An Introduction to SAS/DMI® Software
Glenn H. Itano, SAS Institute Inc., Cary, NC
Kerrie Brannock, CIBA-Geigy Corporation, Greensboro, NC

Introduction
SAS/DMI® software is a software product that binds the SAS® System with IBM'S ISPF product. The purpose of this tutorial is to introduce you to ISPF Dialog Management Services with an emphasis on creating dialogs with SAS/DMI software. We assume that you are an experienced TSO, SAS, ISPF, and CLIST user.

ISPF Dialogs
With ISPF you can write interactive applications, or dialogs. An ISPF dialog typically consists of a series of panels on which the end user fills in information or selects operations. Based on the users response, certain actions are taken. A few examples of dialog panels that are part of an ISPF dialog are
- the ISPF Primary Option Menu
- the edit panel where user has to fill in data set name
- a partitioned data set (PDS) member selection list.

Dialog Components
An typical ISPF dialog consists of
- Panels - screens the user sees
- Functions - programs that control the displaying of panels in addition to standard processing
- Variables - used to pass values between panels and functions.
- Other Components - includes messages, skeletons; and tables (not covered in this tutorial).

ISPF Panels
A panel is what the end user sees when he is in an ISPF dialog. Each panel is stored in a member of a PDS and the panel name is the member name. The PDS that contains all the panels, or members, is called a panel library. Although a panel library is no different from any other PDS, you should store only panels in your panel library. You must tell ISPF where your panel library is by allocating it to the DName ISPPLIB before you start the dialog.

There are several different kinds of panels. A menu is a panel with a list of options for the user to choose from. The user types one selection on the command line. The control code in the menu then determines what action will be taken. Another type of panel is a data entry panel. This is a panel that displays text and expects input from the user. The example dialogs in this paper will use only data entry panels. Figure 1 depicts a sample data entry panel called PHONEPAN.

\[
\text{PHONEPAN in use}
\]

The name of the PDS member, and therefore the panel, is PHONEPAN. Many of the words and symbols have special meaning to ISPF and are not shown on the panel the user sees. Figure 2 depicts this panel as it appears when displayed as a part of an ISPF function.

```plaintext
Number Registered: 10
Name ==> JOSEPH, DAVID
Phone ==> \\

Press END to Exit
```

Figure 2 Data entry panel PHONEPAN in use

A typical panel consists of body and processing sections. The body section is the part of the panel to be displayed. The processing section can contain special programming statements for tasks like input data validation and cross-field checking. A "\text{\texttt{\textbackslash{}BODY\texttt{\textbackslash{}}}}" always marks a section boundary. Thus \text{\texttt{\textbackslash{}BODY}} denotes the beginning of the body and \text{\texttt{\textbackslash{}PROC}} denotes the ending of the body and the beginning of the processing section. \text{\texttt{\textbackslash{}END}} denotes the end of the panel.

The body section can contain text, variable input, and variable output fields. Text consists of protected character strings that are displayed. A variable input field is where the user enters or changes a variable value when the panel is displayed. A variable output field is a field in which a variable value is displayed but cannot be modified.

Special symbols within the body section of the panel are called attribute characters. An attribute character defines whether the field that immediately follows is text, variable input, or variable output, and the field's display intensity. The default attribute characters are listed below:

- \text{\texttt{\textbackslash{}text\texttt{\textbackslash{}}}} (protected), high intensity
- \text{\texttt{\textbackslash{}text\texttt{\textbackslash{}}}} (protected), low intensity
- \text{\texttt{\textbackslash{}variable\texttt{\textbackslash{}}}} ( unprotected), high intensity
- \text{\texttt{\textbackslash{}variable\texttt{\textbackslash{}}}} (protected), current text intensity

In Figure 1, \text{\texttt{\textbackslash{}BODY}} marks the beginning of the body section. The text strings Number Registered, Name, and Phone are text to be displayed in low intensity. The \text{\texttt{\textbackslash{}NREG}} indicates that NREG is a variable output field. The \text{\texttt{\textbackslash{}A\textbackslash{}}}} marks the beginning of a high intensity text field. Therefore the current value of the variable NREG will be displayed in high intensity. The \text{\texttt{\textbackslash{}NREG}} is a variable input fields NAME and PHONE. The \text{\texttt{\textbackslash{}A\textbackslash{}}}} marks the beginning of the text fields and therefore the end of the input fields. This means that the user will enter values in the NAME and PHONE fields. In the last line of the body, \text{\texttt{\textbackslash{}A\textbackslash{}}}} will cause \text{\texttt{\textbackslash{}END}} to stand out from the rest of the statement since it will be displayed in high intensity while the rest will be in low. \text{\texttt{\textbackslash{}PROC}} marks the end of the body and the beginning of the processing section. The \text{\texttt{\textbackslash{}VER\texttt{\textbackslash{}}}} statements are special statements that validate variable input. In this example, \text{\texttt{\textbackslash{}VER\texttt{\textbackslash{}}}} verifies that values are entered for NAME and PHONE before the user can leave the panel. \text{\texttt{\textbackslash{}END}} marks the end of the panel.

