SHARE San Jose 2017

Customizing the ISPF HILITE Command

Pedro Vera

pvera@rocketsoftware.com

#SHAREsjc

SHARE is an independent volunteer-run information technology association that provides education, professional networking and industry influence.

SHARE
EDUCATE • NETWORK • INFLUENCE

About Pedro Vera

35 Years at IBM + 4 months at Rocket Software

1. Frequent contributor to social media
2. Ibmmainframes.com - (not part of IBM)
3. ibmmainframeforum.com - (not part of IBM)
4. Linkedin group moderator: Programming in Rexx
5. TSO-REXX list
6. ISPF-L list

About Pedro Vera

I convinced my IBM colleagues to provide additional product support:

1. System REXX,
2. SDSF REXX API,
3. REXX interface for RACF,
4. Inline REXX for ISPF panels,
5. Inline REXX for ISPF skeletons,
6. REXX Trace highlighting
7. XML parser
8. Lotus Notes URLs
9. Sametime name retention
10. TCP/IP put/get to JES2
11. ‘Bare Metal’ C

1. Option 6 - history of commands
2. Editor COMPARE command
3. Multiple levels of command tables
4. SRCHFOR from DSLIST & member list
5. Data set name retrieval
6. VIEW mode
7. DTL
8. Work Station Agent
9. Dynamic Areas

Agenda

1. **Simple panels**
2. **Panels with dynamic areas**
3. Shadow variables
4. How the Editor uses Dynamic Areas
5. More dynamic area topics

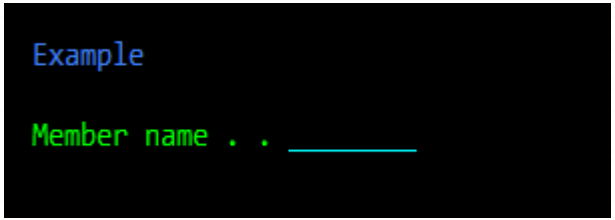
ISPF panels

Interactive System Productivity Facility (ISPF) is the dialog manager for z/OS.

Panels are used to communicate with the user.

Simple Panel

- Panels are a type of program. They are members of a PDS allocated to ISPPLIB DD name.
- The panels will only work when you are in ISPF.
- The format of a panel is very specific. It has tags, which start in the first column.



Simple Panel

Specify the panel content in the)BODY section.

```
1  )ATTR
2   _ type(input) color(turq) hilite(uscore)
3   % type(TEXT)  color(Blue)
4   + type(TEXT)  color(green)
5  )Body
6  %Example
7
8  +Member name . ._mymem   +
9  )End
```

Dynamic Areas

The format of the area is specified by a variable with both attribute characters and data characters.

Simple Panel

```
1  )Body
2  %Example
3
4  +Member name . ._mymem
5  +
6  )End
```

Dynamic Area Panel

```
1  )Body
2  %Example
3
4  |MEMLINE                                    |
5  )INIT
6  &memline = '~Member name . .$          ~'
7  )End
```

Dynamic Areas - Example

The variable is set before the panel is displayed.
Example:

```
)ATTR
 % type(TEXT)     color(Blue)
 | area(dynamic)
 ~ type(dataout) color(green)
 $ type(datain)  color(turq) hilite(uscore)
)Body
%Example

|memline                                |
)INIT
&memline = '~Member name . .$         ~'
)End
```

Attribute Characters

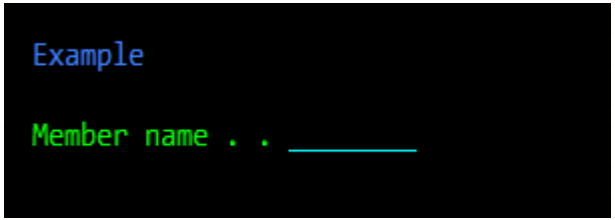
1. One attribute to define boundaries (AREA tag)
2. Other attributes with DATAOUT or DATAIN

```
1  )ATTR
2   % type(TEXT)     color(Blue)
3   | area(dynamic)
4   ~ type(dataout) color(green)
5   $ type(datain)   color(turq) hilite(uscore)
6  )Body
7  ...
```

