Developing Dialog Manager Applications in z/OS

Developing Dialog Manager Applications in z/OS - Course Objectives

Upon successful completion of this course, the student, with the aid of the appropriate reference materials, should be able to:

1. Design and write applications using Dialog Manager services for the TSO environment, using REXX or CLIST as the programming language
2. Design panels and use panel language to display, accept, and process data placed in dialog variables; preprocess panels to improve performance
3. Provide for diagnostic and help messages for use when requested or when the user makes an error
4. Use menus to structure an application in a manner useful for the user
5. Use the dialog test and trace services to debug an application in development
6. Provide access to the PDF BROWSE and EDIT services, where appropriate in an application
7. Create, process, and display ISPF tables, including the ISPF Table Utility
8. Use ISPF Library Access services (formerly Library Management services)
9. Use the ISPF File Tailoring services
10. Use Pop-up windows for error or other processing
11. Decide whether to code an application in a procedures language or a compiled language
12. Create user-oriented commands using the Commands Table capability of ISPF
13. Create action bars with pull-down choices
14. Create keylists for use with multiple panels.

Developing Dialog Manager Applications in z/OS - Topical Outline

Day One

Scrollable fields
- Design issues
- Implementing scrollable fields
- The)FIELD section
- Scrollable Fields: An Example

Day Three

Menus and Debugging
- Command Processing
- Jump function processing
- Menus
- The SELECT service
- Syntax for TRANS and TRUNC in a menu
- Handling lower level requests
- Primary option menus
- Master application menus
- Menus, panels, and SELECT
- Dialog Test tracing services
- Computer Exercise: The Case Study, Add Menus 311

Some new services and tutorials
- Edit Models
- The CONTROL service
- Browse, Edit, and View services
- Browse, Edit, and View: Working with z/OS UNIX files
- Edit recovery interface
- Tutorials
- Computer Exercise: The Case Study: Add Tutorials Support 343

ISPF Tables
- Table types
- Tables and keys
- Defining tables - TBCREATE
- Row variables
- Extension variables
- Working with tables
- Working with rows
- Computer Exercise: The Case Study: Add a Table of Courses 403

Day Four

Table Display services
- Panels for table displays
- The)ATTR section for table display panels
- The)BODY section for table display panels
- The)MODEL section for table display panels
- The TBDISPL service
- Processing selected rows
- Table display variables
- TBSARG and TBSCAN
- Dialog Test and tables
- Computer Exercise: The Case Study: Process Rows in Courses Table 447

The ISPF Table Utility
- Introduction to the Table Utility
- Table List Formats
- Editing and Browsing Tables
- Re-Structuring The Table Display
- Sorting Tables
- Exporting and Importing Tables
- Table Utility Options
- Computer Exercise: Using the Table Utility 468

File Tailoring services
- The file tailoring process
- Skeletons
- File tailoring services - FTOPEN, FTINCL, FTCLOSE, FTERASE
- Computer Exercise: The Case Study: Using File Tailoring 517

More Library Access Services
- LMCOPY, LMMOVE, LMPRINT, LMRENAME, LMERASE
- Library access services to work with true libraries: LMCOMP, LMMFIND, LMMREN, LMMREP, LMMADD, LMMDEL, LMMSTATS, LMMLIST, LMMDISP, MEMLIST
- Library access services to work with lists of data sets: LMDINIT, LMDFREE, LMDLIST, LMDDISP
- DIRLIST - Display a z/OS UNIX Directory List

Day Five

Miscellaneous Topics
- Panel preprocessing
- Dialog Test: the DTEST command
- Obtaining data set information: QLIBDEF, QBASELIB, DSINFO
- Dialog Manager and commands
- Command tables
- Creating command tables
- Using command tables
- Computer Exercise: Create a User Command 599

Introduction to Common User Access (CUA)
- The CUA standard
- CUA panel formats
- Using Action Bars
- Working with pull-down menus
- Creating action bars using panel language
- Computer Exercise: Adding Action Bars 619

