ABAP Code Sample to Display and Edit Data from Text File Using ALV

Applies To:

SAP / ABAP

Article Summary

This code sample brings forth the idea of displaying flat file data using ALV grid control. Also the same can be modified right from ALV Grid. Thus it makes viewing data in text files and editing the same using ALV.
Another feature is that it can also be used to create new flat files.

- Internal tables are created dynamically
- Text file data of any number of columns can be loaded
- Data can be created/modified/deleted right from ALV
- Automatically generated field Catalog
- Code can be used directly (needs no alteration)
- Container used is called 'Docking Container'

By: Immanuel Daniel, Wipro Technologies
Date: 10 Feb 2005

Code Sample

```
*&---------------------------------------------------------------------*
*& Report      : ZITXT_TBL                                             *
*& Description : To display&edit any flat file data Using ALV          *
*& Author      : Immanuel Daniel                                       *
*&---------------------------------------------------------------------*


REPORT  zitxt_tbl                 .
INCLUDE <icon>.
TABLES : sscrfields.

DATA : tab_dat TYPE REF TO data,
dock_cont TYPE REF TO cl_gui_docking_container,
dialog_cont TYPE REF TO cl_gui_dialogbox_container,
alv_grid TYPE REF TO cl_gui_alv_grid,
i_fcat TYPE lvc_t_fcat,
s_layo TYPE lvc_s_layo,
dynnr TYPE sy-dynnr,
repid TYPE sy-repid.


FIELD-SYMBOLS : <fs_tab> TYPE table.
```

```
*---------------------------------------------------------------------*
*       CLASS lcl_grid_event_receiver DEFINITION
*---------------------------------------------------------------------*

CLASS lcl_grid_event_receiver DEFINITION .

  PUBLIC SECTION.
    METHODS:
      toolbar             FOR EVENT toolbar
                          OF cl_gui_alv_grid
                          IMPORTING e_object
                                    e_interactive

    ,user_command         FOR EVENT user_command
                          OF cl_gui_alv_grid
                          IMPORTING e_ucomm

                          .

ENDCLASS.


*---------------------------------------------------------------------*
*Implementation of Grid event-handler class
*---------------------------------------------------------------------*
CLASS lcl_grid_event_receiver IMPLEMENTATION.
*---------------------------------------------------------------------
* Method for handling all creation/modification calls to the toolbar
*---------------------------------------------------------------------
  METHOD toolbar.

    DATA : ls_toolbar TYPE stb_button.
*---------------------------------------------------------------------
* Define Custom Button in the toolbar
*---------------------------------------------------------------------
    CLEAR ls_toolbar.
    MOVE 0 TO ls_toolbar-butn_type.
    MOVE 'EDIT' TO ls_toolbar-function.
    MOVE space TO ls_toolbar-disabled.
    MOVE 'EDIT' TO ls_toolbar-text.
    MOVE icon_change_text TO ls_toolbar-icon.
    MOVE 'Click2Edit' TO ls_toolbar-quickinfo.

    APPEND ls_toolbar TO e_object->mt_toolbar.

    CLEAR ls_toolbar.
    MOVE 0 TO ls_toolbar-butn_type.
    MOVE 'UPDA' TO ls_toolbar-function.
    MOVE space TO ls_toolbar-disabled.
    MOVE 'UPDATE' TO ls_toolbar-text.
    MOVE icon_system_save TO ls_toolbar-icon.
    MOVE 'Click2Update' TO ls_toolbar-quickinfo.

    APPEND ls_toolbar TO e_object->mt_toolbar.
```

```
    CLEAR ls_toolbar.
    MOVE 0 TO ls_toolbar-butn_type.
    MOVE 'EXIT' TO ls_toolbar-function.
    MOVE space TO ls_toolbar-disabled.
    MOVE 'EXIT' TO ls_toolbar-text.
    MOVE icon_system_end TO ls_toolbar-icon.
    MOVE 'Click2Exit' TO ls_toolbar-quickinfo.

    APPEND ls_toolbar TO e_object->mt_toolbar.

  ENDMETHOD.

*---------------------------------------------------------------------
*  Method to handle user commands from toolbar
*---------------------------------------------------------------------
  METHOD user_command.

    CASE e_ucomm .

      WHEN 'EDIT'.
        PERFORM set_input.

      WHEN 'UPDA'.
        PERFORM refresh_disp.
        PERFORM update_table.

      WHEN 'EXIT'.
        PERFORM free_objects.
