



Enhancing Standard Programs Quickly and Easily Via Business Add-Ins (BADIs):

A Guide for SAP Developers and Consultants

By Rehan Zaidi, Siemens Pakistan

Editor's Note: *Some of the most appealing aspects of SAP® are its modularity and “configure-ability”. SAP’s enterprise solution allows a great deal of flexibility in application, as well as a lot of tailoring to specific company needs. One of SAP’s recent offerings in the tools and techniques arena is the concept of Business Add-Ins, or BADIs. ABAP “man of the hour” Rehan Zaidi uses this white paper to explain exactly what Business Add-Ins are, and how they are called in ABAP programs. He presents an overview of the benefits to consultants and users, and he describes the steps required to enhance standard SAP programs that use this functionality. Maybe you’re no Schwarzenegger, but you can be a world class BADI builder when you exercise your new-found knowledge of Business Add-Ins in SAP.*

Introduction

A major strength of SAP is the capability to adapt its module’s standard programs according to customer-specific requirements. One of the recent customization techniques made available is the use of *Business Add-Ins*. This technique uses newer technology and concepts, and has been used by SAP in most of its modules. Moreover, due to the numerous advantages that it provides, learning the basics of this technique is essential for SAP developers and consultants.

The goal of this white paper is to provide a description of the steps required in adapting programs that use Business Add-Ins. These are some of the questions that this paper will address:

- What are Business Add-Ins and how are they called in ABAP programs?
- What are some of the benefits that they provide to consultants and users?
- What steps are required to enhance standard programs that employ this functionality?

I will begin with an overview of BADIs (Business Add-Ins) and their connection with object-oriented concepts. I’ll discuss, in detail, the steps required in implementing a typical Business Add-In. Finally, I will use an example drawn from an easy-for-all company scenario to illustrate my views. I am going to include tips learned from my personal experience and observation.

This paper is primarily intended for SAP developers but may also be of interest to SAP users. I will assume that the reader is familiar with basic ABAP concepts and has some knowledge of object-oriented programming within an SAP environment. For more information, refer to the SAP documentation on <http://help.sap.com/>. Though the screenshots have been taken from Release 4.6, the tips mentioned are relevant for later releases as well.

Business Add-Ins (BADIs): An Overview

Business Add-Ins may be simply defined as an object-oriented extension of the SAP enhancement technique. They consist of special portions provided by SAP developers for incorporating customer (or company) specific logic. The process of adapting your program according to your scenario is known as *implementation* of the BADI.

Business Add-Ins have been used by SAP developers in a number of modules such as HR, SD, and MM. In addition to standard ABAP programs, you may also use BADIs for menu and screen enhancements. Apart from SAP-delivered applications, you may also want to create your own Business Add-Ins in order to provide the option of enhancing your own ABAP programs. The transactions for creating and implementing Business Add-ins are SE18 and SE19 respectively.



Enhancing Standard Programs Quickly and Easily Via Business Add-Ins (BADIs):

A Guide for SAP Developers and Consultants

As already mentioned, BADIs are based upon the concept of object-orientation. The program that incorporates the enhancement option calls a method of a generated BADI class. During the implementation procedure, the customer-specific code is written in the relevant method. The method name is specified via a BADI interface. The name of the interface is of the form `IF_EX_BADI`, where *BADI* is the name of the Business Add-In in question. For example, in the case of the HR Add-In `HR_INDVAL`, the involved interface is `IF_EX_HR_INDVAL`.

There are two main attributes of Business Add-Ins, namely Multiple-Use and/or Filter Dependent. If you want to allow more than one implementation for a given BADI, the attribute of the corresponding BADI is set as Multiple-use. Likewise, Business Add-ins may also be defined as filter-dependent. This allows you to define subtypes for a given Business Add-In. In this case, a different method code is created and executed for each filter specified in the BADI definition.

BADIs provide a number of advantages to developers and consultants:

- They let you quickly and easily adapt SAP according to your users' requirements, without the need for modifying standard code.
- Since the enhancement is not fixed for all scenarios, BADIs allow you to implement a different application logic for a variety of country versions and company requirements.
- For developers having an affinity for object-oriented concepts, this functionality provides a means of effective SAP program enhancement.

