

# Some Techniques for Integrating SAS® Output with Microsoft Excel Using Base SAS®

Vince DelGobbo  
SAS



**SAS Global Forum 2013**  
**April 28 – May 1, 2013**  
**Moscone Center West**  
**San Francisco, CA**

A special "thank you" to Nancy Brucken and Maribeth Johnson for inviting me to present this topic, and to Chris Barrett of SAS Institute Inc. for his valuable contributions to the accompanying paper.

## Goals

- Integrate SAS output w/ Excel
- Compare 3 different techniques
- Give you something you can use TODAY



## Agenda

- Review sample data
- Technique #1: EXPORT procedure
- ODS basics
- Technique #2: ODS CSVALL
- Technique #3: ODS ExcelXP
- Using SAS server technology



We discuss three techniques for using Base SAS® to get data into Excel.

Technique #1 produces a comma-separated value (CSV) file that contains only data and no formatting of any kind.

Technique #2 produces a CSV file with data and some grouping, but still no formatting and color.

Technique #3 produces the desired workbook, complete with formats, colors, and correct groupings.

We conclude with a discussion of how to drive dynamic data to Excel using SAS server technology.

## Software Requirements

- Base SAS, *any* operating system
- SAS 9.1.3 Service Pack 4 or above
- Microsoft Excel 2002 or above (a.k.a. Excel XP)



# SAS Output Viewed with Excel

The screenshot shows an Excel spreadsheet with the following data:

<i>Subject ID: 1</i>					
Vital Signs: Time	Respiratory Rate (min)	Heart Rate (bpm)	SBP (mmHg)	DBP (mmHg)	
-2.00	16	105	135	84	
-1.00	9	59	128	73	
0.02	9	64	126	83	
0.08	9	64	133	91	
0.17	12	72	136	90	
0.25	14	60	130	82	
1.00	16	73	135	73	
2.00	12	72	124	65	
3.00	12	81	132	77	
4.00	21	80	120	70	
5.00	19	80	120	70	
8.00	16	96	150	90	
24.00	20	80	120	70	

<i>Subject ID: 2</i>					
Vital Signs: Time	Respiratory Rate (min)	Heart Rate (bpm)	SBP (mmHg)	DBP (mmHg)	
-2.00	15	72	94	56	
-1.00	12	55	135	78	
0.02	12	50	127	77	
0.08	14	58	104	73	
0.17	13	54	123	79	
0.25	18	55	122	75	
1.00	24	60	116	78	

5

The worksheets contain fictional vital signs data for clinical trials "ABC 123" and "XYZ 987". The worksheets contain one table for each patient enrolled in the trial, resulting in multiple tables in each worksheet.

An Excel format, not a SAS format, is used to control the appearance of the Vital Signs: Time column, and a TITLE statement is used to display the patient's ID number.

## Sample Data – Column Definitions

Sorted by

Column Name	Type	Length	Format	Informat	Label
PROTOCOL	Text	7			Protocol Identifier
PATIENT	Number	8			Patient Identifier
VITTIME	Number	8	6.2		Vital Signs: Time
VITRRATE	Number	8	2.		Respiratory Rate (/min)
VITHRATE	Number	8	3.		Heart Rate (bpm)
VITSYSBP	Number	8	3.		SBP (mmHg)
VITDIABP	Number	8	3.		DBP (mmHg)

6

The code uses the PHCVital SAS table, available in the download package of this workshop.

The PROTOCOL column indicates whether the patient was enrolled in clinical trial "ABC 123" or "XYZ 987".

The data values in VITTIME column are displayed with 2 decimal places.

The table is sorted by the PROTOCOL column, then PATIENT, and then VITTIME.

# Sample Data – Typical Values

	Protocol Identifier	Patient Identifier	Respiratory Rate (/min)	Vital Signs: Time	Heart Rate (bpm)	SBP (mmHg)	DBP (mmHg)
1	ABC 123	1	16	-2.00	105	135	84
2	ABC 123	1	9	-1.00	59	128	73
3	ABC 123	1	9	0.02	64	126	83
4	ABC 123	1	9	0.08	64	133	91
5	ABC 123	1	12	0.17	72	136	90
6	ABC 123	1	14	0.25	60	130	82
7	ABC 123	1	16	1.00	73	135	73
8	ABC 123	1	12	2.00	72	124	65
9	ABC 123	1	12	3.00	81	132	77
10	ABC 123	1	21	4.00	80	120	70
11	ABC 123	1	19	5.00	80	120	70
12	ABC 123	1	16	8.00	96	150	90
13	ABC 123	1	20	24.00	80	120	70
14	ABC 123	2	15	-2.00	72	94	56
15	ABC 123	2	12	-1.00	55	135	78
16	ABC 123	2	12	0.02	50	127	77
17	ABC 123	2	14	0.08	58	104	73
18	ABC 123	2	13	0.17	54	123	79
19	ABC 123	2	18	0.25	55	122	75
20	ABC 123	2	24	1.00	60	116	78
3240	XYZ 987	110	10	0.02	66	134	98
3241	XYZ 987	110	10	0.08	66	154	98
3242	XYZ 987	110	8	0.17	64	148	90
3243	XYZ 987	110	12	0.25	66	148	91
3244	XYZ 987	110	12	1.00	56	148	79
3245	XYZ 987	110	18	2.00	58	115	59
3246	XYZ 987	110	16	3.00	67	113	58
3247	XYZ 987	110	16	4.00	70	123	56
3248	XYZ 987	110	20	5.00	72	110	70
3249	XYZ 987	110	20	8.00	88	125	80
3250	XYZ 987	110	22	24.00	76	110	70

7

**SBP** = Systolic Blood Pressure

**DBP** = Diastolic Blood Pressure

# Technique #1: The EXPORT Procedure



## Creating CSV Files w/ PROC EXPORT

- Creates only delimited files w/ Base SAS®9
- CSV = Comma Separated Values
- SAS/ACCESS® required for native Excel files
- Not available on z/OS
- Procedure output not supported
- Good for exchanging data; limited/no formatting
- Multi-sheet workbooks require manual steps (Excel Text Import Wizard)



The first technique for getting SAS data into Excel uses the EXPORT procedure to read the data from a SAS table and write it to two CSV files.

The procedure is available only for Windows and UNIX operating systems, and is limited to creating delimited files if only Base SAS® software is licensed. The SAS/ACCESS® Interface to PC Files software must be licensed to create native Microsoft Excel workbooks.

"Export Procedure". *Base SAS® 9.3 Procedures Guide*.

<http://tinyurl.com/cltnmza>

*SAS/ACCESS® 9.3 Interface to PC Files: Reference*.

