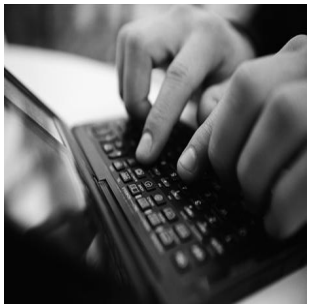
SAP BusinessObjects EPM
How-To Guide



How To... Use the Write Back Pre-Process BAdI

Applicable Releases:

SAP BusinessObjects Planning and Consolidation 7.0 , version for SAP NetWeaver, SP04 and higher.

SAP BusinessObjects Planning and Consolidation 7.5 , version for SAP NetWeaver, SP00 and higher.

Version 1.0

May 2010

© Copyright 2010 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.

SAP NetWeaver “How-to” Guides are intended to simplify the product implementation. While specific product features and procedures typically are explained in a practical business context, it is not implied that those features and procedures are the only approach in solving a specific business problem using SAP NetWeaver. Should you wish to receive additional information, clarification or support, please refer to SAP Consulting.

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.

Disclaimer

Some components of this product are based on Java™. Any code change in these components may cause unpredictable and severe malfunctions and is therefore expressively prohibited, as is any decompilation of these components.

Any Java™ Source Code delivered with this product is only to be used by SAP’s Support Services and may not be modified or altered in any way.

Document History

Document Version	Description
1.00	First official release of this guide

Typographic Conventions

Type Style	Description
Example Text	Words or characters quoted from the screen. These include field names, screen titles, pushbuttons labels, menu names, menu paths, and menu options. Cross-references to other documentation
Example text	Emphasized words or phrases in body text, graphic titles, and table titles
`Example text`	File and directory names and their paths, messages, names of variables and parameters, source text, and names of installation, upgrade and database tools.
`Example text`	User entry texts. These are words or characters that you enter in the system exactly as they appear in the documentation.
`<Example text>`	Variable user entry. Angle brackets indicate that you replace these words and characters with appropriate entries to make entries in the system.
`EXAMPLE TEXT`	Keys on the keyboard, for example, `F2` or `ENTER`.

Icons

Icon	Description
	Caution
	Note or Important
	Example
	Recommendation or Tip

Table of Contents

1. Business Scenario

Note: This guide is intended to demonstrate how to use the Write Back BAdI to implement custom code to fill gap requirements. In this guide, we use the "Disaggregation" business scenario as an example only. The code associated with this guide is not supported in any way by SAP, and is not guaranteed to work in all cases. Remember, this guide is to demonstrate the use of the Write Back BAdI only.

Currently, BPC only supports planning or writing data to base members or leaves of a hierarchy. Often times, the business user wants to plan at a higher level than just the base member level, especially when it comes to "what-if" analysis. This paper covers the case where a user wants to plan data at the parent level and distribute the values evenly to the underlying base members. This guide will demonstrate how you can use the Write Back BAdI to accomplish this task.

In this example, the end user would like to write enter data at a parent level Entity dimension, using a BPC Input Schedule, and evenly disaggregate or distribute that number across all of the children (base members) of that parent Entity dimension. In this guide's example, we are disaggregating evenly, however, you could perform other types of distribution, such as; based on a percentage value, or distribution based on previous year's data, for example.

The sample BAdI Implementation "UJR_BADI_SAMPLE_DISAGGREGATE" is shipped as an example implementation for this disaggregation scenario, but this guide will demonstrate how to implement the functionality from scratch.

2. Background Information

SAP Business Add-Ins (BAdIs) are one of the most important technologies used to adapt SAP software to specific requirements. As of Release 7.0 of the SAP NetWeaver Application Server ABAP, BAdIs are part of the Enhancement Framework, where they represent explicit enhancement options. BAdIs are the basis for *Object Plug-Ins* that can enhance the functions in ABAP programs without having to make core software modifications. As such, BAdI calls can be integrated into customer applications (like BusinessObjects Planning and Consolidation, version for SAP NetWeaver) to allow enhanced customization of standard application functionality.

This How-To Guide (HTG) describes the procedure for implementing a BAdI that allows the end user to write data to a parent level from within the BPC for Excel Front End Client. The Step By Step section will outline the steps needed to create the BAdI itself as well as the configuration required within BPC to actually execute the BAdI. The Appendix section contains the example ABAP code that goes along with this guide's Business Scenario. This code is only meant as an example and while it will perform the actions described in this guide it may not match the exact needs of your own particular Business Scenario – but it is a good starting point for the creation of your own BAdI Implementation.

3. Prerequisites

Required/recommended expertise or prior knowledge

- SAP BusinessObjects Planning and Consolidation 7.0, version for SAP NetWeaver, SP04 and higher
- SAP BusinessObjects Planning and Consolidation 7.5, version for SAP NetWeaver, SP00 and higher
- ABAP programming skills
- Access to SAP NetWeaver transaction codes: SE20, SE18, SE19, SE38, SE80, SE24, STMS

Additional Documentation

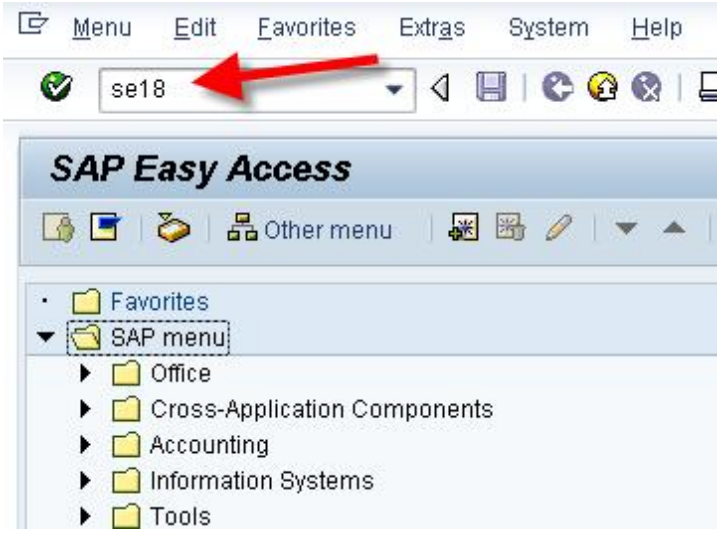
- RKT Online Knowledge Product
  - http://service.sap.com/rkt On the left hand side, navigate to SAP Ramp-Up Knowledge Transfer -> SAP BusinessObjects EPM Solutions -> SAP BO PC 7.5, version for SAP NetWeaver
- Other EPM How-To Guides
  - http://wiki.sdn.sap.com/wiki/display/BPX/Enterprise+Performance+Management+%28EPM%29+How-to+Guides
- SAP Help Library – Business Add Ins
  - http://help.sap.com/saphelp_nw70/helpdata/en/8f/f2e540f8648431e10000000a1550b0/frameset.htm

4. Step-by-Step Procedure

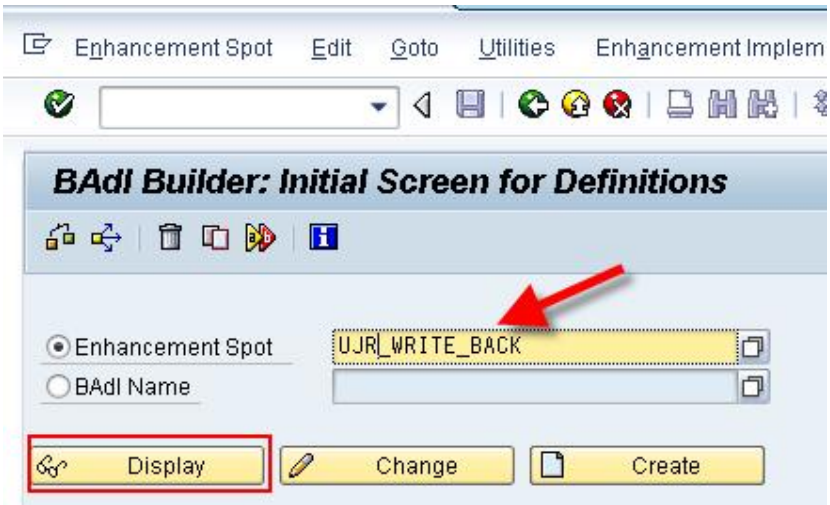
This How-To guide contains all the steps required to create a BAdI Implementation, specifically for the Write Back Pre-Process BAdI definition. A BAdI implementation is the term used in the Enhancement Framework for an enhancement implementation element. A BAdI implementation consists of a *BAdI implementation class* that implements the BAdI interface. The BAdI implementation also contains a filter condition which is specified in the BAdI definition. This filter condition can be used to execute the BAdI implementation at runtime.

4.1 Create a BAdI Implementation

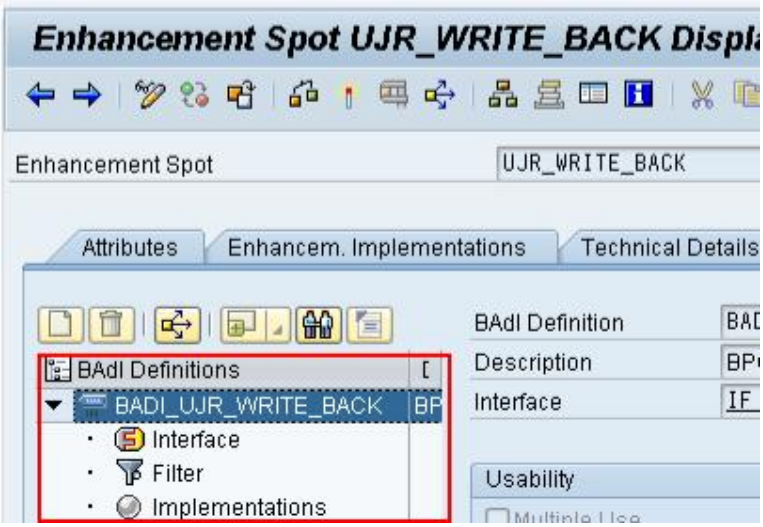
1. Log on to the SAP NetWeaver system via SAPgui. Enter transaction SE18 and press “Enter“.



