



laf.dll

Procedures and functions description

Content

Package PKG_LAF_LOV.....	4
Procedure Show_LOV	4
Function Get_Rows	4
Procedure Set_Return_Values	5
Package PKG_LAF_ROBOT.....	6
Procedure LAF_Robot	6
Procedure Get_Block_Bounds	6
Package PKG_LOB.....	7
Function Read_Lob	7
Function Write_Lob	7
Function Load_Video_From_Base	8
Package PKG_LOOK_AND_FEEL.....	9
Procedure Paint_Canevas	9
Procedure Paint_Block	9
Procedure Set_GUI_Properties	10
Function Open_CSS	10
Function Write_CSS	10
Function Get_Tag_Value	10
Function Get_Tag_All_Values	11
Procedure Set_Tag_Value	11
Procedure Add_Tag_Value	11
Procedure Remove_Tag_Value	12
Function Is_Tag_Exists	12
Function Get_Section_Tags	12
Procedure Set_Section_Tags	12
Procedure Fill_table	13
Function Split	13
Procedure To_String_Collection	13
Procedure Read_Image	14
Procedure Write_Image	14
Package PKG_MULTISELECT.....	15
procedure clear.....	15
procedure clear_all_blocks.....	15
function get_state.....	15
procedure set_state.....	15
procedure change_state.....	16
function get_checked_list.....	16
procedure set_visual_attribute_name.....	16
procedure set_select_visual_attributes.....	16
procedure set_multiselect_properties.....	17
procedure set_modifier.....	17
Package PKG_TOOLS.....	18
Function To_Pixel	18



Function To_Current_Coord	18
Procedure Show_In_Rect	18
Procedure init_laf_blocks.....	19
Procedure Paste_to_block.....	19
Procedure Copy_From_block	19
Procedure highlight_record.....	20
Procedure Get_BigText_Content.....	20
Package PKG_TABLE.....	22
Function Init_Table.....	22
Function Display_Table.....	24
Function Ask_Proceed_DML_Table	24
Function Proceed_DML_Table.....	25
Function Update_Table.....	25
Function Insert_Table.....	25
Function Delete_Table.....	26
Procedure Set_property	26
Function Get_Property.....	33
Procedure Display_Cell_Value.....	35
Procedure Display_Row_Value.....	35



Package PKG_LAF_LOV

This package is used to manage and display the LAF LOVs.

Procedure Show_LOV

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
);
```

PC\$LOV_Name is the LOV name you want to display.

PC\$LOV_Form is the current Form name.

PC\$BeanName is the Bean Area attached to the canvas.

PN\$LOV_X_Pos is the X coordinate you want to display the LOV.

PN\$LOV_Y_Pos is the Y coordinate you want to display the LOV.

PC\$Filter is a restricting WHERE clause.

Function Get_Rows

Select and display a set of rows from a given page.

```
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 ;
```

PC\$LOV_Name is the LOV name you want to display.

PC\$LOV_Form is the current Form name.

PN\$NumPage is the page number.

PC\$SearchValue is a restrictive WHERE clause.

Return value : number of rows fetched.



Procedure Set_Return_Values

Set the returned values to the corresponding Items.

```
Procedure Set_Return_Values
(
    PC$LOV_Name  In Varchar2
    ,PC$LOV_Form  In Varchar2
    ,PC$Line      In Varchar2
) ;
```

PC\$LOV_Name is the LOV name.

PC\$LOV_Form is the current Form name.

PC\$Line is a delimited string that contains the values.

Procedure Check_Value

Check for a value for the LOV validation.

```
Procedure Check_Value
(
    PC$LOV_Name  In  Varchar2
    ,PC$LOV_Form  In  Varchar2
    ,PC$LOV_Col   In  Varchar2
    ,PC$Value     In  Varchar2
)
```

PC\$LOV_Name is the LOV name you want to display.

PC\$LOV_Form is the current Form name.

PC\$LOV_Col is the column name.

PC\$Value is the value to search.

If the value is not found, the LOV is displayed like it is done with standard Forms LOVs.

Package PKG_LAF_ROBOT

This package is used to manage the LAF Robot features.

Procedure LAF_Robot

Handle a Robot order.

```
Procedure LAF_Robot
(
    PC$LAF_BEAN      IN  VARCHAR2
    ,PC$Function     IN  VARCHAR2
    ,PC$Params        IN  VARCHAR2
);
```

PC\$LAF_BEAN is the Bean Area attached to the canvas.

PC\$Function is the function type.

PC\$Params contains the arguments.

For more details about the functions available, see the DrawLAF documentation.

Procedure Get_Block_Bounds

Used to retrieve the bounds of a given block.

```
Procedure Get_Block_Bounds
(
    PC$BlockName   IN Varchar2
    ,PC$X          OUT Varchar2
    ,PC$Y          OUT Varchar2
    ,PC$W          OUT Varchar2
    ,PC$H          OUT Varchar2
);
```

PC\$BlockName is the Forms block name.

PC\$X is the X coordinate of the block.

PC\$Y is the Y coordinate of the block.

PC\$W is the width of the block.

PC\$H is the height of the block.



Package PKG_LOB

This package is used to manage the CLOB transfer between Forms and the database.

Function Read_Lob

Read a CLOB from the database in order to populate a Forms Item.

```
Function Read_Lob
(
    PC$Type      IN      VARCHAR2
    ,PC$Item     IN      VARCHAR2
    ,PN$Record   IN      PLS_INTEGER
    ,PC$Query    IN      VARCHAR2
    ,PC$Msg      IN OUT VARCHAR2
) Return Boolean ;
```

PC\$Type if the LOB type, and can be CLOB or NCLOB.

PC\$Item is the Forms Item that supports the CLOB (Multi-line TextItem with LAF_XP_TextArea Implementation Class).

PN\$Record is the item record number.

PC\$Query is the complete SELECT order to retrieve one and only one CLOB column from the database.

PC\$Msg contains the error message.

Return TRUE is the operation succeeded.

Function Write_Lob

Write a CLOB from a Forms Item content.