Dynamic Areas -)BODY

1. Area attribute defines width of the area.
2. Variable contains the area content.

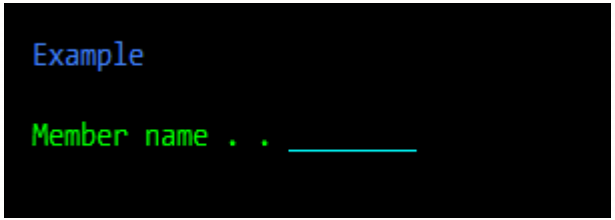
```
1   )Body
2   %Example
3
4   |memline                                  |
5   )INIT
6   &memline = '~Member name . .$          ~'
7   )End
```



Dynamic Areas -)INIT

1. Variable contains the text shown in the area
2. Consists of attribute characters and text.

```
1   )Body
2   %Example
3
4   |memline                                  |
5   )INIT
6   &memline = '~Member name . .$          ~'
7   )End
```



Example of Dynamic Area

The example shows)INIT section, but more likely is done from a calling program.

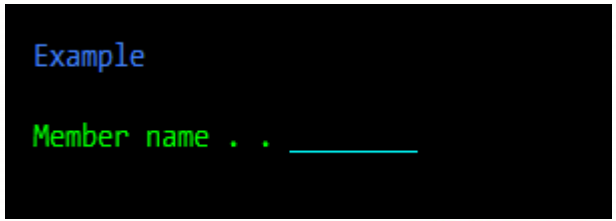
```
)ATTR
 % type(TEXT)     color(Blue)
 | area(dynamic)
 ~ type(dataout) color(green)
 $ type(datain)  color(turq) hilite(uscore)
)Body
%Example

|memline                                |
)INIT
&memline = '~Member name . .$         ~'
)End
```

Dynamic Areas - Rexx

Set area content before the displaying the panel.

```
1  /* rexx */
2  Address ISPEXEC
3  memline = '~Member name . .$          ~'
4  "Display panel(dynam14)"
5
6
```




```
1  )ATTR
2   % type(TEXT)     color(Blue)
3   | area(dynamic)
4   ~ type(dataout) color(green)
5   $ type(datain)  color(turq) hilite(uscore)
6  )Body
7  %Example
8
9  |memline                                  |
10 )End
11
```

Dynamic Areas - Parse

After the user presses Enter, examine the results

```
1  /* rexx */
2  Address ISPEXEC
3  memline = '~Member name . .$          ~'
4  "Display panel(dynam15)"
5  Parse var memline . "$" mymem "~"
6  Say "mymem("mymem")"
7
```


```
1  )ATTR
2   % type(TEXT)     color(Blue)
3   | area(dynamic)
4   ~ type(dataout) color(green)
5   $ type(datain)  color(turq) hilite(uscore)
6  )Body
7  %Example
8
9  |memline                                  |
10 )End
11
```

Agenda

1. Simple panels
2. Panels with dynamic areas
3. **Shadow variables**
4. How the Editor uses Dynamic Areas
5. More dynamic area topics

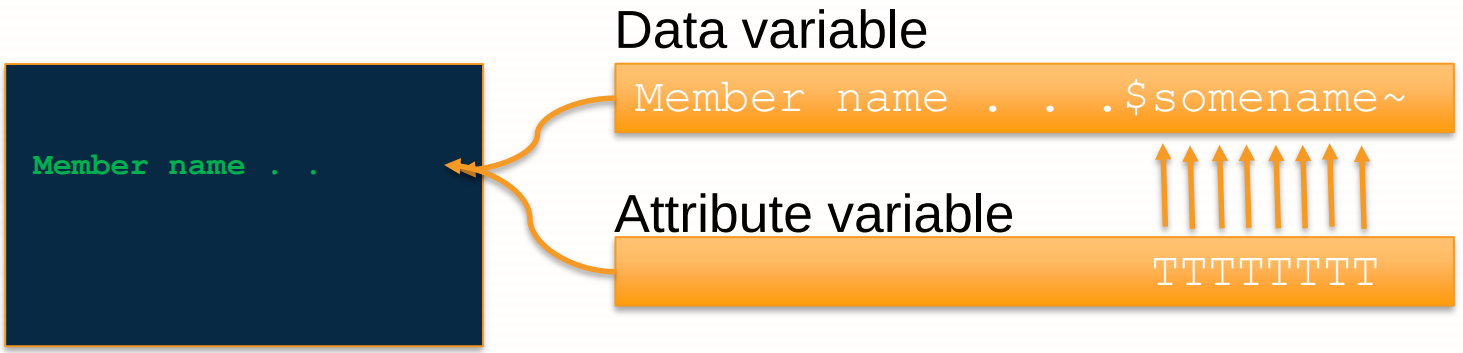
Shadow Variables

Use a shadow variable if a dynamic area will use character-level attributes.

