From SAS® to Excel via XML

Vincent DelGobbo, SAS Institute Inc., Cary, NC

ABSTRACT

Transferring data between SAS and Microsoft® Excel can be difficult, especially when SAS is not installed on a Windows® platform. This paper discusses using new XML support in BASE SAS 9.1 software to move data between SAS and Microsoft Excel (versions 2002 and later). You can use the techniques described here regardless of the platform on which SAS software is installed, such as Windows, OpenVMS™, UNIX® or z/OS®. The use of SAS server technology is also discussed.

INTRODUCTION

The techniques described in this paper are for people who either have not licensed SAS 9.1 SAS/ACCESS® to PC Files, or those who want to display parts of SAS output in separate Excel worksheets.

Techniques currently exist for those in other situations:

- Using SAS 9.1 SAS/ACCESS® to PC Files, you can import Excel data into SAS and write to Excel workbooks from both Windows and UNIX environments (Plemmons, 2003). In previous releases of SAS, access to Excel was only possible with Windows versions of SAS software.

- If you want SAS output in a single Excel worksheet, you can use the Output Delivery System (ODS) to generate an HTML file which you can then be opened with Excel (DelGobbo, 2003).

This paper and all source code are available on the SAS Presents Web site (http://support.sas.com/saspresents/).

WHAT IS XML?

XML is an acronym for Extensible Markup Language, and represents a way to define and format data for easy exchange. XML is similar to HTML in that it uses *tags*. Unlike HTML, whose tags control how data is rendered, XML tags describe the structure and meaning of data but do not control how it is rendered.

For example, consider the XML file shown in Figure 1. The file contains the make, model name, and model year for several vehicles. Note the lack of HTML tags such as <TABLE>, <TR>, and <TD>. Instead, well-structured XML tags are used to describe the data. To find out more about XML, refer to the World Wide Web Consortium® Web site (http://www.w3.org/XML).

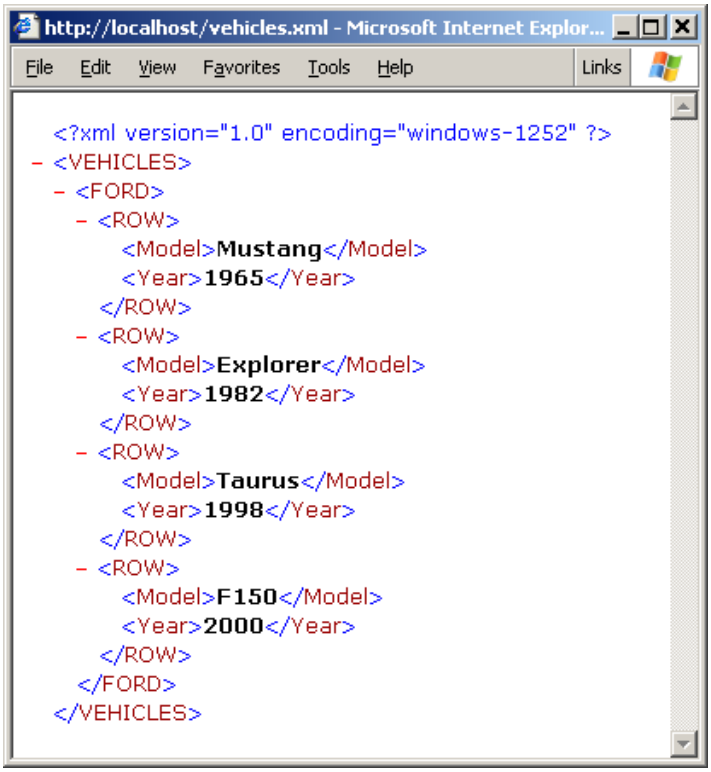


Figure 1. Sample XML file.

XML SUPPORT IN SAS 9.1

Support for XML in BASE SAS 9.1 is better than in any prior release. The two SAS XML tools of interest here are the SAS XML LIBNAME engine and the ExcelXP ODS tagset.

THE SAS XML LIBNAME ENGINE (SXLE)

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.

While 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 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 manually create the XMLMap using a text editor (this is *not* recommended), or you can use the new XML Mapper (formerly know as XML Atlas) which provides a point-and-click interface.

USING THE XML MAPPER

To briefly illustrate the use of the XML Mapper, consider the XML shown in Figure 1. From this XML we want to create a SAS table named Ford that has two columns, one named Model and the other Year. Figure 2 shows the XML Mapper after the mapping has been performed.