```
Function Write_Lob
(
    PC$Type      IN      VARCHAR2
    ,PC$Item     IN      VARCHAR2
    ,PN$Record   IN      PLS_INTEGER
    ,PC$Query    IN      VARCHAR2
    ,PC$Msg      IN OUT VARCHAR2
) Return Boolean ;
```

PC\$Type if the LOB type, and can be CLOB or NCLOB.

PC\$Item is the Forms Item that supports the CLOB (Multi-line TextItem with LAF_XP_TextArea Implementation Class).

PN\$Record is the item record number.

PC\$Query is the complete SELECT order to retrieve one and only one CLOB column from the database.

PC\$Msg contains the error message.

Return TRUE is the operation succeeded.



Function Load_Video_From_Base

Load a video file from a BLOB database column.

```
PROCEDURE Load_Video_From_Base
(
    PC$Bean          IN Varchar2,
    PC$WhereClause  IN Varchar2,
    PC$Name          IN Varchar2
);
```

PC\$Bean if the Bean Area name (LAF_BLOCK.LAF_BEAN).

PC\$WhereClause is the SELECT order that must return one and only one BLOB column.

PC\$Name is the name of the video clip.

Package PKG_LOOK_AND_FEEL

This package is used to read the CSS tags in order to decorate the Canvases, the table-blocks and initialize the common GUI properties.

Procedure Paint_Canevas

Paint the given canvas with the properties read in the CSS file.

```
Procedure Paint_Canevas
(
  PC$Class      IN Varchar2,
  PC$BeanName   IN Varchar2
) ;
```

PC\$Class if the CSS tag entry (of type canvas).
PC\$BeanName if the DrawLAF Bean Area attached to the canvas.

Procedure Paint_Block

Paint the given table-block.

```
Procedure Paint_Block
(
  PC$Block      IN Varchar2,
  PC$BeanName   IN Varchar2,
  PC$VA_Name    IN Varchar2,
  PC$HeadClass  IN Varchar2,
  PC$BodyClass  IN Varchar2,
  PC$TitleClass IN Varchar2 Default Null,
  PC$Title     IN Varchar2 Default Null,
  PB$ScrollBar  IN Boolean Default True,
  PB$SortBlock  IN Boolean Default Null
) ;
```

PC\$Block if the table-block to decorate.
PC\$BeanName if the DrawLAF Bean Area attached to the canvas.
PC\$Block if the Visual Attribute used to highlight the current record.
PC\$HeadClass is the CSS class entry to decorate then table-block header.
PC\$BodyClass is the CSS class entry to decorate then table-block body.
PC\$TitleClass is the CSS class entry to decorate then table-block title.
PC\$Title is the title string.
PB\$ScrollBar indicates if the table block shows a scrollbar.
PB\$SortBlock indicates if the table block can be sorted by clicking the column title.

Procedure Set_GUI_Properties

Set the common GUI properties.

```
Procedure Set_GUI_Properties
(
  PC$Class      IN Varchar2,
  PC$BeanName   IN Varchar2
) ;
```

PC\$Class is the CSS class entry that must be of GUI type

PC\$BeanName is the DrawLAF Bean Area that supports the canvas.

Function Open_CSS

Open and read a LAF dedicated CSS file.

Return TRUE is the file has been correctly read.

```
Function Open_CSS( PC$Filename IN Varchar2 )
Return Boolean ;
```

Function Write_CSS

Write a LAF dedicated CSS file.

Return TRUE is the file has been correctly written.

```
Function Write_CSS( PC$Filename IN Varchar2 )
Return Boolean ;
```

Function Get_Tag_Value

Return a given tag value.

```
Function Get_Tag_Value
(
  PC$Section  IN Varchar2,
  PC$TagName  IN Varchar2,
  PC$Default   IN Varchar2 Default 'none'
) Return Varchar2 ;
```

PC\$Section is the CSS section entry.

PC\$TagName is the CSS section tag entry.

PC\$Default is the default value returned if the tag is not found.



Function Get_Tag_All_Values

Return all values corresponding to a given tag.

The return value is a table of strings (TYP_TAB_STRINGS).

```
Function Get_Tag_All_Values
(
  PC$Section  IN Varchar2,
  PC$TagName  IN Varchar2
) Return TYP_TAB_STRINGS ;
```

PC\$Section is the CSS section entry.

PC\$TagName is the CSS section tag entry.

Procedure Set_Tag_Value

Set a specific tag value.

```
Procedure Set_Tag_Value
(
  PC$Section  IN Varchar2,
  PC$TagName  IN Varchar2,
  PC$TagValue IN Varchar2
);
```

PC\$Section is the CSS section entry.

PC\$TagName is the CSS section tag entry.

PC\$TagValue is the value to set.

Procedure Add_Tag_Value

Add a new tag in the CSS section.

```
Procedure Add_Tag_Value
(
  PC$Section  IN Varchar2,
  PC$TagName  IN Varchar2,
  PC$TagValue IN Varchar2
);
```

PC\$Section is the CSS section entry.

PC\$TagName is the CSS section tag entry.

PC\$TagValue is the value to set.



Procedure Remove_Tag_Value

Removes a tag from the CSS section.

```
Procedure Remove_Tag_Value
(
  PC$Section  IN Varchar2,
  PC$TagName  IN Varchar2
);
```

PC\$Section is the CSS section entry.
PC\$TagName is the CSS section tag entry.

Function Is_Tag_Exists

Return TRUE if the given tag exists.

```
Function Is_Tag_Exists
(
  PC$TagName  IN Varchar2
) Return Boolean ;
```

Function Get_Section_Tags

Returns all tags corresponding to a section.

```
Function Get_Section_Tags(PC$Section IN Varchar2)
Return TYP_TAB_TAG ;
```

Procedure Set_Section_Tags

Set all tags corresponding to a section.

```
Procedure Set_Section_Tags
(
  PC$Section IN Varchar2,
  PT$TTags    IN TYP_TAB_TAG
) ;
```



Procedure Fill_table

Used to colorize the odd an even rows of a table-block.