<http://tinyurl.com/ccpe79o>

## Creating CSV Files w/ PROC EXPORT

```
title 'Subject ID: #BYVAL(patient)';

proc export data=sample.PHCVital (where=(protocol='ABC 123'))
  outfile='VitalSigns-ExportABC.csv'
  dbms=csv label replace;
run; quit;

proc export data=sample.PHCVital (where=(protocol='XYZ 987'))
  outfile='VitalSigns-ExportXYZ.csv'
  dbms=csv label replace;
run; quit;
```



The EXPORT procedure is run twice using the WHERE data set option because the WHERE and BY statements are not supported in that procedure.

The DBMS and LABEL options specify that a CSV file is created that contains column labels when available, instead of column names.

If the output file exists, it will be overwritten because the REPLACE option is specified.

# Creating CSV Files w/ PROC EXPORT

```
VitalSigns-ExportABC.csv - Notepad
File Edit Format View Help
["Protocol Identifier","Patient Identifier","vital signs: Time","Respirat
ABC 123,1,-2.00,16,105,135,84
ABC 123,1,-1.00,9,59,128,73
ABC 123,1,0.02,9,64,126,83
ABC 123,1,0.08,9,64,133,91
ABC 123,1,0.17,12,72,136,90
ABC 123,1,0.25,11
ABC 123,1,1.00,1
ABC 123,1,2.00,1
ABC 123,1,3.00,1
ABC 123,1,4.00,2
ABC 123,1,5.00,1
ABC 123,1,8.00,1
ABC 123,1,24.00,
ABC 123,2,-2.00,
ABC 123,2,-1.00,
ABC 123,2,0.02,1
ABC 123,2,0.08,1
ABC 123,2,0.17,1
ABC 123,2,0.25,1
ABC 123,2,1.00,2
ABC 123,2,2.00,2
VitalSigns-ExportXYZ.csv - Notepad
File Edit Format View Help
["Protocol Identifier","Patient Identifier","vital signs: Time","Respirat
XYZ 987,1,-2.00,18,85,183,112
XYZ 987,1,-1.00,10,94,110,63
XYZ 987,1,0.02,10,61,110,63
XYZ 987,1,0.08,10,59,126,84
XYZ 987,1,0.17,16,83,163,124
XYZ 987,1,0.25,24,75,137,95
XYZ 987,1,1.00,18,74,160,95
XYZ 987,1,2.00,12,78,140,90
XYZ 987,1,3.00,20,76,165,100
XYZ 987,1,4.00,20,88,170,120
XYZ 987,1,5.00,20,76,150,110
XYZ 987,1,8.00,20,80,160,100
XYZ 987,1,24.00,,,
XYZ 987,2,-2.00,18,100,125,75
XYZ 987,2,-1.00,9,75,140,80
XYZ 987,2,0.02,9,76,135,85
XYZ 987,2,0.08,9,81,136,87
XYZ 987,2,0.17,9,92,139,87
XYZ 987,2,0.25,9,98,140,80
XYZ 987,2,1.00,17,98,127,79
XYZ 987,2,2.00,16,93,136,88
```

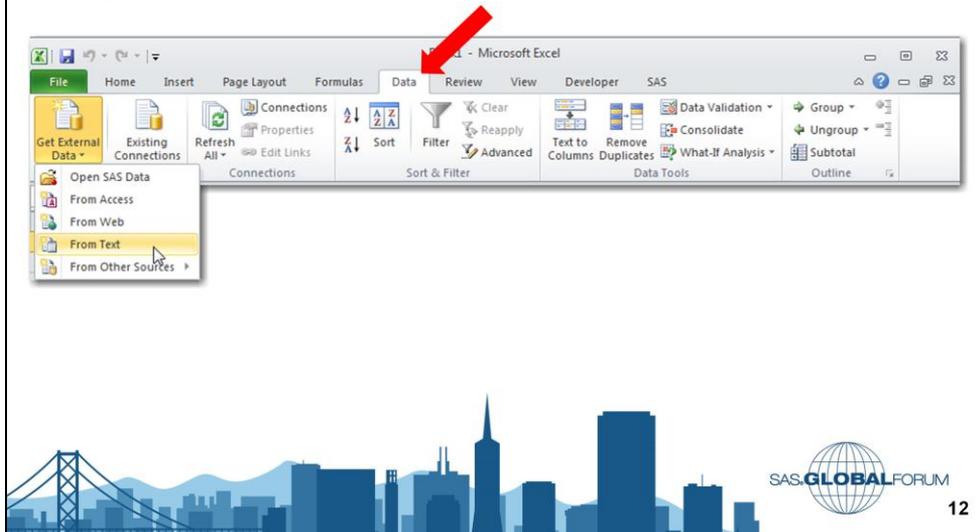
11

CSV is a plain-text format consisting of data fields separated by commas.

Because the exported files only contain the data values, no Excel features such as formats, formulas, and colors, can be supported.

# Creating Multi-Sheet Workbooks from CSV

1. Start Excel
2. **Data** → **Get External Data** → **From Text**



Because Excel cannot automatically create a multi-sheet workbook from delimited files, we must manually import the data from each CSV file into its own worksheet.

