How To…

Pass parameters to custom logic BADI using START_BADI

SAP Product Name: SAP Business Planning and Consolidation 7.0, version for SAP NetWeaver
Applicable Product Versions: 7.0 SP1 & above
Document Version: 1.1
Last Updated: Jan, 2009

Enterprise Performance Management

www.sdn.sap.com/irj/sdn/bpx-epm
1 Scenario

At customers, there often needs the ability to have custom exits to perform a certain piece of business logic or to optimize performance by working with some of the lower level operations of the system. This guide is an introduction to the exit framework with allows customers to utilize ABAP custom logic within their BPC Script Logic. In addition, it describes how these exits can be made generic and be parameter driven. Meaning that parameters can be passed from BPC Script Logic into these ABAP programs.

In this scenario, we will pass parameters to ABAP custom logic using START_BADI statements within Script Logic. As an example, we will be implementing Declining Depreciation function (Similar to DECD keyword from FOX) using BADI implementation.

2 Introduction

There will be multiple scenarios where we want to pass parameters from script logic to ABAP BADI implementation. For example, an BPC User wants to calculate rates from the Finance cube and post the results to rate cube. During this calculation, the administrator may want to pass the target AppSet and Application (cube) when calling the BADI. As of Business Planning and Consolidation 7.0 Version for NetWeaver Support Package 1 (Released on Dec 12 2008), we are introducing new keywords called *START_BADI and *END_BADI.

3 Details and Usage

The START_BADI and END_BADI syntax can be used to pass parameters to the ABAP custom logic. Between these tags, we can pass on multiple parameters and they all will be available inside the BADI implementation.

Syntax:

```
*START_BADI <filter_value_of_your_BADI_implementation>
  <key1> = <value1>
  <key2> = <value2>
  ..
*END_BADI
```

where `filter_value_of_your_BADI_implementation` is the name of the filter you provided during the BADI implementation of UJ_CUSTOM_LOGIC BADI.

There are two optional parameters you can use within a *START_BADI / *END_BADI instruction.

```
*START_BADI <filter_value_of_your_BADI_implementation>
  QUERY=ON/OFF //optional; default is OFF
  WRITE=ON/OFF //optional; default is OFF
*END_BADI
```

Query performs a default query and Write automatically writes back the data. Set QUERY parameter to OFF if you want to perform your own read inside the BADI implementation; set WRITE to OFF to not write back query results automatically. You may code to write to another application from the BADI.

This how to guide walks through a step-by-step example in the ApShell Planning cube. Please note you must have appropriate NetWeaver BI and BPC authorizations to perform the required steps.
4  The Step By Step Solution

We will first create the BADI implementation for the “Declining Depreciation” function and then, use the function in the ApShell Planning Application’s script logic.

4.1  Implement UJ_CUSTOM_LOGIC BADI

1. Log in to ABAP layer of your BPC through SAP Log on pad.

2. Go to transaction SE19.
3. Provide the Enhancement Spot as “UJ_CUSTOM_LOGIC” and click on “Create Impl.”.

4. In the next screen, Provide name and description for Enhancement Implementation. You don’t have to provide the Composite Enhancement Implementation, if you are not doing one.

   In this case, we are calling the enhancement implementation “ZBPC_DECD”, with short text “BPC: Declining-Balance Depreciation”.

5. Click the Green Check.

   NOTE: You will need a developer key to be able to create this implementation.
6. Provide your package name and click on Save button in the bottom.

7. In the next screen, provide name for BADI Implementation and Implementation Class. Choose BADI Definition as BADI_UJ_CUSTOM_LOGIC from the drop down.

In this case, we’re specifying
BADI Implementation: ZBPC_DEC_D
Implementation Class: ZCL_BPC_DEC_D
BADI Definition: BADI_UJ_CUSTOM_LOGIC

8. Click the Green check to continue.
9. Provide package information for the implementation class and then, click on Save.

10. Inside the Enhancement Implementation screen, expand the BADI Implementation.

11. Double click on “Filter Val”
12. Click on the “Create Filter Combination” button.

13. Double click on the “????” under the value 1 column.

14. Fill in the Value 1. This is the name users will be using to the BADI from the script. Use a name which will make sense to business users. Then, click OK.

   In this example, we will specify “DECD”. 
15. Now double click on “Implementing Class”.

16. Double click on method IF_UJ_CUSTOM_LOGIC~EXECUTE to create the implementation class.

17. System will ask for confirmation to create the implementation. Click on yes.
18. System will then prompt for saving Enhancement Implementation so far. Click on Yes.

