BC427

Enhancement Framework

SAP NetWeaver

Date _____

Training Center _____

Instructors _____

_____

Education Website _____

Participant Handbook

Course Version: 92
Course Duration: 2 Day(s)
Material Number: 50099928

An SAP course - use it to learn, reference it for work

Internal Use SAP Partner Only

Copyright

Trademarks

Disclaimer

Internal Use SAP Partner Only

About This Handbook

This handbook is intended to complement the instructor-led presentation of this course, and serve as a source of reference. It is not suitable for self-study.

Typographic Conventions

American English is the standard used in this handbook. The following typographic conventions are also used.

Type Style	Description
Example text	Words or characters that appear on the screen. These include field names, screen titles, pushbuttons as well as menu names, paths, and options. Also used for cross-references to other documentation both internal and external.
Example text	Emphasized words or phrases in body text, titles of graphics, and tables
EXAMPLE TEXT	Names of elements in the system. These include report names, program names, transaction codes, table names, and individual key words of a programming language, when surrounded by body text, for example SELECT and INCLUDE.
`Example text`	Screen output. This includes file and directory names and their paths, messages, names of variables and parameters, and passages of the source text of a program.
`Example text`	Exact user entry. These are words and characters that you enter in the system exactly as they appear in the documentation.
`<Example text>`	Variable user entry. Pointed brackets indicate that you replace these words and characters with appropriate entries.

Internal Use SAP Partner Only

Icons in Body Text

The following icons are used in this handbook.

Icon	Meaning
	For more information, tips, or background
	Note or further explanation of previous point
	Exception or caution
	Procedures
	Indicates that the item is displayed in the instructor's presentation.

Contents

Internal Use SAP Partner Only

Course Overview

This course provides information about using enhancement options in the *Enhancement Framework*, which is delivered as of *SAP NetWeaver 7.0*.

Target Audience

This course is intended for the following audiences:

- Customers and consultants who want to use the enhancement options in the *Enhancement Framework*

Course Prerequisites

Required Knowledge

- Knowledge of ABAP Objects (Object-oriented programming in ABAP)

Recommended Knowledge

- Knowledge of enhancement technology previously used (prior to *SAP NetWeaver 7.0*).

Course Goals

This course will prepare you to:

- Use enhancement options in the *Enhancement Framework* delivered as of *SAP NetWeaver 7.0*.

Course Objectives

After completing this course, you will be able to:

- Provide an overview of the classic enhancement technology and the new enhancement concept *(Enhancement Framework)*
- Use the new options as of *SAP NetWeaver 7.0* to enhance the *ABAP Dictionary*
- Use enhancement points and options and enhancement sections effectively to enhance SAP software
- Find and use enhancements that are determined using BAdI technology
- Use the Switch Framework to activate and deactivate enhancement implementations

Internal Use SAP Partner Only

Internal Use SAP Partner Only

Unit 1

Overview

Unit Overview

This unit will present the basic concepts in the enhancement area. It also gives an overview of the classic and the new (using the *Enhancement Framework*) enhancement technology and options.

Unit Objectives

After completing this unit, you will be able to:

- Explain the basic concepts in the enhancement area
- List the technology used by SAP to implement enhancements prior to *SAP NetWeaver 7.0*
- List the new technology for implementing enhancements as of *SAP NetWeaver 7.0*
- Explain what enhancement points or enhancement options and enhancement sections are

Unit Contents

Lesson: Basic Concepts and an Overview of the Enhancement Technology

Lesson Overview

This lesson describes basic enhancement concepts and it also provides an overview of classic enhancement options and the new enhancement options in the *Enhancement Framework*.

Lesson Objectives

After completing this lesson, you will be able to:

- Explain the basic concepts in the enhancement area
- List the technology used by SAP to implement enhancements prior to *SAP NetWeaver 7.0*
- List the new technology for implementing enhancements as of *SAP NetWeaver 7.0*
- Explain what enhancement points or enhancement options and enhancement sections are

Business Example

You want to obtain an overview of the basic enhancement concepts used by SAP to date, and also of the new enhancements technology as of *SAP NetWeaver 7.0*.