# Creating Multi-Sheet Workbooks from CSV

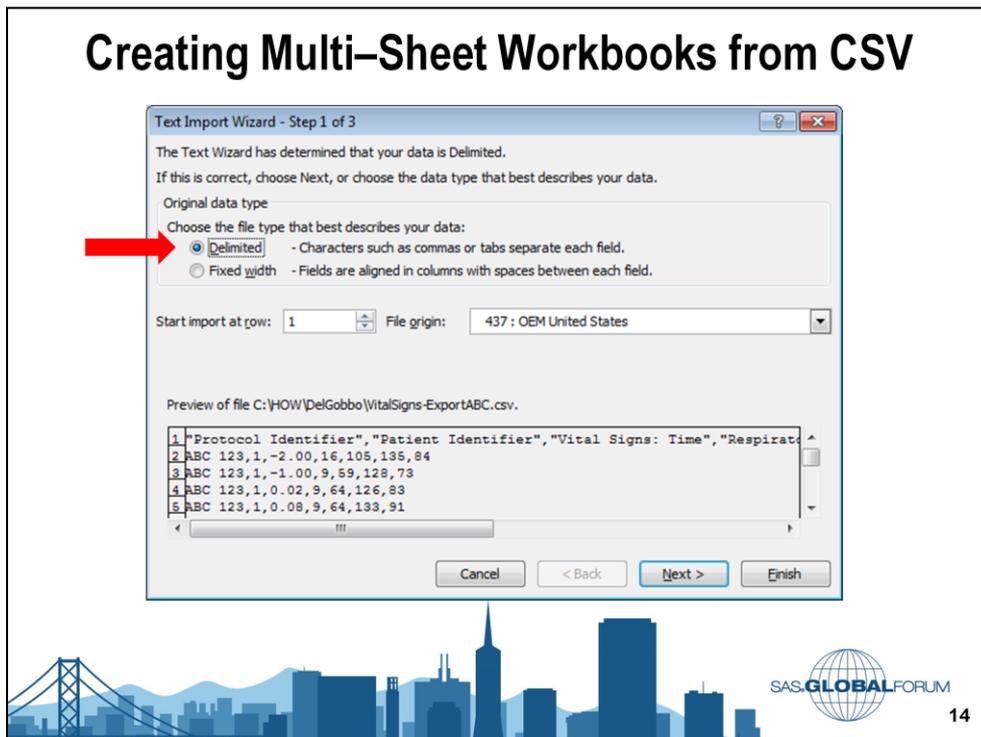
## 3. Choose VitalSigns-ExportABC.csv



## 4. Continue with the Text Import Wizard

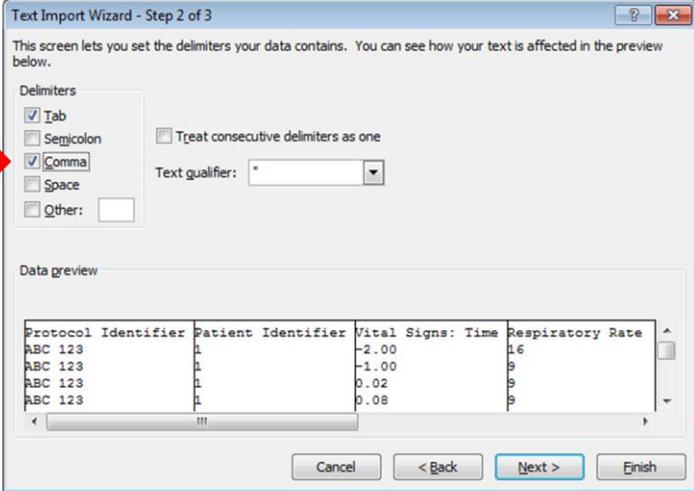


# Creating Multi-Sheet Workbooks from CSV



In Step 1 of the wizard, select **Delimited** in the **Original data type** group, and then click **Next >**.

## Creating Multi-Sheet Workbooks from CSV



This screen lets you set the delimiters your data contains. You can see how your text is affected in the preview below.

**Delimiters**

- Tab
- Semicolon
- Comma
- Space
- Other:

Treat consecutive delimiters as one

Text qualifier:

**Data preview**

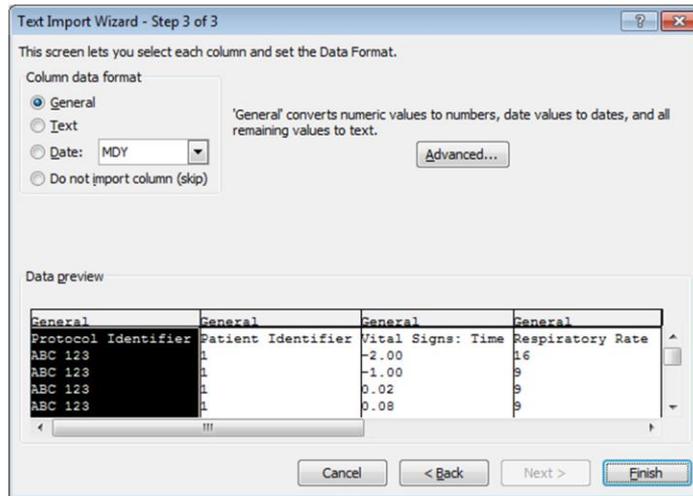
Protocol Identifier	Patient Identifier	Vital Signs: Time	Respiratory Rate
ABC 123	L	-2.00	16
ABC 123	L	-1.00	9
ABC 123	L	0.02	9
ABC 123	L	0.08	9

Buttons: Cancel, < Back, Next >, Finish



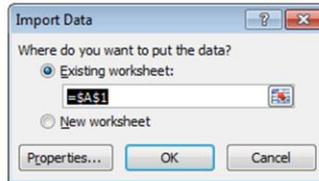
Select **Comma** in the **Delimiters** group and then click **Next >**.

# Creating Multi-Sheet Workbooks from CSV



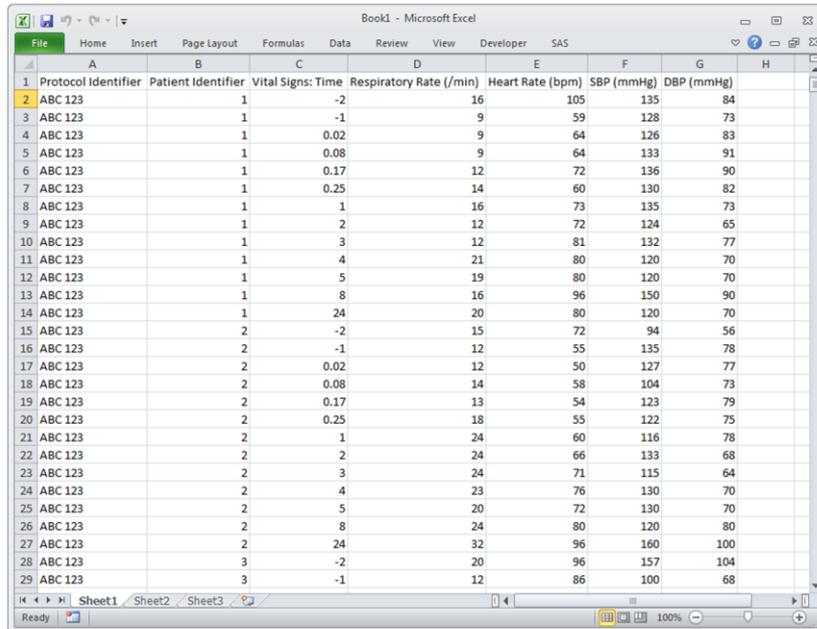
Accept the default values and then click **Finish**.

# Creating Multi-Sheet Workbooks from CSV



Choose cell **A1** for the location to place the data, and then click **OK** to complete the import process.

