAS code here;
❸ ods TagsetName close;
```

The first ODS statement (❶) turns off the standard "line printer" ODS destination. We only want to generate HTML and XML output for use with Excel and Word.

The second ODS statement (❷) generates the HTML or XML output, depending on the tagset that you use, and stores the output in a file of your choosing. The STYLE option controls the appearance of the output, such as the font and color scheme. To see a list of ODS styles that are available for use at your site, submit the following SAS code:

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

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

Later in this paper, we discuss using both the ODS tagset named MSOffice2K to create HTML output and the ExcelXP tagset to create XML output from the PRINT and TABULATE procedures. In both cases, appearance characteristics such as colors and fonts are controlled by the ODS style named Banker.

Although you can store your output on a local disk (where SAS software is installed), a network-accessible disk, or a Web server, 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 HTML and XML files can be accessed by a Web browser in addition to Excel and Word.
- You can take advantage of Web-server authentication and security models.

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

OPENING ODS OUTPUT WITH EXCEL AND WORD

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

1. In Excel or Word, select File ➨ Open.
2. In the "File name" field, specify http://*Web-server*/*directory*/
   where *Web-server* corresponds to the domain name for your Web server.
3. Click Open. (You should see a list of files that are available on your Web server.)
4. Select the file that you want to open and click Open to import the HTML or XML file.

If you don't see a list of available files after step 3, you might have to configure your Web server to allow directory listing. Alternatively, you can specify the full path to the file that you want to open.

http://*Web-server*/*directory*/*filename*

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

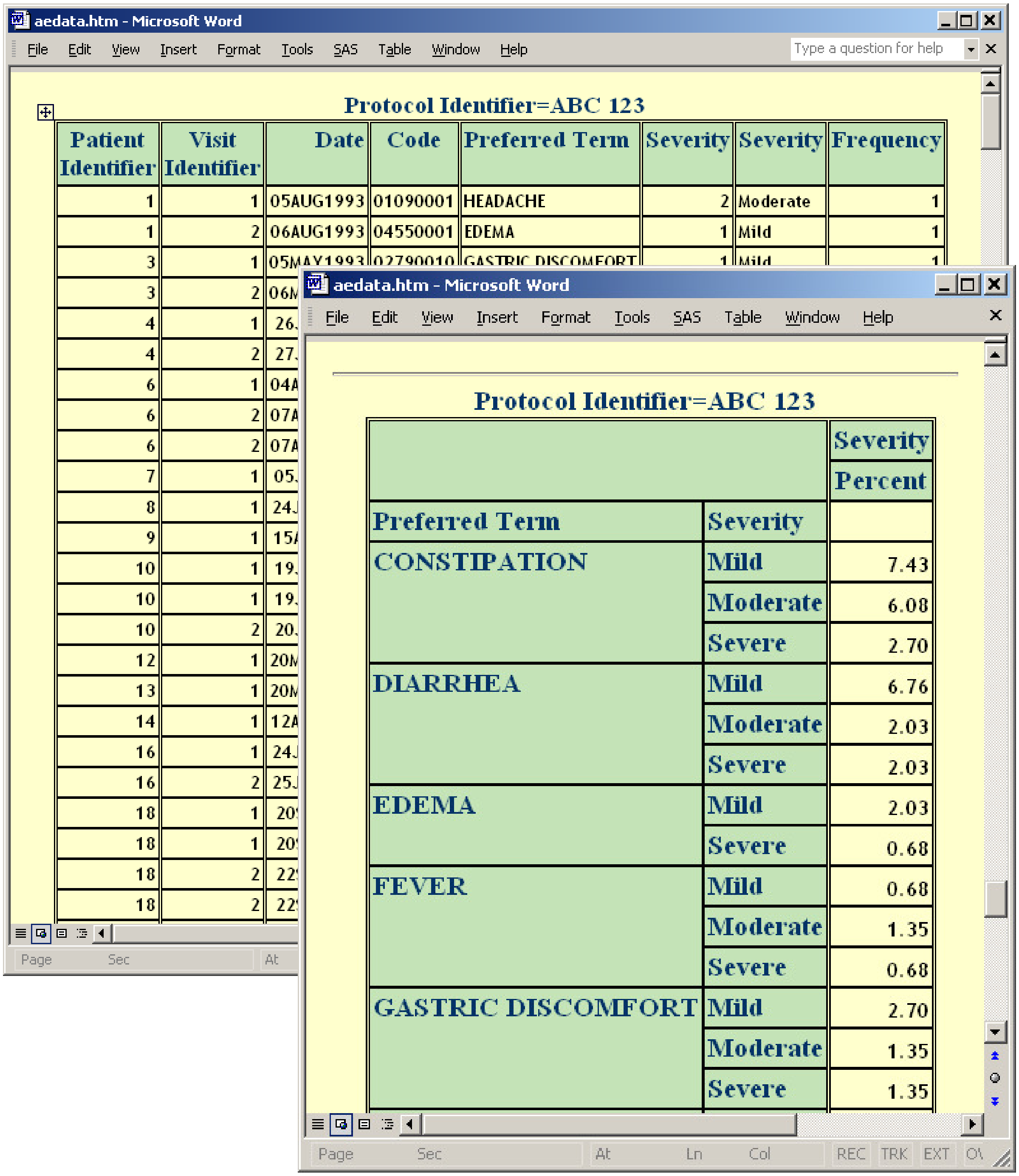
Excel and Word will read and convert the HTML or XML file (Excel only) to their respective native formats. After that, you can perform any Excel or Word function on the data.

USING ODS TO GENERATE HTML FOR EXCEL AND WORD

The following SAS code generates HTML output for use with Excel and Word:

```
ods listing close;
ods tagsets.MSOffice2K style=Banker file='aedata.htm';
  title; footnote;
  proc print data=pharma.phcae noobs label;
    by protocol;
    var patient visit aedate aecode aetext aesev aesevc frequency;
  run; quit;