Transaction SE18

You may use the transaction SE18 to display a list of existing BADIs as well as to view the attributes and structure of a given BADI. In order to search for an appropriate BADI in your functional area, call transaction SE18. The main screen for transaction SE18 appears as shown in Figure 1.

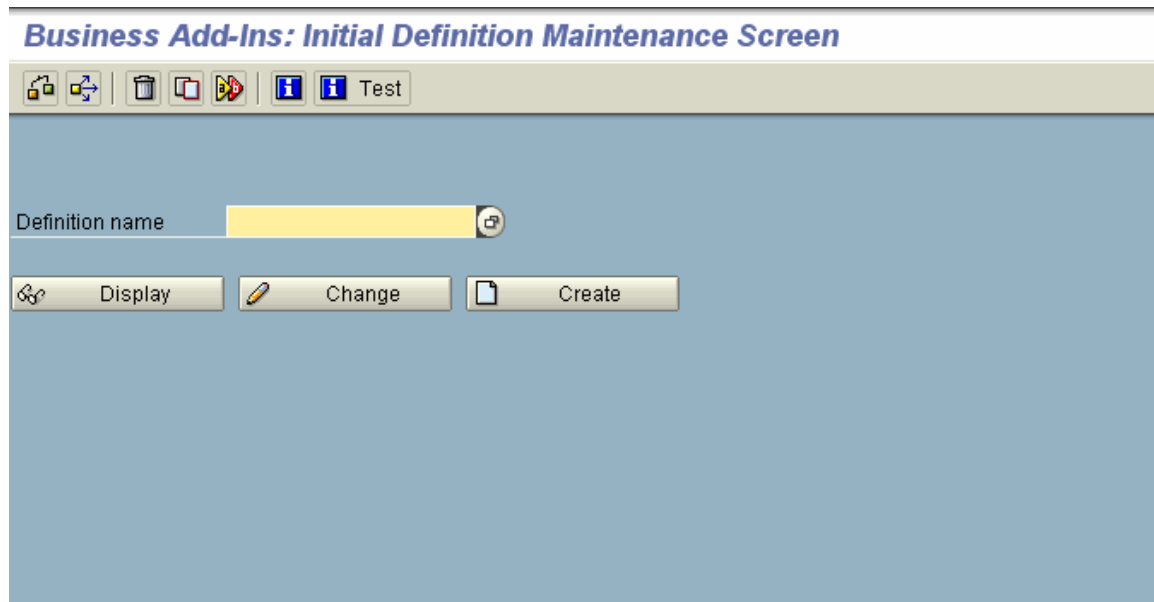


Figure 1: BADI Initial Definition Screen



Enhancing Standard Programs Quickly and Easily Via Business Add-Ins (BADIs):

A Guide for SAP Developers and Consultants

While keeping the cursor on the Definition name field, press the F4 key. A pop-up screen appears. Choose the Application Hierarchy option. The system then displays a tree structure showing the various functional areas (within SAP) pertinent to modules such as MM, SD, and HR. Navigate through this tree in order to find the area (and hence the BADI) in which you are interested.

You also may create new (or display existing) BADI definitions via this transaction. In order to display the attributes and structure of an SAP-provided BADI, enter the BADI name in the field provided on the main screen of transaction SE18, and click Display. The screen appears, as shown in Figure 2.

The screenshot displays the SAP SE18 transaction interface for displaying the definition of BADI HRPAYCA_RP_ROE. The title bar reads "Business Add-Ins: Display Definition HRPAYCA_RP_ROE". The main area is divided into several sections:

- Definition name:** HRPAYCA_RP_ROE
- Definition short text:** Business add-in for Record of Employment (Canada)
- Attributes tab:** Contains sub-tabs for "Attributes", "Interface", and "FCodes".
- General data section:**
 - Development class: PC07
 - Language: EN English
 - Last changed by: KAUL
 - Last change: 14.02.2002 20:59:04
 - Name of bus. add-in class: CL_EX_HRPAYCA_RP_ROE
- Type section:**
 - Multiple use
 - Filter dependent
 - Filter type: [Empty field]
 - Extendable

Figure 2: Displaying Definition of BADI HRPAYCA_RP_ROE

The Attributes tab shows the General Data and the Type of the BADI in question. The General Data shows the relevant Business Add-In class, whereas the Type specifies whether the Add-In is Multiple Use and/or Filter-Dependent. In case the Filter-Dependent indicator is on, a relevant Filter Type is displayed in the field provided. You may then click on the Interface tab. The name of the Interface and the BADI method is displayed (see Figure 3).



