



## LAF Image Viewer Java Bean - public properties

These properties can be set or read from any Bean Area  
whose Implementation Class property is set to:

`oracle.forms.fd.ImageViewer`



**1.1**

## Content

SET_ORIENTATION.....	3
SET_NUMBER_ICONS.....	3
SET_ICON_DIMENSION.....	3
SET_PANEL_COLOR.....	3
SET_PANEL_BORDER_COLOR.....	4
SET_ICON_SELECTED_COLOR.....	4
SET_IMAGE_DIMENSION.....	4
SET_IMAGE_SHIFT.....	4
SET_FRAME_WIDTH.....	5
SET_FRAME_COLOR.....	5
SET_FRAME_MIRROR.....	5
SET_BEAN_COLOR.....	5
SET_BEAN_GRADIENT_COLORS.....	6
READ_IMAGE_FILE.....	6
READ_IMAGE_BASE.....	7
SHOW_PANEL.....	9
CLEAR_IMAGES.....	9
REFRESH_SCREEN.....	9
SET_TOOLTIP_DISPLAY_TIME.....	9
SET_LOG.....	10
The messages raised by the bean.....	10

## ***SET\_ORIENTATION***

**Indicates the scrolling image panel location in the bean.**

value can be one of the following:

- NORTH (main image drawn under the scrolling panel)
- SOUTH (main image drawn on top of the scrolling panel)
- WEST (main image drawn on the right of the scrolling panel)
- EST (main image drawn on the left of the scrolling panel)

```
-- set the panel position on top --
Set_Custom_Property('BL.BEAN', 1, 'SET_ORIENTATION', 'NORTH');
```

## ***SET\_NUMBER\_ICONS***

**Indicates how many icons the scrolling panel should display.**

```
Set_Custom_Property('BL.BEAN', 1, 'SET_NUMBER_ICONS', '6');
```

## ***SET\_ICON\_DIMENSION***

**Indicates the panel icon's size.**

property value: pairs of WIDTH,HEIGHT pixel values

```
-- set icons dimension to 50x50 pixels --
Set_Custom_Property('BL.BEAN', 1, 'SET_ICON_DIMENSION', '50,50');
```

## ***SET\_PANEL\_COLOR***

**Set the scrolling panel background color.**

```
-- set the panel background color --
Set_Custom_Property('BL.BEAN', 1, 'SET_PANEL_COLOR', 'r255g255b255');
```



## ***SET\_PANEL\_BORDER\_COLOR***

**Set the scrolling panel border color.**

```
-- set the panel border color --
Set_Custom_Property('BL.BEAN', 1, 'SET_PANEL_BORDER_COLOR', 'r78g87b97');
```

If not indicated, the panel has no border.

## ***SET\_ICON\_SELECTED\_COLOR***

**Set the selected icon frame color.**

```
-- set the selected icon frame color --
Set_Custom_Property('BL.BEAN', 1, 'SET_ICON_SELECTED_COLOR', 'r255g10b30');
```

When no given, default color is dark gray - r100g100b100

## ***SET\_IMAGE\_DIMENSION***

**Set the main image dimension.**

property value: pairs of WIDTH,HEIGHT pixel values

```
-- set the main image dimension --
Set_Custom_Property('BL.BEAN', 1, 'SET_IMAGE_DIMENSION', '250,200');
```

## ***SET\_IMAGE\_SHIFT***

**Indicates the shift between the scrolling panel and the main image.**

If not indicated, the default value is 20.

```
-- set the image shift value --
Set_Custom_Property('BL.BEAN', 1, 'SET_IMAGE_SHIFT', '30');
```



## ***SET\_FRAME\_WIDTH***

**Set the main image frame width.**

If not indicated, the default is 0, so no frame is drawn.

```
-- set the main image frame width --
Set_Custom_Property('BL.BEAN', 1, 'SET_FRAME_WIDTH', '2');
```

## ***SET\_FRAME\_COLOR***

**Set the main image frame color.**

```
-- set the main image frame color --
Set_Custom_Property('BL.BEAN', 1, 'SET_FRAME_COLOR', 'r78g87b97');
```

## ***SET\_FRAME\_MIRROR***

**Draw a mirrored image under the main image.**

If using this function, I would recommend to not set the frame width (no frame drawn)

```
-- draw mirrored main image --
Set_Custom_Property('BL.BEAN', 1, 'SET_FRAME_MIRROR', 'true');
```

## ***SET\_BEAN\_COLOR***

**Set the bean area background single color.**

```
-- set the bean background color --
Set_Custom_Property('BL.BEAN', 1, 'SET_BEAN_COLOR', 'r100g100b100');
```



## ***SET\_BEAN\_GRADIENT\_COLORS***

**Set the bean area background with gradient.**

First parameter is a pair of RGB colors.

Second parameter (not mandatory) can be one of the following:

- LeftToRight (default)
- UpToDown
- LeftUpToRightDown
- LeftDownToRightUp

```
-- set the bean area gradient background --
Set_Custom_Property('BL.BEAN', 1, 'SET_BEAN_GRADIENT_COLORS'
                     , 'r106g181b255,r204g230b255,uptodown');
```

## ***READ\_IMAGE\_FILE***

**Load an image from the file system.**

```
Set_Custom_Property('BL.BEAN', 1, 'READ_IMAGE_FILE', 'file_name|command_text[|tooltip_text]');
```

**command\_text** is the string returned back to the Forms when the end-user clicks the main image

**tooltip\_text** is the text that will appear as a tool tip when the end-user move the mouse in the scrolling panel icons.

