



## LAF\_LOV 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.LAF\_LOV**

Choose an object

Select a row

Object_Name	Object_Type	Status
AGGRLRS CONCAT	TYPE	VALID
AGGRLRS CONCAT3D	TYPE	VALID
AGGRMBR	TYPE	VALID
AGGRUNION	TYPE	VALID
ALERT_TYPE	TYPE	VALID
ALERT_TYPE	SYNONYM	VALID
ALL_ALL_TABLES	VIEW	VALID
ALL_ALL_TABLES	SYNONYM	VALID
ALL_APPLY	VIEW	VALID
ALL_APPLY	SYNONYM	VALID
ALL_APPLY_CONFLICT_COLUMNS	VIEW	VALID
ALL_APPLY_CONFLICT_COLUMNS	SYNONYM	VALID
ALL_APPLY_DML_HANDLERS	VIEW	VALID
ALL_APPLY_DML_HANDLERS	SYNONYM	VALID
ALL_APPLY_ENQUEUE	VIEW	VALID

Buttons:

1.3

## Content

Introduction.....	3
How to define a LAF LOV.....	5
What can the end-user do when the LOV is displayed.....	10
INIT_LOV .....	11
SET_LOV_MAX_COL_WIDTH .....	11
SET_LOV_SEPARATOR .....	11
SET_LOV_SEARCH_LABEL .....	11
SET_LOV_ARRAY_SIZE.....	12
SET_LOV_HEADER.....	12
SET_LOV_COLS_TYPE .....	12
SET_LOV_SCHEME .....	13
SET_LOV_BOUNDS .....	13
SET_LOV_TITLE .....	13
SET_LOV_PROMPT.....	13
SET_LOV_COL_SEARCH.....	14
SET_LOV_BUTTON_LABELS.....	14
SET_LOV_BUTTON_ICONS .....	14
SET_LOV_DATA .....	14
SET_LOV_CELL_PROPERTY.....	15
SHOW_LOV .....	15
SET_LOV_LOG .....	16
The events raised by the Java Bean .....	17

## Introduction

Actually, the developer would never have to set these properties by him(her)self.  
They are encapsulated in the new **PKG\_LAF\_LOV** package stored in the **laf.dll** PL/SQL library.

It contains the following new procedures:

```
-----
-- display the LOV --
-----
PROCEDURE Show_LOV
(
    PC$LOV_Name    IN Varchar2
  ,PC$LOV_Form    IN Varchar2
  ,PC$BeanName   IN Varchar2
  ,PN$LOV_X_Pos  IN Pls_Integer  DEFAULT NULL
  ,PN$LOV_Y_Pos  IN Pls_Integer  DEFAULT NULL
  ,PC$Filter     IN Varchar2      DEFAULT NULL
  ,PB$Sortable   IN Boolean      DEFAULT TRUE
);

-----
-- get the rows --
-----
FUNCTION Get_Rows
(
    PC$LOV_Name      IN Varchar2
  ,PC$LOV_Form      IN Varchar2
  ,PN$NumPage       IN Pls_Integer
  ,PC$SearchValue   IN Varchar2      DEFAULT NULL
) RETURN PLS_INTEGER ;

-----
-- returned values --
-----
Procedure Set_Return_Values
(
    PC$LOV_Name    In Varchar2
  ,PC$LOV_Form    In Varchar2
  ,PC$Line        In Varchar2
);
Procedure Check_Value
(
    PC$LOV_Name    In Varchar2
  ,PC$LOV_Form    In Varchar2
  ,PC$LOV_Col     In Varchar2
  ,PC$Value       In Varchar2
);

```

*Show\_Lov()* is used to Show a LAF LOV. It is generally called by the KEY-LISTVAL form-level trigger, that is part of the **laf.olb** object library.

You can also use it in your own code, in a When-Button-Pressed trigger for instance.

If you want to filter the LOV at startup, use the named notation to pass the 5<sup>th</sup> argument:

```
PKG_LAF_LOV.Show_Lov
(
  PC$LOV_Name => 'LOV1'
  ,PC$LOV_Form => :SYSTEM.CURRENT_FORM
  ,PC$BeanName => 'LAF_BLOCK.BEAN_LOV'
  ,PC$Filter    => 'JA'
  ,PB$Sortable  => FALSE
);
```

*Get\_Rows()* is used to load a new set of data from the database when the LOV support a pagination system. It is called in the When-Custom-Item-Event trigger attached to the LAF LOV Bean Area.

