How do I export from SAS to Excel files: Let me count the ways

92

By Chris Hemedinger on The SAS Dummy | February 11, 2012 Programming Tips

I have a love-hate relationship with spreadsheet data. I am interested in finding data to analyze, and if it arrives in the form of a spreadsheet, I'll take it. And I like to deliver results and reports, but often my constituents ask for it as a spreadsheet that they can then manipulate further. <*Sigh*.>

A spreadsheet is *not* a database, so it can be a challenge to whip spreadsheet data into analysis-ready shape. Likewise, saving results as a spreadsheet can lose something in the translation -- usually value formatting, appearance attributes, or graphs.

SAS offers many ways to read from and write to Microsoft Excel spreadsheets. They each have pros and cons. This post is an inventory of the methods that I know about for *creating* Excel files from within SAS.

Some "bits" about 32-bit and 64-bit architecture

Before I get to the Big List, let me set the stage by describing a few terms and concepts.

In order to create Excel files directly from SAS, you need SAS/ACCESS to PC Files. This product enables all sorts of file exchanges between SAS and other PC-based applications, including Microsoft Excel, Microsoft Access, SPSS, and more.

SAS/ACCESS to PC Files includes a component called the PC Files Server. The PC Files Server is a service-based application that runs apart from SAS on a Windows node. It accepts requests from SAS to convert data to and from Excel (and other file types). Traditionally, this innovation allows SAS running on UNIX (where there are no native facilities for Excel data) to exchange data with PC-based data programs.

Recent changes in desktop computing have complicated the picture. Newer desktop machines all have 64-bit architecture, and most organizations are now adopting 64-bit versions of Microsoft Windows. All of your 32-bit applications (designed for x86 architecture) still can still run, of course, but there are a few hard-and-fast rules. One of those rules is that a 64-bit application cannot dynamically load 32-bit modules in its process space. And guess what? There is a better-than-even chance that the built-in data providers that you have for Microsoft Excel -- the bits that allow SAS to write to Excel on Windows -- are 32-bit modules. This means that the PROC EXPORT DBMS=EXCEL program that worked in your 32-bit SAS session will not work in your 64-bit SAS session.

There are two remedies for this bitness mismatch. First, you could install the 64-bit data providers (which accompany the 64-bit version of Microsoft Office). But you cannot have both the 32-bit and 64-bit versions of these data providers on the same machine; if you have 32-bit Microsoft Office, then you're stuck with the 32-bit providers for now.

The second remedy is to use the PC Files Server, right there on the same Windows machine where SAS is running. This allows a 64-bit SAS process to delegate the data exchange to a 32-bit PC Files Server process. Thanks to the out-of-process communication, this circumvents the bit architecture mismatch. To make this work you don't have to set up any additional software, but your SAS programs must change to use DBMS=EXCELCS. The EXCELCS keyword tells SAS to use the PC Files Server instead of attempting to use in-process data providers.

Exporting to Excel: ways to get there from SAS

With the architecture lesson behind us, here's my list for how to put SAS content into Microsoft Excel. I won't dive into much detail about each method here; you can follow the links to find more documentation.

These methods use features of SAS/ACCESS to PC Files:

LIBNAME EXCEL – reads/writes Excel files at the sheet level when the bitness of SAS (32- or 64-bit) matches the bitness of Microsoft Office installed (or more specifically, the ACE drivers that accompany Office). An Excel file is viewed as a SAS library, while sheets/ranges are the member tables. Requires exclusive lock on an existing Excel file.

LIBNAME PCFILES – does the same as LIBNAME EXCEL, but uses PC Files Server. Good for UNIX and for Windows configurations where bitness of SAS and Microsoft Office don't match.

LIBNAME XLSX - reads and writes native XLSX files without the need for Microsoft components or the PC Files Server. This works the same way on all platforms: Windows and UNIX based. LIBNAME XLSX is available in SAS 9.4 and later.

PROC EXPORT DBMS=EXCELCS – uses PC Files Server to write an Excel file. There are various options to control the output behavior. Good for UNIX and for Windows configurations where bitness of SAS and Microsoft Office don't match.

PROC EXPORT DBMS=EXCEL - writes Excel files when the bitness of SAS (32- or 64-bit) matches the bitness of Microsoft Office installed (or more specifically, the ACE drivers that accompany Office).

PROC EXPORT DBMS=XLS – writes Excel (XLS) files directly, no driver or PC Files Server needed. Has limits on volume and format. Works on Windows and UNIX.

PROC EXPORT DBMS=XLSX – new in 9.3M1, writes Excel 2010 files (XLSX format) directly. No driver or PC Files Server needed. Works on Windows and UNIX.

The following methods *do not* require SAS/ACCESS to PC Files, so they are popular, even if some don't produce "native" Excel files:

ODS EXCEL - produces a native XLSX file, and can include graphics, formatting, and formulas with the use of special directives. This method is available in SAS 9.4 Maintenance 3 and later. See this post for tips about achieving some fancy effects with ODS EXCEL. This method should replace your use of ODS TAGSETS.EXCELXP if you're still using that.

PROC EXPORT DBMS=CSV – produces comma separated value files, most often used in Excel.

ODS TAGSETS.CSV (or just DATA step and FILE output) – produces comma separated value files, most often used in Excel.

DATA step, using FILE and PUT to create delimited files - a simple approach that provides a little bit more control than TAGSETS.CSV or PROC EXPORT DBMS=CSV.

ODS TAGSETS.EXCELXP – uses ODS to create an Office XML file. Provides a fair amount of control over the content appearance, but recent versions of Excel do not recognize as a "native" format, so user is presented with a message to that effect when opening in Excel.

FILENAME DDE – uses Windows messages to control what goes into an Excel file, down to the cell level. Works only when SAS for Windows and Excel are on the same machine. Doesn't work in SAS workspace servers or stored process servers (often accessed with SAS Enterprise Guide). It's an antiquated approach, but offers tremendous control that many long-time SAS users enjoy -- when it works. See why your DDE programs don't work anymore.

