

Using DDE with Microsoft Excel and SAS to Collect Data from Hundreds of Users

Russell Denslow, Sodexho Marriott Services, Orlando, FL
Yan Li, Sodexho Marriott Services, Orlando, FL

ABSTRACT

A process is demonstrated in this paper to automatically collect data from nearly 900 users using DDE with Microsoft Excel and SAS. SAS will populate (using DDE) an existing file "template" (Excel spreadsheet) with historical data, name and save it to the appropriate directory (based on reporting structure). SAS determines the hierarchical reporting structure from a "work.control" dataset and builds the corresponding directory structure (using %sysexec). These files can be distributed to the needed participants for their input and updating. Upon collection, these "updated" files will be stored in a similar "reporting" directory structure. SAS will then determine the contents of each directory (using %sysexec) and cycle the file names through a macro-loop. The contents of the file will be collected using DDE and appended to a SAS dataset.

INTRODUCTION

The Education Division of Sodexho Marriott Services is responsible for providing food and facility services to nearly 900 client sites (predominately universities and colleges) across the United States. Every year operational budgets need to be completed and "rolled-up" in a hierarchical reporting structure (DVP, RVP, and DM). The historical data from the clients are stored in SAS datasets. A Microsoft Excel worksheet with the historical data for each client site is provided for inputting the next year's preliminary budget (plan) numbers. Upon collection, these nearly 900 worksheets are "extracted" to produce SAS datasets for analysis. DDE is used to exchange data in a timely and accurate manner between the SAS system and Microsoft Excel producing the data ready for analysis.

DDE (Dynamic Data Exchange) is a method for exchanging information between Windows applications. In order for the exchange to occur, both applications need to be simultaneously running in Windows. One of them, the "client", must take the initiative to begin the process of exchanging data, and terminate the connection after the exchange is completed. In this example, SAS will be the "client".

Master or Template Worksheet

A master worksheet, "plan.xls", (Figure1) is created using Excel to incorporate all necessary variables for the budget process. In the master worksheet presented below, each location number represents one client site. Classification or hierarchical reporting structure variables include *DVP*, *RVP* and *DM*. Identification variables are *location* and *client*. Numeric data variables include *lastyr1-lastyr12*, *thisyr1-thisyr12* and *budget1-budget12/planyr1-planyr12*.

DVP	RVP	DM	LOCATION	CLIENT	DESC	PLAN1	PLAN2	PLAN3	PLAN4	PLAN5	PLAN6	PLAN7	PLAN8	PLAN9	PLAN10	PLAN11	PLAN12
53	85	626	2200001000	A	01 Sales	0	0	0	0	0	0	0	0	0	0	0	0
53	85	626	2200001000	A	02 Food	0	0	0	0	0	0	0	0	0	0	0	0
53	85	626	2200001000	A	03 Labor	0	0	0	0	0	0	0	0	0	0	0	0
53	85	626	2200001000	A	04 Control	0	0	0	0	0	0	0	0	0	0	0	0
53	85	626	2200001000	A	05 Non-ctrl	0	0	0	0	0	0	0	0	0	0	0	0
53	85	626	2200001000	A	06 OPC	0	0	0	0	0	0	0	0	0	0	0	0
53	85	626	2200001000	A	08 Safety	0	0	0	0	0	0	0	0	0	0	0	0
53	85	626	2200001000	A	19EP w/safety	0	0	0	0	0	0	0	0	0	0	0	0
53	85	626	2200001000	A	20EP w/safety	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1

