Using DDE and SAS/Macro for Automated Excel Report Consolidation and Generation

Mengxi Li, Sandra Archer, Russell Denslow
Sodexho Campus Services, Orlando, FL

Abstract

Each week, the Sodexho Campus Services finance team receives 80 Excel files containing weekly financial reports from its operating units via email. System commands are used to read the names of the files and generate macro calls. Since the files are all in the same layout, DDE (Dynamic Data Exchange) is used to pull the data from each Excel file into SAS data sets. Base SAS code is used to consolidate the data. DDE is used again to output the consolidated data into standard Excel templates for reporting purposes. SAS code automatically distributes the reports via Outlook. This paper describes the process, including some example code. SAS version 8 is used, along with Excel and Outlook 2000 in a Windows 2000 operating system.

Introduction

Sodexho is the leading provider of food and facilities management services in the U.S. and Canada, offering innovative outsourcing solutions in food service, housekeeping, grounds keeping, plant operations and maintenance, asset and materials management, and laundry services to corporations, health care and long term care facilities, retirement centers, schools, military, college campuses and remote sites. The Campus Services Division is responsible for providing campus dining, facility management, concessions and arena management services to universities, colleges and independent schools across the United States. The purpose of this project is to collect the weekly operating financial statements from 80 district managers in the Campus Services division.

Each district manager has an Excel workbook that is used to input weekly sales, operating costs, and other weekly financial results. After entering the data for all operational sites, the Excel workbook will automatically generate a single Excel sheet “extract”, which is emailed by the district manager to our team. Since each Excel worksheet received by our team is in the same format, we can use DDE to pull the data in to SAS. After data consolidation and cleaning, DDE is again used to generate reports.

Receiving the Excel Files

When the files are received via email, they are saved in one directory for that week. We now have a directory containing all Excel extract files from each district manager.

Generating Macro Calls

We have a macro “loopit” containing DDE code that needs to be run for every Excel file. Since the file names may change from week to week, we want to avoid re-typing the list of macro calls each week. The following code will read the directory listing of Excel files and generate the list of macro calls, one for each Excel file.

Figure 1. Generating Macro Calls

```
1   %let week = 1;
2   %let flashpd = 9;
3   %let path = C:\Sesug\PD&flashpd._WK&week.\ ;
4   %sysexec md "&path.sas";
5   libname flash "&path.sas";

6  options noxwait noxsync obs=max;
7  %sysexec dir &path.>&path.dir.txt;

8  data _null_;
9            X = sleep(3);
10  run;

11  data work.fddir (drop=dirstuff);
12      infile "&path.dir.txt" lrecl=132 pad missover;
13       input @01 dirstuff $char132. @;
14       if index(dirstuff,'XLS') or index(dirstuff,'xls') then input @40
model $8.;
15  run;

16  data _null_;
17           file "&path.dm.txt";
18           set work.fddir;
19           put '%' 'loopit('model+(-1)');';
20           END;
21  run;
```

Line 1 and 2 are the only two lines that the programmer needs to update each week. Line 1 is the week number and line 2 is the period(month) number. The **%let** statement creates a global macro variable that can be used at any time during this SAS session. The variables created, “week” and “flashpd”, are used in line 3 to assign a value to “path”. This folder already exists and contains the Excel extracts sent by the district managers. Line 4 uses the macro statement **%SYSEXEC**. This macro statement will immediately execute the operating system command that follows. In this case, we use the system command **md** (make directory) to create a subfolder named “sas”. Line 9 uses the system command **dir** to generate a text file, “dir.txt”, containing the directory listing. (See Figure 2) You can see SAS put into **sleep** mode for 3 seconds in lines 8-10 for the purpose of giving the system time to generate this text file before running the next lines of code. This number may need to be increased to suit your system. Lines 11-15 demonstrate the use of input and indexing statements to pull each file name out of “dir.txt”. The result is a SAS data set "work.fddir" containing the character variable "model", the list of file names from "dir.txt". Lines 16-21 create a text file, "DM.txt". "DM.txt" contains a list of macro calls to the macro "loopit". The list of macro calls is created using the put statement, concatenating the "% loopit" text with the names of Excel files. See Figure 3 for the contents of "DM.txt".

