Example Implementation for the Sales Order Monitor BAdI

Table of Contents

Introduction

This document describes the BAdI enhancement that may be used to modify the selection screen of the Sales Order Monitor report (transaction `VA06`). The enhanced selection screen is displayed under tab *Enhancements*. By means of the enhancement described here, it is possible to add additional selection parameters into the selection screen of the Sales Order Monitor report. The following selection screen shows the additional selection parameters that can be added by BAdI implementation:

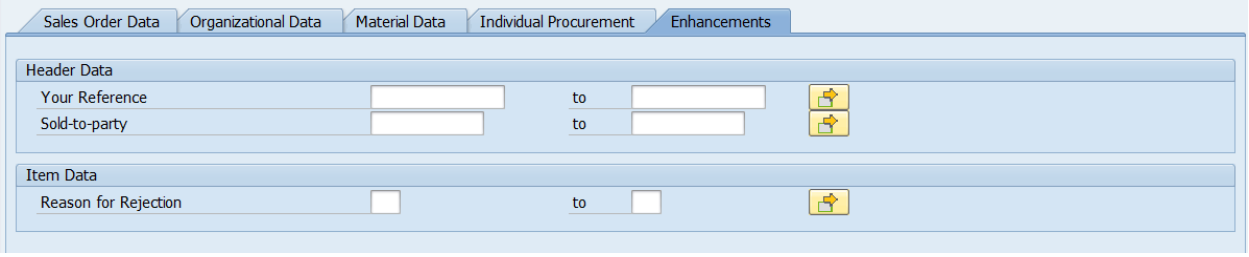

Figure 1: Enhanced Selection Screen

For clarity, sample code is provided to demonstrate many of the principals involved in successfully adding selection parameters. While you are free to use the sample code or any portion thereof in your own development, please note that no attempt has been made to optimize performance of the code or ensure that it will work in all scenarios. The sample code is not supported by SAP. It may be used at your own risk.

Business Add-In SD_OSO_MONITOR

This Business Add-In (BAdI) is used in the Sales (`SD-SLS`) component. You can use this BAdI to implement enhancements for the document selection in the Sales Order Monitor report. The enhanced selection parameter will appear in the result list.

The BAdI comprises an interface and a screen enhancement. They can be checked under *Technical Details*. Please see screenshot in Figure 2:

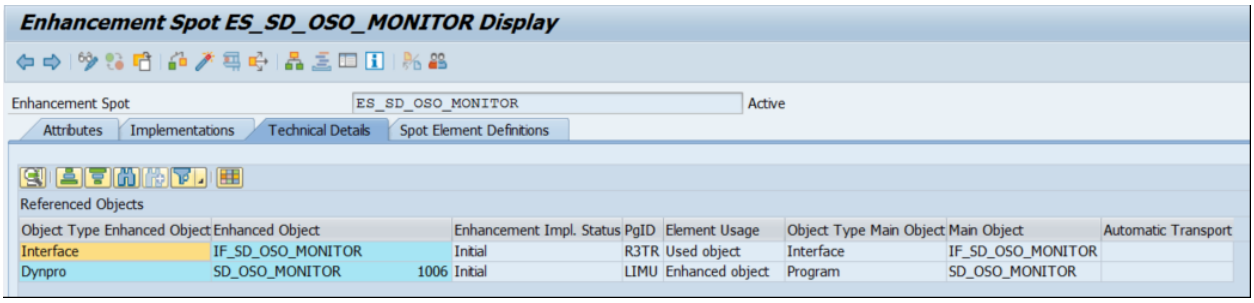

Figure 2: Technical Details

The essential elements of an implementation are sketched in figure 3:

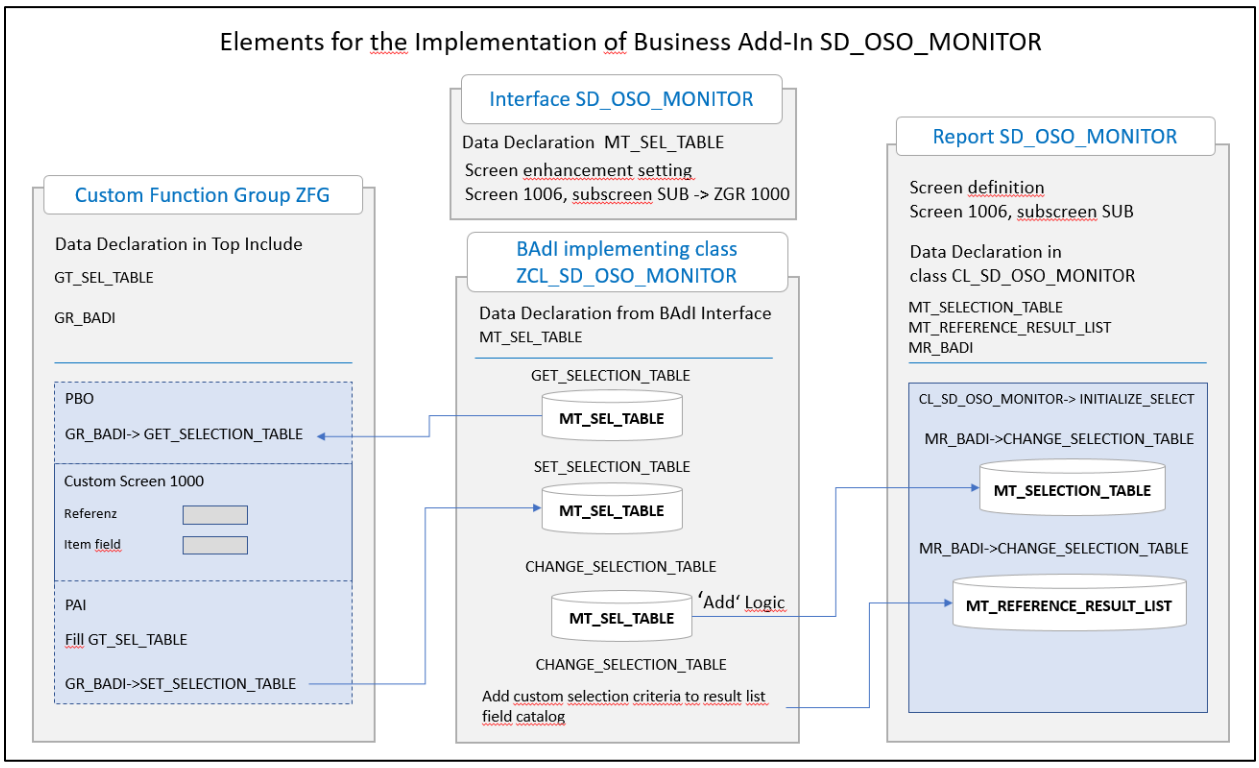

Figure 3: Elements for the Implementation of Business Add-In SD_OSO_MONITOR

The screen enhancement for screen 1006, subscreen SUB defined in program `SD_OSO_MONITOR`, is the central element of this BAdI. In your BAdI implementation, you specify your custom screen with the desired additional selection fields for subscreen SUB.

BAdI Implementation – Interface IF_SD_OSO_MONITOR

In this section, we will discuss how to implement the interface. In the BAdI implementing class, you must define a public instance attribute (e.g. called `MT_SEL_TABLE`) with `IF_SD_OSO_MONITOR~TTY_SEL_TABLE` as associated type. This table buffers the custom selection criteria. You must implement BAdI interface method `CHANGE_SELECTION_TABLE` to add the custom selection criteria to the global program table. Optionally, you can implement BAdI interface method `CHANGE_RESULT_LIST` to add the fields of the additional selection criteria to the result lists for header and item data. Please note that the sample code provided is not supported by SAP. It may be used at your own risk.

Interface	IF_SD_OSO_MONITOR
BAdI Implementation	ZSD_OSO_MONITOR
Description	Implementation: Open Sales Monitor BAdI
Implementing Class	ZCL_IM_SD_OSO_MONITOR
Method	IF_SD_OSO_MONITOR~CHANGE_SELECTION_TABLE
	IF_SD_OSO_MONITOR~SET_SELECTION_TABLE
	IF_SD_OSO_MONITOR~GET_SELECTION_TABLE
	IF_SD_OSO_MONITOR~CHANGE_RESULT_LIST
Attribute	MT_SEL_TABLE

