



This presentation looks at a number of the more advanced features of ISPF panels, including:

•Action Bar and Pull-Down Choices

- Point and Shoot fields
- Dynamic areas
- Scrollable areas
- •Scrollable fields
- Advanced panel logic

Panel REXX

This session also provides a brief overview of the benefits of using Dialog Tag Language (DTL) to develop ISPF panels.

**NOTE**: This session assumes you have a basic knowledge of ISPF panel processing.

Action	Bars and Pull-Down C	hoices	
Action Bar	File Languages Colors Help		
Pull-Down Choices	Language: 15 1. Automatic 2. Default All Settings 3. Save and Exit 4. Cancel Language: 15 1. Automatic 2. Assembler 3. BookMaster 4. C 5. COBOL 6. IDL 7. ISPF DTL 8. ISPF Panel 9. ISPF Skeleton 10. JCL 11. Pascal 12. PL/I 13. REXX 14. SuperC	Color Settings More: Coloring: 1 1. Do not color progra 2. Color program 3. Both IF and D0 logi 4. D0 logic only 5. IF logic only Enter "/" to select option Parentheses matching / Highlight FIND strings / Highlight cursor phrase Note: Information from this panel i saved in the edit profile.	+ m C

Action Bars and Pull-Downs Choices support Common User Access (CUA) guidelines. The CUA guidelines define a user interface in terms of common elements, such as the way information appears on a screen, and interaction techniques, such as the way users respond to what appears on a screen. The <u>Action Bar</u> is the area at the top of an application panel that contains <u>action bar choices</u> for the panel.

<u>Pull-Down Choices</u> represents a group of related choices that appear in the pull-down associated with the action bar choice.

When the user selects an action bar choice, the associated <u>pull-down</u> appears directly below the action bar choice.

<u>Pull-Downs</u> contain choices that, when selected by the user, perform actions that apply to the contents of the panel.

In the example shown, the File action bar choice has been selected.



A panel defines action bars and pulldown choices using the **)ABC**, **) ABCINIT**, and **)ABCPROC** panel sections.

**)ABC DESC(...)** - This section is specified after the )ATTR section. It defines the text of the action bar choice, the location of the mnemonic character (the underscored bit), the action names and responses. This is done with keywords and statements:

•DESC(...) - text of action bar choice. This must match the text in the ) BODY section.

•PDC DESC(...) - each pull-down choice is identified by the PDC statement.
•ACTION RUN(...) - defines command to be run when this choice is selected. Must be 2-8 characters in length.

**)ABCINIT** - Allows setup before the pull-down display. This allows alteration of the text of the action. For example you may want to verify that some conditions are met and make the text of an action state that the choice is unavailable. Within the )ABCINIT section, you must assign a value to .ZVARS of a unique name, even if you do not use the name for anything else. This is to allow ISPF to create a field on the action bar which will contain the name of the action.

**)ABCPROC** - Allows setup after action selection. This section is not required and is only needed in special cases (e.g. verification of the pull-down entry field).



This example of an **)ABC** section has 4 actions, defined by the **PDC** (pull down choice) statement. Each **PDC** has an **ACTION** statement which tells ISPF what action to take when the pull-down choice is selected.

When a pull-down choice is selected, ISPF takes the value of the **ACTION RUN(...)** statement (the part in the RUN() section) and logically places it on the command line and simulates an Enter key. This allows the processing to be handled by the command table, program logic, or panel logic.

You can use the **ISRROUTE** command to invoke the **ISPF SELECT** service (ISRROUTE is a command defined in the ISPF command table ISPCMDS).

```
e.g. ACTION RUN(ISRROUTE) PARM('SELECT PGM(USERLIST) NEWAPPL(USR)')
```

You can use the **PARM** keyword to pass parameters.

For example, you can specify **ACTION RUN(CANCEL)** and have the ISPF command table handle the CANCEL command. Or you could say **ACTION RUN(XYZ)** and have your panel logic and/or program handle the XYZ command (assuming it is not in the active command table).

