

With Excel running and the above spreadsheet open, a simple Data Step can be written to read the above data into a SAS Data Set. The DDE Triplet components for the above example are as follows:

Name of Executable File: EXCEL

Full Path and Name of spreadsheet file:
C:\COURSES\DATA\CLIN.XLS

Location of data (Row and Column coordinates):
R4C1:R17C8

Note how the two sets of coordinates that define the range of data are separated by a colon.

The SAS statements required to read the data into a SAS data set are as follows:

```

0001 INPUT PATNO SEX $ AGE DRUG $ SCORE1-SCORE3 TOTAL_SC ;
0002 RUN;

NOTE: The infile XFILE is:
      FILENAME=excel\courses\data\clin.xls\ref17c8,
      RECFM=W,LRECL=256

NOTE: 14 records were read from the infile XFILE.
      The minimum record length was 24.
      The maximum record length was 25.
NOTE: The data set WORK.CLIN has 14 observations and 8 variables.
NOTE: The DATA statement used 0.71 seconds.

00001 /* Assign a Fileref to the Excel Spreadsheet and specify that
00002 Dynamic Data Exchange is to be used */
00003
00004 FILENAME xfile DDE 'excel\courses\data\clin.xls\ref17c8';
00005 /* Create a Data Set called CLIN in the Work Library and read in
00006 data from the Excel Spreadsheet */
00007
00008 DATA WORK.CLIN;
00009 INFILE XFILE;
00010 INPUT PATNO SEX $ AGE DRUG $ SCORE1-SCORE3 TOTAL_SC ;
00011 RUN;
00012

```

Running the above code generates the following data set:

PATNO	SEX	AGE	DRUG	SCORE1	SCORE2	SCORE3	TOTAL_SC
1	M	56	CHG1	87	77	88	252
2	F	44	DFR4	45	23	67	135
4	F	34	GTR6	44	34	76	154
7	M	45	CHG1	68	76	79	223
8	M	43	GTR6	54	56	87	197
13	M	67	DFR4	78	87	83	248
15	F	76	CHG1	57	59	63	179
16	F	45	DFR4	68	72	78	218
17	M	78	GTR6	47	58	62	167
18	F	61	DFR4	65	68	73	206
21	F	71	GTR6	68	73	78	219
33	M	41	CHG1	73	81	91	245
36	F	70	DFR4	55	54	76	185
38	M	47	CHG1	65	75	82	222

Example of using DDE to write data to Excel

To output data to an Excel spreadsheet, define a DDE Triplet which contains the co-ordinates of the cells to which data is to be written.

Then use a null data step containing 'Put' statements, to write out the values of required variables to the spreadsheet.

For example, the following code uses a Proc Summary to calculate simple statistics on the data

that was originally read in from Excel. The statistics are output to a data set, which is used to update the Excel spreadsheet:

```

00001 /* Derive simple statistics for the three score variables */
00002
00003 proc summary data=work.clin;
00004   var score1-score3;
00005   output out=work.clinsum;
00006 run;
00007
00008 /* Assign a Fileref to the Excel Spreadsheet and specify that
00009 Dynamic Data Exchange is to be used */
00010
00011 FILENAME xfile DDE 'excel\courses\data\clin.xls\ref23c7';
00012 /* Output statistics for Score variables to Excel Spreadsheet */
00013
00014 data _null_;
00015   file xfile;
00016   set clinsum;
00017   put _stat_ score1 score2 score3;
00018 run;

```

Note that the only change to the Filename statement compared with the previous example, is that the DDE Triplet points at a different range of cells in the spreadsheet.

A null data step is used to read the data set containing the statistics and 'Put' statements write out the values of selected variables to the Excel spreadsheet. This is no different to writing out data to a text file.

Note that a null data step is used when you wish to avoid creating another data set.

