Applies to:
SAP BI NetWeaver 7.0. For more information, visit the Business Intelligence homepage.

Summary
This paper describes step by step process of implementing virtual key figures and characteristics. It is accomplished through Business Add-in (BAdi), which uses object oriented programming concepts. This creates an alternate approach while modeling complex scenarios for established systems.

Author: Ashish Tewari
Company: Deloitte Consulting India Pvt. Ltd
Created on: 25 June 2009

Author Bio
Ashish Tewari is currently working with Deloitte. He is a Business Intelligence Solution Consultant (NetWeaver 2004) with 4 years of experience in SAP BW space.
Table of Contents

Requirement .......................................................................................................................................................3

How BAdi works ................................................................................................................................................3

How Virtual KF/Char works .................................................................................................................................3

Historical Background: .....................................................................................................................................3

Step by step process ..........................................................................................................................................3

Step 1: Create Virtual KF ................................................................................................................................4

Step 2: Including it in Infoprovider ...................................................................................................................4

Step 3: Including it in Query ............................................................................................................................5

Step 4: Create Implementation .......................................................................................................................6

Step 5: Create Impl .........................................................................................................................................6

Step 6: Give name ..........................................................................................................................................6

Step 7: Give Description .................................................................................................................................7

Step 8: Define Filter ........................................................................................................................................7

Step 9: Save Implementation ..........................................................................................................................9

Step 10: Add Attribute ................................................................................................................................10

Step 11: Define Methods ..............................................................................................................................11

Step 12: Write Code ......................................................................................................................................13

ABAP Code for Method: IF_EX_RSR_O Lap_BADI~DEFINE ............................................................................13

ABAP Code for Method: IF_EX_RSR_O Lap_BADI~INITIALIZE ......................................................................14

ABAP Code for Method: IF_EX_RSR_O Lap_BADI~COMPUTE ........................................................................15

Step 13: Check Results ................................................................................................................................16

Related Content ................................................................................................................................................18

Disclaimer and Liability Notice ..........................................................................................................................19
Requirement

We want to add one Keyfigure/Characteristic to query but don’t want to remodel our cube or in other word we
want to populate that KF/Char only at query execution in such a way, which is not possible through
CKF/RKF/Formula. This will make our query more dynamic and will enable us to do flexible calculations at
query runtime.

Here we have a KF called ZVIRTKF; it is an Integer (Decimal type). We want to calculate its values at
runtime as Source System is not giving those values and we have a requirement that we have to calculate it
at Query Runtime only. So we’ll use it as a Virtual KF and will populate it with BAdi.

How BAdi works

In contrast to customer exits, Business Add-Ins no longer assume a two-system infrastructure (SAP and
customers), but instead allow for multiple levels of software development (by SAP, partners, and customers,
and as country versions, industry solutions, and the like). Definitions and implementations of Business Add-
Ins can be created at each level within such a system infrastructure.

The Business Add-In enhancement technique differentiates between enhancements that can only be
implemented once and enhancements that can be used actively by any number of customers at the same
time. BADI's can be used any number of times, where as USER-EXITS can be used only one time.

e.g.: if you are assigning a USER-EXIT to a project in (CMOD), then you cannot assign the same to other
project. But you can use BADI's multiple times as they are Object Oriented Programming based.

In addition, Business Add-Ins can be defined according to filter values. This allows you to control add-in
implementation and make it dependent on specific criteria (on a specific Country value, for example).

How Virtual KF/Char works

A virtual characteristic or a virtual key figure is an InfoObject, which is defined within the InfoProvider as
metadata without having any data stored physically

Historical Background:

The BAdi RSR_OLAP_BADI replaces the user exit RSR00002 with the following advantages:

- Easier to implement
- Different classes for different InfoCubes/Implementations
- No large includes

Virtual Key Figures (KF)/ Characteristics can be used to create the Query more dynamic. The value of Virtual
KF/Characteristics does not store in the Data Target, it is calculated at runtime (At Query Execution time).
Complex logic, database access is possible by using Virtual Key Figure and Characteristics.

When we execute a query containing a virtual KF/Char, system looks for its implementation (using BAdi
Definition RSR_OLAP_BADI) or customer exit (RSR00002 - BI: Virtual Characteristics and Key Figures in
Reporting) at Runtime (as for Variable customer exit RSR00001 - BI: Enhancements for Global Variables in
Reporting). At both places we use filter on Infoprov ider through Case Statement. Through CMOD also we
can do it but BAdi uses OO concept so we'll prefer that.

Step by step process
Step 1: Create Virtual KF
Create virtual KF in RSA1

Step 2: Including it in Infoprovieder
Include it in the Cube/ODS
Step 3: Including it in Query

Include it in the Query through Query Designer.
Step 4: Create Implementation
Go to SE19 Tcode and put RSR_OLP_BADI under Create Implementation → Classical BAdi → BAdi Name

Step 5: Create Impl
Click Create Impl

Step 6: Give name
Give name to your implementation e.g. ZVAR_IMPLT and click OK.
Step 7: Give Description

Give Description to your new implementation.
Step 8: Define Filter

On attribute Tab, give Infoprovider name as filter.
**Step 9: Save Implementation**

Double click on “Name of implementing class:” which is ZCL_IM_VAR_IMPLT at Interface tab, when asked to save, say yes.