## Creating Multi-Sheet Workbooks from CSV



The screenshot shows a Microsoft Excel window titled "Book1 - Microsoft Excel". The ribbon includes File, Home, Insert, Page Layout, Formulas, Data, Review, View, Developer, and SAS. The spreadsheet has three sheets: Sheet1, Sheet2, and Sheet3. Sheet1 is active and contains the following data:

	A	B	C	D	E	F	G	H
1	Protocol Identifier	Patient Identifier	Vital Signs: Time	Respiratory Rate (/min)	Heart Rate (bpm)	SBP (mmHg)	DBP (mmHg)	
2	ABC 123	1	-2	16	105	135	84	
3	ABC 123	1	-1	9	59	128	73	
4	ABC 123	1	0.02	9	64	126	83	
5	ABC 123	1	0.08	9	64	133	91	
6	ABC 123	1	0.17	12	72	136	90	
7	ABC 123	1	0.25	14	60	130	82	
8	ABC 123	1	1	16	73	135	73	
9	ABC 123	1	2	12	72	124	65	
10	ABC 123	1	3	12	81	132	77	
11	ABC 123	1	4	21	80	120	70	
12	ABC 123	1	5	19	80	120	70	
13	ABC 123	1	8	16	96	150	90	
14	ABC 123	1	24	20	80	120	70	
15	ABC 123	2	-2	15	72	94	56	
16	ABC 123	2	-1	12	55	135	78	
17	ABC 123	2	0.02	12	50	127	77	
18	ABC 123	2	0.08	14	58	104	73	
19	ABC 123	2	0.17	13	54	123	79	
20	ABC 123	2	0.25	18	55	122	75	
21	ABC 123	2	1	24	60	116	78	
22	ABC 123	2	2	24	66	133	68	
23	ABC 123	2	3	24	71	115	64	
24	ABC 123	2	4	23	76	130	70	
25	ABC 123	2	5	20	72	130	70	
26	ABC 123	2	8	24	80	120	80	
27	ABC 123	2	24	32	96	160	100	
28	ABC 123	3	-2	20	96	157	104	
29	ABC 123	3	-1	12	86	100	68	

18

Repeat the import process for the "VitalSigns-ExportXYZ.csv" file.

The EXPORT procedure cannot be used to create the workbook that we want. Because CSV files are limited to data values, PROC EXPORT is limited to exporting only data values to Excel.

If you're only interested in moving data values from SAS to Excel, then PROC EXPORT is a quick and easy way to accomplish the task.

To save a copy of the file in Excel binary (xls) format using Excel 2002, 2003, or 2010, select **File** → **Save As** and then, from the **Save as type** drop-down list, select **Microsoft Excel Workbook (\*.xls)**. If you're using Excel 2007, click the **Microsoft Office Button**, and then select **Save As** → **Excel 97-2003 Workbook**.

If you're using Excel 2007 or 2010 and want to save the document in the Microsoft Office Open XML format, choose **Excel Workbook (\*.xlsx)** from the **Save as type** drop-down list.

## Run Setup.sas

1. Start SAS
2. **File > Open Program**
3. Navigate to **C:\HOWDeIGobbo\**
4. Select **Setup.sas** and click **Open**
5. Review code and submit



The SAMPDIR global macro variable specifies the directory containing our sample code and data, as well as the SAS output.

The program assigns a SAS library (SAMPLE) for the input SAS data set.

## Ex. 1 – PROC EXPORT

1. Go to SAS
2. **File > Open Program > Exercise1.sas**
3. Review code and submit
4. Start Excel
5. Use **Data > Get External Data ...** to import "VitalSigns-ExportABC.csv"
6. Optional: Repeat import for "VitalSigns-ExportXYZ.csv"
7. Close document; keep Excel open



# ODS Basics



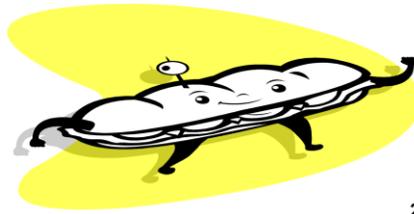
The Output Delivery System (ODS) is capable of creating files that can be opened by Excel, and is used in the next two techniques.

## ODS Basics

- Part of Base SAS
- Easily generate multiple output types (HTML, RTF, PDF, XML, etc.)
- A "destination" creates the actual output
- A "style" controls the appearance
- Usage:

HTML or RTF or PDF ...

```
ods DestName style=StyleName file=... ;  
  * Your SAS procedure code here ;  
ods DestName close ;
```



22

ODS is the part of Base SAS that enables you to generate different types of output from your procedure code. An ODS *destination* controls the type of output that is generated (HTML, RTF, PDF, etc.). An ODS *style* controls the appearance of the output (colors, fonts, border lines, etc.).

Both a destination and a style are needed to generate output. If you do not specify a style, a default style is used.

# Technique #2: ODS CSVALL Destination



## Creating CSV Files w/ ODS CSVALL

- Creates only delimited (CSV) files
- Multi-sheet workbooks require manual steps (Excel Text Import Wizard)
- Good for exchanging data; limited/no formatting
- **Layout different than PROC EXPORT**
- **Available on all operating systems**
- **Non-graphic procedure output supported**



Like PROC EXPORT, we can use the ODS CSVALL destination to create CSV files containing our SAS data, and manually import the files into Excel.

Unlike PROC EXPORT, the CSVALL destination is supported on all operating systems, and you can use it to export non-graphic procedure output to Excel.

## Creating CSV Files w/ ODS CSVALL

```
options nobyline;  
title 'Subject ID: #BYVAL(patient)';  
ods csvall file='VitalSigns-ODSABC.csv';  
proc print data=sample.PHCVital noobs label;  
  by protocol patient;  
  var vittime vitrrate vithrate vitsysbp vitdiabp;  
  pageby patient;  
  where (protocol eq 'ABC 123');  
run; quit;  
ods csvall close;  
  
ods csvall file='VitalSigns-ODSXYZ.csv';  
proc print data=...;  
  by ...; var ...; pageby ...;  
  where (protocol eq 'XYZ 987');  
run; quit;  
ods csvall close;  
options byline;
```

25

The ODS CSVALL destination and the PRINT procedure export the PHCVital table to CSV files "VitalSigns-ODSABC.csv" and "VitalSigns-ODSXYZ.csv".

We use #BYVAL to display the current value of the PATIENT BY-variable in the document title, and the NOBYLINE system option prevents showing the same information in the document.

Because CSV files contain only data values, the ODS STYLE option is not used.

