



LAF TextArea PJC - public properties

These properties can be set or read from any multi-line Text Item whose Implementation Class property is set to:

`oracle.forms.fd.LAF_XP_TextArea`

Oracle Developer Forms Runtime - Web

Action Edition Interrogation Bloc Enregistrement Champ Aide Fenêtre

Look and Feel : Text Items

Blink

Blink text

Blink rate Start Stop

Cursor

Selection Mark selection

Activity

Link

Link

Link URL

Regex

Regex pattern

test

Enreg. : 1/1



Content

Regular Expressions properties.....	4
SET_REGEX_PATTERN	4
SET_REGEX_MESSAGE	4
GET_REGEX_RESULT	4
Cursor/selection properties.....	6
SET_CURSOR_POSITION	6
INSERT_TEXT	6
INSERT_TEXT_AT_POSITION	6
GET_CURSOR_POSITION	6
SET_SELECTION_INDEX	7
GET_SELECTION_INDEX	7
GET_SELECTION_VALUE	7
Big Text (CLOB/NCLOB) properties.....	8
SET_BIG_TEXT	8
BIG_ADD_TEXT	8
BIG_SHOW	9
BIG_CLEAR	9
BIG_SET_OPAQUE	10
BIG_GET_TEXT	10
BIG_GET_LENGTH	11
BIG_READ_FROM_FILE	11
BIG_WRITE_TO_FILE	11
BIG_APPEND_TO_FILE	12
BIG_VALIDATION_TRIGGER (since 7.3.4)	12
BIG_FOCUS_LOST	12
BIG_SET_UPDATABLE (since 7.3.4)	13
BIG_GET_HASH (since 7.3.4)	13
BIG_SEARCH (since 7.3.4)	13
BIG_SEARCH_COLOR (since 7.3.4)	13
BIG_SET_BORDER (since 7.3.4)	14
BIG Item validation.....	14
HTML Content properties.....	15
SET_HTML_CONTENT	15
SET_HTML_GRANTS	16
SET_HTML_EDIT_ATTRS	16
GET_HTML_CONTENT	17
Gradient background properties.....	18
SET_GRADIENT.....	18
SET_BORDER.....	18
SET_CURRENT.....	18
Miscellaneous properties.....	20
SET_MAX_CHAR	20



ENABLE EVENTS20
SET_HORIZONTAL_SCROLLBAR21
SET_VERTICAL_SCROLLBAR21
SET_FOCUS_BACKGROUND.....22
SET_LOG22

Regular Expressions properties

SET_REGEX_PATTERN

Set the Regular Expression pattern.

property value: `regex_pattern`

```
Set_Custom_Property( '...', 1, 'SET_REGEX_PATTERN',  
                    '^([a-z0-9._-]+@[a-z0-9._-]{2,}\.[a-z]{2,4}$' ) ;
```

The result of the expression's evaluation can be obtained with the *GET_REGEX_RESULT* method, or in the *When-Custom-Item-Event* Forms trigger if the *ENABLE_EVENTS()* method is used on this text Item.

SET_REGEX_MESSAGE

Set the error message when the pattern validation fails.

```
Set_Custom_Property( '...', 1, 'SET_REGEX_MESSAGE'  
                    , 'incorrect pattern. must be [A-Z][a-z][a-z]' ) ;
```

GET_REGEX_RESULT

Returns the result of the Regular Expression's evaluation.

```
varchar2 := Get_Custom_Property( '...', 1, 'GET_REGEX_RESULT' ) ;
```

The returned value is 'true' if the pattern is found or 'false'

GET_REGEX_GROUP

Returns the group in the string that matches the pattern.

```
varchar2 := Get_Custom_Property( '...', 1, 'GET_REGEX_GROUP' ) ;
```



GET_REGEX_POSITION

Returns start,end position in the string that matches the pattern.