  proc tabulate data=pharma.phcae;
    by protocol;
    var aesev;
    class aetext aesevc;
    table aetext*aesevc,aesev*pctn;
    keyword all pctn;
    keylabel pctn='Percent';
  run; quit;
ods tagsets.MSOffice2K close;
```

The preceding code uses the MSOffice2K tagset to generate the HTML file and the Banker style to control the appearance of the HTML. Figure 3 shows the resulting HTML file that's opened in Word.


aedata.htm - Microsoft Word
File Edit View Insert Format Tools SAS Table Window Help
Type a question for help
Page Sec At Ln Col REC TRK EXT OV

Protocol Identifier=ABC 123

Patient Identifier	Visit Identifier	Date	Code	Preferred Term	Severity	Severity	Frequency
1	1	05AUG1993	01090001	HEADACHE	2	Moderate	1
1	2	06AUG1993	04550001	EDEMA	1	Mild	1
3	1	05MAY1993	02790010	GASTRIC DISCOMFORT	1	Mild	1
3	2	06M					
4	1	26.					
4	2	27.					
6	1	04A					
6	2	07A					
6	2	07A					
7	1	05.					
8	1	24J					
9	1	15A					
10	1	19.					
10	1	19.					
10	2	20.					
12	1	20M					
13	1	20M					
14	1	12A					
16	1	24J					
16	2	25J					
18	1	20					
18	1	20					
18	2	22					
18	2	22					

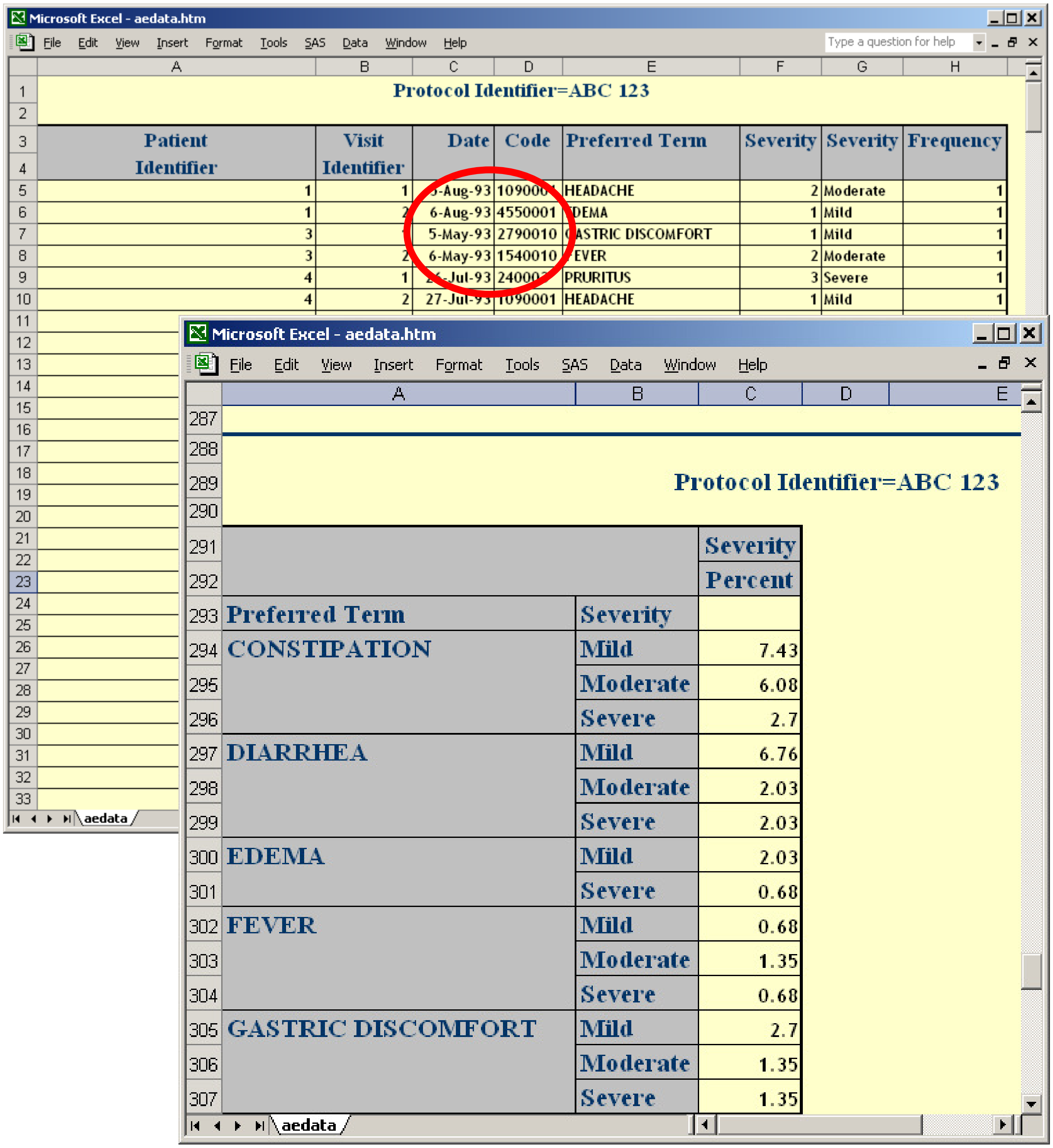
Protocol Identifier=ABC 123

		Severity
		Percent
Preferred Term	Severity	
CONSTIPATION	Mild	7.43
	Moderate	6.08
	Severe	2.70
DIARRHEA	Mild	6.76
	Moderate	2.03
	Severe	2.03
EDEMA	Mild	2.03
	Severe	0.68
FEVER	Mild	0.68
	Moderate	1.35
	Severe	0.68
GASTRIC DISCOMFORT	Mild	2.70
	Moderate	1.35
	Severe	1.35

Figure 3. ODS HTML Output Viewed Using Microsoft Word

Word does a good job of preserving the format and nature of the SAS output. The results shown in Figure 3 very closely match what you would see if you opened the HTML file with a Web browser. Note that all the SAS output is in a single document; Figure 3 shows two different views of the single document.

However, opening the same HTML output in Excel (Figure 4) reveals that there are some formatting and data issues. The background color for some of the cells is gray instead of green. The dates are formatted differently than they were in Word. The leading zeros in the adverse event code have been dropped. The cause and a means of correcting these problems are discussed in the section "Correcting Common Formatting Problems". Notice that all the SAS output is in a single worksheet. Figure 4 displays two different views of the single worksheet.


Microsoft Excel - aedata.htm
File Edit View Insert Format Tools SAS Data Window Help
Type a question for help
aedata

Protocol Identifier=ABC 123

Patient Identifier	Visit Identifier	Date	Code	Preferred Term	Severity	Severity	Frequency
1	1	-Aug-93	109000	HEADACHE	2	Moderate	1
1	2	6-Aug-93	4550001	DEMA	1	Mild	1
3	1	5-May-93	2790010	ASTRIC DISCOMFORT	1	Mild	1
3	2	6-May-93	1540010	EVER	2	Moderate	1
4	1	-Jul-93	24000	PRURITUS	3	Severe	1
4	2	27-Jul-93	1090001	HEADACHE	1	Mild	1

Protocol Identifier=ABC 123

Preferred Term	Severity	Severity Percent
CONSTIPATION	Mild	7.43
	Moderate	6.08
	Severe	2.7
DIARRHEA	Mild	6.76
	Moderate	2.03
	Severe	2.03
EDEMA	Mild	2.03
	Severe	0.68
FEVER	Mild	0.68
	Moderate	1.35
	Severe	0.68
GASTRIC DISCOMFORT	Mild	2.7
	Moderate	1.35
	Severe	1.35

Figure 4. ODS HTML Output Viewed Using Microsoft Excel

USING ODS TO GENERATE XML FOR EXCEL

Up to this point, we used an ODS tagset to create HTML that is imported into Excel, but Excel is also capable of reading XML files. It does an especially good job of handling files that conform to the Microsoft XML Spreadsheet Specification ("XML Spreadsheet Reference", Microsoft Corp.). The SAS®9 ExcelXP tagset generates this form of XML for use with Microsoft Excel. The main benefit of using the ExcelXP (XML) tagset is that each table of the SAS output is placed into a separate worksheet in the workbook, in contrast to a single worksheet as shown in Figure 4 when using the MSOffice2K (HTML) tagset.

However, there are some disadvantages to using the ExcelXP tagset that you should be aware of:

- The tagset is experimental. This means that limited testing has been performed on it, and its functionality might change in the future.
- The tagset has not yet been optimized. It can take an unexpectedly long time to generate an XML file. This problem was partially rectified in SAS® 9.1.3.
- You cannot use graphics in your SAS XML output because Excel does not support embedded graphics in an XML file.
- Because Excel does not automatically resize text-based columns when reading an XML file, when you first open the XML file, most of the columns are not wide enough to show all the data or headings. A future release of the ExcelXP tagset may correct this problem.

Significant enhancements have been made to the ExcelXP tagset since the initial release of SAS 9.1. You can download a recent version of the ExcelXP tagset from the SAS Presents Web site at http://support.sas.com/saspresents. Find the entry "From SAS to Excel via XML". The download contains a file named "excelxp.sas", which contains the SAS code for creating the ExcelXP tagset. Save a copy of this file, and submit the following SAS code to make the tagset available:

```
❶ libname myLib 'directory-for-tagset'; * location to store the tagset;

❷ ods path myLib.tmplmst(update) sashelp.tmplmst(read);

  %include 'excelxp.sas';
```

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

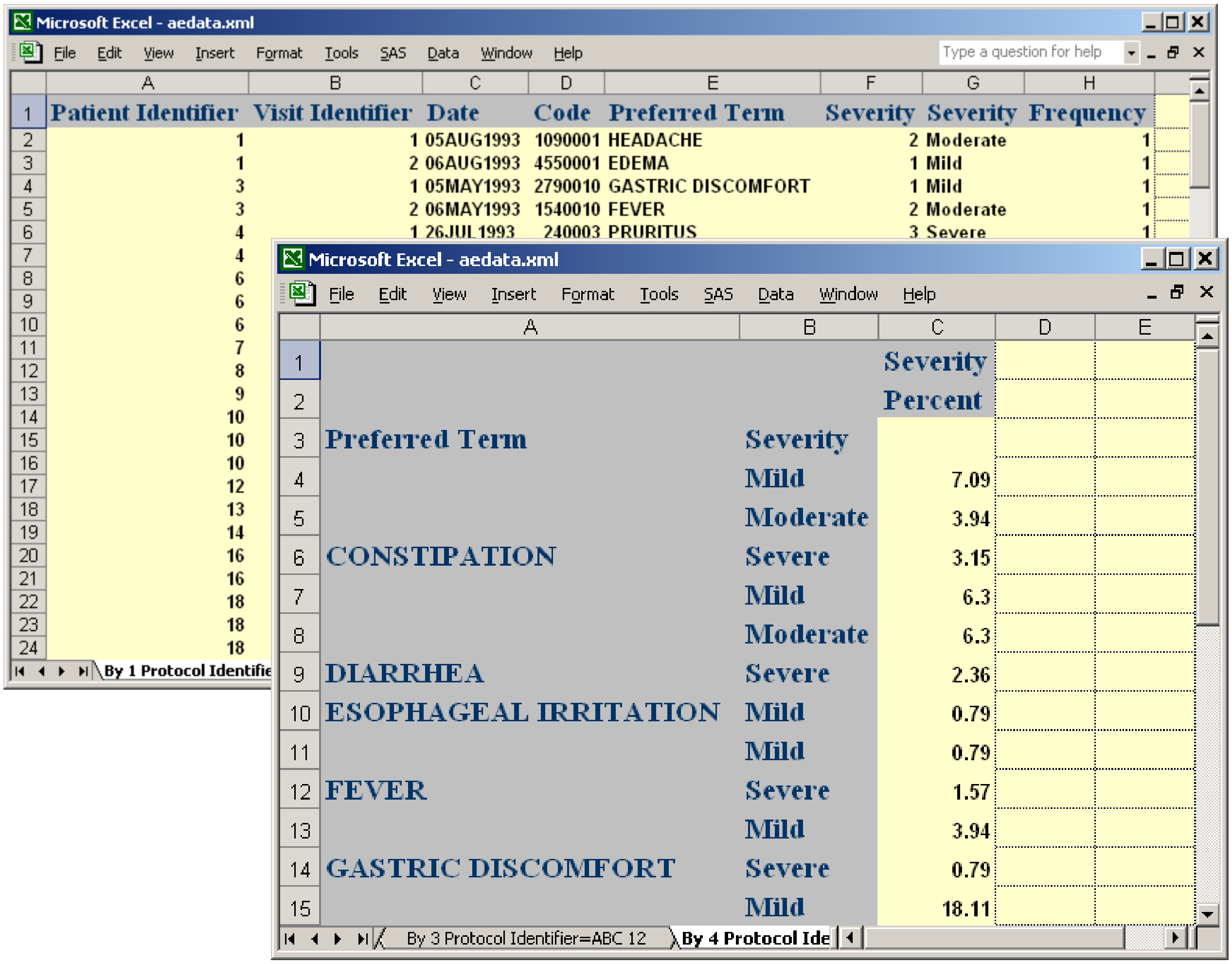
The ODS PATH statement (❷) specifies the locations and the order in which to search for ODS tagsets and styles. Notice that the access mode for `myLib.tmplmst` is specified as "update" and the access mode for `sashelp.tmplmst` is specified as "read". Because ODS searches the PATH in the order given and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

You can use the ExcelXP tagset by specifying the tagset name in an ODS statement and open the resulting XML file with Excel. To continue using the sample code shown earlier (page 3), simply change the ODS statement as shown here:

```
ods listing close;
ods tagsets.ExcelXP style=Banker file='aedata.xml';
  *  PROC PRINT and PROC TABULATE code here;
ods tagsets.ExcelXP close;
```

Figure 5 shows the results of opening the XML file with Excel. As was the case with the HTML file, some cell background colors appear gray instead of green, and the leading zeros in the adverse event code have been dropped. Additionally, border lines are missing from all cells. However, unlike HTML, all the tables created by the PRINT and TABULATE procedures appear in their own worksheet.

The next section discusses how to correct the formatting issues noted previously.


Microsoft Excel - aedata.xml
File Edit View Insert Format Tools SAS Data Window Help
Type a question for help
By 1 Protocol Identifie
Microsoft Excel - aedata.xml
File Edit View Insert Format Tools SAS Data Window Help
By 3 Protocol Identifier=ABC 12
By 4 Protocol Ide

	A	B	C	D	E	F	G	H
1	Patient Identifier	Visit Identifier	Date	Code	Preferred Term	Severity	Severity	Frequency
2	1	1	05AUG1993	1090001	HEADACHE	2	Moderate	1
3	1	2	06AUG1993	4550001	EDEMA	1	Mild	1
4	3	1	05MAY1993	2790010	GASTRIC DISCOMFORT	1	Mild	1
5	3	2	06MAY1993	1540010	FEVER	2	Moderate	1
6	4	1	26JUL1993	240003	PRURITUS	3	Severe	1
7	4							
8	6							
9	6							
10	6							
11	7							
12	8							
13	9							
14	10							
15	10							
16	10							
17	12							
18	13							
19	14							
20	16							
21	16							
22	18							
23	18							
24	18							

	A	B	C	D	E
1			Severity		
2			Percent		
3	Preferred Term	Severity			
4		Mild	7.09		
5		Moderate	3.94		
6	CONSTIPATION	Severe	3.15		
7		Mild	6.3		
8		Moderate	6.3		
9	DIARRHEA	Severe	2.36		
10	ESOPHAGEAL IRRITATION	Mild	0.79		
11		Mild	0.79		
12	FEVER	Severe	1.57		
13		Mild	3.94		
14	GASTRIC DISCOMFORT	Severe	0.79		
15		Mild	18.11		

Figure 5. ODS XML Output Viewed Using Microsoft Excel after Resizing Columns Manually

CORRECTING COMMON FORMATTING PROBLEMS

When Excel and Word import HTML or XML files, the files are converted to native Excel or Word format. Sometimes this conversion process is less than perfect, with Word generally handling the conversion better than Excel. Although you can use the features of Excel and Word to manually correct the appearance of the file, doing so will be cumbersome if you have a large number of files to correct. It's much more efficient to have ODS generate a file that's compatible with Microsoft Office in the first place. This is accomplished by creating custom ODS styles and by using ODS style overrides.

USING STYLE OVERRIDES TO CORRECT OUTPUT-SPECIFIC PROBLEMS

Although ODS styles define a particular appearance for your SAS output, you can change the attributes that are defined in the style at run time by using a "style override". Style overrides are supported by the PRINT, TABULATE, and REPORT procedures, and are best used for changing a small number of features that are associated with a particular output. In this example case, style overrides can be used to correct the missing leading zeros and improperly formatted date columns in the HTML and XML files that were imported into Excel.

When an HTML or XML file is opened, Excel automatically assigns an Excel format based on the data in the cells. Sometimes, the default format is not the best choice. Fortunately, we can use a style override to instruct ODS to assign a specific Excel format to a column. Because only the output from the PRINT procedure was affected, the VAR statement is the only line of code that needs to be changed.

To correct the date formatting and the missing leading zeros problems in the HTML file in Figure 4, change the single VAR statement, in the sample code given earlier (page 3), to the following:

```
var patient visit;
var aedate / style={htmlstyle="mso-number-format:ddmmmyyyy"};
var aecode / style={htmlstyle="mso-number-format:00000000"};
var aetext aesev aesevc frequency;
```

The VAR statements instruct ODS to assign the special Excel format "ddmmmyyyy" for the AEDATE column and the "00000000" format for the AECODE column. The first format is self-explanatory; the second format is comparable to the SAS Z8. format. The HTML file that's created by ODS can be viewed in Excel, Word, or a Web browser, because Web browsers ignore the proprietary Microsoft mso-number-format style information when they render the HTML. Complete code for correcting the HTML output can be found in the Appendix in the section "Corrected Code to Export SAS Output to Excel or Word as HTML".

You can use the same concept to correct the leading zeros problem in the XML file shown in Figure 5, but the syntax is slightly different:

```
var patient visit aedate;
var aecode / style={tagattr="\00000000"};
var aetext aesev aesevc frequency;
```

The next section describes how to correct the remainder of the formatting problems. The complete program for correcting the XML output can be found in the Appendix in the section "Corrected Code to Export SAS Output to Excel as XML". For more information about Excel formats, see "A Beginner's Guide to Incorporating SAS® Output in Microsoft Office Applications" in the *Proceedings of the Twenty-Eighth Annual SUGI Conference* (DelGobbo, 2003).

USING PROC TEMPLATE TO CORRECT GLOBAL STYLE PROBLEMS

Global problems are those that affect *all* output that's generated by ODS. For example, all header and row header cell background colors are wrong in *all* ODS output, as illustrated by Figures 4 and 5. Because these problems affect all output that's generated by ODS, it's more efficient to make a correction to the ODS style, rather than using an override in each of your SAS programs.

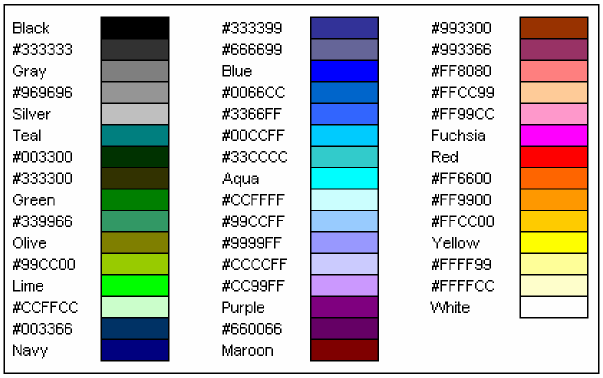
Excel mapped the green to gray because the shade of green that's specified by ODS is not supported by Excel by default. If you view the HTML source code and identify the cells that are affected, you'll notice that the cells are assigned the Cascading Style Sheet (CSS) class Header or RowHeader. That is, the HTML contains class="Header" or class="RowHeader" in the TH tags. If you further examine the HTML file, you'll see the following CSS definitions for the Header and RowHeader classes:

```
.Header
{
  font-family: 'Times New Roman', Times, serif;
  font-size: 14pt;
  font-weight: bold;
  font-style: normal;
  color: #033366;
  background-color: #C4E4B8;
}
```

```
.RowHeader
{
  font-family: 'Times New Roman', Times, serif;
  font-size: 14pt;
  font-weight: bold;
  font-style: normal;
  color: #033366;
  background-color: #C4E4B8;
}
```

Both of these classes use the HTML color `#C4E4B8`. However, in Figure 6, you'll see that this color is not supported by Excel by default; therefore, Excel maps this color to gray. Excel does support a shade of green, `#CCFFCC`, that's very close to the original, and we'll show you how to modify the Banker style to use this new shade of green by using the TEMPLATE procedure.

The TEMPLATE procedure is part of Base SAS. PROC TEMPLATE can be used to list, create, and modify ODS styles and tagsets. Because an in-depth discussion of this procedure is beyond the scope of this paper, see the chapters about the TEMPLATE Procedure in the ODS documentation (SAS Institute Inc., 1999, 2004).



Black
#333333
Gray
#969696
Silver
Teal
#003300
#333300
Green
#339966
Olive
#99CC00
Lime
#CCFFCC
#003366
Navy

#333399
#666699
Blue
#0066CC
#3366FF
#00CCFF
#33CCCC
Aqua
#CCFFFF
#99CCFF
#9999FF
#CCCCFF
#CC99FF
Purple
#660066
Maroon

#993300
#993366
#FF8080
#FFCC99
#FF99CC
Fuchsia
Red
#FF6600
#FF9900
#FFCC00
Yellow
#FFFF99
#FFFFCC
White

Figure 6. Default HTML Colors Supported by Excel 2000 and Later

Before you can make a change to a style, you must first determine the nature of the change that needs to be made. Submit the following SAS code to list the source of the Banker style:

```
ods listing;
proc template;
  source styles.Banker;
run; quit;
```

By examining the output, you will see that the following colors are used:

```
replace colors /
   'headerfgemph'   = cx033366
   'headerbgemph'   = cxFFFFCD
   'headerfgstrong' = cx033366
   'headerbgstrong' = cxFFFFCD
   'headerfg'       = cx033366
   'headerbg'       = cxC4E4B8
                . . .
```

Notice that SAS uses the `cx` notation to represent hexadecimal color values and HTML uses the symbol `#`. If you compare the preceding code to Figure 6, you'll see that the Banker style is using 3 colors that are not supported by Excel: `cx033366`, `cxFFFFCD`, and `cxC4E4B8`. However, Excel maps `cx033366` to `#003366`, which is so close that you will probably not notice the difference. Similarly, `cxFFFFCD` is mapped to `#FFFFCC`, which is also a very good match. Thus, the only change that's needed to the Banker style is to substitute `cxCCFFCC` for `cxC4E4B8`. This is accomplished with the following code (the complete code is in the Appendix in the section "Code for Creating the MSOBanker Style"):

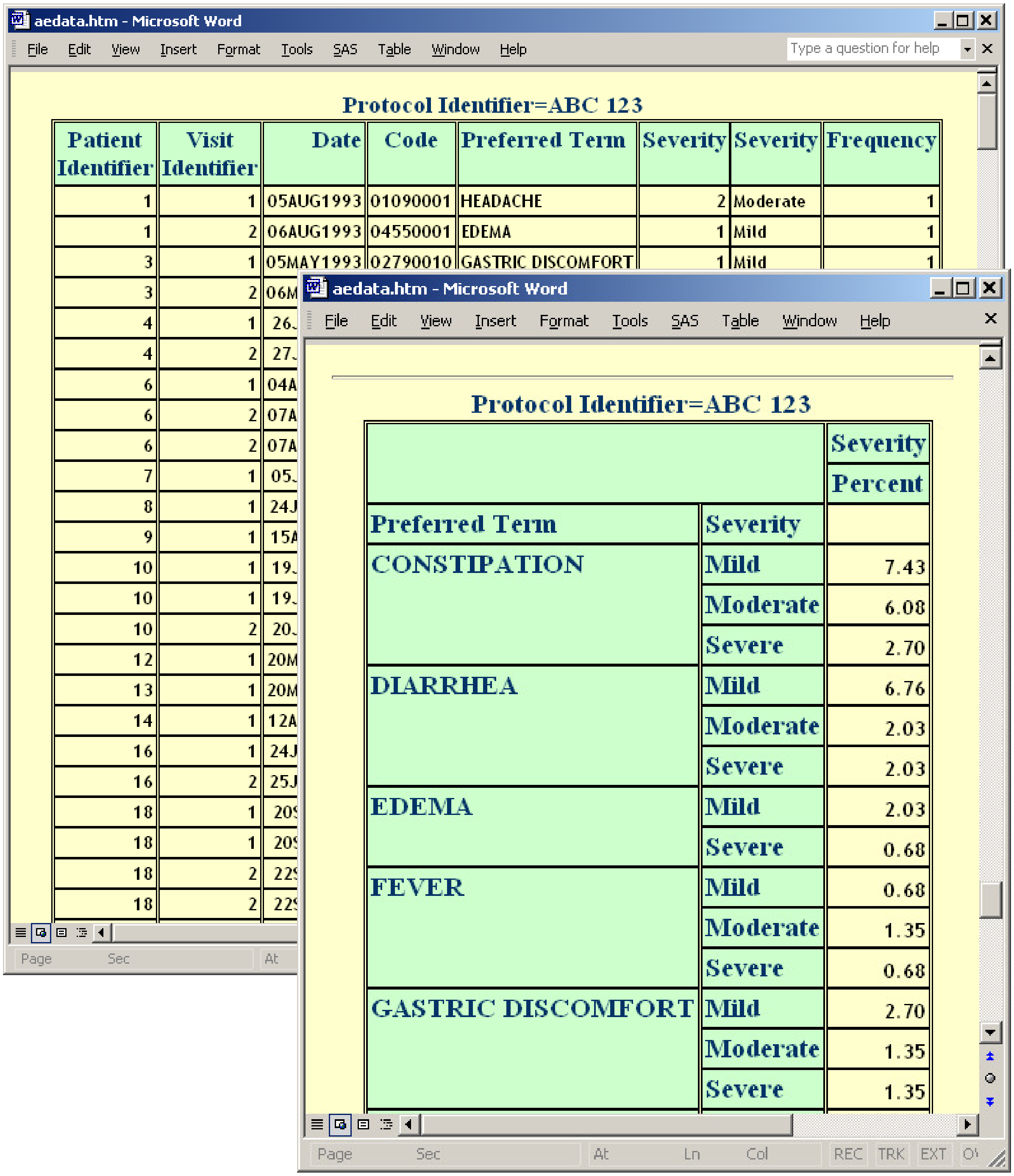
```
libname myLib 'directory-for-style'; * location to store the style;

ods path myLib.tmplmst(update) sashelp.tmplmst(read);

proc template;
  define style styles.MSOBanker;
    parent = styles.Banker;
    replace colors /
      'headerfgemph'   = cx033366
      'headerbgemph'   = cxFFFFCD
      'headerfgstrong' = cx033366
      'headerbgstrong' = cxFFFFCD
      'headerfg'       = cx033366
      'headerbg'       = cxCCFFCC
                   . . .
```

This new style, named MSOBanker, will be stored in `myLib.tmplmst`. It has all the attributes of the parent Banker style; the only difference is that the header and row header cell background colors will be `#CCFFCC` instead of `#C4E4B8`.

You can use the MSOBanker style to create a new HTML file by specifying the style name in an ODS statement and opening the resulting HTML file with Excel or Word. To continue using the original sample code (page 3) with the style overrides discussed in the preceding section, change the ODS statement as shown next:

```
ods listing close;
ods tagsets.MSOffice2K style=MSOBanker file='aedata.htm';
  *  PROC PRINT code w/ style overrides and PROC TABULATE code here;
ods tagsets.MSOffice2K close;
```

Figures 7 and 8 show the results of opening the HTML file with Word and Excel, respectively. All the formatting problems have been corrected by using a combination of style overrides and the new MSOBanker style. Complete code for correcting the HTML output can be found in the Appendix in the section "Corrected Code to Export SAS Output to Excel or Word as HTML".


aedata.htm - Microsoft Word
File Edit View Insert Format Tools SAS Table Window Help
Type a question for help
Page Sec At

Protocol Identifier=ABC 123

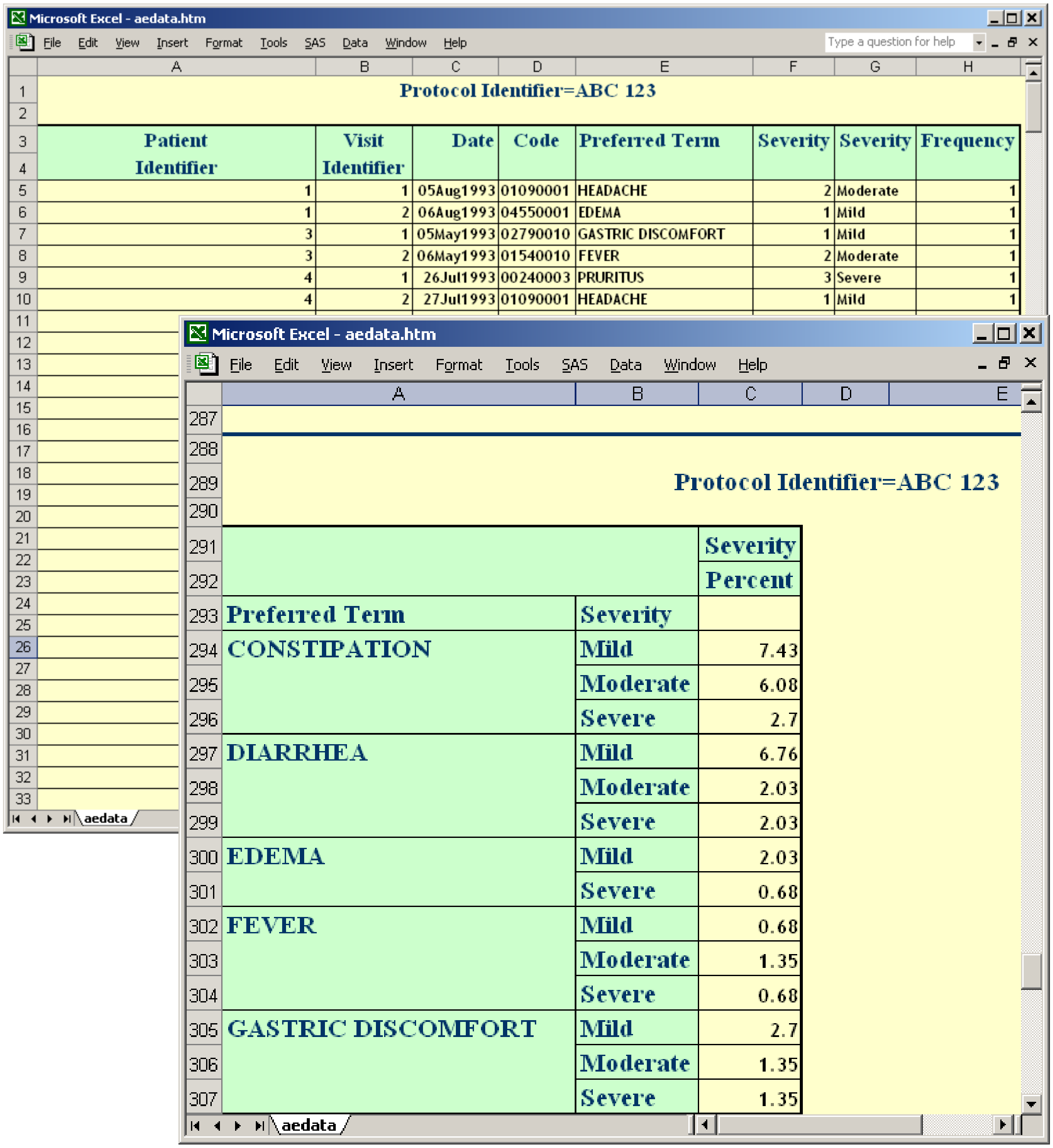
Patient Identifier	Visit Identifier	Date	Code	Preferred Term	Severity	Severity	Frequency
1	1	05AUG1993	01090001	HEADACHE	2	Moderate	1
1	2	06AUG1993	04550001	EDEMA	1	Mild	1
3	1	05MAY1993	02790010	GASTRIC DISCOMFORT	1	Mild	1
3	2	06M					
4	1	26J					
4	2	27J					
6	1	04A					
6	2	07A					
6	2	07A					
7	1	05J					
8	1	24J					
9	1	15A					
10	1	19J					
10	1	19J					
10	2	20J					
12	1	20M					
13	1	20M					
14	1	12A					
16	1	24J					
16	2	25J					
18	1	20S					
18	1	20S					
18	2	22S					
18	2	22S					

aedata.htm - Microsoft Word
File Edit View Insert Format Tools SAS Table Window Help
Page Sec At Ln Col REC TRK EXT OV

Protocol Identifier=ABC 123

		Severity Percent
Preferred Term	Severity	
CONSTIPATION	Mild	7.43
	Moderate	6.08
	Severe	2.70
DIARRHEA	Mild	6.76
	Moderate	2.03
	Severe	2.03
EDEMA	Mild	2.03
	Severe	0.68
FEVER	Mild	0.68
	Moderate	1.35
	Severe	0.68
GASTRIC DISCOMFORT	Mild	2.70
	Moderate	1.35
	Severe	1.35

Figure 7. Corrected ODS HTML Output Viewed Using Microsoft Word


Microsoft Excel - aedata.htm
File Edit View Insert Format Tools SAS Data Window Help
Type a question for help
aedata

Protocol Identifier=ABC 123

Patient Identifier	Visit Identifier	Date	Code	Preferred Term	Severity	Severity	Frequency
1	1	05Aug1993	01090001	HEADACHE	2	Moderate	1
1	2	06Aug1993	04550001	EDEMA	1	Mild	1
3	1	05May1993	02790010	GASTRIC DISCOMFORT	1	Mild	1
3	2	06May1993	01540010	FEVER	2	Moderate	1
4	1	26Jul1993	00240003	PRURITUS	3	Severe	1
4	2	27Jul1993	01090001	HEADACHE	1	Mild	1

Protocol Identifier=ABC 123

		Severity Percent
Preferred Term	Severity	
CONSTIPATION	Mild	7.43
	Moderate	6.08
	Severe	2.7
DIARRHEA	Mild	6.76
	Moderate	2.03
	Severe	2.03
EDEMA	Mild	2.03
	Severe	0.68
FEVER	Mild	0.68
	Moderate	1.35
	Severe	0.68
GASTRIC DISCOMFORT	Mild	2.7
	Moderate	1.35
	Severe	1.35

Figure 8. Corrected ODS HTML Output Viewed Using Microsoft Excel

We will now focus on making the final corrections to the XML file that's generated by ODS. In a previous section, we discussed using style overrides to handle the missing leading zeros problem when the XML file is opened by using Excel. There are still two outstanding issues with the format of the XML file: the green background is mapped to gray by Excel, and all cell border lines are missing. Both of these problems can be solved by creating a new ODS style to use with Excel.

If you examine the XML file that's generated by ODS and identify the cells that have missing border lines, you'll see StyleID="Header", StyleID="RowHeader", and StyleID="Data" in the Cell tags. When you further examine the XML file, you'll see the following definitions for the Header, RowHeader, and Data styles:

```
<Style ss:ID="Header">
<ss:Borders>
<ss:Border ss:Position="Left" />
<ss:Border ss:Position="Top" />
<ss:Border ss:Position="Right" />
<ss:Border ss:Position="Bottom" />
</ss:Borders>
<Interior ss:Color="#C4E4B8"
ss:Pattern="Solid" />
 . . .
</Style>
```

```
<Style ss:ID="RowHeader">
<ss:Borders>
<ss:Border ss:Position="Left" />
<ss:Border ss:Position="Top" />
<ss:Border ss:Position="Right" />
<ss:Border ss:Position="Bottom" />
</ss:Borders>
<Interior ss:Color="#C4E4B8"
ss:Pattern="Solid" />
 . . .
</Style>
```

```
<Style ss:ID="Data">
<ss:Borders>
<ss:Border ss:Position="Left" />
<ss:Border ss:Position="Top" />
<ss:Border ss:Position="Right" />
<ss:Border ss:Position="Bottom" />
</ss:Borders>
<Interior ss:Color="#FFFFCD"
ss:Pattern="Solid" />
 . . .
</Style>
```

The Header and RowHeader styles are using the unsupported green color for the background color of the cells. This can be corrected by using the MSOBanker style that you created earlier. The Border tags of each style have only a Position attribute and lack attributes to indicate the thickness or other properties of the border lines. You could modify the MSOBanker style to change the border width, but that would affect the already good-looking border lines in the HTML files. The best solution is to create a new style for use with the ExcelXP tagset by using the following code:

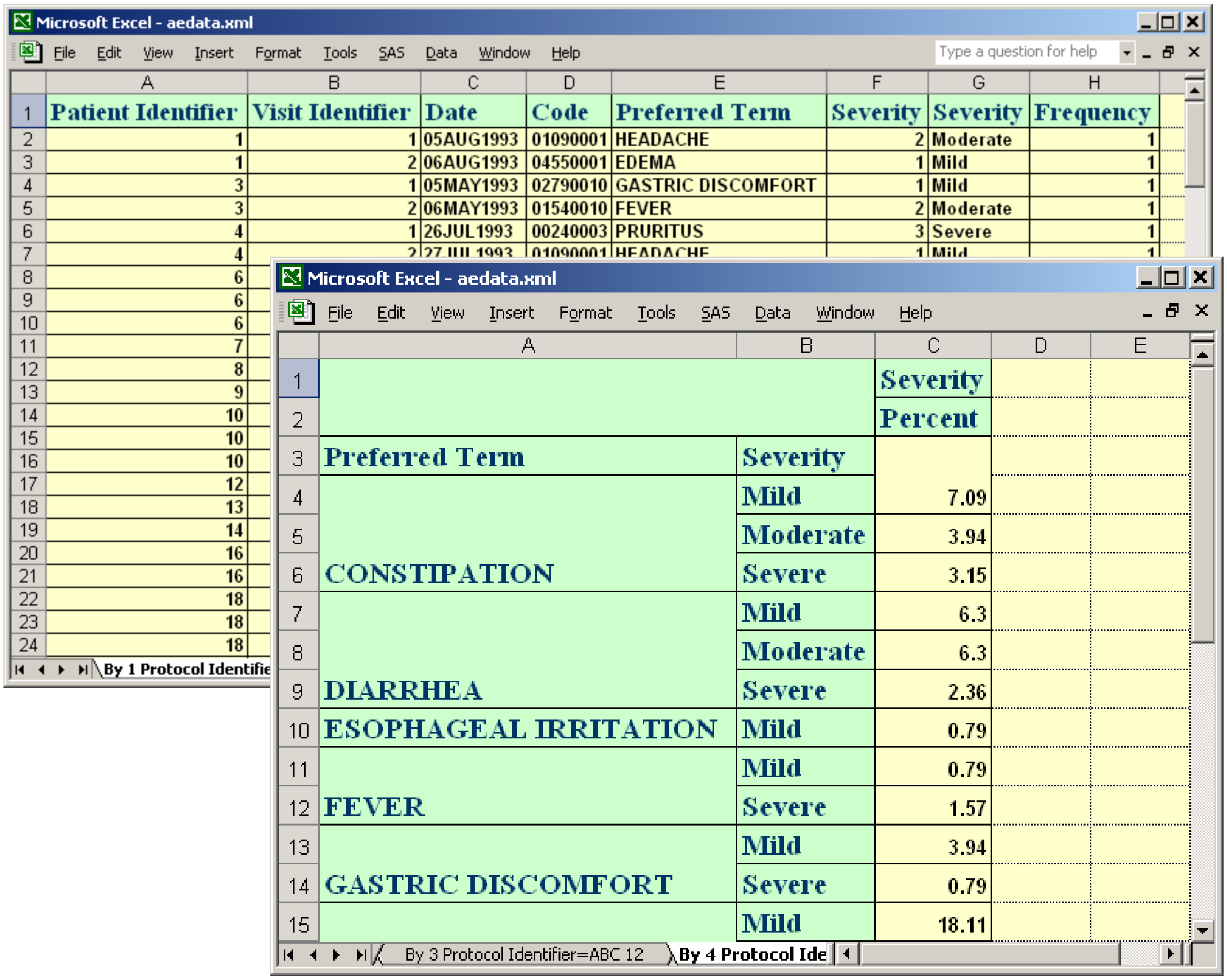
```
libname myLib 'directory-for-style'; * location to store the style;

ods path myLib.tmplmst(update) sashelp.tmplmst(read);

proc template;
  define style styles.XLBanker;
    parent = styles.MSOBanker;
    style Header from Header /
      borderwidth=2;
    style RowHeader from RowHeader /
      borderwidth=2;
    style Data from Data /
      borderwidth=2;
  end;
run; quit;
```

This new style, named XLBanker, will be stored in `myLib.tmplmst`. XLBanker has all the attributes of the parent MSOBanker style that was created earlier. The only difference is that border line widths were adjusted for use with Excel XML.

You can use the XLBanker style to create a new XML file by specifying the style name in an ODS statement and opening the resulting XML file with Excel. To continue using the original sample code (page 3) with the style overrides discussed in a previous section, change the ODS statement as shown next:

```
ods listing close;
ods tagsets.ExcelXP style=XLBanker file='aedata.xml';
  *  PROC PRINT code w/ style overrides and PROC TABULATE code here;
ods tagsets.ExcelXP close;
```

Figure 9 shows the results of opening the XML file by using Excel. All the formatting problems have been corrected by using a combination of style overrides and the new XLBanker style. Complete code for correcting the XML output can be found in the Appendix in the section "Corrected Code to Export SAS Output to Excel as XML".


Microsoft Excel - aedata.xml
File Edit View Insert Format Tools SAS Data Window Help
Type a question for help
By 1 Protocol Identifie
Microsoft Excel - aedata.xml
File Edit View Insert Format Tools SAS Data Window Help
By 3 Protocol Identifier=ABC 12
By 4 Protocol Ide

Patient Identifier	Visit Identifier	Date	Code	Preferred Term	Severity	Severity	Frequency
1	1	05AUG1993	01090001	HEADACHE	2	Moderate	1
1	2	06AUG1993	04550001	EDEMA	1	Mild	1
3	1	05MAY1993	02790010	GASTRIC DISCOMFORT	1	Mild	1
3	2	06MAY1993	01540010	FEVER	2	Moderate	1
4	1	26JUL1993	00240003	PRURITUS	3	Severe	1
4	2	27JUL1993	01090001	HEADACHE	1	Mild	1
6							
6							
6							
7							
8							
9							
10							
10							
10							
12							
13							
14							
16							
16							
18							
18							
18							

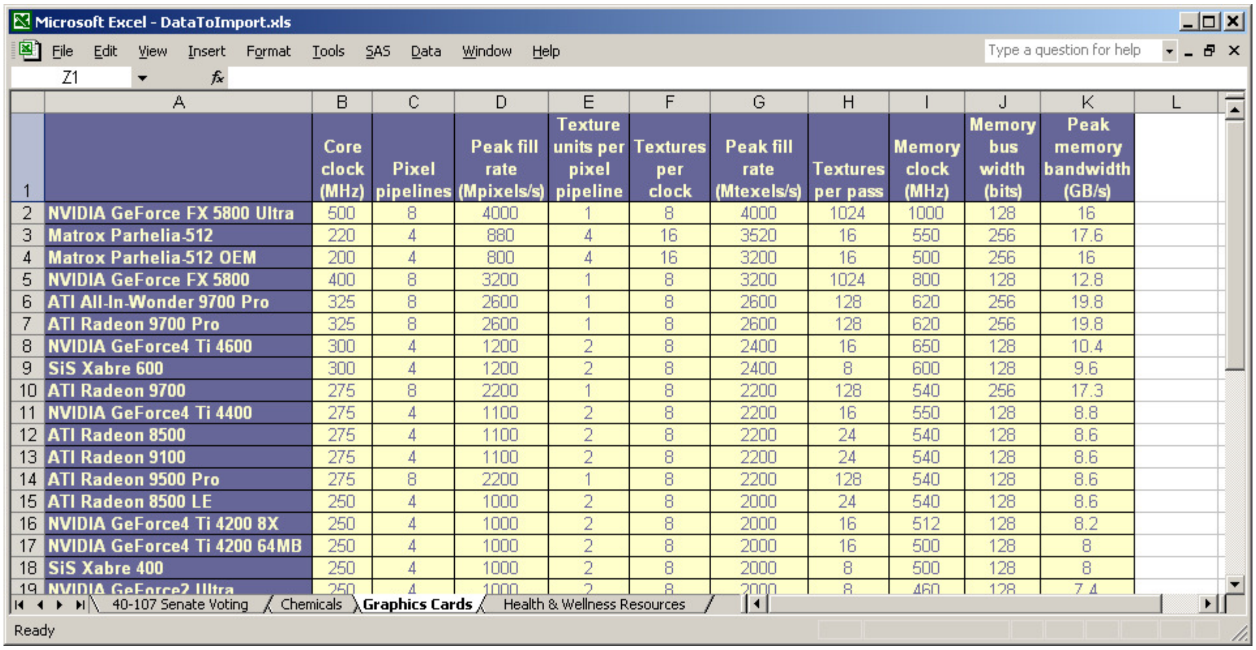
		Severity
		Percent
Preferred Term	Severity	
CONSTIPATION	Mild	7.09
	Moderate	3.94
	Severe	3.15
DIARRHEA	Mild	6.3
	Moderate	6.3
	Severe	2.36
ESOPHAGEAL IRRITATION	Mild	0.79
FEVER	Mild	0.79
	Severe	1.57
GASTRIC DISCOMFORT	Mild	3.94
	Severe	0.79
	Mild	18.11

Figure 9. Corrected ODS XML Output Viewed Using Microsoft Excel after Resizing Columns Manually

MOVING EXCEL DATA TO SAS

Figure 10 shows the Excel workbook that we want to import into SAS. The workbook contains four worksheets. We want to import the data from each worksheet into a different SAS table, and use the name of the worksheet for the table name. The sections that follow explain how to save the workbook as an XML file, and then how to load the workbook into SAS *using a provided SAS macro and SAS XMLMap, both designed specifically for Excel XML data*. **All the necessary SAS code is supplied for you.**

To download the sample data, SAS macro, and SAS XMLMap, go to the SAS Presents Web site at http://support.sas.com/saspresents. Find the entry "From SAS to Excel via XML". There you'll find a download package that has all the files you need.



	A	B	C	D	E	F	G	H	I	J	K
1		Core clock (MHz)	Pixel pipelines	Peak fill rate (Mpixels/s)	Texture units per pixel pipeline	Textures per clock	Peak fill rate (Mtexels/s)	Textures per pass	Memory clock (MHz)	Memory bus width (bits)	Peak memory bandwidth (GB/s)
2	NVIDIA GeForce FX 5800 Ultra	500	8	4000	1	8	4000	1024	1000	128	16
3	Matrox Parhelia-512	220	4	880	4	16	3520	16	550	256	17.6
4	Matrox Parhelia-512 OEM	200	4	800	4	16	3200	16	500	256	16
5	NVIDIA GeForce FX 5800	400	8	3200	1	8	3200	1024	800	128	12.8
6	ATI All-In-Wonder 9700 Pro	325	8	2600	1	8	2600	128	620	256	19.8
7	ATI Radeon 9700 Pro	325	8	2600	1	8	2600	128	620	256	19.8
8	NVIDIA GeForce4 Ti 4600	300	4	1200	2	8	2400	16	650	128	10.4
9	SiS Xabre 600	300	4	1200	2	8	2400	8	600	128	9.6
10	ATI Radeon 9700	275	8	2200	1	8	2200	128	540	256	17.3
11	NVIDIA GeForce4 Ti 4400	275	4	1100	2	8	2200	16	550	128	8.8
12	ATI Radeon 8500	275	4	1100	2	8	2200	24	540	128	8.6
13	ATI Radeon 9100	275	4	1100	2	8	2200	24	540	128	8.6
14	ATI Radeon 9500 Pro	275	8	2200	1	8	2200	128	540	128	8.6
15	ATI Radeon 8500 LE	250	4	1000	2	8	2000	24	540	128	8.6
16	NVIDIA GeForce4 Ti 4200 8X	250	4	1000	2	8	2000	16	512	128	8.2
17	NVIDIA GeForce4 Ti 4200 64MB	250	4	1000	2	8	2000	16	500	128	8
18	SiS Xabre 400	250	4	1000	2	8	2000	8	500	128	8
19	NVIDIA GeForce2 Ultra	250	4	1000	2	8	2000	8	460	128	7.4

Figure 10. Excel Workbook with Data to Import into SAS

SAVING AN EXCEL WORKBOOK AS XML

You must save an Excel workbook as an XML file before you can import it into SAS. To do this, in Excel, select File ➨ Save As, and choose "XML Spreadsheet (*.xml)". Specify a name and a directory for the file. For the purpose of this paper, we saved the XML in a file named `mydata.xml`.

To ensure that SAS can access the `mydata.xml` file, you can save the file on a network-accessible drive. Then you can access the file as if it were native to the operating environment where SAS is installed. Another solution is to save the file to a location that is under the control of a Web server. SAS will then be able to read the file using the URL access method, which is part of Base SAS. These techniques are useful if SAS and Excel are installed on different machines. If neither of these options is available to you, using FTP or some other method, move the file from the Windows machine to the machine that SAS is installed on.

If you place the files where others can access them, be sure to set file permissions to prevent accidental alteration.

AUTOMATIC DATA CONVERSION BETWEEN EXCEL AND SAS

You need to be aware of some of the behaviors to expect when converting Excel data to SAS data. As mentioned earlier, when the worksheets in Figure 10 are imported into SAS tables, the name of the worksheet (for example, "Chemicals") is used as the SAS table name. However, the technique of using the worksheet name for the table name doesn't always work. For example, a worksheet named "40-107 Senate Voting" can't be used for the SAS table name because it begins with a number and contains invalid characters (the spaces and a dash). Therefore, the provided SAS code that loads the worksheet into SAS must convert the "4" and other invalid characters to an underscore (_). In this case, the resulting SAS table name would be "_0_107_senate_voting". In the same way, importing a worksheet named "Health & Wellness Resources" would result in a table named "Health___wellness_resources".

Attempting to use Excel column labels for SAS table column names can result in a similar problem. Notice that none of the column labels in Figure 10 can be used as SAS column names because they all contain invalid characters such as blanks, parentheses, and the forward slash (/). Additionally, the first column does not have a label. The SAS code that loads the worksheet into SAS must create valid column names by converting invalid characters to an underscore (_). SAS column labels are set to the original value of the Excel label. Table 1 shows examples of how some of the Excel column labels in Figure 10 will be converted to SAS column names.

Excel Column Label	SAS Column Name	SAS Column Label
(blank)	A unique 32-character name starting with an underscore (_)	.
Pixel pipelines	Pixel_pipelines	Pixel pipelines
Peak fill rate (Mpixels/s)	Peak_fill_rate__Mpixels_s_	Peak fill rate (Mpixels/s)

Table 1. Conversion of Excel Column Labels to SAS Column Names

THE SAS XML LIBNAME ENGINE (SXLE) AND XMLMAPS

The SAS XML LIBNAME Engine can import an XML file into a SAS table or export a SAS table as an XML file. Thus, you can use the SXLE to exchange data between SAS and third-party, XML-aware applications such as Excel.

Although the SXLE has been available since SAS release 8.1, recent improvements have made it possible to precisely control how data is imported. The new SAS XMLMap enables you to map any XML element or attribute to a column or a row in a SAS table. The SXLE then uses the XMLMap to control how the XML data is imported into a SAS table. You can create the XMLMap by using the new XML Mapper (formerly know as XML Atlas), which provides a point-and-click interface.

READING EXCEL XML INTO SAS

You can use the SAS LIBNAME Engine and a SAS XMLMap that you created to import an Excel XML file into SAS. However, because the Excel XML format and the data conversion issues are quite complex, SAS provides an XMLMap and corresponding SAS code specific to Excel that loads the Excel XML into SAS tables. As mentioned earlier, both of these components and a SAS macro to import the XML data are available for download from the SAS Presents Web site at http://support.sas.com/saspresents.

Download the XMLMap and the SAS macro and make the two files available on the platform where SAS is installed. This paper assumes that you saved the SAS XMLMap in a file named `excelxp.map` and the SAS macro for loading the XML data was stored in a file named `loadxl.sas`. The file `loadxl.sas` contains a SAS macro named XLXP2SAS, which is used to import the Excel XML file into SAS tables.

The easiest way to explain how to use the macro is with a few examples. First, let's look at importing the XML workbook shown in Figure 10 when either SAS is installed on the same machine that has the XML file or SAS is installed on a different machine or platform, but the XML file is accessible via a network drive. To import the workbook into SAS, submit the following code, making sure to include the appropriate directory paths:

```
❶ %include 'loadxl.sas';

❷ %xlxp2sas(excelfile=mydata.xml,
            mapfile=excelxp.map);
```

The first statement (❶) makes the XLXP2SAS macro available to SAS. The second statement (❷) imports the data from all the worksheets into separate SAS tables. By default, the SAS tables are created in the WORK library. You can control which library is used to store the SAS tables by specifying the LIBRARY argument in the XLXP2SAS macro. For example, to store the tables in the SASUSER library, submit this code:

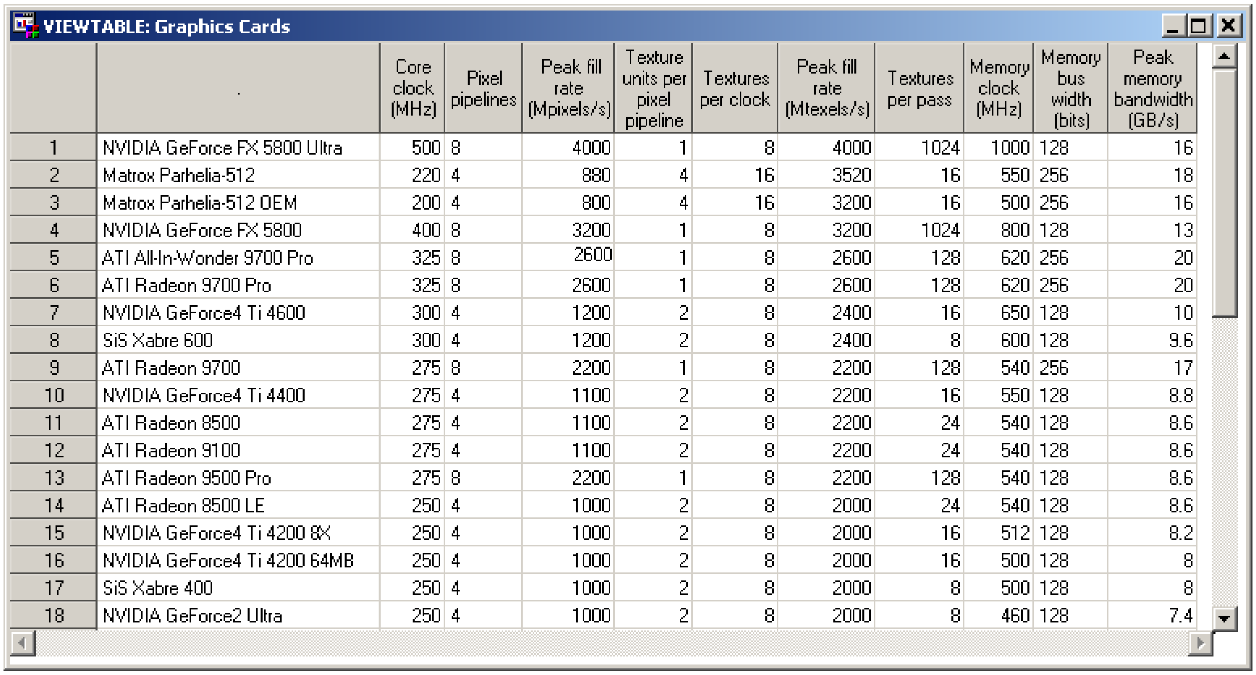
```
%xlxp2sas(excelfile=mydata.xml,
          mapfile=excelxp.map,
          library=sasuser);
```

Table 2 lists the SAS tables that were created as a result of importing the Excel workbook shown in Figure 10. Although the SAS table names might look a bit odd, the actual worksheet names are used in the SAS labels.

SAS Table Name	SAS Label
Chemicals	Chemicals
Graphics_cards	Graphics Cards
Health___wellness_resources	Health & Wellness Resources
_0_107_senate_voting	40-107 Senate Voting

Table 2. SAS Tables Created from the Excel Workbook

Figure 11 shows a portion of the Graphics Cards table. By comparing Figures 10 and 11, you can see that the XLXP2SAS macro successfully imported the "Graphics Cards" worksheet as a SAS table. For instance, the columns "Pixel pipelines" and "Memory bus width (bits)" were both correctly typed as character, because those columns contain data such as "2*2" and "128*2", respectively.



		Core clock (MHz)	Pixel pipelines	Peak fill rate (Mpixels/s)	Texture units per pixel pipeline	Textures per clock	Peak fill rate (Mtexels/s)	Textures per pass	Memory clock (MHz)	Memory bus width (bits)	Peak memory bandwidth (GB/s)
1	NVIDIA GeForce FX 5800 Ultra	500	8	4000	1	8	4000	1024	1000	128	16
2	Matrox Parhelia-512	220	4	880	4	16	3520	16	550	256	18
3	Matrox Parhelia-512 OEM	200	4	800	4	16	3200	16	500	256	16
4	NVIDIA GeForce FX 5800	400	8	3200	1	8	3200	1024	800	128	13
5	ATI All-In-Wonder 9700 Pro	325	8	2600	1	8	2600	128	620	256	20
6	ATI Radeon 9700 Pro	325	8	2600	1	8	2600	128	620	256	20
7	NVIDIA GeForce4 Ti 4600	300	4	1200	2	8	2400	16	650	128	10
8	SiS Xabre 600	300	4	1200	2	8	2400	8	600	128	9.6
9	ATI Radeon 9700	275	8	2200	1	8	2200	128	540	256	17
10	NVIDIA GeForce4 Ti 4400	275	4	1100	2	8	2200	16	550	128	8.8
11	ATI Radeon 8500	275	4	1100	2	8	2200	24	540	128	8.6
12	ATI Radeon 9100	275	4	1100	2	8	2200	24	540	128	8.6
13	ATI Radeon 9500 Pro	275	8	2200	1	8	2200	128	540	128	8.6
14	ATI Radeon 8500 LE	250	4	1000	2	8	2000	24	540	128	8.6
15	NVIDIA GeForce4 Ti 4200 8X	250	4	1000	2	8	2000	16	512	128	8.2
16	NVIDIA GeForce4 Ti 4200 64MB	250	4	1000	2	8	2000	16	500	128	8
17	SiS Xabre 400	250	4	1000	2	8	2000	8	500	128	8
18	NVIDIA GeForce2 Ultra	250	4	1000	2	8	2000	8	460	128	7.4

Figure 11. SAS Graphics Cards Table Created by Importing the Excel Workbook

Up to this point, the assumption is that the XML file resides on the same machine as SAS or was available via a network drive. However, if the XML file resides on the Web server of a remote machine, you can use the URL access method to retrieve the file by submitting the following code:

```
filename myxml URL 'http://Web-server/mydata.xml';
%xlxp2sas(excelfile=FILEREF:myxml,
          mapfile=excelxp.map);
```

The submitted code causes XLXP2SAS to contact the Web server to retrieve the XML file, rather than looking for it on a local disk. The FILEREF modifier can also be used with the MAPFILE argument to retrieve the SAS XMLMap from a Web server. Documentation for the XLXP2SAS macro can be found in the Appendix in the section “XLXP2SAS Macro”.

XLXP2SAS DRAWBACKS AND LIMITATIONS

Here are a few issues surrounding XLXP2SAS that you should be aware of.

- XLXP2SAS creates temporary tables in the WORK library. These tables can get very large. However, this is generally not a problem, unless your system is very low in disk space, because the temporary files are automatically cleaned up after XLXP2SAS runs (unless the argument CLEANUP=N was specified).
- The data in every worksheet that you want to import must be fairly rectangular. Although the XLXP2SAS macro attempts to handle non-rectangular data by adding missing values, the results can be unpredictable if the data is too sparse.
- By default, XLXP2SAS runs with the argument HASLABELS=Y. This implies that *all* worksheets in a workbook have column labels in the first row. If *none* of your worksheets contain column labels in the first row, specify HASLABELS=N when you invoke XLXP2SAS. The HASLABELS argument applies to all worksheets in a workbook.
- If you specify HASLABELS=N, the column names in the SAS table(s) will be in the form "COLUMN1", "COLUMN2", "COLUMN3", and so on, and the respective column labels will be "Column 1", "Column 2" and "Column 3".

SAS SERVER TECHNOLOGY

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

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

Detailed information about using these products to facilitate data interchange between SAS and Microsoft Office is presented in the previously mentioned SUGI Conference Papers (DelGobbo, 2002, 2003, 2004).

CONCLUSION

Using ODS to generate HTML and XML output is an effective method of incorporating SAS output in Excel and Word documents. Although, initially, you might encounter formatting problems when using this technique, by using ODS style overrides and PROC TEMPLATE, you can correct these problems.

The SAS 9.1 ExcelXP tagset complies with the Microsoft XML Spreadsheet Specification and provides an easy way to export your data to Excel workbooks that contain multiple worksheets. While it might be a bit cumbersome to use the SXLE and XMLMaps to import Excel data to SAS, the use of the XLXP2SAS macro greatly simplifies the task.

APPENDIX

CODE FOR CREATING THE MSOBANKER STYLE

```
libname myLib 'directory-for-style'; * location to store the style;

ods path myLib.tmplmst(update) sashelp.tmplmst(read);

proc template;
  define style styles.MSOBanker;
    parent = styles.Banker;
    replace colors /
      'headerfgemph'   = cx033366
      'headerbgemph'   = cxFFFFCD
      'headerfgstrong' = cx033366
      'headerbgstrong' = cxFFFFCD
      'headerfg'       = cx033366
      'headerbg'       = cxCCFFCC
      'datafgemph'     = cx000000
      'databgemph'     = cxFFFFCD
      'datafgstrong'   = cx000000
      'databgstrong'   = cxFFFFCD
      'datafg'         = cx000000
      'databg'         = cxFFFFCD
      'batchfg'        = cx000000
      'batchbg'        = cxFFFFCD
      'tableborder'    = cx000000
      'tablebg'        = cx000000
      'notefg'         = cx033366
      'notebg'         = cxFFFFCD
      'bylinefg'       = cx033366
      'bylinebg'       = cxFFFFCD
      'captionfg'      = cx033366
      'captionbg'      = cxFFFFCD
      'proctitlefg'    = cx033366
      'proctitlebg'    = cxFFFFCD
      'titlefg'        = cx033366
```

```
         'titlebg'         = cxFFFFCD
         'systitlefg'      = cx033366
         'systitlebg'      = cxFFFFCD
         'Conentryfg'      = cx033366
         'Confolderfg'     = cx033366
         'Contitlefg'      = cx033366
         'link2'           = cx800080
         'link1'           = cx0000FF
         'contentfg'       = cx000000
         'contentbg'       = cxFFFFCD
         'docfg'           = cx033366
         'docbg'           = cxFFFFCD;
     end;
   run; quit;
```

Notice that `cx000000` is black, `cx800080` is purple and `cx0000FF` is blue. According to Figure 6, these colors are supported by Excel by default, so no remapping is needed for these colors.

CORRECTED CODE TO EXPORT SAS OUTPUT TO EXCEL OR WORD AS HTML

```
ods listing close;
ods tagsets.MSOffice2K style=MSOBanker file='aedata.htm';
  title; footnote;
  proc print data=pharma.phcae noobs label;
    by protocol;
    var patient visit;
    var aedate / style={htmlstyle="mso-number-format:ddmmmyyyy"};
    var aecode / style={htmlstyle="mso-number-format:00000000"};
    var aetext aesev aesevc frequency;
  run; quit;

  proc tabulate data=pharma.phcae;
    by protocol;
    var aesev;
    class aetext aesevc;
    table aetext*aesevc,aesev*pctn;
    keyword all pctn;
    keylabel pctn='Percent';
  run; quit;
ods tagsets.MSOffice2K close;
```

CORRECTED CODE TO EXPORT SAS OUTPUT TO EXCEL AS XML

```
ods listing close;
ods tagsets.ExcelXP style=XLBanker file='aedata.xml';
  title; footnote;
  proc print data=pharma.phcae noobs label;
    by protocol;
    var patient visit aedate;
    var aecode / style={tagattr="\00000000"};
    var aetext aesev aesevc frequency;
  run; quit;

  proc tabulate data=pharma.phcae;
    by protocol;
    var aesev;
    class aetext aesevc;
    table aetext*aesevc,aesev*pctn;
    keyword all pctn;
    keylabel pctn='Percent';
  run; quit;
ods tagsets.ExcelXP close;
```

XLXP2SAS MACRO

The following table contains a list of the arguments that are supported by the XLXP2SAS macro. All arguments require a value, except where a default value is indicated.

Argument	Description	Default Value
EXCELFILE	Specifies the name and path for the Excel XML file that you want to import to SAS. Do not use quotation marks in this value. To specify a SAS FILEREF instead of a file, use FILEREF:*fref*, where *fref* is the FILEREF.	
MAPFILE	Specifies the name and path of the SAS XMLMap for reading Excel XML files. Do not use quotation marks in this value. To specify a SAS FILEREF instead of a file, use FILEREF:*fref*, where *fref* is the FILEREF. You can download a copy of the XMLMap that's provided by SAS under the entry "From SAS to Excel via XML" on the SAS Presents Web site at http://support.sas.com/saspresents.	
LIBRARY	Specifies the name of the SAS library where imported tables are stored.	WORK
HASLABELS	Specifies whether the worksheets have column labels in the first row of the Excel table. This setting applies to all worksheets in a workbook. If set to Y, the labels are used for the SAS column names and labels. If your workbook does not have column labels in the first row of the Excel table, specify N.	Y
CLEANUP	Controls whether temporary SAS files are deleted and whether to de-assign FILEREFs that were used when importing the Excel data to SAS. FILEREFs that you explicitly assign with a FILEREF statement will not be de-assigned. To disable this feature, specify N.	Y
VERBOSE	Controls the level of debugging information written to the SAS Log. Specify Y to activate this feature.	N

REFERENCES

DelGobbo, V. (2002), "Techniques for SAS® Enabling Microsoft® Office in a Cross-Platform Environment," Proceedings of the Twenty-Seventh Annual SAS Users Group International Conference, 27, CD-ROM. Paper 174. Available http://www2.sas.com/proceedings/sugi27/p174-27.pdf

DelGobbo, V. (2003), "A Beginner's Guide to Incorporating SAS® Output in Microsoft® Office Applications", Proceedings of the Twenty-Eighth Annual SAS Users Group International Conference, 28, CD-ROM. Paper 52. Available http://www2.sas.com/proceedings/sugi28/052-28.pdf

SAS Institute Inc. (1999), "Chapter 5: The TEMPLATE Procedure", *The Complete Guide to the SAS Output Delivery System, Version 8*, Cary, NC: SAS Institute Inc.

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

"XML Spreadsheet Reference", Microsoft Corporation. Available http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnexcl2k2/html/odc_xmlss.asp

FURTHER READING

DelGobbo, V. (2004), "From SAS to Excel via XML". Available http://support.sas.com/saspresents

"Processing XML Documents with the Version 9 SAS XML LIBNAME Engine (SXLE)", SAS Institute Inc. Available http://support.sas.com/rnd/base/topics/sxle90/#doc

"XML Atlas", SAS Institute Inc. Available http://support.sas.com/rnd/base/topics/sxle90/#atlas

"XMLMap Syntax Version 1.1", SAS Institute Inc. Available http://support.sas.com/rnd/base/topics/sxle90/#xmlmap

ACKNOWLEDGMENTS

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

CONTACT INFORMATION

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

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

sasvcd@unx.SAS.com

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