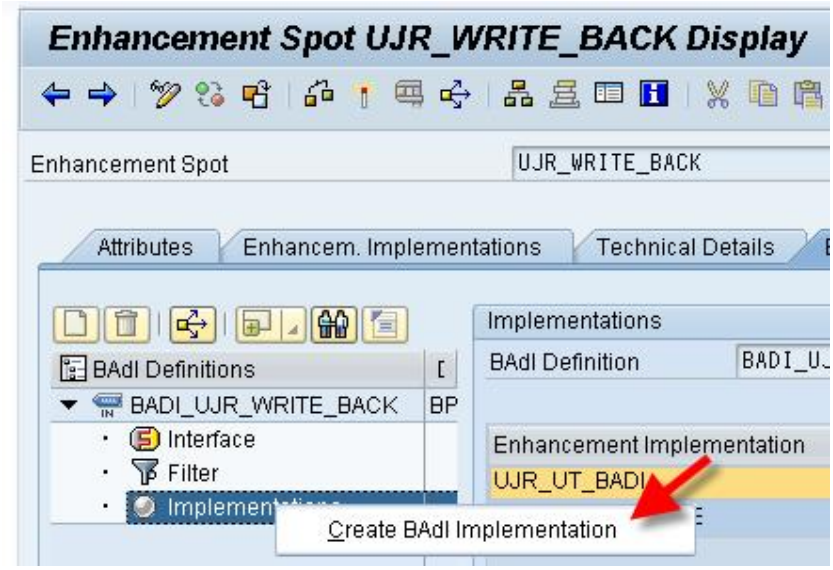
2. In the initial screen, enter the name of the corresponding enhancement spot. Enter UJR_WRITE_BACK, and click “Display“.



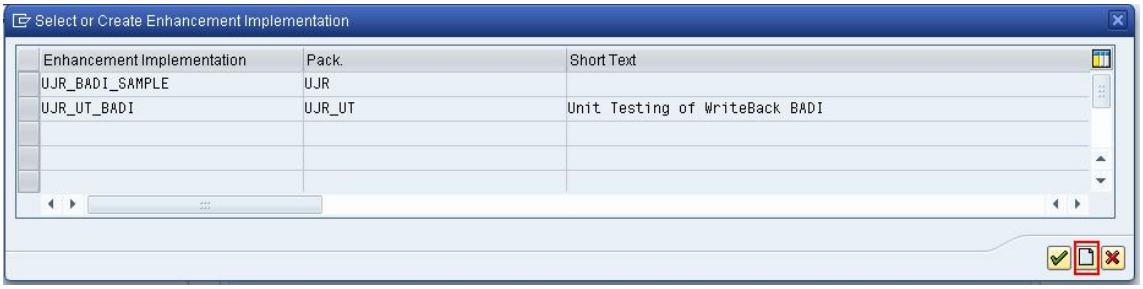
3. On the left side of the screen, expand the BAdI definition tree by clicking on the icon. You should then see the following nodes.
   - Interface
   - Filter
   - Implementations



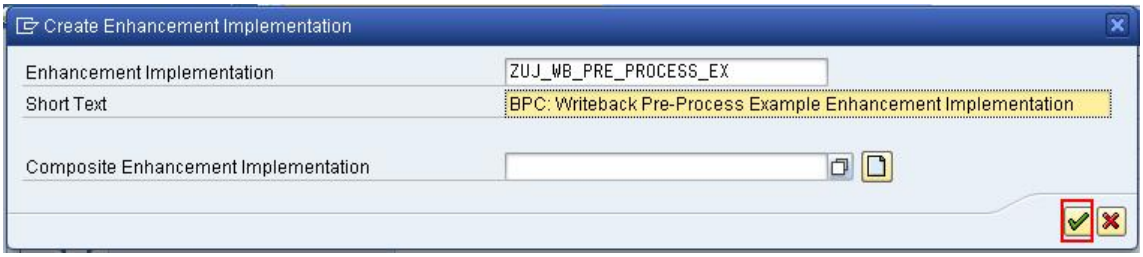
4. Right-click on the “Implementations” node, and choose “Create BAdI Implementation”.



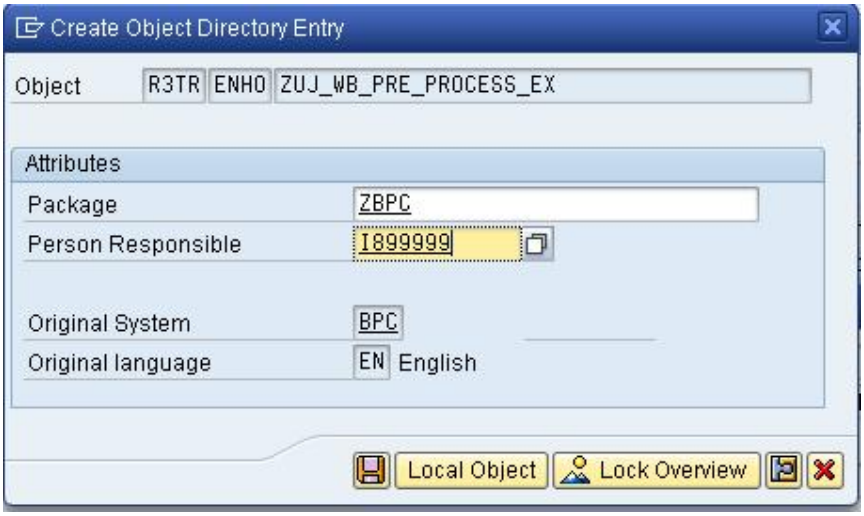
5. In some cases, a developer may have already created an enhancement implementation for this enhancement spot for a different BAdI definition. If an enhancement implementation already exists, a dialog listing all implementations will be displayed. Click the “Create” button in the lower right hand corner. If this dialog is not displayed, continue to step 6.



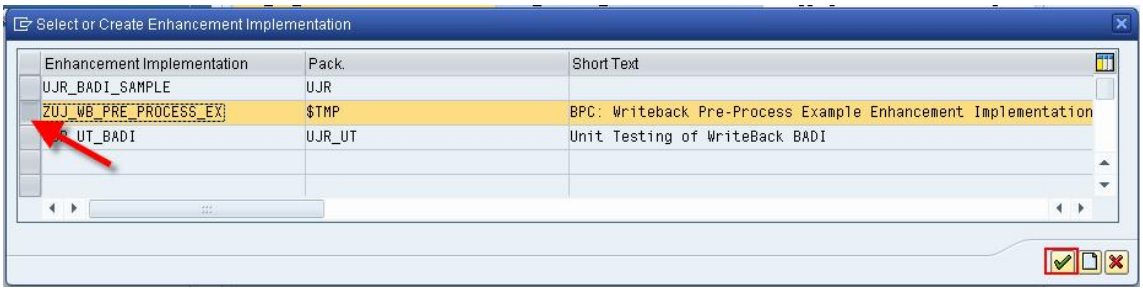
6. In this dialog, enter the name of the enhancement implementation and the short description. Name it as ZUJ_WB_PRE_PROCESS_EX. Then click the green check to continue.



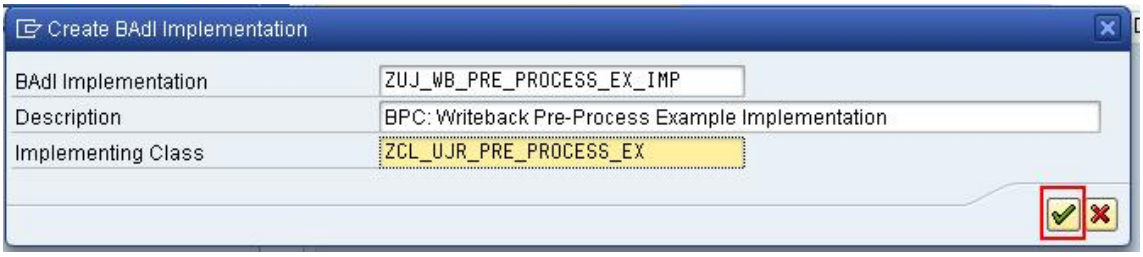
7. Enter the name of a package name for transporting this BAdI to another system in your landscape, or click “Local Object” if you do not plan to transport this BAdI.



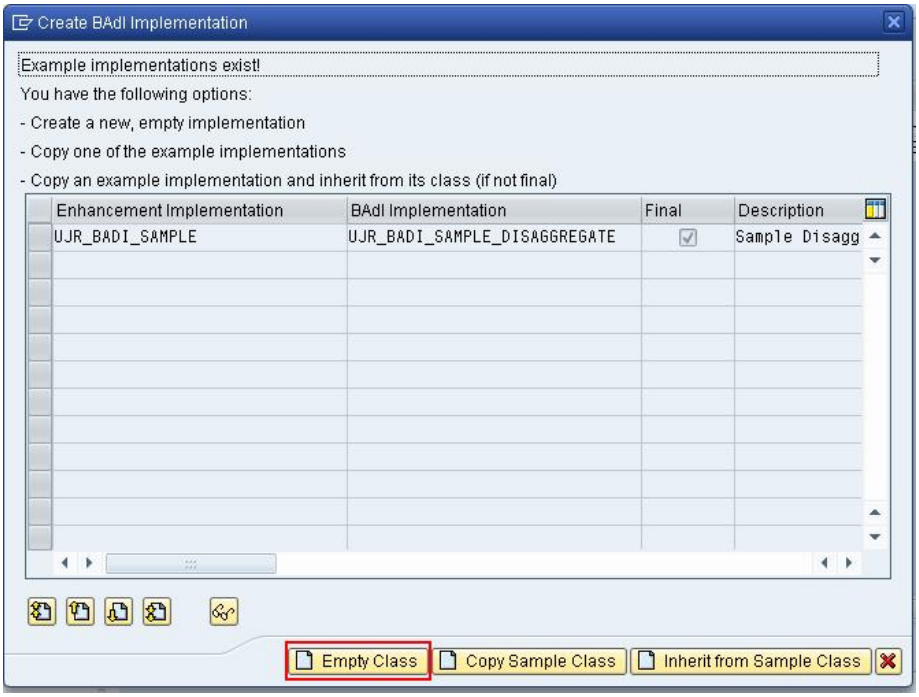
8. If you are presented with this dialog, then select your enhancement implementation which was just created from step 6, and click the green check. If there are no other previous implementations, go directly to the dialog box shown in step 9.



