CHAPTER

11

Using Dynamic Data Exchange

Overview of Dynamic Data Exchange (DDE) **217**
DDE Syntax within SAS **217**
Referencing the DDE External File **218**
Using the DDE Triplet **218**
Controlling Another Application Using DDE **219**
DDE Examples **220**
Using the X Command to Open a DDE Server **220**
Using DDE to Write Data To Microsoft Excel **220**
Using DDE to Write Data To Microsoft Word **220**
Using DDE to Read Data from Microsoft Excel **221**
Using DDE to Read Data from Microsoft Word **221**
Using DDE and the SYSTEM Topic to Invoke Commands in an Application Using Excel **221**
Using the NOTAB Option with DDE **222**
Using the DDE HOTLINK **223**
Using the !DDE_FLUSH String to Transfer Data Dynamically **224**
Reading Missing Data **225**

Overview of Dynamic Data Exchange (DDE)

Dynamic Data Exchange (DDE) is a method of dynamically exchanging information between Windows applications. DDE uses a client/server relationship to enable a client application to request information from a server application. In Version 8, the SAS System is always the client. In this role, the SAS System requests data from server applications, sends data to server applications, or sends commands to server applications.

You can use DDE with the DATA step, the SAS macro facility, SAS/AF applications, or any other portion of the SAS System that requests and generates data. DDE has many potential uses, one of which is to acquire data from a Windows spreadsheet or database application.

Note: Many Windows programs, including the SAS System, now support OLE to facilitate communication between applications. If you need to share data with an application that supports OLE, you might prefer to use the OLE support that is built into the SAS System. For more information, see “About OLE” on page 188. △

DDE Syntax within SAS

To use DDE in SAS, issue a FILENAME statement with the following syntax:

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

where:

fileref
is a valid fileref (as described in "Referencing External Files" on page 106).

DDE
is the device-type keyword that tells the SAS System you want to use Dynamic Data Exchange.

'*DDE-triplet*'
is the name of the DDE external file.

DDE-options
can be any of the following:

HOTLINK
instructs the SAS System to use the DDE HOTLINK. For an example of using this option, see "Using the DDE HOTLINK" on page 223.

NOTAB
instructs the SAS System to ignore tab characters between variables. For an example of using this option, see "Using the NOTAB Option with DDE" on page 222.

COMMAND
allows remote commands to be issued to DDE server applications. For more information, see "Controlling Another Application Using DDE" on page 219.

CAUTION:
Use caution when using DDE with data values that are blank or missing. For sample code, see "Reading Missing Data" on page 225. △

Referencing the DDE External File

When you define a fileref to use with DDE, the *DDE-triplet* argument refers to the DDE external file.

Using the DDE Triplet

The DDE triplet is application-dependent and is different for every application you run. For information on an application's DDE triplet, see the application's documentation.

The triplet takes the following form:

'*application-name*|*topic*!*item*'

where:

application-name
is the executable filename of the server application. For example, the *application-name* for Microsoft Word is **winword**, and for Microsoft Excel it is **excel**.

topic
is the topic of conversation (between SAS and the DDE server application). This is typically the full path filename of the document or spreadsheet with which you want to share data.

item
is the range of conversation specified between the client and server applications. In spreadsheet applications, this is usually a range of cells. For document-based applications (for example, Microsoft Word), the item is something that defines a location in the document, such as a bookmark.

Valid values for all of these arguments vary depending on the server application. A software package supporting DDE as a server should list acceptable values for the triplet information in documentation supplied with the application.

Note: The server application must be started before trying to communicate with it using DDE. Also, the DDE triplet format might differ among different applications and among different versions of the same application. △

For example, in order to place text into a Microsoft Word document TESTDDE.DOC located at C:\TEMP with a bookmark named NUMBER, you could use this code:

```
filename test dde 'winword|"c:\temp\testdde.doc"
                  !NUMBER' notab;
```

The application-name is **winword**, the topic is **"c:\temp\testdde.doc"**, and the range is **!NUMBER**.

This following example assumes you are using Microsoft Excel 5.0 or greater.