SAS client applications make it easier

While I've focused on the SAS programming methods for creating Excel files, applications like SAS Enterprise Guide and the SAS Add-In for Microsoft Office make the operation a breeze. SAS Enterprise Guide can import and export Excel files through point-and-click methods, and SAS/ACCESS to PC Files is not needed to make that work. (However, the work is not captured in a SAS program, so it cannot be run in SAS batch jobs or stored processes.)

SAS Add-In for Microsoft Office turns the problem on its head. By allowing you to access SAS data and analytics from within Microsoft Excel, you *pull* the results into your Excel session, rather than export them from your SAS session.

Tags

SAS/ACCESS PC Files | pc files server | export | excel

Share

ABOUT AUTHOR

Chris Hemedinger

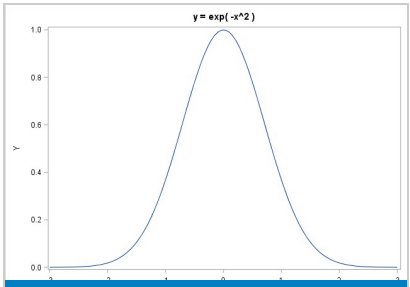
Senior Manager, SAS Online Communities

+Chris Hemedinger is the manager of SAS Online Communities. Since 1993, Chris has worked for SAS as an author, a software developer, an R&D manager and a consultant. Inexplicably, Chris is still coasting on the limited fame he earned as an author of *SAS For Dummies*. He also hosts the SAS Tech Talk webcasts each year from SAS Global Forum, connecting viewers with smart people from SAS R&D and the impressive work that they do.

RELATED POSTS

Data for Good | Programming Tips

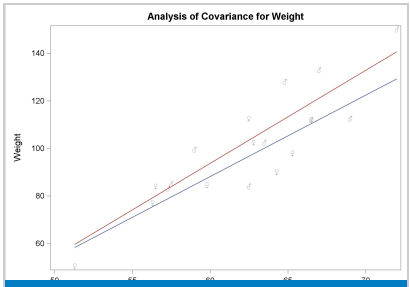
September 17, 2018



Programming Tips

September 17, 2018

Linearly spaced vectors in SAS



Data Visualization | Programming Tips

September 10, 2018

Essential SAS tools to bring to your next


Chris Hemedinger


Rick Wicklin

Advanced ODS Graphics: Unicode


Warren F. Kuhfeld

92 COMMENTS


Alan Churchill on February 11, 2012 5:35 pm

Thanks Chris for the specifics.

There are loads of other ways to handle the issue. SAS datasets are ODBC and OleDb compliant so you can read them in using Excel. A pull strategy vs a push. Not perfect, but an alternative.

I have used .NET apps to convert SAS datasets to Excel. The advantage to that is pure power and speed. Far more control over how the worksheets come out and the speed is incredible.

Finally, to address the issue with SAS on Unix and being able to generate Excel worksheets in a data step, I have an entry on sascommunity.org called SaviCellsPro. That allows a data step to write a simplistic XML structure and then execute a .NET app to convert that into binary Excel workbooks.

There are others but it is a fascinating world. I tend to use everything depending upon the particular client and need...and the 32/64 bit issue has 'bitten' me multiple times now.

- Alan

Reply ›

Alan Churchill on February 11, 2012 5:37 pm

Before I forget, the advantage on the .NET side is that we can control things like print options, graphs, OLAP cubes, etc. Things outside of just dumping the data.It is not always needed so that approach is not always the best, however...it is there if needed.

Reply ›

Tricia Aanderud on February 12, 2012 9:31 am

In the LinkedIn forums someone was talking about the IOM bridge - I had never heard of this method to working with EXCEL - can you make a few comment about it?

Thanks!

Reply ›

Chris Hemedinger on February 12, 2012 9:47 am

Tricia, the "IOM bridge" is part of SAS Integration Technologies, the SAS product that allows any client application to communicate with a SAS server. It's the mechanism that Enterprise Guide and the SAS Add-In for MS Office use to talk to SAS metadata and workspaces. It can also be used directly by a number of Windows programming technologies: C++, .NET (C# and VB) and even VBA (within Excel). Programmers can find more information in the SAS Integration Technologies Windows Development Guide.

Reply ›

Alan Churchill on February 12, 2012 11:51 am

Tricia,

The IOM bridge is used, especially for submitting jobs to SAS. I use it all of the time.It has a lot of nice features.

If you have standalone SAS datasets, you can also READ them using ODBC or OleDb without the IOM bridge. Writing is a separate subject.

Reply ›

Shri Sastry on February 12, 2012 12:20 pm

Chris,
I have been using ODSTagsets.ExcelXP in my stored processes, to generate the results in Excel and was always annoyed with the message that comes to warn the users before opening the file in excel. I did some research and found that if I used one of the follwoing two methods, you don't get the message any more
1. Make some registry changes so excel doesn't display the warning message anymore. SAS Usage Note 31956, provides the details, but some places do not allow for registry changes on clients machines so this may not work for them.
2. I feel this is a better option, if you are using ODSTagsets.excelxp in a stored process. You can set the content type of the output file to be "text/xml" using the "stpsrv_header" function, and then generate your xml file. This file opens in excel and here is the fun part, you don't get the warning message in excel when trying to open the file.

Shri

Reply ›

Chris Hemedinger on February 12, 2012 1:09 pm

Shri, thanks for the tips! There are numerous SAS Global Forum papers on this topic as well -- check the conference proceedings!

Reply ›

Jerry on February 13, 2012 9:58 am

I'm using a proc export on my web page and it works for some but it mostly generates a 7kb xlsx file with a tab name of "A266FF2A662E84b639DA" and no data. I've seen this be an issue with Microsoft project but not sas except for me. Has anyone seen this issue? Is there a work around?

