

FI report using the ALV OM model

Applies to: SAP ABAP.

Summary

This is an FI report displaying asset data along with the WBS element information. The output is an ALV list created using the new ALV object model. This report is on the same lines as that of an ALV OM template provided by Mr Thomas Jung in an earlier article on this topic. In this program I have covered briefly the main features of the ALV OM model. There are many more exciting features waiting to be explored !

Author: Ravi Shankar Rajan

Company: GMI

Created on: 23rd August 2006

Author Bio

Ravi Shankar Rajan is a senior developer analyst working for GMI.

```

*&-----*
*& Report  Z_TEST_ZFIUASSETC_REPORT
*&
*&-----*
*&
*&
*&-----*
*****
* AUTHOR      : Ravi Shankar Rajan
* FIRM        : GMI
* DATE        : 08/2006
* PURPOSE     : Test report using the ALV OM model
* TASK ID     :
*****

```

```
REPORT z_test_zfiuassetc_report.
```

```
TABLES: anla, sscrfields.
```

```
INCLUDE <color>.
```

```
INCLUDE <icon>.
```

```
INCLUDE <symbol>.
```

```
TYPES: BEGIN OF t_asset.
```

```
* Required fields - 12
```

```
TYPES : exception  TYPE char1.
```

```

TYPES:  aktiv  TYPE aktivd.      "Asset capitalization date
TYPES:  anln1  TYPE anln1.      "Main Asset Number - key(non-editable)
TYPES:  anln2  TYPE anln2.      "Asset Subnumber - key(non-editable)
TYPES:  bukrs  TYPE bukrs.      "Company Code - key(non-editable)
TYPES:  ankl1  TYPE ankl1.      "Asset class
TYPES:  ord42  TYPE ord42.      "Smart entity
TYPES:  gdlgrp TYPE gdlgrp.     "Tax Facility Code
TYPES:  kostl  TYPE kostl.      "Cost Center
TYPES:  werks  TYPE werks_d.    "Plant
TYPES:  stort  TYPE stort.      "Asset location
TYPES:  txt50  TYPE txa50_anlt. "Asset description
TYPES:  posnr  TYPE am_posnr.   "WBS element - internal key (numc 8)

```

```

TYPES:   posid   TYPE am_posid.   "WBS element - external key (char 24)
* Optional fields - 17
TYPES:   sernr   TYPE sernr.      "Serial number
TYPES:   invnr   TYPE invnr_anla. "Inventory number
TYPES:   invzu   TYPE invzu_anla. "Supplementary inventory note
TYPES:   kfzkkz  TYPE am_kfzkkz.  "License plate no. of vehicle
TYPES:   ord41   TYPE ord41.      "Platform
TYPES:   ord43   TYPE ord43.      "Class Code
TYPES:   ord44   TYPE ord44.      "Location Type
TYPES:   liefc   TYPE liefc.      "Name of asset supplier
TYPES:   herst   TYPE herst.      "Manufacturer of asset
TYPES:   typbz   TYPE typbz_anla. "Asset type name
TYPES:   izwek   TYPE izwek.      "Reason for investment
TYPES:   ivdat   TYPE ivdat_anla. "Last inventory date
TYPES:   gsber   TYPE gsber.      "Business Area
TYPES:   kostlv  TYPE kostlv.     "Cost center responsible for asset
TYPES:   raumn   TYPE raumnr.     "Room
TYPES:   fiamt   TYPE fiamt.      "Local tax office
TYPES:   txa50   TYPE txa50_more. "Additional asset description
* Layout style
TYPES:   celltab TYPE lvc_t_styl. "Layout style
TYPES : aggr TYPE c .
TYPES : t_color TYPE lvc_t_scol.
TYPES: END OF t_asset.

TYPES: BEGIN OF t_prps.
TYPES:   posid   TYPE ps_posid.   "WBS Element
TYPES:   prart   TYPE ps_prart.   "Project type
TYPES: END OF t_prps.

DATA: gt_asset TYPE STANDARD TABLE OF t_asset.
DATA: gt_table TYPE REF TO cl_salv_table.

SELECT-OPTIONS s_user FOR anla-ernam MATCHCODE OBJECT user_addr
               NO INTERVALS NO-EXTENSION MODIF ID m2. "User name
SELECT-OPTIONS s_date FOR anla-erdat MODIF ID m2. "Date
PARAMETERS p_pspid2 TYPE ps_pspid MODIF ID m2. "Project definition

SELECTION-SCREEN SKIP 2.

