Implementing a BAdI in an Enhancement Project (CMOD)

SAP DEVELOPER NETWORK

Applies To:

SAP R3 v4.70, however can be adapted for other releases, including Netweaver 2004 v7.

Summary

This tutorial explains how to implement a Business Add In (BAdI), in a Customer Modification – CMOD, environment and benefit from all the flexibility of the BADI, while avoiding all the limitations associated with CMOD.

By: Glen Spalding

Company: **gin**gle

Date: 10 February 2006

Implementing a BAdI in an Enhancement Project (CMOD)

SAP DEVELOPER NETWORK

Table of Contents

Implementing a BAdI in an Enhancement Project (CMOD)

SAP DEVELOPER NETWORK

Enhancements

SAP provides a variety of “Enhancements” for functional enrichment.

Enhancements prior to Netweaver 2004 exist in a number of guises – User Exits, Customer Exits, Menu/Screen/Field Exits, and BADIs. Each has their own manner of implementation.

This topic is concerned with User Exits and Customer Exits.

Enhancements are scattered throughout the SAP System at predefined strategic points, where changes could prove beneficial without jeopardizing the integrity of the surrounding functionality.

You may be lucky to find such an Enhancement right where you need it however there are times when the Enhancement needs to be shared among other developments.

Historic Problem

Because User and Customer Exits are implemented via code, it can become very cumbersome when the Exit is in a popular location and used for many purposes or requirements, especially if it is shared throughout developments.

An example of this could be when using an “IDoc Extension”. The extended SAP IDoc could be required by a number of departments, each attempting to develop their own specific functionality, and the location to populate or read the customized segments of the IDoc exists in a single area of code as either a “Function Module” or "Include”.

It becomes even more complicated when each developer shares the coded area and has to manage “Transports” of shared objects amongst them. Not to mention the trouble caused when a syntax or program error occurs across a shared object affecting all who are using it.

Yet another problem exists when a developer has completed their work, and is ready to “Release” the associated Transport, which contains the shared code, and all remaining that use, program, or develop that object are not ready.

A final runtime problem exists when each piece of functionality within an Enhancement must be implemented and run as a whole. Enhancements must exist in a “Project” when they are implemented – see transaction CMOD. One or more Enhancements can exist in a Project. But, if an Enhancement exists in a Project, it cannot coexist in another Project.

The Project is “Activated” or “Deactivated”, i.e. on or off, not the individual Enhancements, therefore anything in the Project must be deemed as an “All or Nothing”.

Needless to say, User and Customer Exits can require very careful management.

Solution

To address the problem above, I propose a simple and elegant way: -

Implement a custom BAdI inside the Enhancement

The Enhancement can then benefit from discrete pieces of work and remain detached and independent from others that may share the Enhancement.

Implementing a BAdI in an Enhancement Project (CMOD)

The initial setup for implementing the BAdI into the Enhancement is best performed away from any other piece of work. This way, it too, remains separate and in its own Transport that must be Released prior to all other surrounding work that maybe required.

Example

I have chosen a scenario, “Maintain Company Address” that should exist in all SAP versions concerned, including the Netweaver Sneak Preview 2004s, and I hope all other training mini versions of SAP.

Transaction SUCOMP.

SAP Menu

Tools -> Administration -> User Maintenance -> SU01 Users

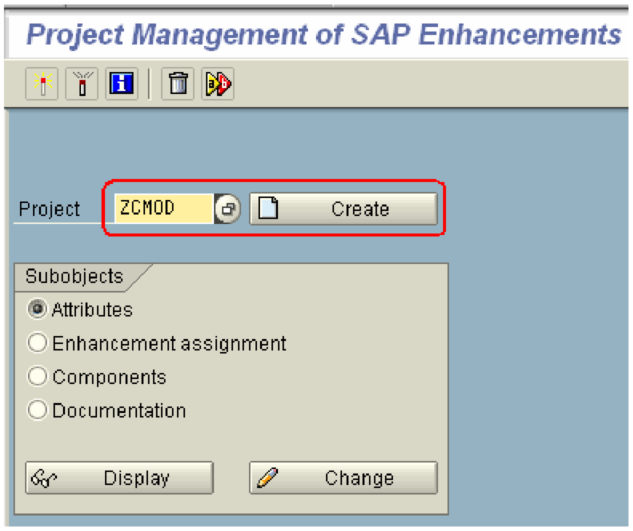
Environment -> Maintain company address

We will simply edit a newly created Company Address, and implement the related SAP Enhancement SZSR0003.

We will program two separate Popup Dialogs to act as our multiple areas of custom functionality. The Popups will be programmed in their own BAdIs. They will be “fired” when the user, for example, changes the “Time Zone” of the Customer Address” and presses the Enter key.

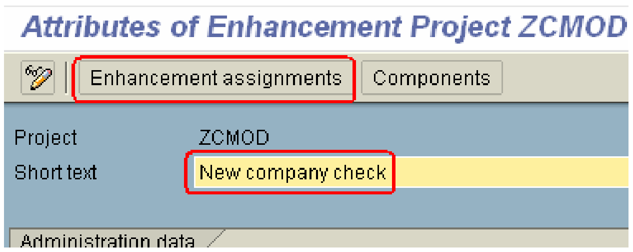
Create an Active Project with Enhancement Assigned

To begin, we need to create a Project to implement the Enhancement. Transaction CMOD.


Project Management of SAP Enhancements

Project ZCMOD

Give the “Project” name, and press the Create button.

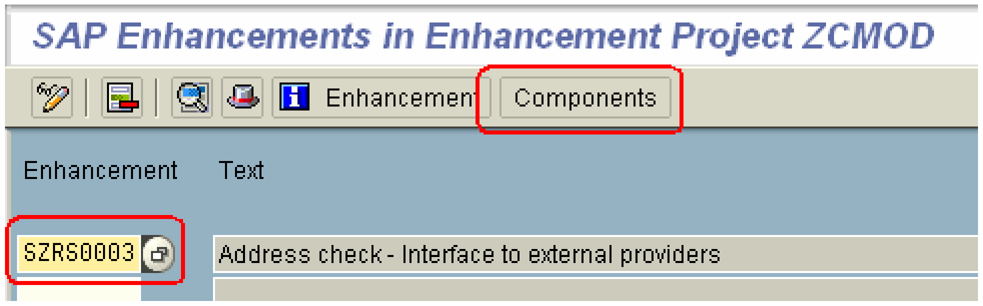

Attributes of Enhancement Project ZCMOD

Project ZCMOD
Short text New company check

Give the Project a Description and Save. When prompted for a “Package”, simply choose “Local Object”. The “Project” now exists and is empty.

Now we need to add our Enhancement, so choose the “Enhancement assignments” button.

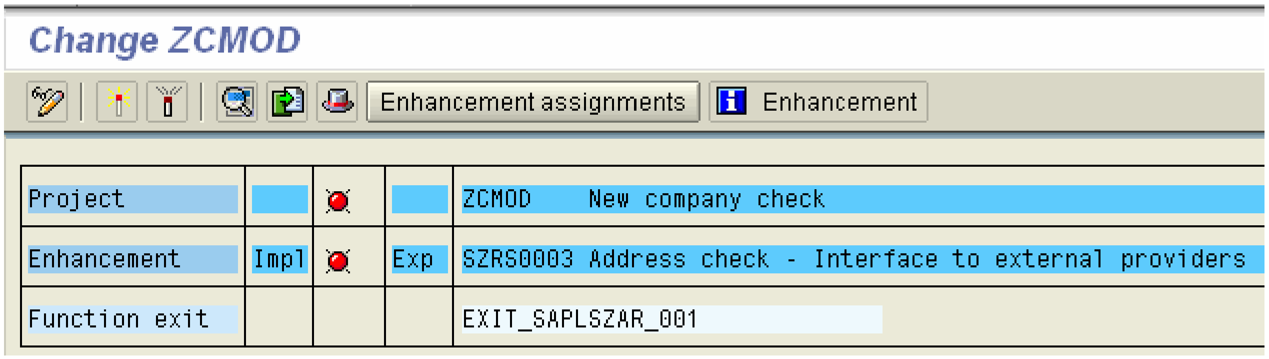
Add the Enhancement “SZRS0003” and Save. Now press the “Components” button to see the possible modifications for this Enhancement.



SAP Enhancements in Enhancement Project ZCMOD

Enhancement	Text
SZRS0003	Address check - Interface to external providers

From the component list, seen below, we can see we have a single Function Exit. We can also see here, that the Project is has yet to be “Activated”. This is indicated by the red Activation icons, which turn green when active.



Change ZCMOD

Project		●		ZCMOD New company check
Enhancement	Impl	●	Exp	SZRS0003 Address check - Interface to external providers
Function exit				EXIT_SAPLSZAR_001

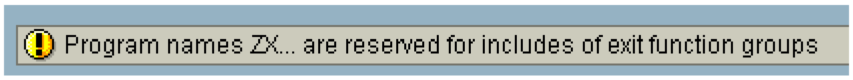
Double click the “EXIT_SAPLSZAR_001” entry.

