Techniques for SAS® Enabling Microsoft® Office in a Cross-Platform Environment

Vincent DelGobbo, SAS Institute Inc., Cary, NC

ABSTRACT

With the ubiquity of the Microsoft Office suite of products, you often need to use SAS data and analytical results in the Office products Excel or Word. But transferring data from SAS to the Office applications can be difficult when the SAS System is installed on a non-Windows platform, such as OpenVMS™, UNIX® or OS/3790®. This paper provides techniques for integrating the analytic processing power of the SAS system, irrespective of platform, with Microsoft Excel and Word. The use of the Output Delivery System (ODS) is also discussed.

CROSS-PLATFORM TECHNIQUES

The situation: your SAS data reside on a platform other than Windows, and you don't have the SAS System installed on the Windows machine where you have Excel and Word. How do you get the data, SAS, and the Office products together? The most straightforward way to get SAS data into your Office documents is to generate files that can be imported directly into Excel and Word. The methods described below will work on any platform, and all follow the same process: you generate the file, move the file to the Windows system where Office is installed, via FTP for example, and then use either Excel or Word to open the file. In later sections of this paper there are examples of how to automate the entire process using SAS server technologies.

COMMA SEPARATED VALUE (CSV) FILE

Perhaps the simplest way to transfer data across platforms is by utilizing a Comma Separated Value file. A CSV file is a plain-text file consisting of columns of data delimited by commas. It is generally used to transport raw data, not formatted analytical results. Figure 1 shows the CSV file corresponding to the SASHELP.CLASS data set.

Excel can read this type of file and place the data into discrete cells. In contrast, if you open this type of file with Word, a Word table is not be generated, and you end up with basically what you see in Figure 1, a list of comma separated values.

There are a number of ways to generate CSV files. Starting in Release 8.2 of the SAS System, you can generate CSV files from a data set using the Output Delivery System (ODS):

```
ods listing close;
ods csv file="class.csv";
  proc print data=sashelp.class noobs; run;
ods csv close;
```

If you license SAS/IntrNet® software, you can use the DS2CSV macro to create the file:

```
%ds2csv(data=sashelp.class,
        csvfile=class.csv, runmode=b);
```

Other ways of creating CSV files include using the SAS Export Wizard and writing your own custom DATA step code.

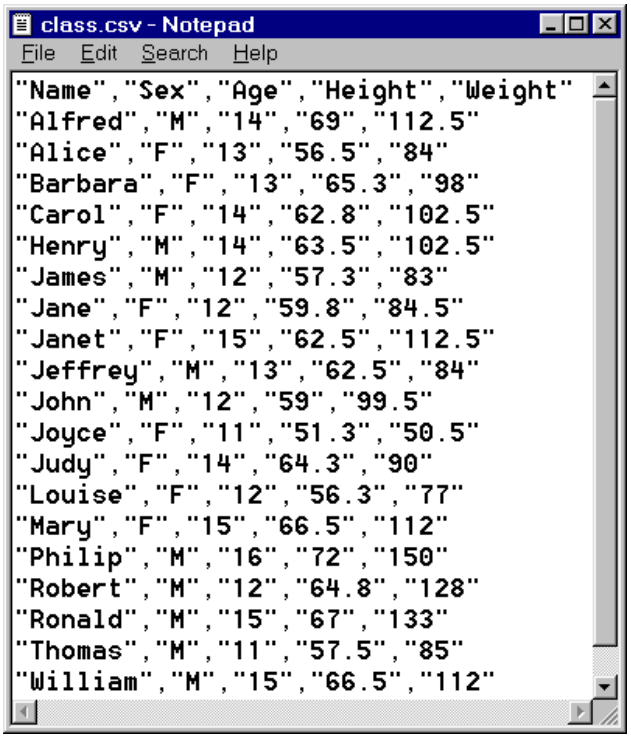


Figure 1. The CLASS data set rendered as a CSV file.

SYMBOLIC LINK (SYLK) FORMAT

SYLK is a plain-text data interchange format created in the mid-1980's. Excel 97 and later have the ability to import and export worksheets in the SYLK format. Although SYLK supports many Excel features, not all text formats and formulas are supported ("SYLK (Symbolic Link) Format"). There are other limitations, as well.

Documentation on SYLK is sparse. Microsoft published a document covering SYLK's structure and format ("sylksum.doc"), and similar information is located at the Wotsit Web site (http://www.wotsit.org/). Other pertinent information is available in the comp.apps.spreadsheets newsgroup FAQ (FAQ).

After learning about SYLK, you can write DATA step code to create a SYLK file that can be imported into Excel. An example of such code can be found in the Appendix.

