Need to Control Sequence of Calls of Business Add In (BAdI) Implementations…?

Applies to:

Applications on NetWeaver 2004s.

Summary

You want to control the sequence of BAdI Implementation calls? The kernel based BAdIs introduced with NetWeaver 2004s allows you to do so for any multiple use BAdI...

Authors: Michael Acker and Christian Hissler

Company: SAP AG

Created on: 27 June 2007

Authors Bio

Michael Acker works as development architect of the ABAP Workbench, Christian Hissler in the same role in Retail development. Both are SAP employees.

Table of Contents

A Typical Use Case

SAP delivers an implementation of a multiple use BAdI to round schedule quantity in a Purchase Order. You want to adjust the rounded values for certain scenarios. You decide to create an own implementation of this 'Rounding' BAdI. Now you need to ensure that standard implementation is always called before yours is processed so that you can see and modify the rounded values.

How it Works

Compiler usually lists the implementations of a BAdI in an arbitrary sequence. You can modify this arbitrary sequence so that implementations are called in a way you define it. All you need to do is to implement a Sorter BAdI (BADI_SORTER) and define a sub screen in an own function group. The sub screen allows you to enter data used for sorting.

Attention

The BADI_SORTER to determine sequence of implementation calls is a Single Use Filter dependent BAdI, i.e. for one single filter value only one implementation is allowed. A situation results in runtime errors where in a customer environment a sorting mechanism is implemented and SAP delivers later the same.

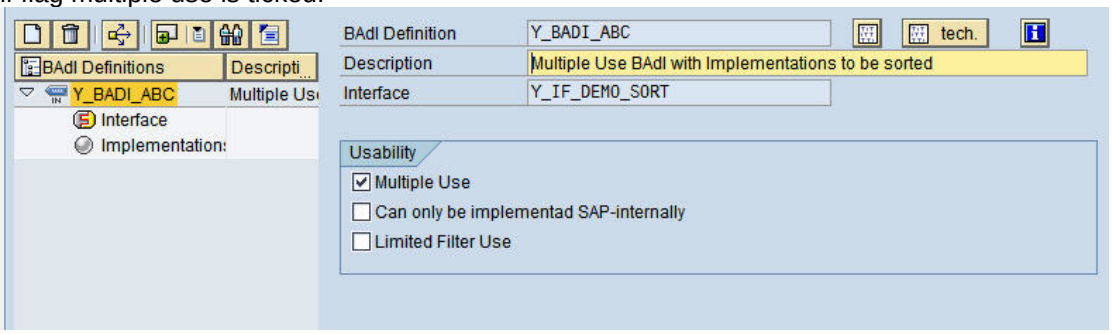
Therefore an SAP customer should contact support before implementing BADI_SORTER for BAdIs also used by SAP.

Details

BAdI Implementations for Sorting

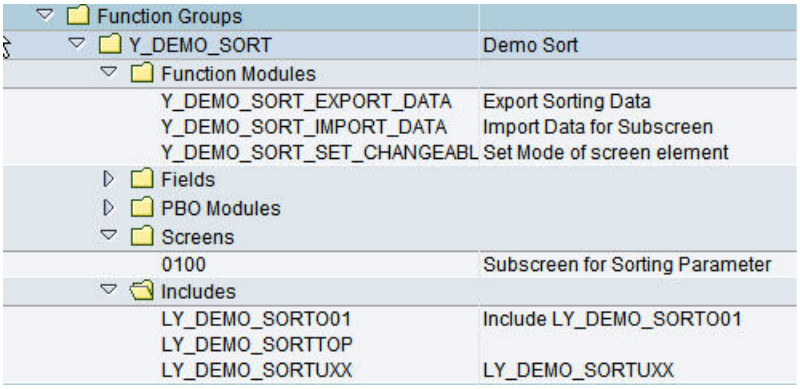
The feature is only available for multiple use kernel based BAdIs. Kernel Based BAdIs are assigned to an enhancement spot.

Let's say you want to sort implementations for a BAdI ABC. Call transactions SE18, enter ABC and check if flag multiple use is ticked.



Function Group and Sub screen

You define a sub screen SUB_ABC in an own function group. You can define any kind of input field at this sub screen. When you later on create an implementation for BAdI ABC sub screen SUB_ABC is visible and allows you to define specific data for your implementation. This data can be used when sorting your implementations. If you do not need to specify further data to sort implementations (implementation ID is always available) you can just define an empty sub screen. For data transfer from and to the BAdI implementation you create 2 Function Modules.