This navigates us to the Function Builder – SE37, in which we can see there is a single line inside the Function.

```
include zxszaru01 .
```

As you can see, the Include begins with a Z, therefore it is in the Customer Namespace, which enables us to maintain our custom Exit code separately from the SAP code.

Double click the Include name. A warning may appear.

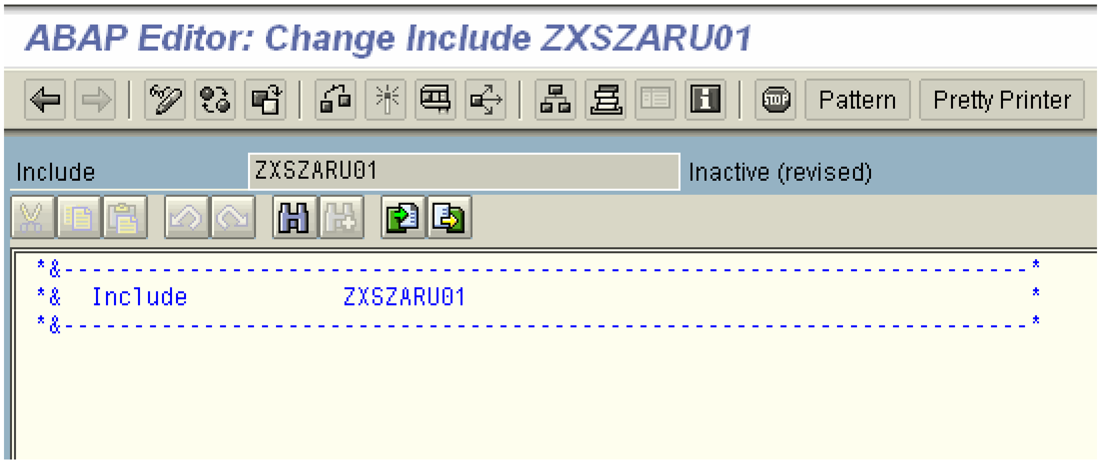


Program names ZX... are reserved for includes of exit function groups

This is fine. Press enter to continue.

A “Create Object” popup may appear, in which we need to press “Yes” to create the Include Object. Again, create the object as a “Local Object”, when prompted.

Now we are navigated to the ABAP Editor– SE38, where we can program the Enhancement.



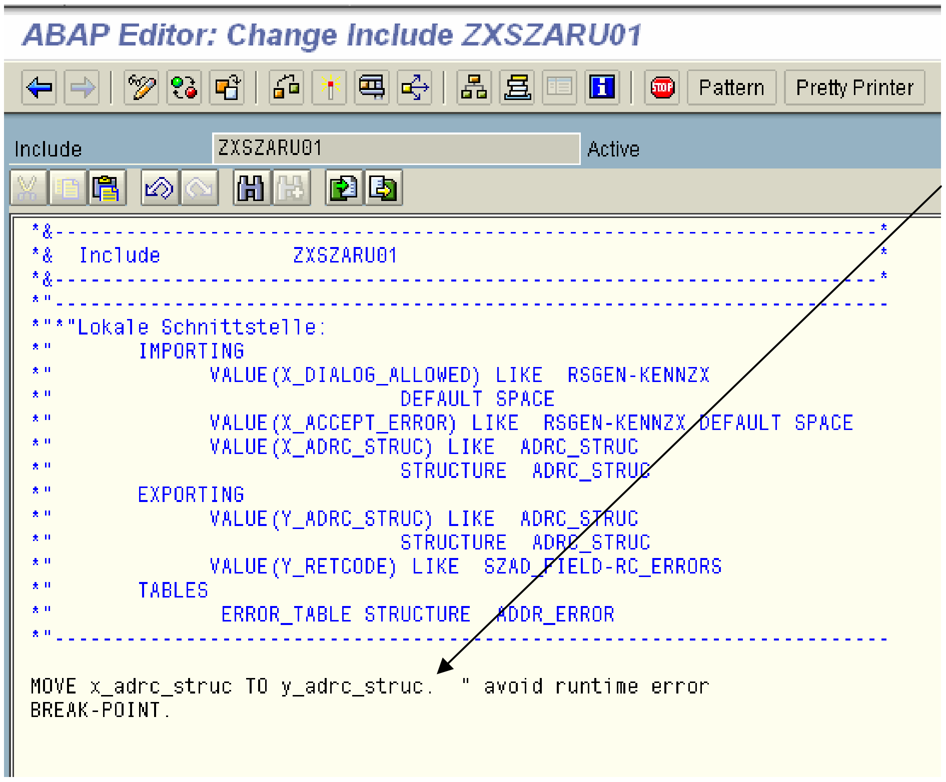
```
*&---------------------------------------------------------------------*
*&  Include           ZXSZARU01                                        *
*&---------------------------------------------------------------------*
```

It is in this type of Include where all necessary programming will need to be programmed or at least, called from. The single Include, in this example, ZXSZARU01 would be the main cause of problems when managing multiple developments.

For now, we will program a break point and watch it reached.

Add the statement “BREAK-POINT.

Notice, I have copied the parameters from the previous screen to enable me to see, what variables are present in this include that I can use.

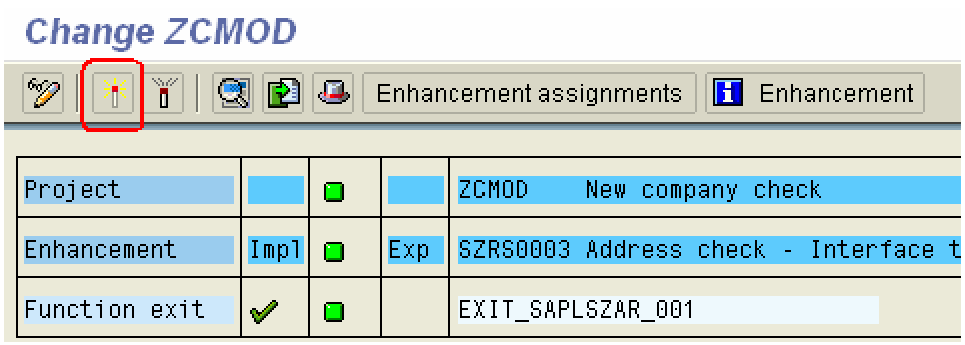


```
*&---------------------------------------------------------------------*
*&  Include           ZXSZARU01                                        *
*&---------------------------------------------------------------------*
*"----------------------------------------------------------------------
*"*"Lokale Schnittstelle:
*"       IMPORTING
*"             VALUE(X_DIALOG_ALLOWED) LIKE  RSGEN-KENNZX
*"                             DEFAULT SPACE
*"             VALUE(X_ACCEPT_ERROR) LIKE  RSGEN-KENNZX DEFAULT SPACE
*"             VALUE(X_ADRC_STRUC) LIKE  ADRC_STRUC
*"                             STRUCTURE  ADRC_STRUC
*"       EXPORTING
*"             VALUE(Y_ADRC_STRUC) LIKE  ADRC_STRUC
*"                             STRUCTURE  ADRC_STRUC
*"             VALUE(Y_RETCODE) LIKE  SZAD_FIELD-RC_ERRORS
*"       TABLES
*"              ERROR_TABLE STRUCTURE  ADDR_ERROR
*"----------------------------------------------------------------------

MOVE x_adrc_struc TO y_adrc_struc.  " avoid runtime error
BREAK-POINT.
```

Also, I have added a MOVE statement simply transferring what is being passed into the Function Module, back out. This avoids a runtime error in this example.

Each implementation of an Enhancement is likely to have its own interface and may require certain coding requirements, similar to this. However, in this case it is just a side issue and should not detract from the point of this paper.

Now "Syntax Check" and "Activate". Back out, to the Project Screen and Activate the Project. If this is not done, our custom code will not be executed. This is Activating the Project, not the code.

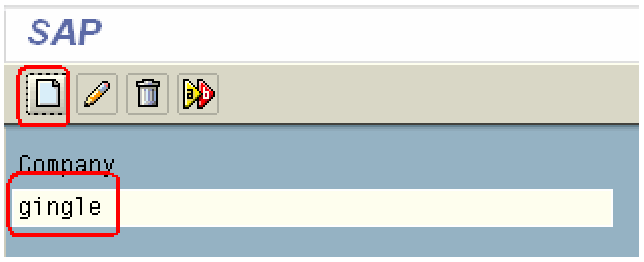


Project		🟩		ZCMOD New company check
Enhancement	Impl	🟩	Exp	SZRS0003 Address check - Interface t
Function exit	✓	🟩		EXIT_SAPLSZAR_001

The red Activation icons have now turned green. We are ready to test the Exit and our Break Point.

