The Trainer's Friend, Inc.

6790 E. Cedar Ave., Suite 201
Denver, Colorado 80224
U.S.A.

Telephone: (303) 355-2752

E-mail trainers@trainersfriend.com
Internet www.trainersfriend.com

Developing Dialog Manager Applications in z/OS

Student Materials

The following terms that may appear in these course materials are trademarks or registered trademarks:

Trademarks of the International Business Machines Corporation:

AIX, BookManager,CICS, DB2, DRDA, DS8000, ESCON, FICON, HiperSockets, IBM, ibm.com, IMS, Language Environment, MQSeries, MVS, NetView, OS/400, POWER7, PR/SM, Processor Resource / Systems Manager, OS/390, OS/400, Parallel Sysplex, QMF, RACF, Redbooks, RMF, RS/6000, SOMobjects, S/390, System z, System z9, System z10, VisualAge, VTAM, WebSphere, z/OS, z/VM, z/VSE, z/Architecture, zEnterprise, zSeries, z9, z10

Trademarks of Microsoft Corp.: Microsoft, Windows

Trademarks of Micro Focus Corp.: Micro Focus

Trademark of American National Standards Institute: ANSI

Trademarks of America Online, Inc.: America Online, AOL

Trademarks of Quercus Systems: Personal REXX, REXXTERM

Trademark of Chicago-Soft, Ltd: MVS/QuickRef

Trademark of Phoenix Software International: (E)JES

Trademark of Triangle Systems: IOF

Trademarl of Syncsort Corp.: SyncSort

Trademark of CA: Endevor

Trademark of Serena Software International: ChangeMan

Registered Trademarks of Institute of Electrical and Electronic Engineers: IEEE, POSIX

Registered Trademarks of Corel Corporation: Corel, CorelDRAW, Corel VENTURA

Registered Trademark of Oracle Corporation: Oracle

Registered Trademark of The Open Group: UNIX

Trademarks of Sun Microsystems, Inc.: Java, EmbeddedJava, Enterprise JavaBeans, EJB, Java Naming and Directory Interface, JavaBeans, JavaOS, JavaScript, JavaServer, JavaServerPages, JSP, JDBC, JDK, JVM, J2EE, Sun Microsystems, 100% Pure Java

Registered Trademark of Linus Torvalds: LINUX

Registered Trademark of Unicode, Inc.: Unicode

Trademarks held on behalf of World Wide Web Consortium: W3C, XHTML, XSL, WebFonts

Trademark of Object Management Group: CORBA

Trademarks of Apple Computer: QuickTime, Safari

Trademarks of Adobe Systems, Inc.: Macromedia, PDF, Shockwave, Flash

Trademark of The Eclipse Foundation: Eclipse

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

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
- Computer Exercise: Setting Up for Dialog Manager 31

Panel Definitions
- The LIBDEF Service
- User libraries
- Defining panels
- Headers and sections in a panel definition
- Panel design
- Attribute characters
- Panel definition - the)BODY section
- The)END section
- Panel layout concerns
- DISPLAY services
- Computer Exercise: Defining Panels 74

Dialog Variables and pools
- Applications
- Variable Pools
- Function pools for execs and CLISTs
- The shared pool
- The application profile pool
- Z variables
- The system profile pool
- Variable pools relationships
- System variables
- Variable services: VGET, VPUT, VERASE

Brief TSO Review
Brief REXX Review
Brief CLIST Review

Developing Dialog Manager Applications in z/OS - Topical Outline, p.2.

Developing Dialog Manager Applications in z/OS - Topical Outline, p.3.

Developing Dialog Manager Applications in z/OS - Topical Outline, p.4.

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

Developing Dialog Manager Applications in z/OS - Topical Outline, p.5.

Day Five

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 <u>dialog</u> is an interaction between a person and a computer system**

- ♦ **Assisted by one or more <u>functions</u> 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 <u>display of a panel</u> or <u>execution of a function</u> (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 <u>names</u>**

- ♦ **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 <u>values</u>**

- ♦ **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 (+)**

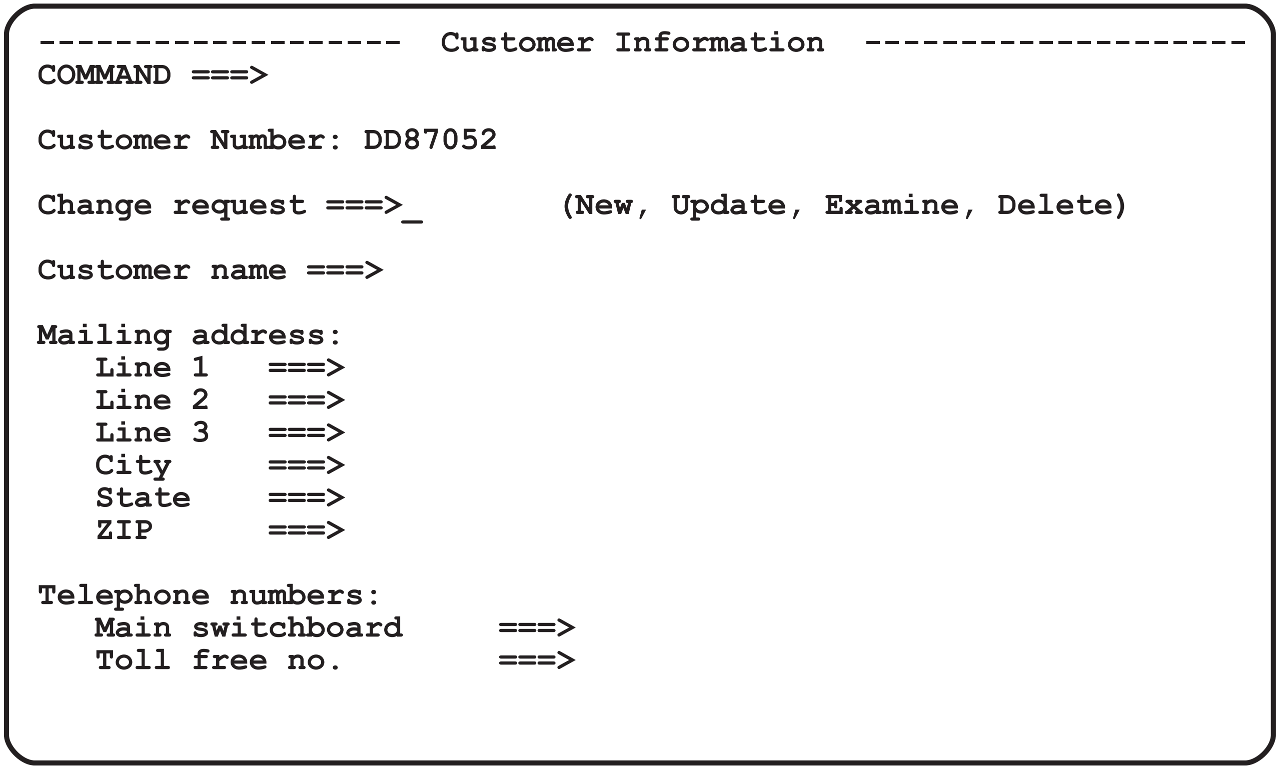
- ♦ **Input variable names do not show on the display**