Enhancing Standard Programs Quickly and Easily Via Business Add-Ins (BADIs):

A Guide for SAP Developers and Consultants

Business Add-Ins: Display Definition HRPAYCA_RP_ROE

Definition name: HRPAYCA_RP_ROE
Definition short text: Business add-in for Record of Employment (Canada)

Attributes | Interface | FCodes

Interface name: IF_EX_HRPAYCA_RP_ROE

Method	Description
SAP_SCRIPT_TABLES	Export/import SAPSCRIPT tables

Figure 3: BADI Interface and Method Name

Double-click on the Method name in order to view the details about the importing, exporting, and changing parameters of the BADI method in question (see Figure 4).

Class Builder: Display Interface IF_EX_HRPAYCA_RP_ROE

Interface: IF_EX_HRPAYCA_RP_ROE (Implemented / Active)

Properties | Interfaces | Attributes | Methods | Events | Aliases

Method parameters: SAP_SCRIPT_TABLES

Parameter	Type	P...	O...	Typing	Reference type	Default value	Description
PERNR	Importing	<input type="checkbox"/>	<input type="checkbox"/>	Type	PERNR - PERNR		Personnel number
ROE_RUN_DATE	Importing	<input type="checkbox"/>	<input type="checkbox"/>	Type	SY-DATUM		ROE run date
FIRST_DAY_WORKED	Importing	<input type="checkbox"/>	<input type="checkbox"/>	Type	SY-DATUM		First day worked
LAST_DAY_WORKED	Importing	<input type="checkbox"/>	<input type="checkbox"/>	Type	SY-DATUM		Last day worked
LAST_DAY_PAID	Importing	<input type="checkbox"/>	<input type="checkbox"/>	Type	SY-DATUM		Last day for which paid
SAP_SCRIPT_CHAR	Changing	<input type="checkbox"/>	<input type="checkbox"/>	Type	HRPAYCA_ROE_SAPSCRIPT_CHAR		SAP_SCRIPT_CHAR
SAP_SCRIPT_TXT	Changing	<input type="checkbox"/>	<input type="checkbox"/>	Type	HRPAYCA_ROE_SAPSCRIPT_TXT		SAP_SCRIPT_TXT
SAP_SCRIPT_DATES	Changing	<input type="checkbox"/>	<input type="checkbox"/>	Type	HRPAYCA_ROE_SAPSCRIPT_DATES		SAP_SCRIPT_DATES
SAP_SCRIPT_AMT	Changing	<input type="checkbox"/>	<input type="checkbox"/>	Type	HRPAYCA_ROE_SAPSCRIPT_AMT		SAP_SCRIPT_AMT
RETURN_CODE	Changing	<input type="checkbox"/>	<input type="checkbox"/>	Type	SY-SUBRC		Return code
		<input type="checkbox"/>	<input type="checkbox"/>	Type			
		<input type="checkbox"/>	<input type="checkbox"/>	Type			
		<input type="checkbox"/>	<input type="checkbox"/>	Type			

Figure 4: Parameters of a Given BADI Method



Enhancing Standard Programs Quickly and Easily Via Business Add-Ins (BADIs):

A Guide for SAP Developers and Consultants

In case of Filter-Dependent BADIs, one important parameter that is passed on to the method is the filter value. The name of this parameter, in most cases, is FLT_VAL.