Returned value is a coma delimited string : start_pos,end_pos

```
varchar2 := Get_Custom_Property( '...', 1, 'GET_REGEX_POSITION' ) ;
```



Cursor/selection properties

SET_CURSOR_POSITION

Move the caret to the given position.

```
Set_Custom_Property( '...', 1, 'SET_CURSOR_POSITION', '12' ) ;
```

It accepts two special keywords : START or END

INSERT_TEXT

Insert the given text at the current cursor position.

```
Set_Custom_Property( '...', 1, 'INSERT_TEXT', 'Hello world' ) ;
```

INSERT_TEXT_AT_POSITION

Insert the given text at the given position.

property_value : position,text

```
Set_Custom_Property( '...', 1, 'INSERT_TEXT_AT_POSITION', '12,Hello world' ) ;
```

GET_CURSOR_POSITION

Returns the current cursor position.

```
Varchar2 := Get_Custom_Property( '...', 1, 'GET_CURSOR_POSITION' ) ;
```



SET_SELECTION_INDEX

Pre-select the given characters in the text.

property_value : start_index,end_index

```
-- select the substring from the 12th to the 18th characters --  
Set_Custom_Property( '...', 1, 'SET_SELECTION_INDEX', '12,18' ) ;
```

GET_SELECTION_INDEX

Returns the current selected index.

the returning value is a coma delimited string : start_pos,end_pos

```
Varchar2 := Get_Custom_Property( '...', 1, 'GET_SELECTION_INDEX' ) ;
```

GET_SELECTION_VALUE

Returns the current selected value.

```
Varchar2 := Get_Custom_Property( '...', 1, 'GET_SELECTION_VALUE' ) ;
```



Big Text (CLOB/NCLOB) properties

SET_BIG_TEXT

Indicates that the Text Field will handle huge texts as CLOB or NCLOB. You need to set this method first before using any other BIG methods.

```
Set_Custom_Property( '...', 1, 'SET_BIG_TEXT', '' ) ;
```

BIG_ADD_TEXT

Add some text at the end of the Text Item. It is generally used in a loop to populate the Text Item.

```
For i In 1 .. 3000 Loop
    Set_Custom_Property( '...',1, 'BIG_ADD_TEXT'
        , i || ' This is a very long text' || chr(10));
End loop;
```

Here is the complete code to read the (N)CLOB content from the database, then send it to the Text Item:

```
Declare
    LN$Length pls_integer ;
    LC$Chunk Varchar2(32767) ;
    LC$Req Varchar2(1024);
Begin
    -----
    -- Build the Select SQL order --
    -----
    LC$Req := 'Select TEXT From TEST_CLOB Where ID=1' ;

    Set_Custom_Property('CTRL.BIG',1, 'BIG_CLEAR','');

    -----
    -- select the database CLOB column --
    -----
    If pkg_laf.select_clob(LC$Req) then
        Loop
            -----
            -- get the chunks... --
            -----
            LC$Chunk := pkg_laf.get_chunk ;
            Exit when LC$Chunk is null ;
            -----
            -- ... then send them to the Bean --
            -----
        End Loop;
    End If;
```



```

    Set_Custom_Property('CTRL.BIG',1, 'BIG_ADD_TEXT',LC$Chunk);
End loop ;
-----
-- display the complete text --
-----
Set_Custom_Property('CTRL.BIG',1, 'BIG_SHOW','');
Else
message('problem reading CLOB');
End if ;
End;

```

It requires the PKG_LAF database package to run.

The first step is select the LOB content from the table via the *PKG_LAF.SELECT_CLOB()* function. (use the *PKG_LAF.SELECT_NCLOB()* function to select a *NCLOB* column). The SQL order must point to one and only one row in the table.

Next step is to get the chunks to send them to the Text Item via the *PKG_LAF.GET_CHUNK()* procedure.

Last step is to show the Text Item new content via the *BIG_SHOW()* method.

This complete task is encapsulated in the PKG_LOB.Read_Lob() laf.pll's package.

```

Declare
  LC$Msg      Varchar2(1024);
  LB$Ok       Boolean ;
Begin
  LB$Ok :=  PKG_LOB.Read_Lob('CLOB','CTRL.BIG',1
                          , 'Select TEXT From TEST_CLOB Where ID=1',LC$Msg) ;
  If Not LB$Ok Then
    Message(LC$Msg);
  End if ;
End;

```

BIG_SHOW

Ends the Text Item population, then show the content.

```

Set_Custom_Property( '...', 1, 'BIG_SHOW', '' ) ;