**NOTE**: The **)ABCINIT** section must contain a **.ZVARS** control variable assignment statement to associate a variable name with the pull-down entry field.

Interactive Program Development Facility (ISP	F)	IBM
Action Bars and Pull-Down Ch	noices	
ATTR section - define attribute types from the section bar */ \$ TYPE(AB) /* action bar */ @ TYPE(ABSL) /* action bar sep.	or AB, ABSL	
<pre>"Define )ABC sections with DESC() key )ABC desc(Menu) PDC DESC('Save') ACTION RUN(SAVI &lt; additional logic here &gt; )AECINIT .zvars = 'MENUX' )ABC desc(Help) &lt; additional logic here &gt;</pre>	word E)	
6 ISPF Panels - Advanced   Session 11708	August 2012	© 2012 IBM Corporation

To create an action bar on a panel, you need to:

1. Add at least the TYPE(AB) to the )ATTR section.

2. Add )ABC and )ABCINIT sections to describe the action bar choices.

Remember to:

•set the DESC value in )ABC to the corresponding text in the )BODY section
•assign .ZVARS in the )ABCINIT section

## Notes:

•The "additional logic here" shown in the example refers to other pull-down choices, not processing logic.

•In the example the **TYPE(ABSL)** attribute character is used to define the action bar separator line.



3. Add an action bar to the panel using an **AB** attribute character (\$) for each pull down. After the attribute, place the name of the pull-down from the description in the )ABC section.

## Notes:

•there must be a blank between the AB attribute and the text.

•the '+' attribute character represents normal text.

Program or panel logic, or entries in a command table are required to process the action bar selections.

Inte	eractive Program Development Facility (ISPF)		IBA
Actic	on Bars and Pull-Down Cho	ices - Example	
	<pre>. Info(2) / Faint Line // * TTPE(NT) // Faint Lext */ + TTPE(NT) //* Normal text */ _ TTPE(NEP) /* Normal entry field */ \$ TTPE(AB) /* Action bar separator line */ ABC desc(Menu) PC DESC('Save') ACTION RUN(SAVE) PC DESC('Cancel') ACTION RUN(SAVE) PC DESC('Cancel') ACTION RUN(CANCEL) )ABCINIT .zvars = 'MENUX' )ABC DESC(Halp) PC DESC('Extended Help') ACTION RUN(HEP)</pre>		
	)ABCINIT .ZUARS = HELPX .RESP = ENTER /* Don't even show choice */ /* This is an example and */ /* is NOT CUA */ )BODY WINDOW(48,6) +\$ Menu \$ Help + Panel Title	*	
	%===>_ZCMD +This is normal text. Right?_ANS+ (Yes, No) )END	*	
8	ISPF Panels - Advanced   Session 11708	August 2012	© 2012 IBM Corporation

This is a working example of a panel which will appear in a window (if an ADDPOP is issued). It has 2 pull-downs, **Menu** and **Help**. It shows the use of some CUA attributes (e.g. TYPE(NEF)), overriding the defaults (&, + , and \_) and it shows an example of adding processing to the **)ABCINIT** section for the **Help** pulldown.

If the **Help** pull-down is selected, ISPF will automatically simulate an extra enter key because of the **.RESP=ENTER** in the )ABCINIT section. Thus the pull-down will never be shown and since the default is for ISPF to select the first pull-down choice when just the enter key is pressed and the pull-down is selected for the 1st time, ISPF will run the HELP command and will never actually show the pull-down.

This is a method you can use to have action bar choices automatically take some action.

٨ ــــــ	
ACTI	on Bars and Pull-Down Choices - Example
	The panel looks like this when displayed:
	Monu Help
	menu netp
	Panel Title
	This is normal text Right? (Yes. No)
	F1=HELP F2=SPLIT F3=END
	When 'Menu' is selected a nulldown is shown:
	Menu Help
	1. Save anel Title
	2. End
	3. Cancel xt Right? (Yes, No)

Shown here is the format of the panel when initially displayed and when the **Menu** choice from the action bar is selected.