        LEAVE PROGRAM.

    ENDCASE.

  ENDMETHOD.
ENDCLASS.

SELECTION-SCREEN COMMENT /15(40) comm.
SELECTION-SCREEN ULINE.
PARAMETERS : filename LIKE rlgrap-filename.
PARAMETERS : n_fields TYPE int1.


SELECTION-SCREEN : BEGIN OF BLOCK buttons WITH FRAME TITLE titl1,
BEGIN OF LINE,
PUSHBUTTON 7(20) new USER-COMMAND newtab,
PUSHBUTTON 30(7)  open USER-COMMAND open,
PUSHBUTTON 40(7) exit USER-COMMAND exit,

END OF LINE,
END OF BLOCK buttons.


DATA : grid_handler TYPE REF TO lcl_grid_event_receiver.

INITIALIZATION.
  dynnr = sy-dynnr.
  repid = sy-repid.
```

```
  open = 'OPEN'.
  exit = 'EXIT'.
  new = 'Create New File'.
  titl1 = 'Menu'.
  comm  ='DISPLAY & EDIT FLAT FILE DATA USING ALV'.

AT SELECTION-SCREEN ON VALUE-REQUEST FOR filename.

****Getting Filename***

  CALL FUNCTION 'WS_FILENAME_GET'
   EXPORTING
*   DEF_FILENAME          = ' '
    def_path              = 'C:'
    mask                  = ',*.*,*.*.'
    mode                  = 'o'
    title                 = 'Select File '
   IMPORTING
    filename              = filename
*   RC                    =
   EXCEPTIONS
    inv_winsys            = 1
    no_batch              = 2
    selection_cancel      = 3
    selection_error       = 4
    OTHERS                = 5
          .
  IF sy-subrc <> 0.
   EXIT.
  ENDIF.

AT SELECTION-SCREEN.


  CASE sscrfields.
   WHEN 'NEWTAB'.
    CLEAR i_fcat[].

    IF n_fields IS INITIAL.
     CALL FUNCTION 'POPUP_TO_INFORM'
    EXPORTING
        titel = '       Input Required       '
        txt1  = '        ****Enter Values****'
        txt2  = 'Field N_FIELDS is mandatory to create new file'

    . "Period

     EXIT.
    ENDIF.

    PERFORM build_fcat.

    PERFORM build_dyn_itab.

    PERFORM build_objects.
```

```
  PERFORM layo_build.


  CALL METHOD alv_grid->set_table_for_first_display
    EXPORTING
      is_layout                     = s_layo
    CHANGING
      it_outtab                     = <fs_tab>
      it_fieldcatalog               = i_fcat

          ."Period


  IF sy-subrc <> 0.
    EXIT.
  ENDIF.
  PERFORM set_input.

  CALL FUNCTION 'POPUP_TO_INFORM'
       EXPORTING
         titel = 'INFORMATION'
         txt1  = 'Click   "APPEND ROW"    icon '
         txt3  = 'of ALV Tool-Bar to enter data'
         txt2  = icon_create
       . "Period


 WHEN 'OPEN'.
   CLEAR i_fcat[].
   IF filename IS INITIAL OR n_fields IS INITIAL.
     CALL FUNCTION 'POPUP_TO_INFORM'
          EXPORTING
              titel = '       Input Required       '
              txt1  = '***All fields are mandatory***'
              txt2  = 'Enter Values'
          . "Period

     EXIT.
   ENDIF.


   PERFORM build_fcat.

   PERFORM build_dyn_itab.

   PERFORM build_objects.
   PERFORM layo_build.

   CALL FUNCTION 'WS_UPLOAD'
       EXPORTING
           filename = filename
           filetype = 'DAT'
       TABLES
           data_tab = <fs_tab>
       . "Period
   IF sy-subrc <> 0.
```

```
      EXIT.
    ENDIF.

    CALL METHOD alv_grid->set_table_for_first_display
      EXPORTING
   is_layout                     = s_layo
      CHANGING
        it_outtab                  = <fs_tab>
        it_fieldcatalog              = i_fcat

            ."Period

    IF sy-subrc <> 0.
      EXIT.
    ENDIF.


  WHEN 'EXIT'.

    PERFORM free_objects.
    LEAVE PROGRAM.

 ENDCASE.
*&---------------------------------------------------------------------*
*&      Form  build_fcat
*&---------------------------------------------------------------------*
*       Building Field Catalogue
*----------------------------------------------------------------------*

FORM build_fcat.

 DATA : wa_fcat LIKE LINE OF i_fcat.
 DO n_fields TIMES.

   wa_fcat-fieldname = sy-index.
   wa_fcat-coltext = sy-index.
   APPEND wa_fcat TO i_fcat.
 ENDDO.
ENDFORM.                    " build_fcat

*&---------------------------------------------------------------------*
*&      Form  layo_build
*&---------------------------------------------------------------------*
*       Constructing Layout
*----------------------------------------------------------------------*

FORM layo_build.

 s_layo-edit = 'X'.
 s_layo-grid_title = filename.


ENDFORM.                    " layo_build
```