```

BIG_CLEAR

Clear the Text Item content.

```

Set_Custom_Property( '...', 1, 'BIG_CLEAR', '' ) ;

```



BIG_SET_OPAQUE

Set the big Text Item background opaque true|false, so that you can see the canvas background behind if false.

```
Set_Custom_Property( '...', 1, 'BIG_SET_OPAQUE', 'false' ) ;
```

BIG_GET_TEXT

**Returns a text chunk back to Forms.
You need to use it in a loop until it returns the whole Text item content.**

```
Loop
  LC$Chunk := Get_Custom_Property('...',1,'BIG_GET_TEXT') ;
  Exit When LC$Chunk Is Null ;
  -- work with the chunk ... --
End loop;
```

Here is the code required to update a database CLOB with the current content:

```
Declare
  LC$Chunk  Varchar2(32767) ;
  LC$Req    Varchar2(1024) ;
  LC$Msg    Varchar2(4000);
Begin
  -----
  -- set the UPDATE SQL order --
  -----
  LC$Req := 'Update TEST_CLOB Set TEXT = :1 Where ID=1' ;
  -----
  -- Init the transfer for a NCLOB --
  -----
  pkg_laf.Init_Transfer_lob ( 'CLOB' );
  Loop
    LC$Chunk := Get_Custom_Property('CTRL.BIG',1,'BIG_GET_TEXT') ;
    Exit When LC$Chunk Is Null ;
    -----
    -- Send the chunks to the database --
    -----
    pkg_laf.Set_Chunk ( LC$Chunk ) ;
  End loop;
  -----
  -- Update the CLOB/NCLOB database column --
  -----
  LC$Msg := pkg_laf.Transfer( LC$Req, 'CLOB' ) ;
  If LC$Msg = 'OK' Then
    Forms_ddl('Commit') ;
  Else
    Message(LC$Msg);
  End if ;
End;
```

It requires the PKG_LAF database package to run.

The first step is to inform the package that you want to transfer a new content via the `PKG_LAF.INIT_TRANSFER_LOB()` procedure.

Next step is to send the chunks to the package via the `PKG_LAF.SET_CHUNK()` procedure.

Last step is to finalize the transfer via the `PKG_LAF.TRANSFER()` function.

This complete task is encapsulated in the `PKG_LOB.Write_Lob()` laf.pll's package.

```
Declare
  LC$Msg      Varchar2(1024);
  LB$Ok       Boolean ;
Begin
  LB$Ok :=  PKG_LOB.Write_Lob('CLOB','CTRL.BIG',1
                             , 'Update TEST_CLOB set TEXT = :1 Where ID=1',LC$Msg) ;
  If Not LB$Ok Then
    Message('Error:' ||LC$Msg);
  End if ;
End;
```

BIG_GET_LENGTH

Get the Big Text Item length.

```
v_size := Get_Custom_Property( '...', 1, 'BIG_GET_LENGTH' ) ;
```

BIG_READ_FROM_FILE

Populate the Text Area with the content of a file stored on the client machine on on a Web server.

```
-- read from a local machine file --
Set_Custom_Property( '...', 1, 'BIG_READ_FROM_FILE', 'c:/file.txt' ) ;

-- read from a Web server --
Set_Custom_Property( '...', 1, 'BIG_READ_FROM_FILE', 'http://.../file.txt' ) ;
```

BIG_WRITE_TO_FILE

Write the Text Area content to a local machine file.

```
Set_Custom_Property( '...', 1, 'BIG_WRITE_TO_FILE', 'c:/file.txt' ) ;
```



BIG_APPEND_TO_FILE

Add the Text Area content at the end of a local machine file.

```
Set_Custom_Property( '...', 1, 'BIG_APPEND_TO_FILE', 'c:/file.txt' );
```

BIG_VALIDATION_TRIGGER (since 7.3.4)

Indicates what Forms named trigger to use to validate the content.

```
Set_Custom_Property( '...', 1, 'BIG_VALIDATION_TRIGGER', 'VALIDATE_BIGTEXT' );
```

(see the test_laf_big_text.fmb sample dialog)

BIG_FOCUS_LOST

Is the event raised back to Forms when the Big Text Item has lost the focus.