Running the above code updates the Excel spreadsheet accordingly:

The updated Excel spreadsheet complete with summary statistics:

	A	B	C	D	E	F	G	H
1	Clinical Trials Data							
2								
3	Patient	Sex	Age	Drug	Score 1	Score 2	Score 3	Total Score
4	1	M	56	CHG1	87	77	88	252
5	2	F	44	DFR4	45	23	67	135
6	4	F	34	GTR6	44	34	76	154
7	7	M	45	CHG1	68	76	79	223
8	8	M	43	GTR6	54	56	87	197
9	13	M	67	DFR4	78	87	83	248
10	15	F	76	CHG1	57	59	63	179
11	16	F	45	DFR4	68	72	78	218
12	17	M	78	GTR6	47	58	62	167
13	18	F	61	DFR4	65	68	73	206
14	21	F	71	GTR6	68	73	78	219
15	33	M	41	CHG1	73	81	91	245
16	36	F	70	DFR4	55	54	76	185
17	38	M	47	CHG1	65	75	82	222
18								
19					N	14	14	14
20					MIN	44	23	62
21					MAX	87	87	91
22					MEAN	62.42857	63.78571	77.35714
23					STD	12.7141	18.01358	8.863123

Invoking an application from SAS

In the previous examples, the invocation of Excel and opening of a spreadsheet file has been done by clicking on the required Icon and using the pull down menus within Excel.

The application remains running in the background until the user actively closes it down.

Ideally, you only want an application running for the time that it takes to access the data. Otherwise, the idle application ties up valuable memory that can be better used. The solution to this problem is to open and close applications from your SAS program.

An application may be called from SAS by using the following statements:

```
OPTIONS NOXWAIT NOXSYNC;
```

```
X 'C:\EXCEL\EXCEL';
```

```
DATA _NULL_;  
    T=SLEEP(5);  
RUN;
```

NOXWAIT causes control to be returned to SAS as soon as the specified 'X' command has been executed. In the above example, as soon as Excel has been loaded, control will return to SAS.

NOXSYNC causes the application called from SAS to execute asynchronously so that it does not interfere with the SAS session.

Remember to reset the above options to XWAIT and XSYNC when the application is closed. The **X** statement contains the path and name of the executable file that calls the application. The **Sleep** statement inside the null data step simply gives the called application time to load. In the above example, 5 seconds has been allowed.

Issuing Commands from SAS

Having invoked an application like Excel using the above statements, you need to issue commands that carry out tasks such as opening the required spreadsheet file and selecting data. This is done through what is called the SYSTEM Topic:

Assign a Fileref that includes the DDE topic SYSTEM to enable Excel or Word commands to be issued from SAS.

```
FILENAME CMDS DDE 'Excel|SYSTEM';
```

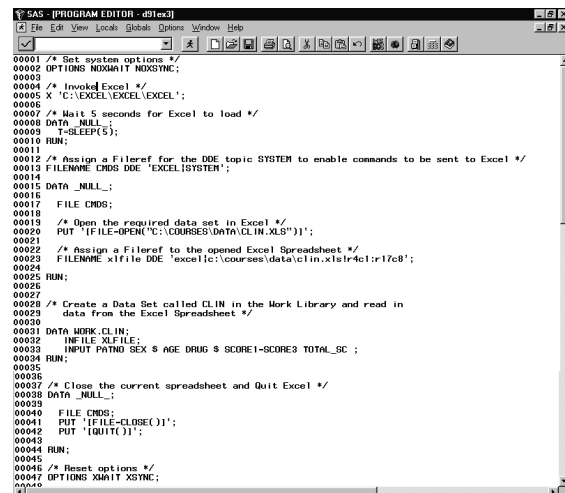
Commands may now be sent to Excel by using 'Put' statements inside a null data step. Note that the syntax of the commands issued is application dependent.

```
DATA _NULL_;  
  
    FILE CMDS;  
  
    PUT                                     '[ FILE-  
OPEN( || C:\COURSES\DATA\CLIN.XLS=) ]';  
  
RUN;
```

Example of invoking and issuing commands to Excel

The example program does the following:

1. Invokes Excel.
2. Opens a Spreadsheet.
3. Reads spreadsheet data into a SAS data set.
4. Closes the Spreadsheet.
5. Closes Excel.