Keylists
- Keylists
- Dialog Tag Language (DTL)
- Defining keylists
- The ISPDTLC utility
- Using keylists
- Computer Exercise: Creating a Keylist 636

Final Topics
- Using Compiled Languages for Dialogs
- Tradeoffs
- Installing an ISPF Application
- Using ISPSTART
- Read-Only Profile Pool Extensions

Optional Exercise:
- The Case Study: Table to Sequential File [and Back] 646

Section Preview

- **Introduction To Dialog Manager**
  - **Dialog components**
  - **Dialog variables**
  - **Panel definitions**
  - **Data set requirements**
  - **Invoking Dialog Manager services from a CLIST or Exec**
  - **Invoking Dialog Manager services from a program**
  - **Dialog testing**
  - **Setting Up for Dialog Manager (Machine Exercise)**

Dialog Manager

- **Dialog Manager (ISPF) is an application development and execution tool that provides a number of services relating to displaying panels and messages, processing data, and so on**

- **Typically, a dialog is driven by a program written in CLIST, REXX, or a compiled language, such as COBOL, PL/I, Assembler, C, *etc.***

- **In any language, requests for Dialog Manager services are made by invocations of the ISPEXEC routine**

- **The Dialog Manager product is supported under z/OS TSO, OS/390 TSO, z/VM/CMS, and z/VSE**

Dialog

❒ **A dialog is an interaction between a person and a computer system**

- **Assisted by one or more functions written in CLIST, REXX, or some compiled language (or some combination of these)**
  - ✗ CLIST functions are only supported under TSO
  - ✗ REXX functions are supported under TSO and CMS
  - ✗ Program functions are supported in all environments
- **And that:**
  - ✗ Runs under the Dialog Manager
  - ✗ Uses Dialog Manager services

Dialog Components

❒ **The most commonly used components of a dialog are:**

Panels

- **Definitions of what a screen should look like, as well as some elementary processing of input commands and data**
- **Created by using a text editor**

Functions

- **Provide the bulk of logic in an application**
- **May be written in CLIST, REXX, APL2, PL/I, COBOL, Assembler, Pascal, FORTRAN, C**

Variable pools

- **Allow communication between panels, functions, and other Dialog Manager facilities**

Dialog Manager services

- **Support routines for invoking panels, functions, and other Dialog Manager facilities**
- **Invoked by ISPEXEC commands (CLISTs and REXX execs) or CALL to ISPEXEC or ISPLINK (compiled programs)**

A Dialog and Its Environment

❒ **A dialog itself may either**

- ♦ **Stand by itself under the Dialog Manager (as a turnkey system)**
- ♦ **Or it may be added to the standard list of applications in use by an installation**
  - ✗ For example, a dialog may be added as a choice on the ISPF/PDF Primary Option Menu

Dialog Structures

- **The structure of a dialog is described in terms of a hierarchy of functions and panels**
  - **Begin with display of a panel or execution of a function (CLIST, exec, or program) that ultimately displays a panel**
  - **User responds to a panel by entering data or commands**
    - **Pressing a PF key is the same as entering the data or command string assigned to the key and pressing <ENTER>**
  - **The dialog examines the user data or command and decides what to do next ...**

Dialog Structures, 2

❒ **Possible dialog actions on return from the display of a panel**

- ✗ Retry the panel display until valid information is gathered (possibly issuing an error message)
- ✗ Process information gathered as appropriate
- ✗ Handle user-defined commands
- ✗ Issue TSO or CMS commands
- ✗ Request ISPF services
- ✗ Invoke a subsequent panel or function
- ✗ Repeat this panel or function, in a loop
- ✗ Return to the previous panel or function in the hierarchy
- ✗ Take a side trip to a tutorial or HELP screen (then return)
- ✗ Terminate the dialog