Data Preparation

The following code (figure 2) generates a sample SAS dataset. This dataset

(work.control) has the same data structure or layout as the master worksheet "plan.xls".

```
data work.control;
  informat dvp rvp dm 3. location 10. client $1. i 2. desc $16. lastyr1-lastyr12 4.1;
  input dvp rvp dm location client i desc 35-50 lastyr1-lastyr12;
cards;
069 069 477 2200001000 A 1 01.Sales 31.8 32.9 31.6 32.2 32.6 31.9 31.9 31.9 33.2 31.6 31.9 31.9
069 069 477 2200001000 A 2 02.Food 17.5 18.5 17.5 17.5 18.5 17.5 17.5 17.5 18.5 17.5 17.5 17.5
069 069 477 2200001000 A 3 03.Labor 8.5 8.5 8.5 8.5 8.5 8.5 8.5 8.5 8.5 8.5 8.5 8.5
069 069 477 2200001000 A 4 04.Control 2.3 2.3 2.2 2.4 2.2 2.3 2.3 2.3 2.4 2.2 2.3 2.3
069 069 477 2200001000 A 5 05.Non-ctrl 2.3 2.3 2.2 2.4 2.2 2.3 2.3 2.3 2.4 2.2 2.3 2.3
069 069 477 2200001000 A 6 06.OPC 1.2 1.3 1.2 1.4 1.2 1.3 1.3 1.3 1.4 1.2 1.3 1.3
069 069 477 2200001000 A 8 08.Safety 1.0 0.5 2.0 0.5 0.5 1.5 1.0 1.5 2.5 1.5 1.0 0.5
069 069 477 2200001000 A 19 19.EP w/o Safety 0.2 0.3 0.2 0.4 0.2 0.3 0.3 0.3 0.4 0.2 0.3 0.3
069 069 477 2200001000 A 20 20.EP w/Safety 1.2 1.3 1.2 1.4 1.2 1.3 1.3 1.3 1.4 1.2 1.3 1.3
051 061 324 2200002000 B 1 01.Sales 31.5 32.5 31.5 31.5 31.5 32.5 31.5 31.5 33.5 31.5 31.5 31.5
051 061 324 2200002000 B 2 02.Food 17.5 18.5 17.5 17.5 18.5 17.5 17.5 17.5 18.5 17.5 17.5 17.5
051 061 324 2200002000 B 3 03.Labor 8.5 8.5 8.5 8.5 8.5 8.5 8.5 8.5 8.5 8.5 8.5 8.5
051 061 324 2200002000 B 4 04.Control 2.5 1.5 2.5 2.5 2.5 2.5 1.5 2.5 1.5 2.5 2.5 2.5
051 061 324 2200002000 B 5 05.Non-Ctrl 0.5 0.5 1.5 0.5 0.5 0.5 0.5 1.5 0.5 1.5 0.5 0.5
051 061 324 2200002000 B 6 06.OPC 2.5 3.5 1.5 2.5 1.5 3.5 3.5 1.3 4.5 1.5 2.5 2.5
051 061 324 2200002000 B 8 08.Safety 1.0 0.5 2.0 0.5 0.5 1.5 1.0 1.5 2.5 1.5 1.0 0.5
051 061 324 2200002000 B 19 19.EP w/o Safety 1.0 1.0 1.0 1.1 1.0 1.0 1.1 1.0 1.0 1.1 1.1 1.1
051 061 324 2200002000 B 20 20.EP w/Safety 2.1 1.5 2.2 1.3 1.5 3.5 1.0 2.5 3.4 1.3 1.6 0.4
053 065 626 2200003000 C 1 01.Sales 61.5 62.5 61.5 61.5 61.5 62.5 61.5 61.5 62.5 61.5 61.5 61.5
053 065 626 2200003000 C 2 02.Food 37.5 38.5 37.5 37.5 38.5 37.5 37.5 37.5 38.5 37.5 37.5 37.5
053 065 626 2200003000 C 3 03.Labor 13.5 13.5 13.5 13.5 13.5 13.5 13.5 13.5 13.5 13.5 13.5 13.5
053 065 626 2200003000 C 4 04.Control 4.5 3.5 4.5 4.5 4.5 4.5 3.5 4.5 3.5 4.5 4.5 4.5
053 065 626 2200003000 C 5 05.Non-Ctrl 2.5 2.5 1.5 0.5 0.5 0.5 1.5 0.5 1.5 0.5 1.5 0.5
053 065 626 2200003000 C 6 06.OPC 4.5 4.5 4.5 5.5 4.5 6.5 6.5 4.5 6.5 4.5 5.5 5.5
053 065 626 2200003000 C 8 08.Safety 1.0 0.5 2.0 0.5 0.5 1.5 1.0 1.5 2.5 1.5 1.0 0.5
053 065 626 2200003000 C 19 19.EP w/o Safety 1.3 1.3 1.2 1.1 1.2 1.0 1.3 1.0 1.1 1.2 1.3 1.1
053 065 626 2200003000 C 20 20.EP w/Safety 0.1 0.1 0.2 0.1 0.1 0.1 0.2 0.2 0.1 0.1 0.1 0.1
;
run;
```

Figure 2

Directory Structure Generation

The necessary directory structures are generated using the SAS %SYSEXEC statement, which executes operating system commands. The %SYSEXEC statement can be used inside a macro or in open code. The syntax for this command is: %SYSEXEC <command>;. The command argument can be any of the operating system commands, for instance, "md" for "make directory". In this program %SYSEXEC is used to create a directory structure corresponding to the organization's hierarchical reporting relationship, (i.e. dvp or dvp\rvp or dvp\rvp\dm).