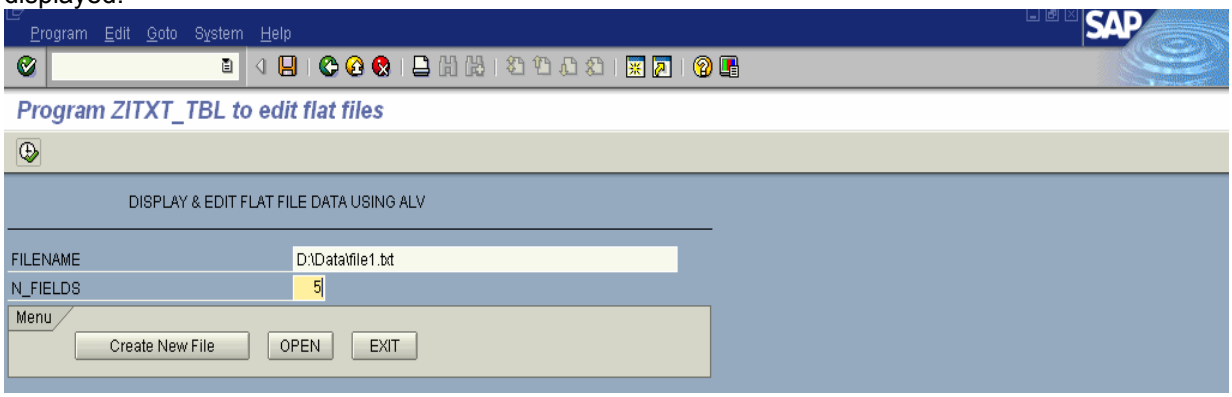
```
*&---------------------------------------------------------------------*
*&      Form  build_objects
*&---------------------------------------------------------------------*
*       Creating objects.....
*----------------------------------------------------------------------*

FORM build_objects.


  IF dock_cont IS INITIAL.

    CREATE OBJECT dock_cont
  EXPORTING
    repid                     = repid
    dynnr                      = dynnr
    side                      = dock_cont->dock_at_bottom
    extension                   = 200
  EXCEPTIONS
    cntl_error                = 1
    cntl_system_error           = 2
    create_error               = 3
    lifetime_error             = 4
    lifetime_dynpro_dynpro_link = 5
    others                    = 6
    ."Period

    IF sy-subrc <> 0.
      EXIT.
    ENDIF.

  ENDIF.

  IF alv_grid IS INITIAL.
    CREATE OBJECT alv_grid
    EXPORTING

      i_parent          = dock_cont
      ."Period

    IF sy-subrc <> 0.
      EXIT.
    ENDIF.
  ENDIF.

  CALL METHOD alv_grid->set_ready_for_input
   EXPORTING
     i_ready_for_input = 0.


  IF grid_handler IS INITIAL.
    CREATE OBJECT grid_handler.
    SET HANDLER:
       grid_handler->user_command FOR alv_grid,
      grid_handler->toolbar FOR alv_grid.

  ENDIF.
```

```
ENDFORM.                    " build_objects

*&---------------------------------------------------------------------*
*&      Form  build_dyn_itab
*&---------------------------------------------------------------------*
*       Building the internal Table
*----------------------------------------------------------------------*

FORM build_dyn_itab.

  CALL METHOD cl_alv_table_create=>create_dynamic_table
    EXPORTING

      it_fieldcatalog           = i_fcat
    IMPORTING
      ep_table                  = tab_dat
        ."Period


  ASSIGN tab_dat->* TO <fs_tab>       .

ENDFORM.                    " build_dyn_itab
*&---------------------------------------------------------------------*
*&      Form  refresh_disp
*&---------------------------------------------------------------------*
*       Refresh ALV GRID
*----------------------------------------------------------------------*

FORM refresh_disp.

  CALL METHOD alv_grid->refresh_table_display.
  IF sy-subrc <> 0.
    EXIT.
  ENDIF.


ENDFORM.                    " refresh_disp


**---------------------------------------------------------------------*
**       FORM update_table                                             *
**---------------------------------------------------------------------*
FORM update_table.

  PERFORM refresh_disp.

  DATA : c1 VALUE '.',
  filename1 LIKE rlgrap-filename.
  filename1 = filename.
  REPLACE c1 WITH '_modified.' INTO filename1.

  CALL FUNCTION 'DOWNLOAD'
      EXPORTING
          filename = filename1
          filetype = 'DAT'
```