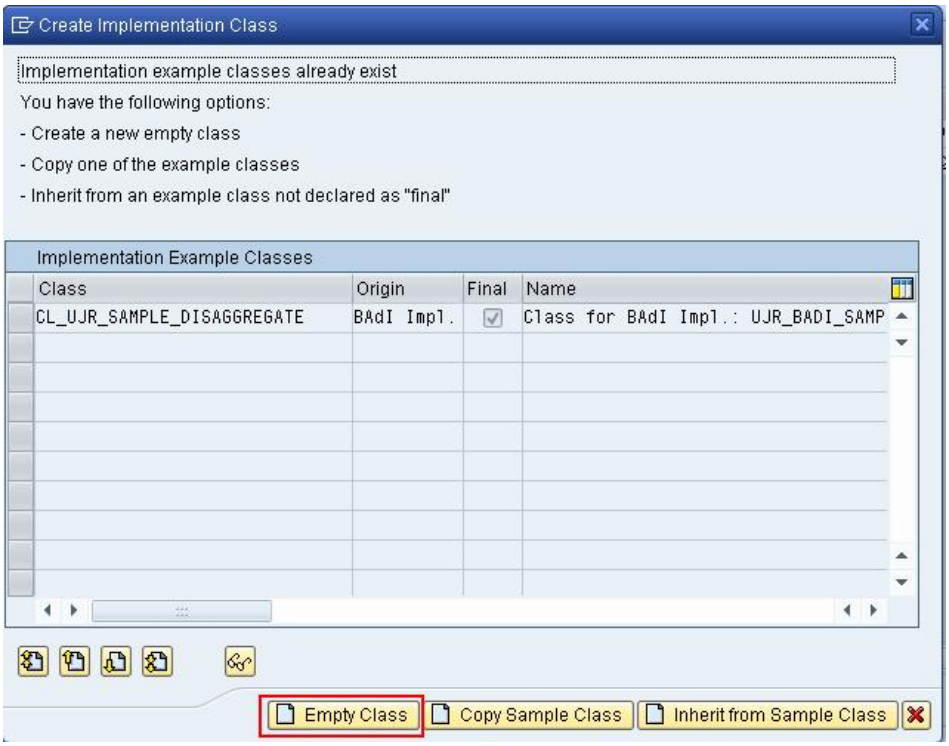
9. In this dialog, enter the name of the BAdI Implementation as ZUJ_WB_PRE_PROCESS_EX_IMP, and enter the description. Also, enter the name of the implementing class as ZCL_UJR_PRE_PROCESS_EX.



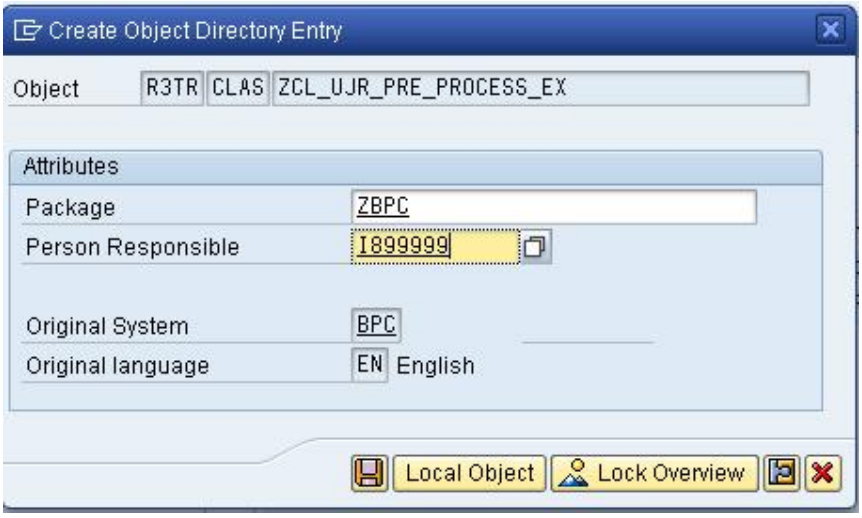
10. If you are presented with the following dialog for “Creating BAdI Implementation”, simply click the “Empty Class” button. You will not use the existing example BAdI implementation.



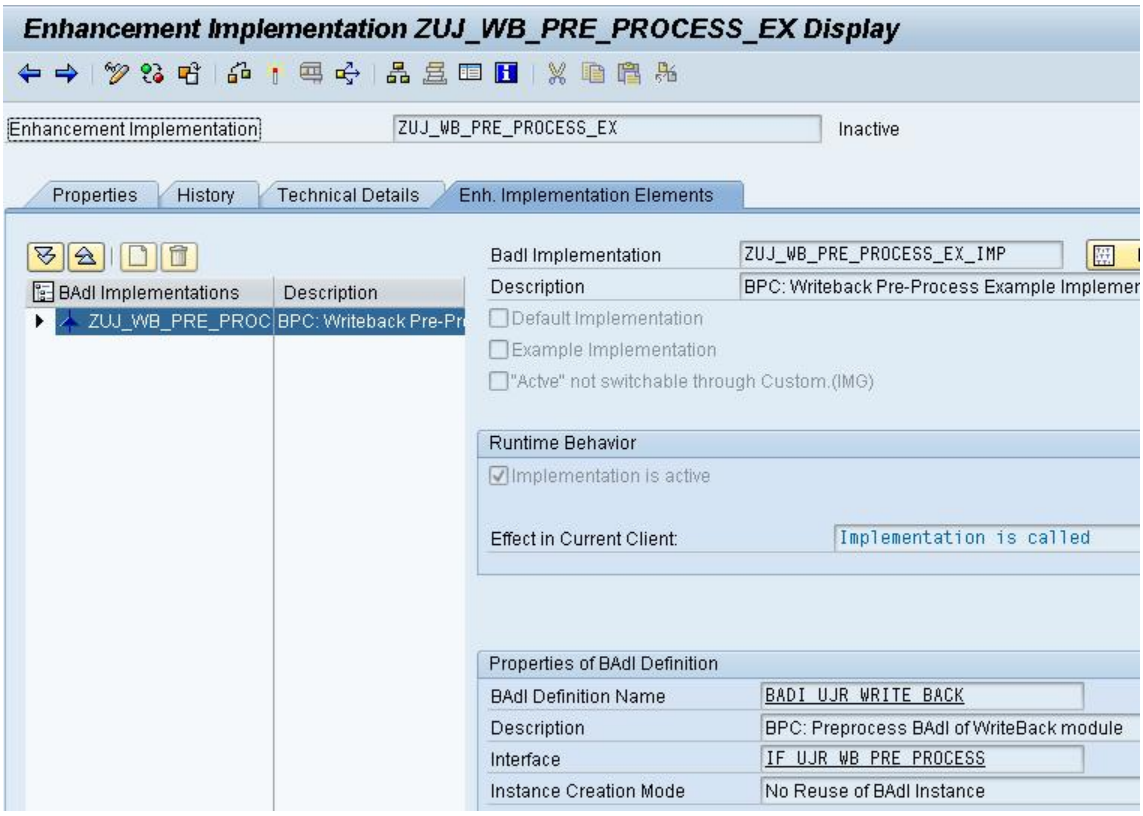
11. If you are presented with the following dialog for “Creating Implementation Class”, simply click the “Empty Class” button. This means that instead of copying the example class, you will create your implementation from scratch.



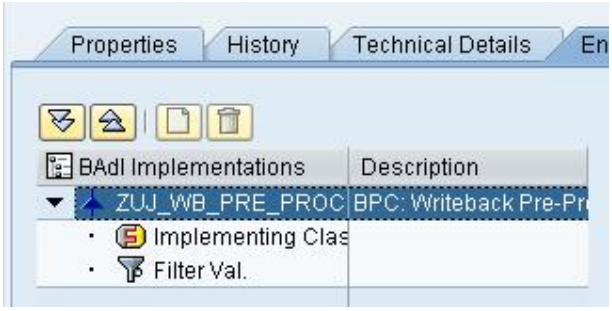
12. Enter the name of a package name for transporting this BAdI to another system in your landscape, or click “Local Object” if you do not plan to transport this BAdI.



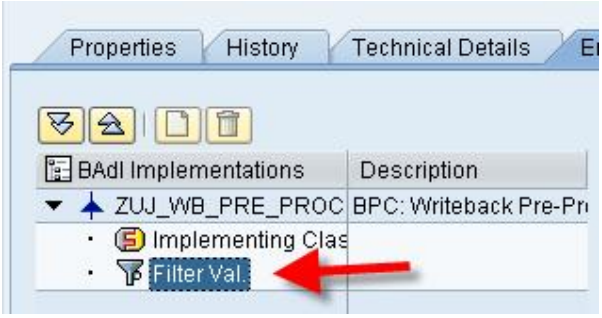
13. The BAdI Implementation will then be saved. Notice it is not yet active.