❒ **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) <u>after</u> 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, VARX 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 <u>positional</u> 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**

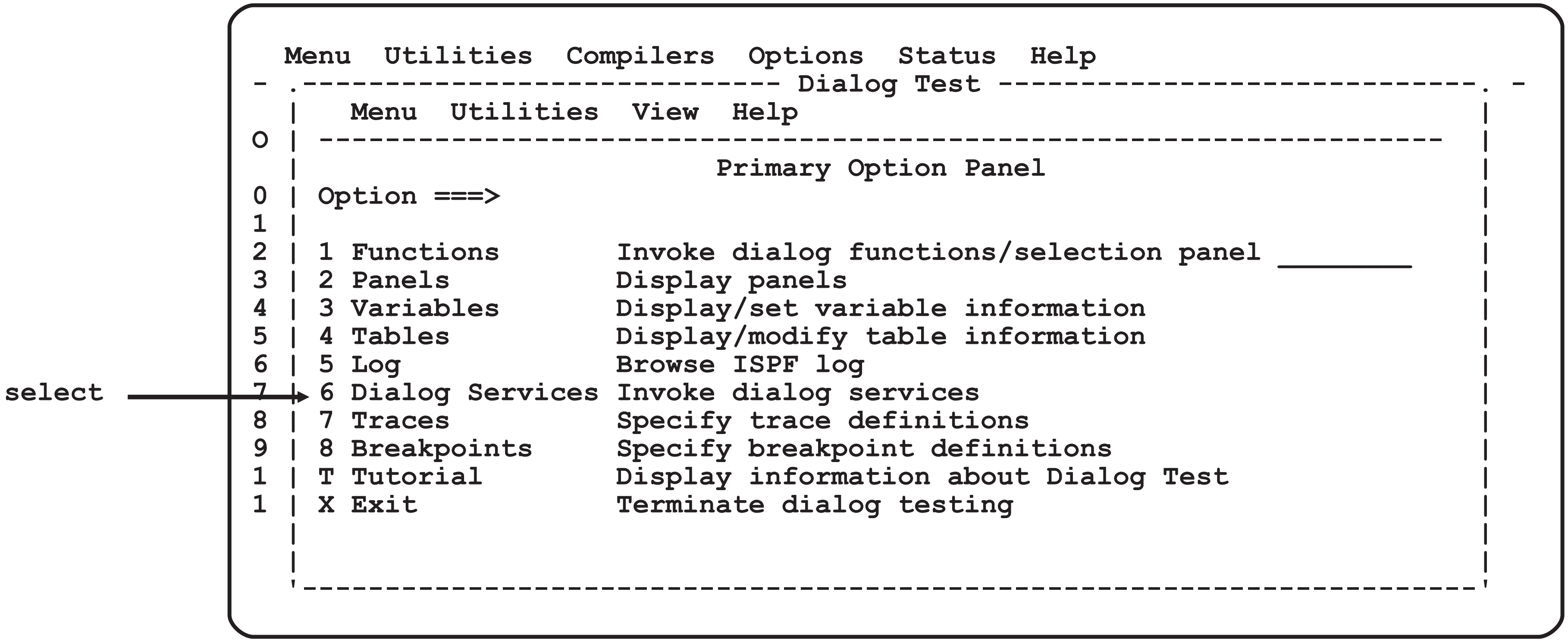
- ♦ **This is a Primary Option Menu**
  - ✗ **Which means once you are in it, you can not "jump" out of it to some other option outside of Dialog Test:**
    - ✗ For example, if you specify ===> =3.4 on a panel under option 7, you will be sent to Dialog Test option 3 suboption 4 (an error) instead of 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

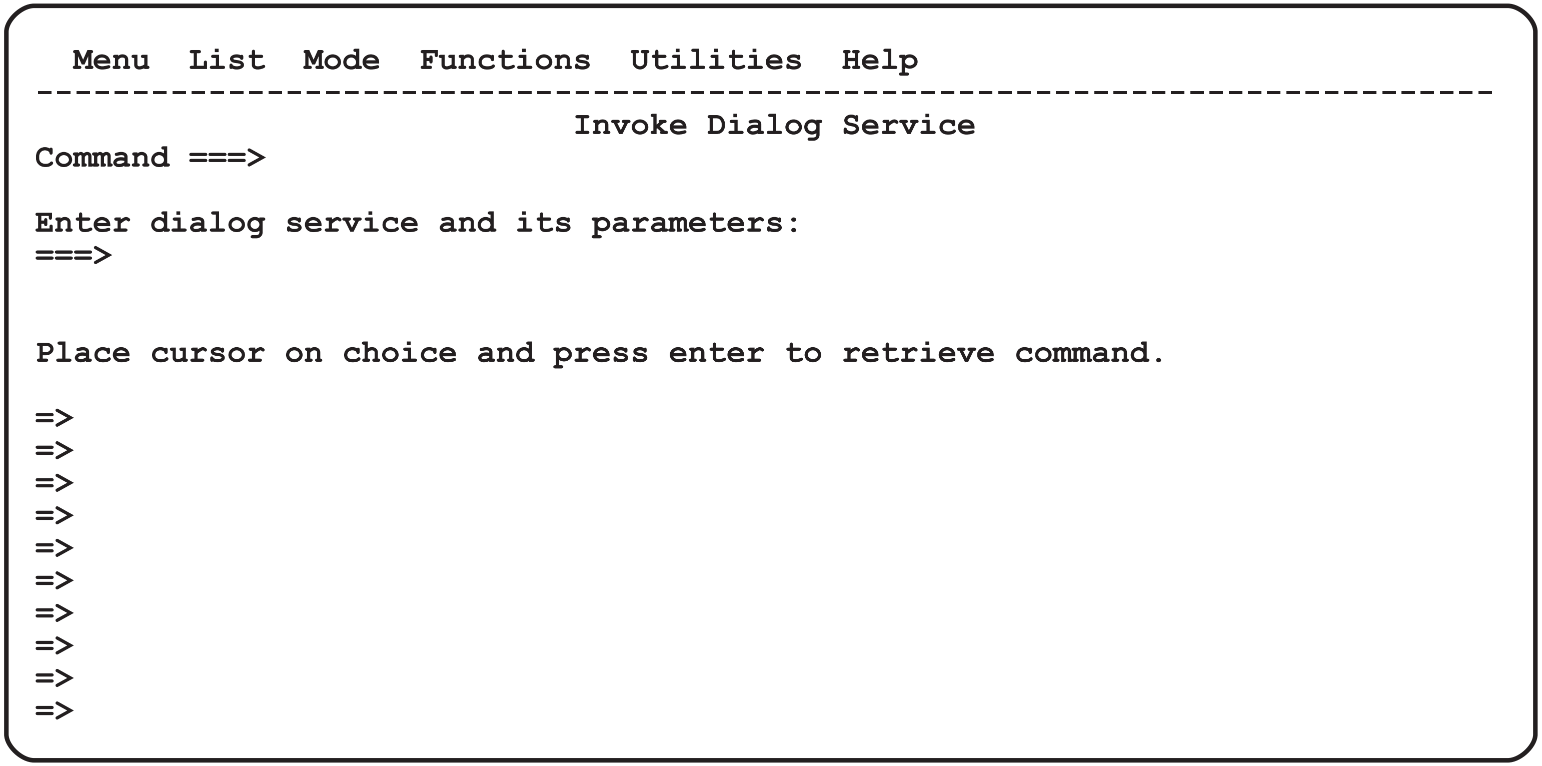
❒ **Here is a typical menu for Dialog Test:**



```
  Menu  Utilities  Compilers  Options  Status  Help
- .----------------------------- Dialog Test --------------------------------. -
  |   Menu  Utilities  View  Help                                             |
O | --------------------------------------------------------------------      |
  |                         Primary Option Panel                              |
0 | Option ===>                                                               |
1 |                                                                           |
2 | 1 Functions       Invoke dialog functions/selection panel ________        |
3 | 2 Panels          Display panels                                          |
4 | 3 Variables       Display/set variable information                        |
5 | 4 Tables          Display/modify table information                        |
6 | 5 Log             Browse ISPF log                                         |
7 | 6 Dialog Services Invoke dialog services                                  |
8 | 7 Traces          Specify trace definitions                               |
9 | 8 Breakpoints     Specify breakpoint definitions                          |
1 | T Tutorial        Display information about Dialog Test                   |
1 | X Exit            Terminate dialog testing                                |
  |                                                                           |
  |                                                                           |
  '---------------------------------------------------------------------------'
```