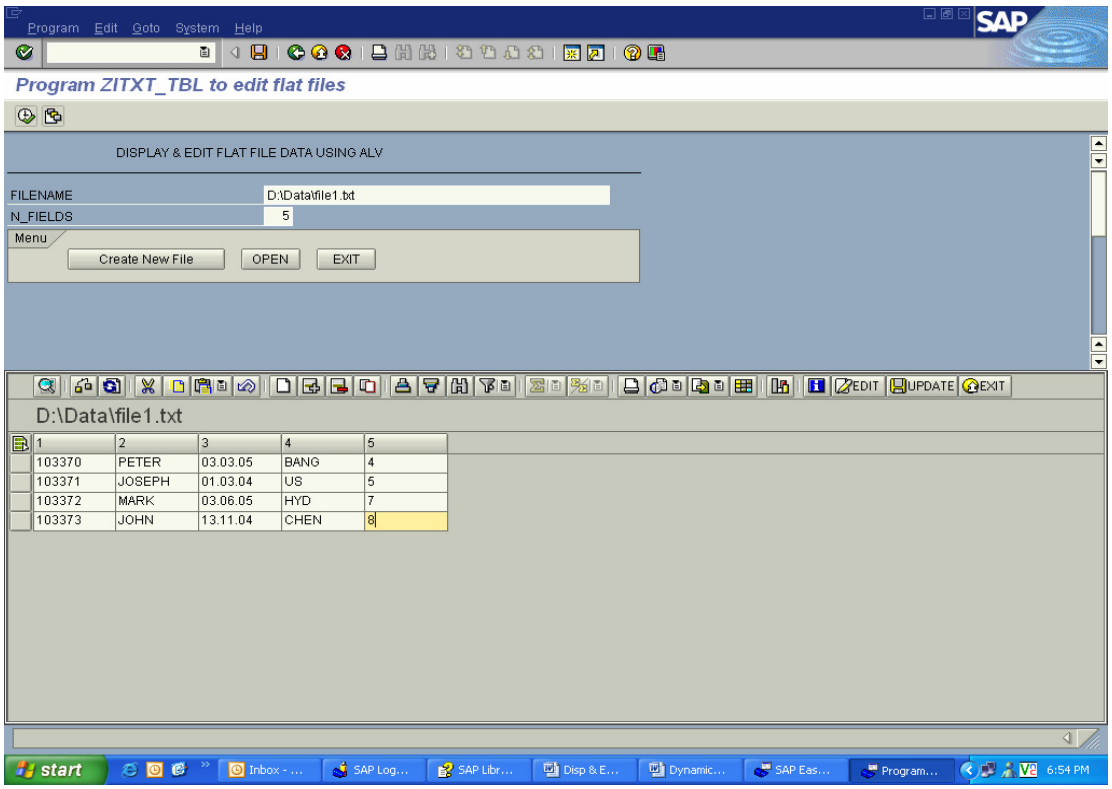
```
          mode     = ''
        TABLES
          data_tab = <fs_tab>.
  IF sy-subrc <> 0.
    EXIT.
  ENDIF.


  CALL METHOD alv_grid->set_ready_for_input
      EXPORTING
        i_ready_for_input = 0.
*
ENDFORM.


*---------------------------------------------------------------------*
*       FORM set_input                                                *
*---------------------------------------------------------------------*
FORM set_input.

  CALL METHOD alv_grid->set_ready_for_input
     EXPORTING
       i_ready_for_input = 1.

ENDFORM.


START-OF-SELECTION.
*No coding neede here..Selection screen alone will do...

END-OF-SELECTION.
*&---------------------------------------------------------------------*
*&      Form  free_objects
*&---------------------------------------------------------------------*

FORM free_objects.

  IF alv_grid IS INITIAL.
    EXIT.
  ELSE.
    CALL METHOD alv_grid->free.
    CALL METHOD dock_cont->free.
  ENDIF.
ENDFORM.                    " free_objects
```