Basic Concepts and Classic Enhancement Technology

This section introduces the basic concepts of enhancements. It also introduces classic enhancement technology used by SAP for implementing enhancements prior to *SAP NetWeaver 7.0*.

There are three different enhancement types, which will be described in the following section.

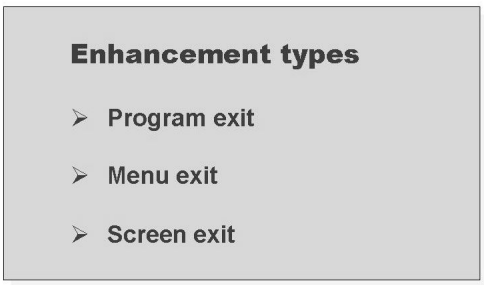

Figure 1: Enhancement types

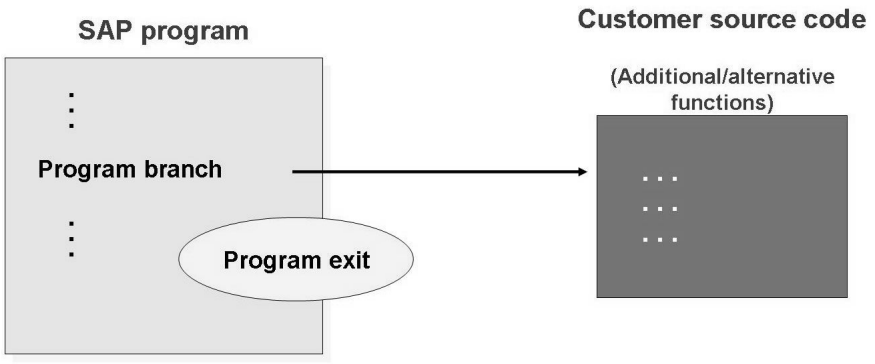

Figure 2: Program exit

A program exit is a jump or link from an SAP program to a customer source code. These exits are used by customers to implement additional or alternative functions for the SAP program without making modifications.

SAP can use various enhancement technologies to implement program exits.

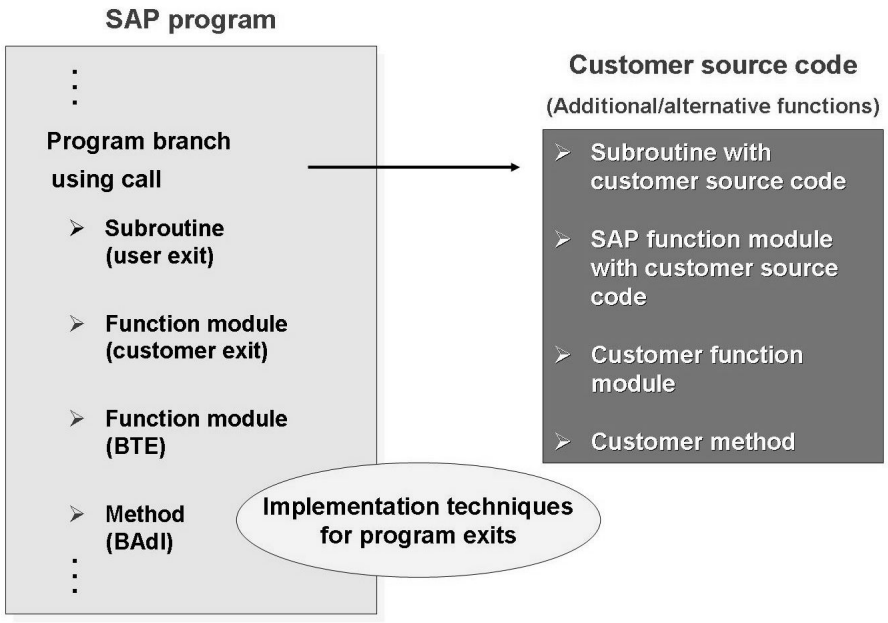

Figure 3: Implementation techniques for program exits

At the end of this section, we will see why SAP has used so many different techniques.