You can also use ODS to produce SYLK files. Starting with Release 8.2 of the SAS System you can create custom tagsets to define the exact type of output you desire. Information on creating tagsets and a customer-contributed tagset for the SYLK format are available from the SAS Web site ("Creating Customized Tagsets to Use With ODS and XML LIBNAME Engine"). The supplied tagset can be modified and extended to meet your specific needs. Below is sample code that illustrates how to create a SYLK file using ODS, after the tagset code is imported and executed.

```
title; footnote;

ods path work.template(update)
         sashelp.tmplmst(read);
```

```
*  Include customer-contributed tagset code;
%include "sylkTagset.sas";

ods listing close;
ods markup type=sylk file="class.slk";
  proc print data=sashelp.class noobs; run;
ods markup close;
```

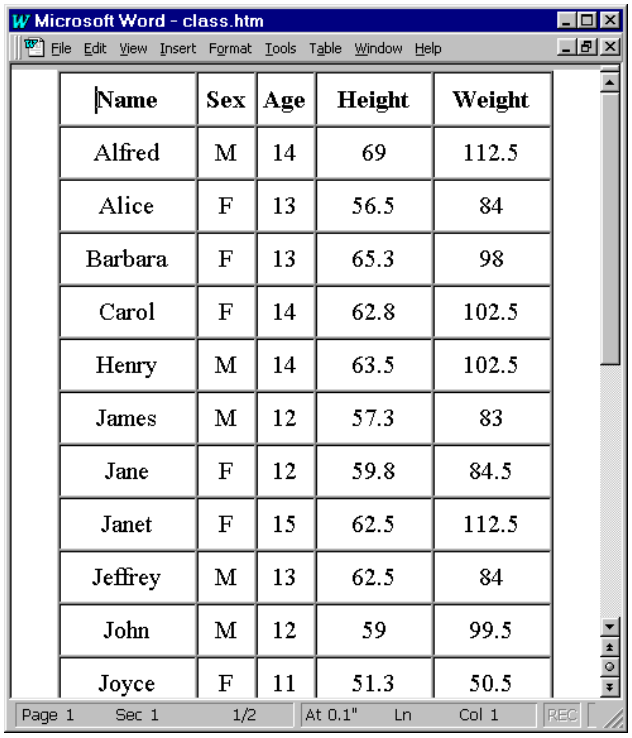
HYPERTEXT MARKUP LANGUAGE (HTML)

Because versions 97 and later of Excel and Word are capable of importing HTML files, and SAS software is quite adept at generating HTML, it is a simple matter to export SAS data and results as HTML, and then open those HTML files with Excel or Word.

Starting with Release 6.09E of the SAS System, you can use the HTML Formatting Tools ("HTML Formatting Tools") to generate HTML output. The code below uses the HTML Data Set Formatter to generate HTML from the CLASS data set.

```
title; footnote;
%ds2htm(data=sashelp.class,
        htmlfile=class.htm,
        twidth=50, septype=none,
        pagepart=body);
```

Figure 2 shows the results of opening the HTML file with Word 97. Note that the HTML table has been automatically converted to a Word table.



Name	Sex	Age	Height	Weight
Alfred	M	14	69	112.5
Alice	F	13	56.5	84
Barbara	F	13	65.3	98
Carol	F	14	62.8	102.5
Henry	M	14	63.5	102.5
James	M	12	57.3	83
Jane	F	12	59.8	84.5
Janet	F	15	62.5	112.5
Jeffrey	M	13	62.5	84
John	M	12	59	99.5
Joyce	F	11	51.3	50.5

Figure 2. Portion of DS2HTM HTML output in Word 97.

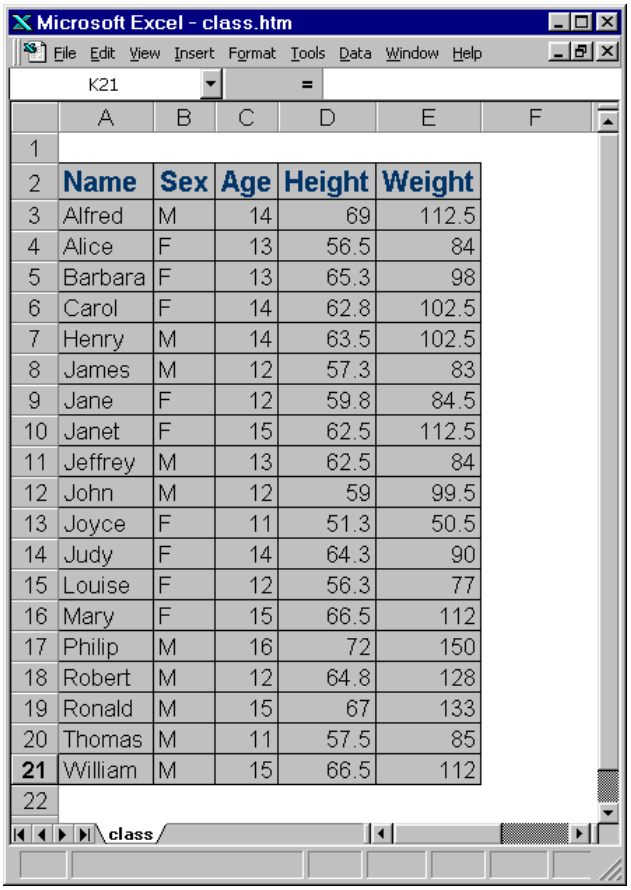
Starting in Version 8 of the SAS System, you can use ODS to generate HTML output for inclusion in Excel and Word:

```
options nocenter nodate nonumber;
title; footnote;

ods listing close;
ods html file="class.htm"
         (no_top_matter no_bottom_matter)
         rs=none;
  proc print data=sashelp.class noobs; run;
ods html close;0
```

Figure 3 shows the results of opening the HTML file with Excel 97. Note that the HTML table has been automatically converted to an Excel table.