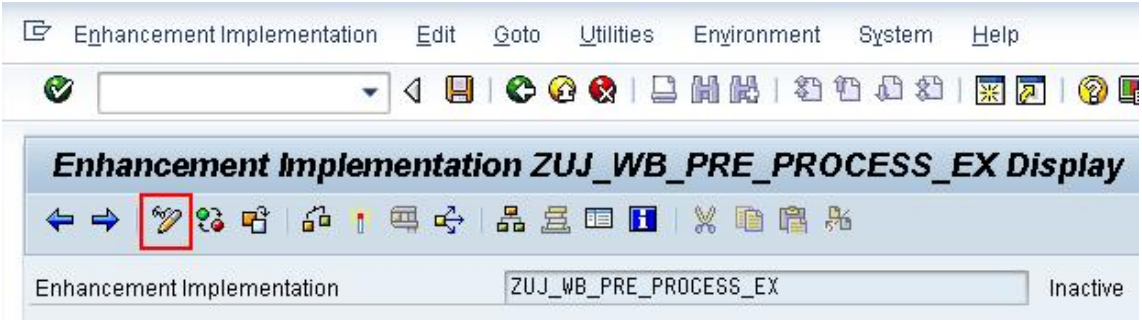
14. Click on the ▸ icon next to the name of the BAdI Implementation. This will expose the following nodes below.
    - Implementing Class
    - Filter Values



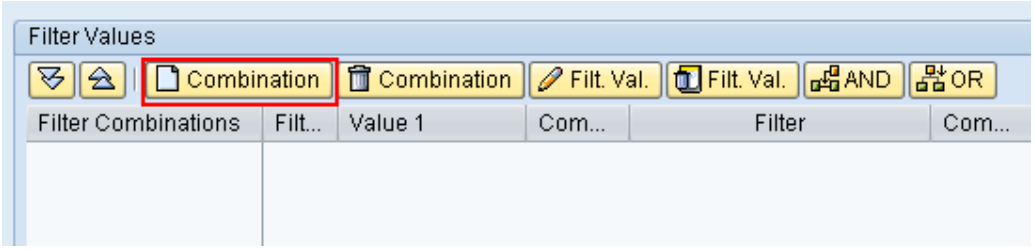
15. Double click on the “Filter Val.” Node.



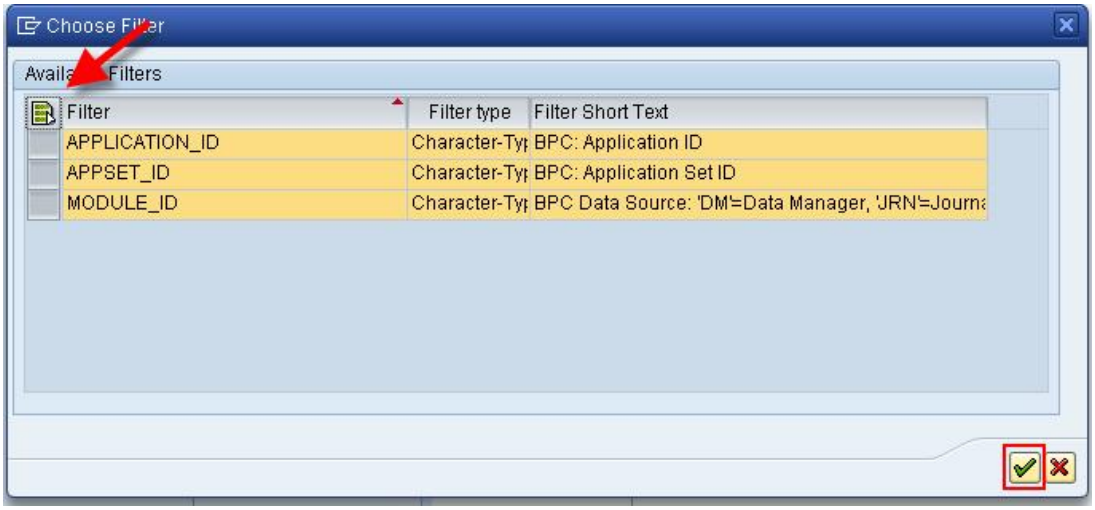
16. Click the “Change” icon.



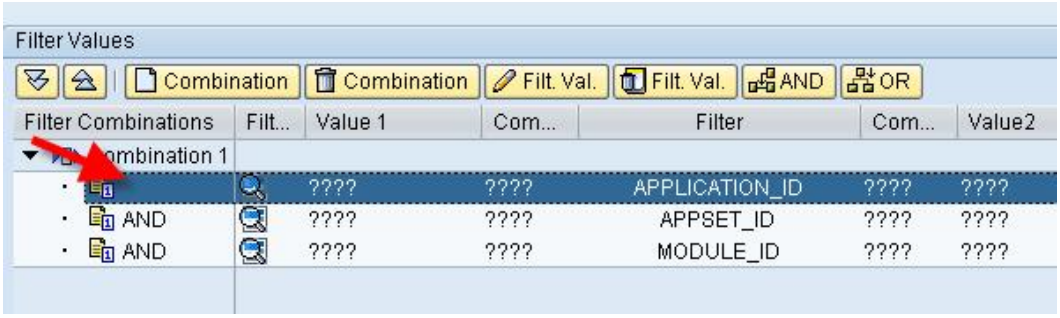
17. Click on the “Combination” button from the filter values screen.



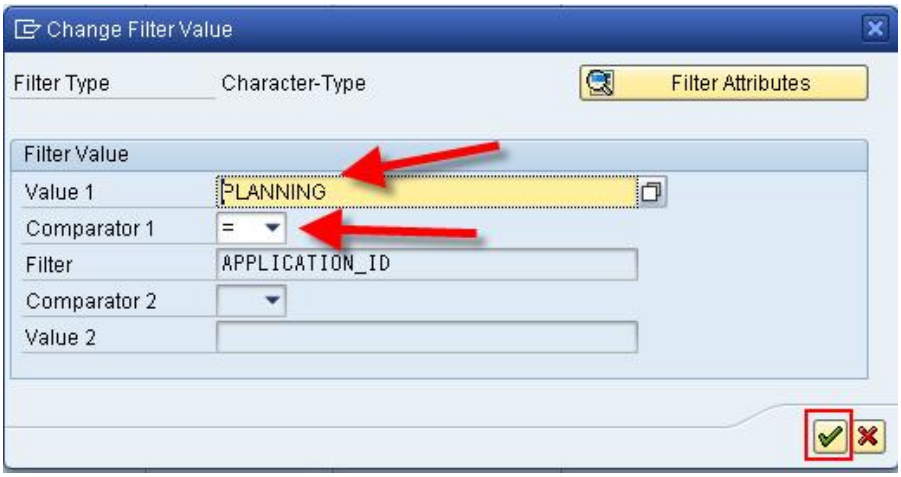
18. Select APPSET_ID, APPLICATION_ID and MODULE_ID, or click the “Select All” button, and then click the “Green Check” button to continue.



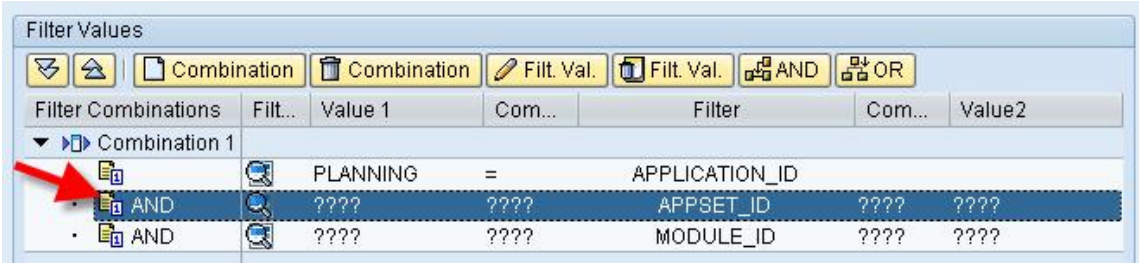
19. Now double click on the APPLICATION_ID line of the combination.



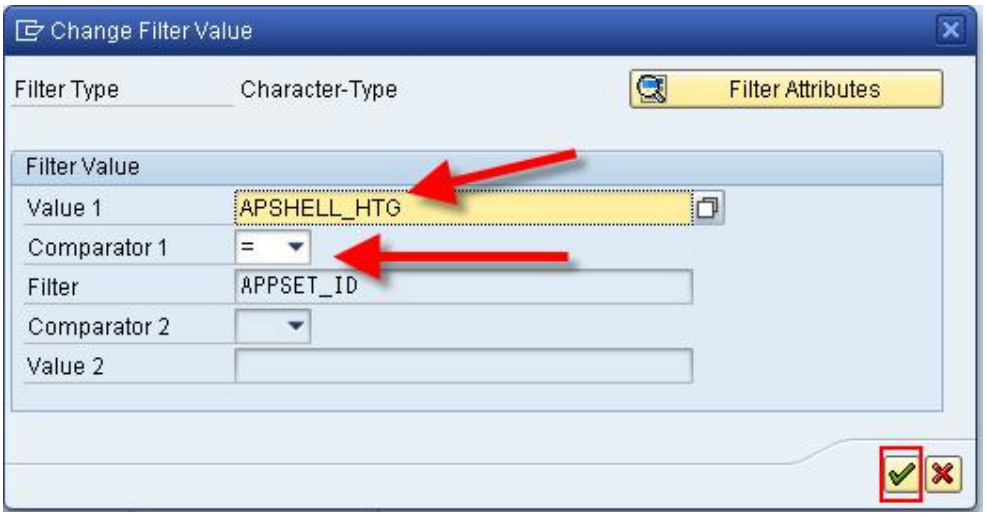
20. Enter the name of the application, which uses this BAdI implementation, into the “Value 1” field. In this example, the PLANNING application id used. Next set the drop down box for “Comparator 1” to “=”. Finally click the “Green Check” to continue.