I have a web app that allows users to select options and generate an xlsx file. All the data is collected and prepared in sas and then exported. If it is an xls file it works fine but when I change the options to xlsx it doesn't.

Here is how my code looks:
proc export data = testdata
outfile= "c:\temp\data.xlsx"
dbms=excel replace ;
run ;

Reply ›

Chris Hemedinger on February 13, 2012 10:34 am

If running on SAS 9.2, you might try:
- ensuring the PC Files Server is installed on your SAS machine
- change your program to use DBMS=EXCELCS

Even though this isn't necessary when using 32-bit SAS on Windows, it does delegate the EXPORT process through a different path that might work more consistently.

On SAS 9.3 maint 1 or later, you can use DBMS=XLSX in PROC EXPORT.

Reply ›

jerry on March 2, 2012 12:58 pm

Chris,

Thanks for the tips.

I tried DBMS=EXCELCS and it gives me an error "ERROR: DBMS type EXCELCS not valid for export."

I ran PROC SETINIT NOALIAS; RUN; and this was listed
"SAS/ACCESS Interface to PC Files" does this mean it is installed?

Thank you for your help. This issue has driven me crazy and I can't find any reason why this is happening.

Thanks

Reply ›

Chris Hemedinger on March 2, 2012 1:02 pm

Jerry, for this to work you need to have the PC Files Server installed. You have the license with SAS/ACCESS to PC Files; you just have to install the bits. See SAS note 43802 for details.

Reply ›

FriedEgg on February 23, 2012 3:21 pm

Very nice post, also not mentioned yet are the msoffice2k and tagsets.msoffice2k_x markups for ODS which are html variants for Excel books and have some slightly different capabilities than tagsets.excelxp such as embedding images which Microsoft does not allow in their XML markup that excelxp uses.

Reply ›

Chris Hemedinger on February 23, 2012 3:25 pm

Thanks! I knew that I could count on readers to fill in the gaps.

Fact: in early releases of SAS Add-In for Microsoft Office, tagsets.msoffice2k was used to generate results to pull back into the Office clients. When SAS Report was invented later, it enabled far more flexibility (enabling support for PowerPoint, for example).

Reply ›

Ralph Winters on February 23, 2012 4:14 pm

If I am in Enterprise Guide (table browse mode) and need data quickly, I often simply copy and paste the data directly into Excel. Unfortunately the column labels do not come along with the paste.

Reply ›

Chris Hemedinger on February 23, 2012 4:32 pm

Ralph, yes, that's a limitation.

But here's one thing you *can* do. Right-click on the data node in the process flow, select Properties. On the Columns tab, select "Copy to Clipboard". You can then paste that into Excel. It is row-oriented though, and not column-oriented.

Then you *could* copy just the column names in Excel and use Paste->Transpose to make new column headings. This is a bit klunky, but it may save you some typing.

Reply ›

Pingback: Do I have to choose between Excel and SAS? - The SAS Training Post

LeRoy Bessler on February 28, 2012 10:16 am

Chris, “Let me count the ways” is, for me, exactly the problem. I long ago coined the term "Options Over-Choice" for the situation faced by users of SAS (and other) software.

The single best alternative to create a highly formatted report from a SAS production batch application, with the widest capability (even if it sometimes means using an empty, but partially formatted, template workbook to be loaded and formatted further dynamically from SAS code, or the desperate measure of programmatically sending keystrokes to Excel from SAS code), is Dynamic Data Exchange (aka DDE).

This old technology (whose death has been prematurely forecasted repeatedly) offers more single-solution power than any of the partial solutions provided by SAS developers to date.

Various non-SAS-Institute DDE-oriented authors have discussed what I call "SAS-with-Excel Application Development". One covers a complex multi-function SAS macro, another has a toolkit of single-function SAS macros and sample programs, and the initial conspicuous contributor in this domain (Vyverman) covered parts of the general problem in a few early papers.

I’m still hoping for a single, reliable solution created, endorsed, and supported by the SAS development team for programmers to create production batch applications that can create an Excel workbook formatted exactly as desired.

For a quick, one-time dump, PROC EXPORT is great.

For a simple, slightly formatted report, ODS HTML (or ODS HTML3) with a file type of XLS is adequate.

I experiment with ALL of the ODS tagsets for Excel, and inevitably find that whatever is my current choice meets some needs, but not all needs.

Call me lazy, but just give me one way that I can depend on.

Thanks for starting the dialogue on this important topic. The commonest ultimate destination of data prepared with SAS software is an Excel workbook. Almost all computer users have Excel, and it allows them to work with the data further with a tool that they already know how to use. But I like to deliver to the data viewer/user exactly what she/he wants to see. Not more work to do in order reshape the data delivery.

Thanks.

LeRoy Bessler

Reply ›

Chris Hemedinger on February 28, 2012 10:42 am

Dr. Bessler, thank you for the thoughtful response. I agree that DDE is an incredibly flexible (and fast) approach to the problem. In my early days at SAS I was tasked with writing some simple, repeatable DDE examples for the *SAS Companion for the Microsoft Windows Environment* (my first SAS book); I remember the challenge with fondness.

These days, the main obstacle for DDE is not one of technology, but of topology. If you're a SAS professional who provides Excel-based reports as one-off requests, you can manage that all from your desktop SAS environment. But if you need to scale that to run from stored processes, and/or within SAS Enterprise Guide in distributed environment, or within any number of other "enterprise-wide" deployed systems, DDE might not cut it. As you know, DDE relies on Windows messages between two Windows application processes on the same machine. If your SAS server is on one

box and Excel is on another, you must settle for one of the other approaches to create your Excel content. DDE is simply not on the menu.