Point-and-Shoot fields are areas of a panel that cause an action to take place when the cursor is placed on the area and the **ENTER** key is pressed. To create a Point-and-Shoot field you need to:

1) In the )ATTR section, define the attribute byte used for point-and-shoot input/output fields and text

- (a) PAS(ON|OFF) defines the attribute byte for input and output point-andshoot field. It is not a CUA type, therefore, it can have the color, intensity, and highlight changed.
- (b) TYPE(PS) defines the attribute byte for point-and-shoot text. It is a CUA type, so it does not allow color, intensity, and highlight changes.

2) For each panel field defined with a point-and-shoot attribute, there must be a corresponding **FIELD** entry in the **)PNTS** section. If there is no corresponding entry in the )PNTS section no action is taken when the field is selected. The )PNTS section is specified after the )BODY and )AREA sections. The name specified in the FIELD statement is *field\_name* for input/output fields, and *ZPSxxyyy* for text fields. When a point-and-shoot field is selected, ISPF sets *variable* to *value*. The dialog should take an appropriate action based on the value found in variable.



This is a working example of point-and-shoot fields on a selection panel displayed in a popup panel. The menu options "Settings", "View" and "Edit" have the PS attribute. When the cursor is placed on the text with a PS attribute and Enter is pressed, the corresponding value specified in the ) PNTS section is assigned to the ZCMD variable. Logic in the )PROC section of the panel then causes the appropriate action to be taken.

The variable EXITPB is assigned to an output point-and-shoot field by using the \$ attribute character which has PAS(ON) specified. When the cursor is placed on this field the value EXIT is assigned to the variable ZCMD.



ISPF allows a section of an ISPF panel to be defined as a scrollable area. A scrollable area can be thought of as a window (or "hole") in a panel through which a larger section of data may be viewed. ISPF handles any requests by the user to scroll the data.

ISPF supports multiple scrollable area on one ISPF panel, in which case scrolling is cursor sensitive.



Defining a scrollable area requires 3 actions:

- 1. Define the attribute character for the scrollable area
- 2. Define the "hole" in the )BODY section using the attribute character to delimit the area
- 3. Define the format of the area to be displayed in the hole This is the ) **AREA** section
- The name specified after the first attribute byte of the scrollable area (in the )BODY section) is the **NAME** of the area and corresponds with the name specified on the )AREA section.

ISPF references scrollable area from left to right, top to bottom.



The attribute byte for a scrollable area is defined with **AREA(SCRL)**, and optionally **EXTEND(ON|OFF)**. This character is used to define the borders of the scrollable area in the )BODY section.

EXTEND(ON) specifies that the depth of the AREA automatically increases to the physical screen size. There can only be ONE extendable area on a panel and the value for the EXTEND option cannot be a dialog variable.

Interactive Program Development Facility	(ISPF)	IBM
Defining a Scrollable Area		
Demining a Scrullable Area		
<ul> <li>Define the "hole" in the )BODY susing the scrollable area attribut</li> <li>Bound the area with the attr</li> <li>Specify the area name imm</li> <li>The top line of the scrollable indicator that ISPF automat</li> <li>Minimum width is 20 character</li> </ul>	ection where the scrollable area appea e character ibute character ediately after the first attribute characte e area is reserved for the More: -+ cally displays ters, minimum depth is 2 lines	ars er
) BODY		
% Panel Til	e	
+===>_ZCMD	+	
+Static Data +		
\$SCRL1 \$ \$ \$	\$ \$ +Static \$ +Data \$	
15 ISPF Panels - Advanced   Session 11708	August 2012 ©	2012 IBM Corporation

The area that we want to define as a scrollable area is **bound** with the attribute character (**\$** was defined as the attribute character for the scrollable area on the previous page).

The **name** of the scrollable area (in this example it is **SCRL1**), is specified immediately after the first attribute character for the scrollable area.

## Notes:

•The first line of the scrollable area is reserved for the indicator: MORE: -+

•The minimum size of a scrollable area is 20 characters by 2 lines.