Suppose you want to use SAS to populate the first 4 rows and 2 columns of the Microsoft Excel spreadsheet named Sales Data stored in C:\EXCEL\SALES.XLS. You would use the following code:

```
filename test dde 'Excel|c:\excel\
                  [Sales.xls]Sales Data!R1C1:R4C2'
```

The application-name is **Excel**, the topic is **c:\excel\[Sales.xls] Sales Data**, and the range is **R1C1:R4C2**.

If your server application will copy the DDE-triplet to the Windows clipboard, you can display the DDE-triplet in the SAS System. You do this by selecting

Solutions ▶ Accessories ▶ DDE triplet

Controlling Another Application Using DDE

DDE server applications support certain commands that you can issue by using a DDE link to control the application. To use these commands, use the special topic name SYSTEM in the DDE triplet and leave the item name blank. You can then use the INPUT statement for input from an application and the PUT statement to issue commands to the server application.

For those DDE server applications that do not recognize the SYSTEM topic name, you can specify the COMMAND option in the FILENAME statement you use to define the DDE link. When you specify the COMMAND option, you do not specify the item name in the DDE triplet.

Note: With SAS/AF software and OLE automation, you can automate any Windows application that supports OLE 2.0 as a server. For more information about using SAS and OLE, see "Automating OLE Objects and Applications" on page 196. △

DDE Examples

This section provides several examples of using DDE with the SAS System under Windows. These examples use Microsoft Excel and Microsoft Word as DDE servers, but any application that supports DDE as a server can communicate with the SAS System.

Before you run these examples, you must first invoke Microsoft Excel and Microsoft Word, and open the spreadsheet or document used in the example.

Note: DDE examples are included in the host-specific sample programs that you access from the **Help** menu. △

Using the X Command to Open a DDE Server

A DDE server application can be opened using the X command within SAS code. The XWAIT and XSYNC options must be turned off.

```
options noxwait noxsync;
x 'excel'; /* you might need to specify */
           /* the complete pathname     */
```

Using DDE to Write Data To Microsoft Excel

The first example sends data from a SAS session to an Excel spreadsheet. The target cells are rows 1 through 100 and columns 1 through 3. To do this, submit the following program:

```
/* The DDE link is established using   */
/* Microsoft Excel SHEET1, rows 1      */
/* through 100 and columns 1 through 3 */
filename random dde
   'excel|sheet1!r1c1:r100c3';
data random;
   file random;
   do i=1 to 100;
      x=ranuni(i);
      y=10+x;
      z=x-10;
      put x y z;
   end;
run;
```

Using DDE to Write Data To Microsoft Word

This example sends a text string to a Micorosft Word document at a given bookmark. Note the difference between using DDE with Microsoft Word and Microsoft Excel.

```
filename testit dde 'winword|"c:\temp\testing.doc"
                     !MARK' notab;

data _null_;
   file testit;
   put ' This is a test.';
run;
```

Note: If you are writing to Microsoft Word97, use Visual Basic commands such as FileOpen.Name, FileSave, FileClose, and Insert. If the PUT statement contains a macro that Word97 does not understand, you will see this message:

```
Ambiguous name detected:  TmpDDE
```

△

Using DDE to Read Data from Microsoft Excel

You can also use DDE to read data from an Excel application into the SAS System, as in the following example:

```
/* The DDE link is established using   */
/* Microsoft Excel SHEET1, rows 1      */
/* through 10 and columns 1 through 3  */
filename monthly
  dde 'excel|sheet1!r1c1:r10c3';
data monthly;
   infile monthly;
   input var1 var2 var3;
run;
proc print;
run;
```

Using DDE to Read Data from Microsoft Word

This example reads data from a Microsoft Word document at a given bookmark.

```
filename testit dde 'winword|"c:\temp\testing.doc"
                     !MARK' notab;

libname workdir 'c:\temp';

/* Get ready to read the first bookmark.  */

data workdir.worddata;
   length wordnum $5;
   infile testit;
   input wordnum $;
run;
```

