How-To Get Consistent Results with UJ_CUSTOM_LOGIC BAdi’s

Applicable Releases:

SAP BusinessObjects Planning and Consolidation 7.5, version for SAP NetWeaver
SAP BusinessObjects Planning and Consolidation 7.0, version for SAP NetWeaver

IT Practice / Topic Area:
Development
Version 1.0.0
November 2010
## Document History

<table>
<thead>
<tr>
<th>Document Version</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.0.0</td>
<td>Initial release</td>
</tr>
</tbody>
</table>
### Typographic Conventions

<table>
<thead>
<tr>
<th>Type Style</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><em>Example Text</em></td>
<td>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.</td>
</tr>
<tr>
<td><em>Example text</em></td>
<td>Emphasized words or phrases in body text, graphic titles, and table titles.</td>
</tr>
<tr>
<td><em>Example text</em></td>
<td>File and directory names and their paths, messages, names of variables and parameters, source text, and names of installation, upgrade and database tools.</td>
</tr>
<tr>
<td><em>Example text</em></td>
<td>User entry texts. These are words or characters that you enter in the system exactly as they appear in the documentation.</td>
</tr>
<tr>
<td><code>&lt;Example text&gt;</code></td>
<td>Variable user entry. Angle brackets indicate that you replace these words and characters with appropriate entries to make entries in the system.</td>
</tr>
<tr>
<td><strong>EXAMPLE TEXT</strong></td>
<td>Keys on the keyboard, for example, F2 or ENTER.</td>
</tr>
</tbody>
</table>

### Icons

<table>
<thead>
<tr>
<th>Icon</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>⚠</td>
<td>Caution</td>
</tr>
<tr>
<td>🔄</td>
<td>Note or Important</td>
</tr>
<tr>
<td>🌐</td>
<td>Example</td>
</tr>
<tr>
<td>🌐</td>
<td>Recommendation or Tip</td>
</tr>
</tbody>
</table>
Table of Contents

1. Business Scenario ............................................................................................................. 1
2. Background Information .................................................................................................. 1
3. Prerequisites ...................................................................................................................... 1
4. Example Configuration .................................................................................................... 2
   4.1 Create a copy of Apshell .............................................................................................. 2
   4.2 Add LEQ/AST Members to P_ACCT ........................................................................... 3
5. Development Tasks ......................................................................................................... 6
   5.1 Import the Z_CL_REVERSE_SIGN Class .................................................................. 6
       5.1.1 Copy R/K files to NetWeaver System .................................................................. 6
       5.1.2 Import the Transport of Copies ......................................................................... 6
   5.2 Create the Sample BAdi Implementation .................................................................. 9
       5.2.1 Creating the BAdi Implementation ................................................................... 9
       5.2.2 Set the Filter Value ......................................................................................... 12
       5.2.3 Implement IF_UJ_CUSTOM_LOGIC~EXECUTE ............................................... 14
       5.2.4 Update Default Logic to call the Sample BAdi ................................................ 16
6. Example Execution ........................................................................................................ 19
   6.1 Use Case 1 ................................................................................................................ 19
       6.1.1 Disable REVERSE_SIGNS call: ...................................................................... 19
       6.1.2 Submit Data via an Input Schedule ................................................................... 21
       6.1.3 Executing Default Logic via Data Manager ...................................................... 27
   6.2 Use Case 2 ................................................................................................................ 37
       6.2.1 Enable REVERSE_SIGNS call: ....................................................................... 37
       6.2.2 Submit Data via an Input Schedule ................................................................... 39
       6.2.3 Executing Default Logic via Data Manager ...................................................... 44
7. Combining with existing BAdis ...................................................................................... 48
1. Business Scenario

Sap BusinessObjects Planning and Consolidation, version for Netweaver (hereafter referred to as “BPC”) provides a BAdi (UJ_CUSTOM_LOGIC) that can be used to extend the functionality of script logic packages. Due to the way Input Schedule initiated script logic operations (i.e. included in default logic) differ from data manager initiated script logic operations you get inconsistent results with the same logic depending on where it was executed from. The reason for this is that Data Manager Packages return SIGNEDDATA whereas Input Schedules send Periodic values. The reason for the variation is as follows: Periodic values do not have signs reversed for LEQ/INC (Liability-Equity/Income) accounts while the SIGNEDDATA does.