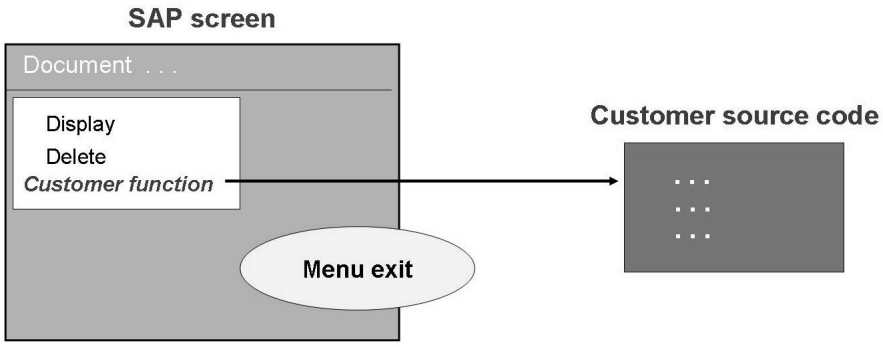

Figure 4: Menu exit

A menu exit is the option to include an additional menu entry with customer functions in an SAP screen without making modifications.

Menu exits can be implemented only using the *Customer Exit* and *BAdI* technology and they can be used just once.

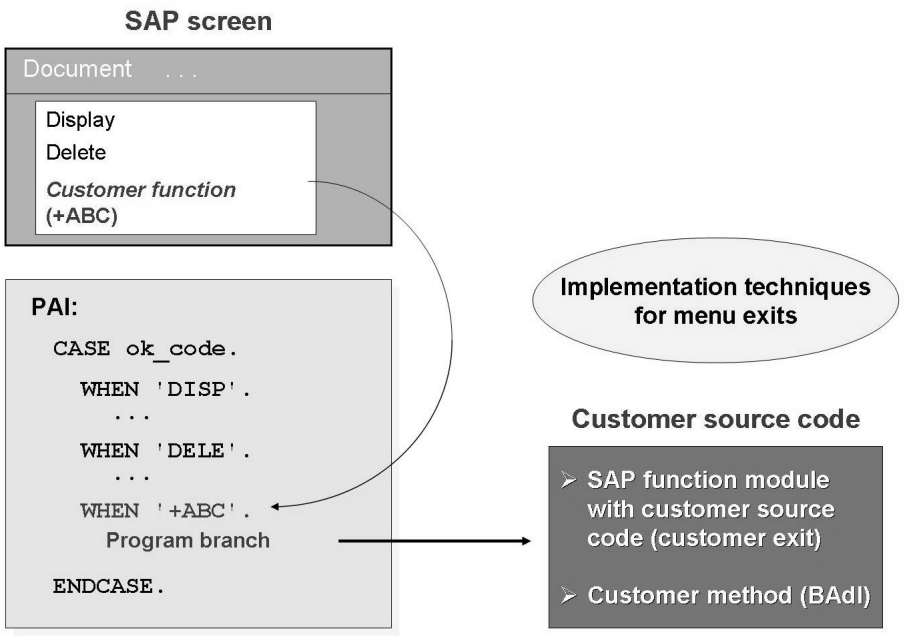

Figure 5: Implementation techniques for menu exits

A screen exit is the option to include additional fields in an SAP screen without making modifications. SAP fields and customer append fields can be used for this.