Next to DDE, I believe that the SAS Add-In for Microsoft Office (running in Excel) offers the most out-of-the-box flexibility and control over formatting/behavior. That's because it uses SAS Report to retrieve the substance of your results from SAS, and client-side code (using Excel automation, the practical successor to DDE) to merge that content with your Excel formatting and non-SAS content.

Reply ›

LeRoy Bessler on February 28, 2012 4:58 pm

Chris,

I understand that you are in, or work with, the BI client tool development team, not the ODS development team.

My concern is with production batch applications, not BI client addressable needs.

Until late 2009, I actually did support a production batch application that ran on a BI server, and did use DDE with great success, but I was not really thrilled with the situation. If a SAS DDE program DOES have a problem in that situation, you can end up with a hung Excel session on the remote server (and a locked Excel workbook), and not everyone has an easy way to deal with that.

Lately, my production batch work for other SAS sites has also entailed creating formatted reports in Excel, and I have not turned to DDE. But, because of the incomplete solutions so far available with the (at least three) different ODS tagsets, I am in the situation of not being able to deliver as much as I would like. I once heard the comment that when a programmer (or a provider of anything) says to a client (or a customer) "You're too picky.", it is really just a substitute for "I can't do that."

Please pass on my SAS for Excel thoughts to the ODS development team. Thanks.

Regards,
LeRoy Bessler

Reply ›

Chris Hemedinger on February 28, 2012 5:21 pm

I **did** work on the BI Clients team for many years, and it's true that there is a special place in my heart for SAS Enterprise Guide and SAS Add-In for Microsoft Office. I know the capabilities (and limitations) of those client apps pretty well.

These days I work in SAS Professional Services on a variety of projects. I spend more time programming with SAS than I ever have, and my projects often have less-than-neat requirements. That means that I don't always get to just pick a technology and deliver what works best within that technology. Instead, I have to get creative in order to deliver exactly what meets the requirement. (I'm guessing that sounds familiar :))

I can appreciate that there doesn't seem to be a single foundation approach from SAS to solve all of the Excel reporting output scenarios -- hence the 10 methods listed in the post. I know that the ODS group is working towards a more comprehensive approach. I will make sure that they see the comments that you've shared.

Thanks for reading and for participating in this discussion!

Reply ›

RT REDDY on March 6, 2012 3:53 am

```
/*SYNTAX FOR SAS TO EXCEL*/

PROC EXPORT OUTFILE="DESTINATION FILE PATH"
DATA=DSN DBMS=EXCEL;
RUN;

/*EXAMPLE FOR SAS TO EXCEL*/
/*EMP DATASET FROM WORK LIB TO D DRIVE WITH NAME EMP1*/

PROC EXPORT OUTFILE="D:/EMP1.XLS"
DATA=EMP DBMS=EXCEL;
RUN;
```

Reply ›

Chris Hemedinger on March 6, 2012 9:34 am

Yes, this syntax is an example of using the MS Jet / ACE drivers to create Excel output from SAS. This particular code would work when SAS is 32-bit and Excel is 32-bit, or when both are 64-bit.

Reply ›

arunpandian on November 16, 2016 2:42 am

dear sir,
your information was very use full to me , i had another one doubt regarding that is there is any way to convert excel to xpt.

Reply ›

Chris Hemedinger on November 16, 2016 8:51 am

If you have SAS 9.4, you can use the XPORT engine to create these files. See the doc here.

Example syntax:

```
libname source 'SAS-data-library'; libname xportout xport 'transport-file'; proc copy in=source out=xportout memtype=data; run;
```

Reply ›

Amar Harolikar on March 29, 2012 2:05 pm

Hi Chris,

Very useful post for end users like me.

Regards
Amar

Reply ›

neha on April 22, 2012 9:56 am

Hi,
I am using ods tagsets.excelxp to write my SAS results in an Excel. Now as part of the second iteration I want to open the same excel spreadsheet and write the results. All this happens dynamially meaning that I do not know for sure how many rows get populated in that spreadsheet.
The challenge here is that I am not finding any option which allows me to open the same sheet again for writing the results.
Any suggestions??

Thanks

Reply ›

Chris Hemedinger on April 24, 2012 8:23 am

Neha,

One of the limitations of EXCELXP is that the tagset doesn't create a true native Excel file. It's a format that Excel can read, but not a form that allows you to update the content "in place" until Excel has had a chance to read it in and save it natively.

That said, you can probably get some more information from the experts on communities.sas.com - check out the Office Integration discussion forum.

Reply ›

Pingback: The top gotchas when moving to 64-bit SAS for Windows - The SAS Dummy

Wahil on May 24, 2012 3:39 pm

Hi All.

What I need to do, is just the opposite of what was discussed here...We need to not let EGuide users save data into his desktops...

Reply ›

Chris Hemedinger on May 24, 2012 3:46 pm

Some of that can be controlled via role-based capabilities. But not all "save local" opportunities can be completely shut down in the desktop environment.

Reply ›

Roger on September 13, 2012 10:13 am

Any ideas on how to export multiple tables to a separate tabs on a single Excel workbook? Anyone have a macro idea for this?

Reply ›

Chris Hemedinger on September 13, 2012 10:15 am

Roger,

You can do this with LIBNAME EXCEL (or LIBNAME PCFILES for 64-bit SAS), where each data set represents a different worksheet (tab). Or you can use options in TAGSETS.EXCELXP to control this behavior.

Reply ›

Nancy on October 11, 2012 3:25 pm

I am using SAS Enterprise Guide for the first time. I have a process that creates an output file. The name of the output file was created using a macro variable that I capture at the beginning of the process from a prompt to the user. Example - I ask for the Month and Year to use for processing. I use this macro variable fine throughout the process. When it comes to exporting the file to excel I am in EG and have the file open and click on the EXPORT. When it says Export as Step in Project, I click yes and follow the wizard. But it saves the export process with the macro name resolved, so now when I want to redo this project and select another month and year it doesn't know how to do the Export part. Can you help me with this section. I don't want just a generic name, I need the month and year attached to the exported file name. Can you give me a suggestion for how to do that? Thanks

Reply ›

Chris Hemedinger on October 11, 2012 4:15 pm

Nancy,

Do you have a way to export the data using code? That is, do you have the ability to run PROC EXPORT in your SAS environment? It's coding (and not point-and-click), but you could use that to pick up your macro variables/prompts.