Screen Shots

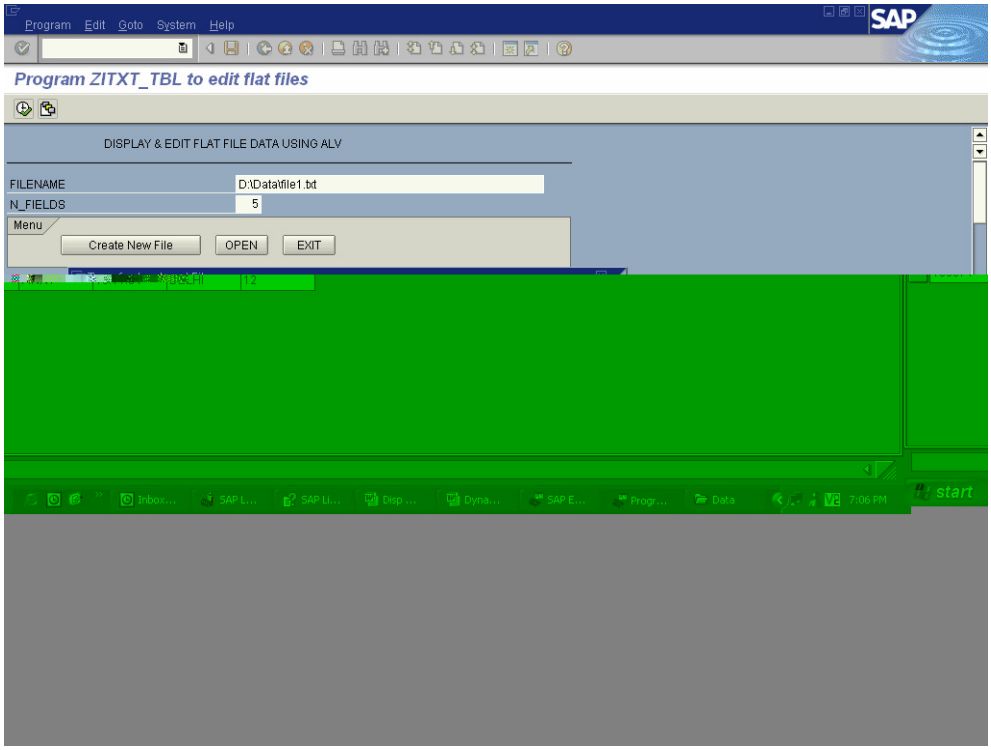
On the Selection Screen enter the text file's name and the number of fields that has to be displayed.



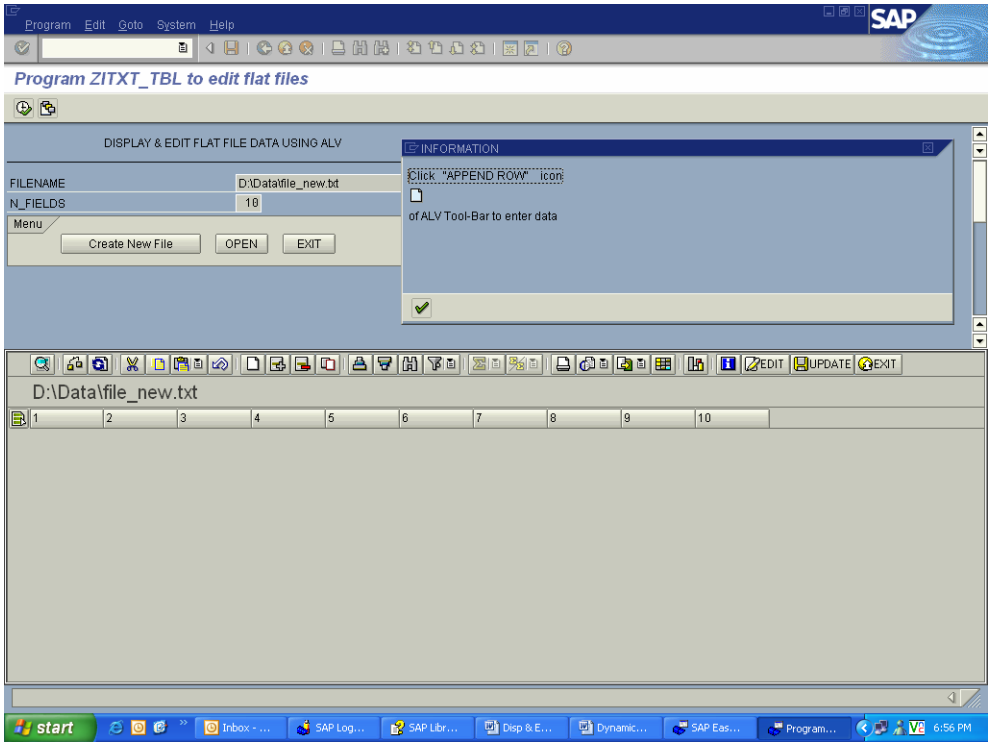
The flat file data is being displayed in ALV grid (docking Container)