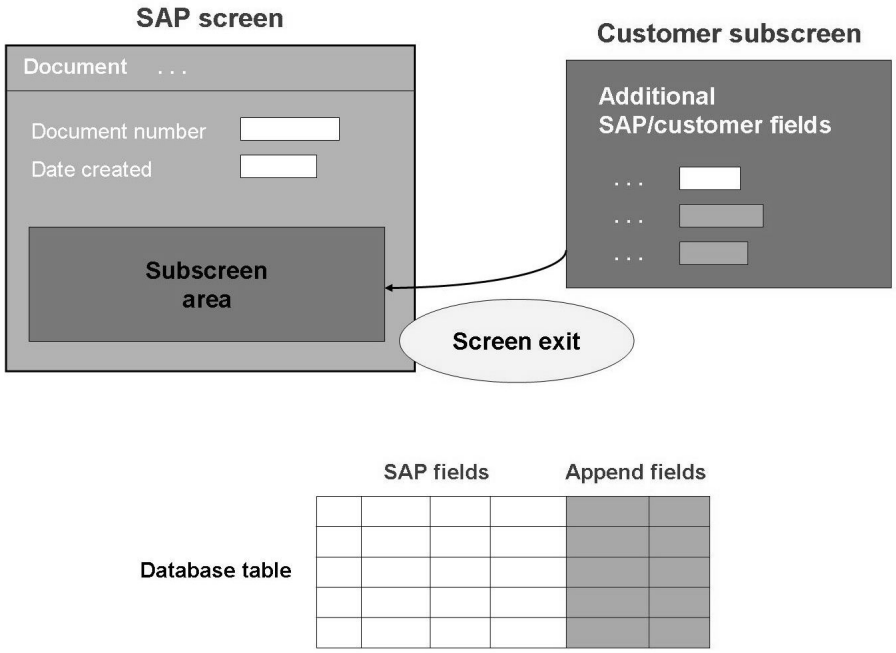

Figure 6: Screen exit

Screen exits can also be implemented only using the *Customer Exit* and *BAdI* technology. Like menu exits, they can be used just once.

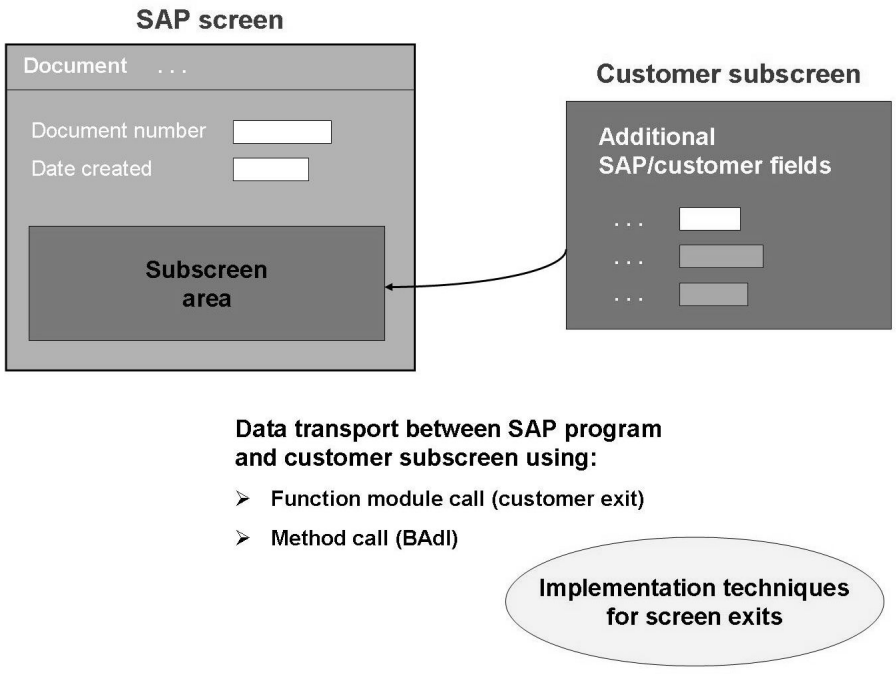

Figure 7: Implementation techniques for screen exits

The following figure shows why SAP "created" the various enhancement methods and used them to implement enhancements.