Structure of Programs that Employ the BADI Functionality

It is a good idea for you to familiarize yourself with the structure of programs that employ Business Add-Ins. The programs (whether standard or custom-built) that incorporate the BADI functionality include a somewhat common block of code. One such example is shown in Figure.5

```
.....  
.....  
CLASS CL_EXITHANDLER DEFINITION LOAD.  
DATA MYEXIT TYPE REF TO IF_EX_BADINAME.  
  
CALL METHOD CL_EXITHANDLER=>GET_INSTANCE “ fetching instance of the  
    CHANGING INSTANCE = MYEXIT.      “ BADI class  
  
CALL METHOD MYEXIT->BADIMETHOD      “ calling the BADI method  
    EXPORTING....  
    .....  
    IMPORTING....  
.....  
.....
```

Figure 5: BADI Method Call

The block of code first declares the class CL_EXITHANDLER, and then declares a reference variable (in this case, MYEXIT) to the interface of the BADI in question. Then, the program calls the static method GET_INSTANCE of the CL_EXITHANDLER class. This method is used to access an active instance of the Business Add-In class, which is placed in the declared variable MYEXIT.

Tip: One of the ways of finding out whether a program supports BADI enhancement is to search for the text “EXITHANDLER” in the program’s source code.

The instance of the BADI class may then be used to call the implemented BADI method. Finally, the BADI method is called. This method contains the actual enhancement logic pertinent to the customer or the country in question (we’ll discuss this in detail in the next section).

Steps Required in Implementing a BADI

As already mentioned, you may create new (or amend existing) implementations of Business Add-Ins via transaction SE19. There are a few steps required in order to implement a Business Add-In. Let us go through them one by one.

Step 1: Creating an Implementation

The first step involves creating a BADI implementation. Call transaction SE19. The BADI implementation screen appears, as shown in Figure 6.



Enhancing Standard Programs Quickly and Easily Via Business Add-Ins (BADIs):

A Guide for SAP Developers and Consultants

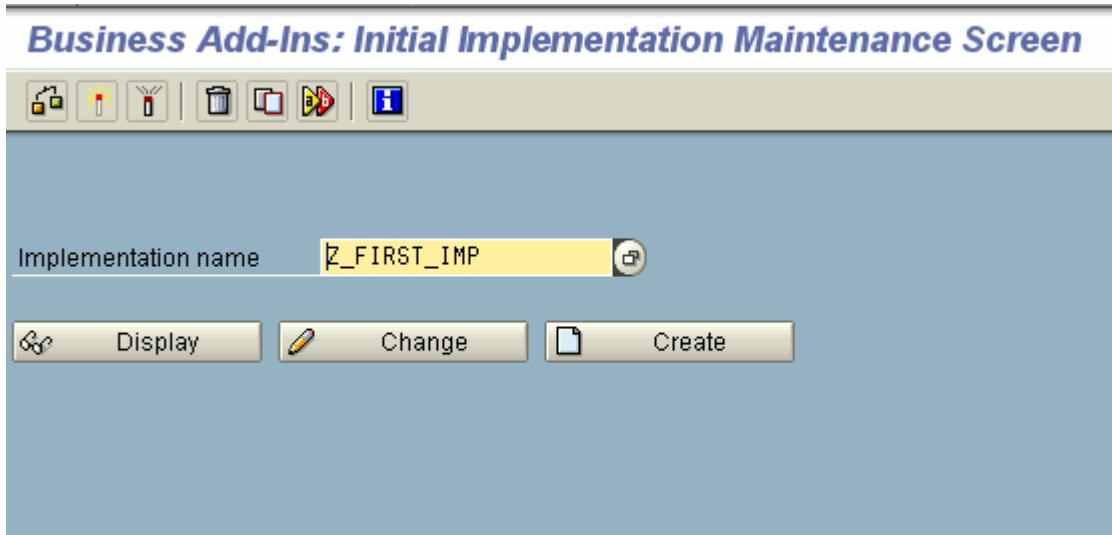


Figure 6: Transaction SE19 – BADI Implementation Screen

Enter a suitable name for your implementation in the field provided, and click the Create button.

A pop-up screen appears, as shown in Figure 7. Enter the name of the BADI involved and press the Enter button.