*Set\_Return\_Values()* is used to populate the selected values in the corresponding return items. It is called in the When-Custom-Item-Event trigger attached to the LAF LOV Bean Area.

*Check\_Value()* is used for item validation through the LOV. It is call in the When-Validate-Item form-level trigger.

*It needs a package to compile in the database : PKG\_DB\_LAF\_LOV  
The script is located in the /script folder of the zip file.*

It contains PL/SQL structures, procedures and functions used to store the LAF LOV descriptions and send data to the Java Bean.

The LAF LOV feature is a set of objects you need to add to your Forms module:

- A Bean Area which Implementation Class is set to : **oracle.forms.fd.LAF\_LOV** with its When-Custom-Item-Event associated trigger.
- A Program Unit to define the LAF LOVs used in the current module.  
This procedure is called : *LAF\_PREPARE\_LOVS()* in the sample dialog shipped in the zip file - **test\_laf\_lovs\_alerts.fmb**.
- 5 form-level triggers
  - PRE-FORM
  - POST-FORM
  - KEY-LISTVAL
  - WHEN-NEW-ITEM-INSTANCE
  - WHEN-VALIDATE-ITEM

These elements are part of the new Object Group – **GRP\_LAF\_LOV** – added to the new **LOV** tab of the **laf.olb** Object Library.

◇ The **LAF\_LOV\_TEMPLATE.fmb** template has been added in the sample dialogs directory to help you creating a new module from scratch.

## How to define a LAF LOV

The **laf.olb** Object Library contains a tab called : **LOV** that contains an object group : **GRP\_LAF\_LOV**.

This object group contains a set of form-level triggers and a program unit called : **LAF\_PREPARE\_LOVS()**.