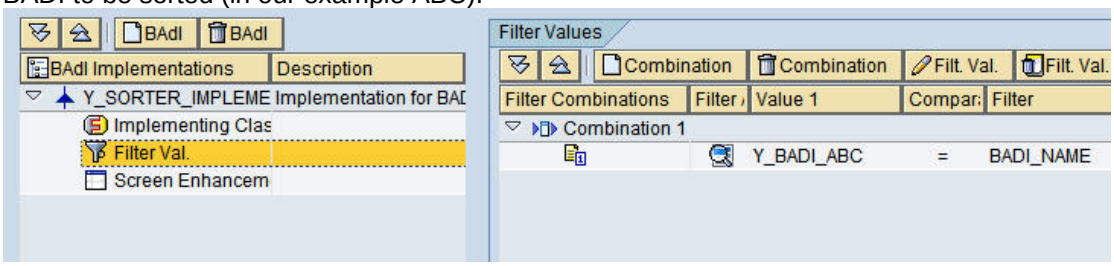


BADI_SORTER Implementation

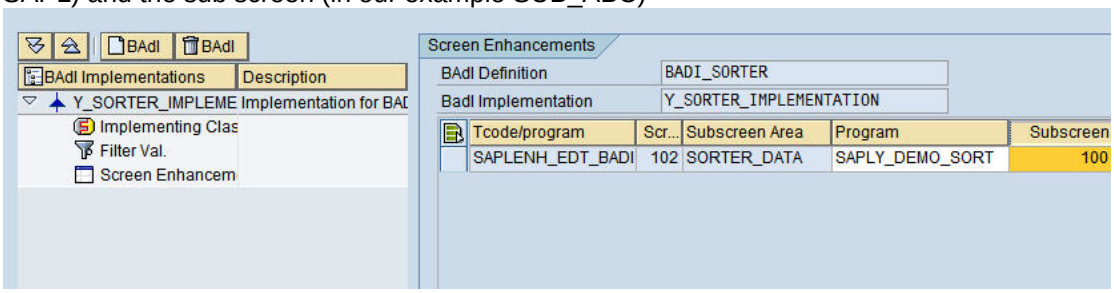
Go to transaction SE18, enter BADI_SORTER and display. You will find definition of BADI_SORTER. It is a filter dependent BAdI allowing screen enhancements.

By right mouse click you can implement the BADI_SORTER.

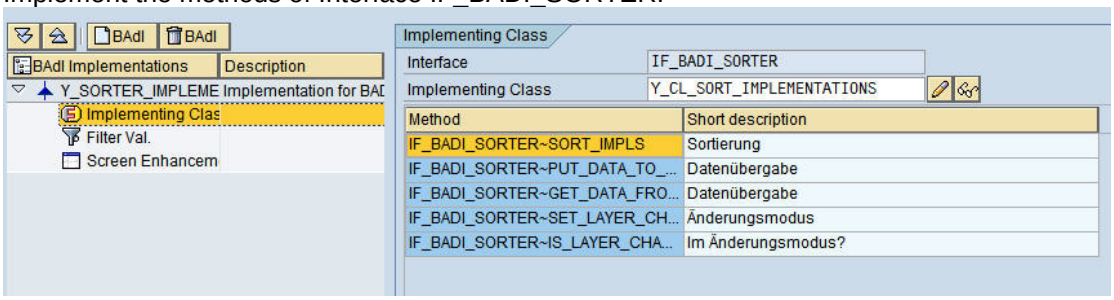
- Enter a filter value for your BADI_SORTER implementation. Filter value must be the name of the BADI to be sorted (in our example ABC).



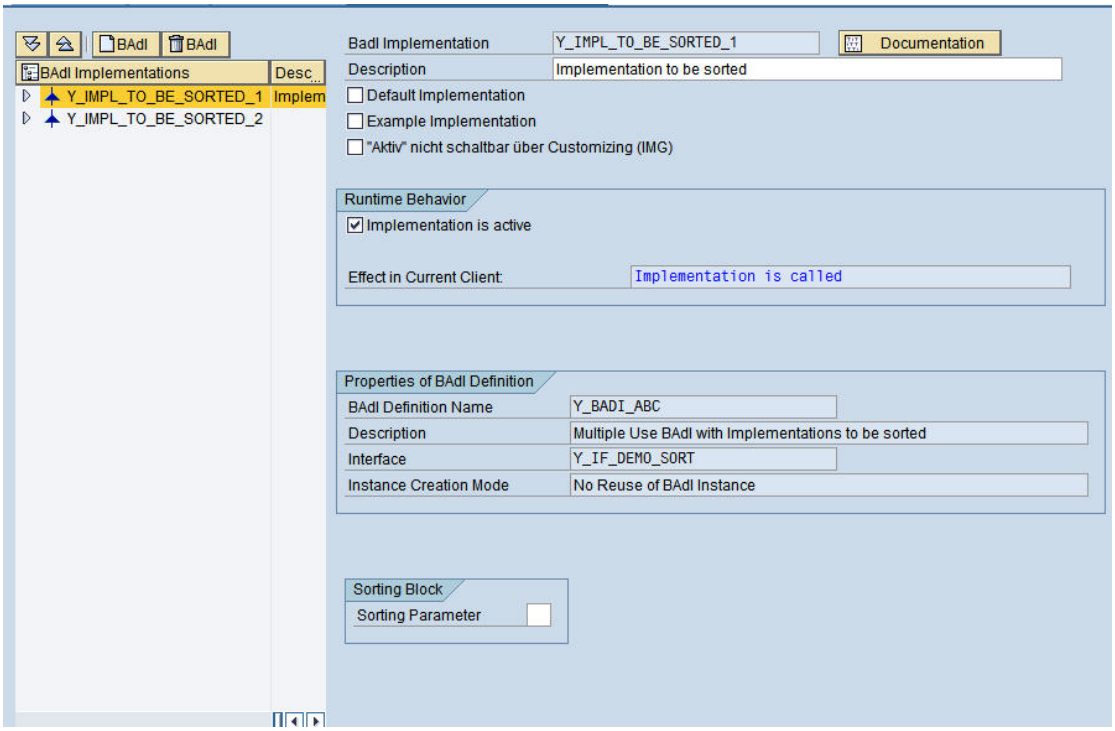
- Enter a screen enhancement. Please enter the program name (function group name + prefix SAPL) and the sub screen (in our example SUB_ABC)