Also, if you had a way to then **download** the exported Excel files from your SAS environment, as a step in your process flow, would that help?

Reply ›

Ton Wiegman on October 22, 2012 5:16 am

Hi Chris,
Can you elaborate a bit more on the statement below.
'*PROC EXPORT DBMS=XLS – writes Excel (XLS) files directly, no driver or PC Files Server needed. Has limits on volume and format. Works on Windows and UNIX*.'

We are using Marketing Automation (still an old version: 4.4), and would like to be able to use the Excel output for some of our agencies, in stead of doing post-processing.
I am aware of the limit of 65k, but the output becomes unreliable.
I have been able to put out 60k records in a correct way, whereas 50k records had an unrecoverable read error in excel, and the data were the same. When trying to repair, no column headings were in the sheet, and random columns were populated.
Can I perhaps avoid the format and volume limitations?

Reply ›

Chris Hemedinger on October 22, 2012 8:20 am

Tom, there are a few limitations on size and content with DBMS=XLS. That method writes the XLS file directly using a file format that Excel recognizes, although it's not the most recent file format that Excel supports. For ideas about some of the limitations, check the SAS notes related to DBMS=XLS.

Reply ›

Mike Orlov on November 28, 2012 12:22 pm

Chris,
great article and discussion. What could you say about ODBC, OLEDB and other "pull" strategies Alan Churchill has mentioned? I found pulling SAS data in excel to be very smooth from the process point of view, since you could create multiple views/summaries (using pivot tables for example) graphs etc from the same detailed SAS dataset very quickly in Excel and it is very easy to refresh the presentation. The SAS data set is produced independently from the presentation update, adding flexibility to the process. The downsides I see so far are:
1. dealing with SAS dates (not translated automastically to MS Office "start from 1/1/1900 format")
2. having to set up ODBC libraries (if ODBC method used)
Also if OLE DB providers are used, Microsoft lists multiple providers (SAS Base... SAS IOM... SAS Local... SAS OLAP... SAS share...) not sure what is the difference.

Reply ›

Chris Hemedinger on November 28, 2012 1:21 pm

Mike,

The "Pull" strategy is exactly how the SAS Add-In for Microsoft Office works to pull SAS content (data and reports) into Excel and other clients. Under the covers, it uses the SAS IOM OLE DB provider to connect to a SAS session.

You're correct that the basis for Microsoft and SAS dates are different. On a previous post I offered a SAS-based method of conversion from Microsoft DateTime to SAS; what you need from Excel (when using .NET or VBA) is something that goes the other way around.

The different providers are related to different types of SAS "endpoints". If you don't have a SAS session to connect to but have a SAS data set file, you can use the Local provider. For IOM servers (SAS Workspace sessions), you use the IOM provider. For SAS OLAP Server connections, you use the SAS OLAP provider (complies with OLE DB for OLAP - ODBO), and for remotely administered SAS libraries in a SAS/SHARE server, you can use the SHARE provider.

Reply ›

Mike Orlov on November 28, 2012 3:08 pm

Thank you

Reply ›

Chandrta Shekhar on December 17, 2012 1:44 am

Hi,

I have an excell file with 7 sheets.I have to count these sheets and import into sas datasets. Excell is not installed on my system.I have to compare these sheets with other excell sheets.
Kindly help me to solve this problem.

Reply ›

Chris Hemedinger on December 17, 2012 9:46 am

Chandrta,

In that situation, I usually use LIBNAME EXCEL (or LIBNAME PCFILES on a 64-bit version of SAS). This allows you to navigate the Excel sheets as SAS tables. This requires SAS/ACCESS to PC Files.

Reply ›

S. Hurley on May 6, 2013 3:42 pm

Hi,
I agree there should be a general approach with good formatting capabilities. Frankly, I am tired of spending hours counting cells to enter in easily misplace column and row numbers into DDE macros. Those ODS tagsets are a pain. Life should not be this annoying.
- SH, faithful programmer

Reply ›

Pingback: The best of SAS blogs for 2012 - SAS Voices

Mark Rengstorff on May 14, 2013 6:46 pm

Chris, I have been using PROC EXPORT DBMS=EXCEL REPLACE for a long time. Recently some of my data is not replacing (or updating). I have multiple sheets, some update and some don't. The log says that everything is good (no errors). I am using SAS 9.3 (32 bit) with windows 7, Excel 2007. Have you or anyone else come across this problem?

Reply ›

Chris Hemedinger on May 14, 2013 7:25 pm

Mark, I haven't seen that. It sounds like you should open a Tech Support track. They may have some advice to gather additional diagnostics.

Reply ›

manish on May 22, 2013 8:55 am

Mark, I had a similar issue recently. Were you able to get this resolved?

Reply ›

Jenny Fehr on May 30, 2013 2:15 pm

One of our developers created some software to gather some Oracle data into SAS and then manipulate it according to our business requirements. She converts the output to .xls. But when I open the output file in Notepad, I notice the following:

```
000001679801821
```

The Oracle datatype was Text. Somehow, somewhere along the line, it gets changed to Number, which we can't have, because opening the file in Excel causes the leading zeroes to get dropped. The developer says she doesn't think it can be fixed. It's not that I don't believe her, but it just seems odd that it can't be fixed. Any ideas as to what point in the process the datatypes are getting changed? Is it something in her code or is it at point of the export process?

I'm not a developer so please reply using small words ;)

Thanks!