Figure 1 Data entry panel PHONEPAN
Dialog Functions

Now that you know how to create panels, you need a means of displaying them. A dialog function controls execution of dialog management services, including the display of dialog panels. Dialog functions can be written as CLISTS or in languages such as PL/I or the SAS DATA step.

Invoking ISPF Services

The ISPEXEC statement is used to invoke ISPF services from a CLIST. The syntax of the ISPEXEC statement is

ISPEXEC servicename options

where servicename is the name of the ISPF dialog management service being invoked and service specific options. The IBM manual ISPF Dialog Management Services has complete listings of the available ISPF services, their parameters, and their return codes.

The CLIST statement that actually displays a panel is

ISPEXEC DISPLAY PANEL (panelname)

Return Codes

After any ISPF service has been invoked, the return code is placed into automatic CLIST variable LASTCC. For the ISPF service DISPLAY, a return code of 0 means the user pressed ENTER. A return code of 8 means the user pressed END. Each ISPF service has its own set of return code meanings. Consult the IBM ISPF Dialog Management Services manuals for the return code meanings for other services.

Dialog Variables

You have seen the use of variable input and output fields on panels. These variables are called ISPF dialog variables. Not only are they available to panels, but they are also available to dialog functions, in this case the CLIST. The value of a dialog variable is a character string. A dialog variable name follows the same rules as a SAS variable name, except that it cannot contain underscores.

Dialog variables are global within a dialog much like SAS macro variables within a SAS program. When the user fills in a value in an input variable field on a panel, a CLIST variable with the same name is automatically set to have that same value. In other words, for any given ISPF dialog variable, there is a corresponding CLIST variable with the same value. In addition, any panel can access that dialog variable value with variable input or output fields of the same name.

Sample Function

Now you are ready for a sample application that uses a dialog panel, a dialog function, and dialog variables. Figure 3 is the listing of a CLIST named PHONENUM. The sample application displays a panel into which the user enters a name and phone number and then presses ENTER. The CLIST then retrieves the values and keeps track of the number of names entered. The CLIST stores the values into a sequential file. Then, the data entry panel is redisplayed for another name and phone number. The procedure continues like this until the user presses the END PF key, after which the file is saved and the application is finished.

PROC 0
ALLOC F(info) DA(PHONENUM,DATA) OLD
OPENFILE INFO OUTPUT
SET RC=0
DO WHILE( (RC=0) )
    ISPEXEC DISPLAY PANEL(PHONENUM)
    SET RC=LASTCC
    IF RC=0 THEN DO
        SET INFO=INFO + 1
        PUTFILE INFO
        SET NAME=
        PUT FILE=ENTER(SPHONE)
        PUTFILE INFO
        SET PHONE=
    END
    CLOSFILE INFO
    FREE F(info)
END

Figure 3 CLIST dialog function PHONENUM

Using SAS/DMI Software

Initiating a SAS/DMI Session

A SAS/DMI session is functionally nothing more than a SAS session with which you have the ability to invoke ISPF dialog management services from the DATA step. In order to initiate the SAS/DMI session, you can use the CLISTS that are provided with the software. Alternatively, you can modify an existing SAS CLIST as described below:

• If you want to pass special SAS/DMI options when the session is initiated, insert a keyword parameter with a null value on the PROC statement.
• Instead of invoking SASCP, use the ISPF SELECT service to invoke the SAS/DMI program.

The CLIST modifications are listed in Appendix A.

The modified CLIST should be invoked from within an active ISPF session. Because all of the original CLIST options are maintained, you can execute any normal SAS programs with it. More importantly, you can now use ISPF dialog management services from the DATA step.

Invoking ISPF Services

The ISPLINK CALL subroutine is used to invoke ISPF dialog management services from the DATA step. The syntax of the ISPLINK CALL subroutine is

CALL ISPLINK(value1,...,value15);

where value1,...,value15 are variables, literals, or expressions to be passed as parameters to ISPF. The parameters being passed are documented in the IBM manual ISPF Dialog Management Services as the PL/I form of the parameters. If the parameter is supposed to be in the form of a fullword fixed binary number, use the PUT function with the PIB format to construct the parameter.

CALL ISPLINK(TBCLOSE,'TELBOOK',PUT(10,PIB4));

will close the table named TELBOOK with a ten percent padding space.