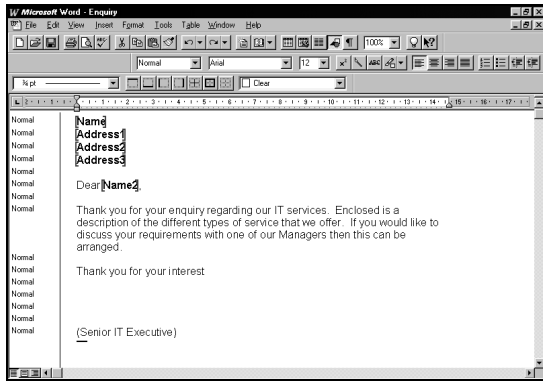


```
SAS - (PROGRAM EDITOR - d91ex3)  
File Edit View Layout Global Options Window Help  
00001 /* Set system options */  
00002 OPTIONS NOXWAIT NOXSYNC;  
00003  
00004 /* Invoke Excel */  
00005 X 'C:\EXCEL\EXCEL';  
00006  
00007 /* Wait 5 seconds for Excel to load */  
00008 DATA _NULL_;  
00009     T=SLEEP(5);  
00010 RUN;  
00011  
00012 /* Assign a Fileref for the DDE topic SYSTER to enable commands to be sent to Excel */  
00013 FILENAME CMDS DDE 'Excel|SYSTER';  
00014  
00015 DATA _NULL_;  
00016  
00017     FILE CMDS;  
00018  
00019     /* Open the required data set in Excel */  
00020     PUT 'FILE=OPEN('C:\COURSES\DATA\CLIN.XLS=)';  
00021  
00022     /* Assign a Fileref to the opened Excel Spreadsheet */  
00023     FILENAME xfile DDE 'excel|c:\courses\data\clin.xls|r4c1:r17c8';  
00024  
00025     RUN;  
00026  
00027  
00028 /* Create a Data Set called CLIN in the Work Library and read in  
00029 data from the Excel Spreadsheet */  
00030  
00031 DATA WORK.CLIN;  
00032     INFILE XFILE;  
00033     INPUT PATNO SEX $ AGE DRUG $ SCORE1-SCORE3 TOTAL SC ;  
00034 RUN;  
00035  
00036  
00037 /* Close the current spreadsheet and Quit Excel */  
00038 DATA _NULL_;  
00039  
00040     FILE CMDS;  
00041     PUT 'FILE=CLOSE()';  
00042     PUT 'QUIT()';  
00043  
00044     RUN;  
00045  
00046 /* Reset options */  
00047 OPTIONS XWAIT XSYNC;
```

Dynamic Data Exchange to Word

Using DDE to access data stored in a Microsoft Word document is the almost the same procedure as used for Excel. The only difference is that the location of data in Word is specified by referencing Bookmarks rather than a cell range.

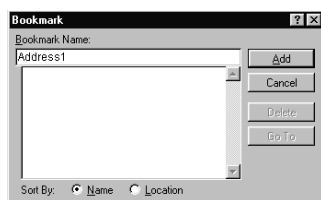
A bookmark is simply a named marker placed within the Word document. The following Word Document uses bookmarks to denote the position of Name and Address fields:



To create a bookmark within a Word document, highlight the text which you want to use as a bookmark, then select **Edit** and **Bookmark** from the pull down menu:



Enter a name for the bookmark and click on **Add**



The components of the DDE Triplet for the above example are as follows:

Name of Executable File: WINWORD

Full Path and Name of Word document:
C:\COURSES\DATA\ENQUIRY.DOC

Location of data (Name of Bookmark): Address1

Invoking Word from SAS

The procedure used to call Word from SAS is the same procedure used to call Excel or any other Windows application from SAS. Set the appropriate options, use an 'X' statement to call Word, and allow Word a few seconds to load.

```
OPTIONS NOXWAIT NOXSYNC;
```

```
X 'C:\WINWORD\WINWORD' ;
```

```
DATA _NULL_;
```

```
T=SLEEP(5);
```

```
RUN;
```

Remember to reset the above Options to XWAIT and XSYNC when the application is closed.

Issuing Commands to Word from SAS

As with Excel, this is done by assigning a Fileref to the DDE topic SYSTEM:

```
FILENAME CMDS DDE 'WINWORD|SYSTEM' ;
```

Once the above Fileref is assigned, commands can be sent to Word by using 'Put' statements inside a null data step.

```
DATA _NULL_;
```

```
FILE CMDS;
```

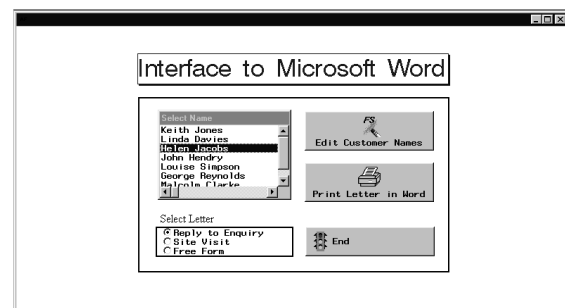
```
PUT
```

```
'[FILEOPEN( || C:\COURSES\DATA\ENQUIRY.DOC=) ] ' ;
```

```
RUN;
```