Interactive Program Development Facility (ISPF) Defining a Scrollable Area		IBM
Define the area to be displayed in the "hole Aputhing that can be put in a RODY s	" in the )AREA section	
<ul> <li>Arlything that can be put in a JBODT's section except:         <ul> <li>Action bar lines</li> <li>Graphics area</li> <li>MODEL section</li> <li>Command line</li> <li>Alternate message locations</li> <li>Another scrollable area</li> <li>Dynamic area using EXTEND(ON) or SCROLL(ON)</li> </ul> </li> <li>The DEPTH(d) parameter may be used to of lines in the scrollable area when EXTEN</li> </ul>	)AREA SCRL1 +Here you can have + +field prompts, etc. + +Enter A:_VARA + + B:_VARB + + The Sum: &SUMAB + Difference:!DIFAB + Product:!PRODAB specify the minimum number D(ON) is specified.	
16 ISPF Panels - Advanced   Session 11708	August 2012 © 2012 IB	M Corporation

The )AREA section defines the format of the data displayed in the associated scrollable area.



ISPF allows a section of an ISPF panel to be defined as a dynamic area. Dynamic areas can contain:

•Simple Text

•Input fields, and

•Output fields.

The Calendar on the ISPF primary menu is and example of a dynamic area. Dynamic area could be used for a "better" table display, as is the case with the ISPF data set list (option 3.4).

It is the application program that controls EVERYTHING within the Dynamic area, including:

•Text and where it is located

•Input fields and where they are located (and how long)

•Text attributes (including color and highlighting)



A Dynamic Area is an area set aside in the )BODY section of a panel and is controlled by an ISPF variable (called a dynamic area variable). The value of the dynamic area variable defines the data displayed in the dynamic area.

It is the responsibility of the application program displaying the panel with the dynamic area to:

•Set up the dynamic area variable

•Display the panel

•Evaluate changes to the data within the dynamic area and then update any ISPF variables

•Handle any requests to scroll the dynamic area (UP, DOWN, TOP, BOTTOM)



A dynamic area can be thought of as a long string of characters that wraps to create the multiple lines within the area. The string of characters contains the attribute characters, text, and input/output fields. Attributes are defined in the )ATTR section using TYPE(DATAOUT) and TYPE(DATAIN) options.



From the example above, the % attribute character identifies in the dynamic area an output/text area that will be yellow and high intensity. The \_ (underscore) attribute character is used for an input field that is pink. An attribute character is also required to define the dynamic area within the )BODY section. The examples above use the @ attribute character to define where in the )BODY section the dynamic area is located. Like scrollable areas, dynamic areas can be defined with either EXTEND(ON) or EXTEND(OFF).



Defining a shadow variable for a Dynamic Area allows the dynamic area to be colored on a character by character basis (eg: ISPF Edit highlighting). A shadow variable is also required to generate graphic escape sequences in dynamic area for generating boxes and diagrams.

The shadow variable is another dialog variable that maps the dynamic area byte for byte. The TYPE(CHAR) attribute in the )ATTR section is used to define attributes used in the shadow variable.

Interactive Program Development Facility (	SPF)
Shadow Variable - Example	
TYPE(CHAR) will not conflict with characters in the dynamic variable *	DEFAULT(%+_) EA(DYNAMIC) SCROLL(OFF) PE(CHAR) COLOR(YELLOW) HILITE(USCORE) Panel Title and ===>_ZCMD + e static text NVAR_SHADVAR @
Shadow variable maps dynamic area variable byte for byte Command ===> Some static Your Underl	NVAR = 'Your Underline' ADVAR= 'Y Y ' el Title - text ine
22 ISPF Panels - Advanced   Session 11708	August 2012 © 2012 IBM Corporation

Here is a simple example showing how a dynamic area can be defined.

The @ attribute byte is used to identify the location of the dynamic area. It is non-extendable and non-scrollable.

The **Y** attribute character defines the character that will be used in the shadow variable value to display data in the dynamic area as yellow and underlined.