```

```

PARAMETERS p_max TYPE i DEFAULT 200 MODIF ID m2. "Maximum number of records
DATA : gv_records TYPE sy-dbcnt.
CONSTANTS: gc_true  TYPE sap_bool VALUE 'X',
           gc_false TYPE sap_bool VALUE space.
PARAMETERS : p_var TYPE slis_vari. "Layout Setting
DATA: gs_asset TYPE t_asset,
      gv_count TYPE i.

```

```

CLASS lcl_handle_events DEFINITION DEFERRED.
DATA: gr_events TYPE REF TO lcl_handle_events.

```

```

*-----*
*       CLASS lcl_handle_events DEFINITION
*-----*
* §5.1 define a local class for handling events of cl_salv_table
*-----*

```

```

CLASS lcl_handle_events DEFINITION.

```

```

    PUBLIC SECTION.

```

```

    METHODS:

```

```

        on_user_command FOR EVENT added_function OF cl_salv_events
            IMPORTING e_salv_function,

```

```

        on_before_salv_function FOR EVENT before_salv_function OF
cl_salv_events

```

```

            IMPORTING e_salv_function,

```

```

        on_after_salv_function FOR EVENT after_salv_function OF cl_salv_events

```

```

            IMPORTING e_salv_function,

```

```

        on_double_click FOR EVENT double_click OF cl_salv_events_table

```

```

            IMPORTING row column,

```

```

        on_link_click FOR EVENT link_click OF cl_salv_events_table

```

```

            IMPORTING row column.

```

```

ENDCLASS.                                "lcl_handle_events DEFINITION

```

```

*-----*
*       CLASS lcl_handle_events IMPLEMENTATION

```

```

*-----*
* §5.2 implement the events for handling the events of cl_salv_table
*-----*
CLASS lcl_handle_events IMPLEMENTATION.
  METHOD on_user_command.
    PERFORM show_function_info USING e_salv_function 'Own Function'.
  ENDMETHOD.                                "on_user_command

  METHOD on_before_salv_function.
    PERFORM show_function_info USING e_salv_function 'Before ALV Function'.
  ENDMETHOD.                                "on_before_salv_function

  METHOD on_after_salv_function.
    PERFORM show_function_info USING e_salv_function 'After ALV Function'.
  ENDMETHOD.                                "on_after_salv_function

  METHOD on_double_click.
    PERFORM show_cell_info USING row column 'Execute a Function, Double-
    Click or Hotspot-Click'.
  ENDMETHOD.                                "on_double_click

  METHOD on_link_click.
    PERFORM show_cell_info USING row column 'Execute a Function, Double-
    Click or Hotspot-Click'.
  ENDMETHOD.                                "on_single_click
ENDCLASS.                                   "lcl_handle_events IMPLEMENTATION

INITIALIZATION.
  PERFORM get_default_layout.

AT SELECTION-SCREEN ON VALUE-REQUEST FOR p_var.
  PERFORM f4_layouts.

START-OF-SELECTION.
  PERFORM select_data.

END-OF-SELECTION.
  PERFORM display_fullscreen.

```

```

*   gt_table->display( ).

*&-----*
*&   Form select_data
*&-----*
*   text
*-----*
* --> p1      text
* <-- p2      text
*-----*

FORM select_data .
  DATA: ls_asset TYPE t_asset,
         ws_index TYPE i,
         lt_celltab TYPE lvc_t_styl,
         lt_anlz TYPE STANDARD TABLE OF anlz,
         ls_anlz TYPE anlz,
         ws_where TYPE string,
         ls_pspnr TYPE ps_posnr,
         lt_pspnr TYPE STANDARD TABLE OF ps_posnr,
         lr_posnr TYPE RANGE OF anla-posnr,
         ls_posnr LIKE LINE OF lr_posnr,
         lt_color  TYPE lvc_t_scol,
         ls_color  TYPE lvc_s_scol,
         l_erg     TYPE i VALUE 3.

  IF p_pspid2 IS INITIAL.
    SELECT * INTO CORRESPONDING FIELDS OF TABLE gt_asset
           UP TO p_max ROWS FROM anla
           WHERE ernam IN s_user
           AND   erdat IN s_date.

  ELSE.
    CONCATENATE p_pspid2(7) '%' INTO ws_where.
    SELECT pspnr INTO TABLE lt_pspnr FROM prps
           WHERE posid LIKE ws_where.

    REFRESH: lr_posnr.
    ls_posnr-sign = 'I'.
    ls_posnr-option = 'EQ'.
    LOOP AT lt_pspnr INTO ls_pspnr.