Save under package (I used just $TMP Package for demonstration purpose)
Step 10: Add Attribute

Add the attributes to the Implementation Class. The attributes must be of type I and should follow the standard naming convention.

In this example the Name of the implementing class is: ZCL_IM_VAR_IMPLT and add the following attributes.

If it is a key figure use the convention P_KYF_<Key Figure InfoObject> and P_CHA_<Char. InfoObject>. 
Step 11: Define Methods

Add the implementation code. To do this, go to the methods Tab:

Up to this you can follow the steps given in this attached document, but we were not able to get desired result with this so we went for debugging and made some adjustments.

This Badi’s Class Interface consists of three methods:

1. IF_EX_RSR_Olap_Badi~DEFINE
2. IF_EX_RSR_Olap_Badi~INITIALIZE
3. IF_EX_RSR_Olap_Badi~COMPUTE

Have a look at their parameter list so that you can have an idea of the code, where to return and in what format you can return values.

In short, you define the case statement and the cube names under it along with other declarations in Method IF_EX_RSR_Olap_Badi~DEFINE.

If you have to provide further some initial values and other declarations, you can use Method IF_EX_RSR_Olap_Badi~INITIALIZE. (I am not an ABAP’er, so better have a look their parameters and decide accordingly for the Initialize code)

Finally the calculation part take places under 3rd Method IF_EX_RSR_Olap_Badi~COMPUTE.

Have a look at our code but at the same time you can see parameters and modify as per your requirement.
Step 12: Write Code

Double click on each method to write code inside.

**ABAP Code for Method: IF_EX_RSR_Olap_BADI~DEFINE**

```abap
METHOD if_ex_rsr_olap_badi~define.
    CASE i_s_rkb1d-infocube.
        WHEN 'ZTEST_IC'.
            APPEND 'ZVIRTNK' TO c_t_kyfnm.
        ENDCASE.
    ENDCLASS.
ENDMETHOD.
```
Step By Step Process for Virtual Key Figures and Characteristics through BAdi

ABAP Code for Method: IF_EX_RSR_OLAP_BADI~INITIALIZE

```abap
method IF_EX_RSR_OLAP_BADI~INITIALIZE.
endmethod.
```

* No code written here, you can use your code as per the requirement.

**ABAP Code for Method: IF_EX_RSR_OLAP_BADI~COMPUTE**

```abap
METHOD if_ex_rsr_olap_badi~compute.
  FIELD-SYMBOLS <fs_zvirtkf> TYPE ANY.

  p_kyf_zvirtkf = 16.
  ASSIGN COMPONENT p_kyf_zvirtkf OF STRUCTURE c_s_data
    TO <fs_zvirtkf>.
  <fs_zvirtkf> = '10.0'.
ENDMETHOD.
```

Check and Activate.

Note: This is a simple code to put the constant value 10 in this virtual kf. We can also declare table or other data elements in define method to calculate complex logics as well.

`p_kyf_zvirtkf = 16` This statement points to the location of Virtual KF in the `<fs_zvirtkf>` structure, generated at execution time, it is derived by code debugging at runtime. And we are modifying the value of that location by putting constant value 10 in statement `<fs_zvirtkf> = '10.0'`. 
Step 13: Check Results

Run your Query...

Result before putting this BAdi in place:

![Query Result Before BAdi](image1)

Result after putting this BAdi in place:

![Query Result After BAdi](image2)
Step By Step Process for Virtual Key Figures and Characteristics through BAdi

Also check Data at Infoprovider level:

Virtual KF is empty but when you execute query, it gets populated with your BAdi.
Related Content

A new kid(BAdI) in town

Enhancing DataSources with BAdI RSU5_SAPI_BADI

How To Define a New BAdI Within the Enhancement Framework

How to implement a BAdI And How to Use a Filter

How to implement BAdi in Enhancement Framework

BAdI: Customer-Defined Functions in the Formula Builder

Difference between BADI and User Exits

To Use BADI - Business Add In you need to Understand ABAP OO Interface Concept

Using Virtual Key Figure / Characteristics in an InfoSet

For more information, visit the Business Intelligence homepage.
Disclaimer and Liability Notice

This document may discuss sample coding or other information that does not include SAP official interfaces and therefore is not supported by SAP. Changes made based on this information are not supported and can be overwritten during an upgrade.

SAP will not be held liable for any damages caused by using or misusing the information, code or methods suggested in this document, and anyone using these methods does so at his/her own risk.

SAP offers no guarantees and assumes no responsibility or liability of any type with respect to the content of this technical article or code sample, including any liability resulting from incompatibility between the content within this document and the materials and services offered by SAP. You agree that you will not hold, or seek to hold, SAP responsible or liable with respect to the content of this document.