Method IF_SD_OSO_MONITOR~CHANGE_SELECTION_TABLE

You can use this Business Add-In (BAdI) method to add your custom selection criteria to the table with the standard selection criteria. The custom selection criteria are stored in an instant attribute of your implementing class. This BAdI method is called in the SD_OSO_MONITOR program in method INITIALIZE_SELECT. The table that is used as basis for the selection statement for sales orders contains both the selection criteria entered on the standard screen and the selection criteria entered on your custom screen. In the sample code below, CT_SEL_TABLE is changing the table with selection criteria. It contains the standard selection criteria as import and all selection criteria as export. Typically, you loop at the instance attribute table with your custom selection criteria and then you insert them into table CT_SEL_TABLE according to the table sorting with key fields TABLE and FIELD.

```
FIELD-SYMBOLS: <fs_sel_table_impl> TYPE sel_table.

    LOOP AT mt_sel_table ASSIGNING <fs_sel_table_impl>.
      READ TABLE ct_sel_table WITH KEY
        table  =  <fs_sel_table_impl>-table
        field  =  <fs_sel_table_impl>-field
        BINARY SEARCH
        TRANSPORTING NO FIELDS.

      IF sy-subrc <> 0 AND ( <fs_sel_table_impl>-
low IS NOT INITIAL OR <fs_sel_table_impl>-high IS NOT INITIAL ).
        INSERT <fs_sel_table_impl> INTO TABLE ct_sel_table.
      ENDIF.

    ENDLOOP.
```

Method IF_SD_OSO_MONITOR~SET_SELECTION_TABLE

You can use this Business Add-In (BAdI) method to copy the imported custom selection criteria to the corresponding instance attribute of your implementing class (e.g. `MT_SEL_TABLE`). This BAdI method should be called in the PAI module of your custom screen. The sample code below copies the imported custom selection criteria to the instance attribute of your implementing class. Therefore, the instance attribute table will contain the selection criteria entered on the custom screen.

```
   me->mt_sel_table  =  it_sel_table.
```

Method IF_SD_OSO_MONITOR~GET_SELECTION_TABLE

You can use this Business Add-In (BAdI) method to export the custom selection criteria stored in the corresponding instance attribute of your implementing class to the global table of the function group that contains the custom screen with your desired additional selection criteria. This BAdI method should be called in the PBO module of your custom screen. The requirement is to define the custom enhancement screen in the custom function group. You also need to define the table for the custom selection criteria in the global data of the function group. The enhancement screen will be discussed next section. The sample code below will return changing table `CT_SEL_TABLE`, which contains the custom selection criteria.

```
    FIELD-SYMBOLS <fs_sel_table> TYPE sel_table.

    LOOP AT me->mt_sel_table ASSIGNING <fs_sel_table>.
      READ TABLE ct_sel_table
        WITH KEY
          table = <fs_sel_table>-table
          field = <fs_sel_table>-field
        BINARY SEARCH
        TRANSPORTING NO FIELDS.
      IF sy-subrc <> 0.
        INSERT <fs_sel_table> INTO TABLE ct_sel_table.
      ENDIF.
    ENDLOOP.
```

Method IF_SD_OSO_MONITOR~CHANGE_RESULT_LIST

This method is used to enhance result list parameters in the main program. In the sample code, we add the field VBAK-IHREZ and VBAK-KUNNR to the result list at header level. We also add the field VBAP-ABGRU to the result list at item level.

```
  types:
* Structured type for the reference result list
    BEGIN OF ty_result_list.
  TYPES level     TYPE char10.
  TYPES levelstep TYPE char10.
  TYPES tabname   TYPE tabname.
  TYPES fieldname TYPE fieldname.
  TYPES END OF ty_result_list .
  data: lty_result_list type ty_result_list.

  lty_result_list-level ='HEADER'.
  lty_result_list-levelstep = 'HD_STEP11'.
  lty_result_list-tabname = 'VBAK'.
  lty_result_list-fieldname = 'IHREZ'.

  insert lty_result_list into table ct_result_list.

  lty_result_list-level ='HEADER'.
  lty_result_list-levelstep = 'HD_STEP11'.
  lty_result_list-tabname = 'VBAK'.
  lty_result_list-fieldname = 'KUNNR'.

  insert lty_result_list into table ct_result_list.

  lty_result_list-level ='ITEM'.
  lty_result_list-levelstep = 'IT_STEP1'.
  lty_result_list-tabname = 'VBAP'.
  lty_result_list-fieldname = 'ABGRU'.

  insert lty_result_list into table ct_result_list.
```

BAdI Implementation – Screen Enhancement

BAdI Definition	SD_OSO_MONITOR
BAdI Implementation	ZSD_OSO_MONITOR
Tcode/program	SD_OSO_MONITOR
Screen	1006
Subscreen Area	SUB
Function Group	ZSL_FG_SCREEN_ENH
Program	SAPLZSL_FG_SCREEN_ENH
Subscreen	500

The screen enhancement requires you to create a custom function group (ZFG) and custom screen. The custom screen will appear as a selection screen in the *Enhancement* tab of transaction VA06. For example, the function group ZSL_FG_SCREEN_ENH is created and program SAPLZSL_FG_SCREEN_ENH is generated. You also need to create a subscreen under the custom function group. A subscreen 0500 is created in this example. After creation of ZFG and custom screen, please enter the corresponding values to screen enhancements.