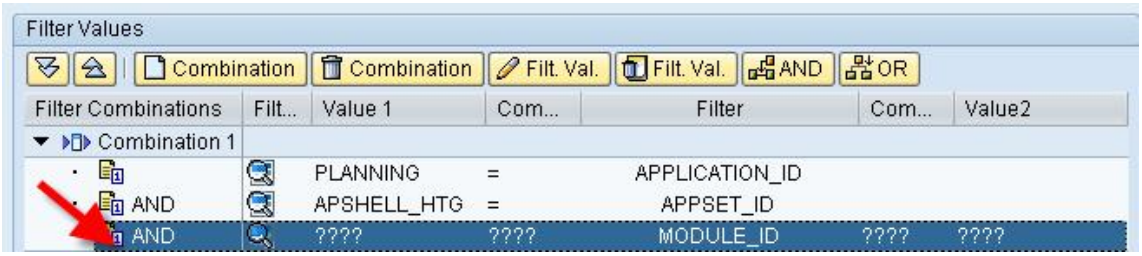
21. Next, double click on the APPSET_ID line of the combination.



22. Enter the name of the application set, which uses this BAdI implementation, into the “Value 1” field. In this example, APSHELL_HTG is used. Next set the drop down box for “Comparator 1” to “=”. Finally click the “Green Check” to continue.



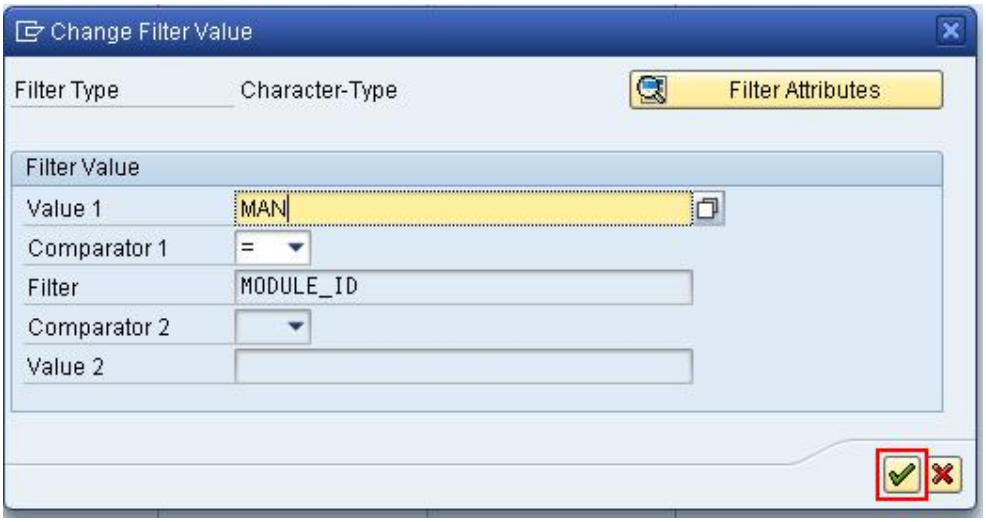
23. Next, double click on the MODULE_ID line of the combination.



24. Enter the module id, which is triggering this BAdI implementation, into the “Value 1” field. In this example, the MAN value is used. Possible values are:
    - DM Data Manager
    - MAN Manual Input
    - JRN Journal Entry
    - COMM Comment Entry
    - DOCS Document Modifications

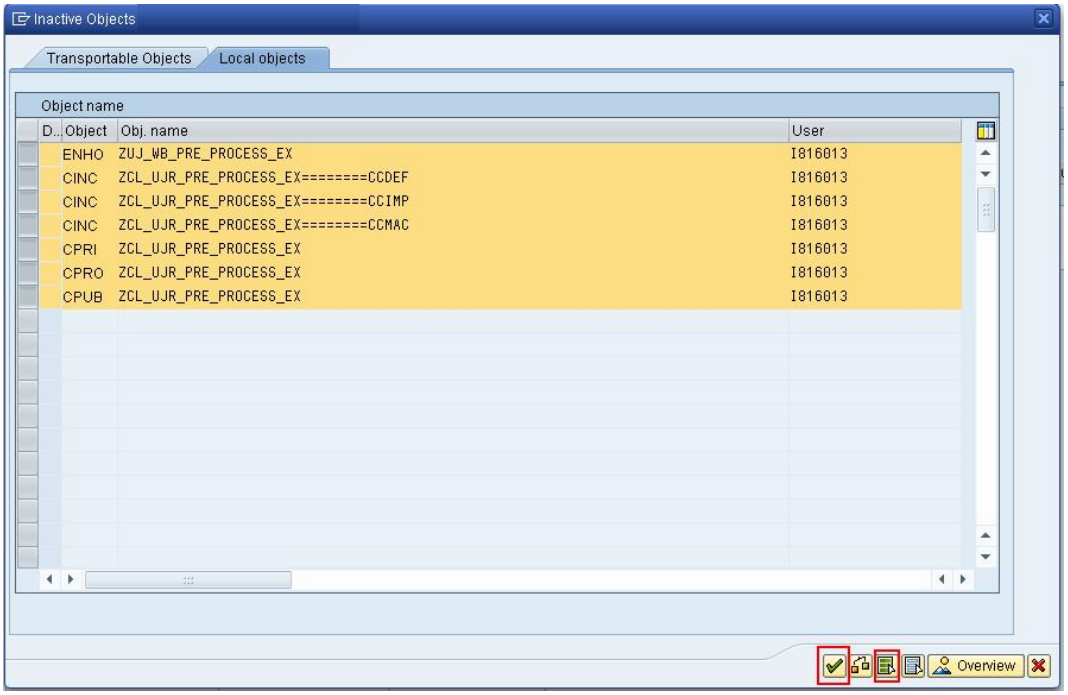
    Note: This value controls whether the BAdI implementation is executed based on the specific module in which the write back function is called. In this example, we only want the BAdI to be executed if the user does a manual entry from an input schedule, so we use the MAN module id. Multiple values for module id, within the same filter, are supported.

    Next set the drop down box for “Comparator 1” to “=”. Finally click the “Green Check” to continue.

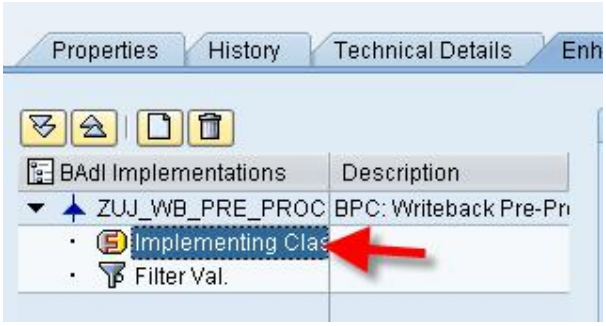


25. Finally, save and activate by clicking the appropriate buttons.

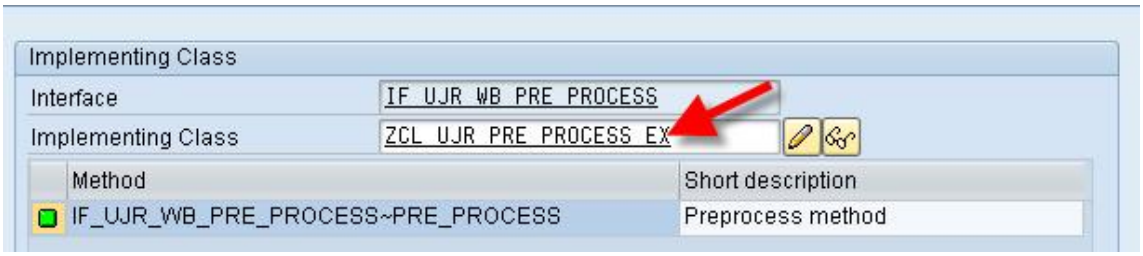
26. In the following dialog, first click the “Select All” button, and then the “Green Check” button. All objects should then be active.



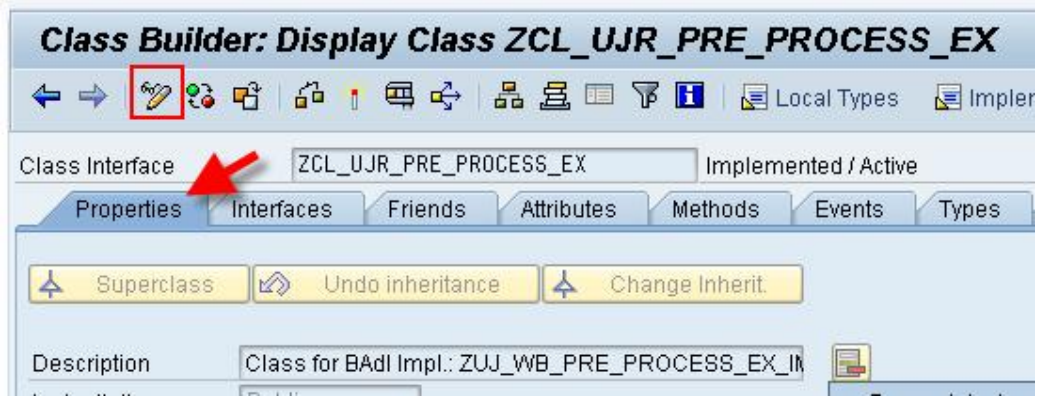
27. Finally, the code which will be executed by the BAdI implementation can be inserted into the implementing class. Double click on the “Implementing Class” node from the left side of the screen.