```
Procedure Fill_table
(
  PC$Type  IN Varchar2 DEFAULT 'ODD',
  PB$Start IN Boolean DEFAULT TRUE
) ;
```

Function Split

Returns a token from a delimited string.

```
Function Split
(
  PC$Chaine IN VARCHAR2,          -- input string
  PN$Pos IN PLS_INTEGER,         -- token number
  PC$Sep IN VARCHAR2 DEFAULT ',', -- separator character
) Return Varchar2 ;
```

```
Function Split
(
  PC$Chaine IN CLOB,            -- input string
  PN$Pos IN PLS_INTEGER,        -- token number
  PC$Sep IN VARCHAR2 DEFAULT ',', -- separator character
) Return Varchar2 ;
```

Procedure To_String_Collection

Returns a collection of strings from a delimited string.

```
Procedure To_String_Collection
(
  LC$String      IN Varchar2,
  PT$StringTable IN OUT NOCOPY TYP_TAB_STRINGS
) ;
```



Procedure Read_Image

Read an image from a BLOB database column.

```
Procedure Read_Image
(
  PC$Bean      IN VARCHAR2,
  PC$Image     IN VARCHAR2,
  PC$Clause    IN VARCHAR2
);
```

PC\$Bean is the DrawLAF Bean Area name.

PC\$Image is the Image name of the NEW IMAGE item.

PC\$Clause is the select order that retrieve the single BLOB column in the table.

The NEW IMAGE item must have been created with the IMG_NEW DrawLAF method.

Procedure Write_Image

Write a NEW IMAGE content to a BLOB database column.

```
Procedure Write_Image
(
  PC$Bean      IN VARCHAR2,
  PC$Image     IN VARCHAR2,
  PC$Table     IN VARCHAR2,
  PC$Column    IN VARCHAR2,
  PC$Clause    IN VARCHAR2
);
```

PC\$Bean is the DrawLAF Bean Area name.

PC\$Image is the Image name of the NEW IMAGE item.

PC\$Table is the table name that contains the BLOB column.

PC\$Column is the BLOB column name.

PC\$Clause is the WHERE clause that identifies the single row in the table.

The NEW IMAGE item must have been created with the IMG_NEW DrawLAF method.



Package PKG_MULTISELECT

This package is used to manage the multi-select feature in a table-block.

procedure clear

Remove the selection in the current record.

```
clear;
```

procedure clear_all_blocks

Remove the selection in all current form blocks.

```
clear_all_blocks;
```

function get_state

Return the current state of the given record number.

```
function get_state( numrec in pls_integer )
return pls_integer ;
```

numrec is the block record number to test.

Returns 1 if the record is marked as selected otherwise it returns 0 (zero)

procedure set_state

Set the given record state.

```
procedure set_state
(
  numrec in pls_integer,
  value in pls_integer
) ;
```

numrec is the block record number to set.

value is the value to set: 1 : selected 0 : unselected.



procedure change_state

Swap the state of the given record.

```
procedure change_state( numrec in pls_integer ) ;
```

numrec is the block record number to set.

function get_checked_list

Return the list of given block selected records.

```
function get_checked_list (blockname in varchar2 default null)
return TAB_SEL ;
```

TAB_SEL is a collection of pls_integer

procedure set_visual_attribute_name

Set the Visual Attribute name used to color the selected records.

```
procedure set_visual_attribute_name ( vaname in varchar2);
```

vaname is the name of an existing Forms Visual Attribute.

procedure set_select_visual_attributes

Set the Visual Attribute properties used to color the selected records.

```
procedure set_select_visual_attributes
(
    fontname    in varchar2 default null
  ,fontweight   in varchar2 default null
  ,fontsize    in varchar2 default null
  ,foreground   in varchar2 default null
  ,background   in varchar2 default null
);
```

fontname is the Font name.

Fontweight is the Font weight (P)lain, (B)old, (I)talic, PI and BI

fontsize is the Font size.

Foreground is the Font foreground color.

Background is the Font background color.

procedure set_multiselect_properties

Set the Visual Attribute properties from a delimited string.

```
procedure set_multiselect_properties (props in varchar2);
```

the properties in the string are separated by a - (minus).

procedure set_modifier

Set the keyboard modifier used to select/unselected the records.

```
procedure set_modifier(mdf in varchar2);
```

mdf can be:

- shift
- shift+ctrl
- alt
- ctrl



Package PKG_TOOLS

This package contains miscellaneous functions.

Function To_Pixel

Returns the pixel correspondence from current coordinate system.

```
Function To_Pixel( PN$Coord1  In Number )
Return Pls_Integer ;
```

```
Function To_Pixel
(
    PN$Coord1      In Number,
    PN$Coord2      In Number,
    PC$Separator   In Varchar2 Default '|'
)
Return Varchar2 ;
```

Function To_Current_Coord

Convert the pixel value to the current system coordinate value.

```
Function To_Current_Coord( PN$Coord1  In Number )
Return Number ;
```

```
Function To_Current_Coord
(
    PN$Coord1      In Number,
    PN$Coord2      In Number,
    PC$Separator   In Varchar2 Default '|'
)
Return Varchar2 ;
```

Procedure Show_In_Rect

Prepare the given item for the Show_In_Rect feature.

```
Procedure Show_In_Rect
(
    PC$Item    in varchar2
    ,PC$Strip  in varchar2
    ,PC$Bean   in varchar2
);
```

PC\$Item is the Forms Item name.

PC\$Strip is the delimited string that contains the strip bounds.

PC\$Bean is the DrawLAF Bean Area name.

In the PC\$Strip string, the values are separated by a - (minus).

Procedure init_laf_blocks

Display the canvases in order to use the Set_Custom_Property() without Forms error.

```
Procedure init_laf_blocks
(
    PC$Blk1    in varchar2 default null
    ,PC$Blk2    in varchar2 default null
    ,PC$Blk3    in varchar2 default null
) ;
```

PC\$Blk1 is the name of a block to ignore.

PC\$Blk2 is the name of a block to ignore.

PC\$Blk3 is the name of a block to ignore.

For Forms modules that have multiple canvases, and not all canvases are shown at startup, this procedure has to be called in the When-New-Form-Instance trigger to force the display of those hidden canvases. This done, you are able to use the Set_Custom_Property() on the Bean Area attached to those canvases.

Procedure Paste_to_block

Populate a table-block with the content of the clipboard.