Dialog Test Option 6: Requesting Dialog Services

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



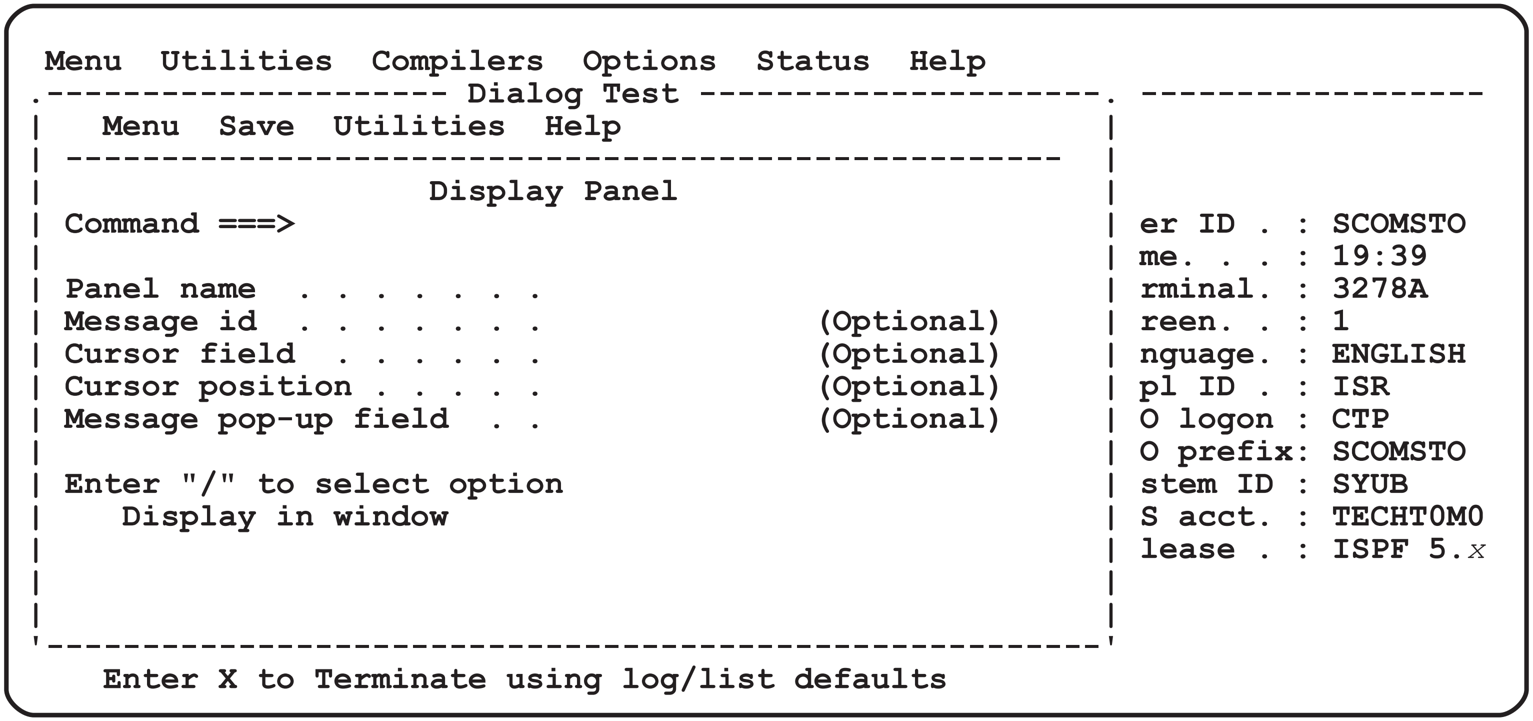
```
  Menu  List  Mode  Functions  Utilities  Help
----------------------------------------------------------------------------
                            Invoke Dialog Service
Command ===>

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:



```
 Menu  Utilities  Compilers  Options  Status  Help
.-------------------- Dialog Test ---------------------.  ------------------
|  Menu  Save  Utilities  Help                          |
| ----------------------------------------------------  |
|                  Display Panel                        |
| Command ===>                                          | er ID . : SCOMSTO
|                                                       | me. . . : 19:39
| Panel name  . . . . . . .                             | rminal. : 3278A
| Message id  . . . . . . .             (Optional)      | reen. . : 1
| Cursor field  . . . . . .             (Optional)      | nguage. : ENGLISH
| Cursor position . . . . .             (Optional)      | pl ID . : ISR
| Message pop-up field  . .             (Optional)      | O logon : CTP
|                                                       | O prefix: SCOMSTO
| Enter "/" to select option                            | stem ID : SYUB
|    Display in window                                  | S acct. : TECHT0M0
|                                                       | lease . : ISPF 5.x
|                                                       |
|                                                       |
'-------------------------------------------------------'
   Enter X to Terminate using log/list defaults
```

- ♦ 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.

Section Preview

❒ **Panel Definitions**

- ♦ **LIBDEF service**
- ♦ **User libraries**
- ♦ **Defining panels**
- ♦ **Headers and sections in a panel definition**
- ♦ **Panel design**
- ♦ **Attribute characters**
- ♦ **Panel definition - the)BODY section**
- ♦ **The)END section**
- ♦ **Panel layout concerns**
- ♦ **Display services**
- ♦ **Defining panels (Machine Exercise)**

LIBDEF Service

❐ The "LIBDEF" ISPF service allows you to specify application level data sets to be concatenated ahead of the ISPF system level data sets

Syntax

```
ISPEXEC  LIBDEF    lib-ddname

         [ID(dsname-list)|ID(ddname)]

         [DATASET|LIBRARY]

         [COND|UNCOND|STACK|STKADD]
```

Where

lib-ddname	Used for application level
ISPMLIB	Message library
ISPPLIB	Panel library
ISPSLIB	Skeleton library
ISPTLIB	Table input library
ISPTABL	Table output library
ISPFILE	File tailoring output library
ISPLLIB	Load module library
ISPILIB	Image library (for ISPF GUI mode)
xxxxxxxx	Generic library (may be used for table input, table output, or file tailoring output)

Other LIBDEF Parameters

DATASET

- ♦ **Indicates the "ID" field contains a list of data set names**

LIBRARY

- ♦ **Indicates the "ID" field contains the DDname of a previously ALLOCATEd file**

COND

- ♦ **Do the LIBDEF only if there is not already some active file of this type**

UNCOND

- ♦ **Do the LIBDEF unconditionally (replace any pre-existing LIBDEF for this lib-ddname); this is the default**

STACK

- ♦ **Save the current LIBDEF setting in a stack, then do the LIBDEF unconditionally**

STKADD

- ♦ **Concatenate the new library of this type ahead of the current library (ies); only valid if DATASET also specified**

❐ **If DATASET and LIBRARY are omitted, or if ID has a null value, ISPF is to remove any data set already associated with this lib-ddname**

- ♦ **If the previous LIBDEF value was STACKed, then it is restored**

Return Codes From LIBDEF

0	**—**	**Successful completion**
4	**—**	**Attempt to remove library, but library did not already exist**
8	**—**	**COND specified, and application library was already established; current LIBDEF was not established**
12	**—**	**Invalid** *lib-ddname* **specified**
16	**—**	**Invalid contents of ID parameter**
20	**—**	**Severe error**

LIBDEF: Special Cases

- Two additional, special parameters apply to the link (load module) library, ISPLLIB, only:

"EXCLDATA"	instead of "DATASET"
"EXCLLIBR"	instead of "LIBRARY"

- When either of these is specified, the search for ISPF program and command load modules will only consider the ISPLLIB dataset(s), ignoring any user and ISPF system libraries

    - Although the Link Pack Area and Linklist data sets will still be searched for programs invoked dynamically by these programs (and the ISPLLIB will not be searched)

- To find out the status of LIBDEFs, one can issue the ISPLIBD command from any command line

    - This brings up a panel displaying the current status of all LIBDEFs

User Libraries

- ❒ When you establish application level libraries for your dialog or application, individual users may wish to have their private libraries searched ahead of any LIBDEFed libraries

  - ♦ Or you may want each user to have their own private library of one or more types, to tailor the application even more for each individual user

- ❒ This can be accomplished by using TSO "ALLOCATE" commands prior to starting ISPF

  - ♦ Could be in the TSO logon procedure, for example

  - ♦ Or in a CLIST or exec you invoke prior to starting ISPF

- ❒ A different set of DDnames must be used for these ALLOCATE commands:

DDname	For user level
ISPMUSR	Message library
ISPPUSR	Panel library
ISPSUSR	Skeleton library
ISPTUSR	Table library
ISPTABU	Table output library
ISPFILU	File tailoring output library
ISPLUSR	Load module library
ISPIUSR	Image library

User Libraries, 2

❒ **Allocation of these DDnames prior to starting ISPF, and the presence or absence of a LIBDEF service request, interact to determine the order of library search:**

- ♦ **If a user library has been ALLOCATEd, with no LIBDEF, the user library allocation is ignored!**

- ♦ **If a user library is allocated, and a LIBDEF request is made for the same library type, the search order is**

    - ✗ User library

    - ✗ LIBDEF library

    - ✗ System library

- ♦ **If no user library is allocated, LIBDEF causes library search to use the LIBDEF library before the system library**

Panel Types

Selection Panel

- ♦ **A** MENU**; Select from a list of options**

Parameter Entry Panel

- ♦ **A Fill-in-the-blanks screen; for example, specify a dataset name for browse or edit**

Scrollable Data Display

- ♦ **Table Format (Data arranged in rows and columns)**
  - ✗ **For example, a member selection list for members in a library**
- ♦ **Dataset Format (Displays one row per record)**
  - ✗ **For example, PDF edit and browse**

Information-only Panels

- ♦ **Full screen of text, such as tutorial or help panels**

Defining Panels

❒ **A panel is defined by coding a series of lines in a member of a library used to store panel definitions: a panel library**

❒ **These lines defining a panel are written in what is called "Panel Language" or "Panel Definition Language" in the literature**

- ♦ **There is an alternative language for defining panels called "Dialog Tag Language" or "DTL"; DTL is not covered in this course**

    - ✗ Except for a short section near the end needed for creating keylists

❒ **A panel definition can have up to 17 kinds sections**

❒ **Each section is designated by a header of a right parenthesis, ")", followed immediately by the name of the section**

❒ **The right parenthesis must be in column 1 ...**

Headers and Sections in a Panel Definition

Header	Designates Section
)INEXIT	*Identify an exit to process panel source on the fly
)CCSID	*Coded Character Set IDentifier - foreign language support
)PANEL	Panel information - any keylist id; forces CUA mode
)ATTR	Attribute - Define attribute characters for display (these attribute characters may then be used in the BODY section)
)ABC	Action Bar Choice - defines action bar components
)ABCINIT	Action Bar Choice Initialization - required if ")ABC" specified
)ABCPROC	Action Bar Choice Processing - panel processing for action bar
)BODY	Panel Body - The representation of the screen appearance
)MODEL	Model Lines - For table displays
)AREA	*Scrollable Area Definition - for non-table scrollable areas
)INIT	Initialization - Pre-DISPLAY processing
)REINIT	Reinitialization - Processing prior to reshowing a panel
)PROC	Process - Describes how to process any input variables entered on the panel
)FIELD	Scrollable field section; define a field as scrollable
)HELP	*Help section - used for field-level help
)PNTS	*Point and shoot - declarations for all point and shoot fields
)LIST	*List - specifies list to build on panel
)END	End - Signifies the end of the panel definition

Headers and Sections in a Panel Definition — Notes

❒ Only the "BODY" and "END" sections are required

❒ Sections with asterisks (*) at head of their description are not covered in this course

❒ If a section from)INEXIT to)PROC is included in a panel definition, it must appear in the relative order listed on the previous page

- ♦ Section types)FIELD,)HELP,)LIST, and)PNTS may occur in any order as long as they appear after sections)CCSID through)PROC

- ♦ And)END must always be the last section in a panel definition

Panel Design

❒ There are four types of panels supported by ISPF, and the type of panel implies the necessary sections needed in a panel definition

❒ Also, there are some conventions about panel layout we should follow, in the interest of providing a consistent, familiar environment for the user

❒ At this point, we are looking at defining basic parameter entry / data display (information only) panels

❒ All panel definitions require)BODY and)END sections, and many use the)ATTR section, so these sections are our first concern ...

Attribute Characters

❒ **Each field on a screen must be assigned a collection of attributes that indicate data types (Input, Output, Text), display characteristics (Intensity, Color, Highlighting) and so on**

❒ **We do this by assigning collections of attributes to specific characters**

- ♦ **And then placing these attribute characters in the panel body, thus assigning attributes to the following text or variable fields**

❒ **The characters to use may be any alphanumeric or special character or two-digit hex number, except the ampersand (&) and the following hex values: 00, 0E, 0F, 40**

- ♦ **Up to 127 attribute characters may be defined for a single panel**
- ♦ **Recommend: Do not use characters that could conflict with panel text**
  - ✗ **For example, if you will be using "===>" for arrows, do not use the equals sign ("=") or greater than sign (">") for attribute characters**

Attribute Characters, 2

Sample Attribute Section