Figure 2. Contents of "dir.txt"

```
 Volume in drive C has no label.
 Volume Serial Number is 3821-FEF7

 Directory of C:\Sesug\PD9_WK1\ ;

05/02/2003  11:22a     <DIR>          .
05/02/2003  11:22a     <DIR>          ..
04/30/2003  04:38p           701,952 01715000.xls
04/30/2003  12:02p           968,192 10150050.xls
04/30/2003  02:29a           224,768 10230001.xls
04/30/2003  06:07p           402,432 10380050.xls
04/30/2003  03:32p           900,096 10435002.xls
04/28/2003  06:35p           213,504 10635000.xls
04/30/2003  08:32a           309,760 10690001.xls
04/30/2003  06:57p           744,448 10730050.xls
…
```

Figure 3. Contents of "DM.txt"

```
%loopit(01715000);
%loopit(10150050);
%loopit(10230001);
%loopit(10380050);
%loopit(10435002);
%loopit(10635000);
%loopit(10690001);
%loopit(10730050);
…
```

Now that we have macro calls written, we can have the macro "loopit" run once for each file name.

Using DDE to Read the Excel Files

DDE (Dynamic Data Exchange) allows the dynamic exchange of data between Windows applications. In a client/server relationship, the SAS System acts a client by requesting data, sending data, or sending commands to the server application. In the example in this paper, Excel 2000 is the server application, although any application that supports DDE as a server can communicate with the SAS System. To use DDE is SAS, issue the file name statement:

FILENAME *filref* DDE 'DDE-triplet' <DDE-options>

The DDE-triplet argument refers to the DDE external file in the following form:

'application-name|topic!item'

(SAS Institute, 1999)

In this project, DDE is used to issue system commands, such as opening a specified Excel worksheet, to read data from an Excel sheet, and to put data into an Excel sheet. The syntax above will be referred to in the examples.

Figure 4. Beginning DDE Session

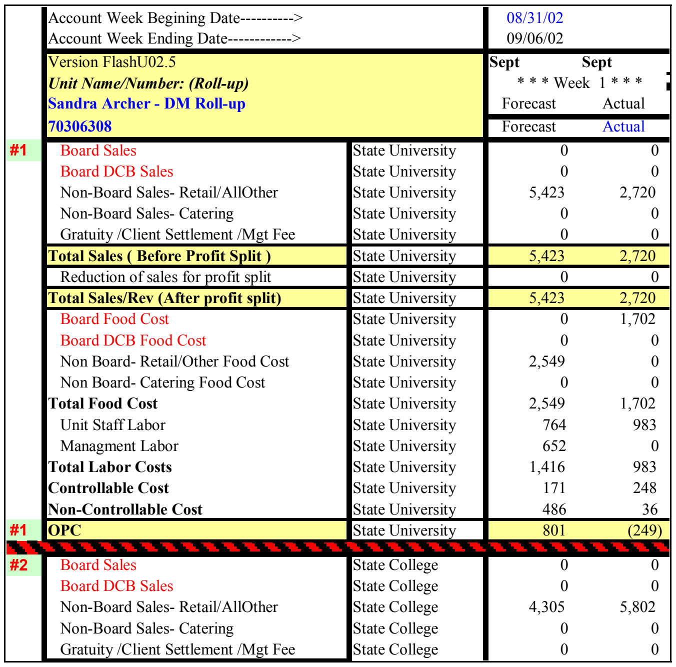
```
22  %include "C:\Sesug\flash_var.sas";

23  X '"C:\Program Files\Microsoft Office\Office\EXCEL.exe"';

24  DATA _NULL_;
25          X = SLEEP(3);
26  RUN;

27  FILENAME commands DDE 'EXCEL|SYSTEM';
```

Line 22 includes a piece of SAS code "flash_var.sas". "Flash_var.sas" contains a series of %Let statements. The %Let statements names lists of variables that can then be referenced later at any time in this SAS session without having to re-type the list of variables. This is helpful if the list of variables may change from week to week. For example, the first line of code in "Flash_var.sas" is "%Let FOODVAR = pd1_wk1_fc pd1_wk1_ac pd1_wk2_fc pd1_wk2_fc..."