Drag this Object Group in your Forms module, then edit the procedure.

```
PROCEDURE LAF_Prepare_LOVs IS
    LR$LovItm      PKG_DB_LAF_LOV.TYP_TAB_CHAR ;
    LR$LovRec      PKG_DB_LAF_LOV.LOV_RECORD ;
    LR$MappRec     PKG_DB_LAF_LOV.TYP_TAB_LOV_MAPPING ;
    LR$Widths      PKG_DB_LAF_LOV.TYP_TAB_LOV_COL_WIDTH ;
    LR$Valids     PKG_DB_LAF_LOV.TYP_TAB_LOV_COL_ITEM_VAL ;
    LB$Code        Boolean ;
    LC$Code        Varchar2(4000) ;
    LC$LOV_Name   Varchar2(200) ;
Begin
-----
--                                     --
--   change this setting to suit your own need  --
--                                     --
-----
LC$LOV_Name := 'LOV1' ;

-- LOV description --
LR$LovRec.LOV_NAME          := LC$LOV_Name ;
LR$LovRec.LOV_FORM           := :SYSTEM.CURRENT_FORM ;
LR$LovRec.LOV_BEAN_NAME      := 'CTRL.BEAN_LOV' ;
LR$LovRec.LOV_TITLE          := 'Choose an employee' ;
LR$LovRec.LOV_SELECT         := 'Select * From EMP ' ;
LR$LovRec.LOV_PROMPT         := 'Select a row' ;
LR$LovRec.LOV_WIDTH          := 480 ;
LR$LovRec.LOV_HEIGHT          := 500 ;
LR$LovRec.LOV_X_POSITION     := -1 ; -- center the LOV on the screen
LR$LovRec.LOV_Y_POSITION     := -1 ;
LR$LovRec.LOV_MAX_COL_WIDTH  := 80 ;
LR$LovRec.LOV_COL_SEARCH      := 'ENAME' ;
LR$LovRec.LOV_VALIDATION      := 'Y' ;
LR$LovRec.LOV_PAGING          := 200 ;
LR$LovRec.LOV_SCHEME          := 'silver' ;
LR$LovRec.LOV_BUTTON1         := 'OK' ;
LR$LovRec.LOV_BUTTON2         := 'Cancel' ;

-- LOV items --
LR$LovItm(1) := 'EMP.EMPNO' ;
LR$LovItm(2) := 'EMP.ENAME' ;
LR$LovRec.LOV_ITEMS := LR$LovItm ;

-- LOV columns width --
LR$Widths(1).LOV_COLUMN      := 'ename' ;
LR$Widths(1).WIDTH            := 5 ;
LR$Widths(1).MIN_WIDTH        := 3 ;
LR$Widths(1).MAX_WIDTH        := 10 ;
LR$Widths(2).LOV_COLUMN      := 'job' ; - hidden column
LR$Widths(2).WIDTH            := 0 ;
LR$Widths(2).MIN_WIDTH        := 0 ;
LR$Widths(2).MAX_WIDTH        := 0 ;
```



```

-- LOV item mapping --
LR$MappRec(1).LOV_COLUMN      := 'EMPNO' ;
LR$MappRec(1).LOV_ITEM1       := 'EMP.EMPNO' ;
LR$MappRec(2).LOV_COLUMN      := 'ENAME' ;
LR$MappRec(2).LOV_ITEM1       := 'EMP.ENAME' ;
LR$MappRec(3).LOV_COLUMN      := 'JOB' ;
LR$MappRec(3).LOV_ITEM1       := 'EMP.JOB' ;
LR$MappRec(4).LOV_COLUMN      := 'HIREDATE' ;
LR$MappRec(4).LOV_ITEM1       := 'EMP.HIREDATE' ;
LR$MappRec(5).LOV_COLUMN      := 'COMM' ;
LR$MappRec(5).LOV_ITEM1       := 'EMP.COMM' ;

-- LOV item/column validations --
LR$Validids(1).LOV_ITEM      := 'EMP.EMPNO' ;
LR$Validids(1).LOV_COLUMN     := 'EMPNO' ;
LR$Validids(2).LOV_ITEM      := 'EMP.ENAME' ;
LR$Validids(2).LOV_COLUMN     := 'ENAME' ;

-- add the LOV to the list --
LC$Code := PKG_DB_LAF_LOV.Add_Lov
(
    'CTRL.BEAN_LOV'
    ,LC$LOV_Name
    ,:SYSTEM.CURRENT_FORM
    ,LR$LovRec
    ,'I'
) ;

If LC$Code = 'OK' Then
    -- add the LOV column width --
    LC$Code := PKG_DB_LAF_LOV.Set_Lov_Col_Width( LC$LOV_Name
                                                ,:SYSTEM.CURRENT_FORM, LR$Widths ) ;
    If LC$Code <> 'OK' Then
        PKG_LOOK_AND_FEEL.ShowError(LC$Code);
    End if ;
    -- add the LOV mapping --
    LC$Code := PKG_DB_LAF_LOV.Set_Lov_Mapping( LC$LOV_Name, :SYSTEM.CURRENT_FORM,
                                                LR$MappRec ) ;
    If LC$Code <> 'OK' Then
        PKG_LOOK_AND_FEEL.ShowError(LC$Code);
    End if ;
    -- add the item validation --
    LC$Code := PKG_DB_LAF_LOV.Set_Lov_Validators( LC$LOV_Name,
                                                :SYSTEM.CURRENT_FORM, LR$Validids ) ;
    If LC$Code <> 'OK' Then
        PKG_LOOK_AND_FEEL.ShowError(LC$Code);
    End if ;
Else
    PKG_LOOK_AND_FEEL.ShowError(LC$Code);
End if ;

End;

```



## **LOV description record:**

```
LR$LovRec.LOV_NAME          -- unique LOV name in the module
LR$LovRec.LOV_FORM           -- module name
LR$LovRec.LOV_BEAN_NAME    -- LOV Bean Area name
LR$LovRec.LOV_TITLE          -- LOV title
LR$LovRec.LOV_SELECT        -- LOV Select order
LR$LovRec.LOV_PROMPT         -- LOV Prompt
LR$LovRec.LOV_WIDTH          -- LOV Width (in pixel)
LR$LovRec.LOV_HEIGHT          -- LOV Height (in pixel)
LR$LovRec.LOV_X_POSITION     -- LOV X Position (in pixel) (1)
LR$LovRec.LOV_Y_POSITION     -- LOV Y Position (in pixel) (1)
LR$LovRec.LOV_MAX_COL_WIDTH  -- Maximum column width (in pixel)
LR$LovRec.LOV_COL_SEARCH      -- Column to search (column alias not allowed here)
LR$LovRec.LOV_VALIDATION      -- LOV for validation (Y/N)
LR$LovRec.LOV_PAGING          -- number of lines per set (pagination)
LR$LovRec.LOV_SCHEME          -- LOV Scheme
LR$LovRec.LOV_BUTTON1         -- LOV first button label (Ok) (2)
LR$LovRec.LOV_BUTTON2         -- LOV second button's label (Cancel) (2)
```

- (1) if LOV\_X\_POSITION and LOV\_Y\_POSITION < 0, the LOV is centered on the screen  
if LOV\_X\_POSITION and LOV\_Y\_POSITION = 0, the LOV is centered on the Forms window (default)
- (2) Button1 and button2 can also contains path to an icon file stored in the JAR file, in the client machine or in the Internet. In this case, the value is given in brackets. (/icon.gif)

Record elements displayed in bold are required.