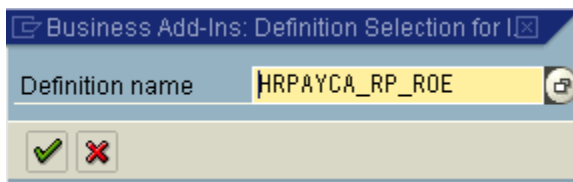


Figure 7: Entering the Name of the BADI to Be Implemented

This leads you to the screen shown in Figure 8.



Enhancing Standard Programs Quickly and Easily Via Business Add-Ins (BADIs):

A Guide for SAP Developers and Consultants

Business Add-In Builder: Change Implementation Z_FIRST_IMP

Def. documentatn | Documentation

Implementation name: Z_FIRST_IMP Inactive
Implementation short text: My first BAdI Implementation
Definition name: HRPAYCA_RP_ROE

Attributes | Interface

Interface name: IF_EX_HRPAYCA_RP_ROE
Name of implementing class: ZCL_IM__FIRST_IMP

Method	Description
SAP_SCRIPT_TABLES	Export/import SAPSCRIPT tables

Figure 8: BADI Implementation Maintenance

Enter an appropriate short text in the field provided. Then, click on the Interface tab. This shows the name of the class that will be generated as a result of the implementation. You may change the class if you like. The Interface tab also contains the name of the BADI method.

Note: In case you are implementing a Filter-Dependent BADI, you need to enter a suitable filter value in the table control provided in the Type portion of the Attributes tab. The Filter Value(s) field (in this case) is available for input, as shown in Figure 9.

Type

Multiple use

Filter dependent Filter type: PADIV_MODULE Extendable

Indirect evaluation module

Filter value(s)

Filter val.	Shor...
Z123	

Figure 9: Entering a Filter Value for Filter Dependent BADIs



Enhancing Standard Programs Quickly and Easily Via Business Add-Ins (BADIs):

A Guide for SAP Developers and Consultants

Then, double-click on the name of the method (in our case SAP_SCRIPT_TABLES). This takes you to the Class Builder's method editor screen. This is the area where you may write the code that you would like to be executed when the BADI method is called (see Figure 10).