If you are using ISPF Version 2, you can use the ISPEXEC CALL subroutine with the command form for the service request.
Return Codes

Every time that you execute the ISPJNUK or ISPEXEC call subroutine, a return code is placed in the SAS numeric variable ISP_RC. Return codes are documented along with the services in ISPF Dialog Management Services. ISP_RC should be initialized with a RETAIN statement to facilitate testing of the return codes.

   RETAIN ISP_RC 0;

Special SAS/DMI Services

If you use DMI as the first parameter with the ISPLINK CALL subroutine, you have access to several special SAS/OMI services, documented in the SAS/OMI User's Guide. The TRACE/NOTRACE parameter is particularly useful to new SAS/OMI users.

   CALL ISPLINK('DMI','TRACE');

specifies that extended SAS/DMI information and error diagnostics be written to the SAS log.

ISPF Variable Services

If the ISPF dialog management services Invoked by your program need to use ISPF variables, your program has to use the VDEFINE and VDELETE services. The VDEFINE service gives ISPF access to SAS DATA step variables. The syntax of the VDEFINE service is

   CALL ISPLINK('VDEFINE',variable list,...);

Use the special parameter '_ALL_' to define all potentially valid SAS variables to ISPF.

   CALL ISPLINK('VDEFINE', '_ALL_');

Since ISPF variables cannot have underscores in their names, you can restrict the list of variables defined to ISPF by using some care when naming your variables.

The VDELETE service is used to remove the association between ISPF and SAS variables. Use the special parameter '_ALL_' to VDELETE all VDEFINEd variables:

   CALL ISPLINK('VDELETE', '_ALL_');

Sample DATA Step Construction

The dialog function unit for SAS/OMI software is the SAS DATA step. Figure 4 depicts a generalized DATA step that has a structure that you might find useful for developing your SAS/OMI functions.

Notice that the VDEFINE service executes once. The DATA step then enters a DO-loop that continues execution until an ending condition is satisfied. At that point, the VDELETE service is invoked and the DATA step is terminated.
The CLIST uses the ISPEXEC statement to invoke ISPF services. The SAS/DMI dialog uses the ISPLINK CALL subroutine to invoke ISPF services. The CLIST executes very quickly. The SAS/DMI dialog must wait for the SAS System to initialize before the SAS code listed above can be executed.

Given the choice of writing an ISPF dialog in CUST or as a SAS data step, the advantage in our first example clearly belongs to CLIST. There is less code to write and the function initializes much more quickly. However, if you have ever tried to write a statement like this in a CLIST,

```
IF INDEX(UCase('COMPANY'), 'INC.') > 0 THEN DO;
```

you can see one reason why dialogs might be developed with SAS/DMI software rather than with CLISTS; the DATA step is a far more powerful programming tool than the CLIST.

**Examples**

**An Interactive Mortgage Calculator**

The DATA step MORT function calculates parameters for a mortgage with equal periodic payments and a fixed interest rate compounded each period. This example uses an ISPF panel to receive information from a user and to repeatedly calculate mortgage payments.

**The Panel**

The panel for the function is displayed in Figure 6. The name of the panel is MORTPAN.

```
%----- MORTGAGE LOAN CALCULATOR --------
```

```
+ AMOUNT BORROWED => ...AMOUNT +
+ INTEREST RATE => ...RATE +
+ NUMBER OF YEARS => ...YEARS+
+ MONTHLY PAYMENT => ...PAYMENT
+ PRESS ENTER KEY TO EXIT
```

**Figure 6** ISPF panel MORTPAN.

- Variable input fields begin after the underscores and continue to the plus signs. Dialog variables AMOUNT, RATE, and YEARS are entered from this panel.
- An output variable, PAYMENT, is designated by the ampersand and is updated by the SAS DATA step.

The SAS program that will display the panel and calculate the monthly payments is shown in Figure 7.

```
DO WHILE(ISP_RC=0);
CALL ISPLINK('DISPLAY', 'MORTPAN');
IF ISP_RC=0 THEN DO;
  PAYMENT=MORT(AMOUNT, RATE, YEARS);
END;
CALL ISPLINK('DELETE', 'ALL');
RUN;
```

**Figure 7** Dialog function for calculating mortgage payments

**Variable Exchange Between CLISTS and the DATA Step**

Information can be passed from CLISTS to the SAS System by the following methods:

- A character string can be passed to the SAS System at invocation through SYSPRAM.
- A CLIST can write to a file with PUTFILE statements. A DATA step can subsequently read from the same file with INFILE and INPUT statements.
- A CLIST can terminate with different return codes, which can be retrieved through the automatic macro variable &SYSRC if the CLIST is executed with the %TSO statement, or through a SAS variable if the CLIST is executed with the DATA step TSO function.