The dynamic area variable name is **DYNVAR**. The shadow variable name is **SHADVAR**.

The )INIT section of the panel contains logic to set values for the variables DYNVAR and SHADVAR. Note that the position of the Y characters in the value for SHADVAR cause the uppercase Y and U in the dynamic area to be displayed as yellow and underlined.



Scrollable Fields is a feature of ISPF added with z/OS 1.5. It was also made available to z/OS 1.2 and later via APAR OW57368 (PTF UA02839). A scrollable field allows the display of dialog variables that are larger than the field on an ISPF panel. In fact the maximum data length supported for a scrollable field is 32K.

A new **)FIELD** section is used to define a field that is scrollable. The **LEFT** and **RIGHT** commands are used to scroll the data displayed in a scrollable field. The new **ZEXPAND** command displays the data from a scrollable field in a popup window.



The Syntax of the **)FIELD** section is shown here. Options are provided to allow the association of other variables with the scrollable field for: •field length

- •LEFT/RIGHT Scroll indicators (defaults: +)
- •Separator lines (default: <->)
- •Left and Right displayed column values
- •Scale lines ----- 1 ----- 2 ----- 3 . . .
- •Scroll control ON or OFF

Int	eractive Program Develo	pment Facil	ity (ISPF)		
Scro	llable Fields -	Evamo	ام		
0010		∟латтр			
	)ATTR   TYPE(OUTPUT) CA TYPE(INPUT) CA	PS(OFF) JU	ST(ASIS ) ST(ASIS ) FO	RMAT(MIX)	
	)BODY				
	% Left/Right	Scroll Par	nel		
	%OPTION ===>_ZCMD				
	+ Field	Value			
	+				
	+ Value	:_LR1	+		
	+ Scroll Indicator	: lrlin			
	+ Left & Right	: lril	lrir		
	+ Left/Right cols	:_Lr1lf	_lr1ri		
	+ Length	:_lr1ln			
	+ Scale	: lr1sc	+		
	+ Separator	: lrisp	+		
	)FIELD				
	Field(LR1)				
	<pre>Ind(lrlin,'&lt;&gt;')</pre>				
	Lind(lril,'<') Ri	nd(lrir,'>	')		
	Lcol(1r11f) Rcol(	lrlri)			
	Len(Irlin)				
	Scale(Irisc)				
	sind(irisp)				
	) END				
		44700		August 2012	© 2042 IDM Comparation

Shown here is the source of a panel demonstrating the use of scrollable fields. The panel has a single **scrollable field** for variable **LR1**. The panel also displays fields for the following variables associated with current state of the scrollable field. These associations are established through the ) FIELD section definition.

•LR1IN - scroll indicator with value '<>'

•LRIL & LRIR - Left and Right scroll indicators with values '<' and '>'

•LR1LF & LR1RI – Left and Right column values

•LR1LN - Field data length

•LR1SC – Scale line

•LR1SP - separator line



Shown here are examples of the panel display with some suitable data. The variable LR1 is 36 characters longs initialized with the 26 letters of the alphabet followed by 0-9. Initially the display shows the first 12 characters (abcdefghijkl) and only the RIGHT scroll indicators (as we are at the maximum LEFT position). If we position the cursor over the value and scroll RIGHT, the display is updated to show the **next 12** characters, both the LEFT and RIGHT scroll indicators are shown and the separator line indicates we can scroll both directions, and the left/right column values are updated. Also the scale line is updated to reflect the current position.



Here are some 'other' panel logic statements, not covered during the Dialog Developers Boot Camp.