2. Background Information

This How-To guide provides a solution to this in the form of a class that can be included in all BAdi implementations. It checks the source of data submitted to the BAdi, and if it originated from an Input Schedule it reverses the signs for INC/LEQ accounts and returns the updated “SIGNED DATA” in a new standard table. Otherwise, it will return the original, unaltered records in a new standard table. Logic in the BAdi can then perform calculations based on the data in the returned table which will allow for consistent results regardless of the logic execution method. This HTG also includes an example BAdi implementation based on the delivered Apshell application set.

3. Prerequisites

- SAP GUI / Developer Access to SAP Business Warehouse (SAP BW)
- SAP BusinessObjects Planning and Consolidation, version for SAP NetWeaver
  - Version 7.5
  - Version 7.0
4. Example Configuration

This section outlines the configuration of an example application set for use with the sample BAdi.

4.1 Create a copy of Apshell

The sample BAdi is based on the Planning application delivered in Apshell. The first step of configuring the example is to create a copy of Apshell. To do this:

1. Log into the BPC Administration Client

2. Select “Add New Application Set” from the Action Pane

<table>
<thead>
<tr>
<th>Manage Application Sets</th>
</tr>
</thead>
<tbody>
<tr>
<td>Application Set: Task</td>
</tr>
<tr>
<td>Add a new application set</td>
</tr>
<tr>
<td>Set application set status</td>
</tr>
<tr>
<td>Set template version</td>
</tr>
<tr>
<td>Refresh client-side dimension files</td>
</tr>
<tr>
<td>Delete an application set</td>
</tr>
<tr>
<td>User Activity</td>
</tr>
</tbody>
</table>

3. Enter the Application Set name, a description and select “APSHELL” as the application to duplicate. Select “Go to next step”.

4. Leave all options selected. Click “Add new Application Set”.

---

THE BEST-RUN BUSINESSES RUN SAP™
4.2 Add LEQ/AST Members to P_ACCT

In this section we will create master data in the P_ACCT dimension with the account types LEQ and AST (Assets).

To update the master data:

1. Log into the application set created in step 4.1 via BPC Administration.

2. Expand the Dimension Library in the left hand pane, and then select P_ACCT.

   **P_ACCT Dimension Tasks**
   - Maintain dimension property
   - Maintain dimension members
   - Refresh dimension member

4. Add the following members to P_ACCT highlighted in rows 31 through 34:

<table>
<thead>
<tr>
<th>ID</th>
<th>EVDESCRIPTION</th>
<th>PARENTID</th>
<th>ACCTYPE</th>
<th>SCALING</th>
<th>RATETYPE</th>
<th>D</th>
<th>I</th>
<th>HS_INPUT</th>
<th>ST</th>
<th>STK</th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td>Validation Account</td>
<td></td>
<td>Exp</td>
<td>Y</td>
<td>AVG</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>32</td>
<td>Total Liabilities and Equities</td>
<td></td>
<td>LRO</td>
<td>Y</td>
<td>AVG</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>33</td>
<td>Example LBO Account</td>
<td>CE00002000</td>
<td>LRO</td>
<td>Y</td>
<td>AVG</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>34</td>
<td>Total Current Assets</td>
<td></td>
<td>AST</td>
<td>Y</td>
<td>AVG</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>34</td>
<td>Example Asset</td>
<td>CE00003000</td>
<td>AST</td>
<td>Y</td>
<td>AVG</td>
<td>Y</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

5. Select “Save to Server” in the Action Pane:

   **Dimension Member Tasks**
   - View member report
   - Save to Server
6. Select “Process Dimension” in the Action Pane:

   **Dimensions Tasks**
   - Copy a dimension
   - Modify dimension
   - Process dimension
   - Delete dimension

7. Deselect “Take System Offline” and click “OK”.

![Process dimensions interface](image-url)
5. Development Tasks

This section outlines the development tasks involved in importing the Z_CL_REVERSE_SIGN class, as well as setting up and configuring the sample BAdi for use case testing in section 6.