You can get it via the When-Custom-Item-Event trigger:

```
Declare
  eventName varchar2(30) := :system.custom_item_event;
  eventValues ParamList;
  eventValueType number;
  p1          varchar2(1024);
  p2          varchar2(1024);
  p3          varchar2(1024);
  p4          varchar2(1024);
  p5          varchar2(1024);
BEGIN

  eventValues := get_parameter_list(:system.custom_item_event_parameters);
  -----
  -- BIG Text --
  -----

  If (eventName = 'BIG_FOCUS_LOST') then
    get_parameter_attr(eventValues,'ITEM_NAME',eventValueType, p1);
    get_parameter_attr(eventValues,'BIG_UPDATED',eventValueType, p2);
    get_parameter_attr(eventValues,'BIG_TRG_NAME',eventValueType, p3);
    If p2 = 'true' and p3 is not null Then
      execute_trigger(p3);
    End if ;
  END IF;
END;
```

ITEM_NAME is the Forms Item name that raised the event.

BIG_UPDATED is set to true if the content has been updated.

BIG_TRG_NAME is the named Forms trigger given via the BIG_VALIDATION_TRIGGER method.



BIG_SET_UPDATABLE (since 7.3.4)

Indicates if the content can be updated.

```
Set_Custom_Property( '..', 1, 'BIG_SET_UPDATABLE', 'false' );
```

(see the test_laf_big_text.fmb sample dialog)

BIG_GET_HASH (since 7.3.4)

Returns the BIG text hash.

It is used to see if the content has changed and has to be saved.

```
Declare
  hash varchar2(20);
Begin
  hash := Get_Custom_Property( '..', 1, 'BIG_GET_HASH' );
End;
```

(see the test_laf_big_text.fmb sample dialog)

BIG_SEARCH (since 7.3.4)

Search for a given text in the BIG text content.

You move from word to word by pressing the F3 key, then stop searching by pressing the Escape key.

```
Set_Custom_Property( '..', 1, 'BIG_SEARCH', 'Blablabla' );
```

(see the test_laf_big_text.fmb sample dialog)

BIG_SEARCH_COLOR (since 7.3.4)

Set the color used to highlight a word found with the search feature.

```
Set_Custom_Property( '..', 1, 'BIG_SEARCH_COLOR', 'r255g150b150' );
```

The default color used is the current scheme color.



BIG_SET_BORDER (since 7.3.4)

Set the BIG text border type.

Border can be:

- line [,width[,rgbColor]]
- raisedetched
- loweredetched
- raisedbevel
- loweredbevel
- null

```
Set_Custom_Property( '...', 1, 'BIG_SET_BORDER', 'line,2,r255g100b100');
```

The default line width/color is 1, black.

BIG Item validation

As the BIG Item feature is built around a Java Swing JTextPane component, it does not use the underlying Forms Text Field, reason why it does not fire the standard Forms triggers, like When-Validate-Item.

So, you need more work to have the validation step working.

When the BIG text loses the focus, an event is sent back to Forms – BIG_FOCUS_LOST – with information about what Forms Item raised the event, and if a named Forms trigger exist to execute the validation code. This named trigger has to be passed in a previous BIG_VALIDATION_TRIGGER method.

The code of this trigger has to rebuild the whole CLOB content from the PJC in a local CLOB variable:

```
Declare
    LL$Clob CLOB;
Begin
    -- get the entire CLOB content -
    dbms_lob.createtemporary( LL$Clob, TRUE ) ;
    PKG_TOOLS.Get_BigText_Content('CTRL.BIG', LL$Clob);

    -- do the validation here with the LL$Clob variable --
    ...
    ...

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

With the CLOB variable, you can use any of the DBMS_LOB method.

HTML Content properties

SET_HTML_CONTENT

Prepare the Text Item to display and/or edit(*) some HTML content.

Property : allow_edit, follow_link, can_extend

Set `allow_edit` to true to allow the user to modify the content.

Set `follow_link` to true to allow the user to click and follow HTML links.

Set `can_extend` to true to allow the user to double-click the Text Item so that it will fit the whole canvas size. Another double-click will restore the initial size.

```
Set_Custom_Property( '...', 1, 'SET_HTML_CONTENT', 'true,true,true' ) ;
```

If you don't want to change any precedent setting for a keyword, put a - instead.