```
)ATTR
      $    TYPE(INPUT)        INTENS(LOW)    CAPS(ON)
      @    TYPE(OUTPUT)       CAPS(OFF)      COLOR(PINK)
```

- **An attribute character applies to all following data (even on subsequent lines) until another attribute character is encountered (or end of screen)**

- **Each attribute character occupies a location (character position) on the screen (although the character will not display)**

- **The most useful attribute options are discussed on the following pages**

Programming Note

- **All panel language statements, variable names, keywords, and keyword values can be entered in uppercase or lowercase**

  - Literal values are stored as entered; other elements are translated to uppercase before processing

TYPE(TEXT|INPUT|OUTPUT)

❒ **Field TYPE basic attributes are**

TEXT

- ♦ **Display on screen exactly as shown in panel definition; except, substitute values for symbolic variables**
  - ✗ **A symbolic variable is a dialog variable name preceded by an ampersand (*e.g.*: &CUSTNO)**

INPUT

- ♦ **Single variable; current value is displayed; user may type over**

OUTPUT

- ♦ **Single variable; current value is displayed; user may not type over**

❒ **Additional TYPE attributes are used for Dynamic Areas and for CUA panel elements; these topics are not discussed in this course**

TYPE(TEXT|INPUT|OUTPUT), 2

- **User may not type over or change TEXT or OUTPUT fields (we say the field is protected)**

- **On a panel definition, a field defined with an attribute character indicating a TEXT field will display exactly as coded; symbolic variables will have their values substituted, with trailing blanks stripped**

- **A field defined with an attribute character indicating an INPUT field must be immediately followed by a dialog variable name, with no intervening ampersand (*e.g.:* _EMPID if "_" is an INPUT attribute character)**

  - **No text may be included for an INPUT field type; ISPF initializes the display with the current value in the variable, and the user may overtype or enter new data**

- **A field defined with an attribute character indicating an OUTPUT field must be immediately followed by a dialog variable name, with no intervening ampersand**

  - **No text may be included for an OUTPUT field type; ISPF initializes the display with the current value in the variable, but the user may not overtype or enter new data**

INTENS(HIGH|LOW|NON)

❐ **Field INTENSITY attributes are**

HIGH

- ♦ **Display with high intensity (if supported on terminal)**

LOW

- ♦ **Display with low intensity (if supported on terminal)**

NON

- ♦ **Non-display; do not display this field**

Dynamic Attribute Specification

❐ **Many of these field attributes may be specified using symbolic variables**

For Example:

```
INTENS(&XYCH)
```

❐ **Then dynamically changing the value in the symbolic variable changes the characteristics of the attribute character:**

