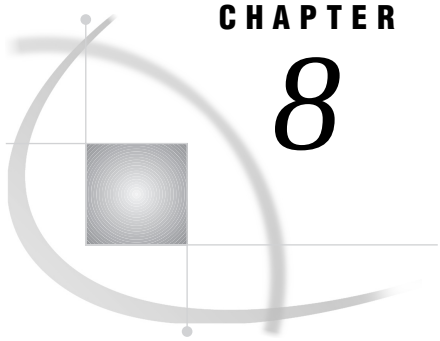

CHAPTER

8

Using Dynamic Data Exchange

Overview of Dynamic Data Exchange

Dynamic Data Exchange (DDE) is a method of dynamically exchanging information between OS/2 applications. DDE uses a client/server relationship to enable a client application to request information from a server application. 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 an OS/2 spreadsheet or database application.

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 82.

DDE
is the device-type keyword that tells the SAS System that 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 150.

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 149.

COMMAND
allows remote commands to be issued to DDE server applications. For more information, see "Controlling Another Application Using DDE" on page 147.

CAUTION:
Use caution when using DDE with data values that are blank or missing. For sample code that reads in missing data, see "Reading Missing Data" on page 151. △

Referencing the DDE External File

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

Determining the DDE Triplet

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

The triplet uses the following syntax:

'application-name | topic!item'

where

application-name
is the executable filename of the server application.

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 that is specified between the client and server applications. In spreadsheet applications, this is usually a range of cells. For document-based applications, 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 that supports DDE as a server should list acceptable values for the triplet information in documentation that is supplied with the application.

Note: You must invoke the server application before you try to communicate with it by using DDE. Also, the DDE triplet format might differ among different applications and among different versions of the same application. △

If your server application will copy the DDE-triplet to the OS/2 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 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 that you use to define the DDE link. When you specify the COMMAND option, you do not specify the item name in the DDE triplet.

DDE Examples

These examples use Microsoft Excel as the DDE server, but any application that supports DDE as a server can communicate with the SAS System. Before you run most of these examples, you must first invoke the server application and open the spreadsheet that is used in the example. The only exception to this requirement is found in the example shown in "Using DDE and the SYSTEM Topic to Execute Commands in an Application" on page 148. In this example, you should not invoke Excel first because the example program invokes Excel for you.

If you did not start Excel before you began your SAS session, you can turn off the XWAIT and XSYNC options and use the X statement to start Excel, as follows:

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

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

Using DDE to Write Data

Suppose you want to send data from a SAS session to an Excel spreadsheet. You want to use 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 Read Data

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 and the SYSTEM Topic to Execute Commands in an Application

You can issue commands to Excel or other DDE-compatible programs directly from the SAS System by using DDE. In the following example, the Excel application is invoked; 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 that are required to select a data range and to 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 */
/* executed 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 that a TAB character is 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 you use the NOTAB option 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 if a SAS variable contains a character string that contains 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 this DATA step executes, 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 that was 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 you complete data entry, you can terminate the DDE link by typing any character in this field:

```
/* 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 multiple cells are 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 that are 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'** that you issue in a PUT statement instructs the SAS System to dump the contents of the DDE buffer. This function gives 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 how to read missing data from an Excel spreadsheet that is 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 extract 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
```

Following is 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 should represent a missing value. The default delimiter is a comma. For more information, see the DSD option in *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. When you use the MISSOVER option, values are set to missing if the INPUT statement reaches the end of the current record but does not find the expected values.

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 delimited by blanks or commas.

The correct bibliographic citation for this manual is as follows: SAS Institute Inc., *SAS® Companion for the OS/2® Environment, Version 8*, Cary, NC: SAS Institute Inc., 1999. 448 pp.

SAS® Companion for the OS/2® Environment, Version 8

ISBN 1-58025-521-3

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

1st printing, October 1999