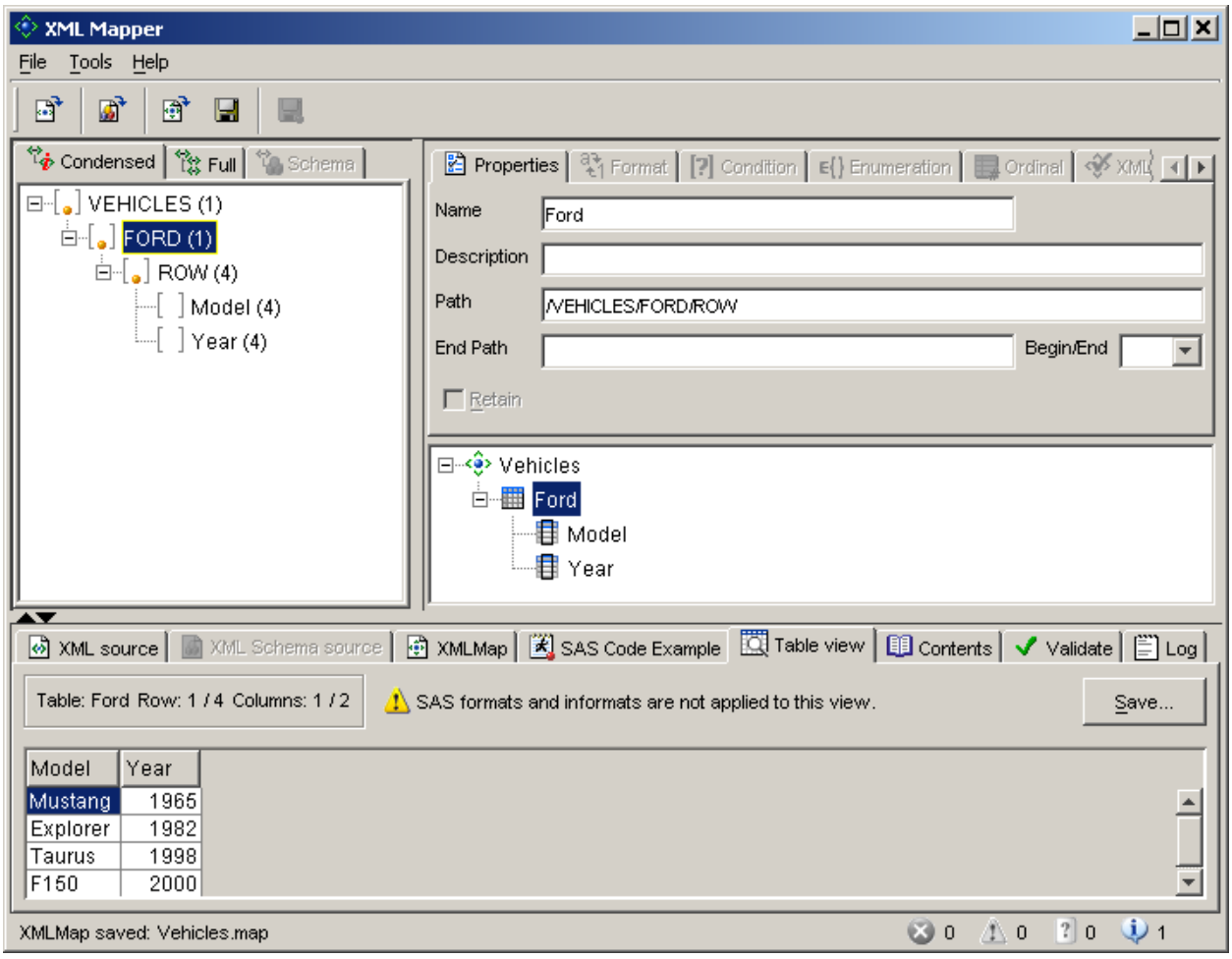


Figure 2. Building a simple SAS XMLMap with the XML Mapper.

In the upper left, the Condensed tab displays the structure of the XML data that you want to import. On the right, the XMLMap pane contains fields and a tree view of the XMLMap. The bottom pane shows the rows and columns of the SAS table that are a result of the mapping.

To create and use the XMLMap shown in Figure 2, follow these steps:

1. Open the Vehicles.xml file.
2. Click on VEHICLES in the Condensed tab and drag it onto the Name **field** of the XMLMap pane. VEHICLES will appear in the Name field. Change the name to Vehicles. A node named Vehicles appears in the XMLMap tree.
3. In the Condensed tab, click on ROW and drag it onto the **node** named Vehicles in the XMLMap tree. You will now have a node named ROW under the Vehicles node. Click on the ROW node and change the name to Ford.
4. In the Condensed tab, click on Model and drag it onto the Ford node in the XMLMap tree.
5. In the Condensed tab, click on Year and drag it onto the Ford node in the XMLMap tree.
6. Save the XMLMap file by selecting File ➨ Save XMLMap As. Name the XMLMap file Vehicles. This creates a file named Vehicles.map.
7. Access the Vehicles.xml file as a SAS table by submitting the following SAS statements, supplying any necessary path information on the FILENAME statements:

```
filename MYMAP 'Vehicles.map'; * created and saved by XML Mapper;
filename MYXML 'Vehicles.xml'; * raw XML file to import;
libname  MYXML xml xmlmap=MYMAP access=readonly;
```

   You can now access the Vehicles.xml file as a SAS table, via the MYXML LIBREF:

```
proc print data=myxml.Ford; run; quit;
```

Details about how to import Excel workbooks into SAS tables are covered in the “Moving Excel Data to SAS” section of this paper.

THE EXCELXP ODS TAGSET

An ODS tagset is a template that controls the type of tags (markup) applied to SAS output. For example, an HTML-based tagset would embed HTML tags in SAS output. There are over 50 tagsets shipped with BASE SAS 9. To see a full listing of the tagsets available on your system, use the TEMPLATE procedure as follows:

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

Once you determine which tagset you want to use, specify the name of the tagset in the ODS statement, for example:

```
ods tagsets.TagsetName file=... ;
  *  your SAS code here;
ods tagsets.TagsetName close;
```

In SAS 9, the MSOffice2K tagset generates HTML that can be imported by Microsoft Excel and Microsoft Word, versions 2000 and later (DelGobbo, 2003).

In contrast, the SAS 9 ExcelXP tagset generates XML, and can be used with Microsoft Excel 2002 (also known as Excel XP). To use the ExcelXP tagset, simply use the ExcelXP tagset name in an ODS statement, as illustrated above, and open the resulting XML file with Excel. The main benefit of using the ExcelXP tagset and not the MSOffice2K tagset is that each table of the SAS output is placed into a separate worksheet within the workbook.

However, there are some current 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 may change in the future.
- The tagset has not yet been optimized. It may take an unexpectedly long time to generate an XML file. This problem has been fixed 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.