ODS is extremely flexible. By making use of ODS styles and tagsets, you can generate highly specialized output for use in Excel or Word.



	A	B	C	D	E	F
1						
2	Name	Sex	Age	Height	Weight	
3	Alfred	M	14	69	112.5	
4	Alice	F	13	56.5	84	
5	Barbara	F	13	65.3	98	
6	Carol	F	14	62.8	102.5	
7	Henry	M	14	63.5	102.5	
8	James	M	12	57.3	83	
9	Jane	F	12	59.8	84.5	
10	Janet	F	15	62.5	112.5	
11	Jeffrey	M	13	62.5	84	
12	John	M	12	59	99.5	
13	Joyce	F	11	51.3	50.5	
14	Judy	F	14	64.3	90	
15	Louise	F	12	56.3	77	
16	Mary	F	15	66.5	112	
17	Philip	M	16	72	150	
18	Robert	M	12	64.8	128	
19	Ronald	M	15	67	133	
20	Thomas	M	11	57.5	85	
21	William	M	15	66.5	112	
22						

Figure 3. ODS HTML output in Excel 97.

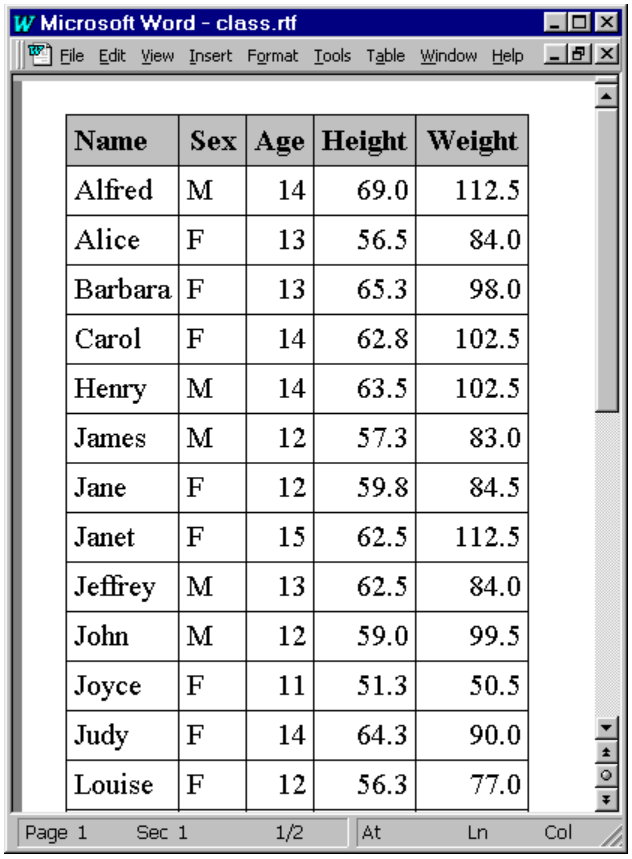
RICH TEXT FORMAT (RTF)

ODS is also capable of generating RTF output. With minor modifications to the ODS statement that was used to generate the HTML output, you can generate RTF output:

```
options nocenter nodate nonumber;
title; footnote;

ods listing close;
ods rtf file="class.rtf";
  proc print data=sashelp.class noobs; run;
ods rtf close;
```

When the resulting RTF file is opened in Word 97 or later, it should appear very similar to Figure 3 if the RTF file was created in Release 8.0 of the SAS System. In later releases of the SAS System, the RTF file will appear as in Figure 4. Note that Excel cannot import an RTF file as of the time of this writing.



Name	Sex	Age	Height	Weight
Alfred	M	14	69.0	112.5
Alice	F	13	56.5	84.0
Barbara	F	13	65.3	98.0
Carol	F	14	62.8	102.5
Henry	M	14	63.5	102.5
James	M	12	57.3	83.0
Jane	F	12	59.8	84.5
Janet	F	15	62.5	112.5
Jeffrey	M	13	62.5	84.0
John	M	12	59.0	99.5
Joyce	F	11	51.3	50.5
Judy	F	14	64.3	90.0
Louise	F	12	56.3	77.0

Figure 4. Portion of ODS RTF output in Word 97.

You can also generate graphics for use in Word documents by using the ODS RTF destination. Figure 5 shows the mean weight of students by sex and age based on the data in the CLASS data set. The GCHART procedure and the GIF driver were used to generate this static graphic. Refer to the Appendix for the SAS/GRAPH® code used to generate the RTF output.