```

```

ls_posnr-low = ls_pspnr.
APPEND ls_posnr TO lr_posnr.
ENDLOOP.

SELECT * INTO CORRESPONDING FIELDS OF TABLE gt_asset
      UP TO p_max ROWS FROM anla
      WHERE ernam IN s_user
      AND   erdat IN s_date
      AND   posnr IN lr_posnr.
ENDIF.

IF sy-subrc = 0.
  gv_records = sy-dbcnt.

  LOOP AT gt_asset INTO ls_asset.

    SELECT * INTO TABLE lt_anlz FROM anlz
          WHERE bukrs = ls_asset-bukrs
          AND   anln1 = ls_asset-anln1
          AND   anln2 = ls_asset-anln2
          AND   bdatu > sy-datum.

    IF sy-subrc = 0.
      READ TABLE lt_anlz INTO ls_anlz INDEX 1.
      PERFORM zf_anlz_to_asset USING ls_anlz CHANGING ls_asset.

      CALL FUNCTION 'CONVERSION_EXIT_ABPSP_OUTPUT'
        EXPORTING
          input  = ls_asset-posnr
        IMPORTING
          output = ls_asset-posid.

      MODIFY gt_asset FROM ls_asset.
    ENDIF.
  ENDLOOP.
ENDIF.

LOOP AT gt_asset INTO gs_asset.
  l_erg = sy-tabix MOD 3.
  CLEAR lt_color.

```

```

CLEAR ls_color.
gs_asset-exception = l_erg + 1.
IF gs_asset-anln1 = '000001180882'. "Add your own data to test this !!!
*..set the color of a complete row
*   (2) set the color information to the row in the output
*       table.
*       note: do not fill the parameter FNAME for setting the
*           color of the complete row
*           fill the parameter FNAME for setting the color
*           of a cell (Row and Column are definied)
*   ls_color-fname      = 'ANLN1'.
*   ls_color-color-col = col_total.
*   ls_color-color-int = 1.
*   ls_color-color-inv = 0.
*   APPEND ls_color TO lt_color.
ENDIF.

ls_color-fname      = 'ANLN2'.
ls_color-color-col = col_key.
ls_color-color-int = 1.
ls_color-color-inv = 0.
APPEND ls_color TO lt_color.

gs_asset-t_color = lt_color.

gv_count = gv_count + 1.
gs_asset-aggr = gv_count.
MODIFY gt_asset FROM gs_asset TRANSPORTING aggr t_color exception.
ENDLOOP.
ENDFORM.                " select_data
*&-----*
*&   Form  zf_anlz_to_asset
*&-----*
*   text
*-----*
*   -->PS_ANLZ      text
*   -->PS_LS_ASSET text
*-----*
FORM zf_anlz_to_asset USING ps_anlz TYPE anlz

```



```
CHANGING ps_ls_asset TYPE t_asset.
```

```
ps_ls_asset-kostl = ps_anlz-kostl.  
ps_ls_asset-werks = ps_anlz-werks.  
ps_ls_asset-stort = ps_anlz-stort.  
ps_ls_asset-kfzkz = ps_anlz-kfzkz.  
ps_ls_asset-gsber = ps_anlz-gsber.  
ps_ls_asset-kostlv = ps_anlz-kostlv.  
ps_ls_asset-raumn = ps_anlz-raumn.
```

```
ENDFORM. " zf_anlz_to_asset
```

```
*&-----*  
*& Form display_grid  
*&-----*  
* text  
*-----*  
* --> p1 text  
* <-- p2 text  
*-----*
```

```
FORM display_fullscreen .
```

```
DATA: lr_columns TYPE REF TO cl_salv_columns_table,  
lr_column TYPE REF TO cl_salv_column_table,  
lr_sorts TYPE REF TO cl_salv_sorts,  
lr_sort TYPE REF TO cl_salv_sort,  
lr_aggrs TYPE REF TO cl_salv_aggregations,  
lr_aggr TYPE REF TO cl_salv_aggregation.  
DATA: lr_content TYPE REF TO cl_salv_form_element.  
DATA: ls_color TYPE lvc_s_colo.  
DATA : lr_grid TYPE REF TO cl_salv_form_layout_grid.
```

```
TRY.
```

```
cl_salv_table=>factory(  
IMPORTING  
r_salv_table = gt_table  
CHANGING  
t_table = gt_asset ).
```

```
CATCH cx_salv_msg.
```

```
ENDTRY.
```

```

*Setting Own PF-STATUS
*... §3 Functions
*... §3.1 activate ALV generic Functions
*... §3.2 include own functions by setting own status
  gt_table->set_screen_status(
    pfstatus      = 'ZSALV_STANDARD' "Copy from standard PF Status
"SALV_STANDARD"
    report        = sy-repid
    set_functions = gt_table->c_functions_all ).