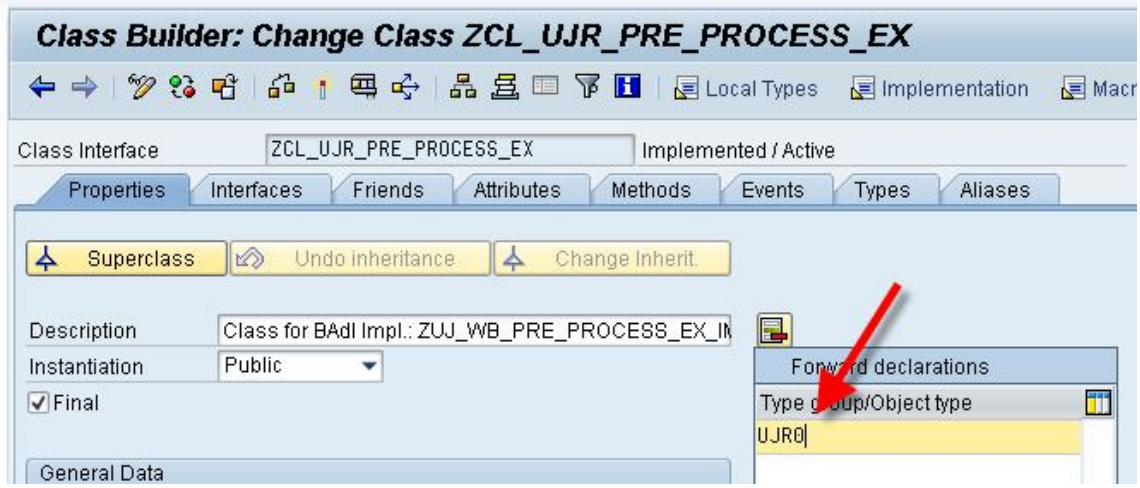
28. Now double-click on the implementing class name.



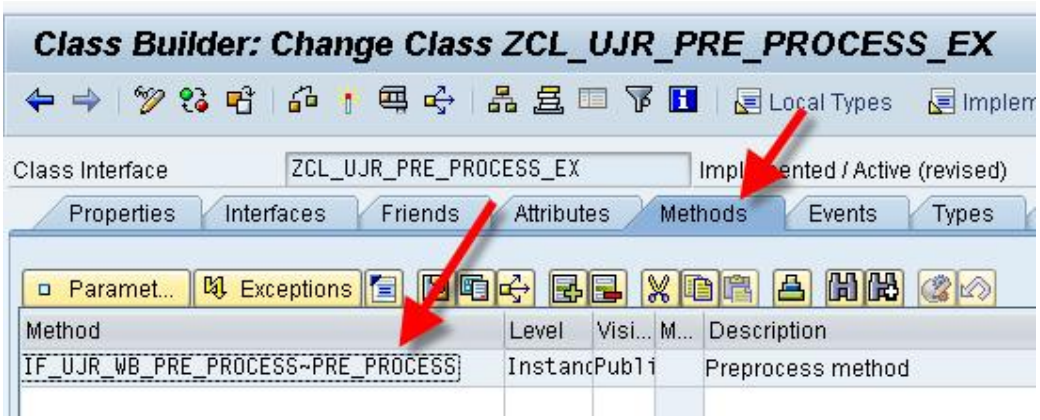
29. Due to forward navigation, the implementing class will be displayed in the class builder tool. Click on the “Properties” tab, and click the “Change” icon to enter change mode.



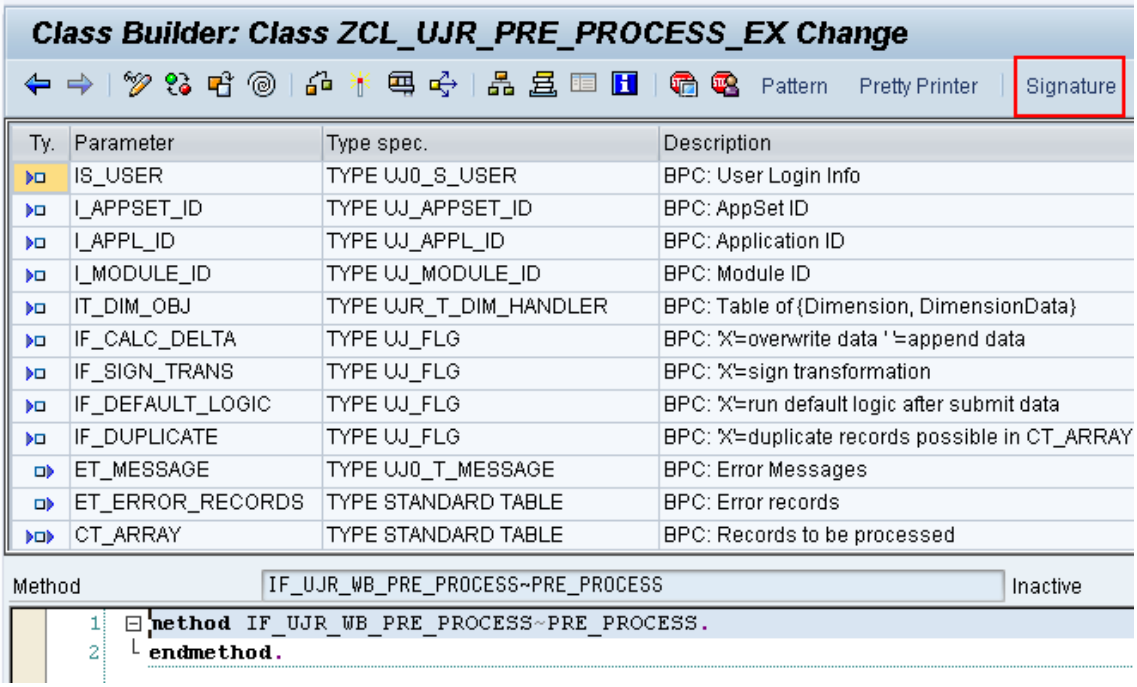
30. Add the type group UJR0 to this class by entering it in the box on the right.



31. Click on the “Methods” tab, then double-click on the PRE_PROCESS method.



32. An empty method implementation is shown. Notice, the method signature is displayed at the top. If the method signature is not displayed, click the “Signature” button on the application toolbar.



33. Copy and paste the source code from Appendix 5.1 into the PRE_PROCESS method.

Class Builder: Class ZCL_UJR_PRE_PROCESS_EX Change

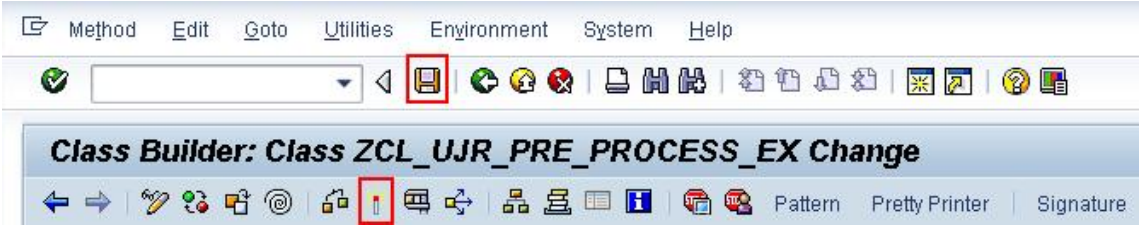
Pattern | Pretty Printer | Sig

Ty.	Parameter	Type spec.	Description
	IS_USER	TYPE UJ0_S_USER	BPC: User Login Info
	I_APPSET_ID	TYPE UJ_APPSET_ID	BPC: AppSet ID
	I_APPL_ID	TYPE UJ_APPL_ID	BPC: Application ID
	I_MODULE_ID	TYPE UJ_MODULE_ID	BPC: Module ID
	IT_DIM_OBJ	TYPE UJR_T_DIM_HANDLER	BPC: Table of (Dimension, DimensionDat
	IF_CALC_DELTA	TYPE UJ_FLG	BPC: 'X'=overwrite data ' '=append data
	IF_SIGN_TRANS	TYPE UJ_FLG	BPC: 'X'=sign transformation
	IF_DEFAULT_LOGIC	TYPE UJ_FLG	BPC: 'X'=run default logic after submit data
	IF_DUPLICATE	TYPE UJ_FLG	BPC: 'X'=duplicate records possible in CT_
	ET_MESSAGE	TYPE UJ0_T_MESSAGE	BPC: Error Messages
	ET_ERROR_RECORDS	TYPE STANDARD TABLE	BPC: Error records
	CT_ARRAY	TYPE STANDARD TABLE	BPC: Records to be processed

Method IF_UJR_WB_PRE_PROCESS~PRE_PROCESS Inact

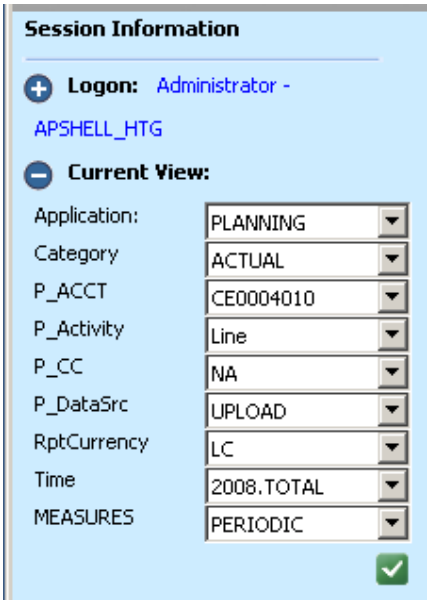
```
1  ***********************************************************************
2  * Sample BAdI implementation
3  * - Disaggregation only happens on Entity dimension
4  * - Disaggregate value into all children base members averagely
5  * - It is highly recommended BAdI only implements for InputSchedul
6  *   which means I_MODULE_ID = 'MAN'
7  ***********************************************************************
8  METHOD if_ujr_wb_pre_process~pre_process.
9
10   DATA: ls_entity TYPE ujr_s_dim_handler,
11         lt_entity_members TYPE uja_t_dim_member,       " dimension
12         lo_entity TYPE REF TO if_uja_dim_data,         " Object Ref
13         lt_hier_info TYPE uja_t_hier,                  " Hierachies
14         ls_hier_info TYPE uja_s_hier,
15         lt_hier_name TYPE uja_t_hier_name,             " Hierachies
16         lt_attr_list TYPE uja_t_attr,                  " Attributes
17         ls_attr_list TYPE uja_s_attr,
18         lt_attr_name TYPE uja_t_attr_name,             " Attributes
19         lf_non_base  TYPE uj_flg,                      " 'X'=non ba
20         lt_entity_mbr TYPE uja_t_dim_member,           " childen en
21         l_num_base   TYPE i,                           " number of
22         lr_data   TYPE REF TO data.
```