5.1 Import the Z_CL_REVERSE_SIGN Class

This section describes the process of importing the Z_CL_REVERSE_SIGN class from a “transport of copies” into your NetWeaver system.

The Z_CL_REVERSE_SIGN class exposes one public method, REVERSE_SIGNS. This method takes the following import parameters:
- P_APPSET_ID – the application set where the incoming data resides.
- P_APP_ID – the application id where the incoming data resides.
- P_CT_DATA – the data to check for INC/LEQ accounts and reverse signs (when appropriate).

It exports the following parameter:
- E_CT_DATA – a table including all of the records in P_CT_DATA, with signed data reversed for INC/LEQ accounts (when appropriate).

This method checks the source of the request to see if it has been manually submitted (i.e. via an input schedule). If it has, the signs will be reversed for INC/LEQ accounts and all data, will be returned in P_CT_DATA. If the source of the request is a Data Manager package, all data will be returned in P_CT_DATA without any modification. This allows you to perform calculations based on P_CT_DATA in a consistent manner.

5.1.1 Copy R/K files to NetWeaver System

Download R/K files from this link.

1. Copy K900322.EPM to the usr\sap\trans\cofiles directory.

2. Copy R900322.EPM to the usr\sap\trans\data.

5.1.2 Import the Transport of Copies

1. Log into SAPGUI.

2. Go to the transport management system (t-code STMS)
3. Select the import overview (F5)

![Transport Management System Import Overview](image)

4. Double click the appropriate transport queue.

5. Select Extras -> Other Requests -> Add

![Other Requests](image)

6. Enter EPMK900322 as the Transport Request and click continue.

![Add Transport Request to Import Queue](image)

7. Select “Yes” in the Add Transport Request dialog.

![Add Transport Request](image)

8. Select EPMK900322 in the Request list and select Import Request (F11 – small truck) 

9. Enter the appropriate “Target Client” and select continue.
10. Select “Yes” to start the import.

11. You should see a line similar to the following in the request list once the import has completed successfully.
5.2 Create the Sample BAdi Implementation

This section describes the process of creating a sample BAdi that utilizes the Z_CL_REVERSE_SIGN class that was imported in section 5.1.

5.2.1 Creating the BAdi Implementation

To create the BAdi Implementation:

1. Log into SAPGUI

2. Go to the BAdi Builder (t-code SE18)
3. Enter UJ_CUSTOM_LOGIC as the Enhancement Spot and click the Display button.

4. Click the Create BAdi Implementation Button

5. Click the “Create Enhancement Implementation” button (F8).

6. Enter “Z_EL_REVERSE_SIGN” as the Enhancement Implementation as well as a descriptive short text, click continue.

7. Select “Local Object” in the “Create Object Directory Entry” dialog.

8. Select the Z_IE_REVERSE_SIGN Enhancement Implementation and click continue.
9. Enter the following in the “Create BAdi Implementation” dialog, then click continue:

```
Enhancement Implementation: Z_BADI_REVERSE_SIGN
Short Text: Impersonation BADI Sample
```

5.2.2 Set the Filter Value

This section continues from where section 5.2.1 leaves off.

1. Expand the BAdi implementation Z_BAdi_REVERSE_SIGN
2. Select Filter Val.

3. Click the Display/Change button.

4. Click the “Create Filter Combination” button.

5. Double click on the question marks “???” in the Value1 column.
6. Enter “REV_SIGN” in the Value 1 textbox and click continue

![Change Filter Value](image1)

7. Click the Activate button ( 

8. Activate all objects related to the sample BAdi.

![Inactive Objects for SETTD](image2)
5.2.3  Implement IF_UJ_CUSTOM_LOGIC~EXECUTE

In this section we will implement the Execute method for our sample BAdi. This sample BAdi calculates the difference between two sample accounts – an LEQ account (CE0002010) and an AST account (CE0003010) and store the value in the VALIDATION account. Before calculating the difference,

The instructions continue from where section 5.2.2 left off.