```

```

*Fill the sort table

```

```

  lr_sorts = gt_table->get_sorts( ).

```

```

  TRY.

```

```

    CALL METHOD lr_sorts->add_sort

```

```

      EXPORTING

```

```

        columnname = 'AKTIV'

```

```

        position   = 1

```

```

        sequence   = if_salv_c_sort=>sort_up

```

```

        subtotal   = if_salv_c_bool_sap=>>true

```

```

        group      = 1

```

```

        obligatory = if_salv_c_bool_sap=>>true

```

```

      RECEIVING

```

```

        value      = lr_sort.

```

```

    CATCH cx_salv_not_found .

```

```

    CATCH cx_salv_existing .

```

```

    CATCH cx_salv_data_error .

```

```

  ENDTRY.

```

```

  TRY.

```

```

    CALL METHOD lr_sorts->add_sort

```

```

      EXPORTING

```

```

        columnname = 'ANLN1'

```

```

        position   = 2

```

```

        sequence   = if_salv_c_sort=>sort_up

```

```

        subtotal   = if_salv_c_bool_sap=>>false

```

```

        group      = if_salv_c_sort=>group_none

```

```

        obligatory = if_salv_c_bool_sap=>>true

```

```

      RECEIVING

```

```

        value      = lr_sort.

```

```

    CATCH cx_salv_not_found .

```

```

    CATCH cx_salv_existing .
    CATCH cx_salv_data_error .
ENDTRY.

*Columns
lr_columns = gt_table->get_columns( ).
lr_columns->set_optimize( gc_true ).
*... §4.1 set exception column
TRY.
    lr_columns->set_exception_column( 'EXCEPTION' ).
    CATCH cx_salv_data_error.                "#EC NO_HANDLER
ENDTRY.
TRY.
    lr_column ?= lr_columns->get_column( 'EXCEPTION' ).
    lr_column->set_short_text( 'MY EXCEPT' ).
    lr_column->set_medium_text( 'MY EXCEPTION' ).
    lr_column->set_long_text( 'MY EXCEPTION COLUMN' ).
    CATCH cx_salv_not_found.                "#EC NO_HANDLER
ENDTRY.

TRY.
    lr_column ?= lr_columns->get_column( 'AGGR' ).
    lr_column->set_short_text( 'AGGREGATE' ).
*    lr_column->set_medium_text( 'AGGREGATE' ).
*    lr_column->set_long_text( 'AGGREGATE' ).
*    lr_column->set_cell_type( if_salv_c_cell_type=>hotspot ).
    CATCH cx_salv_not_found.
ENDTRY.

*Set the color for a column
*This is one more way of coloring a column without any conditions
TRY.
    lr_column ?= lr_columns->get_column( 'BUKRS' ).
    ls_color-col = col_negative.
    ls_color-int = 0.
    ls_color-inv = 0.
    lr_column->set_color( ls_color ).
    CATCH cx_salv_not_found.