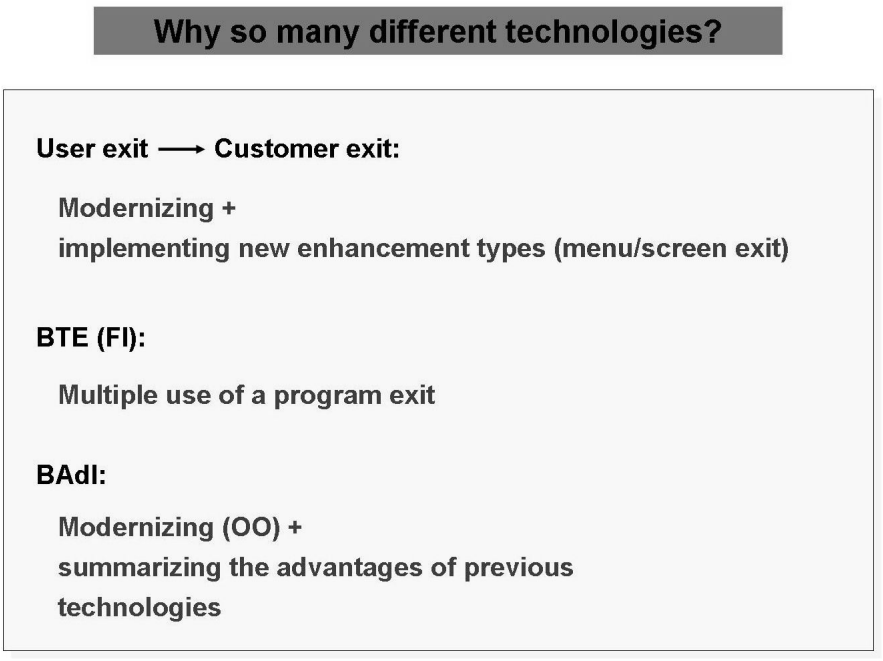

Figure 8: Origin of the implementation techniques

Although menu and screen exits can never be used more than once, program exits that are implemented using the *Business Transaction Event* and *BAdI* technology can be defined for multiple use. In this case, a program exit may have several active implementations that are processed in succession at runtime.

The following figure shows the runtime architecture of multiple-use program exits.

