

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.....	3
How it Works	3
Attention	3
Details	4
BAdI Implementations for Sorting.....	4
Function Group and Sub screen	4
BADI_SORTER Implementation	4
Methods of IF_BADI_SORTER	6
Copyright.....	8

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.

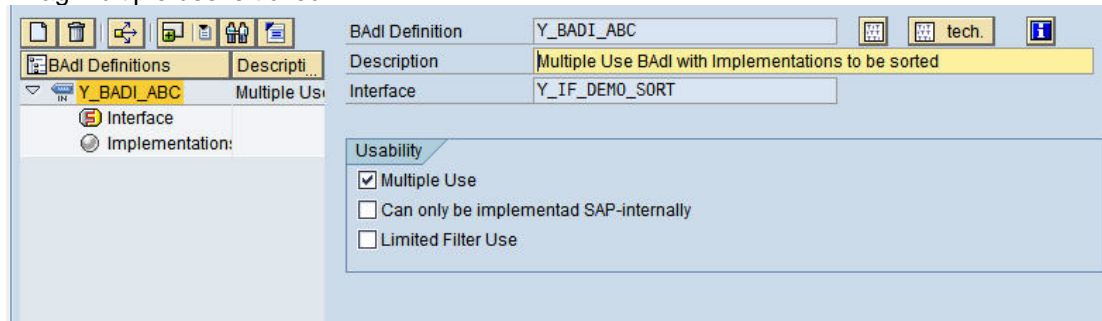
Therefore an SAP customer should contact support before implementing BADI_SORTER for BAdIs also used by SAP.

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.

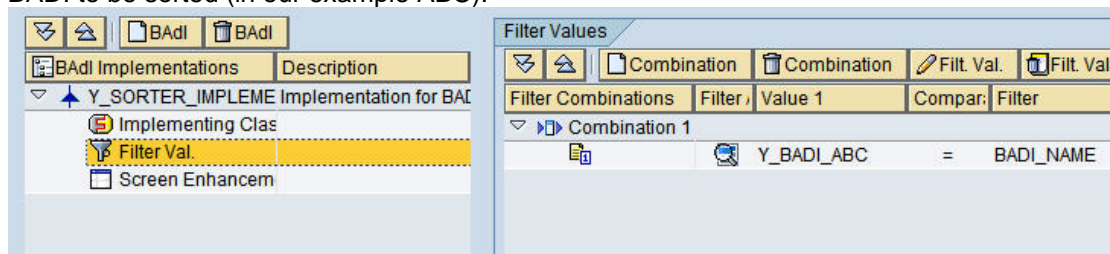
Function Groups	
Y_DEMO_SORT	Demo Sort
Function Modules	
Y_DEMO_SORT_EXPORT_DATA	Export Sorting Data
Y_DEMO_SORT_IMPORT_DATA	Import Data for Subscreen
Y_DEMO_SORT_SET_CHANGEABL	Set Mode of screen element
Fields	
PBO Modules	
Screens	
0100	Subscreen for Sorting Parameter
Includes	
LY_DEMO_SORT001	Include LY_DEMO_SORT001
LY_DEMO_SORTTOP	
LY_DEMO_SORTUXX	LY_DEMO_SORTUXX

BADI_SORTER Implementation

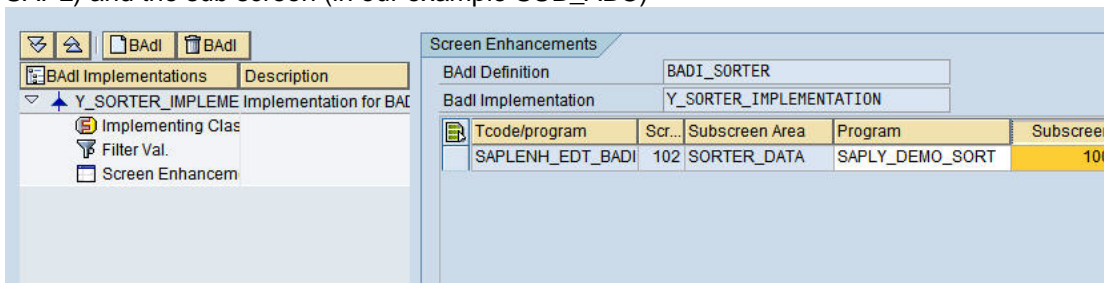
Go to transaction SE18, enter BADI_SORTER and display. You will find definition of BADI_SORTER. It is a filter dependent BAdI allowing screen enhancements.