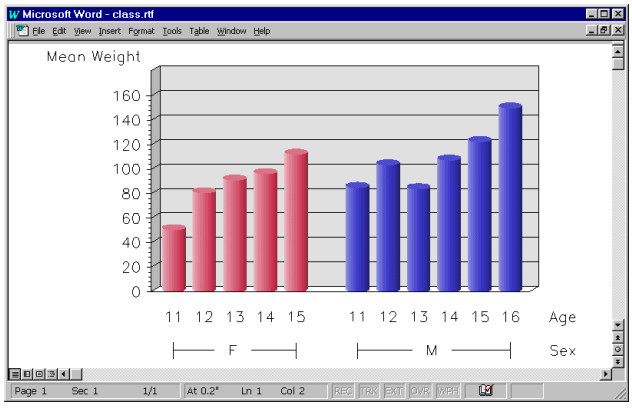


Figure 5. ODS RTF GIF graphic in Word 97.

By modifying the SAS code to use the ACTIVEX driver instead of the GIF driver, you can use the SAS/GRAPH ActiveX Control in your Word document. The advantage of using this technique is that you get a somewhat dynamic graphic inserted into your Word document. As illustrated in Figure 6, you can access the control's popup menu to alter the appearance of the graphic. You can even change the category and response variables along with the displayed statistic to come up with a completely different chart.

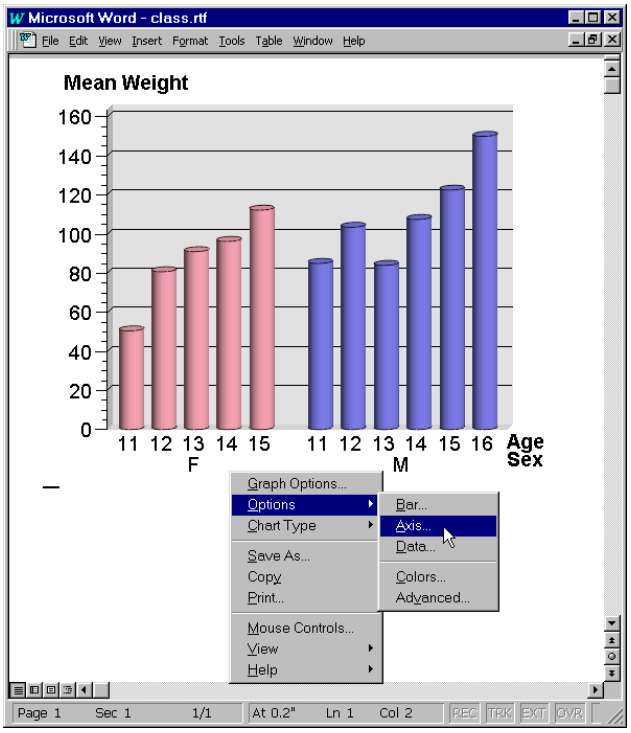


Figure 6. ODS RTF ActiveX graphic in Word 97.

AUTOMATING THE PROCESS USING SAS SERVER TECHNOLOGY

Up to this point, all the techniques presented for using SAS content in Excel and Word have been primarily manual: you start SAS, submit code to generate an output file, move that file to the machine where Office is installed, and then open the file with the appropriate application. This process can be automated by making use of a SAS server technology such as the SAS/IntrNet® Application Dispatcher or the SAS Integration Technologies software. These techniques will also work if your SAS data does reside on the Windows platform.

Automating the process first involves storing the SAS code in a file that can be accessed by the server. The server can be running on any platform, such as OS/390, OpenVMS, Unix or Windows. You then start the appropriate Office application and run the SAS code *from within the Office application*, providing you have a network connection to the server. The resulting SAS output is then imported into your Office document.

SAS/INTRNET APPLICATION DISPATCHER

The Application Dispatcher, a component of SAS/IntrNet software, enables you to execute SAS programs from a Web browser or any other client that can open an HTTP connection to the SAS server. These SAS programs 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 the Dispatcher. Small modifications must be made to the SAS code in order to make it usable via the Web. In the case of the DS2HTM sample code, change `htmlfile=class.htm` to `htmlfref=_WEBOUT, runmode=s`. For the ODS samples, change `file="`*`file-name`*`"` to `file=_WEBOUT`.

The Application Dispatcher is comprised of three components, as shown in Figure 7:

- Thin Client
- SAS Application Broker
- SAS Application Server

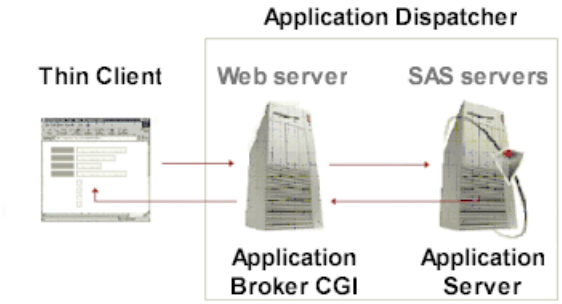


Figure 7. Application Dispatcher Architecture and Execution Flow

The Thin Client is simply a Web browser or any other application that can open an HTTP connection via a URL. The Thin Client interacts with SAS programs that execute on the SAS Application Server. In our case, Microsoft Excel or Word serve as the Thin Client.