Dialog Variables

❒ **Dialog variable names**

- ♦ **1 to 8 alphanumeric or national characters (A - Z, 0 - 9, $, #, @)**
- ♦ **First character of name must not be numeric**
- ♦ **APL2 names may not contain $, #, or @**
- ♦ **FORTRAN names may only be 6 characters maximum**
- ♦ **Dialog Manager system dialog variable names all begin with the letter Z (so do not begin your own dialog variable names with a Z)**

❒ **Dialog variable values**

- ♦ **Are always considered to be only character strings**
  - ✗ **Provisions exist for converting formats when placed into, or retrieved from, program functions**
- ♦ **Zero to 32K bytes long**

Panel Definitions

- ❒ **Panel definitions may be 80 to 160 characters wide**

- ❒ **Resulting display may not be wider than the screen being used**

- ❒ **Most common to edit and store panel definitions in libraries as 80-byte records**

  - ♦ **No sequence numbers (NUM OFF in ISPF/PDF editor)**

Sample Panel Definition

```
)BODY
%--------------------  Customer Information  -------------------
%COMMAND ===>_ZCMD
+
%
%Customer Number: &custno
+
+   Change request%===>_CHGREQ  + (New, Update, Examine, Delete)
+
+   Customer name%===>_CUSTNAM                          +
+
+   Mailing address:
+      Line 1  %===>_ADDR1                          +
+      Line 2  %===>_ADDR2                          +
+      Line 3  %===>_ADDR3                          +
+      City    %===>_CITY               +
+      State   %===>_ST+
+      ZIP     %===>_MAILCODE    +
+
+   Telephone numbers:
+      Main switchboard    %===>_SWITCHBD      +
+      Toll free no.       %===>_TOLLFREE      +
+
)END
```

Notes On The Panel Definition

❒ **"+", "%", and "_" are examples of attribute characters**

❒ **Each attribute character takes a position on the screen, even though the character itself does not display**

❒ **Input variable names immediately follow an underscore (_)**

- ♦ **The value that can be entered goes from the underscore to the plus sign (+)**
- ♦ **Input variable names do not show on the display**

❒ **All other items on this screen are called "text" fields**

- ♦ **Text fields may contain dialog variable names, preceded by an ampersand (&), in which case the current value in that variable will be displayed on the screen at the location shown**

Resulting Display

- **Assuming the current value in the variable CUSTNO is "DD87052", the display the user would see from the previous definition would look like this:**

```
-------------------  Customer Information  --------------------
COMMAND ===>

Customer Number: DD87052

Change request ===>_        (New, Update, Examine, Delete)

Customer name ===>

Mailing address:
   Line 1   ===>
   Line 2   ===>
   Line 3   ===>
   City     ===>
   State    ===>
   ZIP      ===>

Telephone numbers:
   Main switchboard     ===>
   Toll free no.        ===>
```

Data Set Requirements

❒ **For the Dialog Manager to find your panels, functions, messages, and so on, you must allocate certain DDnames:**

DDname	Description
ISPPLIB	Panel definition library
ISPSLIB	Skeleton library
ISPTLIB	Input table library
ISPTABL	Output table library
ISPMLIB	Messages library
ISPFILE	File tailoring output file
ISPLLIB	Link library (program function load modules)
ISPPROF	User profile tables
ISPILIB	Image library (GIFs, when running in GUI mode)
SYSPROC	CLIST library
SYSEXEC	REXX exec library

❒ **These DDnames must be allocated prior to invoking ISPF**

❒ **Usually there must be a concatenated list to include the installation libraries and your library for each type**

❒ **This allocation may be part of your logon procedure, or it may be done in a CLIST or a REXX exec**

❒ **Alternatively, a Dialog Manager service, LIBDEF, can be used for dynamic allocation (except for ISPPROF and any SYSPROC or SYSEXEC files) after ISPF invocation, from a dialog function**