Line 23 uses the **X** statement to submit a command to the operating system, and open a DDE server. It is here that Excel is established as a DDE server. The path points to where Excel.exe is located on your hard drive. Excel.exe is launched, opening Excel. Please note that the XWAIT and XSYNC options must be turned off (Figure 1, Line 6). **NOWAIT** allows the command processor to automatically return to the SAS system after the command is executed. **NOSYNC** allows you to return to the SAS system without closing the process generated by the X command. You can see sleep mode used again in lines 24 – 26, giving the system 3 seconds to open Excel before the next lines of code run. The **filename** statement in line 27 is used to establish a DDE link to the Excel application. This will allow us to later issue commands to Excel, using the **fileref** "commands". In the **DDE triplet**, the **application-name** is Excel, the **topic** is SYSTEM and the item is not specified.

Each file received from the district managers is in the same layout. The columns represent the forecast and actual weekly time periods, from 1 to 52. The client accounts are stacked on top of each other with the rows representing the different financial data and statistics that is collected each client account. See Figure 5 for an idea of the Excel files' layout.

Figure 5. Example of Excel Files' Layout



	Account Week Begining Date---------->		08/31/02	
	Account Week Ending Date------------>		09/06/02	
	Version FlashU02.5		Sept	Sept
	Unit Name/Number: (Roll-up)		* * * Week	1 * * *
	Sandra Archer - DM Roll-up		Forecast	Actual
	70306308		Forecast	Actual
#1	Board Sales	State University	0	0
	Board DCB Sales	State University	0	0
	Non-Board Sales- Retail/AllOther	State University	5,423	2,720
	Non-Board Sales- Catering	State University	0	0
	Gratuity /Client Settlement /Mgt Fee	State University	0	0
	Total Sales (Before Profit Split)	State University	5,423	2,720
	Reduction of sales for profit split	State University	0	0
	Total Sales/Rev (After profit split)	State University	5,423	2,720
	Board Food Cost	State University	0	1,702
	Board DCB Food Cost	State University	0	0
	Non Board- Retail/Other Food Cost	State University	2,549	0
	Non Board- Catering Food Cost	State University	0	0
	Total Food Cost	State University	2,549	1,702
	Unit Staff Labor	State University	764	983
	Managment Labor	State University	652	0
	Total Labor Costs	State University	1,416	983
	Controllable Cost	State University	171	248
	Non-Controllable Cost	State University	486	36
#1	**OPC**	State University	801	(249)
#2	Board Sales	State College	0	0
	Board DCB Sales	State College	0	0
	Non-Board Sales- Retail/AllOther	State College	4,305	5,802
	Non-Board Sales- Catering	State College	0	0
	Gratuity /Client Settlement /Mgt Fee	State College	0	0

We are able to employ DDE within a macro because the files received are in the same format.

Figure 6. DDE Macro

```
28  %macro loopit(model);
29  data _null_;
30      file commands;
31          put "[open(""&path.&model..xls"")]";
32  run;

33  filename flash1 dde
    "Excel|[&model..xls]PL!R55C2:R1440C216";

34  data work.food;
35          infile   flash1 missover notab   LRECL=10000
                     dlm='09'x dsd;
36          informat desc client $char50. &FOODVAR comma10.0
                        location client $char20
37          input    desc $ client $ &FOODVAR
                        location $  client $;
38  run;

39  data food;
40          set food;
41          where desc in("Total Sales/Rev (After profit split)"
                              "OPC");
42          source="FD-&model";
43  run;

44  filename dm dde "Excel|[&model..xls]PL!R5C2:R5C2";

45  data dm;
46          infile   dm missover notab LRECL=10000
                    dlm='09'x dsd;
47          informat district $char50.;
48          input    district $;
49  run;

50  data fd&model.;
51          retain district;
52          set dm food;
53          retain district2;
54          if district^='' then district2=district;
55          if district='' then district=district2:
56           if desc="" then delete;
57           drop district2;
58  run;

59  data _null_;
60          file commands;
61          put "[File.Close()]";
62  run;

63 %mend;
```

Line 28 begins the macro "loopit", which will receive one macro variable, "model". Recall that the macro calls have already been created in a file "DM.txt", with the macro variable "model" as the names of the Excel files received. (Figure 3)

Line 29 – 32 uses the DDE link established in line 27 and a **put** statement to issue a command to the server application (Excel). The command issued, **open**, will open the Excel file (located in the path specified in line 3) with the file name that is passed to this macro. Other commands that Excel will accept from SAS include any Excel macro commands, including save or quit. Note that the Excel system must be open in order for SAS to pass commands to it. Likewise, the Excel file must be open in order for a data exchange between Excel and SAS to take place using DDE.