1. Expand the BAdi implementation Z_BAdi_REVERSE_SIGN

2. Select Implementing Class.

3. Double click on method IF_UJ_CUSTOM_LOGIC~EXECUTE

4. Select “Yes” in the Create Method Implementation dialog box.
5. Click the Display <-> Change button 🔄

6. Copy and paste the following code into the IF_UJ_CUSTOM_LOGIC~EXECUTE method body:

```abap
method if_uj_custom_logic~execute.
  constants:
    c_sdata type string value 'SIGNEDDATA',
    c_id type string value 'P_ACCT',
    c_common_stock type string value 'CE0002010',
    c_cash type string value 'CE0003010',
    c_difference type string value 'VALIDATION'.
  data:
    l_ref type ref to data,
    lo_reverse_sign type ref to z_cl_reverse_sign,
    l_log type string,
    l_reverse_signs type abap_bool.
  field-symbols:
    <l_signeddata> type any,
    <l_cs_signeddata> type any,
    <l_cash_signeddata> type any,
    <l_id> type any,
    <ls_line> type any,
    <ls_new_line> type any,
    <e_ct_data> type standard table,
    <lt_new_data> type standard table.

* setup ls_line and e_ct_data field symbol
create data l_ref like line of ct_data.
assign l_ref->* to <ls_line>.
assign l_ref->* to <ls_new_line>.

create data l_ref like table of <ls_line>.
assign l_ref->* to <e_ct_data>.

create object lo_reverse_sign.

* set to true to reverse signs
l_reverse_signs = abap_true.
* set to false to see default behavior
* l_reverse_signs = abap_false.

if l_reverse_signs = abap_true.
  try.
    lo_reverse_sign->reverse_signs(
      exporting
        p_appset_id = i_appset_id
        p_app_id = i_appl_id
        p_ct_data = ct_data
      importing
        e_ct_data = <e_ct_data>
    )
  esac.
```
catch cx_uj_custom_logic .
catch cx_uja_admin_error .

l_log = 'Exception encountered calling lo_reverse_sign_reverse_signs()' .
cl_ujk_logger->log( i_object = l_log ).
endtry.
else.
  <e_ct_data> = ct_data.
endif.

* this is a very simple implementation that will loop through all data sent
* set cash and commonstock signed data if found
* only works for a single time period
loop at <e_ct_data> assigning <ls_line>.
  assign component c_id of structure <ls_line> to <l_id>.
  if <l_id> eq c_common_stock .
    assign component c_sdata of structure <ls_line> to <l_cs_signeddata> .
  elseif <l_id> eq c_cash .
    assign component c_sdata of structure <ls_line> to <l_cash_signeddata> .
  endif.
endloop.

create data l_ref like ct_data.
assign l_ref->* to <lt_new_data> .

* if we have set signed data for cash and common stock, calculate kpi100
if <l_cs_signeddata> is assigned and <l_cash_signeddata> is assigned.
  move-corresponding <ls_line> to <ls_new_line>.
  assign component c_id of structure <ls_new_line> to <l_id> .
  assign component c_sdata of structure <ls_new_line> to <l_signeddata> .
  <l_id> = c_difference .
  <l_signeddata> = <l_cash_signeddata> + <l_cs_signeddata> .
  collect <ls_new_line> into <lt_new_data> .
endif.

ct_data = <lt_new_data> .
endmethod.

7. Click the Save button (.uintex remaining).

8. Click the Activate button (uintex remaining) and activate the updated method.

5.2.4 Update Default Logic to call the Sample BAadi
In this section, we will update default logic to call the sample BAadi.