Invoking Dialog Manager Services From a CLIST or Exec

❒ **From a CLIST or REXX exec, you request Dialog Manager services using the ISPEXEC command:**

ISPEXEC ***command-name*** ***parameters***

❒ **On completion of the service, the Dialog Manager places a return code value in the CLIST variable &LASTCC or the REXX variable RC**

Convention is:

- **"0" means service was completed successfully**
- **Other values may mean errors, or they may just be informative**
- **Possible values are documented as part of the description of each service**

Invoking Dialog Manager Services From a CLIST or Exec, 2

❒ **For example, to request allocation of your panel library, code:**

ISPEXEC LIBDEF ISPPLIB DATASET ID(*panel-lib-name*)

- **This places your panel library ahead of the system libraries allocated to ISPPLIB for your session**

❒ **Then, to request a panel display, issue:**

ISPEXEC DISPLAY PANEL(*panel-name*)

The Dialog Manager

- **Searches the dataset(s) pointed to by ISPPLIB**
- **Displays the panel (if the panel cannot be found, Dialog Manager terminates the request with a non-zero return code)**

The user then keys in data or commands and presses <ENTER>

- **Any data entered into input variables are stored into the appropriate dialog variables**

Control returns to the next statement in your CLIST or exec

Notes For Dialogs Written In REXX

❒ **Before you make your first request for ISPF services from an exec, you may issue**

ADDRESS ISPEXEC

- **In which case you may omit ISPEXEC on your subsequent Dialog Manager requests:**

DISPLAY PANEL(MYPAN1)

- **Also, then, commands directed to other host environments must be explicitly ADDRESSed to the name of the intended host environment:**

ADDRESS TSO ALLOCATE ...

❒ **Host statements with special characters (especially parentheses) need to be bounded by [single or double] quotes**

- **But make sure variables to be substituted are left un-quoted:**

"DISPLAY PANEL("VARX")"'

- **Here, <u>VARX</u> will have its value substituted before this request is passed to the Dialog Manager, while the rest of the string is simply passed on as is**

❒ **ISPEXEC statements are case insensitive, even in quotes**

❒ **In this course, we follow the convention used in the IBM manuals: omit ADDRESS, include ISPEXEC, and do not generally code quotes (except when needed for accuracy)**

Invoking Dialog Manager Services From a Program

- **If you write a function in a compiled language, you simply issue a CALL to the ISPLINK routine (the examples assume the appropriate variables have been initialized):**

Assembler

```
CALL  ISPLINK,(LIBDEF,ISPPLIB,DATASET,LIBNAME),VL
```

COBOL

```
CALL  'ISPLINK' USING LIBDEF ISPPLIB DATASET LIBNAME
```

PL/I

```
CALL  ISPLINK ('LIBDEF', 'ISPPLIB', 'DATASET',
                                    '(''libname'')';
```

C

```
ISPLINK (LIBDEF, ISPPLIB, DATASET, LIBNAME);
```

Invoking Dialog Manager Services From a Program, continued

❒ **Or, you can CALL ISPEXEC, using this format:**

```
CALL    ISPEXEC   (buf-len,buffer)
```

or, in C:

```
rc = ispexec(&buf_len,buffer);
```

✗ **Where** buf-len **is a fullword binary integer containing the length of the buffer**

✗ **And** buffer **contains the name of the service and its parameters, as if the command had been issued from a CLIST or exec**

❒ **Since FORTRAN only allows six character module names, you must use the synonyms ISPLNK and ISPEX for ISPLINK and ISPEXEC, respectively**

❒ **On completion of the service, the standard return code value is returned, using the linkages expected by the CALLing language**

Programming Notes

- CALLs to Dialog Manager services from program functions must pass parameters in a predetermined positional sequence

- For teaching purposes, we do not always list all possible parameters for a service call, and sometimes we may list parameters out of sequence