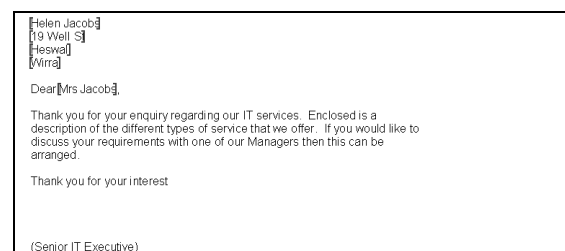
Example of using DDE to access a Word document

The following SAS/AF Application automatically inserts the name and address of a customer into the selected Word document and prints it:



The name and address information is stored in a SAS data set that may be accessed and updated by clicking on the 'Edit Customer Names' icon.

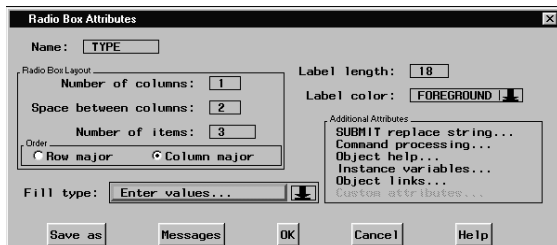
The name and address information of the selected customer is inserted into the Word document as follows:



Note how the name and address details have been written to their respective bookmarks. The square brackets, which denote the boundary of a bookmark, do not appear in the printed document.

Note the following about the SCL Program that controls the Frame:

1. The physical paths for the Word executable file and Word documents, are assigned to Macro variables in the Autoexec file which runs before the application is called. Call Symget routines are used in the INIT section to assign the values of these Macro variables to SCL variables. This approach avoids having to hard code path names in the application.
2. The name of the Word document to open is derived by concatenating the document path, letter type, and the suffix '.DOC'.
3. The letter type is obtained by sending the _GET_TEXT_ method to the Radio Box, which returns the Character value assigned to the selected item. Each Radio Box item has a Character return value specified in the attributes window:



Click on **Enter values..** to access Return values for items



Clicking on a Radio Box Item displays its Return value.

4. A Submit Block contains all of the SAS code used to load Word, access the selected

document and print it. Recall that an ampersand within a Submit Block references an SCL variable.

5. Five different Filrefs are assigned to point at the various name and address bookmarks in the Word document.

The SCL Program is as follows (see next column):

```

%BUILD: SOURCE D02X1 SCL (E)

00001 /* Declare the length of two Character variables */
00002 length selname $50 lettype $8;
00003
00004 INIT:
00005
00006 /* Get the physical paths to the Word Documents and to the Word
00007 executable which are stored as Macro variables */
00008 path=symget('path');
00009 wordpath=symget('wordpath');
00010
00011 return;
00012
00013
00014
00015 NAMES:
00016
00017 /* Get the Name selected from the List Box */
00018 call notify('names', '_get_last_sel_', row, isel, selname);
00019
00020 return;
00021
00022
00023 EDIT:
00024
00025 /* Open the F$EDIT window */
00026 call fedit('ddata.names');
00027
00028 /* Refresh the contents of the List Box */
00029 call notify('lbnames', '_repopulate_');
00030
00031 return;
00032
00033
00034 PRINT:
00035
00036 /* If a name has been selected then generate and print the Word document */
00037 if selname ne '' then do;
00038
00039 /* Get the Type of Letter selected from the Radio Box */
00040 call notify('type', '_get_text_', lettype);
00041
00042 /* Concatenate the Path and Letter Type to generate the name of the
00043 document to be opened in Word */
00044 wordfile=path!!lettype!!'.doc';
00045
00046 SUBMIT CONTINUE;
00047
00048 /* Set system options */
00049 OPTIONS NOXMIT NOXSYN;
00050
00051 /* Invoke Word */
00052 X wordpath;
00053
00054
00055 /* Wait 5 seconds for Word to load */
00056 DATA _NULL_;
00057 T-SLEEP(5);
00058 RUN;
00059
00060 /* Assign a Fileref for the DOE topic $ISTEM to enable commands
00061 to be sent to Word */
00062 FILENAME CHDS DOE 'WINWORD$ISTEM';
00063
00064 DATA _NULL_;
00065 FILE CHDS;
00066
00067 /* Open the required Word Document */
00068 PUT 'FILEOPEN('wordfile')';
00069
00070
00071 /* Assign Filerefs to the named bookmarks in the Word Document */
00072 FILENAME uname DOE 'winword!wordfile!name';
00073 FILENAME wadd1 DOE 'winword!wordfile!address1';
00074 FILENAME wadd2 DOE 'winword!wordfile!address2';
00075 FILENAME wadd3 DOE 'winword!wordfile!address3';
00076 FILENAME uname2 DOE 'winword!wordfile!name2';
00077
00078 RUN;
00079
00080 /* Open the Data Set called names and read the observation that
00081 corresponds to the selected row in the List Box. Use File and Put
00082 statements to write out the values of variables to the appropriate
00083 position in the Word Document */
00084
00085 DATA _NULL_;
00086 SET DATA.NAMES (FIRSTOBS=&row OBS=&row);
00087 FILE UNAME;
00088 PUT NAME;
00089 FILE WADD1;
00090 PUT WADD1;
00091 FILE WADD2;
00092 PUT WADD2;
00093 FILE WADD3;
00094 PUT WADD3;
00095 FILE UNAME2;
00096 PUT UNAME2;
00097
00098 RUN;
00099
00100 /* Print the Word Document */
00101 DATA _NULL_;
00102 FILE CHDS;
00103 PUT 'FILEPRINT()';
00104
00105 RUN;
00106
00107
00108 /* Close the current document and Quit Word */
00109 DATA _NULL_;
00110
00111 FILE CHDS;
00112 PUT 'FILECLOSE()';
00113 PUT 'FILEEXIT()';
00114
00115 RUN;
00116
00117 /* Reset options */
00118 OPTIONS XMIT XSYN;
00119
00120
00121 END;
00122
00123
00124 return;
00125
00126
00127
00128 END;
00129
00130 call exccmd('end');
00131
00132 return;
00133
00134
00135