```
Procedure Paste_to_block
(
    PC$Block    in varchar2
    ,PC$Bean    in varchar2
    ,PB$Header   in boolean default FALSE
    ,PN$From     in pls_integer default 1
    ,PC$Items    in varchar2 default null
    ,PC$FieldSep in varchar2 default CHR(9)
) ;
```

PC\$Block is the target block name.

PC\$Bean is the DrawLAF Bean Area name.

PB\$Header indicates if the clipboard content contains a header line.

PN\$From sets the start column to copy.

PC\$Items is a delimited string that contains the Item list to populate.

PC\$FieldSep is the field separator in the clipboard content.

Procedure Copy_From_block

Copy the table-block content in the clipboard.

```

Procedure Copy_From_block
(
  PC$Block      in varchar2
  ,PC$Bean      in varchar2
  ,PB$Header    in boolean default FALSE
  ,PN$From      in pls_integer default 1
  ,PC$Items     in varchar2 default null
  ,PC$FieldSep  in varchar2 default CHR(9)
) ;

```

PC\$Block is the target block name.

PC\$Bean is the DrawLAF Bean Area name.

PB\$Header indicates if the clipboard content contains a header line.

PN\$From sets the start column to copy.

PC\$Items is a delimited string that contains the Item list to read.

PC\$FieldSep is the field separator in the clipboard content.

Procedure highlight_record

Highlight the current record that has a gradient background.

```

Procedure highlight_record
(
  PC$Block  in varchar2
) ;

```

Use only with Items that have oracle.forms.fd.LAF_XP_TextField or forms.fd.LAF_XP_TextField Implementation Class AND a gradient background.

PC\$Block is the block name.

Procedure Get_BigText_Content

Get the full BIG TEXT content.

```

Procedure Get_BigText_Content
(
  PC$Item   in varchar2,
  PL$Clob   in out nocopy CLOB
) ;

```

PC\$Item is the Item Name with LAF_XP_TextArea Implementation Class and the SET_BIG_TEXT setting.

PL\$Clob is a Clob used to get the value stored in the BIG Text item.



This procedure can be used in a Forms named trigger used to validate a CLOB content:

```
Declare
    LL$Clob    CLOB;
    vhash    varchar2(32000);
Begin
    -- got the entire CLOB content --
    dbms_lob.createtemporary( LL$Clob, TRUE ) ;
    PKG_TOOLS.Get_BigText_Content('CTRL.T2', LL$Clob);

    -- do the validation here --
    clear_message;
    message('New text size: ' || to_char(dbms_lob.getlength(LL$Clob), '9G999G999'));

    -- clear the temporary CLOB --
    dbms_lob.trim(LL$Clob, 0);
End;
```

Package PKG_TABLE

This package contains the procedures and functions to handle the DrawLAF dynamic table-blocks.

Every Set_Custom_Property() is embedded in a Forms PL/SQL equivalent procedure.

Function Init_Table

Define a new dynamic table-block. Must be the very first method to call.

```
Procedure Init_table
(
    PC$TableName  IN Varchar2
  ,PC$Bean        IN Varchar2
  ,PC$Table       IN Varchar2
  ,PC$Title       IN Varchar2
  ,PC$Query       IN Varchar2
  ,PC$Where       IN Varchar2
  ,PC$Order       IN Varchar2
  ,PN$PageSize    IN PLS_INTEGER
  ,PC$Update      IN Varchar2    default 'true'
  ,PC$Insert      IN Varchar2    default 'true'
  ,PC$Delete      IN Varchar2    default 'true'
  ,PI$PosX        IN Pls_Integer default 1
  ,PI$PosY        IN Pls_Integer default 1
  ,PI$Width       IN Pls_Integer default 400
  ,PI$Height      IN Pls_Integer default 200
  ,PC$DecSep      IN Varchar2    default '..'
  ,PC$DateFormat  IN Varchar2    default 'dd/MM/yyyy'
  ,PC$NumFormat   IN Varchar2    default '#0.00'
  ,PC$IntFormat   IN Varchar2    default '#####'
  ,PC$Trace       IN Varchar2    default 'false'
  ,PC$BeanLog     IN Varchar2    default 'false'
);
;
```

PC\$TableName is the unique table-block name
PC\$Bean is the DrawLAF Bean Area name
PC\$Table is the database table
PC\$Title is the table-block title
PC\$Query is the full SQL query
PC\$Where is the WHERE clause
PC\$Order is the ORDER BY clause
PN\$PageSize is the page size
PC\$Update is the update flag
PC\$Insert is the insert flag
PC\$Delete is the delete flag
PI\$PosX is the X position on the canvas
PI\$PosY is the Y position on the canvas
PI\$Width is the table-block width
PI\$Height is the table-block height
PC\$DecSep is the decimal separators
PC\$DateFormat is the date format mask
PC\$NumFormat is the number format mask
PC\$IntFormat is the integer format mask
PC\$Trace is the PL/SQL library log flag
PC\$BeanLog is the Java Bean log flag



If you query all the columns of a database table, just provide the table name in the **PC\$Table** argument and let the **PC\$Query** null.

If you want only a selection of the columns, set the **PC\$Table** argument to main table name and put the full query in the **PC\$Query** argument. In this case, if you want to update the database table, you need to get the ROWID in the full query statement like the following:

```
PC$Query => 'SELECT t.ROWID,t.id,t.name,t.photo from photos t'
```

PN\$PageSize indicates how many rows to retrieve from the database for a page. While querying big tables, it could take too many time and too much memory to get all the rows at once. The minimum value is 10.

If you want to query all the rows at once, put -1

Default value is 100

PC\$Update, **PC\$Insert** and **PC\$Delete** must be 'true' or 'false'.

If the query does not retrieve the ROWID, these 3 arguments are forced to false, and the table-block is not updatable.

PC\$DateFormat is the Java date format, like 'dd/MM/yyyy'

PC\$NumFormat is the Java decimal format like '#0.00'

PC\$IntFormat is the Java integer format like '#####'

To harmonize the format between Forms and Java, you can set them in the When-New-Form-Instance trigger:

```
-----
-- formats --
--
-- see http://java.sun.com/docs/books/tutorial/i18n/format/simpleDateFormat.html
-----
-- Java format mask --
:GLOBAL.LC$DJavaFormat := 'dd/MM/yyyy' ;
:GLOBAL.LC$NJavaFormat := '#0.00' ;
:GLOBAL.LC$IJavaFormat := '#####' ;
-- PL/SQL format mask --
:GLOBAL.LC$DPLSFormat := 'dd/mm/yyyy' ;
:GLOBAL.LC$NPLSFormat := '999,990.00' ;
:GLOBAL.LC$NUMSEPARATORS := ',' ;

set_application_property(PLSQL_DATE_FORMAT, :GLOBAL.LC$DPLSFormat);
set_application_property(BUILTIN_DATE_FORMAT, :GLOBAL.LC$DPLSFormat);
forms_ddl('ALTER SESSION SET NLS_DATE_FORMAT = ''|| :GLOBAL.LC$DPLSFormat || ''');
forms_ddl('ALTER SESSION SET NLS_NUMERIC_CHARACTERS = ''|| :GLOBAL.LC$NUMSEPARATORS || ''');
```

[Link to the Java date formats](#)

[Link to the Java numeric formats](#)

Minimum call is:

```
PKG_TABLE.Init_table
(
    PC$TableName => PC$Name          -- unique table-block name
    ,PC$Bean      => 'LAF_BLOCK.LAF_BEAN'   -- Bean area
    ,PC$Table     => PC$Name          -- table to request
    ,PC$Title     => PC$Name          -- title
    ,PC$Query     => null            -- full query
    ,PC$Where     => :BL1.WHERE       -- where clause
    ,PC$Order     => :BL1.ORDER       -- order by clause
    ,PN$PageSize  => :BL1.LIMIT       -- paging size
    ,PC$Update    => :BL1.CB         -- can update
```



```

    ,PC$Trace      => 'true'                      -- trace PL/SQL
    ,PC$BeanLog    => 'true'                      -- Java Bean log
);

```

Function Display_Table

Get the rows from the database corresponding to the given page number.

```

Function Display_table
(
    PC$TableName IN Varchar2
    ,PN$PageNum   IN PLS_INTEGER
    ,PB$Init      IN boolean default true
) Return PLS_INTEGER;

```

PC\$TableName is the unique table-block name given in the *Init_table()* procedure.

PN\$PageNum is the page number to fetch from the database.

PB\$Init indicates if the initialization has to be done.

The first call of this procedure is done by the developer to display the first rows. **PN\$PageNum** is always 1 and **PB\$Init** is always true.

The consecutive calls are done through the *When-Custom-Item-Event* trigger of the DrawLAF Bean Area, while the end user scrolls the table-blocks content:

```

If (eventName='TB_GET_PAGE') THEN
    eventValues := get_parameter_list(:system.custom_item_event_parameters);
    get_parameter_attr(eventValues,'TB_TABLE_EVENT_MSG',eventValueType, p1);
    get_parameter_attr(eventValues,'TB_NAME',eventValueType, p2);
    If p1 Is Not Null Then
        -- proceed DML operations if paging defined --
        If PKG_TABLE.Ask_Proceed_DML_table(p2) Then
            If PKG_TABLE.Display_table( p2, p1, FALSE ) > 0 Then
                PKG_Table.Set_Property(p2, 'PAGE_REFRESH=' || p1);
            End if ;
        End if ;
    End if ;
END IF;

```

Returns the number of rows retrieved.

Function Ask_Proceed_DML_Table

proceed all table DML operations only if PAGESIZE > -1.

```

Function Ask_Proceed_DML_table
(
    PC$TableName IN Varchar2
) RETURN Boolean;

```

PC\$TableName is the unique table-block name given in the *Init_table()* procedure.

This function is called in the *When-Custom-Item-Event* trigger, so you should not call it directly. It returns TRUE if the pagination is set (PageSize != -1)



Function Proceed_DML_Table

proceed all table-block DML operations.

```
Function Proceed_DML_table
(
    PC$TableName IN Varchar2
    ,PB$Silent     IN Boolean default true
) Return PLS_INTEGER;
```

PC\$TableName is the unique table-block name given in the *Init_table()* procedure.
PB\$Silent indicates if debugging messages are to be displayed.

This function proceeds all required DML operations concerning the table-bloc like Updates, Inserts and Deletes by calling the *Update_Table*, *Insert_Table* and *Delete_table* functions.

You know, at any time, how many rows have been modified by the user by calling the following:

```
LN$Delete := PKG_Table.Get_Property(PC$TableName, 'COUNT_DELETE');
LN$Update := PKG_Table.Get_Property(PC$TableName, 'COUNT_UPDATE');
LN$Insert := PKG_Table.Get_Property(PC$TableName, 'COUNT_INSERT');
```

It returns the number of rows impacted by the DML orders.

To know if any kind of modification has been done by the user, call the following:

```
n := PKG_Table.Get_Property(PC$TableName, 'COUNT_CHANGES');
```

Call this function at commit time.

Function Update_Table

Proceed the updates concerning the table-block.

```
Function Update_table
(
    PC$TableName IN Varchar2
    ,PC$Rows      IN Varchar2
    ,PB$Silent     IN Boolean default true
) Return PLS_INTEGER;
```

PC\$TableName is the unique table-block name given in the *Init_table()* procedure.

PC\$Rows is the comma delimited list of rows to update.

PB\$Silent indicates if debugging messages are to be displayed.

The comma delimited list of updated rows is provided by the following:

```
LC$Rows := PKG_TABLE.Get_Property(PC$TableName, 'ROWS_CHANGED');
```

The function returns the number of rows proceeded.

Function Insert_Table

Proceed the inserts concerning the table-block.