19. System will take to EXECUTE method implementation. Click on the pencil button to go into change mode. Copy and the sample code provided in the Appendix or code your own.

20. Then, click on Activate button.
21. System will prompt for all the objects which you created. Choose the objects related to this whole implementation (Shown in the screen for this example) and click OK in the bottom. With successful Activation, we are ready with BADI implementation.
4.2 Create a new process chain template

In this step, we will create a new process chain template to execute a script logic file. In this guide, we go through the process of creating it manually, but if you want, you can begin by copying a delivered process chain. If you copy a delivered process chain, then you must be careful of all changes you make to shared process variants, as those changes will affect all process chains that are using those process variants.

1. Go to transaction RSPC within the ABAP System.

2. Click New.

3. Provide a meaningful name and description. In NW world, customers usually start new process chain with ZBPC*. But, follow your company naming convention.

   In this example, we’ll specify the process chain as “ZBPC_EXECUTE_LGF” and the long description as “ZBPC: Execute lgf”.

   ![SAP Easy Access](image1)

   ![Process Chain Display Planning View](image2)

   ![New Process Chain](image3)
4. System will prompt start variant. Create new start variant by clicking on new button.

5. Enter the process variant and long description and click the green check.

In this example, the process variant will be “ZBPC_EXECUTE_LGF_START” and the long description “ZBPC: Execute lgf start”.

6. In the next screen, click on “Change Selections” button under Scheduling Options.
7. Click on “Immediate” button on the top and press “Save” button.

8. Then, press save and then, go back.

9. Click OK.
10. Click save. In the whole process chain creation, remember to save your work for every step. Then, change the right hand side view to process type by clicking on the chain button.

11. Then, expand “BPC – Process Types”. Drag “BPC: Modify Dynamically” and drop in the right hand side area.

12. System will prompt to choose a variant. Create a new variant by clicking on the new button.
13. Provide Tech name and description and click OK.

In this example, specify process variant “ZBPC_DYN_MODIFY” and long description “ZBPC: Modify”.

14. Click save in the next screen and go back.

15. Click OK
16. System would have added the new variant in the right hand side.

17. Now, choose the first variant by clicking on it and then, drag & drop to the second variant to link them.

18. After successful link, system will show an arrow.
19. Now, drag and drop the “BPC: Run Logic” process.

20. Create new Run Logic step when prompted.


In this example, specify the process variant as “ZBPC_RUN_LOGIC” and the long description as “ZBPC: Run Logic”
22. In the next screen, enter below fields:

   TAB
   SUSER
   SELECTION
   SAPPSET
   SAPP
   REPLACEPARAM
   LOGICFILENAME
   EQU

   (TAB, REPLACEPARAM and EQU are the new fields to support dynamic replacements of prompts)

23. Click Ok. Then, connect this process variant with previous step (i.e. “Dynamic Modify”).

25. Expand General Services in the right hand side and choose “OR” process.

26. Create a new OR process variant and click OK.

27. Then, connect this OR process with the “Dynamic Modify” step.
28. Choose Errors action this time.

29. Connect the same OR step with Run Logic step. When the popup box comes up, choose “always” as the Action.
30. Choose this process from left hand side and drop.

31. Create new variant.

32. Provide tech name and description.

   In this example, proves variant “ZBPC_CLEAR_TEMP_TAB” and long description “ZBPC: Clear Temp Tables”.
33. In the next screen, add CLOSE_MODE as a field and give 0 as value. Press save and go back.

34. Then, connect this step with previous OR steps.
35. Save and activate the process chain by clicking on the candle button.

36. Now, we will be re-assigning this process chain to the right component so that it will be visible in BPC data manager. Click on “Display Components”
37. Choose BPC Examples component.

38. Save and activate again.

Now, the required process chain template is created.
4.3 **Develop the required script**
In this step, we will develop the script calling the BADI which we created in the previous section.

1. Login to the BPC Administration.

2. Navigate to the AppSet/Application where you want to create the script Logic. Then, click on “Create New Logic” from right hand side Action Pane.

   In this example, we’re going to use a copy of the ApShell AppSet and use the “Planning” application.

3. Provide a meaningful name.
4. Write your script logic here. In this example, copy and paste the script logic which is included in the appendix of this guide. Then, click on “Save Logic”
4.4 Develop the data manager package to execute the script

In this step, we will create a data manager package to execute the script logic using the process chain we created in the previous step.

1. Log in to Excel Client.

2. After logging in, click on “Manage Data” in the Action pane.
3. Click on “Maintain data management” in the next menu.

4. Then, click on “Manage packages (organize list)”.
5. Choose the desired team and click on “Create package” to add a new package.