Place TYPE(CHAR) attributes in the shadow variable such that they **map to the text** in the dynamic area affected by the attribute.

Shadow Variables

- The shadow variable contains character-level attribute characters.
- There is a one-to-one correspondence between the attribute and the character it applies to.



Shadow Variables

- The dynamic area definition includes a second variable name!

```
1  /* rexx */
2  Address ISPEXEC
3  memline = '~Member name . .$PSV0023 ~'
4  memshad = '                  TTTTTTTT '
5  "Display panel(dynam33)"
6
```

```
1  )ATTR
2   % type(TEXT)     color(Blue)
3   | area(dynamic)
4   ~ type(dataout) color(green)
5   $ type(datain)  color(yellow) hilite(uscore)
6   T type(CHAR)     color(turq)
7  )Body
8  %Example
9
10 |memline,memshad                          |
11 )End
```

Order of Processing

When you edit a panel:

-)BODY section
-)INIT
-)PROC

When the panel is displayed:

- Executes)INIT section
- Displays)BODY
- Executes)PROC

With REXX as Driving Program

When it executes:

- Rexx program
  - Executes)INIT section
  - Displays)BODY
  - Executes)PROC
- Returns to Rexx program

Agenda

1. Simple panels
2. Panels with dynamic areas
3. Shadow variables
4. **How the Editor uses Dynamic Areas**
5. More dynamic area topics

Editor Supports the Hilite Command

Issue HI ON in primary command line.

```
EDIT       TS5781.CLIST.CLIST(RTC09) - 01.00
Command ===> hi on
000011 cnt = 0
000012 Do z = 1 to rtc01.0
000013   Select
000014     When Pos('DNM1=',        rtc01.z) > 0 Then
000015       Call save_memname
000016     When Pos('XINCLUDE',     rtc01.z) > 0 Then
000017       Call save_include
000018     Otherwise
000019       Nop
000020   End
```