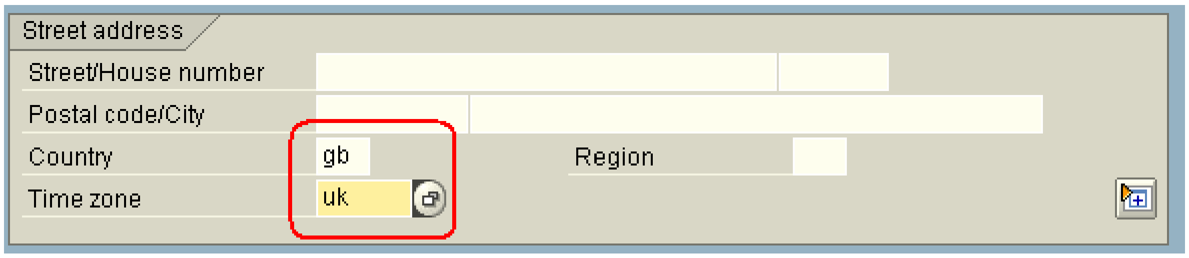
Test Enhancement

Navigate to the Company Address Maintenance Screen – SUCOMP, and enter a Company Name, then press Create.



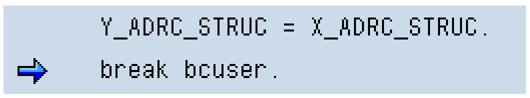
Company
gingle

Enter the remaining mandatory fields, “Country” and “Time Zone” then press Enter


Street address

Street/House number	
Postal code/City	
Country	gb
Region	
Time zone	uk

The Exit should have been entered and the code halted in the Debugger due to our break point statement in code.



```
    Y_ADRC_STRUC = X_ADRC_STRUC.
⇨   break bcuser.
```

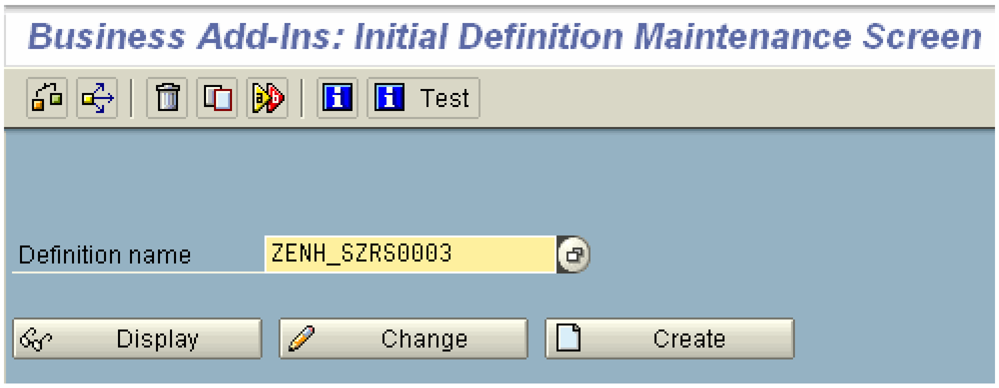
Simply press the F8 key to continue, and Save.

This has proved that we now have a working Active Exit in place. For future testing will be using this company in edit mode, and simply be alternating the Time Zone between UK and UTC.

Create BAdI Definition

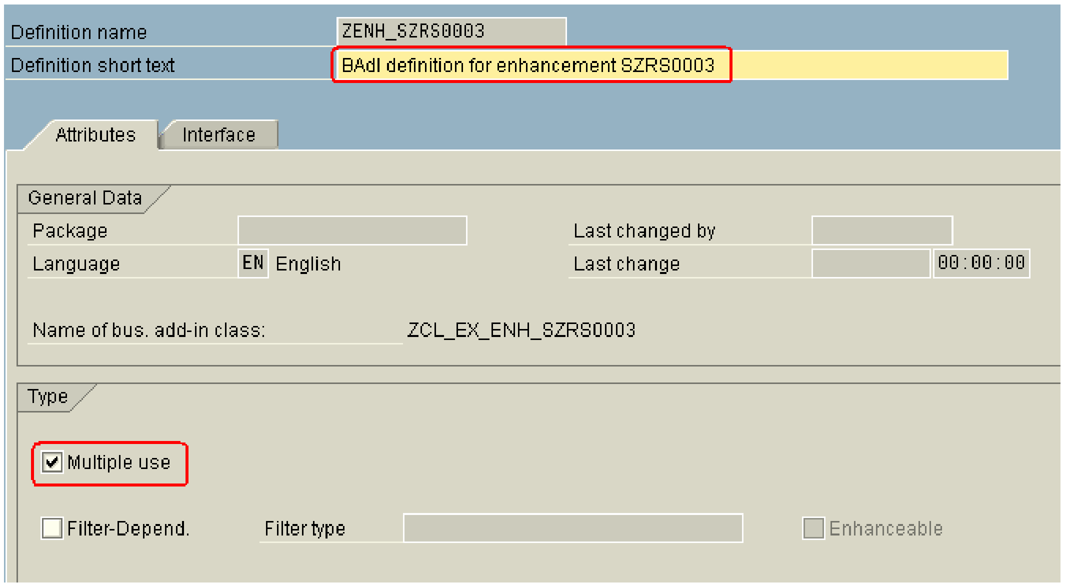
Now we need to create a custom BAdI Definition that is going to be used for our two BAdI Implementations. The custom BAdI's interface will mirror the Enhancement's interface as the Enhancement will serve purely as a wrapper for the BAdI.

Navigate to the BAdI Definition – SE18. Create a BAdI name and press Create.



Definition name: ZENH_SZRS0003

Provide a short text, and make sure the "Multiple Use", option for "Type", is checked. It is this that enables the multiple implementations for a single definition.

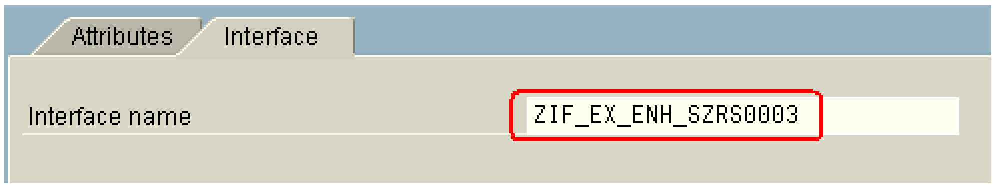


Definition name: ZENH_SZRS0003
Definition short text: BAdI definition for enhancement SZRS0003

Package:
Language: EN English
Last changed by:
Last change: 00:00:00
Name of bus. add-in class: ZCL_EX_ENH_SZRS0003

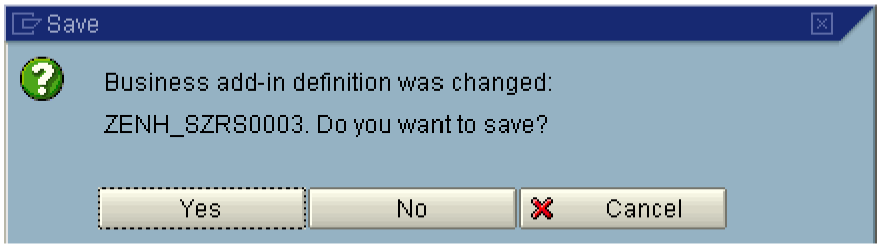
☑ Multiple use
☐ Filter-Depend.
Filter type:
☐ Enhanceable

On the "Interface" tab, double click the provided "Interface Name".



Interface name: ZIF_EX_ENH_SZRS0003

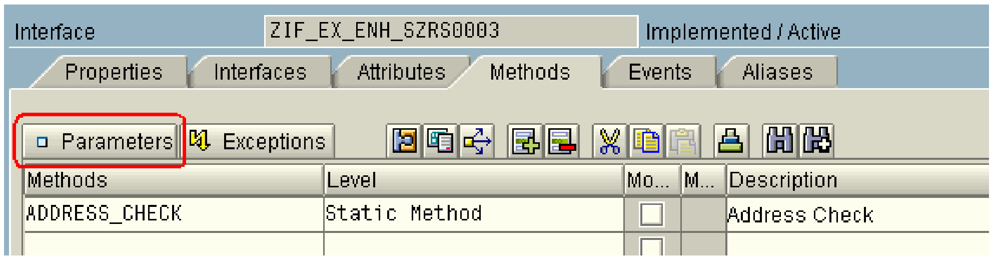
When prompted, Save the BAdI definition, a “Local object” will suffice.



Business add-in definition was changed:
ZENH_SZRS0003. Do you want to save?

We are now in the Class Builder – SE24, in which we have a Class Interface to create. This Class Interface will serve the purpose of providing the BAdI Definition with a Method Call, and its appropriate interface parameters. To this end, we must provide a Method Name and configure its interface parameters, so that the Enhancement can server as a wrapper, passing it’s parameters in and out correctly.

Enter a Method name and for this purpose, make it a Static Method, then press the “Parameters” button.