Information can be passed from the SAS System to CLISTS by the following methods:

- A DATA step can write to a file with FILE and PUT statements. A CLIST can read the same file with GETFILE statements.
- Positional and keyword parameters on the CLIST can be set when the CLIST is executed from within the SAS System.

As you can see, information exchange between the SAS System and CLISTS can be cumbersome. However, ISPF variables, whether used by CLISTS or SAS/DMI software, are the same. They can be shared. The VPUT service places variables into a variable pool. If you are familiar with the SAS macro facility, the VPUT service is functionally similar to the %GLOBAL statement. The VGET service retrieves variables from the variable pools.

Examine the CLIST in Figure 8. It creates two variables, PREFIX and CPU. PREFIX is the TSO user prefix, which might be different from the TSO user id. CPU is the elapsed CPU time for current TSO session. Note that the automatic CLIST variable SYSCPU is available under TSO/E only. If the variable is not available, a null value will be placed in CPU. The VPUT service will place the variables into a pool, from which they can be retrieved.

```
DATA _NULL;
RETAIN AMOUNT PAYMENT RATE YEARS ISP_RC 0;
FORMAT AMOUNT PAYMENT DOLLAR10.2
INFORMAMT AMOUNT PAYMENT COMMA;
CALL ISPLINK('VERIFY', 'ALL');
DO WHILE(ISP_RC=0);
  CALL ISPLINK('DISPLAY', 'MORTPAN');
  IF ISP_RC=0 THEN DO;
    PAYMENT=MORT(AMOUNT, RATE, YEARS);
  END;
END;
CALL ISPLINK('DELETE', 'ALL');
RUN;
```

**Figure 8** 'USERID,STATS,CLIST'
The DATA step in Figure 9 executes the CLIST, retrieves the two dialog variables, and places their values in macro variables and in a SAS data set.

```
DATA STATS(KEEP=CPU PREFIX);
  LENGTH CPU PREFIX $ 8;
  CALL ISPLINK( 'VDEFINE', '....ALL')
    BUF....R=' CND( EXEC "USERIO. STATS .CLIST' , ) ,
    BUF....L=PUT( LENGTH( BUF....R) ,PIBiI.)
    CALL ISPLINK( 'SELECT',BUF....L,BUF....R)
    CALL ISPLINK( 'VGET',' (CPU PREFIX)');
  CALL SYMPUT( 'CPU' • CPU) ;
  CALL SYMPUT(' PREFIX' ,PREFIX);
  CALL ISPLINK( 'VDELETE', '....ALL');
RUN;
```

Figure 9 Function for exchanging variables with a CLIST

In order for the variables to be properly retrieved with the VGET service, the CUST has to be executed with the SELECT service rather than with the TSO function. The SELECT service requires a character variable that contains a command string and a fullword binary number that is the length of the command string. Notice the use of the PUT function to create the fullword binary number.

Notice that the DATA step can use the VPUT service and a CLIST can use the VGET service, allowing a CLIST to retrieve the values of SAS variables.

Summary

In summary, if you want to use the SAS DATA step as your programming language and would like to use ISPF dialog management services to create a dialog between the program and user, SAS/OMI software offers an easy-to-use interface.

SAS and SAS/OMI are registered trademarks of SAS Institute Inc., Cary, NC, USA.

Appendix A

Below is a listing of the beginning and the end of a modified version of the release 5.16 CLIST that allows the use of SAS/OMI software. CLIST option DMILOPT allows the passing of SAS/OMI invocation options. Instead of invoking SASCP, the CLIST executes the ISPF SELECT service. This CLIST must be invoked from within an active ISPF session or with the ISPSTART TSO command. Note that this CLIST allows only one SAS/OMI session to be active.

```
PROC OPTIONS.Add /* Options */
    WORK( / * Work library * /)
    OPTIONS( /* Options */)
    INPUT ( /* Input data set */) ;
    /* Below is the old SAS CLIST code, commented out */
    /* */
    /* */
    /* SASCP TASKLIB(LOAD SASLLIB $PLODSN $SORTLDSN) */
    /* ENTRY ('ENTRY) */
    /* OPTIONS(' 'OPTIONS' I */
    /* OPTIONS(' 'OPTIONS' I */
    /* FLUSH 'NO FLUSH */
    /* UTAE 'NOSTAT */
    /* UTAX 'NOSTACK */
    /* 'SHARE 'NOSHARE */
    /* Substitutions below to allow SAS/DFI session */
    /* SUBSASOPT='STR(ENTRY(iEBTRY) */
    /* FLUSH iNSHUSH UTAE 'NOSTAT */
    /* UTAX 'NOSTACK 'SHARE iNOSHARE */
    /* End of substitutions for SAS/DFI session */
/* */