```

```

ENDTRY.

*... §5.2 set the color of a complete row
*   (1) register the column in which the color information
*       for the row is held
TRY.
    lr_columns->set_color_column( 'T_COLOR' ).
    CATCH cx_salv_data_error.
ENDTRY.

*Aggregations
lr_aggrs = gt_table->get_aggregations( ).

TRY.
    CALL METHOD lr_aggrs->add_aggregation
        EXPORTING
            columnname = 'AGGR'
            aggregation = if_salv_c_aggregation=>total
        RECEIVING
            value = lr_aggr.
    CATCH cx_salv_data_error .
    CATCH cx_salv_not_found .
    CATCH cx_salv_existing .
ENDTRY.

*Form Events : TOP_OF_LIST AND END_OF_LIST

*1st METHOD OF CREATING TOP-OF-LIST.
*   PERFORM create_alv_form_content_tol USING space CHANGING lr_content.
*   gt_table->set_top_of_list( lr_content ).

*2ND METHOD OF CREATING TOP-OF-LIST
PERFORM create_top_of_list CHANGING lr_grid.
gt_table->set_top_of_list( lr_grid ).

*Interactive events
*... §6 register to the events of cl_salv_table
DATA: lr_events TYPE REF TO cl_salv_events_table.

```

```

lr_events = gt_table->get_event( ).

CREATE OBJECT gr_events.

*... §6.1 register to the event USER_COMMAND
  SET HANDLER gr_events->on_user_command FOR lr_events.
*... §6.2 register to the event BEFORE_SALV_FUNCTION
  SET HANDLER gr_events->on_before_salv_function FOR lr_events.
*... §6.3 register to the event AFTER_SALV_FUNCTION
  SET HANDLER gr_events->on_after_salv_function FOR lr_events.
*... §6.4 register to the event DOUBLE_CLICK
  SET HANDLER gr_events->on_double_click FOR lr_events.
**... §6.5 register to the event LINK_CLICK
*  SET HANDLER gr_events->on_link_click FOR lr_events.

*... set list title
  DATA: lr_display_settings TYPE REF TO cl_salv_display_settings,
         l_title TYPE lvc_title.

  l_title = 'Test Report using ALV OM Model'.
  lr_display_settings = gt_table->get_display_settings( ).
  lr_display_settings->set_list_header( l_title ).
*Set the striped pattern
  CALL METHOD lr_display_settings->set_striped_pattern
    EXPORTING
      value = 'X'.
  .
*Process the variant for future save
  PERFORM process_variant.
  gt_table->display( ).

ENDFORM.                " display_grid
*&-----*
*&      Form  create_alv_form_content_tol
*&-----*
*      text
*-----*
*      -->I_PRINT      text

```

```

*      -->CR_CONTENT text
*-----*
FORM create_alv_form_content_tol
      USING i_print TYPE sap_bool
      CHANGING cr_content TYPE REF TO cl_salv_form_element.

DATA: lr_header TYPE REF TO cl_salv_form_header_info,
      l_text     TYPE string.

*... create header information
CASE i_print.
  WHEN space.
    CONCATENATE 'TOP_OF_LIST' text-h01 INTO l_text SEPARATED BY space.
  WHEN gc_true.
    CONCATENATE 'TOP_OF_LIST_PRINT' text-h01 INTO l_text SEPARATED BY
space.
ENDCASE.

CREATE OBJECT lr_header
  EXPORTING
    text      = l_text
    tooltip = l_text.

cr_content = lr_header.

ENDFORM.          " create_alv_form_content_tol

*&-----*
*&      Form create_top_of_list
*&-----*
*      text
*-----*
*      -->LR_GRID      text
*-----*

FORM create_top_of_list CHANGING lr_grid TYPE REF TO
cl_salv_form_layout_grid.

DATA : lr_grid1 TYPE REF TO cl_salv_form_layout_grid,
      lr_flow  TYPE REF TO cl_salv_form_layout_flow,
      lr_label TYPE REF TO cl_salv_form_label,

```

```

        lr_text TYPE REF TO cl_salv_form_text,
        l_text TYPE string.

CREATE OBJECT lr_grid.
CALL METHOD lr_grid->create_header_information
    EXPORTING
        row      = 1
        column   = 1
    * ROWSPAN =
    * COLSPAN =
        text     = 'ALV Object Model Example'
        tooltip  = 'ALV Object Model Example'.
    * receiving
    * r_value =
    * .
CALL METHOD lr_grid->create_grid
    EXPORTING
        row      = 2
        column   = 1
    * ROWSPAN =
    * COLSPAN =
    RECEIVING
        r_value = lr_grid1
    .
CALL METHOD lr_grid->create_text
    EXPORTING
        row      = 2
        column   = 1
    * ROWSPAN =
        colspan  = 2
        text     = 'XYZ International Mumbai'
        tooltip  = 'XYZ International Mumbai'
    RECEIVING
        r_value = lr_text
    .
CALL METHOD lr_grid->create_flow
    EXPORTING
        row      = 3
        column   = 1