```
&XYCH = LOW

&XYCH = NON
```

❐ **In this instance, you may selectively display or not display certain lines on a panel, depending on the circumstances**

- ♦ **For example, display some text only if a particular option is requested**
- ♦ **Or display some text only for novices**

❐ **Exception: TYPE(TEXT) may not be specified as a result of symbolic variable substitution**

Dynamic Attribute Specification, Continued

❒ The actual changing of the symbolic variable values may be done in the ")INIT" section of the panel definition (as shown on the previous page)

❒ Or it may be done in the function (EXEC, CLIST, or program) prior to displaying the panel

- ♦ In REXX: xych = low
- ♦ In CLIST: SET &XYCH = LOW

CAPS(ON|OFF|IN|OUT)

❐ Field CAPS attributes are

ON

♦ Translate data to upper case before display and (if an INPUT field) when entered

OFF

♦ Display and accept data as is (do not translate to upper case)

IN

♦ Display as found in variables, but force to upper case when data entered

OUT

♦ Display all variables in upper case; accept input data as is

❐ For a CLIST function, CONTROL ASIS must be specified for "OFF", "IN", or "OUT" to take effect

❐ Default is CAPS(ON) for most situations

HILITE(USCORE|BLINK|REVERSE)

❒ **Field HILITE attributes are**

USCORE

- ♦ **Underscore the data (display data as underlined)**

BLINK

- ♦ **Display data blinking**

REVERSE

- ♦ **Display data in reverse video**

❒ **These are useful for drawing attention to a field**

- ♦ **Especially when a user enters bad data, you can dynamically alter the attribute for that field to have one of these attributes to emphasize which field is in error**

❒ **The default is to have no highlighting**

❒ **If terminal does not support extended highlighting, this parameter is ignored (or forces high intensity in some cases)**

PAD(NULLS|USER|char)
or
PADC(NULLS|USER|char)

❒ **Field padding attributes ...**

- **Are optional; but if one is specified, can not specify the other**
- **Are not valid for TEXT fields**
- **Specify what character to initialize a field with when first displayed**

❒ **<u>PADC</u> character is used conditionally: only if field is initially blank; default is PAD(USER) for INPUT fields, PAD(" ") (blank) for OUTPUT fields**

<u>NULLS</u>

- **Null padding character**

<u>USER</u>

- **Allow user to specify PAD character (using PDF option 0.1)**

<u>char</u>

- **Enter a literal character (*e.g.:* PAD("*"))**

SKIP(ON|OFF)

❒ Field SKIP attributes are valid only for TEXT and OUTPUT fields (protected fields)

❒ The possible values are

ON

♦ Cursor skips over field when a character is entered into the last position of the preceding INPUT field

OFF

♦ Cursor does not automatically skip over this field

❒ SKIP(ON) is user-friendly for screens that have lots of TEXT and OUTPUT fields interspersed with INPUT fields

COLOR(value)

❒ **Field COLOR attributes are chosen from**

- ♦ **WHITE|RED|BLUE|GREEN|PINK|YELLOW|TURQ**

❒ **If a terminal does not support color, the color value is used to imply a default intensity**

- ♦ **BLUE, GREEN or TURQUOISE imply LOW intensity**
- ♦ **WHITE, RED, PINK or YELLOW imply HIGH intensity**
- ♦ **Depending on terminal type, if INTENSITY is explicitly specified, COLOR may be ignored**

❒ **If a terminal does support color, and COLOR is not specified, colors are chosen implictly based on field TYPE and INTENSITY:**

FIELD TYPE	INTENSITY	COLOR
TEXT/OUTPUT	HIGH	WHITE
TEXT/OUTPUT	LOW	BLUE
INPUT	HIGH	RED
INPUT	LOW	GREEN

Options Supported But Not Discussed Here:

❒ Light Pen / Cursor Select Key feature

❒ Text justification

❒ Numeric only input

❒ Disallow jumping from input fields

❒ Double Byte Character Set (DBCS) parameters

❒ Dynamic Areas

❒ Graphic Areas

❒ CUA Panel Elements

❒ GUI mode elements

❒ Rules for defaults and mutually exclusive combinations are spelled out in the Dialog Developer's Guide And Reference manual

Default Attribute Characters

- If you omit the)ATTR section, ISPF provides three default attribute characters implicitly:

Character	Attributes
%	TEXT, HIGH INTENSITY
+	TEXT, LOW INTENSITY
_	INPUT, HIGH INTENSITY

- You can override what characters should play these roles by specifying the DEFAULT parameter in the)ATTR header:

```
)ATTR   DEFAULT($@#)
```

  - The three characters in the parentheses must be adjacent, no intervening spaces or commas

- You may override default characters <u>and</u> specify additional attribute characters:

```
)ATTR      DEFAULT(?$#)
     %        TYPE(INPUT)          COLOR(GREEN)
     @        TYPE(TEXT)           COLOR(PINK)
     /        TYPE(OUTPUT)         COLOR(BLUE)
```

Standard Panel Format

❒ **The first three lines of most panels have special reserved functions, leaving the rest of the screen available for specific application use**

LINE 1	Title of panel	Short Message
LINE 2	Command / option line	
LINE 3	Long Message	

- ♦ **Short Message area is used to pass information about a request (status, error, etc.)**

- ♦ **Long Message area is normally used for headers, data, etc.**

  - ✗ But if an error message appears in the Short Message area, issuing a HELP command or pressing the HELP PF key will cause further explanation to display in the Long Message area; a Long Message may cover multiple lines, being up to 512 characters long

❒ **Note that as of ISPF Version 4, the standard style has been supplemented with the CUA (Common User Access) style, which we discuss briefly later in the course**

Panel Definition - The BODY Section

- ❒ **The header for the BODY section may contain some keyword parameters:**

)BODY [CMD(*field-name*)] [SMSG(*field-name*)] [ASIS]

[LMSG(*field-name*)] [WIDTH(*nn*)] [EXPAND(*xy*)]

- ❒ **Note that the)BODY statement cannot extend past one line**

CMD / SMSG / LMSG Parameters:

- ♦ **Normally, the first INPUT field on a panel is treated as the command field, the place where the user may enter a command**

- ♦ **However, you may designate any INPUT field on the panel as the command field; do this by specifying the name of a variable that follows an INPUT attribute character in the BODY in the "CMD(*field-name*)" parameter on the header**

- ♦ **Similarly, the short message field is normally the last 26 characters of line 1, but you can change that by specifying "SMSG(*field-name*)"; the specified field must be TYPE(OUTPUT)**

- ♦ **Also, the long message field is, by default, the third line of a panel, but you can change it by coding "LMSG(*field-name*)" in the header; this field must also be TYPE(OUTPUT)**

The BODY Section, continued

ASIS:

- ♦ **Means: do not let the user override the placement of the command and long message lines using PDF option 0**

WIDTH:

- ♦ **Specify the character width (from 80 to 160) of the panel (it must not be greater than the screen width of the terminal being used)**
- ♦ **If the width is greater than 80, this parameter must be specified**

❒ **If you specify CMD as “CMD()”, no command field will be used**

- ♦ **For a table display panel you must have a command field**

❒ **You may code the DEFAULT parameter (to change the default attribute characters) on the)BODY header if you do not do so on the)ATTR header**

The BODY Section, 3

EXPAND:

- ♦ **You can request ISPF to repeat a character as many times as it will fit on a line**

- ♦ **Do this by identifing the starting and ending delimiter characters in the EXPAND parameter of the BODY header:**

```
)BODY        EXPAND(//)
```

- ♦ **Note the starting and ending delimiter characters can be the same**

- ♦ **Now, whenever a single character is bounded by the two characters, ISPF repeats the bounded character to fill out the line:**