The following program (figure 4) generates "md" commands that are saved in a file, "pp.bat" (figure 3).

```
*file pp.bat;
Figure 3

*file pp1.bat;
md c:\windows\temp\51
md c:\windows\temp\53
md c:\windows\temp\69

*file pp2.bat;
md c:\windows\temp\51\61
md c:\windows\temp\53\65
md c:\windows\temp\69\69

*file pp3.bat;
md c:\windows\temp\51\61\324
md c:\windows\temp\53\65\626
md c:\windows\temp\69\69\477
```

```
00001 %let builddir=%; * Macro to build DVP, RVP and DM directory structure; Figure 4
00002 %let drive = c:; * This sets drive to build directory structure ;
00003 %let dir = \windows\temp; * This sets path to build directory structure ;
00004 %let var1 = dvp; * This sets first level of directory structure ;
00005 %let var2 = rvp; * This sets second level of directory structure;
00006 %let var3 = dm; * This sets third level of directory structure;
00007
00008 %macro builddir;
00009 %let vars = &var1 &var2 &var3 ;
00010 proc summary data=work.control;
00011 class &vars;
00012 output out=work.dir;
00013 run;
```

```

00014
00015 %sysexec md &drive.&dir;
00016
00017 data _null_;
00018     retain b (-1);
00019     set work.dir;
00020     *----- Script to create 1st Part of directory structure ---;
00021     if _type_ = 4 then
00022         do;
00023             file 'pp1.bat';
00024             put "md &drive.&dir.\" &var1 ;
00025         end;
00026
00027     *----- Script to create 2nd Part of directory structure ---;
00028     if _type_ = 6 then
00029         do;
00030             file 'pp2.bat';
00031             put "md &drive.&dir.\" &var1 +b '\\' &var2 ;
00032         end;
00033
00034     *----- Script to create 3rd Part of directory structure ---;
00035     if _type_ = 7 then
00036         do;
00037             file 'pp3.bat';
00038             put "md &drive.&dir.\" &var1 +b '\\' &var2 +b '\\' &var3 ;
00039         end;
00040     run;
00041
00042 %sysexec copy pp1.bat + pp2.bat + pp3.bat pp.bat;
00043 %sysexec pp.bat;
00044 %mend;&bulddir.bulddir;

```

Figure 4 (continued)

Populating the Excel Master Worksheet

This part of the program will do the following steps:

- 1) start Excel and open the master worksheet (plan.xls);
- 2) populate the master worksheet with data from the SAS dataset (work.control) for each location;
- 3) save and close the worksheet for each location/site in the directory corresponding to the reporting hierarchy.

The file name assigned to each worksheet will be the corresponding location number. (for example, location number 225555333 would be "5555333.xls"). The populating and saving steps (2 & 3) are automatically repeated for each location number using a SAS macro "loopit".

```

00045 options noxwait noxsync obs=max;
00046 proc sort data=work.control;
00047 by location desc; run;
00048
00049 Data _null_;
00050     file 'samples\loc.txt';
00051     set work.control;
00052     by location;
00053     if first.location then DO;
00055     put '%loopit(loc="'&location + (-1)'"');';
00056     END;
00057 run;

```

Figure 6

The following code (figure 6) generates the file "loc.txt" (figure 5) to later execute the macro "loopit" for every location.

```

*file loc.txt;
%loopit(loc='2200001000');
%loopit(loc='2200002000');
%loopit(loc='2200003000');

```

Figure 5

NOXSYNC indicates that the operating system command should be executed asynchronously with the SAS session.

NOXWAIT instructs the SAS System to automatically close the spawned prompt window after the completion of a specified command, eliminating the need to type EXIT.