```

```

*   ROWSPAN =
*   COLSPAN =
RECEIVING
    r_value = lr_flow
.
CALL METHOD lr_flow->create_label
EXPORTING
*   POSITION      =
    text         = 'Program:'
    tooltip      = 'Program:'
*   R_LABEL_FOR =
RECEIVING
    r_value      = lr_label
.
CALL METHOD lr_flow->create_text
EXPORTING
*   POSITION      =
    text         = sy-cprog
    tooltip      = sy-cprog
RECEIVING
    r_value      = lr_text
.

CALL METHOD lr_grid->create_flow
EXPORTING
    row          = 4
    column       = 1
*   ROWSPAN     =
*   COLSPAN     =
RECEIVING
    r_value      = lr_flow
.
CALL METHOD lr_flow->create_label
EXPORTING
*   POSITION      =
    text         = 'System:'
    tooltip      = 'System:'
*   R_LABEL_FOR =
RECEIVING

```



```

        r_value      = lr_label
        .
CALL METHOD lr_flow->create_text
EXPORTING
*   POSITION =
        text      = sy-sysid
        tooltip   = sy-sysid
RECEIVING
        r_value   = lr_text
        .

CALL METHOD lr_grid->create_flow
EXPORTING
        row       = 4
        column    = 2
*   ROWSPAN =
*   COLSPAN =
RECEIVING
        r_value   = lr_flow
        .

CALL METHOD lr_flow->create_label
EXPORTING
*   POSITION      =
        text      = 'Client:'
        tooltip   = 'Client:'
*   R_LABEL_FOR =
RECEIVING
        r_value   = lr_label
        .

CALL METHOD lr_flow->create_text
EXPORTING
*   POSITION =
        text      = sy-mandt
        tooltip   = sy-mandt
RECEIVING
        r_value   = lr_text
        .

CALL METHOD lr_grid->create_flow

```

```

EXPORTING
    row      = 4
    column   = 3
*   ROWSPAN =
*   COLSPAN =
RECEIVING
    r_value = lr_flow
.

CALL METHOD lr_flow->create_label
EXPORTING
*   POSITION    =
    text      = 'Created by:'
    tooltip   = 'Created by:'
*   R_LABEL_FOR =
RECEIVING
    r_value    = lr_label
.

CALL METHOD lr_flow->create_text
EXPORTING
*   POSITION =
    text    = sy-uname
    tooltip = sy-uname
RECEIVING
    r_value = lr_text
.

CALL METHOD lr_grid->create_flow
EXPORTING
    row      = 5
    column   = 1
*   ROWSPAN =
*   COLSPAN =
RECEIVING
    r_value = lr_flow
.

DATA : datel(12) TYPE c,
      timel(8)  TYPE c.

```

```

WRITE: sy-datum TO datel,
       sy-uzeit TO time1.
DATA : tzonesys TYPE tznzonesys.
SELECT SINGLE tzonesys FROM ttzcu INTO tzonesys.
CALL METHOD lr_flow->create_label
  EXPORTING
*   POSITION      =
    text         = 'Date:'
    tooltip      = 'Date:'
*   R_LABEL_FOR =
  RECEIVING
    r_value      = lr_label
    .
CALL METHOD lr_flow->create_text
  EXPORTING
*   POSITION =
    text     = datel
    tooltip  = datel
  RECEIVING
    r_value  = lr_text
    .
CALL METHOD lr_grid->create_flow
  EXPORTING
    row      = 5
    column   = 2
*   ROWSPAN =
*   COLSPAN =
  RECEIVING
    r_value  = lr_flow.
CALL METHOD lr_flow->create_label
  EXPORTING
*   POSITION      =
    text         = 'Time:'
    tooltip      = 'Time:'
*   R_LABEL_FOR =
  RECEIVING

