Reading and Writing Data from Microsoft Excel/Word Using DDE
Prepared by
Destiny Corporation

Dynamic Data Exchange to Excel

Dynamic Data Exchange is a means of transferring data between different Windows applications such as the SAS System and Microsoft Excel or Microsoft Word. The technique uses what is known as a Client / Server type relationship, where the Client application makes the requests and the Server application responds accordingly. When using DDE with the SAS System, SAS is always the Client.

The DDE technique may be used within a Data Step, a Macro or from inside a SAS/AF application.

Using the DDE technique involves specifying the following three components in what is called a DDE Triplet:

1. The path and name of the executable file for the server application. For example, with Microsoft Excel the name of the executable is EXCEL.

2. The full path and file name of the document or spreadsheet with which you wish to share data. This is referred to in SAS documentation as the Topic.

3. The location of data to be read or modified. With a spreadsheet application, this is a range of cells. In applications such as Microsoft Word, the location of data is defined by what are known as bookmarks.

An example of a DDE Triplet for an Excel Spreadsheet is as follows:

EXCEL | C:\COURSES\DATA\CLIN.XLS ! R4C17:R17C8

The DDE Triplet is then incorporated into a Filename statement of the following form:

FILENAME fileref DDE 'DDE-Triplet' | 'CLIPBOARD' <DDE-Options>;

where:

fileref is a valid fileref

DDE is the keyword that primes the SAS System for Dynamic Data Exchange.

'DDE-Triplet' is the three part specification of the DDE external file and takes the following form: 'Executable-file-name|Topic!Data-range'. Note that the triplet is application dependent and may differ between different releases of the same application.

CLIPBOARD' is an alternative to directly specifying the DDE triplet. To use this option, copy the required data from the Server application to the Clipboard and specify the 'CLIPBOARD' option in the Filename statement. SAS will then determine the Triplet for itself. You will be able to see the triplet definition by looking at the Log.
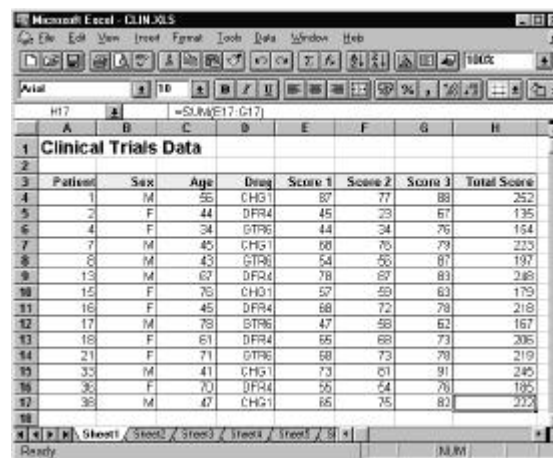
DDE-Options are as follows:

HOTLINK invokes the DDE HOTLINK facility, which causes the DDE link to be activated whenever data in the spreadsheet range is updated.

NOTAB makes the SAS System ignore tab characters between variables.

COMMAND allows remote commands to be sent to DDE server applications.

Example of using DDE to read data from Excel

Consider the following Excel Spreadsheet:

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:

Running the above code generates the following data set:

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:

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:

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.

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:
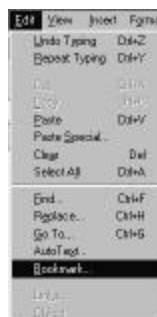
Enter a name for the book

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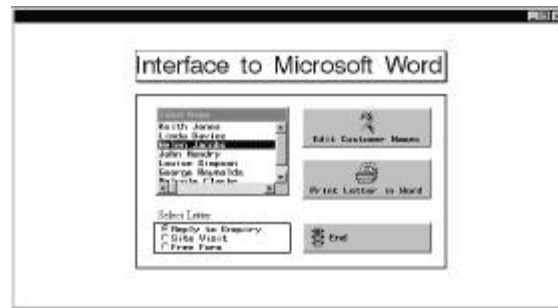
The name and address information is stored in a SAS data set that may be accessed and updated by clicking on the 'Edit Customer Names' icon.

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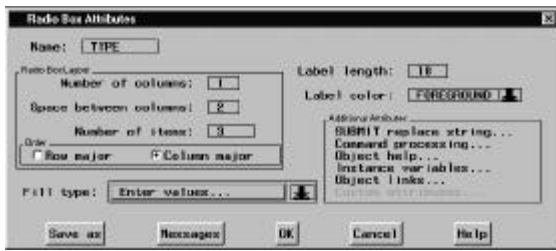
Click **Enter values..** to access Return values for items on

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):

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.

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.

Prepared by:

Destiny Corporation • 100 Great Meadow Rd Suite 601• Wethersfield, CT 06109-2355
Phone: (860) 721-1684 • 1-800-7TRAINING
• Fax: (860) 721-9784
Email: info@destinycorp.com Web Site: www.destinycorp.com