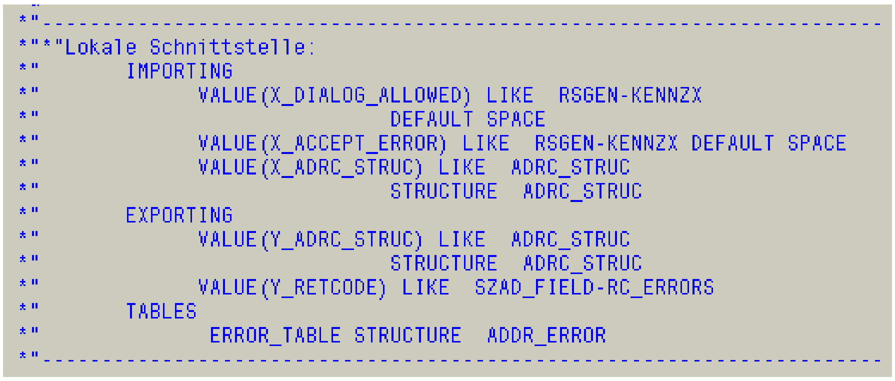


Interface ZIF_EX_ENH_SZRS0003 Implemented / Active

Methods	Level	Mo...	M...	Description
ADDRESS_CHECK	Static Method	☐		Address Check

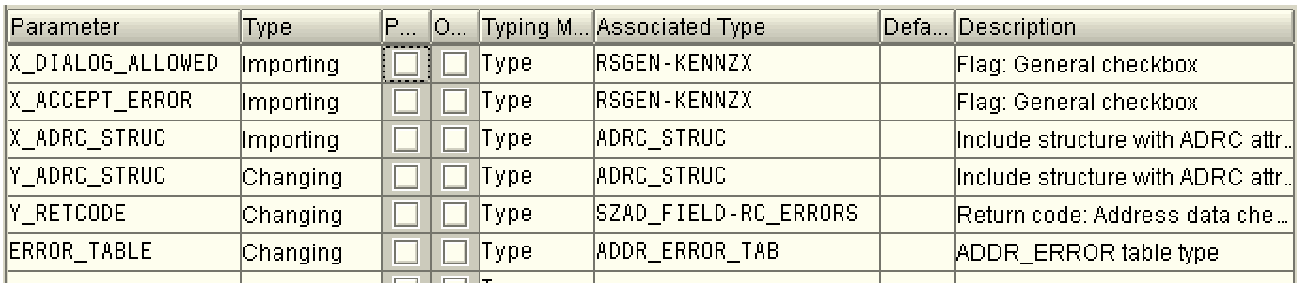
The idea here, is to replicate the Enhancement’s interface in the Method, enabling a somewhat transparent handover. Remember, the Enhancement needs to only serve as a wrapper.

From



```
*"----------------------------------------------------------------------
*"*"Lokale Schnittstelle:
*"       IMPORTING
*"             VALUE(X_DIALOG_ALLOWED) LIKE  RSGEN-KENNZX
*"                             DEFAULT SPACE
*"             VALUE(X_ACCEPT_ERROR) LIKE  RSGEN-KENNZX DEFAULT SPACE
*"             VALUE(X_ADRC_STRUC) LIKE  ADRC_STRUC
*"                             STRUCTURE  ADRC_STRUC
*"       EXPORTING
*"             VALUE(Y_ADRC_STRUC) LIKE  ADRC_STRUC
*"                             STRUCTURE  ADRC_STRUC
*"             VALUE(Y_RETCODE) LIKE  SZAD_FIELD-RC_ERRORS
*"       TABLES
*"              ERROR_TABLE STRUCTURE  ADDR_ERROR
*"----------------------------------------------------------------------
```

To



Parameter	Type	P...	O...	Typing M...	Associated Type	Defa...	Description
X_DIALOG_ALLOWED	Importing	☐	☐	Type	RSGEN-KENNZX		Flag: General checkbox
X_ACCEPT_ERROR	Importing	☐	☐	Type	RSGEN-KENNZX		Flag: General checkbox
X_ADRC_STRUC	Importing	☐	☐	Type	ADRC_STRUC		Include structure with ADRC attr...
Y_ADRC_STRUC	Changing	☐	☐	Type	ADRC_STRUC		Include structure with ADRC attr...
Y_RETCODE	Changing	☐	☐	Type	SZAD_FIELD-RC_ERRORS		Return code: Address data che...
ERROR_TABLE	Changing	☐	☐	Type	ADDR_ERROR_TAB		ADDR_ERROR table type

With “Multiple Use” type BAdIs, we must not specify any Export type parameters. Therefore, as we have Export Parameters in the Enhancement, we have altered them to Changing.

Also, notice the ERROR_TABLE parameter. In the Enhancement it exists as a table, structure ADDR_ERROR. Due to the nature of the ABAP Objects, table definitions must be typed as tables. Therefore we either find a table type that is made up from the structure ADDR_ERROR, as we have, or have to create one.

Perform a Syntax Check, Save and Generate.

That concludes our BAdI definition.

BAdI Hook

We must now code the BAdI hook. This is not coding any functional specific requirements whatsoever. It is merely enabling the Enhancement to call the BAdI, if, and when it may be deployed. This hook is to be coded in the original Enhancement Include – CMOD.

Due to the Type change of the ERROR_TABLE parameter in the BAdI, we need to perform a little extra work.

Variable declaration

```
y_error_table TYPE addr_error_tab
```

Line of code

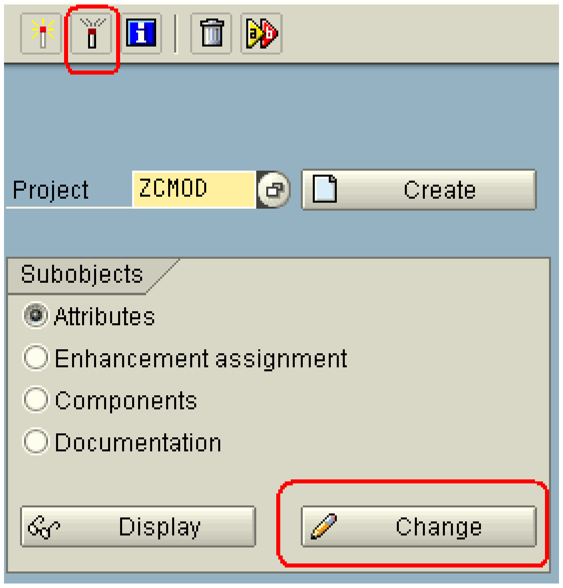
```
APPEND LINES OF y_error_table TO error_table.
```

This enables us to pass the error data table back to the Enhancement, from the BAdI.

Again, due to nature of this particular Enhancement, we must pass through the ADRC structure. Normally, this line of code would not be required.

```
y_adrc_struc = x_adrc_struc.
```

Navigate to CMOD, Deactivate the Project, and Change.



In the following screen, select the “Component” button.

Double click the Function Exit component.

Double click the Function Exit’s Include, and click the Edit button.

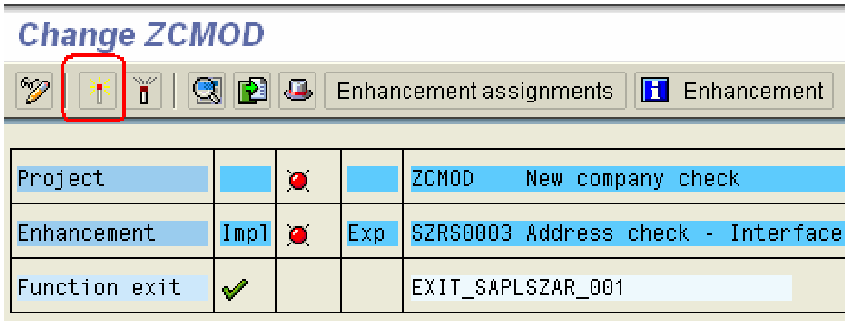
You should now be in the ABAP editor, where you can copy the following code over what we had before. It is up to you, if you wish to keep the comments displaying the parameters available.

```
DATA: lr_exit TYPE REF TO zif_ex_enh_szrs0003,
      y_error_table TYPE addr_error_tab.

************************************************************************
* Enhancement specific, else runtime error occurs when Project activated
y_adrc_struc = x_ adrc_struc.

************************************************************************
* BAdi stuff

* get BAdI implementation
CALL METHOD cl_exithandler=>get_instance
  CHANGING
    instance = lr_exit.

* call BAdI
CALL METHOD lr_exit->address_check
  EXPORTING
```

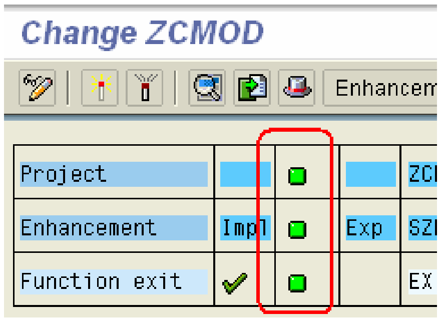
```
      x_dialog_allowed = x_dialog_allowed
      x_accept_error   = x_accept_error
      x_adrc_struc     = x_adrc_struc
    CHANGING
      y_adrc_struc     = y_adrc_struc
      y_retcode        = y_retcode
      error_table      = y_error_table.

  ************************************************************************
  * add the BAdI error table to the Enhancement
  APPEND LINES OF y_error_table TO error_table.
```

The BAdI insertion code above is typically how SAP implements BAdI insertions. Notice the Object reference to the Interface declaration, and the two Method calls, one fetching the BAdI instance (implementation) in runtime, the other, making the actual Method call.

Syntax Check and Activate the Include, back out (F3) to the Project ZCMOD Change screen, and Re-Activate the Project. The Activation icons should turn green.