1. Log into the application set created in section 4.1 via the BPC Administration client.
2. Expand Application -> Planning -> Script Logic and select DEFAULT.LGF.

3. Copy and paste the following script logic into DEFAULT.LGF.

```plaintext
// Setup member selection
*SELECT(%ACCT1%,"[ID]","P_ACCT","[ID]=CE0003010")
*SELECT(%ACCT2%,"[ID]","P_ACCT","[ID]=CE0002010")

*XDIM_MEMBERSET CATEGORY = ACTUAL
*XDIM_MEMBERSET P_ACCT = %ACCT1%,%ACCT2%
*XDIM_MEMBERSET P_ACTIVITY = Corp
*XDIM_MEMBERSET TIME = %TIME_SET%
*XDIM_MEMBERSET RPTCURRENCY = LC
*XDIM_MEMBERSET P_CC = %P_CC_SET%
*XDIM_MEMBERSET P_DATASRC = MANUAL

// Call sample impersonation BAdi
*START_BAdi REV_SIGN
QUERY = ON
WRITE = ON
*END_BAdi
```

4. Click Validate and Save in the Action Pane.
Applications

Manage Script Logic

Script Logic Task
Create New Logic

DEFAULT.LGF Task
Validate Only
Validate and Save
Delete Logic
Save Logic
6. Example Execution

In this section, we will review two use cases that demonstrate the behavior of the Z_CL_REVERSE_SIGN class, as well as our sample BAdi.

6.1 Use Case 1

This use-case demonstrates the default behavior of BPC / UJ_CUSTOM_LOGIC BAdi’s in default logic when sending data from an Input Schedule and executing default logic from a Data Manager package.

This is a three step process involving:

a. Disabling the REVERSE_SIGNS method call in the BAdi
b. Submitting data via an Input Schedule
c. Executing default logic via a Data Manager package against the same region used in step b.

6.1.1 Disable REVERSE_SIGNS call:

1. Log into SAPGUI

2. Go to T-Code SE19

3. Enter “Z_EI_REVERSE_SIGN” as the Enhancement Implementation and click “Change”:

![BAdi Builder: Initial Screen for Implementations](image)

4. Expand “Z_BAdi_REVERSE_SIGNS” and double click “Implementing Class”.

![Properties History Technical Details En](image)
5. Double click on the “IF_UJ_CUSTOM_LOGIC~EXECUTE” method

6. Click on the “Edit” button.

7. Comment out line 37 (by adding an *) and uncomment line 39 (by removing the *).

```
create object lo_reverse_sign.
34  * set to true to reverse signs
35  * l_reverse_signs = abap_true.
36  * set to false to see default behavior
37  l_reverse_signs = abap_false.
```

8. Click the Save button (✓).

9. Click the Activate button (✓) and activate the updated method.
6.1.2   Submit Data via an Input Schedule

1. Log into BPC for Excel
2. Set the current view to following:

```
<table>
<thead>
<tr>
<th>Current View:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Application:</td>
</tr>
<tr>
<td>Category:</td>
</tr>
<tr>
<td>P_ACCT:</td>
</tr>
<tr>
<td>P_Activity:</td>
</tr>
<tr>
<td>P_CC:</td>
</tr>
<tr>
<td>P_DataSrc:</td>
</tr>
<tr>
<td>RptCurrency:</td>
</tr>
<tr>
<td>Time:</td>
</tr>
<tr>
<td>MEASURES:</td>
</tr>
</tbody>
</table>
```

3. Click “Data Input” in the Action Pane.

**Getting Started**

**Available Task Categories**
- Reporting & Analysis
- Data Input
- Journals
- Manage Data
- Open System Reports

4. Click “Open a Blank Workbook” in the Action Pane.

**Data Input Options**

**Build New**
- Open a blank workbook
- Build a schedule using drag & drop
- Build a schedule using a dynamic template

**Open Existing**
- Open an existing schedule
- Open an existing schedule from My Schedules folder
5. Type “=evdre()” in cell A1 and click the Refresh button ( ).