Using DDE and the SYSTEM Topic to Invoke Commands in an Application Using Excel

You can issue commands to Excel or other DDE-compatible programs directly from the SAS System using DDE. In the following example, the Excel application is invoked using the X command; a spreadsheet called SHEET1 is loaded; data are sent from the SAS System to Excel for row 1, column 1 to row 20, column 3; and the commands required to select a data range and sort the data are issued. The spreadsheet is then saved and the Excel application is terminated.

```
/* This code assumes that Excel       */
/* is installed on the current        */
```

```
/* drive in a directory called EXCEL. */

options noxwait noxsync;
x 'excel';  /* you might need to specify */
            /* the entire pathname       */

/* Sleep for 60 seconds to give */
/* Excel time to come up.       */

data _null_;
   x=sleep(60);
run;

/* The DDE link is established using   */
/* Microsoft Excel SHEET1, rows 1      */
/* through 20 and columns 1 through 3  */

filename data
   dde 'excel|sheet1!r1c1:r20c3';
data one;
   file data;
   do i=1 to 20;
      x=ranuni(i);
      y=x+10;
      z=x/2;
      put x y z;
   end;
run;

/* Microsoft defines the DDE topic */
/* SYSTEM to enable commands to be */
/* invoked within Excel.           */

filename cmds dde 'excel|system';

/* These PUT statements are       */
/* executing Excel macro commands */

data _null_;
   file cmds;
   put '[SELECT("R1C1:R20C3")]';
   put '[SORT(1,"R1C1",1)]';
   put '[SAVE()]';
   put '[QUIT()]';
run;
```

Using the NOTAB Option with DDE

The SAS System expects to see a TAB character placed between each variable that is communicated across the DDE link. Similarly, the SAS System places a TAB character between variables when data are transmitted across the link. When the NOTAB option is placed in a FILENAME statement that uses the DDE device-type keyword, the SAS System accepts character delimiters other than tabs between variables.

The NOTAB option also can be used to store full character strings, including embedded blanks, in a single spreadsheet cell. For example, if a link is established between the SAS System and the Excel application, and a SAS variable contains a character string with embedded blanks, each word of the character string is normally stored in a single cell. To store the entire string, including embedded blanks in a single cell, use the NOTAB option as in the following example:

```
/* Without the NOTAB option, column1  */
/* contains 'test' and column2        */
/* contains 'one'.                    */

filename test
  dde 'excel|sheet1!r1c1:r1c2';
data string;
   file test;
   a='test one';
   b='test two';
   put a $15. b $15.;
run;

/* You can use the NOTAB option to store   */
/* each variable in a separate cell. To    */
/* do this, you must force a tab           */
/* ('09'x) between each variable, as in    */
/* the PUT statement.                      */
/* After performing this DATA step, column1*/
/* contains 'test one' and column2         */
/* contains 'test two'.                    */

filename test
  dde 'excel|sheet1!r2c1:r2c2' notab;
data string;
   file test;
   a='test one';
   b='test two';
   put a $15. '09'x b $15.;
run;
```

Using the DDE HOTLINK

If the HOTLINK option is specified, the DDE link is activated every time the data in the specified spreadsheet range are updated. In addition, DDE enables you to poll the data when the HOTLINK option is specified to determine whether data within the range specified have been changed. If no data have changed, the HOTLINK option returns a record of 0 bytes. In the following example, row 1, column 1 of the spreadsheet SHEET1 contains the daily production total. Every time the value in this cell changes, the SAS System reads in the new value and outputs the observation to a data set. In this example, a second cell in row 5, column 1 is defined as a status field. Once the user completes data entry, typing any character in this field terminates the DDE link:

```
/* Enter data into Excel SHEET1 in   */
/* row 1 column 1. When you          */
/* are through entering data, place  */
/* any character in row 5            */
/* column 1, and the DDE link is     */
```

```
/* terminated.                          */

filename daily
  dde 'excel|sheet1!r1c1' hotlink;
filename status
  dde 'excel|sheet1!r5c1' hotlink;
data daily;
   infile status length=flag;
   input @;
   if flag ne 0 then stop;
   infile daily length=b;
   input @;

   /* If data have changed, then the */
   /* incoming record length         */
   /* is not equal to 0.             */

   if b ne 0 then
      do;
         input total $;
         put total=;
         output;
      end;
run;
```

It is possible to establish multiple DDE sessions. The previous example uses two separate DDE links. When the HOTLINK option is used and there are multiple cells referenced in the *item* specification, if any one of the cells changes, then all cells are transmitted.

