

EXCELing in DDE: Unlock Useful Tools for Processing Excel®

Calvin Guan, Abt Associates Inc., Cambridge, MA

Louise S. Hadden, Abt Associates Inc., Cambridge, MA

ABSTRACT

Want to do more with your Excel files than just dumping in data? Got complex reports with custom formatting that won't work with PROC EXPORT? This paper will show you how to gracefully export your reports to native Excel files using Dynamic Data Exchange (DDE). DDE lets you utilize Excel commands via SAS® code, giving you many more options in fine-tuning your spreadsheet. We will explore using DDE to perform Excel tasks such as changing fonts, freezing panes, and saving as a different format. You can further customize the DDE process by using SAS macros; it is somewhat tricky to implement macros with DDE so the how-to is explained in this paper.

Our programs are run on a Microsoft Windows x64 server in batch mode. However, the DDE processes can be used on other platforms, as long as a licensed copy of Microsoft Office Excel is available on the system.

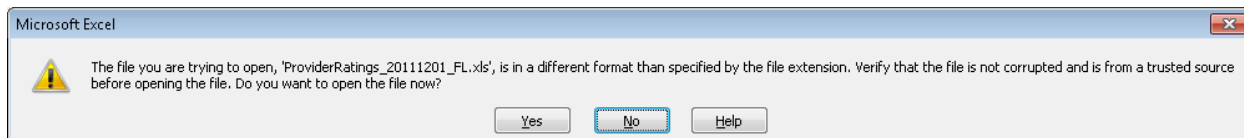
This paper will be most useful to those with an intermediate or greater knowledge of SAS.

INTRODUCTION

The specific project which inspired the use of these tools and techniques is a very large and complex one, with new processing of data occurring each month. The processing is repetitive – a suite of programs is run each month with minor modifications. We take advantage of SAS Macro language to simply change macro variables for date values at the top of each program. Many of the output files generated are in the form of html or xml output. In order to save time and the risk of human error, we also wished to build in a method of saving these outputs in native Excel inside our SAS programs. In addition, on our wish list was the ability to change fonts and freezing panes.

WHY DDE?

One author started looking into using DDE while working on another study with repetitive monthly processing. His desire was to format Excel output to look consistent and attractive every time a report was run, without any manual processing. Our current study, another long-term study with repetitive monthly processing with multiple output files destined to be opened with Microsoft Excel, seemed to be a perfect match with the DDE process. Our outputs have custom formatting including the use of Unicode characters, and were formerly output with the .xls extension so that they would open in Excel on a double-click. However, with the advent of Office 2007 and 2010, a warning is generated on opening these non-native Excel files.



As our files were destined for a client, we wanted to avoid this warning. Since many of the reports have more than 15,000 rows and often more than a screen's worth of columns, we also wished to freeze the header row(s) and first column so that the reports could be viewed more easily by users. We use DDE to save time, effort and risk of manually manipulating non-native Excel files.

A BRIEF DESCRIPTION OF DDE AND HOW IT WORKS

DDE is an acronym for Dynamic Data Exchange, a method by which one software program communicates with or controls another software program. In our case, SAS will be controlling Excel. SAS communicates with Excel by sending X4ML (Excel 4 Macro Language) commands to Excel. Although X4ML is an old macro language, it is still supported by all existing versions of Excel. X4ML syntax is pretty similar to VBA code. In fact, SAS cannot send VBA code with DDE, but CAN run existing VBA macros stored in Excel. Many functions can be performed using DDE, including passing data to excel. Our paper will focus on functions that cannot be easily done with other methods.

For a reference on X4ML functions, consult Microsoft's website or download directly from here:

<https://dl.dropboxusercontent.com/u/18521095/X4ML%20Reference/X4ML%20Reference.zip>.

MANDATORY SET UP

1. `options noxsync noxwait ;`

The `noxwait` option allows the use of `x` command without typing “exit” to continue, while `noxsync` allows SAS to continue its processing while the Excel program runs.

2. Excel must be open on the system you are running your SAS program on. You have two choices: you can open Excel manually or use the `X` command in SAS to open Excel. We choose the least amount of user intervention possible:

```
x "C:\Program Files\Microsoft Office\Office14\excel.exe";
```

3. `filename fileref dde 'excel|system';`

This statement declares the DDE session to be equipped to *fileref*; see the next section for how to call the DDE powered *fileref*.

Replace the path above with the location of your excel program or shortcut.

HOW TO USE DDE WITHIN SAS

General Format

This is the general format for passing a X4ML command:

```
put '[/* X4ML function goes here */]';
```

So the format for a session would look like this:

```
data _NULL_;  
  file fileref;  
  put '[open("/* Full path of file */")]' ;  
  put '[/* a X4ML function */]';  
  put '[/* a X4ML function */]';  
  ...  
run;
```

The file statement calls the DDE-equipped file reference to initialize DDE, while the open function opens a workbook to be the target of our functions.