```
+--/-/-- &TITLE --/-/--
%COMMAND===>_ZCMD / /    +SCROLL%===>_SCRL+
+
%CUSTOMER NAME: _CUSTNAME           / /                  +
```

❒ **See the Dialog Developer's Guide and Reference manual for pointers and concerns with this facility**

BODY Section Contents

❒ **After the ")BODY" header line, code lines that describe the screen image, using the attribute characters you have previously defined**

Remember

- ♦ **Only "TEXT" TYPE fields can use ampersands (&) to request symbolic substitution**
- ♦ **"INPUT" and "OUTPUT" TYPE fields simply have variable names immediately following their respective attribute characters**

Also

- ♦ **A dialog variable may only appear once in a panel body in a non-TEXT field (a field with an attribute of INPUT or OUTPUT)**
  - ✗ A dialog variable may appear any number of times in a panel body in a field with the TEXT attribute

The END Section

❒ **A panel definition must always terminate with an)END statement**

❒ **Any lines following this statement are ignored**

❒ **Blank lines and comments may appear anywhere in the following sections**

)ATTR
)INIT
)REINIT
)PROC

❒ **Comments are of the form: /* comments */**

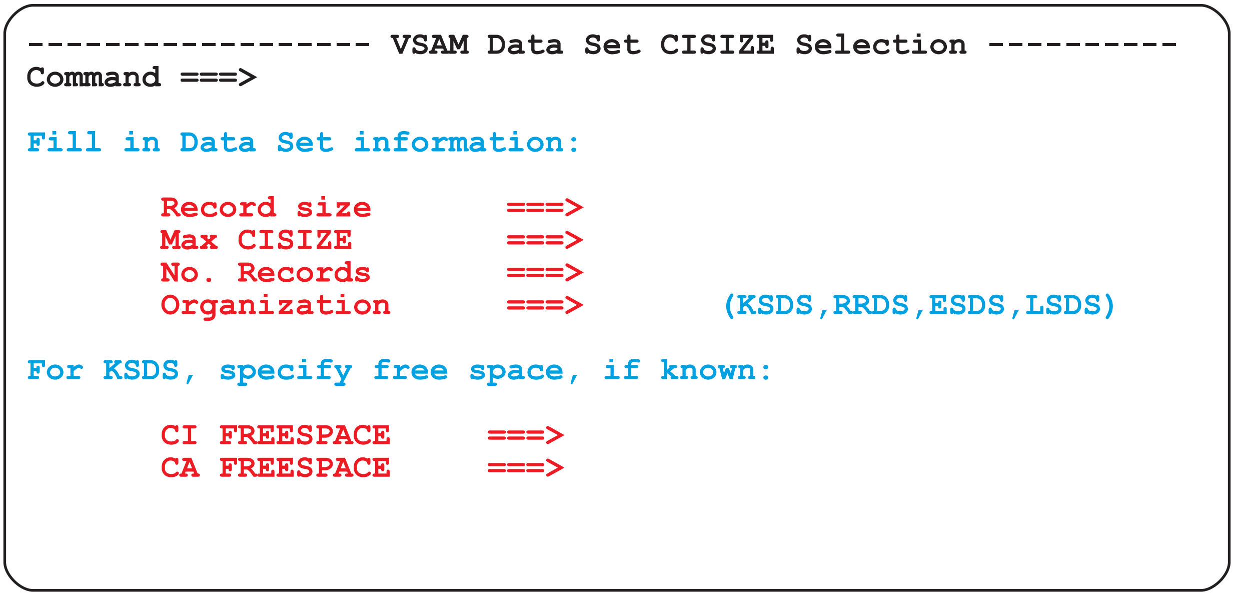
- ♦ **A comment must be the last (rightmost) element on a line**

❒ **Note that since any lines placed after the)END statement are ignored, this is a good place for comments documenting a panel definition**

Another Sample Panel Definition

```
)ATTR
  $ COLOR(RED) TYPE(TEXT)
  * COLOR(BLUE) TYPE(INPUT)
)BODY   EXPAND(//)
%--/-/ VSAM Data Set CISIZE Selection /-/--
%Command ===>_ZCMD
%
+Fill in Data Set information:
+
$       Record size      %===>*recsz+
$       Max CISIZE       %===>*maxci +
$       No. Records      %===>*norecs  +
$       Organization     %===>*org +  (KSDS,RRDS,ESDS,LSDS)
+
+For KSDS, specify free space, if known:
+
$       CI FREESPACE    %===>*cifspc+
$       CA FREESPACE    %===>*cafspc+
)END
/*   panel to extract parameters for calculations     */
/*   Copyright (C) 2000 by Steven H. Comstock         */
```

- **Resulting display:**



```
------------------ VSAM Data Set CISIZE Selection ----------
Command ===>

Fill in Data Set information:

       Record size       ===>
       Max CISIZE        ===>
       No. Records       ===>
       Organization      ===>       (KSDS,RRDS,ESDS,LSDS)

For KSDS, specify free space, if known:

       CI FREESPACE     ===>
       CA FREESPACE     ===>
```

Panel Layout Concerns - Line 1

❒ Line 1 should contain a title and any critical information

- ♦ For example, consider ISPF edit, which identifies the dataset [and member] being edited

❒ If possible, let the right hand 26 characters of line 1 be used for a Short Message area

Panel Layout Concerns - Command Line

❒ **Although the user can change the location of the command line (and, sometimes, the Long Message line), you should plan for the command line to be on line 2**

- ♦ **If the command line is near the bottom, on entering split screen mode, the user may not be able to get to the command line**
- ♦ **The command field may not be longer than 255 characters (other fields may be longer than 255 characters)**

❒ **The command field may have any name, but “ZCMD” is a system variable provided for this purpose and seems to be as good a choice as any; implications:**

- ♦ **If you use “ZCMD”, default initial cursor placement is in the first INPUT field following the command field, if the command field is blank (and in the command field if it is non-blank)**
- ♦ **If you use some other name for the command field, the default initial cursor placement is in the command field**
- ♦ **In any case, the value of certain dialog variables and other influences may be at work here, so trial and error is sometimes called for**

Panel Layout Concerns - Line 3

❐ **Generally leave line 3 (the default Long Message line) blank**

❐ **Never let the Long Message area be on the same line as an INPUT field**

Panel Layout Concerns - General

❒ **Keep panel “look and feel” simple and consistent**

- ♦ **If too many fields, break panel into two successive panels**

- ♦ **Group related fields under a common heading**

- ♦ **Align fields vertically on a panel, whenever possible**

- ♦ **Establish visual “signals” to clue the user in, even subconsciously, to the kinds of fields on the panel**

    - ✗ For example, using arrows (===>) to indicate input fields and colons (:) to indicate protected data

- ♦ **Consider keeping in the same style as ISPF/PDF**

    - ✗ In any case, at least be consistent across an application

- ♦ **Use color and highlighting sparingly**

    - ✗ Be consistent: use the same colors / highlighting for similar types of information

More Panel Layout Concerns

❒ Realize that if the user is requesting “PFSHOW”, two to four lines at the bottom will be taken up with the PFKEY display

❒ Split Screen Impacts

- ♦ Do not use the last line on a panel (may not see in split screen mode)
- ♦ Put most important fields near the top (may need to re-split the screen to see, otherwise)

Display Services

❐ **To understand the basic handling of a DISPLAY request, remember that a panel definition may contain none, some, or all of the following sections:**

)INIT	**- to perform pre-display processing**
)REINIT	**- to perform pre-redisplay processing**
)PROC	**- to perform post-display processing**

❐ **Panel processing includes**

- ♦ **Examining variables, changing variable values, and validating values the user may have entered**
- ♦ **If data is not valid, setting the .MSG panel control variable to indicate what error message to issue**
  - ✗ If .MSG is not blank, the panel will be redisplayed without returning to your dialog function

Display Services, continued

❒ The general processing of a DISPLAY is as follows:

- The named panel is found in the panel library (ISPPLIB)
- The)INIT section, if any, is executed
- ISPF fills variable values into the panel body
- The panel body is displayed
- The user enters data or commands and presses <ENTER> or a PF key
- ISPF copies values from panel into dialog variables
- Control passes to the)PROC section, if any
- If the user entered the CANCEL, END, EXIT, or RETURN commands, return to the function issuing the display
- If the .MSG control variable is blank, return to the function issuing the display
- The)REINIT section, if any, is executed

Request Panel Display

❒ **To display a panel, simply issue the ISPEXEC request DISPLAY:**