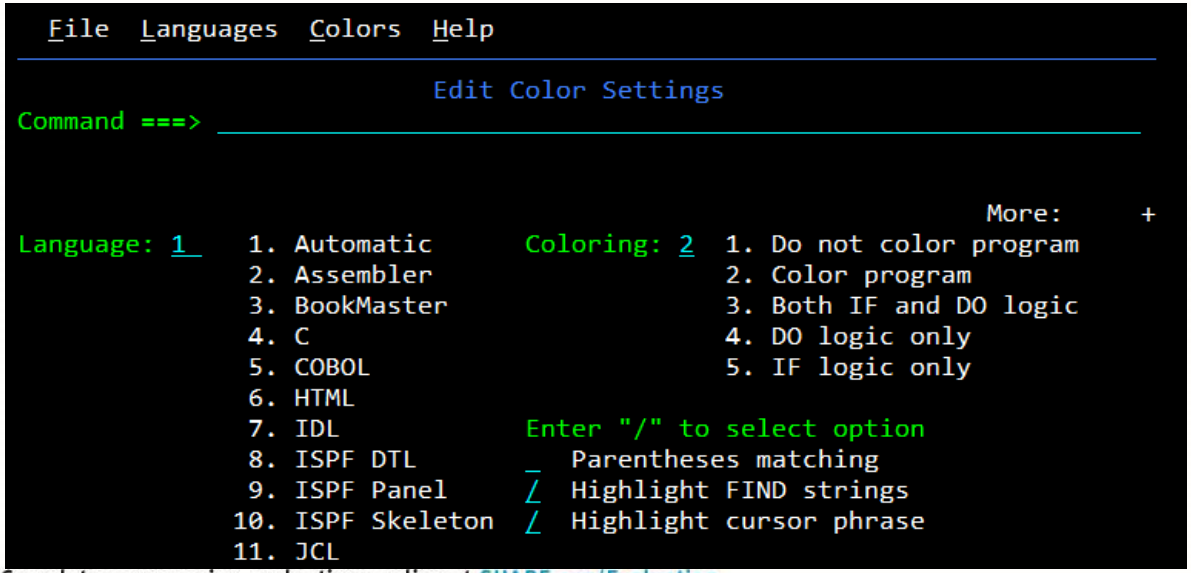
Context Sensitive Highlighting

Language keywords are in a different color.

```
EDIT       TS5781.CLIST.CLIST(RTC09) - 01.00
Command ===> _____________________________________________________________
000011 cnt = 0
000012 Do z = 1 to rtc01.0
000013   Select
000014     When Pos('DNM1=',         rtc01.z) > 0 Then
000015       Call save_memname
000016     When Pos('XINCLUDE',      rtc01.z) > 0 Then
000017       Call save_include
000018     Otherwise
000019       Nop
000020   End
000021 End
000022 cnt=cnt+1;
```

Hilite Control Panel

The user can specify various HILITE settings.



The Editor Uses Dynamic Areas

Even without HILITE, the editor highlights some text, for example:

- When using the FIND command.
- Cursor position

```
EDIT       TS5781.CLIST.CLIST(RTC09) - 01.00
Command ===>
000011 cnt = 0
000012 Do z = 1 to rtc01.0
000013   Select
000014     When Pos('DNM1=',        rtc01.z) > 0 Then
000015       Call save_memname
000016     When Pos('XINCLUDE',     rtc01.z) > 0 Then
```

Arbitrary Highlighting

You can do your own arbitrary highlighting by using customized ISPF parts.

Customize Editor Panel

Customized Edit panels must be patterned after one of these panels:

- ISREFR01 - Edit without action bars or extended highlighting
- ISREFR02 - Edit with action bars and extended highlighting
- ISREFR03 - Edit with action bars and no extended highlighting
- ISREFR04 - Edit with extended highlighting but no action bars

Copy from ISP.SISPENU

Editor as a Driving Program

When it executes:

- Editor
  - Executes)INIT section
  - Displays)BODY
  - Executes)PROC
- Returns to Editor

Editor Panel with Dynamic Area

- ZDATA has text
- ZSHADOW has attributes

```
1   )BODY  EXPAND(//) WIDTH(&ZWIDTH)  CMD(ZCMD)
2   *Z          *Z/ /                                    !Columns*Z     *Z     @
3   !Command ===>#Z/ /                                     !Scroll ===>#Z    @
4   {ZDATA,ZSHADOW/ /                                                        {
5   {/ /                                                                     {
6   )INIT
7   .
8   .
9   )PROC
10  .
11  .
12
13
```

More about ZDATA

- ZDATA has text
  - In 24 x 80 screen, it is a 1920 byte string
- ZSHADOW has text
  - In 24 x 80 screen, it is a 1920 byte string

Arbitrary Hilite Implementation

1. Rexx invokes VIEW service
2. VIEW service uses modified editor panel
3. Modified panel invokes rexx panel exit during)INIT
4. Panel exit examines data & modifies the shadow variable.

Editor as a Driving Program

When it executes:

- Editor
  - Executes)INIT section
    - **Invokes external rexx program**
  - Displays)BODY
  - Executes)PROC
- Returns to Editor

Invoking View

Specify panel name in VIEW service call.

```
/* rexx */
Address ISPEXEC
"VIEW DATASET(clist.clist(mytest))",
     "PANEL(COLORS1)",      /* highlighter */
     "CONFIRM(NO) CHGWARN(NO)"
```

Modified Editor Panel

In the)INIT section, invoke the panel exit

```
1   )BODY  EXPAND(//) WIDTH(&ZWIDTH)  CMD(ZCMD)
2   *Z          *Z/ /                                  !Columns*Z     *Z     @
3   !Command ===>#Z/ /                                    !Scroll ===>#Z    @
4   {ZDATA,ZSHADOW/ /                                                       {
5   {/ /                                                                    {
6   )INIT
7   .
8   .
9   PANEXIT((ZDATA,ZSHADOW),REXX,COLORS1)
10  )PROC
11  .
12  .
13
```

Panel Exit can Update Shadow

A panel exit in the)INIT section gets control:

1. After normal editor highlighting is set
2. Before the panel is displayed.

The exit can update the shadow attributes.