```

```

        r_value      = lr_label
        .
CALL METHOD lr_flow->create_text
EXPORTING
*   POSITION =
        text      = tzonesys
        tooltip   = tzonesys
RECEIVING
        r_value   = lr_text
        .

*IF sy-timlo NE sy-uzeit.
WRITE : sy-datlo TO datel,
        sy-timlo TO timel.

CALL METHOD lr_grid->create_flow
EXPORTING
        row       = 6
        column    = 1
*   ROWSPAN =
*   COLSPAN =
RECEIVING
        r_value   = lr_flow.

CALL METHOD lr_flow->create_label
EXPORTING
*   POSITION      =
        text      = 'Local Date:'
        tooltip   = 'Local Date:'
*   R_LABEL_FOR =
RECEIVING
        r_value   = lr_label
        .
CALL METHOD lr_flow->create_text
EXPORTING
*   POSITION =
        text      = datel
        tooltip   = datel
RECEIVING

```

```

        r_value = lr_text.

CALL METHOD lr_grid->create_flow
EXPORTING
    row      = 6
    column   = 2
*   ROWSPAN =
*   COLSPAN =
RECEIVING
    r_value = lr_flow.

CALL METHOD lr_flow->create_label
EXPORTING
*   POSITION   =
    text      = 'Local Time:'
    tooltip   = 'Local Time:'
*   R_LABEL_FOR =
RECEIVING
    r_value   = lr_label
.
CALL METHOD lr_flow->create_text
EXPORTING
*   POSITION =
    text     = time1
    tooltip  = time1
RECEIVING
    r_value  = lr_text.
CALL METHOD lr_flow->create_text
EXPORTING
*   POSITION =
    text     = sy-zonlo
    tooltip  = sy-zonlo
RECEIVING
    r_value  = lr_text.

*ENDIF.
ENDFORM.                "create_top_of_list
*&-----*
```

```

*&      Form  show_cell_info
*&-----*
*      text
*-----*
FORM show_cell_info USING i_row      TYPE i
                        i_column TYPE lvc_fname
                        i_text  TYPE string.

DATA: l_row_string TYPE string,
      l_col_string TYPE string,
      l_row        TYPE char128.

WRITE i_row TO l_row LEFT-JUSTIFIED.

CONCATENATE text-i02 l_row INTO l_row_string SEPARATED BY space.
CONCATENATE text-i03 i_column INTO l_col_string SEPARATED BY space.

MESSAGE i000(0k) WITH i_text l_row_string l_col_string.

ENDFORM.              " show_cell_info
*&-----*
*&      Form  show_function_info
*&-----*
*      text
*-----*
*      -->I_FUNCTION text
*      -->I_TEXT      text
*-----*
FORM show_function_info USING i_function TYPE salv_de_function
                        i_text  TYPE string.

DATA: l_string TYPE string.

CONCATENATE i_text i_function INTO l_string SEPARATED BY space.

MESSAGE i000(0k) WITH l_string.

ENDFORM.              " show_function_info
*&-----*

```

```

*&      Form  process_variant
*&-----*
*      text
*-----*
*      -->P_P_VAR  text
*-----*
FORM process_variant.
  DATA : lr_layout TYPE REF TO cl_salv_layout,
         ls_key     TYPE salv_s_layout_key.
  CALL METHOD gt_table->get_layout
    RECEIVING
      value = lr_layout.
*Set the layout key
  ls_key-report = sy-cprog.
  CALL METHOD lr_layout->set_key
    EXPORTING
      value = ls_key.
*Set usage of default layout
  CALL METHOD lr_layout->set_default
    EXPORTING
      value = abap_true.
*Set layout save restriction
  CALL METHOD lr_layout->set_save_restriction
    EXPORTING
      value = if_salv_c_layout=>restrict_none.
*Set initial layout
  IF p_var IS NOT INITIAL.
    CALL METHOD lr_layout->set_initial_layout
      EXPORTING
        value = p_var.

  ENDIF.
ENDFORM.          " process_variant
*&-----*
*&      Form  f4_layouts
*&-----*
*      text
*-----*
*      --> p1      text