•PANEXIT - used to define a panel exit - we will look at this shortly

•REFRESH - refreshes 1 or more panel fields before a redisplay of a panel

•TOG	- toggle between 2 values if the data in FLD1 is
•VGET/VPUT	- Copy a dialog variable from/to the shared or
application profile pool	

There are other statements, not shown here, and not often used •VEDIT

•EXIT

•GOTO

Interactive Program Development Facility (ISF	PF)	IBM
Panel Logic - Functions		
• TRANS() -Translates	5 N NO * *)	
• TRUNC() – Truncates a variable at a g	given position or character	
<pre>&amp;X = TRUNC(&amp;VAR1,3)    &amp;¥ = TRUNC(&amp;VAR2,'.')    • PFK()   </pre>	apport information	
<pre>&amp;X = PFK(HELP) &amp;Y = PFK(3) /* normally results in</pre>	X = F1 and $Y = END */$	
<ul> <li>LVLINE()         <ul> <li>Provides the line number of or dynamic area</li> </ul> </li> </ul>	of the last visible line within	a graphic
&LVLNE = LVLINE(DYNAREA	)	
28 ISPF Panels - Advanced   Session 11708	August 2012	© 2012 IBM Corporation

There are many built-in functions available for use in panel procedures. Some of the main ones are:

•TRANS - to translate values from 1 value to another. Most often used in menu selection panels (as we saw earlier). The '\*,\*' as the last pair is the anything-else condition. In the example, if &VAR1 is neither 'Y' or 'N' the value of VAR1 is put into VAL.

•TRUNC - truncates a variable. Truncation can either occur at a specified position, or at a particular character. The remaining data after the truncation point is stored in the .TRAIL control variable.

•PFK - provides function key information

•LVLINE - provides the line number of the last visible line within a graphic or dynamic area. This is useful to determine how many lines to scroll (up or down). (And it takes into account the presence or not of the function key display)



Two new functions were added in z/OS 1.5 (ISPF 5.5):

•LENGTH - can occur on the right side of an assignment statement to evaluate the length of a dialog variable. The variable length returned will be the <u>maximum</u> of the actual length of the variable if it exists and the length specified in the )FIELD section.

•**UPPER** - can occur on the right side of an assignment statement and will return the uppercase value of a variable.

Support added to z/OS 1.2 and above via APAR OW57368 (PTF UA02839).



With z/OS 1.6 support was added to allow REXX to be invoked from ISPF panel procedures. This extended the processing capability in a panel procedure by making available the many features and functions of REXX. Tasks previously difficult or impossible to do from a panel procedure are now simple, such as reading data from a data set or performing mathematical calculations.

While it was possible to use a panel exit to invoke REXX from a panel procedure, using panel REXX is much simpler. The \*REXX statement is used to invoke panel REXX and identify the ISPF variables to be processed by the REXX. The REXX can be coded directly into the panel procedure after the \*REXX statement or the name of a member containing the REXX can be specified with the \*REXX statement.



Some restrictions apply to the processing that can be done in panel REXX:

•ISPF services cannot be called

•Compiled REXX requires calls to module ISPPRXVP to make ISPF dialog variables available to REXX and to update the variable back in the ISPF variable pool.

•Coding REXX in the panel procedure could result in error message ISPP321 if the combined length of panel statements exceeds 32K.

•The REXX can only process the ISPF dialog variables identified on the \*REXX statement.

•The \*ENDREXX statement is required to terminate REXX coded directly into a panel procedure.

•If EXIT rather than RETURN is used for interpreted REXX, the call to ISPPRXVP generated by ISPF to update the dialog variables will not be executed. Therefore updates to these variables from within the REXX will not be 'seen' by ISPF.



•Dialog variables passed to panel REXX must be in character format and must remain in character format.

•On releases prior to z/OS 1.9, panel REXX cannot change the length of the value of a dialog variable. This generally required the value of the dialog variable to be padded to the required length before being passed to panel REXX. After changes provided with z/OS 1.9, this is no longer necessary.



Dialog variable ZRXRC is provided by ISPF for panel REXX to set a return code. Dialog variable ZRXMSG is provided by ISPF for panel REXX to set the ID for a message to be displayed when the panel REXX sets a return code of 8.