6. Click “Yes” when asked if you want to clear existing input data.

7. Add TIME to columns and P_ACCT to rows. Set member selection to “BAS” and click OK.
8. Click the “+” sign above column D.

9. Enter “SELF” in the MemberSet for Expansion 1 and “CE0003010,CE0002010,VALIDATION” as the MemberSet for Expansion 2.

10. Click the Expand All button ( ).
11. Enter “10.00” as the value for 2005.JAN / Account CE0003010 and 2005.JAN / Account CE0002010.

12. Click “Send Data” in the Action Pane

13. Click “Send Active Worksheet” in the Action Pane:
14. Click “Yes” to send the data values.

15. Close the “Refresh Schedule results” dialog box
16. Notice that the VALIDATION account now has a value of “20”. The expected result would be 0. We are adding CE0003010 (10) and CE0002010 (10) however CE0002010 is a LEQ account so it should come across with the sign reversed (-10).

<table>
<thead>
<tr>
<th>D</th>
<th>E</th>
<th>F</th>
<th>G</th>
<th>I</th>
<th>J</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>App</td>
<td>Planning Application</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td></td>
<td></td>
<td>CATEGOR</td>
<td>ACTUAL</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
<td>MEASURE</td>
<td>Periodic</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
<td>P_ACCT</td>
<td>Total Costs</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
<td>P_ACTIVIT</td>
<td>Corporate management</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
<td></td>
<td>P_CC</td>
<td>US</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td></td>
<td></td>
<td>P_DATASI</td>
<td>Manual Planning</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td></td>
<td></td>
<td>RPTCURRU</td>
<td>Local Currency</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td></td>
<td></td>
<td>TIME</td>
<td>2005 JAN</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>12</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>13</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>CE0003010</td>
<td>Example Asset</td>
<td>10.00</td>
<td>2005 JAN</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>CE0002010</td>
<td>Example LEQ Account</td>
<td>10.00</td>
<td></td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>VALIDATION</td>
<td>Validation Account</td>
<td>20.00</td>
<td></td>
<td></td>
</tr>
<tr>
<td>17</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>18</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

⚠️ Note

If both Data Manager and Input Schedules viewed data with LEQ/INC in the same way the above behavior would not be an issue – however they perform differently as we will see in the next section.
6.1.3 Executing Default Logic via Data Manager

6.1.3.1 Create the Data Manager Package

1. Log into BPC for Excel.

2. Select “Manage Data” in the Action Pane.


4. Select “Manage Packages (Organize List)” in the Action Pane.
5. Click the “Add Package” button

6. Click to browse Process Chains

7. Select “Business Planning and Consolidation: System” in the left hand pane and /CPBM/RUNCALCACCOUNT in the right hand pane.
8. Enter a package name and description. Select the “User Package” checkbox. Click Add.

9. Click “Save”

10. Select “Manage Packages (Organize List)” in the Action Pane.
11. Right-click on the newly created package and select “Modify Package”.

12. Click the “View Package” button.
13. Click the “Advanced” button.

14. Overwrite the content in the right hand pane with the following and click “OK”.

```
PROMPT(SELECTINPUT,,,,%ENTITY_DIM%,%TIME_DIM%,,)
TASK(/CPMB/RUNCALCACCOUNT_LOGIC,SUSER,%USER%)
TASK(/CPMB/RUNCALCACCOUNT_LOGIC,SAPPSET,%APPSET%)
TASK(/CPMB/RUNCALCACCOUNT_LOGIC,SAPP,%APP%)
TASK(/CPMB/RUNCALCACCOUNT_LOGIC,SELECTION,%SELECTION%)
TASK(/CPMB/RUNCALCACCOUNT_LOGIC,LOGICFILENAME,DEFAULT.LGF)
```
16. Click “Save” in the “Modify Package” dialog.
17. Click “Save” in the Organize Package List dialog.