By right mouse click you can implement the BADI_SORTER.

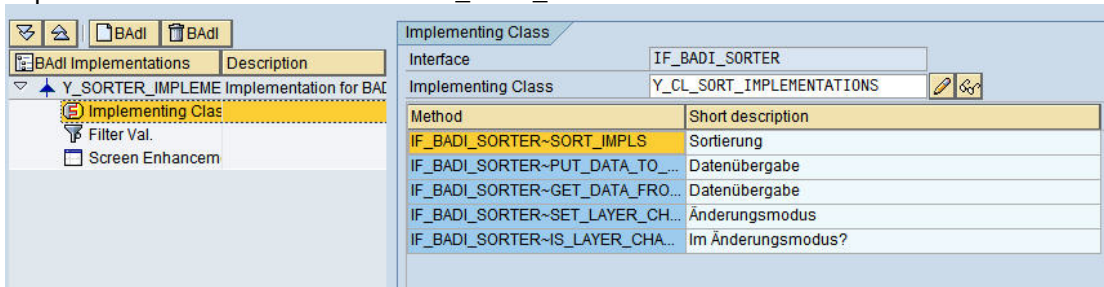
- Enter a filter value for your BADI_SORTER implementation. Filter value must be the name of the BAdI to be sorted (in our example ABC).



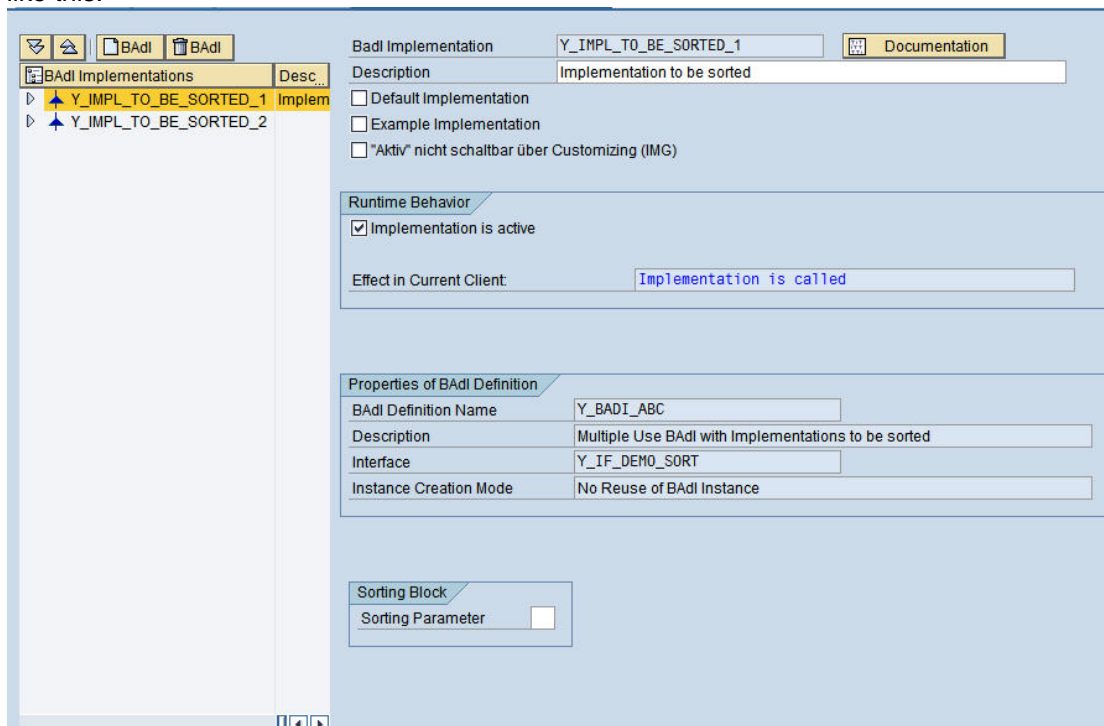
- Enter a screen enhancement. Please enter the program name (function group name + prefix SAPL) and the sub screen (in our example SUB_ABC)



- 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	<i>* get implementations</i>	
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	<i>* sort sorter table</i>	
23	SORT lt_sequence BY sorter_data.	
24	<i>* resort impls_to_sort</i>	
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

© Copyright 2007 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.