Saving as XLSX

```
put '[save.as("/* Full path of name to be saved as*/", 51)]'
```

The second parameter is crucial. Just having `.xlsx` extension in the name is not enough to tell Excel to save as XLSX type. Below lists type number for popular formats:

23	CSV
51	XLSX
52	XLSM
56	XLS

If you want to save as different type, consult the X4ML reference for the other type numbers for 2nd parameter.

Should you wish to rerun the saving as function, you will need to implement a code to delete the final output beforehand. Doing so will avoid answering the over-writing prompt. Ways to delete files are explored further down in this paper.

Note about Saving ODS HTML Output as XLSX

When you are creating the ODS HTML output, DO NOT give it a .XLS extension. This will cause the “different format than specified” warning to pop-up when the file is opened. Instead use .HTML extension, and proceed to open the file and edit with DDE as you normally would. (Credit goes to Cynthia Zender for showing me this work around at NESUG 2012)

Freezing Panes:

```
put '[freeze.panes(true,1,1)]';
```

1st parameter – set to true to freeze panes, set to false to unfreeze frozen panes. The other parameters will be ignored if false.

2nd parameter – input a number N, to freeze N number of columns from left of worksheet

3rd parameter – input a number N, to freeze N number of rows from top of worksheet

USING DDE WITH MACROS

X4ML functions must be surrounded by single quotes when using DDE. This poses a problem when we try to pass in SAS macro variables, as they will not resolve inside single quotes.

We use a work around by using the %str function with % to mask the single quotes during compilation. Because %str will generate surrounding double quotes, we use %unquote to remove the unwanted double quotes.

```
put %unquote(%str('%'[/ * X4ML function goes here */] %'));
```

We use macro variables in our monthly processing extensively, so the ability to pass macro variables through to the DDE process is essential. The code snippet bellows demonstrates using macro variables for processing date in our DDE commands. The tables created in this case are HTML, and the final destination is native Excel (.XLSX).

```

**** Use this line in Interactive Mode;
%let a= %sysget(SAS_EXECFILEPATH);

**** Use this line in Batch Mode;
%let a = %sysfunc(getoption(sysin));

%let b = %scan(&a, -1, \);
%let path= %unquote(%sysfunc(tranwrd(&a,&b,)));
%put &path;

**** Converts .html output to .xlsx when called;

%macro transform(totcalls=1,stlist=AK );
length s2 $5;

%DO X = 1 %TO &totcalls ;
    %LET state = %SCAN(&stlist,&X ) ;
    put
%unquote(%str(%'[open("&path.Ratings\ProviderRatings_&fileyear.&filedate._&state..html
")]'%'));
    put '[freeze.panes(true,1,1)]';
    put
%unquote(%str(%'[save.as("&path.Ratings\ProviderRatings_&fileyear.&filedate._&state..x
lsx", 51)]%'));
    put '[close()]';
%END ;

%mend transform;

ods listing close ;

filename htm2xlsx dde 'excel|system';

**** Opens Excel;
x 'C:\Program Files\Microsoft Office\Office14\excel.exe';
**** Converts .html to .xlsx;
data _null_;
    file htm2xlsx;
        %transform(totcalls= 55,
                    stlist = AK AL AR AZ CA CO CT DC DE FL
                               GA GU HI IA ID IL IN KS KY LA
                               MA MD ME MI MN MO MS MT NC ND
                               NE NH NJ NM NV NY OH OK OR PA
                               PR RI SC SD TN TX UT VA VI VT
                               WA WI WV WY ALL );

    put '[quit()]';
run;

```

USING DDE IN CONJUNCTION WITH THE EXCELXP TAGSET

For the same project, we create a multi-tab workbook in SAS which contains one or more reports on each tab, using the EXCELXP tagset, which gives us the ability to retain special formatting such as Unicode characters and to name each tab in our worksheet. The generation of this workbook also uses macro variables and processes extensively. .EXCELXP generates XML, not HTML. The final destination of this workbook is native Excel (.xlsx). Using DDE, we “save as” within the table generation program in a similar fashion to the HTML tables discussed above. Note that we use fully qualified pathnames for the files, which is important as Excel is not in the same directory as the SAS process.

```

* This next routine starts the Excel application.    *;

filename cmds dde 'excel|system';

x 'C:\Program Files\Microsoft Office\Office14\excel.exe';