"ODS CSVALL Statement". SAS® 9.3 *Output Delivery System: User's Guide*.  
<http://tinyurl.com/b86s846>

"TITLE Statement". SAS® 9.3 *Statements: Reference*.  
<http://tinyurl.com/acp9kah>

# Excel Text Import Wizard

Subject ID: 1				
Vital Signs: Time	Respiratory Rate (/min)	Heart Rate (bpm)	SBP (mmHg)	DBP (mmHg)
-2	16	105	135	84
-1	9	59	128	73
0.02	9	64	126	83
0.08	9	64	133	91
0.17	12	72	136	90
0.25	14	60	130	82
1	16	73	135	73
2	12	72	124	65
3	12	81	132	77
4	21	80	120	70
5	19	80	120	70
8	16	96	150	90
24	20	80	120	70
Subject ID: 2				
Vital Signs: Time	Respiratory Rate (/min)	Heart Rate (bpm)	SBP (mmHg)	DBP (mmHg)
-2	15	72	94	56
-1	12	55	135	78
0.02	12	50	127	77
0.08	14	58	104	73
0.17	13	54	123	79
0.25	18	55	122	75
1	24	60	116	78
2	24	66	133	68
3	24	71	115	64

26

The result of importing the CSV files using the Excel Text Import Wizard is closer to what's desired, but lacks colors and formatting.

## Ex. 2 – ODS CSVALL

1. Go to SAS
2. **File > Open Program > Exercise2.sas**
3. Follow TO DO instructions & submit code
4. Start Windows Explorer
5. Navigate to **C:\HOWDeIGobbo\**
6. Double click **VitalSigns-ODSABC.csv**
7. Note insufficient column widths
8. Close document; keep Excel open



TO DO:

Line 11: Add the SAS System Option to suppress BY-line text, submit the code, and then view a CSV file in Excel by double-clicking the file.

Note that when opening the CSV file with Excel by double-clicking, the column widths are different that when importing the file using the Excel Text Import Wizard.

# Technique #3: ODS ExcelXP Tagset



## Creating an XML File w/ ODS ExcelXP

- Available on all operating systems
- Non-graphic procedure output supported
- Procedure output layout retained
- Creates Microsoft-supported XML file
- Rich formatting and Excel features supported
- Multi-sheet workbook created automatically



While the two previous techniques produced CSV files, those files don't support the output and styling options that we want. To get those we need to use the ExcelXP tagset, a type of ODS destination that creates ***XML output*** that can be opened with Excel.

Unlike the delimited (CSV) files generated by the prior techniques, using the ODS ExcelXP tagset allows you to take advantage of rich Excel features, such as multiple worksheets, formats, and colors. All formatting and layout are performed by SAS; there is no need to hand-edit the Excel workbook. You simply use Excel to open the file created by ODS.

Because the ExcelXP ODS tagset creates files that conform to the Microsoft XML Spreadsheet Specification, you can create multi-sheet Excel workbooks containing the output from almost any SAS procedure. The exception is that the Microsoft XML Spreadsheet Specification does not support images, so the output from SAS/GRAPH® software procedures cannot be used.

"XML Spreadsheet Reference".

<http://tinyurl.com/3p6sked>

## Creating an XML File w/ ODS ExcelXP

```
title 'Subject ID: #BYVAL(patient)';

ods tagsets.ExcelXP file='VitalSigns-ExcelXP.xml'
  style=Printer;

proc print data=sample.PHCVital noobs label;
  by protocol patient;
  var vittime vitrrate vittrate vitsysbp vitdiabp;
  pageby patient;
  * No WHERE statement needed;
run; quit;

ods tagsets.ExcelXP close;
```



The ExcelXP tagset generates the output, and the Printer style controls the appearance of the output.

Unlike the previous techniques, the PRINT procedure is run only once to create a single file ("VitalSigns-ExcelXP.xml") with data from both trials.

You should use the .xml extension instead of .xls or .xlsx, because Excel 2007 and 2010 display a warning if the .xml extension is not used.

"When you open a file in Excel 2007, you receive a warning that the file format differs from the format that the file name extension specifies".

<http://support.microsoft.com/kb/948615>

## Opening the XML File with Excel

- Start Excel
- **File > Open**
- Navigate to ...\**VitalSigns-ExcelXP.xml** and click **Open**

~ OR ~

- Navigate to output directory and double-click **VitalSigns-ExcelXP.xml**



Excel reads and converts the XML file to the Excel format. After the conversion, you can perform any Excel function on the data.

# Opening the XML File with Excel

Protocol Identifier=ABC 123 Patient Identifier=1				
Vital Signs: Time	Respiratory Rate (/min)	Heart Rate (bpm)	SBP (mmHg)	DBP (mmHg)
-2	16	105	135	84
-1	9	59	128	73
0.02	9	64	126	83
0.08	9	64	133	91
0.17	12	72	136	90
0.25	14	60	130	82
1	16	73	135	73
2	12	72	124	65
3	12	81	132	77
4	21	80	120	70
5	5	80	120	70
8	16	96	150	90
24	20	80	120	70

32

The "VitalSigns-ExcelXP.xml" file resembles our desired layout, but has the following differences:

1. Each trial-patient combination appears in its own worksheet, resulting in 250 worksheets. Patient information should be grouped by trial (PROTOCOL variable), resulting in only 2 worksheets.
2. Unattractive, default worksheet names are used.
3. Standard BY-group text (**Protocol Identifier=ABC 123 Patient Identifier=1**) precedes the table, instead of the TITLE statement text.
4. Not all of the data values in column **A** are displayed with two decimal places.

We can now change the SAS code to correct these problems.

## Ex. 3 – Creating the Initial Workbook

1. Go to SAS
2. **File > Open Program > Exercise3.sas**
3. Review code and submit
4. Go to Windows Explorer
5. Navigate to **C:\HOWDeIGobbo\**
6. Double click **VitalSigns-ExcelXP.xml**
7. Close document; keep Excel open



## ExcelXP Tagset Options

- Syntax: `options (option-name='option-value')`
- Example: Hard-coding worksheet name

```
options (sheet_name='worksheet-name');
```

- Option values must be quoted
- Can have multiple ODS statements
- Options remain in effect until changed



The ExcelXP tagset supports many options that control both the appearance and functionality of the Excel workbook. Most of these tagset options are simply tied directly to existing Excel options or features. Tagset options are specified in an ODS statement using the `OPTIONS` keyword, as shown above.

ODS automatically generated unique worksheet names, but we can use the `SHEET_NAME` option to explicitly specify a worksheet name.

**IMPORTANT NOTE:** Tagset options remain in effect until they are set to another value, or until the ODS destination is closed.

## Grouping & Naming Worksheets by Trial

- `SHEET_INTERVAL` – When to create worksheets
- `SHEET_LABEL` – Prefix for worksheet name

Trial ABC 123

Trial XYZ 987

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



By default, the ExcelXP tagset creates a new worksheet when a SAS procedure creates new tabular output. Because a new table is created for each of the 250 combinations of PROTOCOL and PATIENT, the initial workbook contains 250 worksheets, each with a unique name as required by Excel.