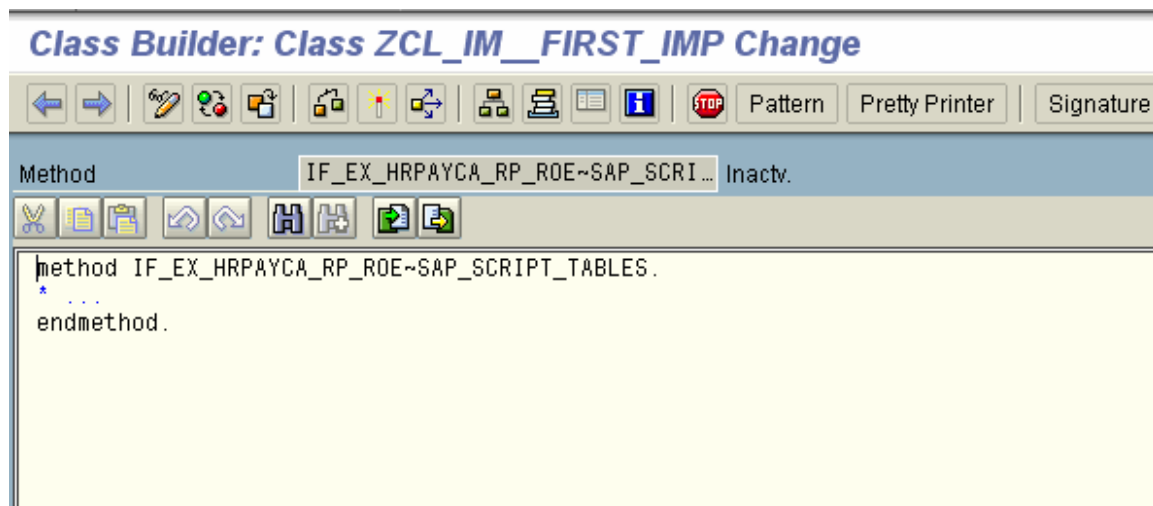


Figure 10: Coding the BADI Method

Step 2: Writing the Code for the BADI Method

The next step is to write the appropriate coding for the BADI method. This code incorporates the enhancement logic and is executed by the application program upon the BADI method call. Most of the ABAP statements are applicable in this case. However, since the BADI technology is based upon ABAP Objects, some ABAP constructs are not allowed. For example, the following form for internal table declaration will give an error:

```
DATA : BEGIN OF ITAB2 OCCURS 0,  
        MYFIELD1,  
        MYFIELD2,  
        END OF ITAB2.
```

As already mentioned, the method has importing, exporting, and changing parameters. The enhancement may be achieved by writing code that assigns suitable values to the changing and exporting parameters of the method. The main application program uses these values for further processing, and in this way the desired enhancement effect is achieved. When you are done with the coding, activate the BADI method.

Tip: Effective BADI implementation lies in using the parameters in the best possible manner. Take some time to explore the various parameters provided and use them in best fulfilling your scenario requirements.

After carrying out the two necessary steps, activate your BADI implementation.



Enhancing Standard Programs Quickly and Easily Via Business Add-Ins (BADIs):

A Guide for SAP Developers and Consultants

Putting It All Together

In this section, I will use the concepts described in this paper to show how the BADI functionality may be applied for indirectly valuating HR allowances. The formula or criteria for indirect valuation, known as Module, is specified via configuration settings. However, the code for writing the mathematical formula is done in a BADI method.

The Business Add-In applicable in this case is HR_INDVAL, and the interface involved is IF_EX_HR_INDVAL. As already mentioned, the first step is to create the implementation. An implementation by the name Z_FIRST_IMP1 was created, as shown in Figure 11.

Business Add-In Builder: Change Implementation Z_FIRST_IMP1

Def. documentatn | Documentation

Implementation name: Z_FIRST_IMP1 Inactive

Implementation short text: HR Indirect Valuation

Definition name: HR_INDVAL

Attributes | Interface

General data

Development class: []

Language: EN English

Last changed by: []

Last change: [] 00:00:00

Type

Multiple use

Filter dependent Filter type: PADIV_MODULE Indirect evaluation module Extendable

Filter value(s)

Filter val.	Shor...
Z123	[]

Figure 11: Creating an Implementation for Business Add-In HR_INDVAL

Since HR_INDVAL is a filter-dependent BADI, we need to specify an appropriate Filter Value in the Attributes section. The Filter Value, in this case, is the name of the Indirect Valuation Module (in our case Z123). For simplicity's sake, I have assumed that the indirect valuation module Z123



Enhancing Standard Programs Quickly and Easily Via Business Add-Ins (BADIs):

A Guide for SAP Developers and Consultants

has already been created. Next, the method on the Interface tab is accessed. In this case, the name of the method is DO_INDIRECT_VALUATION (see Figure 12).