Other elements are not required or/and have default values:

LOV_WIDTH	-1
LOV_HEIGHT	400
LOV_X_POSITION	0
LOV_Y_POSITION	0
LOV_MAX_COL_WIDTH	-1
LOV_VALIDATION	N
LOV_PAGING	-1

◊ See the **PKG\_DB\_LAF\_LOV** database package to see the full record structure.

## **LOV items record:**

It defines the list of every item that supports the LOV.

## LOV item mapping:

Defines the mapping between the LOV columns and the return items in the module. Each column can be mapped to one up to three items. The item must be given with its full block.item syntax. LOV\_COLUMN can contains an alias.

```
TYPE      LOV_MAPPING IS RECORD
(
    LOV_COLUMN          VARCHAR2(100)  -- LOV column name
    ,LOV_COL_MIN_WIDTH  PLS_INTEGER
    ,LOV_COL_MAX_WIDTH  PLS_INTEGER
    ,LOV_ITEM1           VARCHAR2(100)  -- First return item name
    ,LOV_ITEM2           VARCHAR2(100)  -- Second return item name
    ,LOV_ITEM3           VARCHAR2(100)  -- Third return item name
);
```

## LOV item/column validation:

Defines what LOV column is used to validate what item.

```
-- store the Item Validation correspondences --
TYPE      LOV_COL_ITEM_VAL IS RECORD
(
    LOV_ITEM           VARCHAR2(100)
    ,LOV_COLUMN         VARCHAR2(100)
);
TYPE      TYP_TAB_LOV_COL_ITEM_VAL IS TABLE OF LOV_COL_ITEM_VAL INDEX BY BINARY_INTEGER ;
```

So that, every LOV column can be used to validate any item.

Once the LOV record is completed, we add the LOV description in the database package via the *PKG\_DB\_LAF\_LOV.Add\_Lov()* function.

```
-- add the LOV to the list --
LC$Code := PKG_DB_LAF_LOV.Add_Lov
(
    'CTRL.BEAN_LOV'          -- Bean Area that implements the LOV Java Bean
    ,LC$LOV_Name             -- LOV unique name
    ,:SYSTEM.CURRENT_FORM   -- Forms module name
    ,LR$LovRec               -- LOV record
    ,'I'                     -- replace flag (R/I/C)
) ;
```

The replace flag can be one of the following:

- **R** Replace. If the LOV is already defined, it is replaced (default)
- **I** Ignore. If the LOV is already defined, it is ignored
- **C** Cancel. If the LOV is already defined, it is ignored and an error is raised

The function returns 'OK' without any error, or an error message if the process has failed.

When the LOV main record is stored, we can add the sub-records:

```

If LC$Code = 'OK' Then
    -- add the LOV mapping --
    LC$Code := PKG_DB_LAF_LOV.Set_Lov_Mapping( LC$LOV_Name, :SYSTEM.CURRENT_FORM,
                                                LR$MappRec ) ;
    If LC$Code <> 'OK' Then
        PKG_LOOK_AND_FEEL.ShowError(LC$Code) ;
    End if ;
    -- add the item validation --
    LC$Code := PKG_DB_LAF_LOV.Set_Lov_Validators( LC$LOV_Name,
                                                :SYSTEM.CURRENT_FORM, LR$Valids ) ;
    If LC$Code <> 'OK' Then
        PKG_LOOK_AND_FEEL.ShowError(LC$Code) ;
    End if ;
Else
    PKG_LOOK_AND_FEEL.ShowError(LC$Code) ;
End if ;

```

This is done like this because, at this moment, Forms (10.1.2) PL/SQL engine does not support complex PL/SQL records that contains sub-records in a single affectation.

Once the LOV is completely recorded, you can use it with the *PKG\_LAF\_LOV.Show\_Lov()* **laf.dll** procedure.