Change ZCMOD

Project		●		ZCMOD New company check
Enhancement	Impl	●	Exp	SZRS0003 Address check - Interface
Function exit	✓			EXIT_SAPLSZAR_001


Change ZCMOD

Project		🟩		ZC
Enhancement	Impl	🟩	Exp	SZ
Function exit	✓	🟩		EX

Test the work so far by running the “Company address” maintenance screen – SUCOMP, this time we can simply edit the company created earlier. Alter the “Time Zone” and press enter. Nothing should look like it happens.

This proves that the Enhancement is Active, the BAdI Definition is in place, and without any BAdI Implementations so far, all is working.

This would be a harmless state to leave the Enhancement in, active with the BAdI programmed, but without any BAdI Implementations. We should not need to be concerned with CMOD any longer.

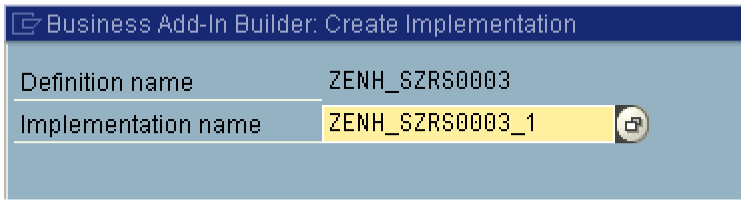
BAdI Implementation 1

Now we are ready to implement the first of two BAdIs from our single BAdI ZENH_SZRS0003 Definition.

Navigate to the BAdI Definition – SE18, enter the BAdI name “ZENH_SZRS0003”, and from the menu select

Implementation -> Create

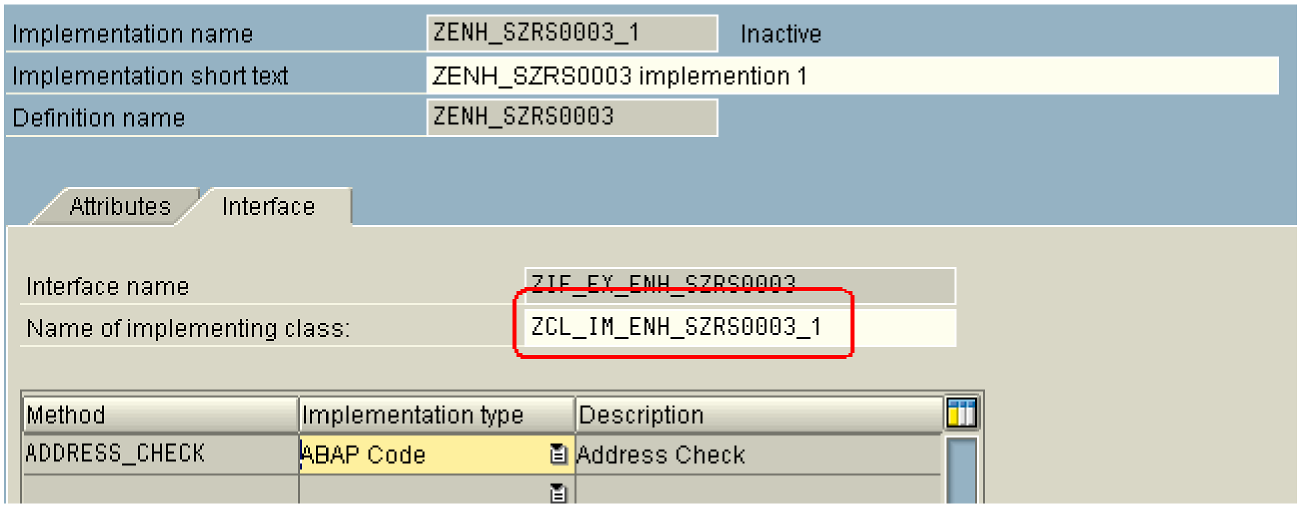
Enter an Implementation name, and continue. I have chosen the Definition name, and added a numeric value indicating the possible implementations of this definition.


Business Add-In Builder: Create Implementation

Definition name	ZENH_SZRS0003
Implementation name	ZENH_SZRS0003_1

Save the Implementation as a "Local Object".

Select the Interface tab & double click the Name of the given Implementing Class

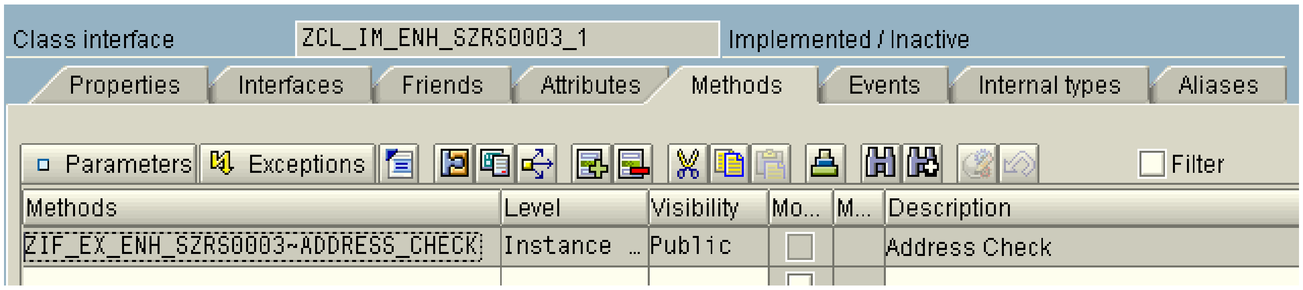

Implementation name	ZENH_SZRS0003_1	Inactive
Implementation short text	ZENH_SZRS0003 implemention 1	
Definition name	ZENH_SZRS0003	

Interface name	ZIF_EX_ENH_SZRS0003
Name of implementing class:	ZCL_IM_ENH_SZRS0003_1

Method	Implementation type	Description
ADDRESS_CHECK	ABAP Code	Address Check

We are now in the Class Builder – SE24, once again.

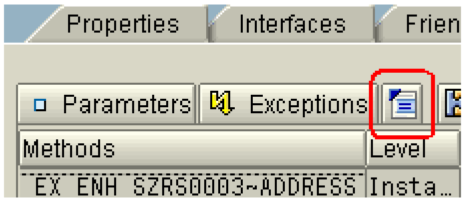
Notice that the Interface built during the BAdI definition has now been utilized in the BAdI implementation's Class – ZCL_IM_ENH_SZRZ0003_1.


Class interface ZCL_IM_ENH_SZRS0003_1 Implemented / Inactive

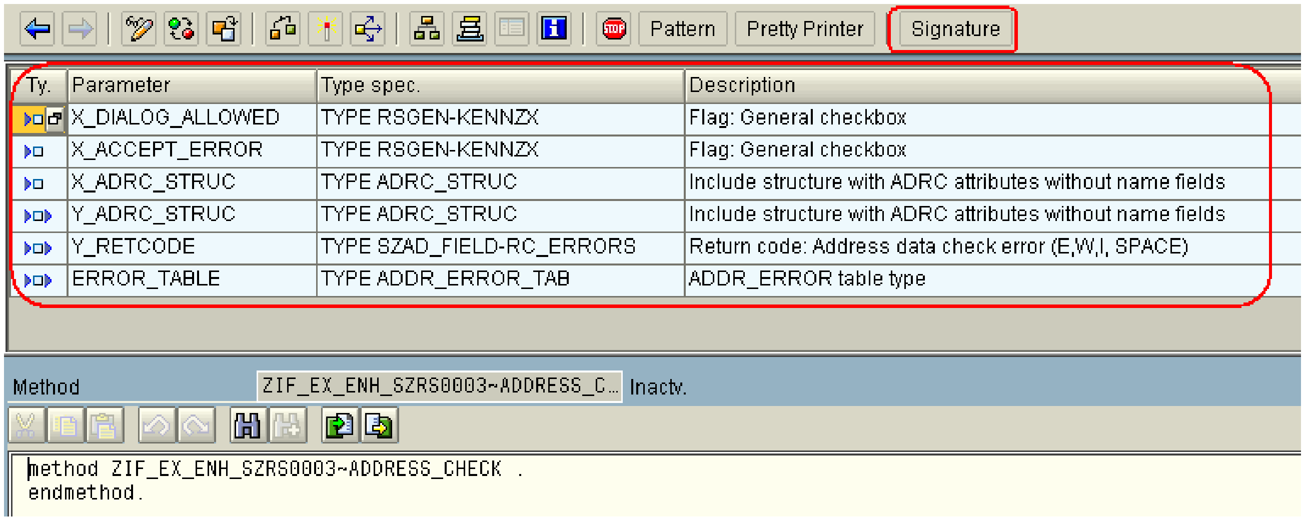
Methods	Level	Visibility	Mo...	M...	Description
ZIF_EX_ENH_SZRS0003~ADDRESS_CHECK	Instance	Public	☐		Address Check

We are finally ready to code our custom functionality that will be called by the Enhancement.

Click once on the Code Button



Press the Signature button and now we can see all the parameters that we have mirrored from the BAdI, available to the Method.