- When in doubt, check the "ISPF Dialog Developer's Guide and Reference" manual

- Especially note the use of apostrophes for languages that allow literals in CALL parameter lists, and how to indicate that positional parameters are omitted

- Check the Appendix to these materials for some simple examples of calling ISPF services from compiled programs and, at the end, some sources of information

Some Other Dialog Manager Services

- **Aside from displaying panels and invoking functions, some of the other services available from the Dialog Manager are:**
  - **Support for messages and tutorials**
  - **Create, display, and modify data in ISPF tables**
  - **Facilities for creating tailored JCL, program code, or data, based on pre-coded "skeleton" JCL, program code, or data and the current values in dialog variables**
  - **Interfaces to library access routines**
  - **Interfaces to ISPF/PDF Browse, View, and Edit services**
  - **Interfaces to command tables, to build your own commands**
  - **Support for Double Byte Character Set (DBCS) data, and other international requirements (punctuation for numeric values, date formats, and so on)**

- **ISPF also has the ability to run in "GUI mode"**
  - **This means dialogs can run on a workstation using the facilities normally associated with GUI interfaces (check boxes, drop down lists, push buttons, etc.)**
  - **However, we only discuss this ability tangentially in this course - it would be a distraction from our main goals, and this capability is no longer being enhanced**

Dialog Testing

- **Dialogs have many pieces to them that all need to fit together for a dialog to work properly**

- **ISPF/PDF has provided a facility for testing the individual pieces of a dialog as they are written, and for debugging errors in existing dialogs**

- **ISPF/PDF option 7 is usually Dialog Test**

  - **This is a Primary Option Menu**

    - **Which means once you are in it, you can not "jump" out of it to some other option outside of Dialog Test:**

      - For example, if you specify ===> =3.4 on a panel under option 7, you will be sent to Dialog Test option 3 suboption 4 (an error) instead of to PDF option 3.4

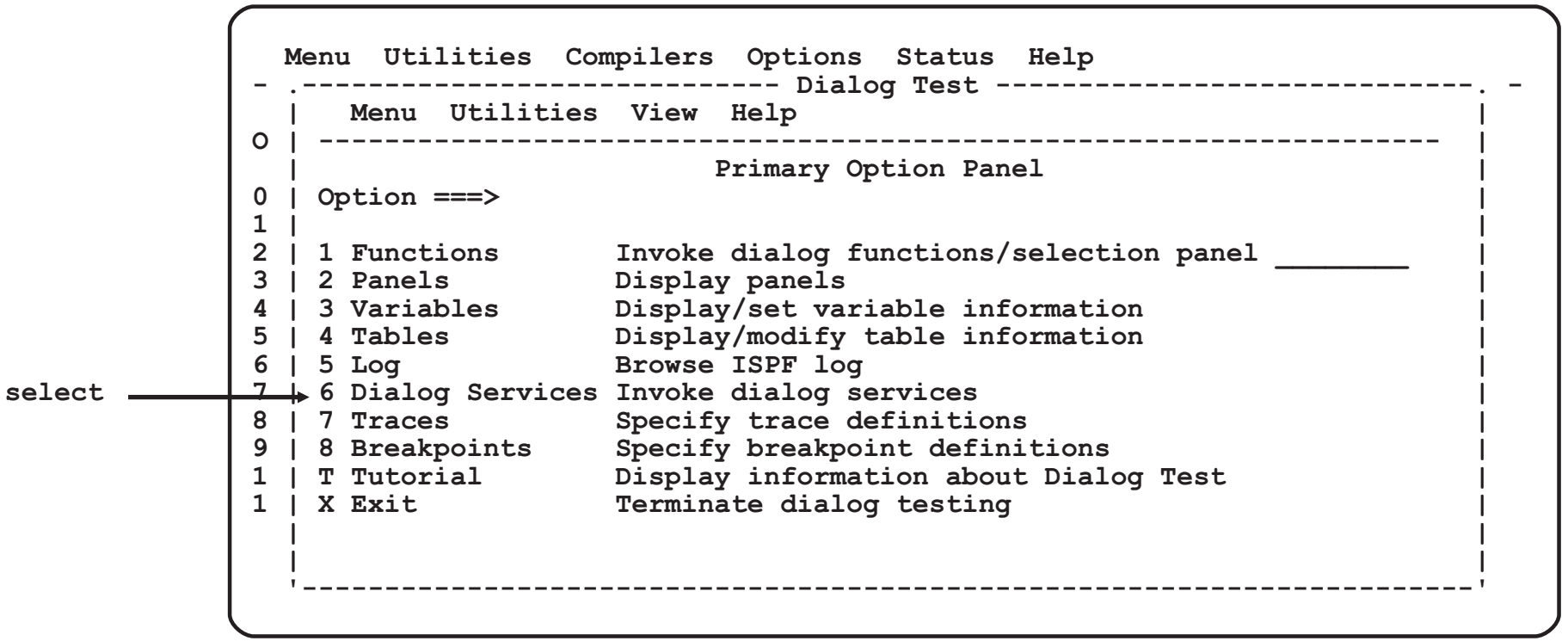
    - **When you are in a suboption of Dialog Test, entering** =X **on the command line takes you to the standard Primary Option Menu, while "RETURN" takes you to the Dialog Test Primary Option Menu**