If you intend to have carriage returns in the tool tip, pass it as a HTML content like in the following example.

```
-- load an image from the disk --
Set_Custom_Property('BL.BEAN', 1, 'READ_IMAGE_FILE'
                     , 'c:/apostrophe.jpg|apostrophe.au|<html>Apostrophe<br>1974</html>');
```



## ***READ\_IMAGE\_BASE***

**Load an image from the database.**

```
Set_Custom_Property('BL.BEAN', 1, 'READ_IMAGE_BASE', 'image_content|command_text[]|  
tooltip_text]');
```

**image\_content** contains as many image chunks as needed to reconstitute the image. These chunks are read from the database BLOB column through the PKG\_LAF package functions.

the end of the image – last chunk – must contains the special value : [END\_IMAGE]

**command\_text** is the string returned back to the Forms when the end-user clicks the main image

**tooltip\_text** is the text that will appear as a tool tip when the end-user move the mouse in the scrolling panel icons.

If you intend to have carriage returns in the tool tip, pass it as a HTML content like in the following example.

```
Replace( LC$Tooltip, CHR(10), '<br>' ) ;
```

It needs the **PKG\_LAF** package compiled in your schema, or, at least, a grant execute on this package to another schema.

The script of the **PKG\_LAF** package is located in the **/script** folder of the zip file.

Here is the code snippet to use to load an image from the database:

```

-- load an image from the database --
PROCEDURE Load_BLOB_Image
(
  PC$WhereClause  IN Varchar2,
  PC$Command      IN Varchar2 Default NULL,
  PC$Tooltip      IN Varchar2 Default NULL
)
IS
  LB$Ok      boolean ;
  LC$Image   Varchar2(32767) ;
  LC$Clause  Varchar2(4000) ;
BEGIN

  --
  -- Read a BLOB content from the database
  --
  -- Select the Blob column --

  If PKG_LAF.Select_Blob(PC$WhereClause) Then
    Loop
      -- Get the image chunks from the database --
      LC$Image := PKG_LAF.Get_B64_Chunk ;
      If LC$Image Is Not Null Then
        -- Send the chunks to the Java Bean --
        Set_Custom_Property( 'BL.BEAN', 1, 'READ_IMAGE_BASE', LC$Image ) ;
      Else
        -- End the sending process --
        LC$Image := '[END_IMAGE]' ;
        If PC$Command is not null then
          LC$Image := LC$Image || '||' || PC$Command ;
        End if ;
        If PC$Tooltip is not null then
          LC$Image := LC$Image || '||' || PC$Tooltip ;
        End if ;
        Set_Custom_Property( 'BL.BEAN', 1, 'READ_IMAGE_BASE', LC$Image ) ;
        Exit ;
      End if ;
    End loop ;
  End if ;
END;

```

The PC\$WhereClause parameter must contains valid SQL SELECT order to identify the single row that contains the BLOB to read.

Here is a call sample:

```
Load_BLOB_Image('SELECT IMAGE FROM CATALOG WHERE NAME
                 ='studiotan.jpg''','studiotan.au', 'tooltip text');
```



## ***SHOW\_PANEL***

**Display the bean area content.**

This method should be the last one executed, after you have loaded the images.

```
-- show the panel --
Set_Custom_Property('BL.BEAN', 1, 'SHOW_PANEL', ''');
```

## ***CLEAR\_IMAGES***

**Remove all images from the scrolling panel.**

You will typically use it in a *When-New-Record-Instance* trigger.

```
-- clear all loaded images --
Set_Custom_Property('BL.BEAN', 1, 'CLEAR_IMAGES', ''');
```

## ***REFRESH\_SCREEN***

**Refresh the screen when you have loaded new images.**

Use this method when you have cleared old image and loaded a new set

```
-- refresh the screen --
Set_Custom_Property('BL.BEAN', 1, 'REFRESH_SCREEN', ''');
```

## ***SET\_TOOLTIP\_DISPLAY\_TIME***

**Set the number of seconds the tool tip remains visible.**

Give a positive value greater than 4, or give the special -1 value to set the infinite time.

```
-- set tool tip visible during 10 seconds --
Set_Custom_Property('BL.BEAN', 1, 'SET_TOOLTIP_DISPLAY_TIME', '10');
```



## ***SET\_LOG***

Enable / disable the logging to the Java console.

```
-- set the loggin ON --
Set_Custom_Property('BL.BEAN', 1, 'SET_LOG', 'true');
```

## ***The messages raised by the bean***

When the main image is clicked, a message is sent back to Forms with the "command text" passed through the *READ\_IMAGE\_XXXX* methods.

The message can be intercepted in a *When-Custom-Item-Event* trigger.

The custom item event name is **VIEWER\_MESSAGE** and the corresponding value is passed via the **VIEWER\_VALUE** parameter:

```
Declare
  BeanValList      ParamList;
  paramType        Number;
  EventName        VarChar2(60);
  value            Varchar2(2000);
Begin
  BeanValList      := get_parameter_list(:system.Custom_Item_Event_Parameters);
  EventName        := :SYSTEM.Custom_Item_Event;

  If (EventName = 'VIEWER_MESSAGE') then
    get_parameter_attr(BeanValList,'VIEWER_VALUE',ParamType, value);
    Message( 'Image value:' || value, no_acknowledge);
  End if;
End;
```



Oracle Forms Look & Feel project

Created and maintained by Francois Degrelle

[Oracle Forms L&F Web site](#)



*Oracle Forms Look & Feel project*