Ty.	Parameter	Type spec.	Description
	X_DIALOG_ALLOWED	TYPE RSGEN-KENNZX	Flag: General checkbox
	X_ACCEPT_ERROR	TYPE RSGEN-KENNZX	Flag: General checkbox
	X_ADRC_STRUC	TYPE ADRC_STRUC	Include structure with ADRC attributes without name fields
	Y_ADRC_STRUC	TYPE ADRC_STRUC	Include structure with ADRC attributes without name fields
	Y_RETCODE	TYPE SZAD_FIELD-RC_ERRORS	Return code: Address data check error (E,W,I, SPACE)
	ERROR_TABLE	TYPE ADDR_ERROR_TAB	ADDR_ERROR table type

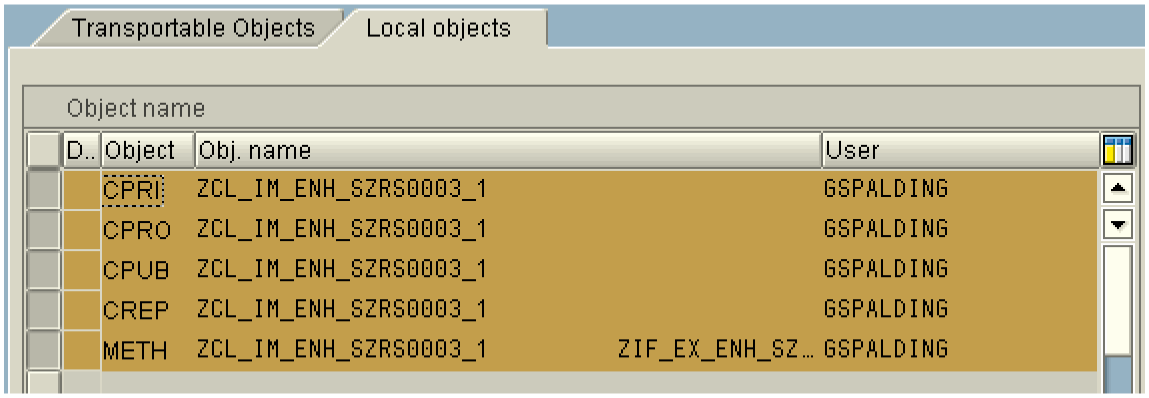
Method ZIF_EX_ENH_SZRS0003~ADDRESS_C… Inactv.

```
method ZIF_EX_ENH_SZRS0003~ADDRESS_CHECK .
endmethod.
```

Using the trusty “POPUP_TO_CONFIRM” function module, we will program this Method accordingly. See below. Simply cut this code and add it to the Method

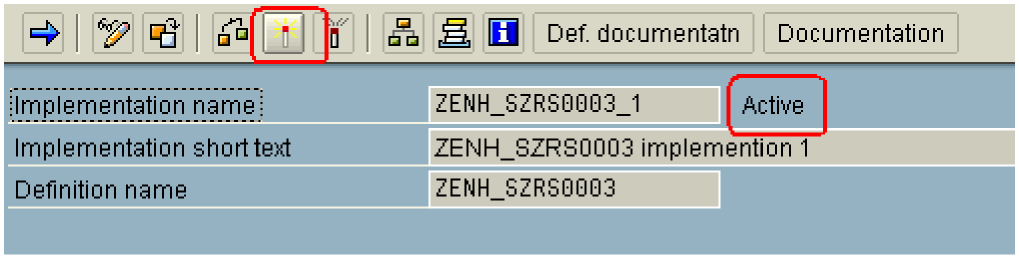
```
CALL FUNCTION 'POPUP_TO_CONFIRM'
  EXPORTING
*   TITLEBAR                    = ' '
*   DIAGNOSE_OBJECT             = ' '
    text_question               = 'This was fired from Implementation 1'.
*   TEXT_BUTTON_1               = 'Ja'(001)
*   ICON_BUTTON_1               = ' '
*   TEXT_BUTTON_2               = 'Nein'(002)
*   ICON_BUTTON_2               = ' '
*   DEFAULT_BUTTON              = '1'
*   DISPLAY_CANCEL_BUTTON       = 'X'
*   USERDEFINED_F1_HELP         = ' '
*   START_COLUMN                = 25
*   START_ROW                   = 6
*   POPUP_TYPE                  =
* IMPORTING
*   ANSWER                     =
* TABLES
*   PARAMETER                  =
* EXCEPTIONS
*   TEXT_NOT_FOUND             = 1
*   OTHERS                     = 2
          .
IF sy-subrc <> 0.
* MESSAGE ID SY-MSGID TYPE SY-MSGTY NUMBER SY-MSGNO
*         WITH SY-MSGV1 SY-MSGV2 SY-MSGV3 SY-MSGV4.
ENDIF.
```

"Syntax Check", "Save", and "Activate" all objects.



D..	Object	Obj. name		User
	CPRI	ZCL_IM_ENH_SZRS0003_1		GSPALDING
	CPRO	ZCL_IM_ENH_SZRS0003_1		GSPALDING
	CPUB	ZCL_IM_ENH_SZRS0003_1		GSPALDING
	CREP	ZCL_IM_ENH_SZRS0003_1		GSPALDING
	METH	ZCL_IM_ENH_SZRS0003_1	ZIF_EX_ENH_SZ...	GSPALDING

Back out twice, returning to the BAdI Implementation screen. We now want to “Activate” the BAdI Implementation. This would be comparative to activating the Project,

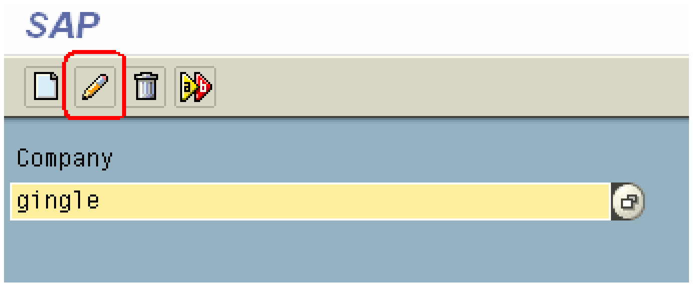


Implementation name	ZENH_SZRS0003_1	Active
Implementation short text	ZENH_SZRS0003 implemention 1	
Definition name	ZENH_SZRS0003	

We have finished programming the first custom piece of functionality for our Company Address maintenance screen.

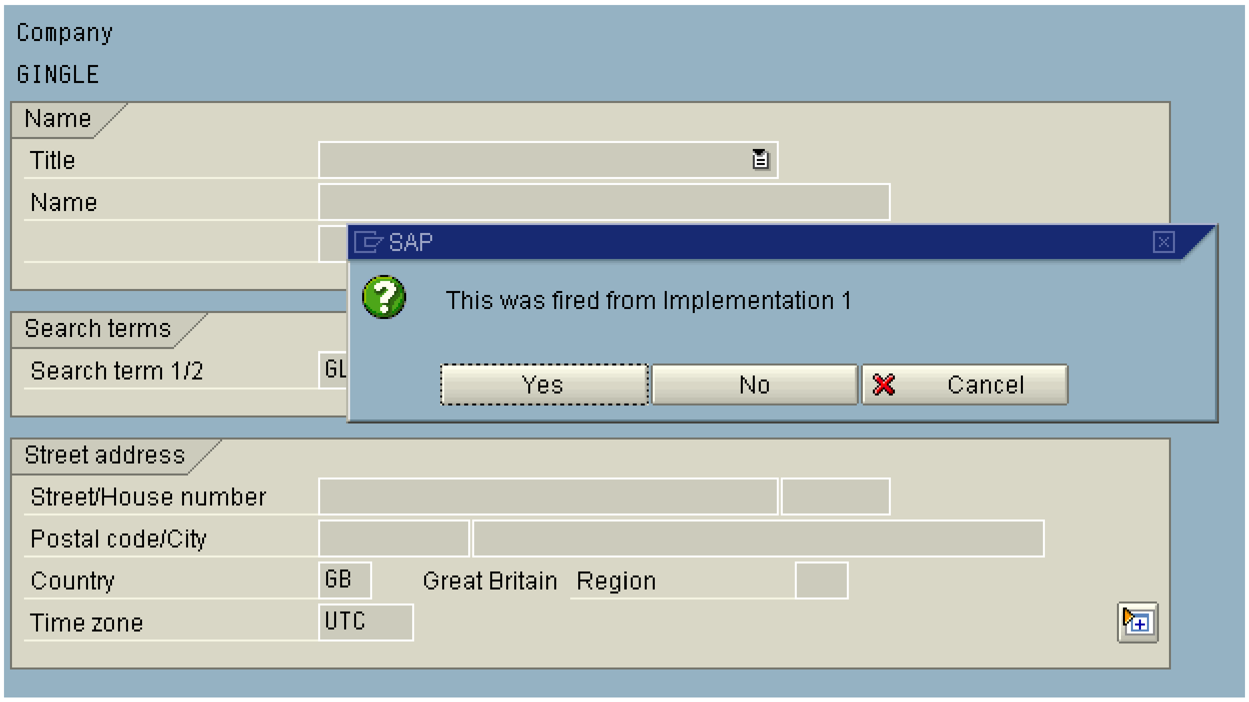
Test BAdI Implementation 1

Navigate to the Company Address maintenance screen – SUCOMP, and enter the company name we used before and press Edit.



gingle

Change the Time Zone to something, and now you should see the popup programmed in the BAdI Implementation.