Significant enhancements have been made to the ExcelXP tagset since the release of SAS 9.1. You can download a recent version of the ExcelXP tagset as well as the source code for this paper from the SAS Presents Web site

(http://support.sas.com/saspresents). The download contains a file named "excelxp.sas", which contains the SAS code needed to create the ExcelXP tagset. Save a copy of this file and submit this 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 at ❶ specifies where to store the tagset. Although you can use the WORK library to temporarily store the tagset, a more efficient process would be to create the tagset one time and store it in a permanent library so that you can reference it in other SAS programs.

The ODS PATH statement at ❷ specifies the locations and the order in which ODS searches when looking for tagsets and styles. Note that the access mode for the location "myLib.tmplmst" is specified as "update", while "sashelp.tmplmst" is specified as "read". Because ODS searches the PATH in the order given, and because myLib.tmplmst has an "update" access mode, PROC TEMPLATE will store the tagset in a file named "tmplmst.sas7bitm" in the directory associated with the MYLIB library.

Details about how to use the ExcelXP tagset are covered in the "Moving SAS Data to Excel" section of this paper.

XML SUPPORT IN EXCEL 2002

Excel 2002 supports importing XML documents as well as saving workbooks as XML files. To save a workbook in XML format, select File ➨ Save As, and then choose the "XML Spreadsheet" format. Your entire workbook is saved as a single XML file, which can then be imported into one or more SAS tables (each worksheet is imported into a separate SAS table).

XML files that conform to the Microsoft XML Spreadsheet Specification can be opened by Excel. The ODS ExcelXP tagset creates such XML. Thus, to import your SAS output that was generated using the ODS ExcelXP tagset into Excel, just open the XML file with Excel.

As mentioned earlier, Excel does not support graphics in XML files.

MOVING SAS DATA TO EXCEL

This section guides you through the process of moving SAS data to Excel: generating the XML output, opening the output in Excel, and then correcting the format of the data.

Figure 3 shows the column properties of the data that we are going to move to Excel. The data is adverse event data for a fictitious drug.

Note that labels have been applied to each column in the table. This allows for a more attractive table when imported into Excel.

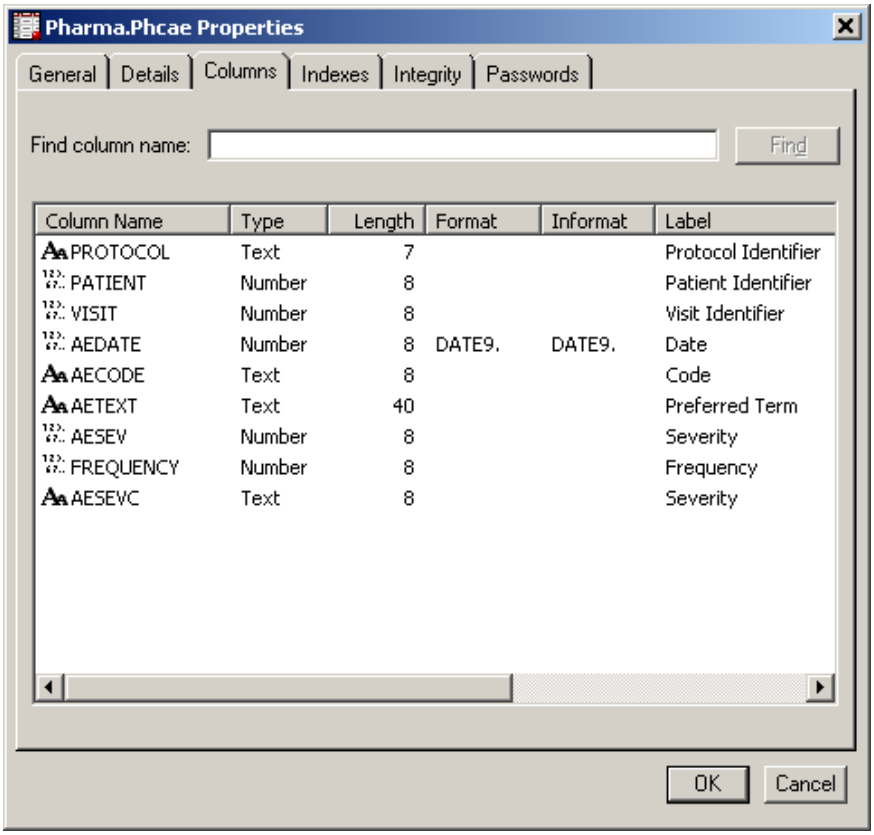
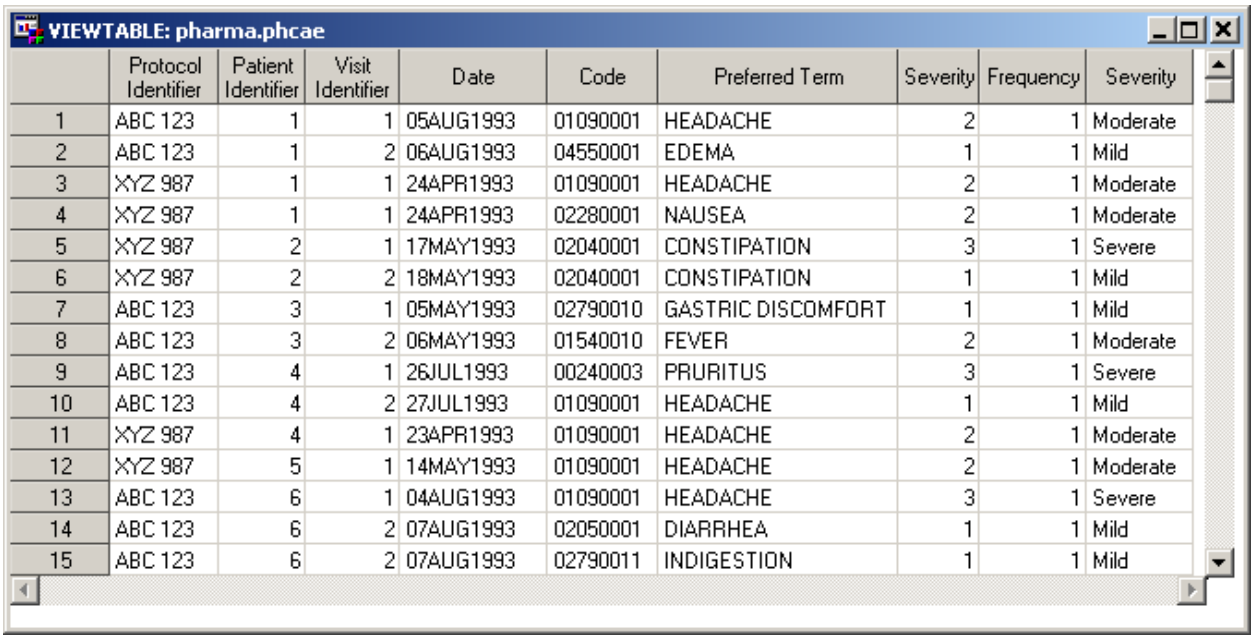