```
ISPEXEC     DISPLAY     [PANEL(panel-name)]
```

❒ **If the PANEL parameter is omitted, the)REINIT section, if any, is processed and the current panel is redisplayed**

❒ **After the DISPLAY, the return code is one of these values:**

0	**- Normal completion**
4	**- Unknown command entered**
8	**- User entered END or RETURN command**
12	**- Panel could not be found**
16	**- Error in storing or processing variables**
20	**- Severe error**

Two other optional parameters on the DISPLAY request

♦ **CURSOR(***field_name***) - where cursor should appear on display**

♦ **CSRPOS(***position***) - position where cursor should appear within the field where it appears**

Computer Exercise: Defining Panels

REXX VERSION

1. Code two panel definitions, called SAMPLX2 and SAMPLX3, as described on the following page. SAMPLX2 is to display some REXX variables using different attributes, as well as to provide a field for user data entry. SAMPLX3 will display the user-supplied data as well as some status information.

2. After you have coded the panel definitions, code an EXEC that performs as follows:

- Put TIME() into UTIME1
- Put DATE() into UDATE
- Put USERID() into USER
- PARSE SOURCE into SRCE
  (Parse source SRCE)
- PARSE VERSION into VER
  (Parse version ver)
- Put LINESIZE() into UWTERM
- Issue a LIBDEF request to put your panel library in the list of panel libraries
- Put TIME() into UTIME2
- Request a DISPLAY of SAMPLX2
- Put RC into USERCC
- Put TIME() into UTIME3
- Put USERIN into USERIN2
  (Parse upper var USERIN USERIN2)
- Request a DISPLAY of SAMPLX3

3. Run this dialog from option 6 (COMMAND) of ISPF.

4. [optional] issue the ISPLIBD command and examine the output.

Computer Exercise: Defining Panels, p. 2

REXX VERSION

Description of panel SAMPLX2

This panel should display various REXX variables using different attribute characters, as follows:

Variable	Attributes
SRCE	Output, low intensity, red
UWTERM	Output, high intensity, red
USER	Output, low intensity, green
UDATE	Output, high intensity, green

These fields should be labelled by text fields (white).

Also display a text field (blue) containing the text:

TIME AT ENTRY TO EXEC: &UTIME1, TIME NOW: &UTIME2
REXX VERSION AND RELEASE: &VER

Ask the user to enter a string of text, including upper and lower case characters.

Save this string in the field USERIN. Remember to define an attribute character with TYPE(INPUT) and CAPS(OFF) for defining the input variable for this string.

Description of panel SAMPLX3

This panel should display the following variables:

USERCC
UTIME1
UTIME2
UTIME3
USERIN as is
USERIN with Caps on (use variable USERIN2)

NOTES

1. Include the panel name on each panel.
2. Experiment with different EXPAND characters.
3. Experiment with whatever HILITE and COLOR options your terminal supports.

Computer Exercise: Defining Panels, p. 3

CLIST VERSION

1. Code two panel definitions, called SAMPL02 and SAMPL03, as described on the following page. SAMPL02 is to display some TSO CLIST variables using different attributes, as well as to provide a field for user data entry. SAMPL03 will display the user-supplied data as well as some status information.

2. After you have coded the panel definitions, code a CLIST that performs as follows:

    - Establish CONTROL ASIS
    - SET &SYSTIME into &UTIME1
    - SET &SYSTSOE into &VER
    - SET &SYSLTERM into &ULTERM
    - SET &SYSWTERM into &UWTERM
    - Issue a LIBDEF request to put your panel library in the list of panel libraries
    - SET &SYSTIME into &UTIME2
    - Request a DISPLAY of SAMPL02
    - SET &LASTCC into &USERCC
    - SET &SYSTIME into &UTIME3
    - SET &USERIN into &USERIN2
    - Request a DISPLAY of SAMPL03

3. Run this dialog from option 6 (TSO) of ISPF.

4. [optional] issue the ISPLIBD command and examine the output

.

CLIST VERSION

Description of panel SAMPL02

This panel should display various TSO CLIST variables using different attribute characters, as follows:

Variable	Attributes
&ULTERM	Output, low intensity, red
&UWTERM	Output, high intensity, red
&SYSUID	Output, low intensity, green
&SYSPROC	Output, high intensity, green

These fields should be labelled by text fields (white).

Also display a text field (blue) containing the text:

TIME AT ENTRY TO CLIST: &UTIME1, TIME NOW: &UTIME2
TSO VERSION AND RELEASE: &VER

Prompt the user to enter a string of text, including upper and lower case characters. Save this string in the field &USERIN. Remember to define an attribute character with TYPE(INPUT) and CAPS(OFF) for defining the input variable for this string.

Description of panel SAMPL03

This panel should display the following variables:

&USERCC
&UTIME1
&UTIME2
&UTIME3
&USERIN as is
&USERIN with Caps on (use variable &USERIN2).

NOTES

1. Include the panel name on each panel.
2. Experiment with different EXPAND characters.
3. Experiment with whatever HILITE and COLOR options, your terminal supports.

This page intentionally left almost blank.

Index

Special characters

This page intentionally left almost blank.

CLASS CRITIQUE SHEET

Course Title : Developing Dialog Manager Applications in z/OS
Date : ____
Location : ____
Instructor : ____

1. Do you feel the course met its objectives? If not, what shortcomings could be corrected?

2. Did you find the content of the course to be of value in your work? Briefly say why or why not.

3. What topics would you have liked to see added or expanded on?

4. What topics would you have liked to see shortened or deleted?

5. What comments do you have regarding the course exercises?

6. Do you feel you had the right background for this course? If not, in what way(s) did you feel unprepared?

7. Other comments, suggestions, bouquets, brickbats...

Name ____________________________
(Optional)

Company __________________________

Use this sheet for additional comments, as necessary.