**Note**: It is not necessary to specify ZRXRC or ZRXMSG on the \*REXX statement.

Inte	ractive Program Development Facility (ISPF)	IBA
Invol	king Panel REXX	
	Invoked using *REXX statement	
	<pre>*REXX[([*,]value,value,[,(member)])]</pre>	
	<ul> <li>*         <ul> <li>Specifies that all the dialog variables defined in the panel )E be passed to the REXX code for processing</li> </ul> </li> <li>value         <ul> <li>Specifies the names of dialog variables passed to the REX. processing</li> </ul> </li> <li>member         <ul> <li>Specifies the name of a member in the standard search set REXX programs             <ul> <li>Member can contain interpreted REXX or compiled REXX</li> <li>Compiled REXX can be either the output generated by the</li> </ul> </li> </ul></li></ul>	30DY section are to X code for quences used to load REXX compiler when
	using the CEXEC option or a load module generated when li generated by the REXX compiler when using the OBJECT o	ink-editing the output option
34	ISPF Panels - Advanced   Session 11708 August 2012	© 2012 IBM Corporation

The \*REXX statement is used to invoke panel REXX. It identifies the names of the dialog variables to be passed to the panel REXX for processing. A list of variable names can be specified or if all dialog variables referenced in the )BODY section may be processed by the panel REXX then just an \* (asterisk) can be specified.

The REXX can be coded in the panel procedure immediately after the \*REXX statement. Alternatively if there is a large amount of REXX code or the code is common amongst a number of panels the name of a member containing the REXX code can be specified on the \*REXX statement.

Inte	ractive Program Development Facility (	ISPF)	IBM
Pane	l REXX - Example		
	)ATTR % TYPE (PT) + TYPE (NT) # TYPE (FP) _ TYPE (NEF) ! TYPE (NEF) CAPS(ON) @ AREA(DYNAMIC) EXTEND(ON)		
	)BODY % #Command ===>_MYVAR + #Data set name!DSN +	- Panel REXX	+
Pad DYNAREA	+@DYNAREA )INIT &DSN = '' &PADDYNA = ' &PADDYNA = '&PADDYNA &PADDYNA = '&PADDYNA		@
to required length	&PADDYNA = '&PADDYNA &PADDYNA = '&PADDYNA &PADDYNA = '&PADDYNA &PADDYNA = '&PADDYNA &PADDYNA = '&PADDYNA &DYNAREA = &PADDYNA )REINIT PEPDFEG(*)		
35	ISPF Panels - Advanced   Session 11708	August 2012	© 2012 IBM Corporation

This and the following page show an example of a panel using panel REXX.

The )BODY section contains some static text and 3 variables (MYVAR, DSN, DYNAREA), 1 of which (DYNAREA) is a dynamic area.

The )INIT section initializes the DSN and the DYNAREA variables to blanks, and pads the DYNAREA variable to the length required by the REXX code.

**Note**: The padding of variable DYNAREA is not required on z/OS 1.9 or later.

Inter	active Program Development Facility (ISPF)		IBM
Define DSNRC as 2 characters - long	)PROC &DYINAREA = &PADDYNA IF (absn ¬= ' ') VER(&DSN DSNAME) IF (.MSG = '0) #REEX(*,DSNRC)	/* REX ADDRE: "DISP] panrc Do wh. "DI Rod exit	X */ SS 'ISPEXEC' LAY PANEL(PANREXX)" = rc lie rc < 8 SPLAY"
Invoke REXX and expose all JBODY variables	DSNRC = LISTDSI(DSN) If DSNRC <= 4 Then Do DYNAREA = , 'Data set name: 'Volume serial ID: Unit or which rolume modes.	<pre>'Left(sysdsname,44)    , 'Left(sysvolume,44)    , 'Left(sysvolume,44)    ,</pre>	Rexx Driver
and DSNRC	<pre>'Data set organisation: 'Data set organisation: 'Record format: 'Logical record length: 'Data size: End Return end Return</pre>	<pre>'Left(systercy,44)    , 'Left(systercy,44)    , 'Left(systerch,44)    , 'Left(systercl,44)    , 'Left(systercl,44)    ,</pre>	
	IF (&DSNRC > 4) .MSG = ISRD028	Alternatively, you can set variable ZRXMSG in the th the .MSG control variable	the system ne REXX code to set
36	ISPF Panels - Advanced   Session 11708	August 2012	© 2012 IBM Corporation

In the )PROC section the VER statement is used to validate the user entered a correctly formed data set name.