Once the Excel file is opened, line 33 will establish a DDE link to the specified range in Excel. In the **DDE triplet**, the **application-name** is Excel, the **topic** is the file "&model" with the tab name "PL" and the **item** is the range of data from Row55, Column 2 to Row 1440, Column 216. This link is named "flash1".

The data step in lines 34 – 38 will read the data in the specified range (flash1) into the SAS data set "work.food" by means of an **infile** statement. The options on the infile statement are for the following purposes: The *missover* option specifies that SAS should continue to read in a record, even if some value are missing. The *notab* option prevents SAS from converting tabs in Excel to blanks. The *dlm = '09x"* specifies that the file is tab delimited. This is used as the row separations in Excel are interpreted as tabs. The *dsd* option specifies that two delimiters represent a missing value. The *LRCL =* option specifies the record length (in bytes). The infile statement reads in the variables listed, with &FOODVAR representing the list of variables that was created by a %let statement by including the code in figure 4, line 22.

The data step in lines 39 – 43 simply filter the data set to only those records that we want to keep and adds a variable "source" that includes the file name. This gives us the ability to assign a source file to every record in our data set.

In line 44, another DDE link is established for the data range Row 5, Column 2. This cell in the spreadsheet contains the name of the district manager who turned in the file to us. Lines 45-49 use this DDE link to read in a simple data set containing only one column and one observation: the value of the cell containing the person's name. The data step in lines 50 – 58 creates a SAS data set named "fd&model" where &model is the name of the

Excel file source data. The **retain** statement and reassignment of variables are used to clean up the data and copy the name of the district manager to every record.

Finally, in lines 59 - 62, you can see that the file name "commands" (established in figure 4, line 27) allows us to use the **put** statement to send the **File.Close** command to Excel. Excel will then close the file that was opened in line 31. The Excel system will remain open in the background. The %mend statement in line 63 ends the macro "loopit".

Figure 7. Calling the Macro

```
64  %include "&path.dm.txt";

65  data _null_;
66          file "&path.dmset.txt";
67          set work.fddir;
68          put 'fd' model+(-1);
69  run;

70  data fdrollup2;
71          set %inc "&path.dmset.txt";
72  run;
```

Figure 7 demonstrates that we can simply use a **%include** statement to include the macro calls contained in "dm.txt" (Figure 3). The macro will run once for each Excel file and create a SAS data set by that name. Lines 65 - 69 use a similar trick to create a file "dmset.txt" that contains a list of the SAS data set names. These individual SAS data sets are then set together in lines 70 – 72.

Once in SAS, the data is validated and cleaned by methods not covered by this paper. Then, it is summarized by organizational hierarchy and prepared for reporting. The output data set is called "weeklysum".

Using DDE for Excel Report Generation

In order to use DDE to generate the reports with the same format for each level of hierarchy in the company, we start with an Excel report template. It is called "Report_Template.xls". In this template, we also used the "Conditional Formatting" to pre-format the output, using the bold font to distinguish the levels of hierarchy.