The SAS Application Broker is a very lightweight program that manages the communication between the Thin Client and the SAS Application Server using the Common Gateway Interface (CGI). It must be installed on your Web server. You do not need CGI programming experience to use the Application Dispatcher.

The final component of the Application Dispatcher is the SAS Application Server. The server executes the SAS programs that you write and the results are returned to the Thin Client via the Application Broker.

So, the entire process starts when the Thin Client opens a connection to the Application Broker and requests that a program be executed. The Application Server executes the requested program and then returns the resulting content to the Application Broker, which in turn returns the content to the Thin Client.

For detailed information on the operation of the Application Dispatcher, refer to the Application Dispatcher documentation ("Application Dispatcher, Release 8.2").

INTEGRATION TECHNOLOGIES

Another method for automating the inclusion of SAS content into Excel or Word involves the Integrated Object Model (IOM). IOM is a key component of the SAS Integration Technologies software, and provides distributed object interfaces for conventional SAS features. While there are a number of ways to use IOM to include SAS content in your Office application, one method is to develop a Java Server Page (JSP) or Active Server Page (ASP)-based application server that mimics the functionality of the SAS/IntrNet Application Dispatcher. If you are interested in developing your own server, refer to the Stored Process documentation available from the Integration Technologies Web site ("Stored Processes").

Work is currently under way to develop an IOM-based server that has essentially the same features as the Application Dispatcher. Refer to the Future Directions section later in this paper for more information.

ENABLING THE CLIENT TO ACCESS A SAS SERVER

There are a number of ways to enable Excel and Word to take advantage of SAS server technology. This section discusses three techniques: the Excel and Word Open dialog box, Excel Web Queries and Visual Basic for Applications (VBA) macros. These are techniques that you can start using today. The following examples use the SAS/IntrNet Application Dispatcher as the SAS server, but you can also use an Integration Technologies server if one is available. The Future Directions section later in this paper discusses another technique for using SAS via Office applications.

EXCEL AND WORD OPEN DIALOG BOX

The Open dialog box of Excel and Word is typically used to open a file that resides on a local or network drive. This dialog box is accessed by selecting File ➨ Open…. By specifying a URL in the "File name" field, you can import content from a Web server. In the case of an Application Dispatcher program, the URL is of the form:

```
http://web-
server/scripts/broker?_program=program-name
```

where *web-server* represents the name of your Web server and *program-name* is the name of the Dispatcher program you wish to execute. For Excel, your SAS program must generate HTML, SYLK, or plain text as output, while Word accepts HTML, RTF, or plain text.

Using this technique you can execute SAS logic ***on any platform*** and include the results in Excel or Word without ever leaving the application.

EXCEL WEB QUERIES

In Excel 97 and later, Web Queries enable you to incorporate HTML from a Web server, a local drive, or a network drive, directly into your worksheet. To do so you must first create a Web Query file that contains the URL of the HTML you wish to retrieve. In the case of an Application Dispatcher program, that Web Query file would look something like this:

```
WEB
1
http://web-
server/scripts/broker?_program=program-name
```

where *web-server* represents the name of your Web server and *program-name* is the name of the Dispatcher program you wish to execute. Note that the Application Dispatcher program may be run on any platform, and **must** output HTML. Any number of parameters can be passed to the SAS program using the URL. The parameter values can be hard coded or dynamic, enabling you to specify the value each time the query executes. Documentation on creating Web Queries is available from Microsoft ("XL97: How to Create Web Query (.iqy) Files.").

From within Excel, you can execute a Web Query by selecting Data ➨ Get External Data ➨ Run Web Query….

You can then navigate to the Web Query file and execute it. After the Application Dispatcher services your request and runs your program, the resulting HTML is automatically included in your worksheet.

If you save the Workbook and open it at a later date, you can easily refresh the data from the Web Query by right clicking on a table cell and choosing "Refresh Data".

VISUAL BASIC® FOR APPLICATIONS (VBA) MACROS

Microsoft Visual Basic for Applications is a powerful development environment for rapidly customizing desktop packaged applications and integrating them with existing data and systems ("VBA Overview"). VBA offers a sophisticated set of programming tools based on the Microsoft Visual Basic development system, and is included as part of the Office 97 suite of products.

A detailed discussion of VBA is beyond the scope of this paper, but some basic examples of integrating SAS content with Excel and Word are provided. The difference between VBA and Web Queries is that *any* content can be returned to the Thin Client, not just HTML. Furthermore, this technique, with some modification, can be used to pipe SAS content into any VBA enabled application.

Because Excel and Word can execute VBA macros, you can write macros to execute SAS/IntrNet Application Dispatcher programs and retrieve the SAS output for inclusion into your document. Generally this output consists of HTML, RTF (Word only), CSV (Excel only), or plain text. Note that the SAS program can reside on any SAS/IntrNet supported platform, and is not limited to just Windows.