Figure 3. Columns in the SAS adverse event table.

The two columns AESEV and AESEVC represent the severity of the adverse event. As illustrated in Figure 4, the same data is represented two ways: AESEV contains a numeric value for the severity while AESEVC contains a character value. The SAS code later in this example makes use of both columns.



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 4. Partial view of the SAS adverse event table.

Notice that the Code column contains values that have leading zeroes. By using an ODS style override you can retain the leading zeros when importing the data into Excel.

GENERATING THE XML FILE

The following SAS code demonstrates the general technique needed to create an XML file that can be opened with Excel:

```
❶ ods listing close;
❷ ods tagsets.ExcelXP file="phdata.xml" path="path-to-output" style=Statistical;
    *  your SAS code here;
❸ ods tagsets.ExcelXP close;
```

The ODS statement at ❶ turns off the standard "line printer" ODS destination. We are only concerned with generating XML output.

The ODS statement at ❷ generates the XML output and stores it in a file named "phdata.xml". The STYLE attribute controls the output color scheme, and in this case is set to the Statistical style, which is new for 9. To see a list of ODS styles that are available for use at your installation, submit the following SAS code:

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

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

Complete SAS code that executes the PRINT and TABULATE procedures and generates an XML file can be found in the Appendix of this paper in the section "Original Code to Export SAS Output to Excel as XML".

OPENING THE XML FILE WITH EXCEL

To open the ODS-generated phdata.xml file with Excel, just select it using File ➨ Open.

Figure 5 (below) shows the XML file opened in Excel. The workbook consists of 4 worksheets. The two worksheets in the foreground are output from the TABULATE procedure. The two worksheets in the background are the result of the PRINT procedure. Each time ODS generates a new table, it is written to a new worksheet.

CORRECTING FORMATTING PROBLEMS

Although Excel does a fairly good job of formatting the phdata.xml file, there are still a few problems you should correct. Unfortunately, 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.

First, notice that the leading zeroes were dropped from the Code column. This is due to Excel applying the General format to the Code column data. By using an ODS style override and applying the Excel format 00000000 to the column (similar to the SAS Z8. format), we can instruct Excel to retain the leading zeroes (DelGobbo, 2003). Use the following PROC PRINT code to instruct Excel to use the 00000000 format, instead of the General format:

```
proc print data=pharma.phcae noobs label;
  by protocol;
  var patient visit aedate;
  var aecode / style={tagattr="\00000000"};
  var aetext aesev frequency aesevc;
run; quit;
```