```

Function Insert_table
(
    PC$TableName IN Varchar2
    ,PC$Rows      IN Varchar2
    ,PB$Silent    IN Boolean default true
) Return PLS_INTEGER

```

PC\$TableName is the unique table-block name given in the *Init_table()* procedure.

PC\$Rows is the comma delimited list of rows to insert.

PB\$Silent indicates if debugging messages are to be displayed.

The comma delimited list of inserted rows is provided by the following:

```
LC$Rows := PKG_TABLE.Get_Property(PC$TableName, 'ROWS_INSERTED');
```

The function returns the number of rows proceeded.

Function Delete_Table

Proceed the deletes concerning the table-block.

```

Function Delete_table
(
    PC$TableName IN Varchar2
    ,PC$Rows      IN Varchar2
    ,PB$Silent    IN Boolean default true
) Return PLS_INTEGER

```

PC\$TableName is the unique table-block name given in the *Init_table()* procedure.

PC\$Rows is the comma delimited list of rows to delete.

PB\$Silent indicates if debugging messages are to be displayed.

The comma delimited list of deleted rows is provided by the following:

```
LC$Rows := PKG_TABLE.Get_Property(PC$TableName, 'ROWS_DELETED');
```

The function returns the number of rows proceeded.

Procedure Set_property

Set a table-block property.

```

Procedure Set_Property
(
    PC$TableName IN Varchar2,
    PC$Props     IN Varchar2
);

```

PC\$TableName is the unique table-block name given in the *Init_table()* procedure.

PC\$Props is a list of properties to set.

Here is the complete property list:

TITLE

Set the table title

TITLE_ALIGN

Set the title horizontal alignment (LEFT/CENTER/RIGHT)

TITLE_SHOW

Show/Hide the table title (true/false)

TITLE_FONT

Set the title Font (font_name,weight [N|B|I|BI],size)

TITLE_COLORS

Set the title colors (RGB foreground [,RGB background])

BEAN

Set the associated (DrawLAF) Bean Area name

TABLE

Set the database table name

QUERY

Set the full SQL query

WHERE

Set the WHERE clause

ORDER

Set the ORDER BY clause

XPOS

Set the X position on the canvas in pixel

YPOS

Set the Y position on the canvas in pixel

WIDTH

Set the width in pixel

HEIGHT

set the height in pixel

PAGESIZE

Set the paging size (-1 or >=10)

To query the whole table without pagination, set to -1

PAGE_REFRESH

refresh the data with new page

UPDATE

set the update flag (true/false)

INSERT

set the insert flag (true/false)

DELETE

set the delete flag (true/false)

SET_DATA

send data to the bean

SET_IMAGE

send image content to the bean

BOUNDS

set the table-block bounds (x,y,width,height) in pixel

DECSEP

Set the decimal separators (2 characters)

DATEFORMAT

set the date format in Java format

NUMFORMAT

set the number format in Java format

INTFORMAT

set the integer format in Java format

BACKGROUND

set the background color (RGB color)

COLSMAXWIDTH

set the column max width, so that the end-user can enlarge more.

CELLPROP

set the cell property. Properties can be:

- ENABLE (true/false)
- VISIBLE (true/false)
- RESIZE (true/false)
- WITH in pixel
- MIN_WIDTH in pixel
- MAX_WIDTH in pixel
- HEIGHT in pixel
- FORMAT
- TITLE
- FGCOLOR foreground color (RGB color)
- BGCOLOR background color (RGB color)
- ALIGNMENT (LEFT/CENTER/RIGHT)
- CHECKBOX (checked value,unchecked value)
- COMBOBOX (val1[^val2[^...[^valn]]])
- IMAGE_SCALE (-1/>=0/WIDTH,-1/>=0/HEIGHT)
- FONT (name,size,weight(N/B/I/BI))
- SCROLLBARS (NEVER/AS_NEEDED/ALWAYS, NEVER/AS_NEEDED/ALWAYS)

CELLINSTPROP

set the cell instance property. Properties can be:

- ENABLE (true/false)
- FGCOLOR foreground color (RGB color)
- BGCOLOR background color (RGB color)
- ALIGNMENT (LEFT/CENTER/RIGHT)
- FONT (name,size,weight(N/B/I/BI))

e.g.:

```
PKG_Table.Set_Property(PC$Table, 'CELLINSTPROP=R' || PC$Record ||
                         'C3,BG_COLOR,r255g0b0');
PKG_Table.Set_Property(PC$Table, 'CELLINSTPROP=R' || PC$Record || 'C3,ENABLE,false');
```

CELLVALUE

set the given cell value ([s]row[s]cell[s]value)

The first character [s] is the argument separator.

e.g. 'CELLVALUE=^1^6^null'

Pay attention to the fact that the ROWID (first) column is never displayed, but is considered as the first cell. So to point to the first visible column, the cell value must be 2.

For an image column, the only authorized value is **null** to clear the image.

CURCELLVALUE

set the current cell value (value)

ROWPROP

set the row property (HEIGHT)

TOTALLINE

Set the total line columns' calculation operation :

e.g. 'C1=COUNT,C3=Avg'

Available operations are:

- COUNT
- SUM
- AVG
- MIN
- MAX

TOTALLINELABELS

Set the total line translations tool tips

e.g. 'COUNT=Count,SUM=Sum,Avg=Average,MAX=Max,MIN=Min'

TOTAL_COLORS

Set the title colors (RGB foreground [,RGB background])

HEADFG

set the header foreground color (RGB color)

HEADBG

set the header background color (RGB color)

HEADHEIGHT

set the header height

CURRENT

Set the current table-block, if you have more than one on the same canvas.

DATAFG

set the data foreground color (RGB color)

DATABG

set the data background color (RGB color)



TABLEBG

set the table background color (RGB color)

GRIDFG

set the grid color (RGB color)

SEPARATOR

set the data separator character

IMAGESIZE

set the image size (width,height) in pixel

REORDER

set the column reorder flag (true/false). If yes, the user can reorder the columns' order with the mouse.

RESIZE

set the column resize flag (true/false). If yes, the user can resize any column with the mouse.

BORDER

set the table border (excluding total line)

PANELBORDER

set the panel border (including total line)

Border description can be:

- line,width[,RGB color]
- raiseddetched
- lowereddetched
- raisedbevel
- loweredbevel
- empty

SELECTBACK

set the selected row background color (RGB color)

SELECTFORE

set the selected row foreground color (RGB color)

LOCALE

set the locale

VLINE

Show/Hide grid vertical lines (true/false)

HLINE

Show/Hide grid horizontal lines (true/false)

RESIZEMODE

set the column resizing mode.

Mode can be:

- ALL_COLUMNS
- LAST_COLUMN
- NEXT_COLUMNS
- SUBSEQUENT_COLUMNS
- OFF (default)

TRACE
set the PL/SQL trace (true/false)

VISIBLEROWS
set the number of visible rows in the table-blocks

ARRAYSIZE
set the sent data array size

HEADER
set the table header (col1[^col2[^coln]])

COLSTYPE
set the table columns data-type (TYPE[^TYPE[^...]])
The allowed types are:

- CHAR
- INTEGER
- NUMBER
- DATE
- IMAGE
- CLOB

INIT
clear the current table-block (full clear)

CLEAR_REFRESH
clear the current table-block to display a new page (partial clear, data only)

VISIBLE
show/hide the table

REMOVE
remove the table-block

LOG
set the Java Bean log

FOCUS
set the focus on given line

ADD_ROW
add a blank row at end

DEL_ROW
delete the current row

EXTEND
set the table extend flag (true/false)
If set to true, the end user can extend the table size to the whole canvas size with a Right-click in the table, another Right-click bring the table back to its original size.

EXTEND_EDITOR
set the CLOB editor extend flag (true/false)
If set to true, the user is allowed to edit the CLOB content in a larger window.

EDITOR_LABELS
Set the CLOB Editor buttons' translations (save[,cancel[,wrap]])

FIND_LABELS
Set the CLOB Editor Find box translations (title[,message])

ALERT_STRINGS

string to display the delete image alert (message, YES button label, NO button label)
The default English translations are : **Delete this image ?, &Yes** and **&No**

SHOW

Show the table (true/false)

If true, display the table-block in an independent frame.

You can set more than one property at a time by separating them with a | (alt-124) character, like:

```
'XPOS=10|YPOS=10|WIDTH=200|HEIGHT=250'
```

Examples:

```
PKG_Table.Set_Property('tb1', 'Bounds=17,100,667,253');
PKG_Table.Set_Property('tb1', 'Background=r255g255b255');
PKG_Table.Set_Property('tb1', 'Extend=true');
PKG_Table.Set_Property('tb1', 'Current=tb1');
PKG_Table.Set_Property('tb1', 'DecSep=.,');
PKG_Table.Set_Property('tb1', 'DateFormat=dd/MM/yyyy');
PKG_Table.Set_Property('tb1', 'NumFormat=#0,0');
PKG_Table.Set_Property('tb1', 'IntFormat=#####');
PKG_Table.Set_Property('tb1', 'ALERT_STRINGS=Supprimer cette image?,&Oui,&Non');

-- !! CHECKBOX must be called BEFORE PKG_Table.Display_Table() !! --
PKG_Table.Set_Property('tb1', 'CHECKBOX=C6,CHECKBOX,Yes,No');

-- set the total line --
PKG_Table.Set_Property('tb1',
'TOTALLINELABELS=COUNT=Nbre,SUM=Somme,AVG=Moy.,MAX=Max,MIN=Min');
PKG_Table.Set_Property('tb1', 'TOTALLINE=C1=COUNT,C3=Avg');

PKG_Table.Set_Property('tb1', 'TABLEBG=r255g255b255');
PKG_Table.Set_Property('tb1', 'SELECTBACK=r0g0b255');
PKG_Table.Set_Property('tb1', 'SELECTFORE=r255g255b255');

-- Always hide the ROWID column --
PKG_Table.Set_Property('tb1', 'CELLPROP=ROWID,VISIBLE,false');

PKG_Table.Set_Property('tb1', 'CELLPROP=C3,ENABLE,false');
PKG_Table.Set_Property('tb1', 'CELLPROP=C5,IMAGE_SCALE,WIDTH,-1');
PKG_Table.Set_Property('tb1', 'CELLPROP=C2,COMBOBOX,true,line 1^line 2^line 3');
PKG_Table.Set_Property('tb1', 'ROWPROP=HEIGHT,80');

-- auto resize mode --
PKG_Table.Set_Property('tb1', 'RESIZEMODE=ALL_COLUMNS');

PKG_Table.Set_Property('tb1', 'PANELBORDER=line,1,r0g0b255');
PKG_Table.Set_Property('tb1', 'BORDER=line,1,r0g0b0');

-- Set the title of the separate frame --
PKG_Table.Set_Property('tb1', 'TITLE=Content of EMP table');
-- Set the header colors --
PKG_Table.Set_Property('tb1', 'HEADFG=r0g0b255');
PKG_Table.Set_Property('tb1', 'HEADBG=r255g255b255');
PKG_Table.Set_Property('tb1', 'DATABG=r255g255b255');

-- Show the table --
PKG_Table.Set_Property('tb1', 'SHOW=false' );
PKG_Table.Set_Property('tb1', 'FOCUS=1');
```



Function Get_Property

Get a table-block property.