Figure 8. DDE Export Macro

```
176        %sysexec md "&path.Report";

177        options noxwait noxsync obs=max;
178        X '"C:\Program Files\Microsoft
               Office\Office\EXCEL.exe"';

179        DATA _NULL_;
180        X = SLEEP(3);
181        RUN;

182        FILENAME commands DDE 'EXCEL|SYSTEM';

183        %macro report(hier,name,outname);
184        data _null_;
185            file commands;
186            PUT '[OPEN("C:\Sesug\Report_Template.xls")]';
187        run;
188        FILENAME report DDE
           "Excel|[report_template.xls]sheet1!R7C1:R150C15"
           NOTAB LRECL=100000;

189        DATA work.report;
190            retain b (-1) t '09'x;
191            set weeklysum;
192         where &hier.=&name.;
193            file report;
194            put  vice_president t+b
195            region_manager t+b
196            district_manager t+b
197            location t+b
198            client  t+b
199            sales_week_1_actual      t+b
200            profit_week_1_actual        t+b
201            sales_week_2_forecast    t+b
202            profit_week_2_forecast      t+b
203            sales_week_3_forecast    t+b
204            profit_week_3_forecast      t+b
205            sales_week_4_forecast    t+b
206            profit_week_4_forecast      t+b
207            sales_september_forecast t+b
208            profit_september_forecast   t+b;
209        run;

210        data work.title;
211        title="Sales & Profit Report - September, 2003";
212        run;

213        FILENAME title DDE
           "Excel|[report_template.xls]sheet1!R3C3:R3C3" notab;

214        data work.reporttitle;
215            retain b (-1) t '09's;
216            set WORK.title;
217            file title;
218            put title;
219        run;

220        data _null_;
221            file commands;
222            put
           "[Save.as(""&Path.Report\&hier._Report_&outname..xls
           "")]";
223         put "[Close]";
224        run;

225        %mend;
```

Line 176 uses the system command **md** (make directory) to create a subfolder named "Report". Line 177 – Line 182 is to begin DDE session (Please refer to Figure 4. Beginning DDE Session).

Line 183 begins the macro "report", which will produce reports for each level of hierarchy (vice president, region manager and district manager). It receives three macro variables: "hier" is the level of hierarchy; "name" is used in the Where statement later to generate a personalized report; "outname" is used to name the output file. Line 184 – Line 187 opens the Excel report template "Report_Template.xls", which is already built before the DDE export. Line 188 establishes the range of output in the Excel template from Row7, Column 1 to Row 150, Column 15.

Line 189-Line 209 will read the data from SAS dataset "weeklysum" and output the data to the specified range (report) in the Excel template. Line 190 the **retain** statement assigns a backspace to “b” and a tab to “t”. When the **Put** statement is used, the tab separates the Excel columns, while the backspace counteracts the extra space that Excel may insert. Line 192 uses the **Where** statement to generate the personalized report. Line 194-Line 208 uses the **Put** statement to tell SAS to export the variables in work.report in that order, which is the order we set up in the template. Line 210-Line 219 use the same method (specify range and put SAS data to the template) to output the title of the report.

Line 220-Line 224 saves the file under the “Report" folder with the name specified in the macro call, and then closes the active worksheet. The Excel system will remain open in the background. The %mend statement in line 225 ends the macro “report”.

Figure 9. Calling the Macro “Report”

```
237  *Division Report;
238  %report(Div,"DIVISION",Division);

239  *Vice President Report;
240  %report(VP,"HAMILTON JONES",HJones);
241  %report(VP,"SANDRA ARCHER",SArcher);

242  *Region Manager Report;
243  %report(RM,"COREY GORDON",CGordon);
244  %report(RM,"MENGXI LI",MLi);
245  %report(RM,"PRESTON OLINGER",POlinger);
246  %report(RM,"SAMANTHA TYLER",STyler);

247  *District Manager Report;
248  %report(DM,"JIAN SHEN",JShen);
249  %report(DM,"JIM SMITH",JSmith);
250  %report(DM,"JOHN JOE",JJoe);
251  %report(DM,"LORI CARR",LCarr);
252  %report(DM,"MINDY SMITH",MSmith);
253  %report(DM,"PETER DENIS",PDenis);
254  %report(DM,"SANDY BRAD",SBrad);
255  %report(DM,"WEI LI",WLi);
```

The code in Figure 9 will produce reports for each level of hierarchy with the name structure “Hierarchy_Report_First&Lastname.xls”. In total 15 reports are generated. The reports are saved with the name of the person who is to receive it.