Sample VBA code for retrieving HTML output is located in the Appendix. Instructions for modifying the code to import other content types are also provided. In general, however, follow these steps to insert and run VBA code from within an Office application:

1. Start the Office application and access the Visual Basic editor by pressing the key sequence Alt+F11 or by selecting Tools ➨ Macro ➨ Visual Basic Editor.
2. Copy and paste the code into a new code module and save the module.
3. Close the Visual Basic editor and return to your Office document.
4. Run the macro by pressing the key sequence Alt+F8 or by selecting Tools ➨ Macro ➨ Macros….
5. Choose the macro you wish to run and click the Run button.

For more information about Visual Basic for Applications, consult one of the many publications on this topic. You may also find the Excel "Visual Basic for Applications 101" Web site of interest ("Microsoft Excel for Windows"). Although the information presented is Excel-specific, some of it is also applicable to Word.

Unfortunately, your macros have no user interface. In order to provide a user interface, you can, via the Visual Basic Editor, build a User Form and associate it with the macros. These User Forms are very similar to standard Visual Basic forms, and have the appearance of native Windows dialogs.

FUTURE DIRECTIONS

While the techniques mentioned thus far are all available for your immediate use, this section discusses development that is currently in progress.

SAS STORED PROCESS SERVICE AND CONTROL

The future IOM-based server, mentioned earlier, is currently being referred to as the SAS Stored Process Service and will have essentially the same features as the SAS/IntrNet Application Dispatcher. The Stored Process Service will provide non-CGI based access to your SAS programs stored on a SAS server. Since it operates in much the same way as the Dispatcher, it will provide cross-platform access to your SAS data and analytic results.

The server-based samples shown previously will run on this server after slight modification: you will have to change the URL to point to the Stored Process Service instead of the Application Dispatcher.

One drawback of the server-based samples shown thus far is that none of them have a customizable user interface. In order to generate a user interface for VBA macros, you need to learn how to build User Forms and write custom VBA code to tie the forms to the macros. If you are not willing to take on this task, you can make use of the SAS Stored Process Control (SPC) which will be available in a future release of SAS software.

The Stored Process Control is an HTML-based user interface to your SAS server program. The SPC can run inside Excel, Word, and a number of HTML editors. With the SPC, instead of creating VBA User Forms and macros, you merely need to develop HTML pages that act as a user interface to your SAS logic. Figure 8 shows a prototype of a Stored Process Control hosting an HTML user interface.

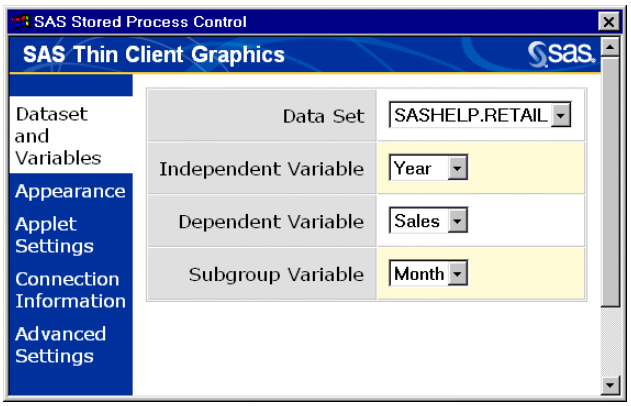


Figure 8. Prototype SAS Stored Process Control hosting an HTML user interface.

The HTML page(s) that you generate can be as simple or as complex as you desire. The sample shown in Figure 8 is fairly complex, making use of 5 different HTML pages. The "Data Set and Variables" page is currently visible, and the other pages, such as "Appearance" and "Applet Settings", are displayed by clicking on the text in the navigation bar on the left side.

After you navigate through the pages and make appropriate selections, you close the dialog to execute the SAS server program. The SPC automatically retrieves the SAS content and inserts it into Excel or Word. Using the Stored Process Control simplifies the automation: you no longer have to be concerned with VBA macros or User Forms, as the SPC replaces both of them. All you need to do is create SAS server based programs (or modify legacy code) and develop an HTML-based user interface. The SPC takes care of the rest.

WINDOWS-SPECIFIC TECHNIQUES

While the main topic of this paper is accessing SAS content that resides on non-Windows platforms, there are two useful, Windows-specific techniques for creating SAS content that can be imported into Excel or Word: the SAS Open Database Connectivity (ODBC) drivers and Dynamic Data Exchange (DDE).

ODBC DRIVERS

The SAS ODBC drivers give you the ability to query SAS or other ODBC-compliant data sources and include the resulting data in Excel. Mehler (1998) outlines how to configure and use the SAS ODBC and the Universal ODBC drivers and even provides a usage scenario for including SAS data in Excel.

Once the appropriate ODBC driver is installed and configured, you can use the Microsoft Query Wizard to construct a query and retrieve the desired information from the SAS data set(s). In Excel, access the Query Wizard by selecting Data ➨ Get External Data ➨ Create New Query….

Refer to Mehler's paper for further details on accessing your SAS data using the ODBC drivers. SAS "Technical Support Document 626" provides information on configuring the ODBC driver for Version 8 of SAS software.

DYNAMIC DATA EXCHANGE (DDE)

In order to use DDE, you must first start the target Office application, or you can try to start the Office application on the fly using some of the techniques proposed by Vyverman (2001) or Roper (2000). Once the target application is started, you can then proceed to move the SAS content to that application.