```

Workshop Session

Create an Excel Spreadsheet and enter the text 'Demographic Data' in cell A1. Save the spreadsheet and make a note of the path and file name used.

Write a SAS program to do the following:

1. Invoke Excel.
2. Open the spreadsheet file that you have created for this exercise.
3. Write out values of Demographic Data Set variables to the Excel spreadsheet.
4. Save the spreadsheet.
5. Close the spreadsheet file.
6. Close Excel.

	A	B	C	D	E	F	G	H	I	J
1	Demographic Data									
2										
3	Dave Derry	22	M	64	60	S	0	1	13592	low
4	Julia Pandlebury	26	F	67	74	M	1	1	8870	low
5	Norman Harvey	100	M	55	62	D	2	1	12672	low
6	Harold Hicks	28	M	84	74	M	3	1	23760	high
7	Mary Moleworth	23	F	84	88	P	1	1	10512	low
8	Bob Bobington	30	M	67	76	S	0	1	7620	low
9	Freda Bamford	34	F	78	72	M	2	1	28512	high
10	Julia Kidd	38	F	69	56	M	2	1	14840	low
11	Helen Canderford	56	F	52	60	M	3	2	47520	high
12	Mary Chapel	52	F	66	69	M	5		23760	high
13	Jennifer Dawson	60	F	88	71	M	3	2	20592	high
14	Julio Jennings	56	F	61	61	M	0	2	13760	low
15	Shirley Walters	46	F	63	76	W	3	1	47620	high
16	Diane Dullely	65	F	67	81	M	2	1	13840	low
17	Dawn Duwet	23	F	78	80	S	0	1	12640	low

The resulting spreadsheet should appear as follows:

Hints:

Use a separate Fileref to define the range of cells into which people's names are written and include the NOTAB option i.e.:

```
FILENAME                XLF1                DDE
'EXCEL|C:\COURSES\DATA\DEMOG.XLS!r3c1:r52c1' NOTAB;
```

The NOTAB option makes SAS ignore Tab Characters between variables. In this example, the space between the first name and surname is taken as a Tab Character, which results in the two parts of the name being put into different columns in Excel unless NOTAB is specified. Use a second Fileref to define the range of cells for the remaining variables but do not include the NOTAB option as non of the variables include space characters:

```
FILENAME                XLF2                DDE
'EXCEL|C:\COURSES\DATA\DEMOG.XLS!r3c1:r52c1';
```

Within a null Data Step, use separate File and Put statements of the following form to output the data:

```
DATA _NULL_;
  FILE XLF1;
  PUT NAME;
  FILE XLF2;
  PUT AGE GENDER etc;
RUN;
```

Remember to reset options NOXWAIT and NOXSYNC to XWAIT and XSYNC respectively.