6. Click on open folder button to select a process chain for this package.

7. Choose the process chain which you created in the earlier section.
8. Enter Package name and Description (Avoid using special characters for naming the package.) and choose a group as well. Make sure to select the desired task type as well. Then, click Save.

9. On the next screen click save to store the package in the system.

10. You will get a message that package list successfully updated.
11. Go back to “Manage packages (organize list)” to modify the package

12. Choose the package and click on Modify Package.

13. Click “view package” (button next to open folder on the same line as process chain).
14. Click on “Advanced” button.

15. Here, maintain the dynamic script (sample code is provided in the appendix). Then, click ok.

16. Press Save in the next screen.
17. Press save again

18. Press save again.
4.5 Load some sample data
In this step, we will load some sample data to work with in order to show the value of this function.

1. Login to the BPC Office Client.

2. Click on “Manage Data”.

![Image of BPC Office Client login and Manage Data interface]
3. Click “Upload data file”.

4. Choose your source file. In this case, this would be the “P_ACCT.CSV” file attached to this how-to guide.

5. Choose the destination file. In this case, we’re loading the P_ACCT.CSV file to the Examples folder under “Company”.

6. Click OK.

7. Click “Upload data file” again.

8. Choose the “PLANNING.CSV” file which is attached to this guide.
9. Specify the destination file. In this example, we’re going to load the “PLANNING.CSV” file to the EXAMPLES folder in the “Company” team.

10. Click OK.
11. Click on “Run a data management package”.

12. Click “Import Master Data” and click Run.
13. For import file, specify the “P_ACCT.CSV” file which we uploaded earlier.

14. For the transformation file, choose the “IMPORT.XLS” transformation file.

15. For the dimension name, choose “P_ACCT”. Click Finish.
16. Click View Status to see that the data loaded successfully.

17. You should see a “SUCCESS” message.

18. Now run the “Append” package under “System Administration”.
19. Under “Import File”, choose the “PLANNING.CSV” file that we uploaded earlier.

20. For the transformation file, choose the “IMPORT.XLS” transformation file.

21. Specify “No” for running default logic and “No” for checking work status and click “Finish”. 
22. Click View Status.

23. You should see that the two records were loaded successfully.

24. Make sure you refresh dimension members in the Etools menu.

4.6 Run the package and verify data
We will now execute the package and validate the data.
1. Create blank excel and type in `EVDRE()` to pull sample data.

2. Click Refresh.

3. Click OK.
4. Depending upon your current view selection, data will get pulled.

5. Adjust dimension filters as necessary. Sample selection is shown here in the current view.

6. Check the data of the account before running the script logic on that account.
7. In Excel client, go back in the action pane to choose “Manage Data”.

8. Then, choose Run a data management package.
9. Choose the data package and click on Run.

10. Choose the script file and enter the deprecation percentage.

   In this example, the depreciation files is “DECD.LGF” and the deprecation percentage is “10”.

11. Fill in the data selection as necessary and choose when you want to run the package. Then, click Next.
12. Confirm the selection and click Finish.

13. Click OK on the message and verify the logs.

14. Now, click on the View Status check the log of the script.
15. Double click on your current run line to see the details of the log.

16. Here you can view the whole log.

17. Formula log gives more info.
18. Sample function developed will post the depreciated value in next year for the same account. Within current view, adjust the time filter to choose the next year (2009.JAN).