The example below illustrates how you can export all data values from any SAS data set into an Excel worksheet. In this case, your worksheet will contain all the records from the data set SASHELP.CLASS, and assumes that Excel is already started and the current workbook has a worksheet named "Sheet1".

```
options noxsync noxwait;

*  Must use upper case names below;
%let libname=SASHELP; %let memname=CLASS;

proc sql noprint;
  select nobs, nvar
  into :nRows, :nCols
  from dictionary.tables
  where libname eq "&libname" and
        memname eq "&memname" and
        memtype eq 'DATA';
quit;

*  Excel starting column and row;
%let startCol=1; %let startRow=1;

%let endCol = %eval(&startCol + &nCols -1);
%let endRow = %eval(&startRow + &nRows -1);

filename excel dde
"Excel|Sheet1!R&startRow.C&startCol.:R&endRow
.C&endCol" notab;

data _null_;  set &libname..&memname;
  file excel;
  put (_char_)    ($20. '09'x) '09'x
      (_numeric_) (8. '09'x);
run;
```

The SQL procedure is used to count the number of columns and rows in the SAS data set. You can choose the destination (starting) column and row in the worksheet, and the ending column and row will be automatically calculated. Finally, the DATA step code will export the data values directly into Excel.

A search of the SAS Technical Support Web site yields several relevant documents. Specifically, "Technical Support Document 325" covers the basics of using DDE to integrate SAS content with Excel, Word and other Windows applications.

CONCLUSION

There are several techniques you can use today to integrate SAS content with Excel and Word, and this paper has just scratched the surface of that topic. By using the SAS Output Delivery System, you can generate attractive content that can be immediately consumed by the Office product line, irrespective of the platform where the SAS System is installed.

Making use of SAS server technology serves to automate the process of integrating SAS content with Excel and Word. With the advent of the Stored Process Service and the Stored Process Control, near seamless integration will be achieved.

APPENDIX – CODE SAMPLES

Sample 1: SAS DATA step code to generate a simple SYLK format file.

```
data _null_;
file 'class.slk';

length dsid nCols nRows colNum rowNum rc 8
       varType  $1
       varLabel $256
       varValue $32767;

dsid = open("&libname..&memname", 'I');

nCols = attrn(dsid, 'NVAR');
nRows = attrn(dsid, 'NOBS');

put 'ID;PWXL;N;E';

do colNum = 1 to nCols;
  varLabel = varlabel(dsid, colNum);
  if compress(varLabel) eq ''
    then varLabel = varname(dsid, colNum);
  if colNum eq 1
    then put 'C;Y1;X1;K"' varLabel +(-1) '"';
    else put 'C;X' colNum +(-1) ';K"'
varLabel +(-1) '"';
end;

do rowNum = 2 to nRows+1;
  rc = fetchobs(dsid, rowNum-1);
  do colNum = 1 to nCols;
    if vartype(dsid, colNum) eq 'C'
      then varValue =
quote(trim(getvarc(dsid, colNum)));
      else varValue =
compress(put(getvarn(dsid, colNum),
best32.));
    if colNum eq 1
      then put 'C;Y' rowNum +(-1) ';X1;K'
varValue +(-1);
      else put 'C;X' colNum +(-1) ';K'
varValue +(-1);
  end;
end;

rc = close(dsid);

put 'E';

run;
```

Sample 2: SAS DATA step code to generate an RTF GIF graphic.

To generate an ActiveX graphic, change the device driver value from GIF to ACTIVEX.

```
options nocenter nodate nonumber;
title; footnote;
goptions reset=all;
```

```
goptions device=GIF
         hsize=6in
         vsize=3.5in
         xmax=6in
         ymax=3.5in
         ftext=simplex;

pattern1 color=lipk;
pattern2 color=vlib;

axis1 label=('Mean Weight');

axis2 label=('Age');

ods listing close;
ods rtf file="class.rtf";

proc gchart data=sashelp.class;
  vbar3d age / group=sex
               sumvar=weight
               patternid=group
               autoref
               discrete
               nozero
               shape=cylinder
               type=mean
               raxis=axis1
               maxis=axis2
               cframe=cxe0e0e0;
run; quit;

ods rtf close;
```

Sample 3: VBA macro code to include SAS HTML output in Microsoft Word.
To insert RTF output, follow these steps:

1. Copy the insertHTML macro and rename it to insertRTF.
2. Change the VBA code for insertRTF so that the file extension is rtf.
3. Supply the appropriate name of the Application Dispatcher program and the Web server.