GINGLE

This was fired from Implementation 1

Country	GB Great Britain
Time zone	UTC

Press any key to continue.

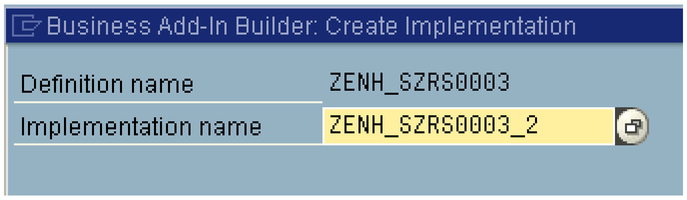
BAdI Implementation 2

Now we are ready to implement the second of our two BAdIs from our single BAdI ZENH_SZRS0003 definition.

Navigate to the BAdI definition – SE18, enter the BAdI name, and from the menu select

Implementation -> Create

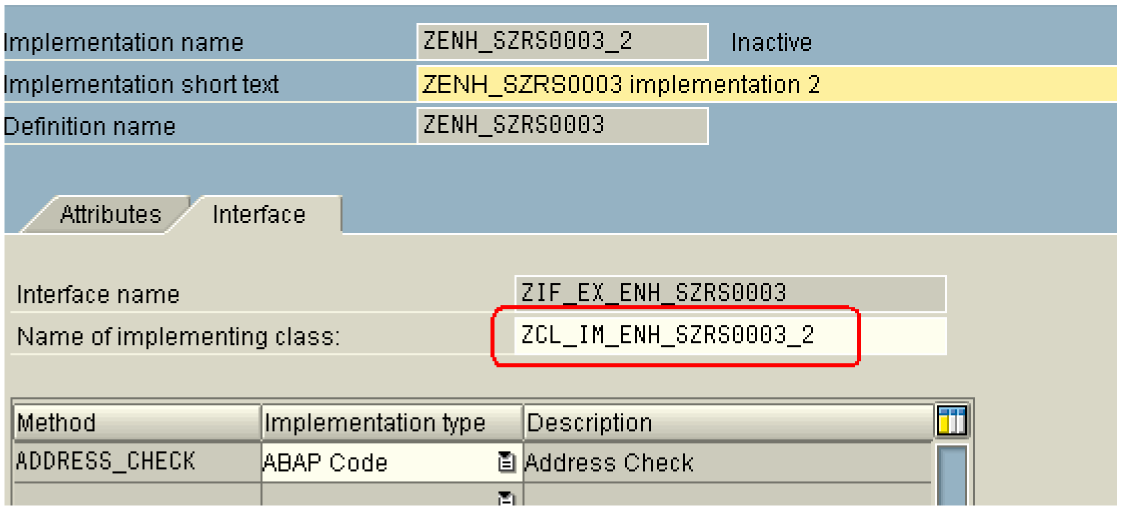
Enter an Implementation name, and continue. Notice this name has a two at the end



Business Add-In Builder: Create Implementation

Definition name	ZENH_SZRS0003
Implementation name	ZENH_SZRS0003_2

On the next screen, supply a short text, and on the Interface tab, double click the Name of the given Implementing Class



Implementation name	ZENH_SZRS0003_2	Inactive
Implementation short text	ZENH_SZRS0003 implementation 2	
Definition name	ZENH_SZRS0003	

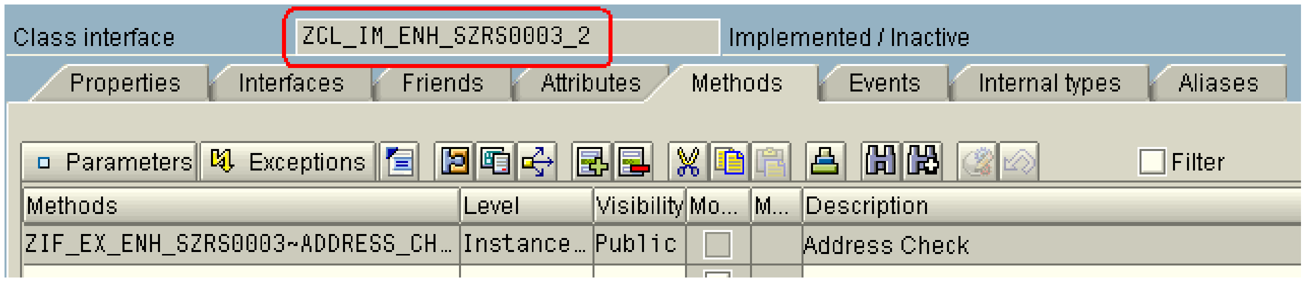
Interface name	ZIF_EX_ENH_SZRS0003
Name of implementing class:	ZCL_IM_ENH_SZRS0003_2

Method	Implementation type	Description
ADDRESS_CHECK	ABAP Code	Address Check

Save the Implementation if prompted and a “Local Object” type will suffice.

We have now been navigated to the Class Builder – SE24, once again.

This time the Interface built during the BAdIs second definition has now been utilized in the BAdI implementation’s Class – ZCL_IM_ENH_SZRZ0003_2.

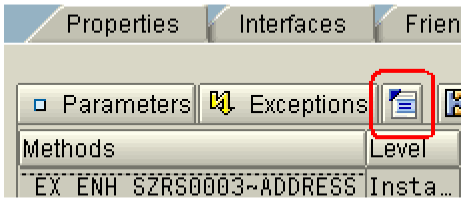


Class interface	ZCL_IM_ENH_SZRS0003_2	Implemented / Inactive

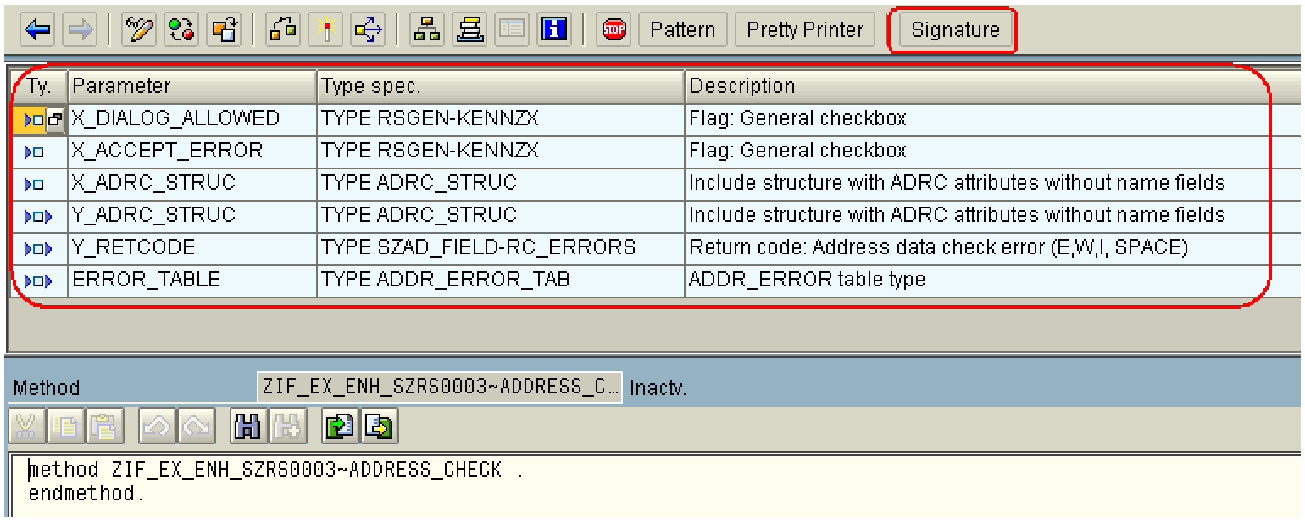
Methods	Level	Visibility	Mo...	M...	Description
ZIF_EX_ENH_SZRS0003~ADDRESS_CH...	Instance...	Public	☐		Address Check

We are finally ready to code our second custom functionality that will be called by the Enhancement.

Click once on the Code Button



Press the Signature button and now we can see all the parameters that we have mirrored from the BAdI, available to the Method.