Unless the HOTLINK option is specified, DDE is performed as a single one–time data transfer. That is, the values currently stored in the spreadsheet cells at the time that the DDE is processed are values that are transferred.

Using the !DDE_FLUSH String to Transfer Data Dynamically

DDE also enables you to program when the DDE buffer is dumped during a DDE link. Normally, the data in the DDE buffer are transmitted when the DDE link is closed at the end of the DATA step. However, the special string **'!DDE_FLUSH'** issued in a PUT statement instructs the SAS System to dump the contents of the DDE buffer. This function allows you considerable flexibility in the way DDE is used, including the capacity to transfer data dynamically through the DATA step, as in the following example:

```
/* A DATA step window is displayed.   */
/* Enter data as prompted.            */
/* When you are finished, enter STOP  */
/* on the command line.               */

filename entry
   dde 'excel|sheet1!r1c1:r1c3';
dm 'pmenu off';
data entry;
   if _n_=1 then
      do;
         window ENTRY color=black
```

```
          #3 'This is data for Row 1 Column 1'
             c=cyan +2 var1 $10. c=orange
          #5 'This is data for Row 1 Column 2'
             c=cyan +2 var2 $10. c=orange
          #7 'This is data for Row 1 Column 3'
             c=cyan +2 var3 $10. c=orange;
       end;
    flsh='!DDE_FLUSH';
    file entry;
    do while (upcase(_cmd_) ne 'STOP');
       display entry;
       put var1 var2 var3 flsh;
       output;
       VAR1='';
       VAR2='';
       VAR3='';
    end;
    stop;
run;
dm 'pmenu on';
```

Reading Missing Data

This example illustrates reading missing data from an Excel spreadsheet called SHEET1. This example reads the data in columns 1 through 3 and rows 10 through 20. Some of the data cells may be blank. Here is an example of what some of the data look like:

```
...
10   John    Raleigh      Cardinals
11   Jose    North Bend   Orioles
12   Kurt    Yelm         Red Sox
13   Brent                Dodgers
...
```

Here's the code that can read these data correctly into a SAS data set:

```
filename mydata
  dde 'excel|sheet1!r10c1:r20c3';
data in;
   infile mydata dlm='09'x notab
          dsd missover;
   informat name $10. town $char20.
            team $char20.;
   input name town team;
run;
proc print data=in;
run;
```

In this example, the NOTAB option tells the SAS System not to convert tabs that are sent from the Excel application into blanks. Therefore, the tab character can be used as the delimiter between data values. The DLM= option specifies the delimiter character, and **'09'x** is the hexadecimal representation of the tab character. The DSD option specifies that two consecutive delimiters represent a missing value. The default delimiter is a comma. For more information about the DSD option, see *SAS Language Reference: Dictionary*. The MISSOVER option prevents a SAS program from going to a

new input line if it does not find values in the current line for all the INPUT statement variables. With the MISSOVER option, when an INPUT statement reaches the end of the current record, values that are expected but not found are set to missing.

The INFORMAT statement forces the DATA step to use modified list input, which is crucial to this example. If you do not use modified list input, you receive incorrect results. The necessity of using modified list input is not DDE specific. You would need it even if you were using data in a CARDS statement, whether your data were blank- or comma-delimited.

The correct bibliographic citation for this manual is as follows: SAS Institute Inc., *SAS Companion for the Microsoft Windows Environment, Version 8,* Cary, NC: SAS Institute Inc., 1999. pp.555.

SAS Companion for the Microsoft Windows Environment, Version 8

ISBN 1–58025–524–8

SAS Institute Inc., SAS Campus Drive, Cary, North Carolina 27513.

1st printing, September 1999

The Institute is a private company devoted to the support and further development of its software and related services.