In the first part of the following code (line 58, figure 7), Excel is started. The "X" allows SAS to pass commands to the operating system (OS) of Windows. The path statement "D:\Progra~1\Micros~1\Office\EXCEL" specifies where the Excel executable is located (to work consistently, the DOS 8 character naming convention is strongly recommended).

The second part (lines 60-62, figure 7) SAS waits for 10 seconds so that Excel can be completely initialized. The sleep period may be reduced or increased to suit the processing speed of the particular system.

The third part (lines 66-82, figure 7) opens the master Excel worksheet "plan.xls" and defines the fileref *planyr*, path, file name, sheet name, and the range of cells to be populated.

The NOTAB option in the filename statement is used to deal with the embedded blanks in the Excel data "plan.xls. '09'X in the PUT statement preserves the blank spaces needed in the data. The CALL SYMPUT routine creates a macro variable &DVP whose value is the DVP number from the SAS dataset (*work.control*). The routine also creates &RVP and &DM. The PUT statement inserts the data to the specified Microsoft Excel cells.

The fourth part (line 86-94, figure 7) instructs SAS to save the populated worksheet to the directories determined by the macro variables (&DVP, &RVP, and &DM) and close the worksheet.

The macro "LOOPIT" (line 65-94, figure 7) is run for each location until all are processed by including the "loc.txt" file (line 95, figure 7).

```

00058 X "D:\Progra~1\Micros~1\Office\EXCEL";
00059
00060 DATA _NULL_;
00061 X = SLEEP(10);
00062 RUN;
00063
00064 FILENAME commands DDE 'EXCEL|SYSTEM';
00065 %MACRO LOOPIT (loc=loc);
00066 data _null_;
00067     file commands;
00068     PUT '[OPEN("D:\Fsm$fast\Samples\plan.xls")]';
00069 run;
00070 filename planyr dde "Excel|[plan.xls]Sheet1!R3C1:R11C18" notab;
00071 data work.dossout;
00072     retain b (-1) t '09'x;
00073     set work.control;
00074     file planyr;
00075     call symput ('dvp',compress(dvp));
00076     call symput ('rvp',compress(rvp));
00077     call symput ('dm',compress(dm));
00078     call symput ('name',substr(&loc,3,5)||substr(&loc,8,3));
00079     put dvp t+b rvp t+b dm t+b location t+b client t+b desc t+b
00080         lastyr1 t+b lastyr2 t+b lastyr3 t+b lastyr4 t+b lastyr5 t+b lastyr6 t+b
00081         lastyr7 t+b lastyr8 t+b lastyr9 t+b lastyr10 t+b lastyr11 t+b lastyr12;
00082 run;
00083
00084 /*%let path=c:\windows\temp\&dvp.\&name..xls; */
00085 /*%let path=c:\windows\temp\&dvp.\&rvp.\&name..xls;*/
00086 %let path=c:\windows\temp\&dvp.\&rvp.\&dm.\&name..xls;
00087 %put Worksheet is &path. ;
00088 data _null_;
00089     file commands;
00090     put "[Save.as("&path.")]";
00091     put "[File.Close()]";
00092 run;
00093 %put Worksheet has been saved as &path.;
00094 %MEND;
00095 %include 'samples\loc.txt';
00096
00097 DATA _NULL_;
00098 FILE commands;
00099 PUT '[QUIT]';

```

Figure 7

00100 RUN;

Figure 7 (continued)

Unit Plans/Budgets Roll-Up

The following program is used to automatically collect the updated worksheets and append the data to a SAS dataset.

The first part of the code (figure 9) sets several macro variables. These are set to assist with error checking collection of data, and setting options. Lines 8-16, figure 9, of the code search for file names by indexing "XLS" or "xls" in "C:\windows\temp\dir.txt" (created by line 6). The filenames are held for the macro *capture(wk)*. A file, "C:\windows\temps\pp.dat", is created

(lines 18-23, figure 9) to contain the multiple calls (figure 8) for the macro *capture(wk)* by the filenames found in the directory.

```
file pp.dat. Figure 8
%capture(wk=00001000.xls );
%capture(wk=00002000.xls );
%capture(wk=00003000.xls );
```