34. Save and activate the class by clicking the appropriate buttons.

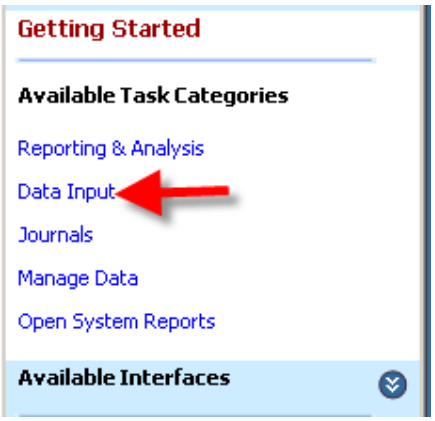


4.2 Test the BAdI Implementation

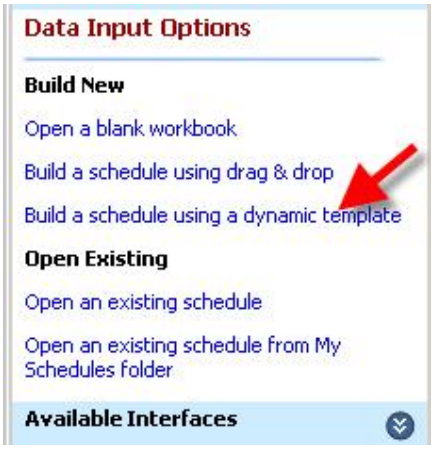
1. Go to the BPC Excel Client and log on to the application set. In this example, a copy of APSHELL called APSHELL_HTG is used. Set the current view as shown in the image below. In this example, the Entity dimension P_CC is used to do parent level planning, so make sure that the current view reflects that planning is being done for “North America”. It is important that all other values in the current view are set to base member values.



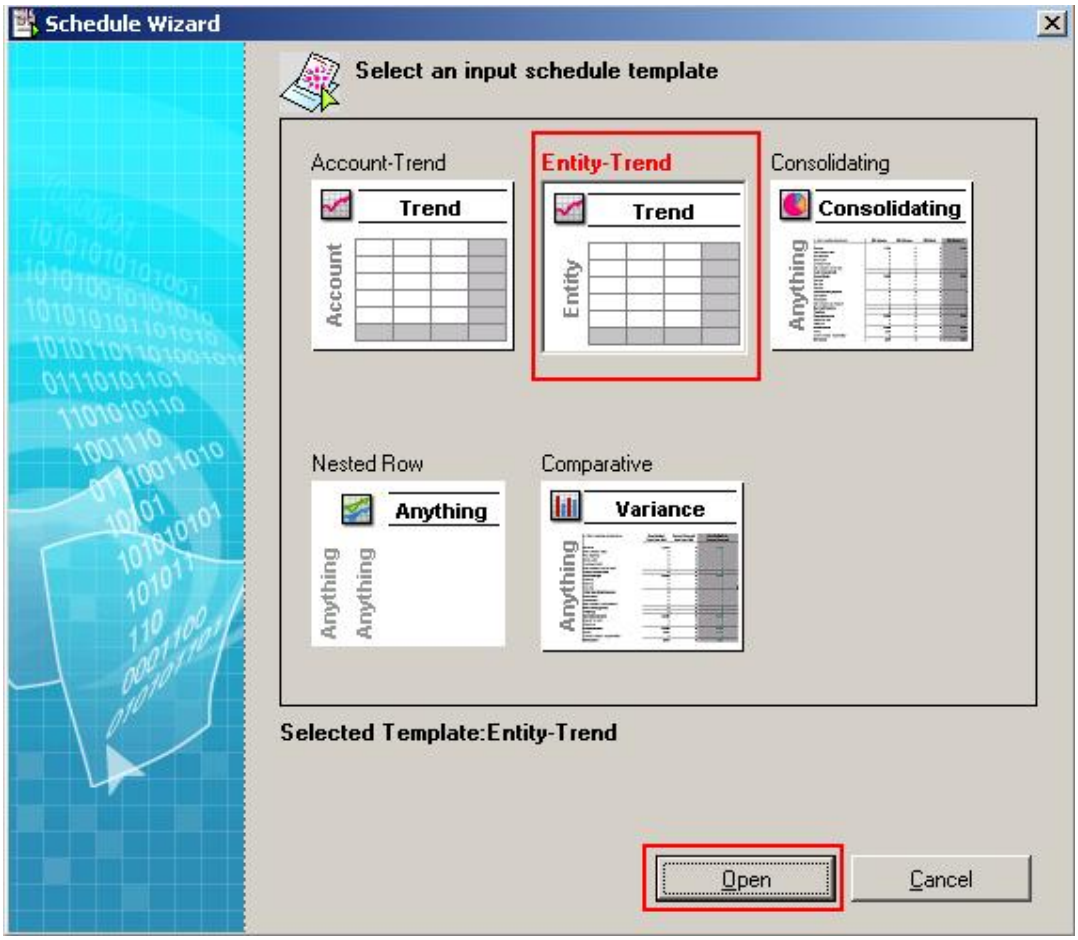
2. Next, from the action pane, choose “Data Input”.



3. Next, choose “Build a schedule using a dynamic template”.



4. Choose “Entity-Trend” and click the “Open” button.



5. You should now see a spreadsheet similar to the one displayed below. Notice that there are two company code lines, Canada and US, and a total line for the parent, North America.

SAP BusinessObjects

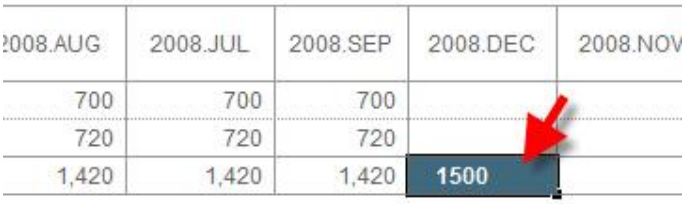
Input Schedule: Trend by Entity

Bonus Expense
ACTUAL, Periodic

	2008.FEB	2008.JAN	2008.MAR	2008.APR	2008.JUN	2008.MAY	2008.AUG	2008.JUL	2008.SEP	2008.DEC	200
Canada	700	700	700	700	700	700	700	700	700		
US	720	720	720	720	720	720	720	720	720		
North America	1,420	1,420	1,420	1,420	1,420	1,420	1,420	1,420	1,420		

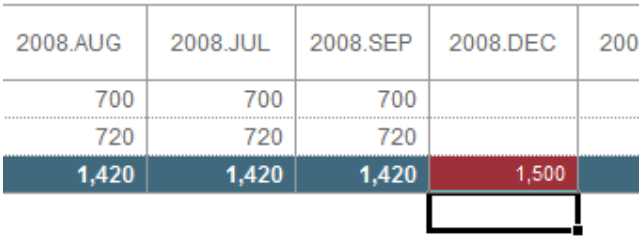
Schedule Data: ACTUAL | Bonus Expense | Uploaded Data | Local Currency | Periodic | Running the manufacturing lines
Refreshed: (GMT-08:00)3/30/2010 8:41:17 AM

6. Enter a value, such as 3,000 into the parent level row for North America, in the 2008.DEC column and press “Enter”.



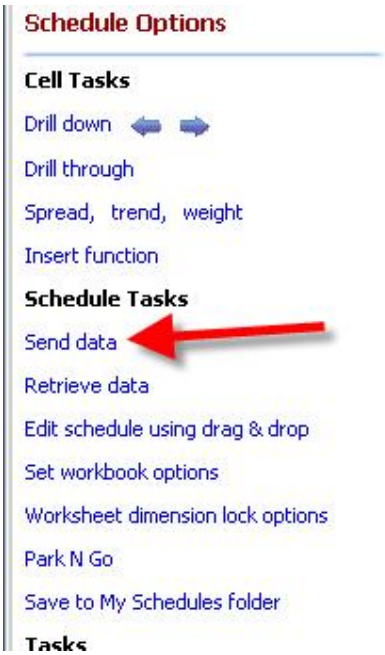
2008.AUG	2008.JUL	2008.SEP	2008.DEC	2008.NOV
700	700	700		
720	720	720		
1,420	1,420	1,420	1500	