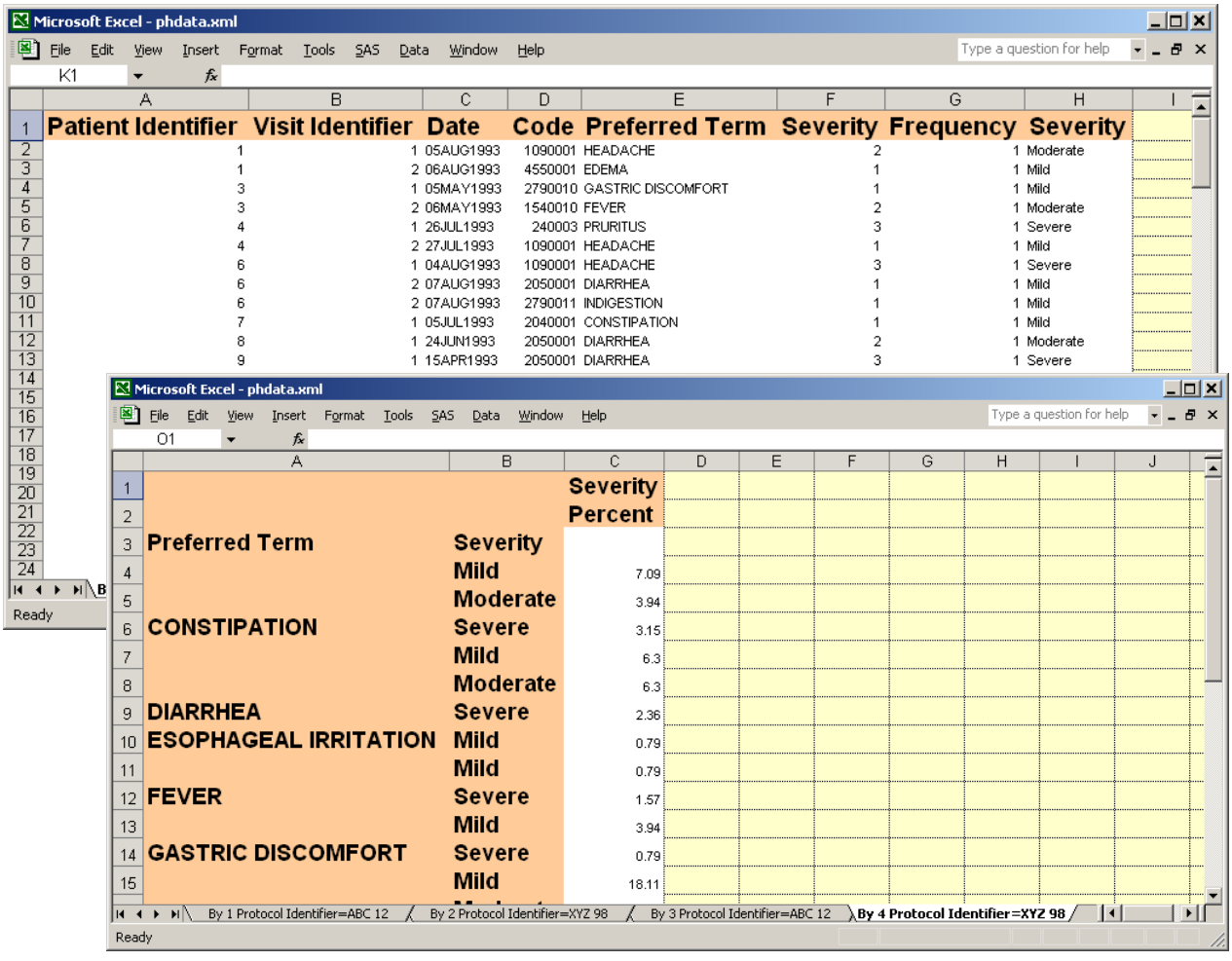


Figure 5. ODS XML output viewed with Microsoft Excel, after manually resizing columns.

The other problem to correct is that of the missing cell border lines (the Statistical style has no definition of border lines for header and data cells). Although you can correct this problem using an ODS style override as we did with the leading zero format problem, a more efficient solution is to create a new style that is tailored to Excel. The following code uses the TEMPLATE procedure to create a new style that adds border lines to the Statistical style:

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

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

*;
*  Apply cell borders to header and data cells.
*    1=None, 2=Thin, 3=Medium, 4=Thick
*;

❸ proc template;
    define style XLStatistical;
      parent = styles.Statistical;
      replace Header from HeadersAndFooters /
        borderwidth=2;
      replace RowHeader from Header /
        borderwidth=2;
      replace Data from Cell /
        font = fonts('docFont')
        background = colors('databg')
        foreground = colors('datafg')
        borderwidth=2;
    end;
  run; quit;
```

The LIBNAME statement at ❶ specifies where to store the new style. Although you can use the WORK library to temporarily store the style, a more efficient process would be to create the style one time and store it in a permanent library so that you can reference it in other SAS programs.

The ODS PATH statement at ❷ specifies the locations and the order in which ODS searches when looking for tagsets and styles. Note that the access mode for the location "myLib.tmplmst" is specified as "update", while "sashelp.tmplmst" is specified as "read". Because ODS searches the PATH in the order given, and because myLib.tmplmst has an “update” access mode, PROC TEMPLATE will store the style in a file named "tmplmst.sas7bitm" in the directory associated with the MYLIB library.

The TEMPLATE procedure code at ❸ creates the new style, named "XLStatistical". This style is based on the built-in "Statistical" style that was used to generate the output in Figure 5. The new style sets the border width to 2 for header and data cells, and adjusts font and color specifications for data cells. Details on how to use the TEMPLATE procedure to create and change styles can be found in chapter 5 of the ODS documentation (SAS Institute Inc., 1999).

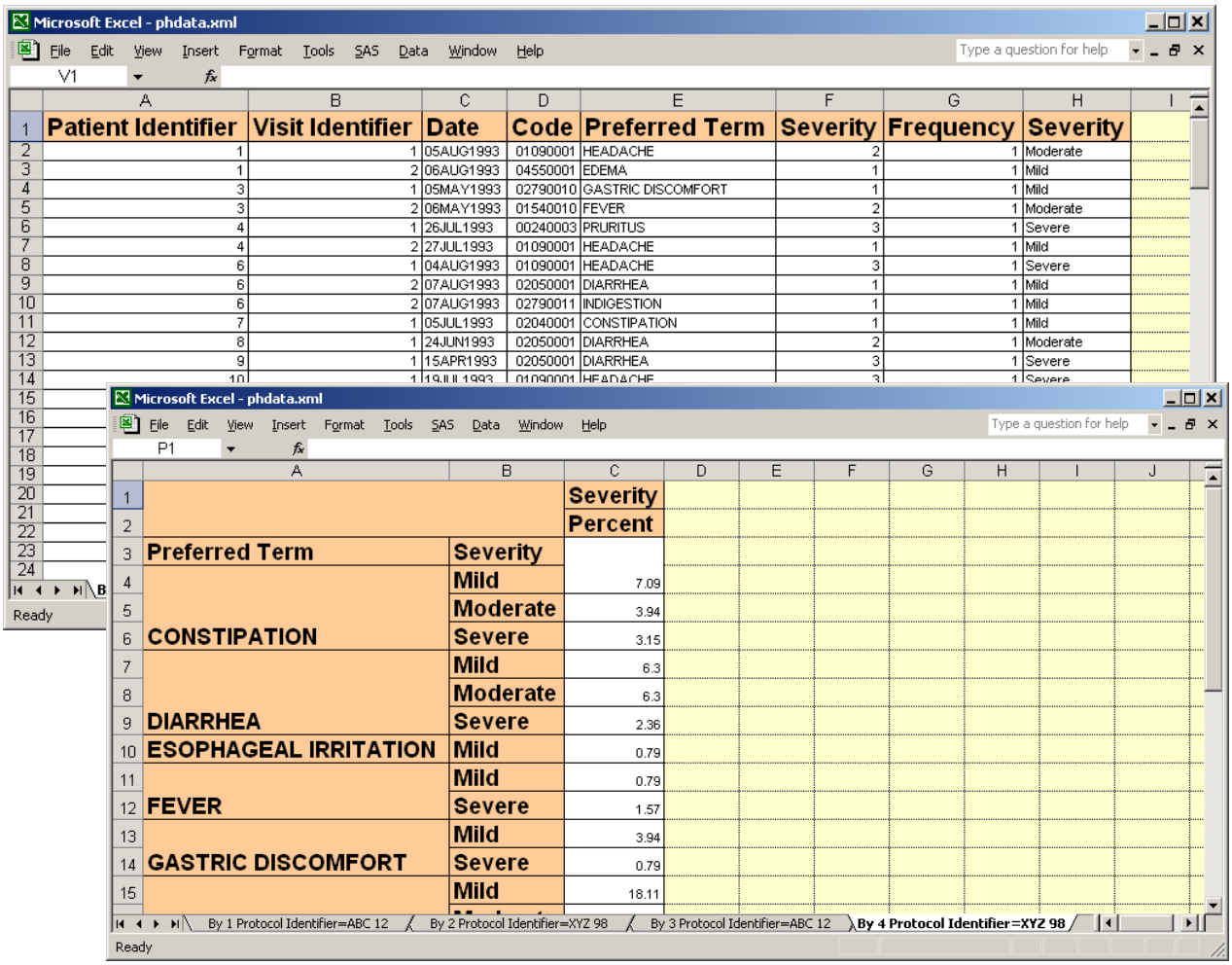
To use the new style in future SAS code, supply an ODS PATH statement and then reference the style by its name. For example, the following code is a modified version of the code used to generate the output in Figure 5:

```
libname myLib 'directory-for-style-and-tagset' access=read;

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

ods listing close;
ods tagsets.ExcelXP file="phdata.xml" path="path-to-output" style=XLStatistical;
  *  your SAS code here;
ods tagsets.ExcelXP close;
```

Figure 6 shows the corrected formats of the XML file as viewed in Excel. All of the table cells have border lines, and the Code column retains the leading zeroes. The complete SAS code to create this XML can be found in the Appendix of this paper in the section "Corrected Code to Export SAS Output to Excel as XML ".



Microsoft Excel - phdata.xml

Patient Identifier	Visit Identifier	Date	Code	Preferred Term	Severity	Frequency	Severity
1	1	05AUG1993	01090001	HEADACHE	2	1	Moderate
1	2	06AUG1993	04550001	EDEMA	1	1	Mild
3	1	05MAY1993	02790010	GASTRIC DISCOMFORT	1	1	Mild
3	2	06MAY1993	01540010	FEVER	2	1	Moderate
4	1	26JUL1993	00240003	PRURITUS	3	1	Severe
4	2	27JUL1993	01090001	HEADACHE	1	1	Mild
6	1	04AUG1993	01090001	HEADACHE	3	1	Severe
6	2	07AUG1993	02050001	DIARRHEA	1	1	Mild
6	2	07AUG1993	02790011	INDIGESTION	1	1	Mild
7	1	05JUL1993	02040001	CONSTIPATION	1	1	Mild
8	1	24JUN1993	02050001	DIARRHEA	2	1	Moderate
9	1	15APR1993	02050001	DIARRHEA	3	1	Severe
10	1	19JUL1993	01090001	HEADACHE	3	1	Severe

Microsoft Excel - phdata.xml

	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

By 1 Protocol Identifier=ABC 12 | By 2 Protocol Identifier=XYZ 98 | By 3 Protocol Identifier=ABC 12 | By 4 Protocol Identifier=XYZ 98

Ready

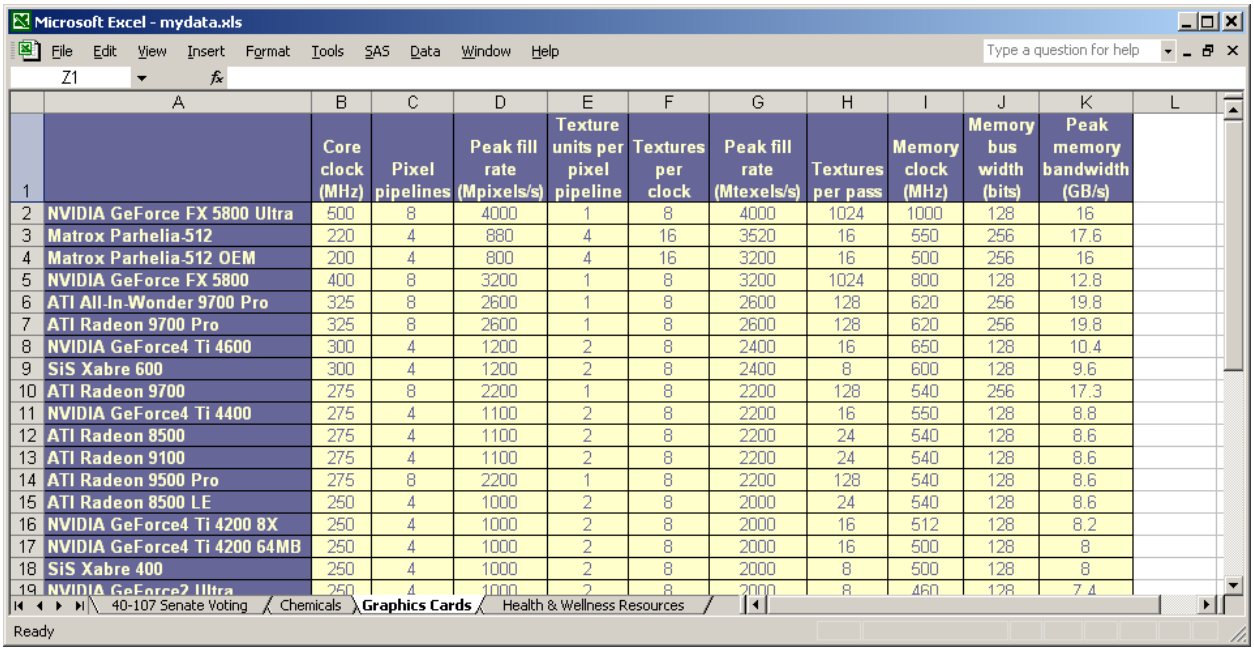
Figure 6. Corrected ODS XML output viewed with Microsoft Excel, after manually resizing columns.

MOVING EXCEL DATA TO SAS

Figure 7 (below) shows the Excel workbook we want to import into SAS. The workbook contains 4 worksheets. We want to import the data from each worksheet into a different SAS table, using 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.**

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 so, in Excel select File ➡ Save As, and choose "XML Spreadsheet". Specify a name and directory for the file. For the purpose of this paper, we saved the XML in a file named "mydata.xml".



	A	B	C	D	E	F	G	H	I	J	K	L
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 7. Excel workbook with data to import into SAS.

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 system 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 is installed on a different machine from Excel. If neither of these options is available to you, move the file from the Windows machine to the machine that has SAS installed using FTP or some other method.

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 worksheet in Figure 7 is imported into a SAS table, the name of the worksheet, "Chemicals" (for example) is used as the SAS table name. However, the technique of using the worksheet name for the table name does not always work. For example, a worksheet named "40-107 Senate Voting" cannot 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". Likewise, 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. Note that none of the column labels in Figure 7 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. The table below shows examples of how some of the Excel column labels in Figure 7 will be converted to SAS column names.

Excel Column Label	SAS Column Name	SAS Column Label
(blank)	A unique 32-character name starting with "_"	.
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.

READING EXCEL XML INTO SAS

You could use the SXLE and an XMLMap of your own creation to import an Excel XML file into SAS, but because the Excel XML format and the data conversion issues are quite complex, SAS provides an XMLMap specific to Excel and corresponding SAS code that loads the Excel XML into SAS tables. As mentioned earlier, both components as well as a SAS macro to import the XML data, are available for download from the SAS Presents Web site. These components may ship with future releases of SAS, so watch the Base SAS Community Web site for further details (http://support.sas.com/rnd/base/)

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 to load 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 XML file into SAS tables.

The easiest way to explain how to use the macro is with a few examples. First, we will consider importing the XML workbook shown in Figure 7 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 this code, making sure to include the appropriate directory paths:

```
❶ %include 'loadxl.sas';

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

The statement at ❶ makes the XLXP2SAS macro available to SAS. The statement at 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 the library used to store the SAS tables by specifying the LIBRARY argument of 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 created as a result of importing the Excel workbook shown in Figure 7. Note that while the table names may look a bit odd, the actual worksheet names are used in the SAS labels.

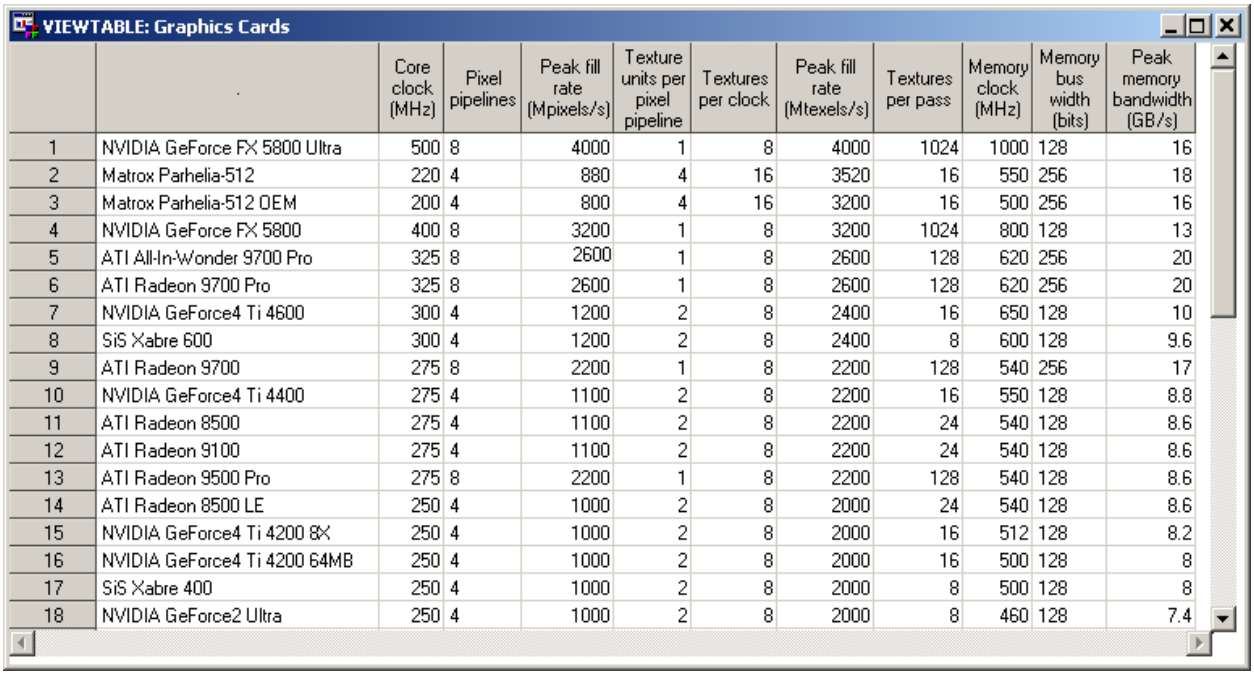
SAS Table Name	SAS Label
Chemicals	Chemicals
Graphics_cards	Graphics Cards
Health___wellness_resources	Health & Wellness Resources
_0_107_senate_voting	40-107 Senate Voting

Table 2. SAS tables created from Excel workbook.