The dialog variable DSNRC is set to a value of "00" to cause the variable to be 2 characters in length.

The Panel REXX is invoked with the \*REXX statement, exposing all variables, including the DSN variable, defined in the )BODY section and the DSNRC variable which was implicitly defined in the )PROC section.

The REXX code calls the TSO function LISTDSI to determine if the data set exists, and to get the attributes of the data set returned in the REXX variables SYSDSNAME, SYSVOLUME, and so on.

The REXX code formats the information returned by LISTDSI into the DYNAREA variable and uses the REXX RETURN statement to return to ISPF.

The REXX code is terminated by the \*ENDREXX statement.

If the REXX code returns a return code > 4, the .MSG control variable is set to ISRD028 to cause that message to be displayed.

Interactive Program Development Facility (ISPF)	IBM
Panel REXX – Example	
Command ===> Data set name	
F1=HELP F2=SPLIT F3=END F4=RETURN F5=RFI F7=UP F8=DOWN F9=SWAP F10=LEFT F11=RIG Command ===>	xx
Data set name <u>CLIST</u>	
F1=HELP F2=SPLIT F3=END F4=1 F7=UP F8=DOWN F9=SWAP F10=1	RETURN F5=RFI LEFT F11=RIG
37 ISPF Panels - Advanced   Session 11708 August 2012	© 2012 IBM Corporation

This and the following page show an example of invoking our panel. When the panel is displayed the user enters the name of the data set that will have it's attributes displayed.

Interactive Program Development	Facility (ISPF)		IB
Panel REXX – Exampl	٩		
Command ===>	Panel REXX		
Data set name <u>CLIST</u>			
Data set name: Volume serial ID: Unit on which volume resides: Data set organisation: Record format: Logical record length: Block size:	STROED.CLIST D\$US18 3390 PO FB 80 27920		
F1=HELP F2=SPLIT F3=	END F4=RETURN	F5=RFIND F	6=RCHANGE
F1=HELP F2=SPLIT F3= F7=IIP F8=DOWN F9= ISPE Panels - Advanced L Session 11	END F4=RETURN SWAP F10=LEFT 708 A	F5=RFIND F <u>F11=RIGHT F1</u> August 2012	6=RCHANGE 2=RETRIEVE © 2012 IBM Corporation

Shown here is the display of the data set attributes formatted into the dynamic area by the panel REXX code.

**Note**: ISPF ships a sample of panel REXX code in the source module ISRVCHIL in the ISPF REXX exec library (ISP.SISPEXEC). This panel REXX code is invoked in the )INIT section of the panel ISRVCALP, which is shipped in the ISPF panel library. This panel REXX code is used to enable color highlighting of the entries in the trace data set generated by the ISPVCALL utility. ISPVCALL is used by the ISPF product support team to assist in debugging customer reported problems.



DTL is the acronym for "Dialog Tag Language". It is markup language based on ISO SGML.

ISO - International Standards Organization

SGML - Standard Generalized Markup Language

Other examples of SGML include Bookmanager, Bookmaster, DCF, HTML, ....

In ISPF you can use DTL to define:

- •Panels
- •Messages
- •Command Tables
- •Keylists

ISPF ships the conversion utility ISPDTLC to convert the DTL into standard ISPF panels, messages, command and keylist tables.



The main reference material to help you develop ISPF panels is the **ISPF Dialog Developer's Guide and Reference** 

•Chapter 6 is the main reference material, while

- •Chapter 4 provides Common User Access (CUA) Guidelines, and
- •Chapter 5 provides many guidelines and examples of ISPF panels
- •Chapter 7 provides additional information for Help and Tutorial panels

For those wishing to venture into DTL, the Dialog Tag Language Guide will help you.