```

```

data _null_;
    x=sleep(3);
run;

filename cmds dde 'excel|system';

data _null_;
file cmds;

X=sleep(10);

put                                "[open("""S:\Projects\NH-COMPARE\Data_From_CMS\Monthly
Processing\Files&fileyear.&filedate.\5Output\Analysis\Outfiles\Tables1_11_&fileyear.-
&filedate..xml""")]";

put '[ERROR("FALSE")]';

put                                "[SAVE.AS("""S:\Projects\NH-COMPARE\Data_From_CMS\Monthly
Processing\Files&fileyear.&filedate.\5Output\Analysis\Outfiles\Tables1_11_&fileyear.-
&filedate..xlsx""",51)]";

x=sleep(2);

put '[close("false")]';

put '[quit]';

run;

```

CLEANING UP

You may have noticed we are generating a LOT of intermediate files each month (the native HTML and XML versions of our Excel files.) In addition, should we need to re-run programs we will need to delete existing files to avoid the over-writing prompt. There are two different methods we use to do this:

For the XML output, which is a single file, we use the X command to delete the XML output.

```

x "cd Outfiles";
x "del Tables1_11_&fileyear.-&filedate..xml";
x "exit";

```

For the HTML output, which is comprised of hundreds of files, we get a little fancier. The macro routine destroyer uses SAS routines to eliminate files with a given extension.

```

%macro destroyer(totcalls=1,stlist=AK, ext = xlsx);

%DO X = 1 %TO &totcalls ;

    %LET state = %SCAN(&stlist,&X ) ;

    filename delit ".\Ratings\ProviderRatings_&fileyear.&filedate._&state..&ext.";

    data _null_;
        if fexist('delit') then rc = fdelete('delit');
    run;

    filename delit clear;

%END;

%mend destroyer;

```

At the top of the program, the macro routine destroyer is used to eliminate any existing Excel files to be rewritten.

```

**** Use this to delete made .xlsx in StateSpreadsheets, set switch = 1 to use;
%destroyer( totcalls= 55,
            stlist = AK AL AR AZ CA CO CT DC DE FL
                    GA GU HI IA ID IL IN KS KY LA

```

```

MA MD ME MI MN MO MS MT NC ND
NE NH NJ NM NV NY OH OK OR PA
PR RI SC SD TN TX UT VA VI VT
WA WI WV WY ALL ,

ext = xlsx);

```

At the bottom of the program, the macro routine destroyer is used to eliminate any HTML files, as the Excel files have replaced them.

```

**** Deletes .html files because .xlsx were made;
%destroyer( totcalls=55,
    stlist = AK AL AR AZ CA CO CT DC DE FL
            GA GU HI IA ID IL IN KS KY LA
            MA MD ME MI MN MO MS MT NC ND
            NE NH NJ NM NV NY OH OK OR PA
            PR RI SC SD TN TX UT VA VI VT
            WA WI WV WY ALL ,
    ext = html);

```

POSSIBLE ALTERNATIVES TO DDE

We explored a number of alternatives to using DDE which requires the presence of Microsoft Excel on the system the program is being run on, specifically to perform the function of “saving as”. PROC EXPORT does not include the presentation controls our Excel outputs require. Similarly, CSV (either ODS CSV or put statements) does not include presentation controls, and while it would open in Excel, would not BE native Excel. SAS’s TableEditor tagset is a possibility, as it functions in a similar way to DDE. If any readers have been successful with this, let us know! A fourth option is to use a Visual Basic script. We were able to execute the script successfully for a single file, but were unable to enable macro processing which is essential to our programs. We welcome discussing the various possibilities with others!

CONCLUSION

With some research and a little creativity, you can easily incorporate external processes within your SAS programs in conjunction with macro coding to save a lot of time and reduce errors in your processing! While DDE is tricky to pick up, it allows more control and functionality. It is very useful for repetitive processing to automate editing and formatting in Excel, and to save HTML and XML output as native Excel. Adding the ability to use DDE in conjunction with macro processing makes DDE an essential part of our programming toolbox.

ACKNOWLEDGEMENTS

We are grateful for the contributions our colleague Christianna Williams makes to our project and our SAS programming knowledge, and for the astute technical assistance of Rick Andrews of the Centers for Medicare and Medicaid Services.

REFERENCES AND RECOMMENDED READING

Parker, Chevell. “The Perfect Marriage: The SAS® Output Delivery System (ODS) and Microsoft Office.” Proceedings of SAS Global Forum 2011 Conference. April 2011.

Vyverman, Koen. “Creating Custom Excel Workbooks from Base SAS® with Dynamic Data Exchange: A Complete Walkthrough”. Proceedings of SAS Users Group International Conference. April 2002.



CONTACT INFORMATION

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

Calvin Guan calvin_guan@abtassoc.com

Louise Hadden louise_hadden@abtassoc.com

Code samples are available upon request.

SAS and all other SAS Institute Inc. product or service names are registered trademarks or trademarks of SAS Institute Inc. in the USA and other countries. ® indicates USA registration.

Other brand and product names are trademarks of their respective companies.