## **What can the end-user do when the LOV is displayed**

When the LOV is displayed, the end-user can do the following:

- Select the LOV column used to filter the list by right-clicking the column header's button.
- Move columns between them by dragging the headers.
- Sort the LOV column by left clicking the column header's button. First click sorts ASC, second click sorts DESC.

If the end-user enters a letter outside the search field, the LOV is filtered with the key typed.

If the end-user uses the search field, he(she) can enter several characters, then start the search by hitting the <ENTER> key.

**The LOV is case sensitive, so, searching for "a" is different than searching for "A".**

## ***INIT\_LOV***

**First step to describe the LOV in the Java Bean.**

property value: full\_LOV\_name

**full\_LOV\_name** is composed of the Forms module name and the LOV name separated by a dot

```
-- create a LOV --
Set_Custom_Property( 'LOV_BEAN', 1, 'INIT_LOV', 'MODULE1.LOVNAME1' ) ;
```

This method must be the very first called.

## ***SET\_LOV\_MAX\_COL\_WIDTH***

**Set the maximum column width for the LOV.**

The value is given in pixel.

property value: length

```
-- set the maximum column width --
Set_Custom_Property( 'LOV_BEAN', 1, 'SET_LOV_MAX_COL_WIDTH', '100') ;
```

The end-user can resize the LOV columns, so this method is used to delimit the maximum size a column can be enlarged.

## ***SET\_LOV\_SEPARATOR***

**Set the character used to separate the data in a LOV row.**

property value: separator\_character

In the database package, it is initialized to : '^':

```
GC$Sep          VARCHAR2(1) := '^' ;
```

```
-- set the data separator --
Set_Custom_Property( 'LOV_BEAN', 1, 'SET_LOV_SEPARATOR', PKG_DB_LAF_LOV.GetSeparator ) ;
```

## ***SET\_LOV\_SEARCH\_LABEL***

**Set the button's label to change the current search column.**

While the LOV is displayed, the end-user can change the column used to search the value by right-clicking the column header.

In the database package, it is initialized to : 'Define search column':

```
GC$SearchLabel    VARCHAR2(100) := 'Define search column' ;
```

Translate it directly in the package to suit your own language.

```
Set_Custom_Property( 'LOV_BEAN', 1, 'SET_LOV_SEARCH_LABEL', PKG_DB_LAF_LOV.GetSearchLabel ) ;
```

## **SET\_LOV\_ARRAY\_SIZE**

**Set the number of columns and rows that the LOV contain.**

property value: column\_number, row\_number

```
Set_Custom_Property( 'LOV_BEAN', 1, 'SET_LOV_ARRAY_SIZE', '4,100' ) ;
```

## **SET\_LOV\_HEADER**

**Set the header column names.**

property value: col1[^col2[^...]]

By default the separator character is '^'

```
Set_Custom_Property( 'LOV_BEAN', 1, 'SET_LOV_HEADER', 'title1^title2^title3' ) ;
```

## **SET\_LOV\_COLS\_TYPE**

**To set the data type for each column.**

Type can be NUMBER or CHAR:

```
-- set columns data type --
Set_Custom_Property( 'LOV_BEAN', 1, 'SET_LOV_COLS_TYPE', 'CHAR^NUMBER^NUMBER' ) ;
```



## ***SET\_LOV\_SCHEME***

**To set the LAF scheme used to decorate the LOV.**

Allowed values:

- blue
- gray
- green
- orange
- purple
- red
- silver
- XP
- yellow
- RGB,RGB

RGB,RGB is provided to give a dark and light color.

```
Set_Custom_Property( 'LOV_BEAN', 1, 'SET_LOV_SCHEME', 'purple' ) ;
```

If not provided, the current scheme is used to decorate the LOV.

## ***SET\_LOV\_BOUNDS***

**To set the LOV bounding box values.**

property value: xPos,yPos,width,height

```
Set_Custom_Property( PC$BeanName, 1, 'SET_LOV_BOUNDS', LC$Bounds ) ;
```

If xPos and yPos < 0 the LOV is centered on the screen.

If xPos and yPos = 0 the LOV is centered on the Forms window.