Reply ›

Chris Hemedinger on May 30, 2013 5:26 pm

Jenny,

Despite the formatting challenges you had while entering this comment (apologies for that), I think I understand the problem. The Oracle column was meant to be VARCHAR, and perhaps it was brought into SAS that way (or perhaps not, you don't know), but when exported to Excel it was numeric.

As this post discusses, there are several different ways to export to Excel, and some of them give more control than others for the output formats. There are many SAS notes and documents for how to retain leading zeros and preserve the type. Here's one, for example.

Also, you have options for how to read data from Oracle into SAS, if you are using SAS/ACCESS to Oracle (or ODBC). The DBTYPE= option provides this control when you access database tables.

Given your description of the problem, I agree with your intuition: this **can** be fixed. Your developer may need some guidance, or it might be that the "fix" comes with a trade-off that the developer finds unpalatable.

Reply ›

Keith Adams on June 26, 2013 1:10 pm

Very interesting and useful post. I've long used DDE to support two-way data transfer between Excel and SAS, and for formatting the spreadsheet. A couple of points that came to me while browsing the article and the comments:

- Frequently, you'll want to add to an existing spreadsheet, not create a new one. What I'd do with DDE is create a template workbook with some pre-formatted hidden sheet, and use SAS with DDE to do things like duplicate a hidden sheet then write to id (for instance, I had to have a dynamic means of converting web form fields into one sheet per form, and this approach worked well for this.)

- However, today has been the day of big education for me as I discovered that my dream of doing a quick conversion of my SAS/DDE application to something that worked through the Add-In with stored processes that still use DDE wasn't going to work!

- So what I've decided to do is to still use a pre-formatted workbook with hidden sheets, still do most of the number crunching with SAS (using a stored process called via the Add-In), and use Excel VBA in place of DDE to transfer data back and forth to SAS and to format the sheets.

- What I'm missing is a handy set of VBA functions that work with the extensions for the Add-In to do common tasks such as write this SAS dataset to cell A of sheet B; or create a SAS dataset by reading an Excel range. I'm going to develop these myself, but since my VBA expertize is not enormous, if anybody already has written such functions, please advise!

Thanks, Keith

Reply ›

chandra sekhar on August 30, 2013 2:25 am

Hi Chris,

Thanks for your post. It is very informative. I am running proc export dbms=xls code using SAS 9.2 version in unix box. I would like to update an excel workbook. The workbook has data in first sheet and remaining 3 sheets having graphs created using data from first sheet. I am not able to update the data in first sheet using proc export dbms=xls opiton. We don't have PC file server on windows machine to explore other options. Please let me know any other solutions that would help us in updating the existing spreadsheet with proc export in unix environment.
Regards,
Chandra

Reply ›

Chris Hemedinger on August 30, 2013 8:23 am

Chandra,

The tricky part of your problem is updating an *existing* spreadsheet, which PROC EXPORT does not support in SAS 9.2. This capability has been added in SAS 9.4.

If you have the SAS Add-In for Microsoft Office, you could "turn the problem around" and use Excel to pull the SAS data you need into the proper location in the spreadsheet.

And if you had a PC Files Server, you could potentially use the LIBNAME PCFILES engine to treat your workbook like a collection of tables (each spreadsheet as a table), and replace just one table as needed.

Reply ›

Arnold Huxtable on September 4, 2013 12:48 am

Excellent info. Used to just write scripts in Python for converting from SAS to SPSS. Now have more options. Have always used SAS2SPSS by Chronicon to convert the datasets on Windows, but linux sufficiently more difficult. Thanks for the help!

Reply ›

Birger Larsen on December 2, 2013 12:48 pm

I've had some problems exporting to Excel through a CSV file. When Excel opens the file, it guesses how to read data, transforming it into what Excel thinks it should be. I wrote a SAS macro to write the CSV file in a way I know Excel will not tamper with. When writing a numeric variable i just write its value. If there is a format on the variable I write the formated value embraced with ="value". Excel will not touch this. It will just be read as a string value. All character variables I write the same

way embraced with ="value". You cannot read the CSV file from other programs, but it works very well with Excel. You can alter the file in Excel and save it again without trouble.

Reply ›

Chris Hemedinger on December 2, 2013 12:58 pm

Birger, Thanks for sharing this. Your technique is one of many that creative SAS programmers will use when they want to get a certain behavior out of Excel, based on the needs of the downstream processes.

Reply ›

Ralph Franklin on December 2, 2013 7:13 pm

Any hints on how to deal with SAS created multi tab Excel workbooks that always open in Excel Group mode?
As a recent convert to the 64 bit world - Windows 7, Excel 2010 and SAS 9.4 this is a new problem that I did not have in the old 32 bit world.

FILENAME RevX 'K:\\SAS Data Exports\06 DEC\RevMon';

.... data step creating a SAS data set "table" and multiple Proc Export sheets with different permutation of the "table" data set.....

PROC EXPORT DATA= Table
DBMS=XLSX
OUTFILE = RevX REPLACE;
SHEET= 'AllGFCASH';
RUN;

Reply ›

Chris Hemedinger on December 3, 2013 10:45 am

Ralph, you might pose this question to the Integration with Microsoft Office support forum. If you add a picture of what you expect/want, and a sample of your code, the participants in the forum can provide you with some good advice.

Reply ›

Gady Kotler on December 6, 2013 4:37 pm

New way to read XLSX files directly and convert it to SAS data set:

%xlsx2sas macro

requires SAS/Base only, support for NLS, support for Unix/Linux and Windows.
SAS versions: 9.1 and above.

Simple to use:

%xlsx2sas(
sheetnumber = 1 ,
xlsxrows = ALL ,
xlsxcols = ALL ,
xlsxlabels = no ,
xlsxmix = ALL ,
infilename = /home/alex45/test/demo.xlsx ,
outdata = work.demo
);

A demo version can be downloaded at : http://www.bixforsas.com
You are all invited to test it on your Excel 2007/2010 files.