19. You can now validate the result value.
5 Appendix

5.1 Sample Code of BADI EXECUTE method:
This EXECUTE method expects number of Years and percentage of depreciation (as whole number). For all incoming accounts, depreciates the value by the given percentage for each year and posts the result value in the final year (only).

```
METHOD if_uj_custom_logic~execute.

  DATA: ls_param TYPE ujk_s_script_logic_hashentry,
        l_log   TYPE string,
        l_ast_acct(16) TYPE c,
        l_year(3) TYPE n,
        l_percentage(3) TYPE p,
        lo_model TYPE REF TO if_uj_model,
        lo_dim TYPE REF TO if_uja_dim_data,
        ls_dim TYPE uja_s_dim,
        time_dim(16) TYPE c,
        lr_rec TYPE REF TO data,
        lr_result_rec TYPE REF TO data,
        l_intermediate_value TYPE uj_sdata,
        lt_final TYPE REF TO data.

  FIELD-SYMBOLS: <ls_rec> TYPE ANY,
                  <ls_result_rec> TYPE ANY,
                  <ls_time> TYPE ANY,
                  <ls_signeddata> TYPE ANY,
                  <lt_final> TYPE STANDARD TABLE.

  * Make sure all the parameters are passed.

  CLEAR ls_param.
  READ TABLE it_param WITH KEY hashkey = 'YEAR' INTO ls_param.
  IF sy-subrc NE 0.
    l_log = 'You have not specified the parameter ''YEAR'' which is required.'.
    cl_ujk_logger=>log( i_object = l_log ).
    RAISE EXCEPTION TYPE cx_uj_custom_logic.
    EXIT.
  ENDIF.
  l_year = ls_param-hashvalue.

  CLEAR ls_param.
  READ TABLE it_param WITH KEY hashkey = 'PERCENTAGE' INTO ls_param.
  IF sy-subrc NE 0.
    l_log = 'You have not specified the parameter ''PERCENTAGE'' which is required.'.
    cl_ujk_logger=>log( i_object = l_log ).
    RAISE EXCEPTION TYPE cx_uj_custom_logic.
    EXIT.
  ENDIF.
  l_percentage = ls_param-hashvalue.

  *Get name of the account dim
  cl_uj_model=>get_model( EXPORTING i_appset_id = i_appset_id
                   RECEIVING ro_model    = lo_model ).

  CALL METHOD lo_model->get_dim_data_by_type
    EXPORTING
      i_dim_type    = uj00_cs_dim_type-time
      i_appl_id     = i_appl_id
    RECEIVING
      ro_dim_data  = lo_dim.
```
TRY.
   CALL METHOD lo_dim->get_info
   IMPORTING
       es_dim_info = ls_dim.
ENDTRY.

 time_dim = ls_dim-dimension.
TRANSLATE time_dim TO UPPER CASE.

CREATE DATA lt_final LIKE ct_data.
ASSIGN lt_final->* TO <lt_final>.
CREATE DATA lr_result_rec LIKE LINE OF ct_data.
ASSIGN lr_result_rec->* TO <ls_result_rec>.
CREATE DATA lr_rec LIKE LINE OF ct_data.
ASSIGN lr_rec->* TO <ls_rec>.

* Loop thro incoming data and create result set
LOOP AT ct_data ASSIGNING <ls_rec>.
   <ls_result_rec> = <ls_rec>.
   ASSIGN COMPONENT time_dim OF STRUCTURE <ls_result_rec> TO <ls_time>.
   <ls_time>+0(4) = <ls_time>+0(4) + l_year.

   ASSIGN COMPONENT 'SIGNEDDATA' OF STRUCTURE <ls_result_rec> TO <ls_signeddata>.
   DO l_year TIMES.
       l_intermediate_value = l_intermediate_value + ( <ls_signeddata> -
       l_intermediate_value ) * l_percentage / 100.
   ENDDO.
   <ls_signeddata> = <ls_signeddata> - l_intermediate_value.
   APPEND <ls_result_rec> TO <lt_final>.
ENDLOOP.

* Send the result data back. BPC always over-writes existing value. So, send
* the latest values.
ct_data = <lt_final>.
ENDMETHOD.

5.2 Sample Source Code for Script Logic file:

//We will let the users choose the scope region

*START_BADI DECD
   QUERY = ON
   WRITE = ON
   YEAR = 1
   PERCENTAGE = $PERCT$
*END_BADI

5.3 Sample data manager dynamic script in the package:

PROMPT(SELECTINPUT,,"%ACCOUNT_DIM%,%ENTITY_DIM%,%CATEGORY_DIM%,%CURRENCY_DIM%,%TIME_DIM%")
PROMPT(TEXT,%SCRIPT_FILE%,"Choose Script Logic File")
PROMPT(TEXT,%PERCT%,"Input Depreciation Percentage")
INFO(%EQU%,=)
INFO(%TAB%,;)
TASK(ZBPC_RUN_LOGIC,TAB,%TAB%)
TASK(ZBPC_RUN_LOGIC,EQU,%EQU%)
TASK(ZBPC_RUN_LOGIC,SUSER,%USER%)
TASK(ZBPC_RUN_LOGIC,SAPPSET,%APPSET%)
TASK(ZBPC_RUN_LOGIC,SAPP,%APP%)
TASK(ZBPC_RUN_LOGIC,SELECTION,%SELECTION%)
TASK(ZBPC_RUN_LOGIC,LOGICFILENAME,%SCRIPT_FILE%)
TASK(ZBPC_RUN_LOGIC,REPLACEPARAM,PERCT%EQU%$PERCT%)