The second part (line 36-62, figure 9) will use the macro *capture(wk)* to collect and extract the needed data from the files in the directory "c:\windows\temp\&budg". It then will close Excel after completion.

```
00001 %let fin = bob;          *bob deb tom1 tom2 tom3 don mike mbu jim bill;
00002 %let budg = bud&fin;
00003 %let drive = c.;
00004 %let dir = \windows\temp;
00005 options noxwait noxsync obs=max;
00006 %sysexec dir &drive.&dir.>&drive.&dir.\dir.txt;run;
00007
00008 data work.dirsd2(drop=dirstuff);
00009   infile "&drive.&dir.\dir.txt" lrecl=132 pad missover;
00010   input @01 dirstuff $char132. @;
00011   if index(dirstuff,'XLS') or index(dirstuff,'xls') then input @40 filename $12.;
00012                                     * @40 windows NT;
00013   if index(dirstuff,'XLS') or index(dirstuff,'xls') then input @45 filename $12.;
00014                                     * @45 win 95,98;
00015   if filename eq ' ' then delete;
00016 run;
00017
00018 data _null_;
00019   file "&drive.&dir.\pp.dat";
00020   set work.dirsd2;
00021   x=filename;
00022   put '%' 'capture(wk=' x ');';
00023 run;
00024
00025 proc datasets lib=work;
00026   delete &budg;
00027   quit;
00028 run;
00029
00030 X "d:\Progra~1\Micros~1\Office\Excel";
00031 data _null_;
00032   x= sleep(10);
00033 run;
00034
00035 FILENAME cmds DDE 'EXCEL|SYSTEM';
00036 %macro capture(wk);
00037 %put Worksheet is &drive.&dir.\&wk.;
00038   data _null_;
00039     file cmds;
00040     put "[open("&drive.&dir.\&wk."")]";
00041   run;
00042
00043 filename planyr dde "Excel|[&wk.]Sheet1!R3C1:R12C18";
00044 data planyr;
00045   infile planyr dlm='09'x notab dsd missover;
00046   informat location $char10. client $char32. desc $char25.;
```

Figure 9

```

00047      input client location desc
00048          budget1 budget2 budget3  budget4  budget5  budget6
00049          budget7 budget8 budget9 budget10 budget11 budget12
00050      ;
00051      if location = ' ' then delete;
00052      format i 2.;
00053      i=substr(desc,1,2);
00054  run;
00055  proc sort data=work.planyr;by location desc;run;
00056  proc append base=work.&budg data=work.planyr;run;
00057
00058  data _null_;
00059      file cmds;
00060      put '[File.Close()]';
00061  run;
00062 %mend;
00063
00064 %inc "&drive.&dir\pp.dat";          /*%capture(wk=00001001.xls)*/
00065
00066 data _null_;
00067      file cmds;
00068      put '[Quit()]';
00069 run;

```

Figure 9 (continued)

CONCLUSION

The process presented in this paper can be used to automatically create a SAS dataset from multiple Excel spreadsheets, or to generate multiple spreadsheets from a SAS dataset. DDE is an efficient method of exchanging data in a timely and accurate manner between SAS and Excel.

ACKNOWLEDGEMENTS

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 registered trademarks or trademarks of their respective companies.

Sodexo Marriott Services is the largest provider of food and facilities management in North America, with \$4.5 billion in annual sales. Sodexo Marriott offers a variety of innovative outsourcing solutions, including food service, housekeeping, groundskeeping, plant operations and maintenance, asset and materials management, and laundry services to corporations, health care facilities, schools, universities and colleges, and remote sites. Headquartered in Gaithersburg, MD, the company has 103,000 employees at 5,000 locations across the U.S. and Canada.

This work was made possible by the unconditional support of Mr. Mohamood Bhatia, Senior Vice President of Finance, Education Division of Sodexo Marriott Services. Additionally, the authors would like to express their deepest gratitude to Mr. Nick Thaler, Senior Director of Decision Support Systems, and Karen Wyland Senior Manager of Decision Support Systems, for their invaluable input and guidance, code, and technical support.

AUTHORS CONTACT INFORMATION

Russell E. Denslow
Senior Finance Manager

Yan Li
Finance Analyst

Sodexo Marriott Services
283 N. North Lake Boulevard, Suite 260
Altamonte Spring, FL 32701

Telephone: 407-339-3230
Fax: 407-260-2305
E-mail: RDenslow@sodexhomarriott.com
YLi@sodexhomarriott.com