We can fix this by setting the `SHEET_LABEL` option to specify the prefix to use for the worksheet names, and setting the value of the `SHEET_INTERVAL` option to `bygroup`. This combination generates 2 worksheets, one for each distinct value of the PROTOCOL BY-variable.

## Grouping & Naming Worksheets by Trial

```
title ...;
ods tagsets.ExcelXP file='VitalSigns-ExcelXP.xml'
  style=Printer;

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

proc print data=...;
  ...;
run; quit;

ods tagsets.ExcelXP close;
```



The additional ODS statement to handle the tagset options.

**NOTE:** When specifying *additional* ODS statements, do not specify the FILE, STYLE, or any other keyword or option that is supported by ODS. Those options should be specified only on the initial ODS statement.

# Grouping & Naming Worksheets by Trial

The screenshot shows an Excel spreadsheet with two worksheets. The first worksheet, 'Trial ABC 123', contains a table with the following data:

Protocol Identifier=ABC 123 Patient Identifier=1					
Vital Signs: Time	Respiratory Rate (/min)	Heart Rate (bpm)	SBP (mmHg)	DBP (mmHg)	
-2	16	105	135	84	
-1	9	59	128	73	
0.02	9	64	126	83	
0.08	9	64	133	91	
0.17	12	72	136	90	
0.25	14	60	130	82	
1	16	73	135	73	
2	12	72	124	65	
3	12	81	132	77	
4	21	80	120	70	
5	19	80	120	70	
8	16	96	150	90	
24	20	80	120	70	

The second worksheet, 'Trial XYZ 987', contains a table with the following data:

Protocol Identifier=ABC 123 Patient Identifier=2					
Vital Signs: Time	Respiratory Rate (/min)	Heart Rate (bpm)	SBP (mmHg)	DBP (mmHg)	
-2	15	72	94	56	
-1	12	55	135	78	
0.02	12	50	127	77	
0.08	14	58	104	73	
0.17	13	54	123	79	
0.25	18	55	122	75	
1	24	60	116	78	
2	24	66	133	68	
3	24	71	115	64	

Red arrows in the image point to the table headers in both worksheets and to the worksheet tabs at the bottom of the Excel window.

We now have 2 worksheets with multiple tables in each worksheet.

## Ex. 4 – Group & Name Worksheets by Trial

1. Go to SAS
2. **File > Open Program > Exercise4.sas**
3. Follow TO DO instructions & submit code
4. Go to Excel
5. **File > Recent**
6. Select **VitalSigns-ExcelXP.xml**
7. Note worksheet names and groupings
8. Close document; keep Excel open



TO DO:

Lines 18 and 19: Supply values for the ExcelXP tagset options to control the worksheet names, submit the code, and then view the XML file in Excel from the Recent Workbooks panel.

## Replacing BY-Line Text w/ TITLE Text

- `SUPPRESS_BYLINES` – Omit BY line text
- `EMBEDDED_TITLES` – Include TITLE statement text

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



BY-line text appears in the worksheets because the PRINT procedure is executed with a BY statement. Some of the text is redundant because the value of **Protocol Identifier** is displayed in the worksheet name.

To omit the BY-line text, specify the `SUPPRESS_BYLINES` tagset option. Do not attempt to use the `NOBYLINE` system option, like we did with the `CSVALL` destination. Doing so would disable BY-group processing in the ExcelXP tagset when `SHEET_INTERVAL` is set to `bygroup`.

By default, SAS titles and footnotes appear as Excel print headers and print footers, respectively, and are displayed when the Excel document is printed. To include title text on-screen, in the worksheet body, use the `EMBEDDED_TITLES` option.

## Replacing BY-Line Text w/ TITLE Text

```
ods tagsets.ExcelXP file=... style=...;

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

proc print data=...; run; quit;

ods tagsets.ExcelXP close;
```



The new options are shown in context.

# Replacing BY-Line Text w/ TITLE Text

The screenshot shows an Excel spreadsheet with the following data:

	A	B	C	D	E
1			Subject ID: 1		
2					
3		Vital Signs: Time	Respiratory Rate (min)	Heart Rate (bpm)	SBP (mmHg) DBP (mmHg)
4		-2	16	105	135 84
5		-1	9	59	128 73
6		0.02	9	64	126 83
7		0.08	9	64	133 91



## Ex. 5 – Replacing BY–Line Text w/ TITLE Text

1. Go to SAS
2. **File > Open Program > Exercise5.sas**
3. Follow TO DO instructions & submit code
4. Go to Excel
5. **File > Recent**
6. Select **VitalSigns-ExcelXP.xml**
7. Note TITLE text in worksheet body
8. Close document; keep Excel open



TO DO:

Lines 21 and 22: Add the 2 ExcelXP tagset options to suppress BY-line text and embed title text into the document, submit the code, and then view the XML file in Excel from the Recent Workbooks panel.

## ODS Style Overrides

- Supported by PRINT, REPORT, and TABULATE
- Change any ODS style attribute via STYLE=
- Example:

```
var vittime /  Location  
  style(column) = [font_style = italic  Attributes  
                  background = orange]
```

- Refer to the ODS documentation for a list of supported attributes
- Refer to PRINT, REPORT, and TABULATE doc for sample usage



Style overrides are supported by the PRINT, REPORT, and TABULATE procedures, and can be specified in several ways, with the most common shown here.

This example overrides the font style and the cell background color attributes for the data values in the VITTIME column.

"Style Attributes Tables". *SAS® 9.3 Output Delivery System User: User's Guide*.  
<http://tinyurl.com/a6vo5yq>

"PRINT Procedure: Examples". *Base SAS® 9.3 Procedures Guide, Second Edition*.  
<http://tinyurl.com/ct3j7t4>

"REPORT Procedure: Examples". *Base SAS® 9.3 Procedures Guide, Second Edition*.  
<http://tinyurl.com/axnqrd4>

"TABULATE Procedure: Examples". *Base SAS® 9.3 Procedures Guide, Second Edition*.  
<http://tinyurl.com/bdqgl4c>

# ODS Style Overrides – Example Output

The screenshot shows an Excel spreadsheet with two data tables. The first table is for 'Subject ID: 1' and the second is for 'Subject ID: 2'. Each table has five columns: 'Vital Signs: Time', 'Respiratory Rate (/min)', 'Heart Rate (bpm)', 'SBP (mmHg)', and 'DBP (mmHg)'. The 'Vital Signs: Time' column contains values from -2 to 24 in increments of 1. The other columns contain numerical values representing vital signs. The cells in the 'Vital Signs: Time' column are highlighted in orange.