- **If you will be using Dialog Test, make sure your LOG default is not "2" (Delete)**

  - **Dialog Test writes trace and debugging type information to the log, and you may want to put data out there too**

Dialog Test Primary Option Menu

❒ **Here is a typical menu for Dialog Test:**



Dialog Test Option 6: Requesting Dialog Services

- Filling in the command field here, you do not need to prefix it with "ISPEXEC"

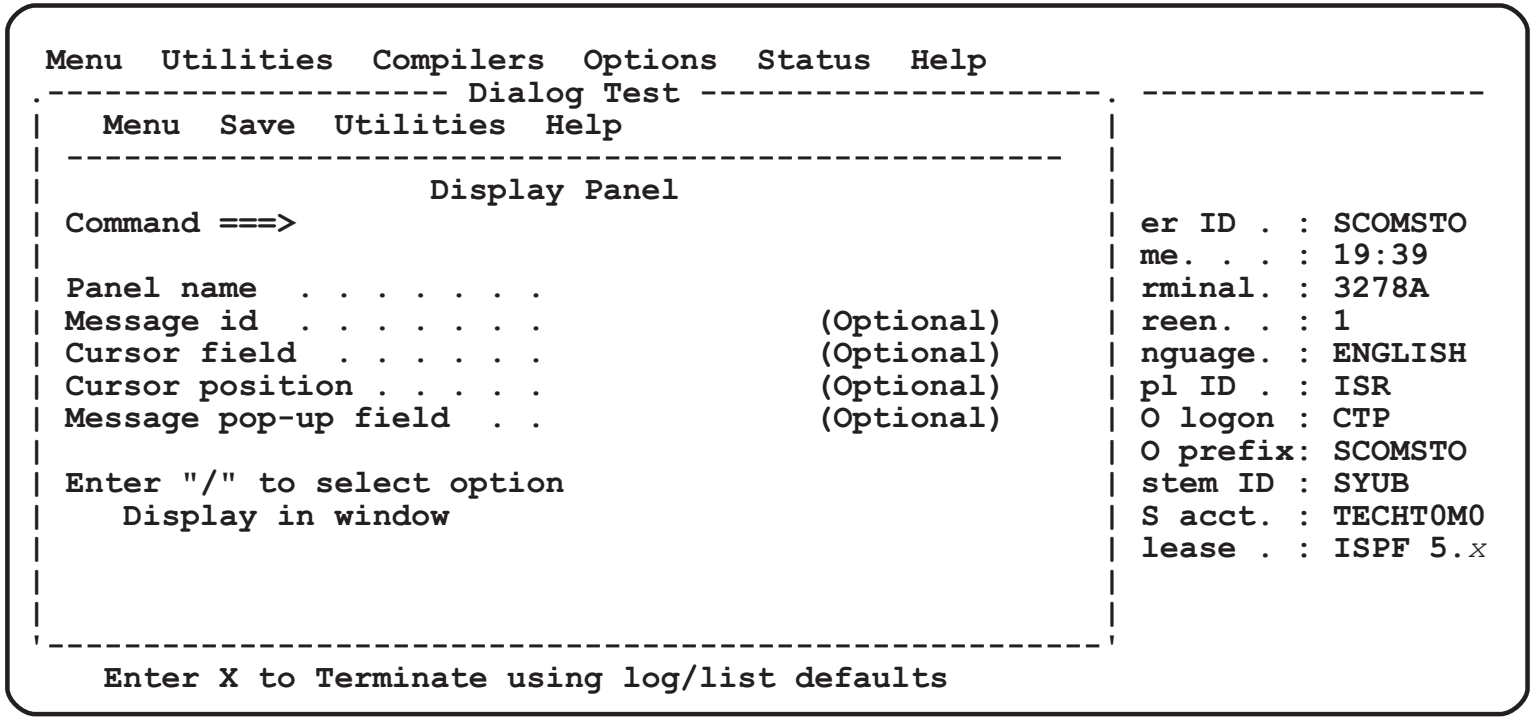
```
  Menu  List  Mode  Functions  Utilities  Help
------------------------------------------------------------------------------
                              Invoke Dialog Service
Command ===>

Enter dialog service and its parameters:
===>


Place cursor on choice and press enter to retrieve command.

=>
=>
=>
=>
=>
=>
=>
=>
=>
=>
```