6.1.3.2 Execute the Package and view results

1. Click the back button (🪪) in the Action Pane

2. Click “Run a package” in the Action Pane
3. Select the newly created data manager package and click “Run”.

4. Enter “US” as P_CC and “2005.JAN” as TIME. Click “Next”.

5. Click Finish.

6. Click “OK” in the “Immediate Run…” dialog box.
7. Click “View Status” in the Data Manager dialog.

8. Ensure the data manager package finishes successfully.

9. Go back to the Input Schedule you created in step 6.1.2 and click the Refresh button ( ). As you can see the VALIDATION account is calculated as we would have expected.
<p>| | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>D</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>App Planning Application</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>CATEGORY ACTUAL</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>MEASURE Periodic</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>P_ACCT Total Costs</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>P_ACTIVI Corporate management</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>P_CC US</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>P_DATAS Manual Planning</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>RPTCURR Local Currency</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>TIME 2005 JAN</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>12</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>13</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>14</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>16</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>17</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>18</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>19</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- **2005 JAN**: 2005 JAN
- **CE0003010**: Example Asset
- **CE0003010**: Example LE0 Account
- **VALIDATION**: Validation Account
- **0.00**: 0.00
- **10.00**: 10.00
6.2 Use Case 2

This use-case demonstrates the new behavior of BPC / UJ_CUSTOM_LOGIC BAdi’s in default logic when using the REVERSE_SIGNS method in your BAdi’s.

This is a three step process involving:
   a. Enabling the REVERSE_SIGNS method call in the BAdi
   b. Re-submitting data via the same Input Schedule as Use Case 1.
   c. Re-executing default logic via the Data Manager package created in Use Case 1.

6.2.1 Enable REVERSE_SIGNS call:

1. Log into SAPGUI

2. Go to T-Code SE19

3. Enter “Z_EI_REVERSE_SIGN” as the Enhancement Implementation and click “Change”:

![BAdi Builder: Initial Screen for Implementations]

4. Expand “Z_BAdi_REVERSE_SIGNS” and double click “Implementing Class”.

![Properties History Technical Details En]

5. Double click on the “IF_UJ_CUSTOM_LOGIC~EXECUTE” method
6. Click on the “Edit” button.

7. Uncomment line 37 (by removing the *) and comment out line 39 (by adding a *).

8. Click the Save button ( )

9. Click the Activate button ( ) and activate the updated method.
6.2.2 Submit Data via an Input Schedule

Note
If you still have the Input Schedule created in section 6.1.2 you can open that and skip to step 11.

1. Log into BPC for Excel
2. Set the current view to following:

```
Current View:
```

```
Application: PLANNING
Category: ACTUAL
P_ACCT: CEO001000
P_Activity: Corp
P_CC: US
P_DataSrc: MANUAL
RptCurrency: LC
Time: 2005_JAN
MEASURES: PERIODIC
```

3. Click “Data Input” in the Action Pane.

Getting Started

Available Task Categories

- Reporting & Analysis
- Data Input
- Journals
- Manage Data
- Open System Reports

4. Click “Open a Blank Workbook” in the Action Pane.

Data Input Options

Build New
- Open a blank workbook
- Build a schedule using drag & drop
- Build a schedule using a dynamic template

Open Existing
- Open an existing schedule
- Open an existing schedule from My Schedules folder
5. Type “=evdre()” in cell A1 and click the Refresh button.

6. Click “Yes” when asked if you want to clear existing input data.

7. Add TIME to columns and P_ACCT to rows. Set member selection to “BAS” and click OK.
8. Click the “+” sign above column D.

9. Enter “SELF” in the MemberSet for Expansion 1 and “CE0003010,CE0002010,VALIDATION” as the MemberSet for Expansion 2.

10. Click the Expand All button ( )
11. Enter “15.00” as the value for 2005.JAN / Account CE0003010 and 2005.JAN / Account CE0002010.