Subject ID: 1				
Vital Signs: Time	Respiratory Rate (/min)	Heart Rate (bpm)	SBP (mmHg)	DBP (mmHg)
-2	16	105	135	84
-1	9	59	128	73
0.02	9	64	126	83
0.08	9	64	133	91
0.17	12	72	136	90
0.25	14	60	130	82
1	16	73	135	73
2	12	72	124	65
3	12	81	132	77
4	21	80	120	70
5	19	80	120	70
8	16	96	150	90
24	20	80	120	70

Subject ID: 2				
Vital Signs: Time	Respiratory Rate (/min)	Heart Rate (bpm)	SBP (mmHg)	DBP (mmHg)
-2	15	72	94	56
-1	12	55	135	78
0.02	12	50	127	77
0.08	14	58	104	73
0.17	13	54	123	79
0.25	18	55	122	75
1	24	60	116	78

## Location Values for PROC PRINT

table		
obsheader	header	header
obs	column	column
bylabel	total	total
grandtotal	grandtotal	grandtotal
n		

45

Style overrides are applied to specific parts of your SAS output. These specific parts of your SAS output are called *locations*.

The COLUMN location of PROC PRINT controls the appearance of data cells.

Two decimal places are not always used for the time values in our workbook. We use a style override to apply an Excel number format to these values.

"SAS® 9 Reporting Procedure Styles Tip Sheet".

<http://tinyurl.com/3cfqdv9>



## Excel Number Format Examples

Positive numbers: #,##0.00; [Red] (#,##0.00); 0.00; "sales "@  
 Negative numbers: [Red] (#,##0.00)  
 Zeroes: 0.00  
 Text: "sales "@

Raw Value	Formatted Value	Features
.5	0.50	Leading & Trailing 0's
5	5.00	
123	123.00	Trailing 0's
1234	1,234.00	Trailing 0's , Comma
-1234	(1,234.00)	Leading/Trailing 0's, Comma, Red ( )
0	0.00	Special Zero Handling
data	sales data	Special Text Handling

47

This table shows the results of applying the Excel format shown to several data values. Because negative values do not need to be considered differently than positive values, we only use the "Positive numbers" section of the format.

Excel 2003 Help. "Create or delete a custom number format".

<http://office.microsoft.com/assistance/hfws.aspx?AssetID=HP005199500>

Excel 2007 Help. "Create or delete a custom number format".

<http://office.microsoft.com/assistance/hfws.aspx?AssetID=HP001216503>

Excel 2010 Help. "Create a custom number format".

<http://office.microsoft.com/assistance/hfws.aspx?AssetID=HP010342372>

"Number Format Codes".

<http://office.microsoft.com/assistance/hfws.aspx?AssetID=HP005198679>

## Apply Excel Format to Time Values

```
proc print data=sample.PHCvital noobs label;  
  by    protocol patient;  
  var   vittime /  
        style(column)=[tagattr='format:0.00'];  
  var   vitrrate vithrate vitsysbp vitdiabp;  
  pageby patient;  
run; quit;
```



We use the Excel format `0.00` with the ODS `TAGATTR` attribute to specify an Excel format. Be sure to quote the entire attribute value and include the `format:` keyword.

To apply the style override to only the `VITTIME` column, we split the single `VAR` statement into two separate statements.

## Ex. 6 – Apply Excel Format to Time Values

1. Go to SAS
2. **File > Open Program > Exercise6.sas**
3. Follow TO DO instructions & submit code
4. Go to Excel
5. **File > Recent**
6. Select **VitalSigns-ExcelXP.xml**
7. Note formatted time values
8. Close document; keep Excel open



TO DO:

Line 25: Supply the value of the Excel format needed to display 2 decimal places, submit the code, and then view the XML file in Excel from the Recent Workbooks panel.

# Using SAS/IntrNet® and SAS Stored Processes



## SAS/IntrNet® and SAS Stored Processes

- SAS code is run from non-SAS client
- SAS is on any platform
- Client needs only a Web browser
- SAS output is delivered in real-time
- Web-enable the code we've been using



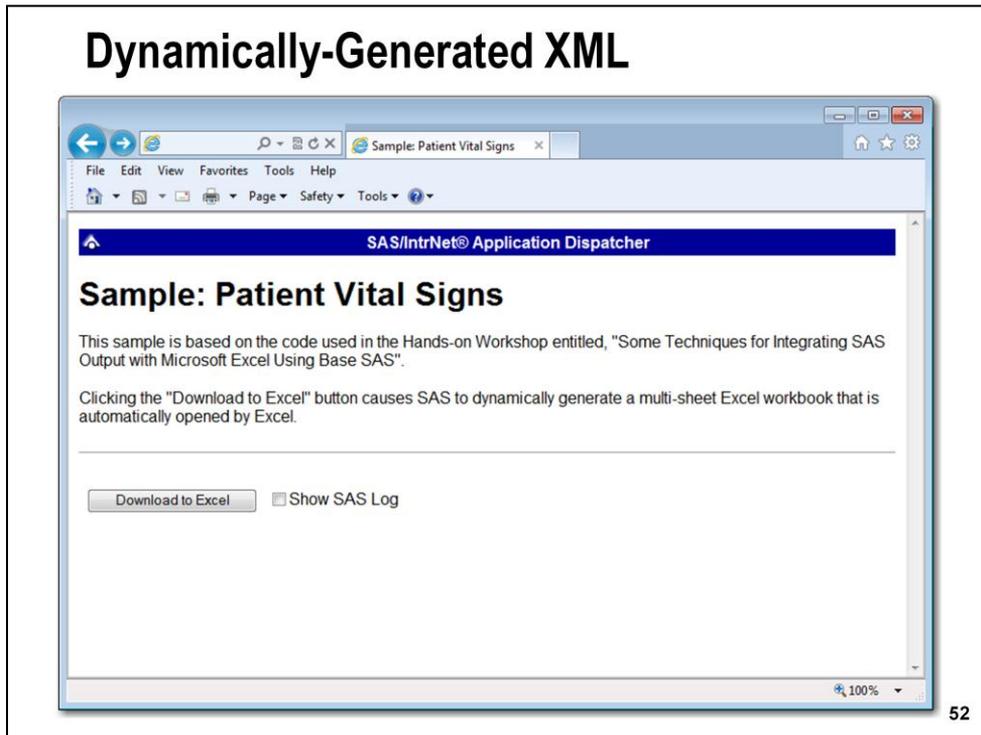
The purpose of the SAS/IntrNet® Application Dispatcher or SAS Stored Processes are to allow you to execute SAS programs from a client machine that does not have SAS installed. The client machine *may* have SAS installed, but that is not required.

A typical client-server model is followed. The SAS server can reside on any hardware platform (Windows, UNIX, z/OS, etc.) and is standing by, waiting to execute a SAS program. The most common client is a Web browser, again, running on any platform.