7. The cell should now be highlighted in red.

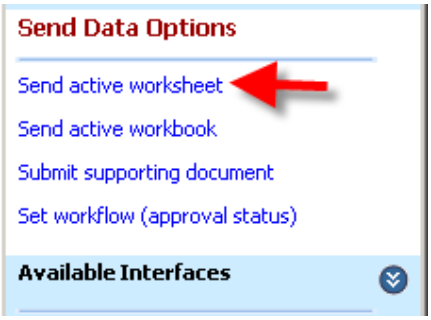


2008.AUG	2008.JUL	2008.SEP	2008.DEC	200
700	700	700		
720	720	720		
1,420	1,420	1,420	1,500	

8. From the action pane, click on “Send Data”.

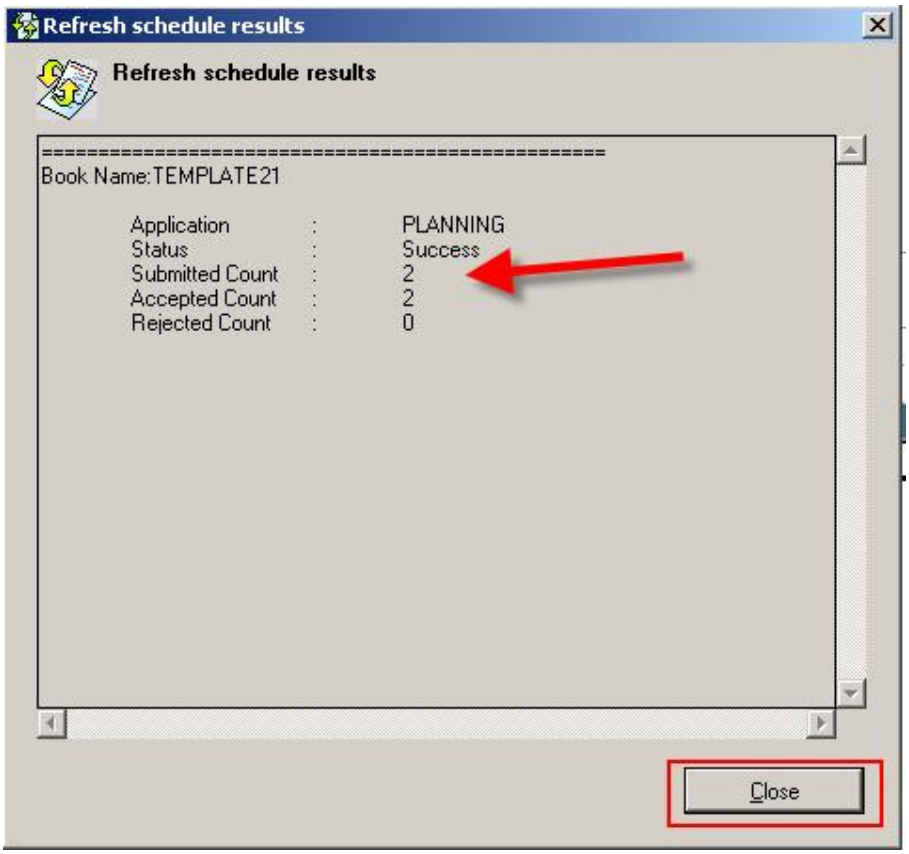


9. Click on “Send active worksheet”.

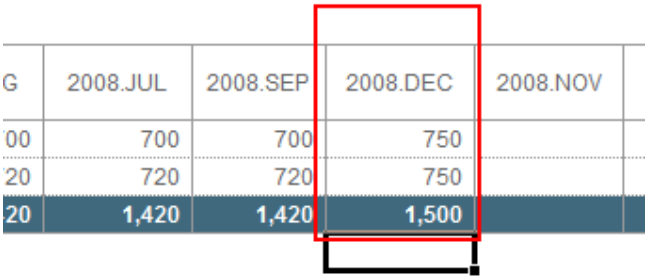


10. In the following dialog, click “Yes”.

11. The results dialog is displayed. Check the status for success and notice the record count. The BAdI implementation removed the record for “North America” and created two records for its children, Canada and US. It then split the inputted value between the two children.



12. Notice that the spreadsheet has been updated with the new values as well.



G	2008.JUL	2008.SEP	2008.DEC	2008.NOV
00	700	700	750	
20	720	720	750	
20	1,420	1,420	1,500	

5. Appendix

5.1 Source Code for BAdI Implementation

```
*********************************************************************
* Sample BAdI implementation
* - Disaggregation only happens on Entity dimension
* - Disaggregate value into all children base members averagely
* - It is highly recommended BAdI only implements for InputSchedule
*   which means I_MODULE_ID = 'MAN'
*********************************************************************
METHOD if_ujr_wb_pre_process~pre_process.

  DATA: ls_entity TYPE ujr_s_dim_handler,
        lt_entity_members TYPE uja_t_dim_member,      " dimension members of Entity
        lo_entity TYPE REF TO if_uja_dim_data,        " Object Reference to Dimension
        lt_hier_info TYPE uja_t_hier,                 " Hierachies Infos
        ls_hier_info TYPE uja_s_hier,
        lt_hier_name TYPE uja_t_hier_name,            " Hierachies name list
        lt_attr_list TYPE uja_t_attr,                 " Attributes Infos
        ls_attr_list TYPE uja_s_attr,
        lt_attr_name TYPE uja_t_attr_name,            " Attributes name list
        lf_non_base  TYPE uj_flg,                     " 'X'=non base member; ' '=base member
        lt_entity_mbr TYPE uja_t_dim_member,          " childen entity members
        l_num_base   TYPE i,                          " number of children entity members
        lr_data   TYPE REF TO data.

  FIELD-SYMBOLS: <ls_dim_obj>    TYPE ujr_s_dim_handler,
                 <ls_record>     TYPE ANY,
                 <l_entity>      TYPE uj_dim_member,  " Entity member of current records
                 <lt_entity_mbr> TYPE HASHED TABLE,   " All entity members
                 <ls_entity_mbr> TYPE ANY,
                 <lf_calc>       TYPE uj_flg,         " 'Y'=non base member
                 <lf_storedcalc> TYPE ANY,            " 'Y'=non base member
                 <l_base_mbr>    TYPE uj_dim_member,
                 <l_keyfigure>   TYPE ANY.            " Keyfigure

  " Find the Entity dimension by its type
  LOOP AT it_dim_obj ASSIGNING <ls_dim_obj> WHERE dim_type = uj00_cs_dim_type-entity.
    lo_entity ?= <ls_dim_obj>-dim_obj.
    ls_entity = <ls_dim_obj>.
```

```
ENDLOOP.

" Get hierachy (PARENTH1, PARENTH2 ...)
lo_entity->get_hier_list( IMPORTING et_hier_info = lt_hier_info ).
LOOP AT lt_hier_info INTO ls_hier_info.
  APPEND ls_hier_info-hier_name TO lt_hier_name.
ENDLOOP.

" Get necessary attributes (CALC and STORED_CALC)
lo_entity->get_attr_list( IMPORTING et_attr_list = lt_attr_list ).
LOOP AT lt_attr_list INTO ls_attr_list
    WHERE attribute_name = ujr0_c_attr_calc OR attribute_name = ujr0_c_attr_storedcalc.
  APPEND ls_attr_list-attribute_name TO lt_attr_name.
ENDLOOP.

" Get Members
CALL METHOD lo_entity->read_mbr_data
  EXPORTING
    if_ret_hashtab = abap_true
    it_attr_list   = lt_attr_name       " columns:attributes name list
    it_hier_list   = lt_hier_name       " columns:hieracies name list
  IMPORTING
    er_data        = lr_data.
ASSIGN lr_data->* TO <lt_entity_mbr>.

" preparation: create data structure and assign fields
CREATE DATA lr_data LIKE LINE OF ct_array.
ASSIGN lr_data->* TO <ls_record>.
ASSIGN COMPONENT ls_entity-dimension OF STRUCTURE <ls_record> TO <l_entity>.
ASSIGN COMPONENT ujr0_c_keyfigure OF STRUCTURE <ls_record> TO <l_keyfigure>.

LOOP AT ct_array INTO <ls_record>.

  READ TABLE <lt_entity_mbr>
              WITH TABLE KEY (ujr0_c_member_id) = <l_entity>
                     ASSIGNING <ls_entity_mbr>.
  IF sy-subrc = 0.

    " lf_non_base = <lf_calc>=Y OR <lf_storedcalc>=Y.
    ASSIGN COMPONENT ujr0_c_attr_calc OF STRUCTURE <ls_entity_mbr> TO <lf_calc>.
    lf_non_base = <lf_calc>.
    ASSIGN COMPONENT ujr0_c_attr_storedcalc OF STRUCTURE <ls_entity_mbr> TO <lf_storedcalc>.
    IF sy-subrc = 0 AND <lf_storedcalc> = ujr0_cs_calc-calculated_member.
```

```
        lf_non_base = ujr0_cs_calc-calculated_member.
      ENDIF.

      " Disaggregate non base member
      IF lf_non_base = ujr0_cs_calc-calculated_member.

        " A more precise version is to retrieve only accessible member of IS_USER
        CALL METHOD lo_entity->get_children_mbr
          EXPORTING
            i_parent_mbr     = <l_entity> " Parent
            i_level          = -99        " -99 = All children in any level; -1 = direct child
            if_only_base_mbr = abap_true  " Only base member
          IMPORTING
            et_member        = lt_entity_mbr.

        " Re-calculate the keyfigure, divide by N = number of base members
        " Usually it doesn't matter with IF_CALC_DELTA = false,
        " if the operation is linear mathematical.
        DESCRIBE TABLE lt_entity_mbr LINES l_num_base.
        " Avoid divide by zero
        IF l_num_base > 0.
          <l_keyfigure> = <l_keyfigure> / l_num_base.

          " Copy N times with new base members
          LOOP AT lt_entity_mbr ASSIGNING <l_base_mbr>.
            <l_entity> = <l_base_mbr>.
            " When IF_CALC_DELTA = true, appending means the latest records take effects,
            "   previous records with same dimension member will be overwritten.
            " The newly appended records will also be looped and processed.
            APPEND <ls_record> TO ct_array.
          ENDLOOP.

        ENDIF. " divide by zero

        " Remove the old one
        DELETE ct_array.

      ENDIF.
    ENDIF.
  ENDLOOP.

ENDMETHOD.
```

www.sdn.sap.com/irj/sdn/howtoguides

THE BEST-RUN BUSINESSES RUN SAP™