- Implement the methods of Interface IF_BADI_SORTER.



After Activation of BADI_SORTER implementation you see for any implementation of BADI ABC a screen like this:



Methods of IF_BADI_SORTER

BADI_SORTER provides the following methods

- SORT_IMPLS
  SORT_IMPLS is the central method to sort the implementations. The CHANGING Parameter IMPLS_TO_SORT provides you with the data from your own defined sub screen and the ID of the corresponding implementations. Implementations are called in the sequence they appear in table IMPLS_TO_SORT (first entry is called first, second is called second…)

Method IF_BADI_SORTER~SORT_IMPLS Active

```
10  * get implementations
11    LOOP AT impls_to_sort INTO ls_sort.
12      ls_sequence-index = sy-tabix.
13      IF ls_sort-sorter_data IS NOT INITIAL.
14        TRY.
15            IMPORT sorter_data = ls_sequence-sorter_data
16             FROM DATA BUFFER ls_sort-sorter_data.
17          CATCH cx_sy_import_format_error.
18        ENDTRY.
19      ENDIF.
20      APPEND ls_sequence TO lt_sequence.
21    ENDLOOP.
22  * sort sorter table
23    SORT lt_sequence BY sorter_data.
24  * resort impls_to_sort
25    LOOP AT lt_sequence INTO ls_sequence.
26      CLEAR ls_sort.
27      READ TABLE impls_to_sort INTO ls_sort
28      INDEX ls_sequence-index.
29      APPEND ls_sort TO lt_sort.
30    ENDLOOP.
31    CLEAR impls_to_sort.
32    impls_to_sort = lt_sort.
33  ENDMETHOD.
```

- PUT_DATA_TO_SCREEN
  Data from your sub screen are stored by the implementations to be sorted. This method allows you to transfer this data to your own function group with sub screen SUB_ABC. You should create a function module to transfer the data.

Method IF_BADI_SORTER~PUT_DATA_TO_SCREEN Active

```
1   METHOD if_badi_sorter~put_data_to_screen.
2     DATA: l_sorter_data TYPE char2.
3
4     IF data IS NOT INITIAL.
5       TRY.
6           IMPORT sorter_data = l_sorter_data
7           FROM DATA BUFFER data.
8         CATCH cx_sy_import_format_error.
9       ENDTRY.
10    ENDIF.
11
12    CALL FUNCTION 'Y_DEMO_SORT_IMPORT_DATA'
13      EXPORTING
14        im_data = l_sorter_data.
15
16
17  ENDMETHOD.
```

- GET_DATA_FROM_SCREEN
  You transfer the data from your screen to the implementation to be sorted. To do so you call

usually a function module in your own function group.

Method | IF_BADI_SORTER~GET_DATA_FROM_SCREEN | Active

```
1  METHOD if_badi_sorter~get_data_from_screen.
2    DATA: l_sorter_data TYPE char2.
3    CALL FUNCTION 'Y_DEMO_SORT_EXPORT_DATA'
4      IMPORTING
5        ex_data = l_sorter_data.
6
7
8    EXPORT sorter_data = l_sorter_data TO DATA BUFFER data.
9  ENDMETHOD.
```

- SET_LAYER_CHANGEABLE
  This method provides the information if you call the BAdI implementation in change or display mode. This information needs to be available also in your own function group to switch for your sub screen SUB_ABC between display and change mode.

Method | IF_BADI_SORTER~SET_LAYER_CHANGEABLE

```
1  METHOD if_badi_sorter~set_layer_changeable.
2    CALL FUNCTION 'Y_DEMO_SORT_SET_CHANGEABLE'
3      EXPORTING
4        im_changeable = changeable.
5
6
7  ENDMETHOD.
```

IS_LAYER_CHANGEABLE
This method is not required.

Copyright