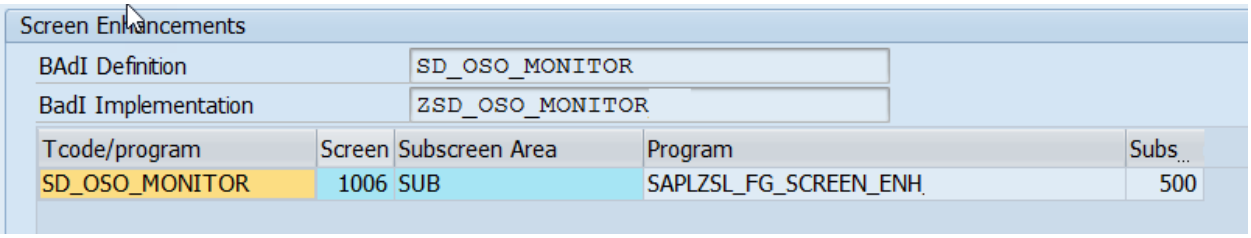

Figure 4: Screen Enhancements

Include LZSL_FG_SCREEN_ENHTOP

The global variables are defined in this Include.
We need to define a table (e.g. called GT_SEL_TABLE) that contains custom selection parameter. In addition, we need to define a global BAdI variable which keeps the reference to the BAdI implementing class. Please check the sample code to see what global variables are needed in this Include.

```
FUNCTION-POOL ZSL_FG_SCREEN_ENH.              "MESSAGE-ID ..

TYPES tty_sel_table TYPE SORTED TABLE OF sel_table WITH NON-
UNIQUE KEY table field.

CONSTANTS  gc_tabname_vbak    TYPE tabname VALUE 'VBAK'.
CONSTANTS  gc_tabname_vbap    TYPE tabname VALUE 'VBAP'.
CONSTANTS  gc_fieldname_ihrez TYPE fieldname VALUE 'IHREZ'.
CONSTANTS  gc_fieldname_abgru TYPE fieldname VALUE 'ABGRU'.
CONSTANTS  gc_fieldname_kunnr TYPE fieldname VALUE 'KUNNR'.

* Reference to BAdI implementation
DATA gr_badi            TYPE REF TO sd_oso_monitor.

* Local copy of selection table from BAdI implementation
DATA gt_sel_table       TYPE tty_sel_table.

DATA : gv_ihrez type ihrez,
       gv_kunnr type kunnr,
       gv_abgru type abgru_va,
       gv_prog type sy-repid value 'SAPLZSL_FG_SCREEN_ENH',
       gv_subdyn type sy-dynnr value '501'.
```

Include LZSL_FG_SCREEN_ENHSEL

The selection parameters on the enhancement screen are defined in this include. In the sample code below, the header fields `IHREZ` (*Your Reference*) and `KUNNR` (*Sold-to-party*) from the sales order header data `VBAK` are used as additional selection criterions. The item field `ABGRU` (*Rejection Reason*) from the sales order item data `VBAP` is used as additional selection criterion. In this include, you can also use selection criteria for database tables `VBAK`, `VBUK`, `VBAP`, `VBUP`, `VBEP`, `KNA1`, `EBAN` and `EKKO`. This is the one of limitation of BAdI and it only supports the table listed above.

```
SELECTION-SCREEN BEGIN OF SCREEN 501 AS SUBSCREEN.
SELECTION-SCREEN BEGIN OF BLOCK blck1 WITH FRAME TITLE text-601.
SELECT-OPTIONS : s_ihrez FOR gv_ihrez.
SELECT-OPTIONS : s_kunnr FOR gv_kunnr.
SELECTION-SCREEN END OF BLOCK blck1.

SELECTION-SCREEN BEGIN OF BLOCK blck2 WITH FRAME TITLE text-602.
SELECT-OPTIONS : s_abgru FOR gv_abgru.
SELECTION-SCREEN END OF BLOCK blck2.

SELECTION-SCREEN END OF SCREEN 501.
```

Custom Screen 0500

A custom screen needs to be created. The table listed below describes the flow logic of custom screen.