```
Set_Custom_Property( '...', 1, 'SET_HTML_CONTENT', '-,-,true' ) ;
```

Once the Text Item is configured to display the HTML content, set the content by assigning the value to the Text Item:

```
:block.item := '<html><body> ... </body></html>' ) ;
```

The content can be some HTML text or an URL:

- `http://...`
- `ftp://...`
- `www....`
- `file:///...`
- `/forms...`

```
:block.item := 'http://download.oracle.com/docs/html/B16010_04/toc.htm' ) ;
```

The special case of the URL starting by `/forms` allows to reach documents stored on the Application Server in the virtual directories defined in the `forms.conf` file:

```
:block.item := '/forms/html/doc.html' ) ;
```

If you want to assign the content immediately after the use of the `SET_HTML_CONTENT()` method, follow it by a `synchronize` instruction:

```
Set_Custom_Property( '...', 1, 'SET_HTML_CONTENT', 'false,true' ) ;  
Synchronize;  
:block.item := 'http://download.oracle.com/docs/html/B16010_04/toc.htm' ) ;
```



(*) Edit options are very limited. You can add/remove text parts, and perform the following formatting actions on the selected text:

- bold **Ctrl+b**
- italic **Ctrl+i**
- underlined **Ctrl+u**
- left alignment **Ctrl+l**
- center alignment **Ctrl+c**
- right alignment **Ctrl+r**
- font family & size **Ctrl+f**
- font color **Ctrl+o**

SET_HTML_GRANTS

Grant rights to the HTML content.

Property : `allow_edit, follow_link, can_extend`

Set `allow_edit` to true to allow the user to modify the content.

Set `follow_link` to true to allow the user to click and follow HTML links.

Set `can_extend` to true to allow the user to double-click the Text Item so that it will fit the whole canvas size. Another double-click will restore the initial size.

```
Set_Custom_Property( '...', 1, 'SET_HTML_GRANTS', 'true,true,false' ) ;
```

SET_HTML_EDIT_ATTRS

Grant edition rights to the HTML content.

You can allow four classes of modification by setting them the yes or no value:

- style (Bold, Italic and Underline)
- align (Left, Center and Right alignment)
- font (Font family and size)
- color (Font color)

```
Set_Custom_Property( '...', 1, 'SET_HTML_EDIT_ATTRS',  
                    , 'style=yes,align=yes,font=no,color=no' ) ;
```



GET_HTML_CONTENT

Get the modified HTML content.

If the HTML content can be modified, you can get the new value:

```
value := Get_Custom_Property( '...', 1, 'GET_HTML_CONTENT' ) ;
```



Gradient background properties

SET_GRADIENT

Set the item gradient background.

property_value : gradient_1,gradient_2[,gradient_3[,gradient_4]]

gradient_1 is the first color of the gradient

gradient_2 is the second color of the gradient

gradient_3 and gradient_4, not mandatory are the gradients used to colorize the current record.

If gradient_3 is not given, it takes the value of gradient_2

If gradient_4 is not given, it takes the value of gradient_1

```
Set_Custom_Property( 'EMP.EMPNO', ALL_RECORDS, 'SET_GRADIENT'  
                    , '#A4BEFF,#fff,#CCDCFF,#467BFF' ) ;
```

To erase the gradient, then return to the original Text Item, set the value to 'null'

```
Set_Custom_Property( 'EMP.EMPNO', ALL_RECORDS, 'SET_GRADIENT' , 'null' ) ;
```

SET_BORDER

Draw a colored border around the item.

property_value : width,color

```
Set_Custom_Property( 'EMP.EMPNO', ALL_RECORDS, 'SET_BORDER' , '1,#fff' ) ;
```

SET_CURRENT

Highlight the current record.

property_value : true | false

It should be used in a *When-New-Record-Instance* trigger of the block:

```

Declare
    LN$Pos pls_integer ;
    LN$Rec pls_integer := Get_Block_Property( :system.current_block
                                              , CURRENT_RECORD) ;
    LC$Itms Varchar2(1000) := 'EMPNO,ENAME,JOB,MGR,HIREDATE';
Begin
    -----
    -- highlight the current record --
    -----
    If get_block_property(:system.trigger_block,TOP_RECORD) > 1 Then
        LN$Pos := LN$Rec - get_block_property(:system.trigger_block,TOP_RECORD) + 1 ;
    Else
        LN$Pos := LN$Rec ;
    End if;