Ty.	Parameter	Type spec.	Description
	X_DIALOG_ALLOWED	TYPE RSGEN-KENNZX	Flag: General checkbox
	X_ACCEPT_ERROR	TYPE RSGEN-KENNZX	Flag: General checkbox
	X_ADRC_STRUC	TYPE ADRC_STRUC	Include structure with ADRC attributes without name fields
	Y_ADRC_STRUC	TYPE ADRC_STRUC	Include structure with ADRC attributes without name fields
	Y_RETCODE	TYPE SZAD_FIELD-RC_ERRORS	Return code: Address data check error (E,W,I, SPACE)
	ERROR_TABLE	TYPE ADDR_ERROR_TAB	ADDR_ERROR table type

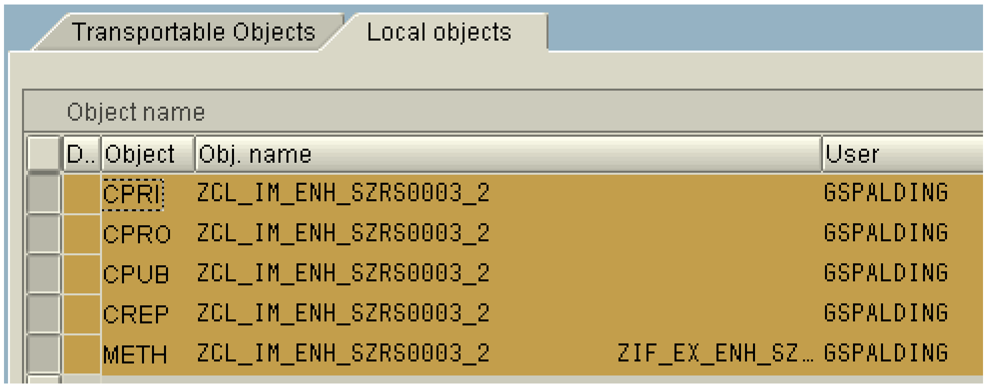
Method ZIF_EX_ENH_SZRS0003~ADDRESS_C… Inactv.

```
method ZIF_EX_ENH_SZRS0003~ADDRESS_CHECK .
endmethod.
```

Using the trusty “POPUP_TO_CONFIRM” function module, we will program this Method accordingly. See below. Only the "text_question" has changed.

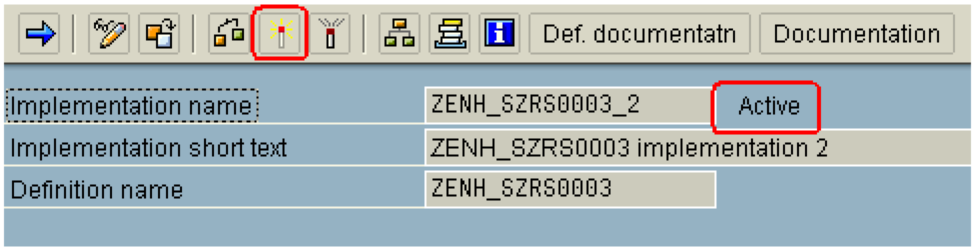
```
CALL FUNCTION 'POPUP_TO_CONFIRM'
  EXPORTING
*   TITLEBAR                     = ' '
*   DIAGNOSE_OBJECT              = ' '
    text_question                = 'This was fired from Implementation 2'.
*   TEXT_BUTTON_1                = 'Ja'(001)
*   ICON_BUTTON_1                = ' '
*   TEXT_BUTTON_2                = 'Nein'(002)
*   ICON_BUTTON_2                = ' '
*   DEFAULT_BUTTON               = '1'
*   DISPLAY_CANCEL_BUTTON        = 'X'
*   USERDEFINED_F1_HELP          = ' '
*   START_COLUMN                 = 25
*   START_ROW                    = 6
*   POPUP_TYPE                    =
* IMPORTING
*   ANSWER                        =
* TABLES
*   PARAMETER                     =
* EXCEPTIONS
*   TEXT_NOT_FOUND                = 1
*   OTHERS                        = 2
          .
IF sy-subrc <> 0.
* MESSAGE ID SY-MSGID TYPE SY-MSGTY NUMBER SY-MSGNO
*         WITH SY-MSGV1 SY-MSGV2 SY-MSGV3 SY-MSGV4.
ENDIF.
```

"Syntax Check", "Save", and "Activate" all objects.



D..	Object	Obj. name		User
	CPRI	ZCL_IM_ENH_SZRS0003_2		GSPALDING
	CPRO	ZCL_IM_ENH_SZRS0003_2		GSPALDING
	CPUB	ZCL_IM_ENH_SZRS0003_2		GSPALDING
	CREP	ZCL_IM_ENH_SZRS0003_2		GSPALDING
	METH	ZCL_IM_ENH_SZRS0003_2	ZIF_EX_ENH_SZ…	GSPALDING

Back out twice, returning to the BAdI Implementation screen. We now want to “Activate” the BAdI Implementation. This would be comparative to activating the Project,

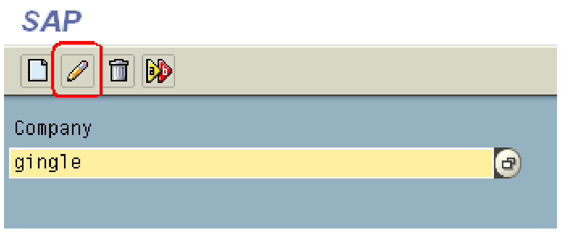


Implementation name	ZENH_SZRS0003_2	Active
Implementation short text	ZENH_SZRS0003 implementation 2	
Definition name	ZENH_SZRS0003	

We have finished programming the second and last custom piece of functionality for our Company Address maintenance screen to prove the point.

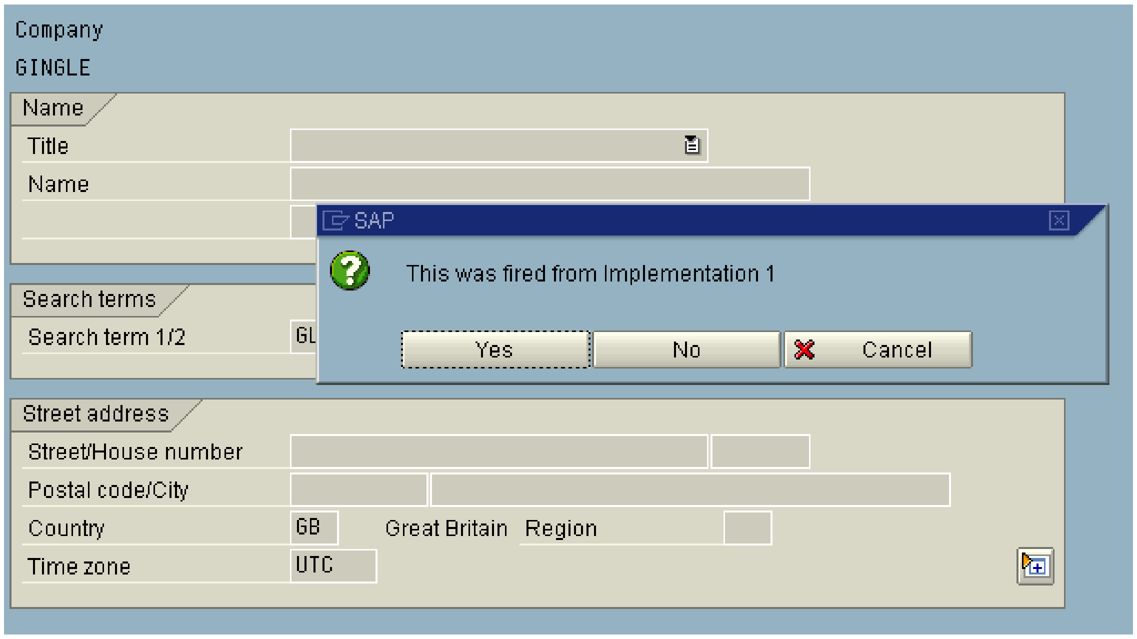
Test BAdI Implementation 1 and 2

Navigate to the Company Address maintenance screen – SUCOMP, and enter the company name we used before and press Edit.



gingle

Change the Time Zone to something, and now you should see the popup programmed in the BAdI Implementation.


Company
GINGLE

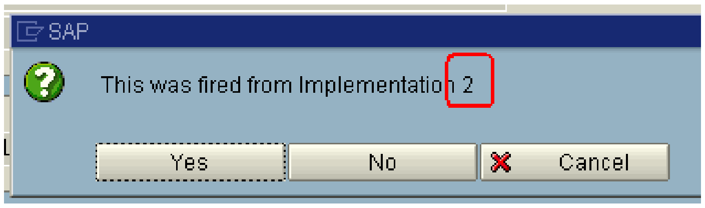
Name
Title
Name

Search terms
Search term 1/2 GL

Street address
Street/House number
Postal code/City
Country GB Great Britain Region
Time zone UTC

SAP
This was fired from Implementation 1

Press any key to continue and now we get the second popup implemented in BAdI implementation 2.


SAP
This was fired from Implementation 2

That concludes the testing.

Implementing a BAdI in an Enhancement Project (CMOD)

SAP DEVELOPER NETWORK

Summary

This tutorial demonstrates how to program a custom BAdI in an SAP Enhancement. It further demonstrates the ability to separate custom functionality away from non-related functionality, in different objects, which can be maintained and activated independently. This behaviour would not be possible using the standard SAP Enhancement CMOD, concept alone.

Author Bio

Graduated a mature student in 1994 with a BSc (Hons), I began work in a company programming MS Access applications. After a year, I began contracting as a VB, MS Access, and Excel Application Programmer. 2 years later, in 1997, I started a permanent job working as an SAP Technical Consultant for an SAP Implementation Partner working on various projects with an array of SAP technologies. In 2001, I decided to leap into contracting. More recently, I have extended my skills by instructing courses at SAP UK.

I fell upon this workaround to Enhancements, when I was teaching myself some basic OO and BAdI skills. When I learnt some BAdIs have multiple uses, i.e. several implementations of the same BAdI, it was then, I thought about using a custom BAdI inside an Enhancement to remove some of the inherent problems that beset Enhancements.

Disclaimer & Liability Notice