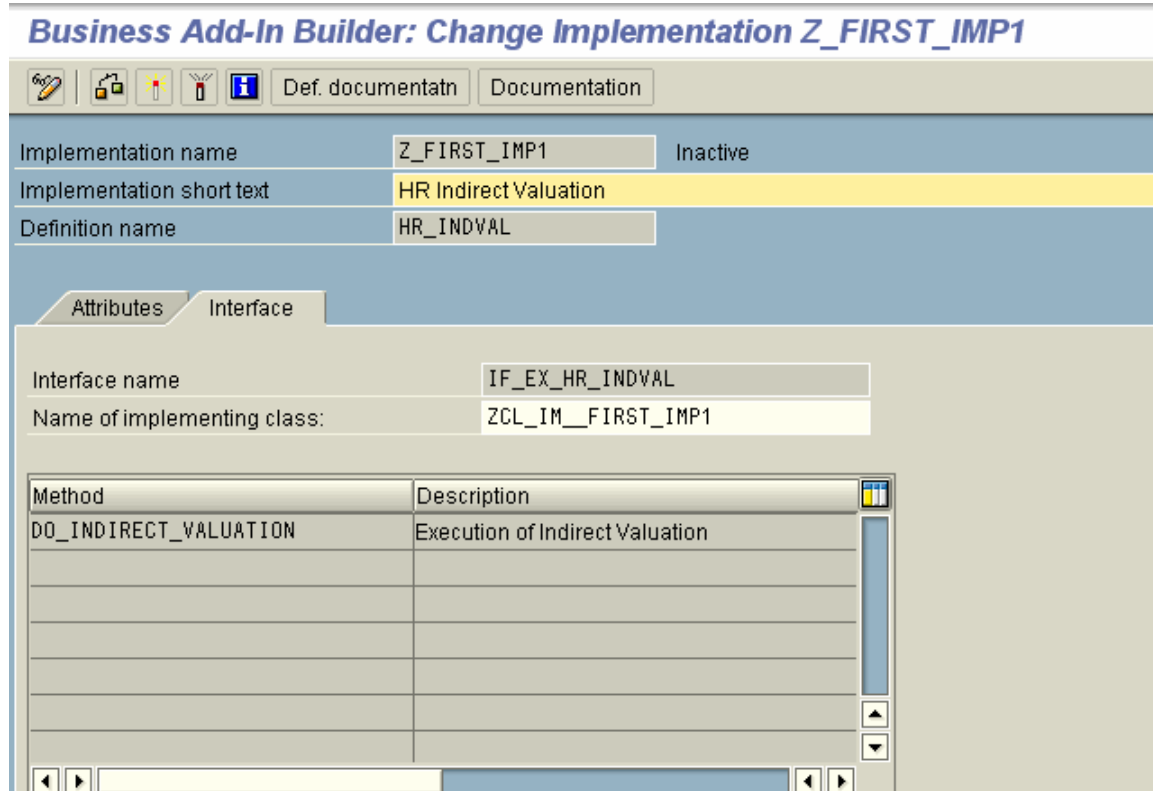


Figure 12: BADI HR_INDVAL's Interface

The important parameters in this case are the VALUATION_INPUT and VALUATION_OUTPUT variables. The appropriate code is written for the method DO_INDIRECT_VALUATION. When the user creates entries for allowances that have been assigned the given module, the formula specified in BADI method DO_INDIRECT_VALUATION is executed, and the resulting allowance value is displayed on the screen.

Conclusion

In this white paper, I reviewed the rudiments of Business Add-Ins and their benefits for SAP users and developers. Then, I discussed in detail the steps required in implementing an add-in. Finally, I discussed an HR requirement that may be fulfilled via the use of this functionality. I hope this paper will provide you with precious insights to help you in adapting standard SAP programs quickly and easily.

Rehan Zaidi, Senior SAP Consultant, Siemens Pakistan

Rehan has been involved in both ABAP development and functional configuration for SAP HR implementations at multinational and local companies. He has also contributed articles to the SAP Professional Journal, the HR Expert newsletter, and to the TechRepublic Web site. He is currently working on his first book, which is specifically designed for SAP HR Users and Managers (planned for release in the beginning of 2006). Rehan's email address is Rehan.Zaidi@SAPtips.com or zaidi.rehan@gmail.com.



Enhancing Standard Programs Quickly and Easily Via Business Add-Ins (BADIs):

A Guide for SAP Developers and Consultants

The information in our publications and on our Website is the copyrighted work of Klee Associates, Inc. and is owned by Klee Associates, Inc.

NO WARRANTY: This documentation is delivered as is, and Klee Associates, Inc. makes no warranty as to its accuracy or use. Any use of this documentation is at the risk of the user. Although we make every good faith effort to ensure accuracy, this document may include technical or other inaccuracies or typographical errors. Klee Associates, Inc. reserves the right to make changes without prior notice.

NO AFFILIATION: Klee Associates, Inc. and this publication are not affiliated with or endorsed by SAP AG. SAP AG software referenced on this site is furnished under license agreements between SAP AG and its customers and can be used only within the terms of such agreements. SAP AG and mySAP are registered trademarks of SAP AG.

All other company and product names used herein may be trademarks or registered trademarks of their respective owners.