- For example, you might enter:

```
===> LIBDEF ISPPLIB DATASET ID('panel-library-name')
```

Dialog Test Option 2: Display A Panel

❒ **You can request a panel be displayed, even if the panel is not in the context of a dialog:**



- **Just key in your panel name, and the panel will be displayed**
- **If you have an error, you will get a diagnostic message**
- **If you have no logic associated with this panel, you exit with an "END" command (PF3)**

Computer Exercise: Setting Up for Dialog Manager

This exercise is intended to help you set up for subsequent Dialog Manager work in the class. There will be more setup work later, also, but this will give you a good start.

Step 1: Setting Up Libraries

First, you need to run A810STRT, a supplied REXX exec that runs a dialog that will prompt you for the high level qualifier (HLQ) you want to use for your data set names; the exec uses a default of your TSO id, and that is usually fine; it also asks if you intend to code your labs in CLIST or REXX; then it creates data sets and copies members you will need.

From ISPF option 6, on the command line enter:

```
===> ex '__________.train.library(a810strt)' exec
```

The files created are:

userid.TR.PANELS	Panel Definitions
userid.TR.MESSAGES	Message Definitions
userid.TR.PEOPLE	data file used in later labs
userid.TR.TABLES	for table handling labs
userid.TR.EXEC	REXX Functions (if using REXX for exercises)
OR	
userid.TR.CLIST	CLIST Functions (if using CLIST for exercises)

Step 2: Creating a Panel Definition

In your panel library, create a member called SAMPL01 based on the definition on page 16 of the student handout.

Step 3: Testing a Panel Definition

Use Dialog Test to allocate your panel library ahead of the system panel libraries. Then display your panel definition using option 2 of Dialog Test.

Computer Exercise: Setting Up for Dialog Manager, p.2.

Optional Step: Start a Dialog

In your EXEC or CLIST library, create a function called DIALOG01 that only contains two commands:

1) A LIBDEF request to put your panel library ahead of the system panel library

2) A request to display your panel (Hint: see page 21)

Under option 6 (TSO, not under Dialog Test), execute your procedure.