Dynpro Number	500

Process Before Output	Sample Code:
	`MODULE status_0500.`
	`  call subscreen SEL including gv_prog gv_subdyn.`

Process After Input	Sample Code:
	`call subscreen SEL.`
	`  MODULE user_command_0500.`

Module Status_0500

In the PBO part of the custom screen, you must call BAdI interface method `GET_SELECTION_TABLE`.

Sample Code:

```
  GET BADI gr_badi.

  CALL BADI gr_badi->get_selection_table
    changing
      ct_sel_table  =  gt_sel_table.
```

Module user_command_0500

In the PAI part of the custom screen, you must fill your new selection criteria in the global table `GT_SEL_TABLE` and hand over this table to the BAdI implementing class with BAdI interface method `SET_SELECTION_TABLE`.

Sample Code:

```
  DATA           ls_sel_table    TYPE sel_table.
  FIELD-SYMBOLS  <fs_sel_table>  TYPE sel_table.

* Take over input fields of enhancement in selection table from table workare
a
  ASSIGN ls_sel_table TO <fs_sel_table>.

* Customer Reference field (IHREZ)
    READ TABLE gt_sel_table WITH KEY table = gc_tabname_vbak
                                     field = gc_fieldname_ihrez low = s_ihrez
```

```
-low high = s_ihrez-high
                                        TRANSPORTING NO FIELDS.
    IF sy-subrc <> 0.
      <fs_sel_table>-table  = gc_tabname_vbak.
      <fs_sel_table>-field  = gc_fieldname_ihrez.
      IF s_ihrez IS NOT INITIAL.
        <fs_sel_table>-low    = s_ihrez-low.
        <fs_sel_table>-high   = s_ihrez-high.
        <fs_sel_table>-sign   = 'I'.
        <fs_sel_table>-option = 'EQ'.
      ELSE.
        CLEAR: <fs_sel_table>-sign, <fs_sel_table>-option, <fs_sel_table>-
low, <fs_sel_table>-high.
      ENDIF.

      INSERT ls_sel_table INTO TABLE gt_sel_table.
    ENDIF.

* sold-to-party (KUNNR)
    READ TABLE gt_sel_table WITH KEY table = gc_tabname_vbak
                                        field = gc_fieldname_kunnr low = s_kunnr
-low high = s_kunnr-high
                                        TRANSPORTING NO FIELDS.
    IF sy-subrc <> 0.
      <fs_sel_table>-table  = gc_tabname_vbak.
      <fs_sel_table>-field  = gc_fieldname_kunnr.
      IF s_kunnr IS NOT INITIAL.
        <fs_sel_table>-low    = s_kunnr-low.
        <fs_sel_table>-high   = s_kunnr-high.
        <fs_sel_table>-sign   = 'I'.
        <fs_sel_table>-option = 'EQ'.
      ELSE.
        CLEAR: <fs_sel_table>-sign, <fs_sel_table>-option, <fs_sel_table>-
low, <fs_sel_table>-high.
      ENDIF.

      INSERT ls_sel_table INTO TABLE gt_sel_table.
    ENDIF.

* Reason for Rejection (ABGRU)
    READ TABLE gt_sel_table WITH KEY table = gc_tabname_vbap
                                        field = gc_fieldname_abgru low = s_abgru
-low high = s_abgru-high
                                        TRANSPORTING NO FIELDS.
    IF sy-subrc <> 0.
      <fs_sel_table>-table  = gc_tabname_vbap.
      <fs_sel_table>-field  = gc_fieldname_abgru.
      IF s_abgru IS NOT INITIAL.
        <fs_sel_table>-low    = s_abgru-low.
        <fs_sel_table>-high   = s_abgru-high.
        <fs_sel_table>-sign   = 'I'.
        <fs_sel_table>-option = 'EQ'.
      ELSE.
```

```
        CLEAR: <fs_sel_table>-sign, <fs_sel_table>-option, <fs_sel_table>-
low, <fs_sel_table>-high.
      ENDIF.

      INSERT ls_sel_table INTO TABLE gt_sel_table.
    ENDIF.

* Write back to BAdI implementing class
  CALL BADI gr_badi->set_selection_table
    EXPORTING
      it_sel_table  =   gt_sel_table.
```

Conclusion

To conclude, the BAdI consists of an implementing class and screen enhancement. Users can add their own defined fields as additional selection parameters through BAdI implementation. The new selection parameter can be showed in a result list as well.