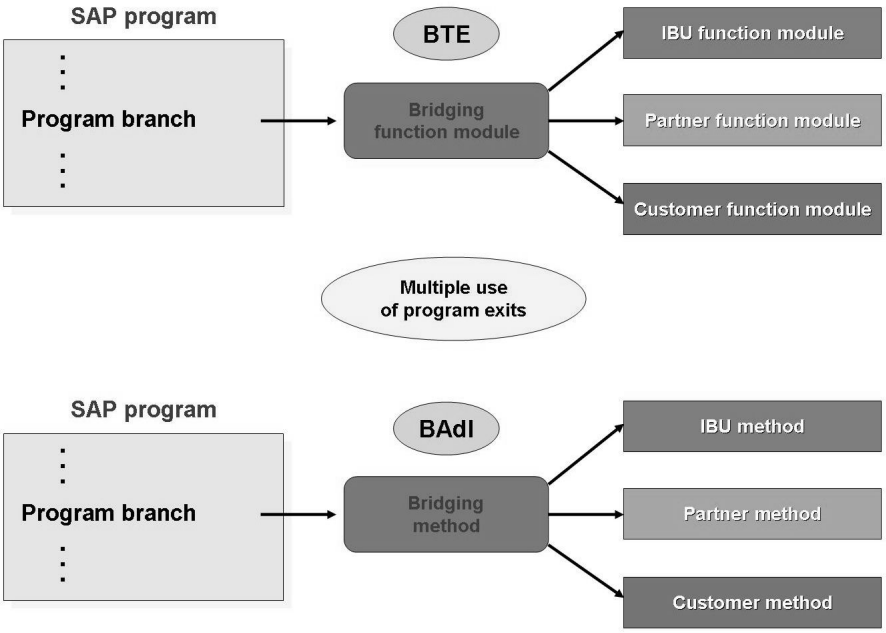

Figure 9: Multiple use of a program exit

Enhancement Technology in the Enhancement Framework

This section provides an overview of the new enhancement options implemented in the Enhancement Framework (as of *SAP NetWeaver 7.0*).

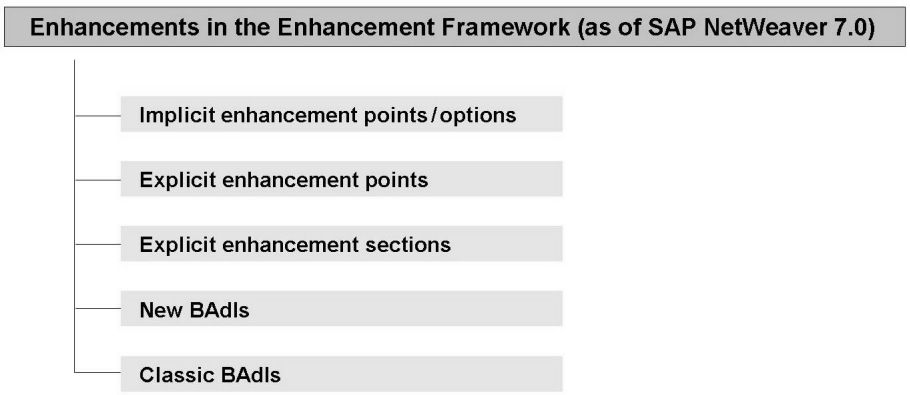

Figure 10: Overview: Enhancements in the Enhancement Framework

In addition to the new enhancement options (enhancement points/options and enhancement sections), the BAdIs are also managed in the *Enhancement Framework*.

Internal Use SAP Partner Only

As of *SAP NetWeaver 7.0*, SAP revised the BAdI technology in order to improve performance. In future, enhancements will be implemented using the BAdI technology. However, the system contains a wide range of BAdI exits that were implemented using the classic BAdI technology. Therefore, this course also provides an overview of the use of classic BAdIs.