```

```

* <-- p2          text
*-----*
FORM f4_layouts .
  DATA :
    ls_layout TYPE salv_s_layout_info,
    ls_key    TYPE salv_s_layout_key.

    ls_key-report = sy-cprog.
    CALL METHOD cl_salv_layout_service=>f4_layouts
      EXPORTING
        s_key      = ls_key
        layout     = p_var
        restrict   = if_salv_c_layout=>restrict_none
      RECEIVING
        value      = ls_layout.
    p_var = ls_layout-layout.

ENDFORM.                " f4_layouts
*&-----*
*&      Form  get_default_layout
*&-----*
*      text
*-----*
*      -->P_P_VAR  text
*-----*
FORM get_default_layout.
  DATA :
    ls_layout TYPE salv_s_layout_info,
    ls_key    TYPE salv_s_layout_key.

    ls_key-report = sy-cprog.
    CALL METHOD cl_salv_layout_service=>get_default_layout
      EXPORTING
        s_key          = ls_key
        restrict       = if_salv_c_layout=>restrict_none
        mandt          = sy-mandt
        bypass_buffer  = if_salv_c_bool_sap=>false
      RECEIVING
        value          = ls_layout.

```



```
p_var = ls_layout-layout.  
ENDFORM.                " get_default_layout
```

Related Content

Please include at least three references to SDN documents or web pages.

1. [ALV OM Template](#)
2. [Reference 2](#)

Copyright

© Copyright 2006 SAP AG. All rights reserved.

No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

Some software products marketed by SAP AG and its distributors contain proprietary software components of other software vendors.

Microsoft, Windows, Outlook, and PowerPoint are registered trademarks of Microsoft Corporation.

IBM, DB2, DB2 Universal Database, OS/2, Parallel Sysplex, MVS/ESA, AIX, S/390, AS/400, OS/390, OS/400, iSeries, pSeries, xSeries, zSeries, z/OS, AFP, Intelligent Miner, WebSphere, Netfinity, Tivoli, Informix, i5/OS, POWER, POWER5, OpenPower and PowerPC are trademarks or registered trademarks of IBM Corporation.

Adobe, the Adobe logo, Acrobat, PostScript, and Reader are either trademarks or registered trademarks of Adobe Systems Incorporated in the United States and/or other countries.

Oracle is a registered trademark of Oracle Corporation.

UNIX, X/Open, OSF/1, and Motif are registered trademarks of the Open Group.

Citrix, ICA, Program Neighborhood, MetaFrame, WinFrame, VideoFrame, and MultiWin are trademarks or registered trademarks of Citrix Systems, Inc.

HTML, XML, XHTML and W3C are trademarks or registered trademarks of W3C®, World Wide Web Consortium, Massachusetts Institute of Technology.

Java is a registered trademark of Sun Microsystems, Inc.

JavaScript is a registered trademark of Sun Microsystems, Inc., used under license for technology invented and implemented by Netscape.

MaxDB is a trademark of MySQL AB, Sweden.

SAP, R/3, mySAP, mySAP.com, xApps, xApp, SAP NetWeaver, and other SAP products and services mentioned herein as well as their respective logos are trademarks or registered trademarks of SAP AG in Germany and in several other countries all over the world. All other product and service names mentioned are the trademarks of their respective companies. Data contained in this document serves informational purposes only. National product specifications may vary.

These materials are subject to change without notice. These materials are provided by SAP AG and its affiliated companies ("SAP Group") for informational purposes only, without representation or warranty of any kind, and SAP Group shall not be liable for errors or omissions with respect to the materials. The only warranties for SAP Group products and services are those that are set forth in the express warranty statements accompanying such products and services, if any. Nothing herein should be construed as constituting an additional warranty.

These materials are provided "as is" without a warranty of any kind, either express or implied, including but not limited to, the implied warranties of merchantability, fitness for a particular purpose, or non-infringement.

SAP shall not be liable for damages of any kind including without limitation direct, special, indirect, or consequential damages that may result from the use of these materials.

SAP does not warrant the accuracy or completeness of the information, text, graphics, links or other items contained within these materials. SAP has no control over the information that you may access through the use of hot links contained in these materials and does not endorse your use of third party web pages nor provide any warranty whatsoever relating to third party web pages.

Any software coding and/or code lines/strings ("Code") included in this documentation are only examples and are not intended to be used in a productive system environment. The Code is only intended better explain and visualize the syntax and phrasing rules of certain coding. SAP does not warrant the correctness and completeness of the Code given herein, and SAP shall not be liable for errors or damages caused by the usage of the Code, except if such damages were caused by SAP intentionally or grossly negligent.