<p>| | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
<td></td>
<td>App Planning Application</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
<td>CATEGOR ACTUAL</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
<td>MEASURE Periodic</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
<td>P_ACCT Total Costs</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
<td></td>
<td>P_ACTIVI Corporate management</td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td></td>
<td></td>
<td>P_CC CS</td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td></td>
<td></td>
<td>P_DATASI Manual Planning</td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td></td>
<td></td>
<td>RFTCURR Local Currency</td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td></td>
<td></td>
<td>TIME 2005.JAN</td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td></td>
<td></td>
<td>2005.JAN</td>
<td></td>
<td></td>
</tr>
<tr>
<td>12</td>
<td></td>
<td></td>
<td>2005.JAN</td>
<td></td>
<td></td>
</tr>
<tr>
<td>13</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>CE0003010</td>
<td>Example Asset</td>
<td>15.00</td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>CE0002010</td>
<td>Example LEQ Account</td>
<td>15.00</td>
<td></td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>VALIDATION</td>
<td>Validation Account</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>17</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

12. Click “Send Data” in the Action Pane

13. Click “Send Active Worksheet” in the Action Pane:
14. Click “Yes” to send the data values.

15. Close the “Refresh Schedule results” dialog box

16. Notice that the VALIDATION account now has a value of “0”. This is the expected result and will be consistent with Data Manager’s result (next section).
6.2.3 Executing Default Logic via Data Manager

1. Log into BPC for Excel.

2. Select “Manage Data” in the Action Pane.

3. Click “Run a package” in the Action Pane.
4. Select the data manager package created in Use Case 1 and click “Run”.

![Data Manager Package Selection]

5. Enter “US” as P_CC and “2005.JAN” as TIME. Click “Next.”

![Data Manager Run Package]

6. Click Finish.

7. Click “OK” in the “Immediate Run…” dialog box.
8. Click “View Status” in the Data Manager dialog.

9. Ensure the data manager package finishes successfully.
10. Go back to the Input Schedule you created in step 6.1.2 and click the Refresh button ( ). As you can see the VALIDATION account is calculated as we would have expected and we now have consistent results with our sample BAdi regardless of where it is executed from.

<p>| | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>12</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>13</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>14</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>16</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

App Planning Application
CATEGORACTUAL
MEASURE Periodic
P_ACCT Total Coste
P_ACTI  Corporate management
P_CC US
P_BATAS Manual Planning
RPTCURR Local Currency
TIME  2005 JAN

[2005 JAN]

<p>| | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>14</td>
<td>CE003010</td>
<td>Example Asset</td>
<td>15.00</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>CE002010</td>
<td>Example LEO Account</td>
<td>15.00</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>VALDATIN</td>
<td>Validation Account</td>
<td>0.00</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
7. Combining with existing BAdis

It is simple to add this functionality to new or existing BAdis, simply:

1. Go into your BAdi implementation.
2. Incorporate the following to the beginning of the BAdi implementation:

   ```
   data:
   l_ref type ref to data,
   lo_reverse_sign type ref to z_cl_reverse_sign,
   l_log type string,

   field-symbols:
   <ls_line> type any,
   <e_ct_data> type standard table,

   * setup ls_line and e_ct_data field symbol
   create data l_ref like line of ct_data.
   assign l_ref->* to <ls_line>.

   create data l_ref like table of <ls_line>.
   assign l_ref->* to <e_ct_data>.

   create object lo_reverse_sign.

   try.
   lo_reverse_sign->reverse_signs(
      exporting
      p_appset_id = i_appset_id
      p_app_id = i_appl_id
      p_ct_data = ct_data
      importing
      e_ct_data = <e_ct_data>
   ).
   catch cx_uj_custom_logic .
   catch cx_uja_admin_error .
   l_log = 'Exception encountered calling lo_reverse_sign->reverse_signs()'.
   cl_ujk_logger=>log( i_object = l_log ).
   endtry.

3. Use <e_ct_data>, which will have the same structure as ct_data, as the basis for all of your calculations.
www.sdn.sap.com/irj/sdn/howtoguides