(Advanced VBA programmers should make use of more sophisticated macros, including the use of User Forms, instead of continually copying and modifying the code.)

```
Sub insertHTML()
  Dim tempFile As String
  Dim tempDoc As Document

  '  Change extension for different
  '  output types.
  tempFile = makeTempFile() & ".htm"

  Set tempDoc = Documents.Open(
FileName:="http://web-
server/scripts/broker?_program=program-
name&_debug=0")

  insertContent tempDoc, tempFile

  Kill tempFile
End Sub

Function makeTempFile()
  Dim tempFile As String, i As Integer
  '  Assumes temp directory is stored in TEMP
  '  environment variable.
  tempFile = Environ("TEMP") & "\"
  For i = 1 To 8
      tempFile = tempFile &
Mid(LTrim(Str(CInt(Rnd * 10))), 1, 1)
  Next

  makeTempFile = tempFile
End Function

Sub insertContent(tempDoc  As Document, _
                  tempFile As String)
  tempDoc.SaveAs (tempFile)
  tempDoc.Close SaveChanges:=False
  Selection.InsertFile tempFile
End Sub
```

Sample 4: VBA macro code to include SAS HTML output in Microsoft Excel.
In principle, you should be able to use the code below to include CSV content, provided the Application Dispatcher program generates valid CSV output. However, some versions of Excel have a problem reading the CSV output directly. To get around this problem, follow these steps:

1. Copy the insertHTML macro and rename it to insertCSV.
2. Change the VBA code for insertCSV so that the end of the URL reads as follows:
   `&_debug=0&foo=/temp.csv`
3. Supply the appropriate name of the Application Dispatcher program and the Web server.

Follow steps 1 and 3 above to create a macro to include SYLK output.

(Advanced VBA programmers should make use of more sophisticated macros, including the use of User Forms, instead of continually copying and modifying the code.)

```
Sub insertHTML()
  Dim currentCell As Range
  Dim tempWorkbook As Workbook
  Dim tempRange As Range

  Set currentCell =
Application.Workbooks(ActiveWorkbook.Name).Ac
tiveSheet.Application.ActiveCell

  Set tempWorkbook =
Workbooks.Open("http://web-
server/scripts/broker?_program=program-
name&_debug=0")
  Set tempRange =
tempWorkbook.ActiveSheet.UsedRange

  tempRange.Copy currentCell

  tempWorkbook.Close False
End Sub
```

REFERENCES

"Application Dispatcher, Release 8.2." SAS Institute Inc. Available http://www.sas.com/rnd/web/intrnet/dispatch/

"Creating Customized Tagsets to Use With ODS and XML LIBNAME Engine." SAS Institute Inc. Available http://www.sas.com/rnd/base/topics/odstagsets/

"DS2CSV Macro." SAS Institute Inc. Available http://www.sas.com/rnd/web/intrnet/ds2csv/ds2csv.html

"FAQ (Frequently Asked Questions) about spreadsheets," the FAQ of the comp.apps.spreadsheets newsgroup. Available http://www.faqs.org/faqs/spreadsheets/faq/

"HTML Formatting Tools." SAS Institute Inc. Available http://www.sas.com/rnd/web/intrnet/format/

Mehler, G. (1998), "Integrating Windows® Clients and the SAS® System into the Enterprise," *Proceedings of the Twenty-third Annual SAS Users Group International Conference*, 23, CD-ROM. Paper 266.

"Microsoft Excel for Windows: Visual Basic for Applications 101." Microsoft Corporation. Available http://support.microsoft.com/support/excel/content/vba101/default.asp

Roper, C. (2000), "Intelligently Launching Microsoft® Excel from SAS®, using SCL Functions Ported to Base SAS," *Proceedings of the Twenty-fifth Annual SAS Users Group International Conference*, 25, CD-ROM. Paper 97.

"Stored Processes." SAS Institute Inc. Available http://www.sas.com/rnd/itech/doc/app/spindex.html

"SYLK (Symbolic Link) Format." Microsoft Corporation. Available http://www.microsoft.com/Office/techinfo/productdoc/2000/en/excel/xlrefSYLKSymbolicLink.htm

"sylksum.doc." Microsoft Corporation. Available http://www.worldgate.ca/~rschulz/misc/sylksum.doc

"Technical Support Document 325: The SAS System and DDE." SAS Institute Inc. Available http://ftp.sas.com/techsup/download/technote/ts325.pdf

"Technical Support Document 626: Configuring the SAS V8 ODBC Driver." SAS Institute Inc. Available http://ftp.sas.com/techsup/download/technote/ts626.html

"VBA Overview." Microsoft Corporation. Available http://msdn.microsoft.com/vba/prodinfo/backgrounder.asp

Wotsit Web Site, Available http://www.wotsit.org/

Vyverman, K. (2001), "Using Dynamic Data Exchange to Export Your SAS® Data to MS Excel: Against All ODS, Part I," *Proceedings of the Twenty-sixth Annual SAS Users Group International Conference*, 26, CD-ROM. Paper 11.

"XL97: How to Create Web Query (.iqy) Files." Microsoft Corporation, *Microsoft Product Support Services*, Nov 3, 1998. Available http://support.microsoft.com/default.aspx?scid=kb;EN-US;q157482

ACKNOWLEDGMENTS

The author would like to thank Bryan Wolfe, Greg Granger and Chris Barrett of SAS Institute Inc. for their valuable input 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 USA

Email: Vincent.DelGobbo@SAS.com
Phone: (919) 677-8000
Fax: (919) 677-4444