Reply ›

Chris Hemedinger on December 9, 2013 10:16 am

Gady, thanks for sharing the information. I'm sure that your tool will be a welcome addition for those looking for additional flexibility.

In the meantime, SAS/ACCESS to PC Files has added more support for XSLX in SAS 9.4, which can work without PC Files Server, and can update and add sheets in existing workbooks. Also, the Maint 1 update for SAS 9.4 (due out very soon) includes an experimental ODS EXCEL destination for creating XLSX files; that's part of SAS Base. Enthusiasts will look forward to playing with that.

Reply ›

Gady Kotler on December 26, 2013 12:26 pm

Stay tune for the next coming macro from BiX:

%sas2xlsx (The opposite direction of %xlsx2sas...) which can directly publish colored and traffic lighted SAS reports and SAS graphics into XLSX spreadsheets creating a complete dashboard within Excel 2010 files.

Have a look at: http://bixforsas.com/?page_id=5895

Reply ›

Chris Hemedinger on January 2, 2014 10:56 am

Good to know, Gady.

SAS 9.4m1 offers ODS EXCEL, which can create native XLSX files that contain formatted SAS output, including images. ODS EXCEL is part of Base SAS, and is "experimental" in the maint 1 release of SAS 9.4. Example program:

```
ods excel file="c:\out\myfile.xlsx";
title "Car stats";
proc means data=sashelp.cars;
run;
title "Distribution of MSRP";
proc sgplot data=sashelp.cars;
  histogram msrp;
run;
ods excel close;
```

Reply ›

tim walters on January 3, 2014 3:09 pm

I tried using dbms=xlsx to replace a specific sheet in a workbook but it only wants to replace the entire workbook. Is there a workaround for this? Why would this dbms= be different from the others?

Reply ›

Chris Hemedinger on January 3, 2014 3:20 pm

The SHEET= syntax is supported with DBMS=XLSX -- but this was only recently added in SAS 9.4. See this summary of What's New with SAS/ACCESS to PC Files.

Reply ›

Pingback: SAS Talks: Favorite SAS Enterprise Guide Tricks - The SAS Dummy

Pingback: Experimenting with ODS EXCEL to create spreadsheets from SAS - The SAS Dummy

ramnath on September 24, 2014 7:23 am

Hi Chris

I'm trying to export a dataset from work folder to a local folder. But the amount of data is very high (may be 1000000) and its taking long time (more tha 2 hours to export 50000 rows) to export the file in local drive.

Below are the steps I did in the process flow.
1. Script to create the required dataset in work folder.
2. Used the output dataset as step in project and export it as csv to a local drive (c drive)
Can you advise is there a way to export the data quicker?

Reply ›

Chris Hemedinger on September 24, 2014 7:41 am

Ramnath, see this thread on the Communities.sas.com area for some ideas. There are good and fast DATA step methods to create CSV files from a data set. For simple CSV, a DATA step with PUT statements to an external file is probably quickest.

Reply ›

Pingback: Pass SAS macro variables between another program (VB or Java) | Live Coder

Pingback: Pass SAS macro variables between another program (VB or Java) | JavaPoint

Gisele Guadalupe on February 2, 2016 1:24 pm

Chris, do you know any same FILENAME DDE function for sas in oracle?

tks,

Gisele

Reply ›

Chris Hemedinger on February 3, 2016 10:00 am

You want to use DDE to interact with Oracle data via SAS? I'm not sure that I would recommend that, as the DDE operations happen in DATA step, and it might be difficult to craft a DATA step that accesses the values of your Oracle table efficiently. If possible, I recommend using SQL as an intermediary to get the values you need from Oracle (or insert/update values into Oracle), and let the DDE operations work between Base SAS tables and Excel. SQL is more likely to push work efficiently to the database process.

Reply ›

Mili on March 3, 2016 5:12 am

Hi.

I Use SAS Enterprise Guide. I export some data from SAS to Excel files. An this is ok - it's working. Now, in the next sheet i want to use the pivot table based on the exporting data from SAS. How can i do that?

Now, when I build the pivot on this exporting data I've got a problem, because the exel file is demaged. But i really need this. Please help.

Reply ›

Chris Hemedinger on March 3, 2016 12:58 pm

See if this post from Chevell Parker helps you -- adding Pivot tables with SAS.

Reply ›

Claudia Jimenez on August 21, 2016 6:48 pm

Hi Chris,

How can I export a SAS file to an xlsx Excel file using Enterprise Guide 7.1? Below is my current code and the error. The path is correct and I have unchecked the "read-only" box in the "common" folder properties. Yet, I still cannot export the file.

```
proc export
data=sandbox.mytab
dbms=xlsx
outfile="common/displaytab.xlsx"
replace;
run;
```

ERROR: Temporary file for XLSX file can not be created -> /sasfs/prod/sas/94/controlserver/sasconfig/Lev1/SASApp94/common//displaytab.$$1. Make sure the path name is correct and that you have write permission.

Thanks,
Claudia

Reply ›

Chris Hemedinger on August 22, 2016 7:37 am

Claudia,

You need to specify a complete path for your outfile. Example:
`outfile="/u/claudia/common/displaytab.xlsx'`

Or if using SAS University Edition, try the /folders/myfolders path as documented here.

Reply ›

Jacqueline on October 14, 2016 10:35 am

How do I deal with importing multiple excel files in a macro program, where there seems to be a bit-type mismatch?

Reply ›

Chris Hemedinger on October 16, 2016 3:22 pm

If using SAS 9.4, the best approach is to use DBMS=XLSX (assume these are XLSX files), which doesn't care about bitness and does not require PC Files Server.