Existing Editor Attributes

(The first letter of the color.)

```
)ATTR
  R  TYPE(CHAR) COLOR(RED)
  G  TYPE(CHAR) COLOR(GREEN)
  B  TYPE(CHAR) COLOR(BLUE)
  W  TYPE(CHAR) COLOR(WHITE)
  P  TYPE(CHAR) COLOR(PINK)
  Y  TYPE(CHAR) COLOR(YELLOW)
  T  TYPE(CHAR) COLOR(TURQ)
.
.
.
```

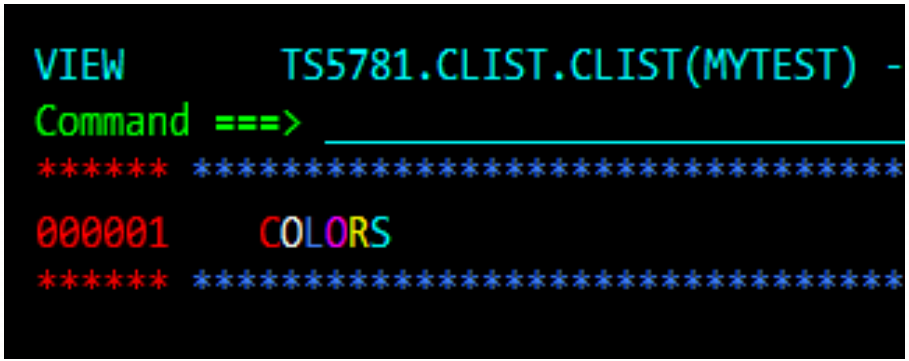
Example of Shadow Update

Find position of target in ZDATA and replace the attribute in ZSHADOW.

```
1  /*rexx */
2  Call ISPREXPX 'i'
3  offset  = Pos('COLORS', zdata)
4  If offset  > 0 Then
5    Do
6      /* Change highlight       */
7      zshadow = Overlay('RWBPYG', zshadow, offset, Length('COLORS'))
8    End
9  Call ISPREXPX 't'
```

Sample Result

The target text is highlighted based on the panel exit.




```
Overlay('RWBPYT',…
```

Example of SQL Hilite, part 1

Use a generalized routine to process an array of keywords

```
1   /*                                  Current                    New      */
2   /*     keyword                     Color                      Color    */
3   /*     --------                    --------                   ---------*/
4    ky.1 =' AND '          ; attr.1 =' DDD'          ; colr.1 =' RRR'
5    ky.2 =' AS '           ; attr.2 =' DD '          ; colr.2 =' RR '
6    ky.3 =' BY '           ; attr.3 =' DD '          ; colr.3 =' RR '
7    ky.4 =' CREATE '       ; attr.4 =' DDDDDD'       ; colr.4 =' PPPPPP'
8    ky.5 =' DATABASE '     ; attr.5 =' DDDDDDDD'     ; colr.5 =' RRRRRRRR'
9    ky.6 =' FETCH '        ; attr.6 =' DDDDD'        ; colr.6 =' RRRRR'
10   ky.7 =' FOR '          ; attr.7 =' DDD'          ; colr.7 =' RRR'
11   ky.8 =' GROUP '        ; attr.8 =' DDDDD'        ; colr.8 =' RRRRR'
```

Example of SQL Hilite, part 2

Process a stem of SQL keywords.

```
    /* Process each keyword                                          */
1   Do ix  = 1 To ky.0
2     strt = 2
3     offset  = 1
4     /* Do until no more instances of this keyword are found       */
5     Do While (offset > 0)
6       offset  = Pos(ky.ix, zdata    ,strt)
7       If offset  > 0 Then
8         Do
9           /* Change highlight if not already highlighted          */
10          If Substr(zshadow,offset,Length(attr.ix)) = attr.ix Then
11            zshadow  = Overlay(colr.ix, zshadow, offset, Length(colr.ix))
12          strt = offset + 1
13        End
14    End
15  End
```