## ***SET\_LOV\_TITLE***

**To set the LOV title.**

```
Set_Custom_Property( 'LOV_BEAN', 1, 'SET_LOV_TITLE', 'Employee LOV' ) ;
```

## ***SET\_LOV\_PROMPT***

**Set the LOV prompt text.**

```
Set_Custom_Property( 'LOV_BEAN', 1, 'SET_LOV_PROMPT', 'Select a row' ) ;
```



## ***SET\_LOV\_COL\_SEARCH***

**Set the LOV column name used to filter the list.**

By default the search column is the first column of the query.

```
Set_Custom_Property( 'LOV_BEAN', 1, 'SET_LOV_COL_SEARCH', 'Ename' ) ;
```

## ***SET\_LOV\_BUTTON\_LABELS***

**Set the LOV button labels.**

By default the buttons have the standard dialog OK/Cancel label. This method permits to change the label of the two buttons.

```
Set_Custom_Property( 'LOV_BEAN', 1, 'SET_LOV_BUTTON_LABELS', 'Label1,Label2' ) ;
```

## ***SET\_LOV\_BUTTON\_ICONS***

**Define an image file for the LOV buttons.**

Use this method if you prefer to replace the buttons' label by an icon.

The image can be stored in a JAR - /image.gif – on the local machine – [c:/image.gif](#) – or on the Internet – [http://.../image.gif](#).

```
Set_Custom_Property( 'LOV_BEAN', 1, 'SET_LOV_BUTTON_ICONS', '/ok.gif,/cancel.gif' ) ;
```

## ***SET\_LOV\_DATA***

**Used to transfer the data from the database to the Java Bean.**

Each column value is separated by the separator define in the package - ^ - by default.

This method is invoked for each row fetched in the data set.

```
Set_Custom_Property( 'LOV_BEAN', 1, 'SET_LOV_DATA', '7623^JONES^SCOTT' ) ;
```



## **SET\_LOV\_CELL\_PROPERTY**

### **Set some LOV cell properties.**

The properties you can set are the following:

- ENABLE (true | false)
- RESIZE (true | false)
- WIDTH
- MIN\_WIDTH
- MAX\_WIDTH
- HEIGHT
- FORMAT
- TITLE
- BG\_COLOR
- FG\_COLOR
- ALIGNMENT
- FONT

First argument is the cell (column) name.

The argument separator is | (alt+124).

ALIGNMENT can be:

- LEFT
- CENTER
- RIGHT

Colors are given with the RGB format.

FORMAT must respect the Java format mask.

```
-- column width --
Set_Custom_Property('LOV_BEAN', 1, 'SET_LOV_CELL_PROPERTY', 'COL1|WIDTH|40');

-- date format -
Set_Custom_Property('LOV_BEAN', 1, 'SET_LOV_CELL_PROPERTY', 'COL1|FORMAT|dd/MM/yyyy');

-- decimal format -
Set_Custom_Property('LOV_BEAN', 1, 'SET_LOV_CELL_PROPERTY', 'COL1|FORMAT|#0.00');

-- integer format -
Set_Custom_Property('LOV_BEAN', 1, 'SET_LOV_CELL_PROPERTY', 'COL1|FORMAT|#####');
```

## **SHOW\_LOV**

### **Used to display the LOV.**

Must be the last method invoked.

If the argument is not null, it indicates the X and Y coordinates to display the LOV. These coordinates are relative to the screen.

```
Set_Custom_Property( 'LOV_BEAN', 1, 'SHOW_LOV', '' ) ;
```

## ***SET\_LOV\_LOG***

**Set the trace system ON/OFF.**

property value: true | false

```
Set_Custom_Property( 'LOV_BEAN', 1, 'SET_LOV_LOG', 'true' ) ;
```

## **The events raised by the Java Bean**

The Bean Area that implements the **oracle.forms.fd.LAF\_LOV** class needs a When-Custom-Item-Event trigger to manage the information sent back by the Java Bean.

5 different events can be raised:

- A row has been selected (LOV\_SELECTION)  
This event is raised when the end-user selects a row in the LOV. It calls the *PKG\_LAF\_LOV.Set\_Return\_Values()* laf.PLL procedure to populate the return items.
- Another data set is required (LOV\_GET\_PAGE)  
This event is raised each time the LOV requires a new data set (pagination).
- The search column has changed (LOV\_TRG\_COL\_SEARCH)  
The end-user has changed the search column.
- The sort column/order has changed (LOV\_TRG\_ORDER\_BY)  
The end-user has changed the sort column/order.
- A new filter requires a new query (LOV\_TRG\_VALUE\_SEARCH)  
The end-user has entered a new criteria that needs a new dataset (pagination)

Oracle Forms Look & Feel project

Created and maintained by Francois Degrelle

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