Reply ›

Pingback: My favorite SAS tips and tools from 2012 - The SAS Dummy

Pingback: Using ODS EXCEL and PROC EXPORT to bundle Excel-based reports - The SAS Dummy

Rob Ross on March 20, 2017 1:47 pm

Hi,

I'm a Stata user reluctantly using SAS. I have a variable coded like this: 1=yes 0=no. I want to export the data labels to excel, as well as the variable (column) labels. I've been using PROC EXPORT with the LABEL option, but that only gets me the column labels.

Rob

Reply ›

Chris Hemedinger on March 20, 2017 3:26 pm

Hi Rob, so do you want the variable name AND label, or are you looking for the raw value (1 or 0) along with the formatted value (Yes or No)? If the latter, then you will want to add an additional variable to your data set (just for Export) that has the unformatted version (or the formatted version). If the latter, then you won't be able to do that with PROC EXPORT. You code probably do this with ODS Excel if you create a report with these details using PROC REPORT.

Reply ›

Rob Ross on March 22, 2017 9:46 am

Thanks Chris. This is what works:

ods html file='\\Admin.....\ATEST.xls';
proc report nowindows data=work.students;
run;
ods html close;

One issue: in the file path, I would prefer to use a macro %LET path = \\Admin.... --> ods html file='&path.\ATEST.xls';

The macro does not resolve, however. This results in an error that the physical path does not exist. I have had this issue of macros not resolving in file paths before, for example:
filename DIRLIST pipe 'dir "\\Admin...*.csv" /b ' ; will not resolve a macro in place of the actual file path.

Why is this?

Reply ›

Chris Hemedinger on March 22, 2017 12:41 pm

Rob, it's the single quotes. Macro vars enclosed in single quotes don't resolve -- that's a feature of the macro processor. If you want them to resolve, use double quotes.

ods html file="&path.\ATEST.xls";

Reply ›

Alexander on April 26, 2017 2:25 pm

Hello Chris,

Recently I found the following issue with proc EXPORT on EG:

```
data a;
a=1.E-24;
run;
proc EXPORT DATA=a
OUTFILE= "path/file_name.xlsx"
DBMS=XLSX REPLACE;
SHEET="a";
run;
```

```
ERROR: Invalid Operation.
ERROR: Termination due to Floating Point Exception
NOTE: The SAS System stopped processing this step because of errors.
NOTE: There were 1 observations read from the data set WORK.A.
NOTE: PROCEDURE EXPORT used (Total process time):
real time 0.08 seconds
cpu time 0.00 seconds
```

I am using EG 5.1 (5.100.0.14335) Hot fix 21 (32-bit) based on SAS 9.3

Reply ›

Chris Hemedinger on April 26, 2017 4:27 pm

Hi Alexender -- that error is coming from SAS. Since you're on 9.3, I'm going to guess that this problem has been fixed in a maintenance or hotfix. DBMS=XLSX was introduced in 9.3 and improved along the way, fixing many of the edge cases.

Reply ›

Alexander on April 27, 2017 12:34 pm

Hello Chris,

Thanks for the prompt answer. How to find out what maintenance/hotfix resolves this issue?

Thanks.

Reply ›

Chris Hemedinger on April 27, 2017 12:50 pm

Alexander, you'll probably need to work with SAS Tech Support on that. I can confirm it works with SAS 9.4m4, on Windows and Linux.

Reply ›

Alexis on May 31, 2017 2:01 pm

Hi, I usually export datas from SAS to excel using the next code:

```
PROC EXPORT DATA= carpeta.tendencia1 outfile= "~/.../" dbms=xlsx replace;
sheet="tendencia1";
PROC EXPORT DATA=res03.tendencia2 outfile="~/../" dbms=xlsx;
```

sheet="tendencia2";
run;

but actually it shows me the next error:

ERROR: DBMS type XLSX not valid for export.

can you help me with this problem? Thanks.

Reply ›

Chris Hemedinger on June 1, 2017 8:16 am

Hi Alexis, don't you need to name a file (xlsx file) in the outfile= option? Aside from that, DBMS=XLSX requires SAS/ACCESS to PC Files and 9.3m2 or later, I think.

Reply ›

Tina on March 13, 2018 5:29 am

Hi Chis
Thank you very much for your list of excel export methods.

I have a little problem, maybe you can tell me the "correct" way to export my data.
My following example is with missing data and I need to change the negative values, because I need to sum in further steps, therefore we decided to use .n.
When I use "View in Excel" (right-klick on the dataset) I got a correct output, but I want to export automatically for QC.
When I use PROC EXPORT, the output is wrong because the "N" will be "deleted".
Does anyone know why?

```
data test;
infile datalines delimiter="," ;
input num char $;
datalines;
1,1
2,2
-2,-2
;

DATA test1 ;
SET test ;
IF num < 0 THEN num = .n ;
char = cats(num) ;
RUN ;

PROC EXPORT DATA= test1
OUTFILE= "\test1.xlsx"
DBMS=XLSX REPLACE;
SHEET="test1";
NEWFILE=YES;
RUN;
```

Reply ›

Chris Hemedinger on March 14, 2018 11:28 am

Instead of the PROC EXPORT, try this:

```
ods excel file='\test1.xlsx';
proc print data=test1;
run;
ods excel close;
```

Reply ›

Peter on April 12, 2018 4:43 pm

Hi,
I wanted to confirm my understanding is correct.
If we want to populate a named range within excel and we have SAS running on a LInux server, the only way to achieve this is by using PCFiles ?
Thanks

Reply ›

Chris Hemedinger on April 13, 2018 8:21 am

Yes, I believe so. Neither PROC EXPORT with DBMS=XLSX, nor ODS EXCEL, can write to a named range or update a partial sheet.

PROC EXPORT can update a workbook with a new/replaced sheet (entire sheet). So if you're workbook-report is organized such that you can source your range from another sheet in the file, then maybe that can work without the additional PC Files Server.

Reply ›

LEAVE A REPLY

Your Comment

Your Name

Your Email

Your Website

Post Comment