Figure 8 (below) shows a portion of the "Graphics_cards" table. By comparing Figures 7 and 8, 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.

Up to this point the assumption is that the XML file resides on the same machine as SAS or it was available via a network accessible 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);
```



VIEWTABLE: Graphics Cards

		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 8. SAS Graphics_cards table, created by importing an Excel workbook.

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 found in the Appendix of this paper in the section "XLXP2SAS Macro Documentation".

XLXP2SAS DRAWBACKS AND LIMITATIONS

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

1. 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, as the temporary files are automatically cleaned up after XLXP2SAS runs (unless the argument CLEANUP=N was specified).
2. The data in every worksheet 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.
3. 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.
4. If you specify HASLABELS=N, the column names in the SAS table(s) will be of the form "COLUMN1", "COLUMN2", "COLUMN3" and so on, and the respective column labels will be "Column 1", "Column 2" and "Column 3".

USING SAS SERVER TECHNOLOGY TO BRING SAS OUTPUT INTO EXCEL

If you have licensed SAS/IntrNet® software, you can dynamically incorporate SAS output into Excel 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 consist of any combination of DATA step, PROC, MACRO, or SCL code. Thus, all of the code shown up to this point can be executed using either Application Dispatcher or the 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.

For an example, the PRINT and TABULATE procedure code we have been using can be modified to use a WHERE clause instead of BY-group processing by replacing the two existing BY statements with the following statement:

```
where &WHERE_CLAUSE;
```

A user can now specify any valid WHERE clause on the URL, and that WHERE clause will be used by your SAS code. The code is executed on the SAS server via a URL such as:

```
http://path-to-server?_program=program-name&where_clause=aesev eq 3
```

Because of the changes to the code, the value specified in the URL for the parameter named WHERE_CLAUSE is used by the PRINT and TABULATE code.

To dynamically create SAS output and place it into an Excel workbook, type the URL into the Open dialog box of Excel. If you want to run your SAS code from a Web browser and have the output automatically included in Excel, use the APPSRV_HEADER function to set the Content-type header for the XML file, as illustrated in the sample code below:

```
%let rc = %sysfunc(appsrv_header(Content-type, application/vnd.ms-excel));
ods listing close;
ods tagsets.ExcelXP file=_WEBOUT …;
  * your SAS code here;
ods tagsets.ExcelXP close;
```

When the code above is executed from a Web browser, Excel will start and the XML content generated by ODS will be output to a new workbook.

These are just two samples of how you can take advantage of SAS server technologies. For details about the operation of the Application Dispatcher or the Stored Process Server, refer to the respective documentation for those servers.

CONCLUSION

SAS 9.1 enhances support for XML. ODS provides an easy way to export your data to Excel workbooks that contain multiple worksheets. While it may 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.

SAS Institute continues to work toward better Microsoft Office integration, and future releases of SAS software will provide even more robust means of using SAS content with Office applications.

APPENDIX

ORIGINAL CODE TO EXPORT SAS OUTPUT TO EXCEL AS XML

```
ods listing close;
ods tagsets.ExcelXP file="phdata.xml" path="directory-for-XML" style=Statistical;
  proc print data=pharma.phcae noobs label;
    by protocol;
    var patient visit aedate aecode aetext aesev frequency aesevc;
  run; quit;

  proc tabulate data=pharma.phcae;
    by protocol;
    var aesev;
    class aetext aesevc;
```

```
    classlev aetext aesevc;
    table aetext*aesevc,aesev*pctn;
    keyword all pctn;
    keylabel pctn='Percent';
  run; quit;
ods tagsets.ExcelXP close;
```

CORRECTED CODE TO EXPORT SAS OUTPUT TO EXCEL AS XML

```
libname myLib 'directory-for-style' access=read; * location where style is stored;

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

ods listing close;
ods tagsets.ExcelXP file="phdata.xml" path="directory-for-XML" style=XLStatistical;
  proc print data=pharma.phcae noobs label;
    by protocol;
    var patient visit aedate;
    var aecode / style={tagattr="\00000000"};
    var aetext aesev frequency aesevc;
  run; quit;

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

XLXP2SAS MACRO DOCUMENTATION

Below is a list of the arguments supported by the XLXP2SAS macro. All arguments require a value, except where a default value is indicated.

Argument	Description	Default Value
EXCELFILE	The name and path for the Excel XML file you want to import to SAS. Do not use quotes in this value. To specify a SAS FILEREF instead of a file, use FILEREF:*fref*, where *fref* is the FILEREF.	
MAPFILE	The name and path of the SAS XMLMap for reading Excel XML files. Do not use quotes 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 SAS-provided XMLMap from the SAS Presents Web site (http://support.sas.com/saspresents)	
LIBRARY	The name of the SAS library where imported tables are stored.	WORK
HASLABELS	Controls whether or not 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 will be 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 or not to delete temporary SAS files and also whether or not to deassign FILEREFs that were used when importing the Excel data to SAS. FILEREFS that you explicitly assign with a FILEREF statement will not be deassigned. 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

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

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

Plemmons, H. (2003), "How to Access PC File Data Objects Directly from UNIX", Proceedings of the Twenty-eighth Annual SAS Users Group International Conference, 28, CD-ROM. Paper 156. Available http://www2.sas.com/proceedings/sugi28/156-28.pdf

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

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

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

FURTHER READING

"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.