```
Function Get_Property
(
  PC$TableName IN Varchar2,
  PC$Prop       IN Varchar2
) Return Varchar2;
```

PC\$TableName is the unique table-block name given in the *Init_table()* procedure.
PC\$Prop is the property name.

Here is the complete property list:

- TITLE
- BEAN
- TABLE
- QUERY
- WHERE
- XPOS
- YPOS
- WIDTH
- HEIGHT
- ORDER
- PAGESIZE
- CURRENTPAGE
- UPDATE
- INSERT
- DELETE
- BOUNDS
- DECSEP
- DATEFORMAT
- NUMFORMAT
- INTFORMAT
- BACKGROUND
- CURRENT
- COLSMAXWIDTH
- CELLPROP
- CELLVALUE
- CURCELLVALUE

- ROWPROP
- TOTALLINE
- TOTALLINELABELS
- FILTER
- HEADFG
- HEADBG
- HEADHEIGHT
- TOTALHEADHEIGHT
- DATAFG
- DATABG
- TABLEBG
- GRIDFG
- CELLPOS
- ROWPOS
- SEPARATOR
- IMAGESIZE
- REORDER
- RESIZE
- BORDER
- SELECTBACK
- SELECTFORE
- LOCALE
- VLINE
- HLINE
- RESIZEMODE
- VISIBLEROWS
- VISIBLE
- EXTEND
- TRACE
- LOG
- ROWVAL
- CELLVAL
- COUNT_INSERT
- COUNT_UPDATE
- COUNT_DELETE
- COUNT_CHANGES
- ROWS_CHANGED
- ROWS_INSERTED

- ROWS_DELETED
- IDENTIFIED_ROW

Procedure Display_Cell_Value

Display the content of the given table-block cell in a pop-up dialog.

```
Procedure Display_Cell_Value
(
    PC$TableName IN Varchar2
    ,PN$Row      IN Pls_integer
    ,PN$Cell      IN Pls_integer
);
```

PC\$TableName is the unique table-block name given in the *Init_table()* procedure.

PN\$Row is the row number

PN\$Cell is the cell number

Procedure Display_Row_Value

Display the content of the given table-block row in a pop-up dialog.

```
Procedure Display_Row_Value
(
    PC$TableName IN Varchar2
    ,PN$Row      IN Pls_integer
);
```

PC\$TableName is the unique table-block name given in the *Init_table()* procedure.

PN\$Row is the row number

Example of some PL/SQL code to build and show a dynamic table:

```
PKG_Table.Set_Property('tbl1', 'Remove');

PKG_TABLE.Init_table
(
    PC$TableName => 'tbl1'                      -- internal table name
    ,PC$Bean      => 'LAF_BLOCK.LAF_BEAN'        -- Bean area
    ,PC$Table     => 'tbl1'                      -- table to request
    ,PC>Title     => 'tbl1'                      -- title
    ,PC$Query    => null                         -- full query
    ,PC$Where    => null                         -- where clause
    ,PC$Order    => null                         -- order by clause
    ,PN$PageSize => 500                          -- paging size
    ,PC$Update   => 'true'                        -- can update
    ,PC$Trace    => 'true'                        -- trace PL/SQL
    ,PC$BeanLog  => 'true'                        -- Java Bean log
);

PKG_Table.Set_Property('tbl1', 'Bounds=17,100,667,253');
PKG_Table.Set_Property('tbl1', 'Background=r255g255b255');
```

```

PKG_Table.Set_Property('tbl1', 'Extend=true');
PKG_Table.Set_Property('tbl1', 'Current=' || 'tbl1');
PKG_Table.Set_Property('tbl1', 'DecSep=' || :GLOBAL.LC$NUMSEPARATORS);
PKG_Table.Set_Property('tbl1', 'DateFormat=' || :GLOBAL.LC$DJavaFormat);
PKG_Table.Set_Property('tbl1', 'NumFormat=' || :GLOBAL.LC$NJavaFormat);
PKG_Table.Set_Property('tbl1', 'IntFormat=' || :GLOBAL.LC$IJavaFormat);
-- !! CHECKBOX must be called BEFORE PKG_Table.Display_Table() !!
PKG_Table.Set_Property('tbl1', 'CHECKBOX=C6,CHECKBOX,Y,N');

-----
-- Get table's rows --
-----
LN$Rows := PKG_TABLE.Display_table( 'tbl1', PN$Page, TRUE ) ;

-- set the total line --
PKG_Table.Set_Property('tbl1',
    'TOTALLINELABELS=COUNT=Nbre,SUM=Somme,AVG=Moy.,MAX=Max,MIN=Min');
PKG_Table.Set_Property('tbl1', 'TOTALLINE=C1=COUNT,C3=AVG');

PKG_Table.Set_Property('tbl1', 'TABLEBG=r255g255b255');
PKG_Table.Set_Property('tbl1', 'SELECTBACK=r0g0b255');
PKG_Table.Set_Property('tbl1', 'SELECTFORE=r255g255b255');

-- Always hide the ROWID column --
PKG_Table.Set_Property('tbl1', 'CELLPROP=ROWID,VISIBLE,false');
-- set some other column properties
PKG_Table.Set_Property('tbl1', 'CELLPROP=C1,ALIGNMENT,CENTER');
PKG_Table.Set_Property('tbl1', 'CELLPROP=C5,IMAGE_SCALE,WIDTH,-1');
PKG_Table.Set_Property('tbl1', 'CELLPROP=C2,COMBOBOX,true,line 1^line 2^line 3');
PKG_Table.Set_Property('tbl1', 'CELLPROP=C7,SCROLLBARS,NEVER,AS_NEEDED');

-- auto resize mode --
PKG_Table.Set_Property('tbl1', 'RESIZEMODE=ALL_COLUMNS');

PKG_Table.Set_Property('tbl1', 'PANELBORDER=line,1,r0g0b255');
PKG_Table.Set_Property('tbl1', 'BORDER=line,1,r0g0b0');

-- Set the title of the separate frame --
PKG_Table.Set_Property('tbl1',
    'TITLE='|| 'Content of ' || :TABLE_NAME || ' table');
-- Set the header colors --
PKG_Table.Set_Property('tbl1', 'HEADFG=r0g0b255');
PKG_Table.Set_Property('tbl1', 'HEADBG=r255g255b255');
PKG_Table.Set_Property('tbl1', 'DATABG=r255g255b255');

-----
-- Show the table --
-----
PKG_Table.Set_Property('tbl1', 'SHOW='||:BL1.GRPBT );
PKG_Table.Set_Property('tbl1', 'FOCUS=1');

```

See the test_laf_DynTable.fmb sample dialog shipped with the zip file.

Oracle Forms Look & Feel project

Created and maintained by Francois Degrelle

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



Oracle Forms Look & Feel project