When the "OK" button of the Web page is clicked, input parameters, if any, are sent to the SAS server. Your SAS code executes, and the output is delivered in real-time to the Web browser.

The following slides illustrate this process, using a Web-enabled version of the SAS code we have been working on.

# Dynamically-Generated XML



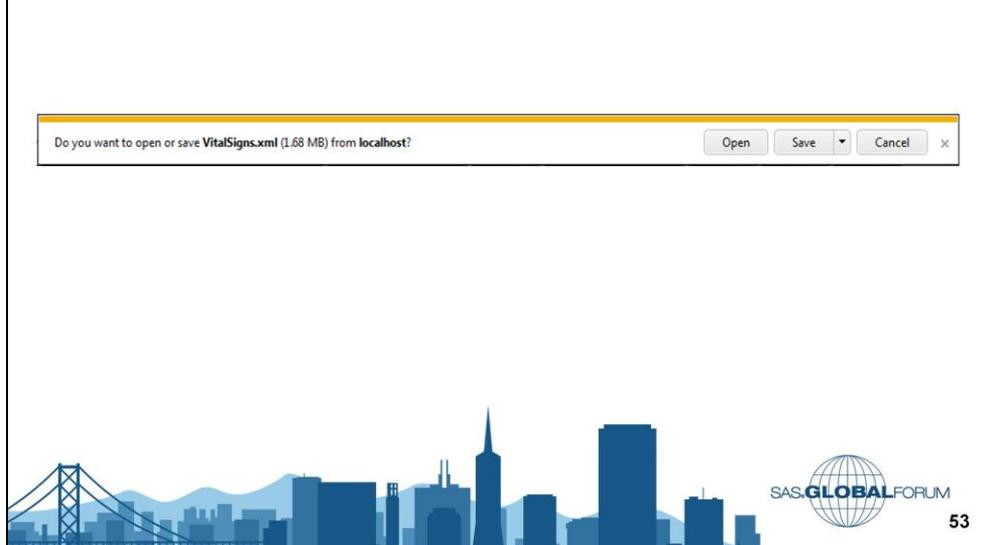
Here is a simple Web page that is used to execute SAS code stored on a server using the SAS/IntrNet Application Dispatcher.

The code that is executed is substantially similar to the final version of the code that we used to generate XML file, with a few changes to "Web-enable" it.

Refer to the accompanying paper for instructions on how to Web-enable your code.

Clicking **Download to Excel** executes the SAS program on the server.

# Dynamically-Generated XML



Once the SAS program executes, the results are sent back to the Web browser.

Instead of the results being displayed in the Web browser, you are presented with a "File Download" dialog box.

You can click **Open** to immediately open your SAS output using Excel, or click **Save** to save a copy for later use.

# Dynamically-Generated XML

The screenshot shows a Microsoft Excel spreadsheet with the following data:

Subject ID: 1				
Vital Signs: Time	Respiratory Rate (/min)	Heart Rate (bpm)	SBP (mmHg)	DBP (mmHg)
-2.00	16	105	135	84
-1.00	9	59	128	73
0.02	9	64	126	83
0.08	9	64	133	91
0.17	12	72	136	90
0.25	14	60	130	82
1.00	16	73	135	73
2.00	12	72	124	65
3.00	12	81	132	77
4.00	21	80	120	70
5.00	19	80	120	70
8.00	16	96	150	90
24.00	20	80	120	70

Subject ID: 2				
Vital Signs: Time	Respiratory Rate (/min)	Heart Rate (bpm)	SBP (mmHg)	DBP (mmHg)
-2.00	15	72	94	56
-1.00	12	55	135	78
0.02	12	50	127	77
0.08	14	58	104	73
0.17	13	54	123	79
0.25	18	55	122	75
1.00	24	60	116	78

54

Here is the SAS output, created in real-time, and delivered to the client. Refer to the accompanying paper for more information about this topic.

*SAS/IntrNet® 9.3: Application Dispatcher.*

<http://tinyurl.com/3qpxzlg>

*SAS® 9.3 Stored Processes: Developer's Guide.*

<http://tinyurl.com/3wyw6fk>

## Conclusion

Feature	PROC EXPORT	ODS CSVALL	ODS ExcelXP Tagset
Supported on all operating systems	● <sup>1</sup>	●	●
Multiple worksheets	● <sup>2</sup>	● <sup>2</sup>	●
BY value in worksheet name	● <sup>2</sup>	● <sup>2</sup>	●
Suppress BY line text	●	●	●
Variable labels in column headings	●	●	●
Support title, footnote & BY text		●	●
Title text contains subject ID		●	●
Multiple tables in a worksheet		●	●
Formatted column headings			●
Support for Excel number formats			●
Style formatting support			●
Little or no manual intervention			●

<sup>1</sup>Except z/OS    <sup>2</sup>Requires manual steps

55

This table lists the features supported by the 3 techniques.

## Conclusion

- Delimited files best for data exchange
- Limited/no formatting w/ delimited files
- Use ExcelXP tagset to create richly-formatted XML file
- Resulting XML file can be viewed with Excel
- Make use of tagset options
- Apply ODS style overrides
- Use Excel formats instead of SAS formats



The EXPORT procedure creates delimited files that are useful for exporting only data values to Excel - style formatting is not supported. The ODS CSVALL destination also creates delimited files, but supports some additional features, such as title, BY-line, and footnote text.

The SAS 9 ExcelXP ODS tagset provides an easy way to export your SAS data to Excel workbooks that contain multiple worksheets. By using ODS styles, style overrides, and a tagset that complies with the Microsoft XML Spreadsheet Specification, you can customize the output to achieve your design goals.

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 Microsoft Office applications.

## Resources

- Paper & Download Package

[support.sas.com/rnd/papers/index.html#excel2013](http://support.sas.com/rnd/papers/index.html#excel2013)

- Vince's ExcelXP Resources

[www.sas.com/reg/gen/corp/867226?page=Resources](http://www.sas.com/reg/gen/corp/867226?page=Resources)



## Contact Information

Please send questions, comments and feedback to:

Vince DelGobbo  
sasvcd@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 ([support.sas.com/usergroups/namerica/lug-form.html](http://support.sas.com/usergroups/namerica/lug-form.html)) at least eight weeks in advance.



### About the author:

Vince DelGobbo is a Senior Software Developer in the Metadata and Execution Services group at SAS. This group's responsibilities include the SAS/IntrNet Application Dispatcher and SAS Stored Processes. He is involved in the development of new Web- and server-based technologies, as well as integrating SAS output with Microsoft Office. He was also involved in the early development of the ExcelXP ODS tagset. Vince has been a SAS Software user since 1982, and joined SAS in 1992.