    -- set all records current to FALSE --
    For i IN 1 .. 5 Loop
        Set_Custom_Property(:system.current_block ||'. '
            || pkg_look_and_feel.split(LC$Itms,i), ALL_ROWS, 'SET_CURRENT', 'false');
    End loop;
    -- set current record to TRUE --
    For i IN 1 .. 5 Loop
        Set_Custom_Property(:system.current_block ||'. '
            || pkg_look_and_feel.split(LC$Itms,i), LN$Pos, 'SET_CURRENT', 'true');
    End loop;
End;

```



Miscellaneous properties

SET_MAX_CHAR

Set the maximum number of characters allowed.

```
Set_Custom_Property( '...', 1, 'SET_MAX_CHAR', '30' ) ;
```

ENABLE_EVENTS

Enable/disable messages to be sent back to Forms.

property_value : Forms_item_name,true|false

```
Set_Custom_Property( '...', 1, 'ENABLE_EVENTS', 'CTRL.TEXT4,true' ) ;
```

Events that can be sent back to Forms are the following:

- cursor_position
- last key pressed (int value, char value, keyboard modifier)

These events are captured in the *When-Custom-Item-Event* Forms trigger:



When-Custom-Item-Event Trigger:

```
Declare
  BeanValListHdl ParamList;
  paramType      Number;
  EventName      VarChar2(60);
  ItemName       Varchar2(61);
  param1         Varchar2(1000);
  param2         Varchar2(1000);
  param3         Varchar2(1000);
Begin
  Clear_Message;
  BeanValListHdl := get_parameter_list(:system.Custom_Item_Event_Parameters);
  EventName      := :SYSTEM.Custom_Item_Event;

  -- cursor position message --
  If (EventName = 'CURSOR_MESSAGE') then
    get_parameter_attr(BeanValListHdl,'ITEM_NAME',ParamType, itemname);
    get_parameter_attr(BeanValListHdl,'CURSOR_POSITION',ParamType, param1);
    If itemname = 'ctrl.multiline' Then
      :CTRL.DISPLAY := 'Item-> ' || itemname || CHR(10) || :CTRL.DISPLAY ;
      :CTRL.DISPLAY := 'Position-> ' || param1 || CHR(10) || :CTRL.DISPLAY ;
    End if ;
  -- key message --
  ElseIf (EventName = 'KEY_MESSAGE') then
    get_parameter_attr(BeanValListHdl,'ITEM_NAME',ParamType, itemname);
    get_parameter_attr(BeanValListHdl,'KEY_CODE',ParamType, param1);
    get_parameter_attr(BeanValListHdl,'KEY_CHAR',ParamType, param2);
    get_parameter_attr(BeanValListHdl,'KEY_MODIFIER',ParamType, param3);
    If itemname = 'ctrl.multiline' Then
      :CTRL.DISPLAY := 'Item-> ' || itemname
        || ' Code-> ' || param1
        || ' Char-> ' || param2
        || ' Modifier-> ' || param3 || CHR(10) || :CTRL.DISPLAY ;
    End if ;
  End if;
  Synchronize ;
End;
```

SET_HORIZONTAL_SCROLLBAR

Show/hide the horizontal scrollbar.

```
Set_Custom_Property( '...', 1, 'SET_HORIZONTAL_SCROLLBAR', 'true' );
```

SET_VERTICAL_SCROLLBAR

Show/hide the vertical scrollbar.

```
Set_Custom_Property( '...', 1, 'SET_VERTICAL_SCROLLBAR', 'true' );
```



SET_FOCUS_BACKGROUND

Show/hide the enclosing border when the item has the focus.

By default, the value is false.

```
Set_Custom_Property( '..', 1, 'SET_FOCUS_BACKGROUND', 'true' ) ;
```

SET_LOG

Enable/disable the log messages.

```
Set_Custom_Property( '..', 1, 'SET_LOG', 'true' ) ;
```



Oracle Forms Look & Feel project
Created and maintained by Francois Degrelle
[Oracle Forms L&F Web site](#)