Using SAS to Automatically Send out Email Through Outlook

Figure 10. Sending Email Through Outlook

```
264  %Macro mail (sendee, attach);

265    filename mail email "Report";
266    data mail;
267          file mail;
268          put '!EM_TO!' &sendee.;
269          put "!EM_SUBJECT! Sales & Profit Report -
                                      September 2003";
270          put "";
271          put "Attached file is the Sales & Profit Report for
                    September 2003.";
272          put "!EM_ATTACH! &path.\report\&attach." ;
273          put '!EM_SEND!';
274          put '!EM_NEWMSG!';
275          put '!EM_ABORT!';
276     run;

277  %Mend;
```

Sending email within SAS through Outlook is extremely efficient and timesaving when distributing massive email messages to a group of people with attachments.

An e-mail program that supports MAPI, VIM or SMTP is required to use the SAS System's e-mail support. Although you can use the SAS System to send messages, you must use your e-mail program to view or read messages.

Line 264 begins the macro “mail”, which will send out emails to each level of hierarchy with the reports we create for them.

Line 265 initiates SAS with the email function. The **EMAIL** option is used as a keyword indicating the use of electronic mail.

Line 268 replaces the current primary recipient addresses. If a single address contains more than one word, you must enclose that address in quotes. If you want to specify more than one address, you must enclose each address in quotes and the group of addresses in parentheses.

Line 269 puts a subject in the subject line of the email. Line 271 generates the text of the message. Line 272 attaches the Excel report. To specify more than one file, you must enclose each filename in quotes and the group of filenames in parentheses.

Line 273 sends the message with the current attributes. Line 274 clears all attributes of the current message that were set using the **Put** statement. Line 275 aborts the current message.

Figure 11. Calling the Macro “mail”

```
279    %mail ("Jones, Hamilton",VP_Report_JJones.xls);
280    %mail ("Archer, Sandra",VP_Report_SArcher.xls);

281    %mail ("Gordon, Corey",RM_Report_CGordon.xls);
282    %mail ("Li, Mengxi",RM_Report_MLi.xls);
283    %mail ("Olinger, Preston",RM_Report_POlinger.xls);
284    %mail ("Tyler, Samantha",RM_Report_STyler.xls);

285    %mail ("Shen, Jian",DM_Report_JShen.xls);
286    %mail ("Smith, Jim",DM_Report_JSmith.xls);
287    %mail ("Joe, John",DM_Report_JJoe.xls);
288    %mail ("Carr, Lori",DM_Report_LCarr.xls);
289    %mail ("Smith, Mindy",DM_Report_MSmith.xls);
290    %mail ("Denis, Peter",DM_Report_PDenis.xls);
291    %mail ("Brad, Sandy",DM_Report_SBrad.xls);
292    %mail ("Li, Wei",DM_Report_WLi.xls);
```

The code in Figure 11 will call the macro “mail” and send an email to each person with his or her personalized report as an attachment.

Note: Microsoft Outlook XP and Outlook 2000 SR2 generate an extra prompt when you attempt to send email automatically from the SAS system:

“A program is trying to automatically send e-mail on your behalf. Do you want to allow this? If this is unexpected, it may be a virus and you should choose "No".”

You then have a choice of three buttons: Yes, No, and Help. To send the mail, select the Yes button.

To avoid this prompt, take the following actions:

1. The administrator needs to change the security settings on the email server itself. Note that a qualified email administrator is required to do this.

2. Another possibility is to use SMTP email to accomplish the task because it talks directly to the SMTP email server. However, SMTP email has certain restrictions on attachments: only text files and zip files work in Version 8. Release 8.1 or later is also required.

Conclusion

Consolidating tens, hundreds, even thousands of Excel files, producing weekly, monthly, quarterly reports in the same format and sending them out to all levels of users in a short amount of time is not a dream any more with the powerful, interactive SAS product. The Macro, DDE and Automated Email we covered here are only the tip of the iceberg of SAS program. The more you explore, the more you can do.

References

SAS Institute Inc., *SAS Companion for the Microsoft Windows Environment, Version 8*, Cary, NC: SAS Institute Inc, 1999. Pp.562

Contact Information:

Russell.Denslow@sodexhoUSA.com

Sandra.Archer@sodexhoUSA.com

Mengxi.Li@sodexhoUSA.com