SQL Result

The SQL keywords are highlighted based on the panel exit logic.

```
000001  SELECT STRIP(CHAR(VALUE(CNT, 0))) AS ADBLOBC,
000002  S.*,
000003  VALUE(HASLOBS,'NO') AS HASLOBS,
000004  '           ' AS PARTNTBL,
000005  STRIP(CHAR(VALUE(XMLCNT, 0))) AS ADBXMLC,
000006  VALUE(HASXML,'NO') AS HASXML,
- - -  - - - - - - - - - - - - - - - - - - - - - -
000016
000017  LEFT OUTER JOIN SYSIBM.SYSTABLESPACE S
000018    ON   S.DBNAME=M.DBNAME
000019    AND  S.NAME  =M.NAME
```

Pros & Cons

- Very powerful function
- Fairly easy to do!

- You only see a subset of the file.
  - Easy to get out of context.
- Can only do in special purpose editor sessions.

Editor Summary

The editor uses standard dynamic areas.

A panel exit can be used to manipulate the shadow variable.

Agenda

1. Simple panels
2. Panels with dynamic areas
3. Shadow variables
4. How the Editor uses Dynamic Areas
5. **More dynamic area topics**

Extending the Dynamic Area

Use the EXTEND(ON) attribute to use all lines at the remainder of the panel.

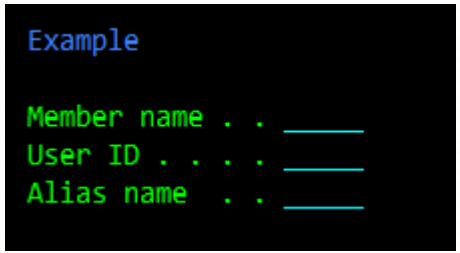
```
1  /* rexx */
2  Address ISPEXEC
3  line1= '~Member name . .$      ~'
4  line2= '~User ID . . . .$      ~'
5  line3= '~Alias name  . .$      ~'
6
7  Memline = left(line1,80) ||,
8            left(line2,80) || line3
9  "Display panel(dynam19)"
10
```

```
1   )ATTR
2    % type(TEXT)     color(Blue)
3    | area(dynamic) EXTEND(ON)
4    ~ type(dataout) color(green)
5    $ type(datain)  color(turq) hilite(uscore)
6   )Body   expand(//)
7   %Example
8
9   |memline          / /              |
10  )End
```

Extending the Dynamic Area

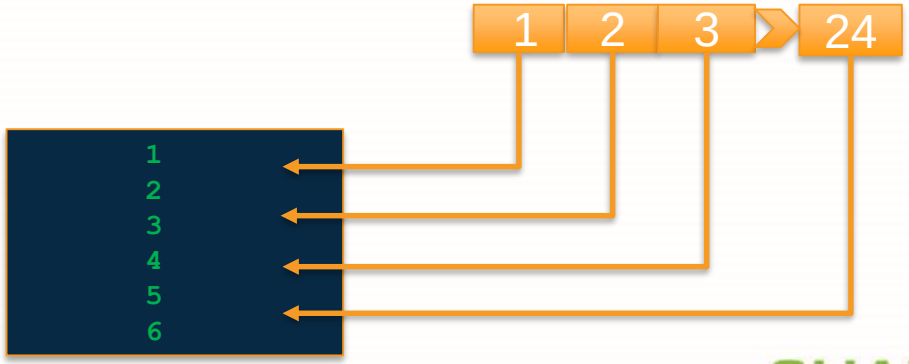
Concatenate three lines in dynamic area.

```
1  /* rexx */
2  Address ISPEXEC
3  line1= '~Member name . .$      ~'
4  line2= '~User ID . . . .$      ~'
5  line3= '~Alias name  . .$      ~'
6
7  Memline = left(line1,80) ||,
8            left(line2,80) || line3
9  "Display panel(dynam19)"
10
```