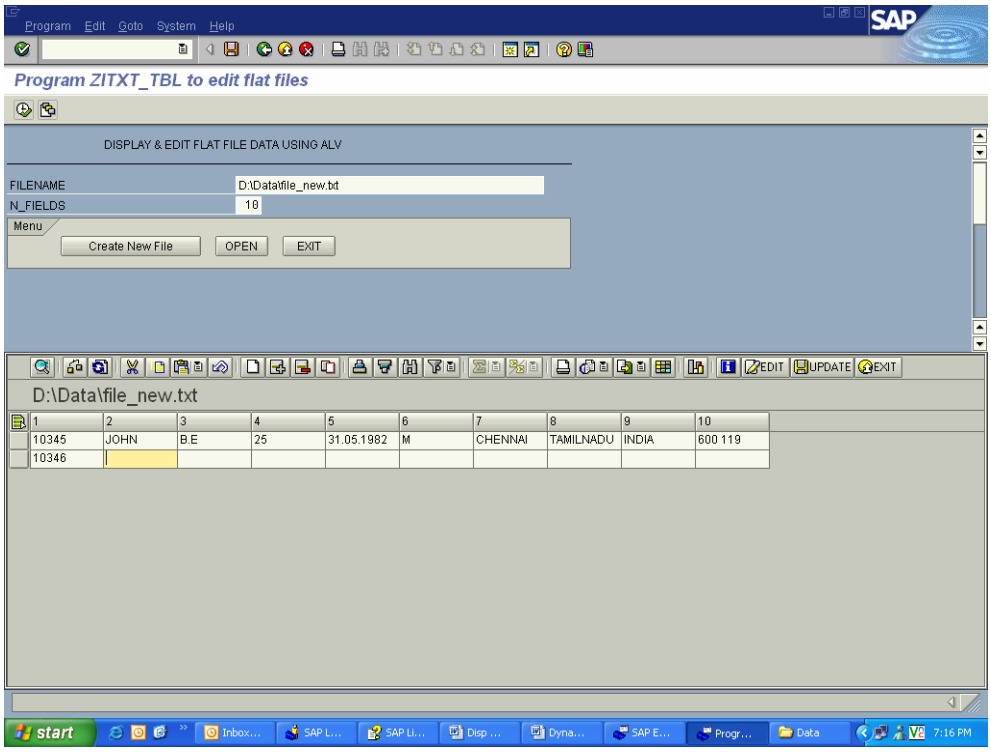
A new row has been appended to the file and is updated. It can be overwritten or be saved in a different file.



Click on ‘Create New File’ to create new text file. Enter the number of fields that the new table is should have. A grid with required number of fields is displayed. Click ‘Append Row‘ in the ALV toolbar to add rows.



Click ‘Append Row’ and add new records. Following is a screenshot of new table created with 10 fields.



Save the file by clicking ‘UPDATE’. The data is saved in a text file (in tab separated format). Flat file data can be created/modified from ALV grid.
This code needs no alteration and can be used as it is and no separate screen is required.

Disclaimer & Liability Notice

This document may discuss sample coding, which does not include official interfaces and therefore is not supported. Changes made based on this information are not supported and can be overwritten during an upgrade.

SAP will not be held liable for any damages caused by using or misusing of the code and methods suggested here, and anyone using these methods, is doing it under his/her own responsibility.

SAP offers no guarantees and assumes no responsibility or liability of any type with respect to the content of the technical article, including any liability resulting from incompatibility between the content of the technical article and the materials and services offered by SAP. You agree that you will not hold SAP responsible or liable with respect to the content of the Technical Article or seek to do so.