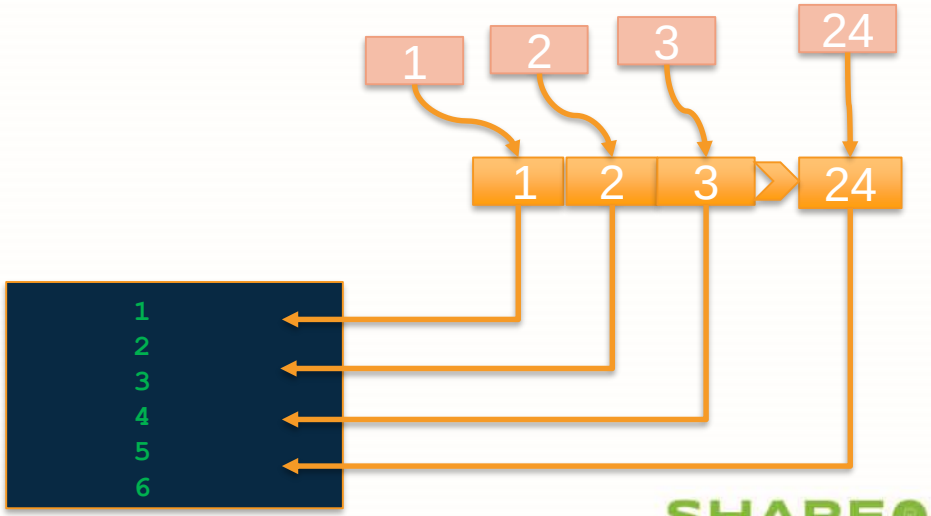
Working with the Data Buffer

- Use one data buffer that maps to the screen area.
- Fill in text based on area dimensions
- ISPF presents it as different lines



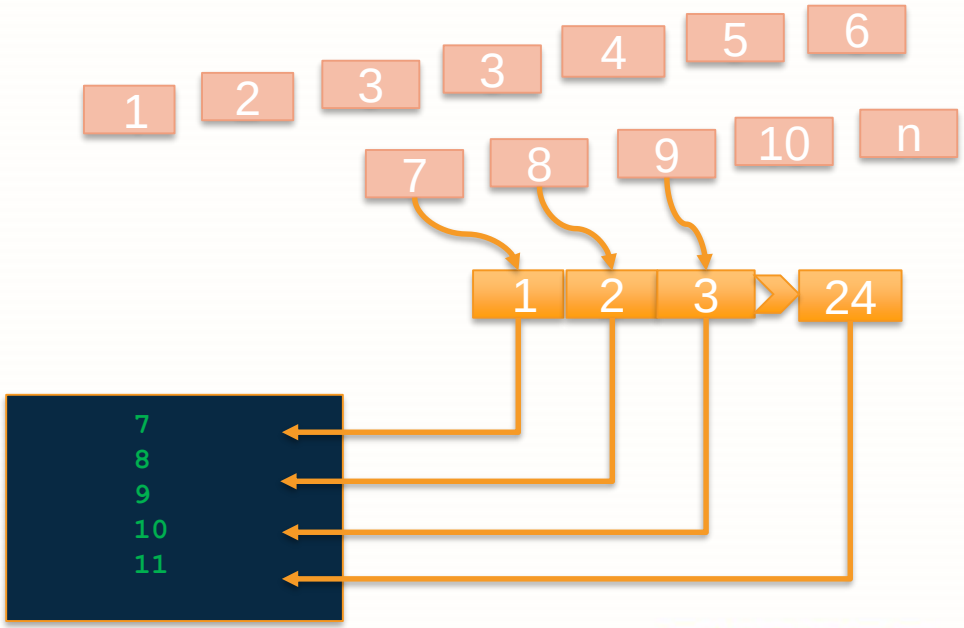
Working with the Data Buffer

Fill in screen data buffer with individual data records



Scrolling

- Determine offset from last display
- Fill in data buffer from that point



Scrolling the Dynamic Area

Use the SCROLL(ON) keyword to allow scrolling

```
1   )ATTR
2    % type(TEXT)     color(Blue)
3    | area(dynamic) EXTEND(ON) SCROLL(ON)
4    ~ type(dataout) color(green)
5    $ type(datain)  color(turq) hilite(uscore)
6   )Body   expand(//)
7   %Example
8
9   |memline          / /              |
10  )End
```

Scrolling

- ZSCROLLD variable has the default scroll amount.
  - PAGE is the default.
- ZSCROLLA contains the value of the scroll amount field, such as MAX or CSR.
- ZVERB contains the scroll direction, DOWN or UP (or LEFT or RIGHT)

Scrolling

- ZSCROLLN and ZSCROLNL contain the number of lines or columns to scroll computed from the value in the scroll amount field or entered as a scroll number .
  - ZSCROLLN can be up to '9999'. (my recommendation is not to use)
  - ZSCROLNL can be up to '9999999'.
- If ZSCROLLA is MAX, ignore ZSCROLLN and ZSCROLNL.

Supporting a Wide Screen

- Use ZSCREENW or PQUERY result to compose lines
- Need to use EXPAND() to allow ISPF to display it in ‘wide mode’

```
1  /* rexx */
2  Address ISPEXEC
3  "VGET (Zscreenw)"
4  line1= '~Member name . .$      ~'
5  line2= '~User ID . . . .$      ~'
6  line3= '~Alias name  . .$      ~'
7  Memline = left(line1,zscreenw) ||,
8            left(line2,zscreenw) ||line3
9  "Display panel(dynam29)"
```

```
1  )ATTR
2   % type(TEXT)     color(Blue)
3   | area(dynamic)  EXTEND(ON)
4   ~ type(dataout)  color(green)
5   $ type(datain)   color(turq)
6  hilite(uscore)
7  )Body   expand(//) width(&zscreenw)
8  %Example
9
10 |memline          / /              |
   )End
```

Dynamic Areas - DATAMOD

```
)ATTR
# AREA(DYNAMIC) DATAMOD(!)
```

The character '!‘ replaces the attribute byte for each field in the dynamic area that has been changed by the user.

All other attribute bytes remain as they are.

DATAMOD

When parsing the result, search for the ‘datamod’ special character.

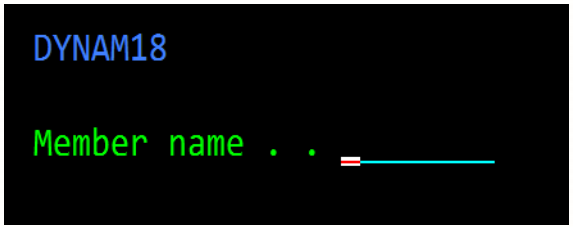
```
1  /* rexx */
2  Address ISPEXEC
3  memline= "~Member name . .$      ~"
4  "Display panel(dynam17)"
5  parse var memline . "!" mymem "~"
6  Say 'mymem('mymem')'
7
```

```
1  )ATTR
2   % type(TEXT)     color(Blue)
3   | area(dynamic) datamod(!)
4   ~ type(dataout) color(green)
5   $ type(datain)  color(turq) hilite(uscore)
6  )Body
7  %Example
8
9  |memline                                  |
10 )End
```

Cursor Position

- Use .CURSOR and .CSRPOS to set location
- Determine location in calling program

```
1  /* rexx */
2  Address ISPEXEC
3  memline = '~Member name . .$         ~'
4  Ksrpos  = pos('$',memline) + 1
5  "Display panel(dynam18)"
6
```




```
1  )ATTR
2   % type(TEXT)     color(Blue)
3   | area(dynamic)
4   ~ type(dataout) color(green)
5   $ type(datain)  color(turq) hilite(uscore)
6  )Body
7  %Example
8
9  |memline                                    |
10 )INIT
11  .CURSOR = MEMLINE
12  .CSRPOS = &Ksrpos
13 )End
```

PQUERY Service

The PQUERY service returns information for a specified area on a specific panel.

- Width
- Depth

```
CALL ISPLINK('PQUERY ', 'DYNAM21 ', 'MEMLINE ','
                '      ', 'WIDTH ', 'DEPTH ');
```

PQUERY Service

Use width and depth to determine size of data areas and for scrolling.

LVLINE Built-in Function

The LVLINE built-in function (used on an assignment statement) provides the line number of the last visible line within a dynamic area of the currently displayed panel.

```
1  )PROC
2  &LVDEPTH   =   LVLINE(MEMLINE)
3
4
```

SHARE
EDUCATE • NETWORK • INFLUENCE

Thank You for Attending!
Please remember to complete your evaluation of this session in the SHARE mobile app.

Session Code: Customizing the ISPF HILITE Command

Complete your session evaluations online at SHARE.org/Evaluation

SHARE San Jose 2017