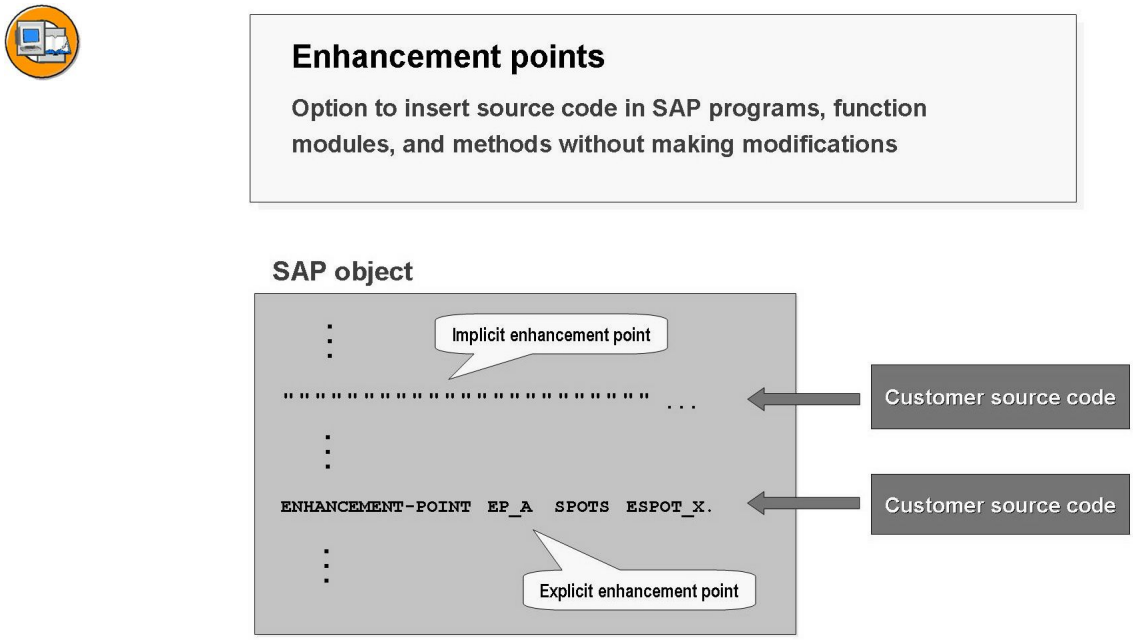

Figure 11: Overview: Enhancement points

Enhancement points are points in SAP programs, in which customers can add additional functions in the form of source code without making modifications.

Implicit enhancement points are at certain points in the SAP program and they are not explicitly provided by SAP Development. Explicit enhancement points are enhancement points provided by SAP Development.

Enhancement options

Option to define the following without making modifications:

- Additional (optional) interface parameters in SAP function modules and SAP methods
- Additional attributes and methods in SAP classes
- Preparation/postprocessing functions for global SAP methods
- Replacements for global SAP methods

Figure 12: Overview: Enhancement options

Enhancement options are always implicitly available. Customers can use them to enhance SAP class definitions and interfaces of SAP function modules and methods.

Hint: Implicit enhancement points and options also exist in SAP objects that were developed before *SAP NetWeaver 7.0*.

Enhancement section:

Option to replace source code in SAP programs, function modules, and methods without making modifications.

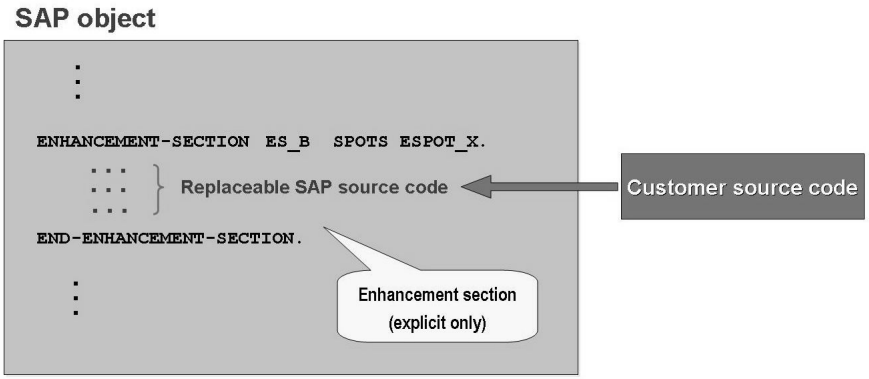

Figure 13: Overview: Enhancement section

An enhancement section is an option provided explicitly by SAP to replace a block of SAP source code with customer source code.

Lesson Summary

You should now be able to:

- Explain the basic concepts in the enhancement area
- List the technology used by SAP to implement enhancements prior to *SAP NetWeaver 7.0*
- List the new technology for implementing enhancements as of *SAP NetWeaver 7.0*
- Explain what enhancement points or enhancement options and enhancement sections are

Unit Summary

You should now be able to:

- Explain the basic concepts in the enhancement area
- List the technology used by SAP to implement enhancements prior to *SAP NetWeaver 7.0*
- List the new technology for implementing enhancements as of *SAP NetWeaver 7.0*
- Explain what enhancement points or enhancement options and enhancement sections are

Internal Use SAP Partner Only

Internal Use SAP Partner Only

Unit 2

Enhancements in ABAP Dictionary

Unit Overview

This unit will present some minor developments in the area of dictionary enhancements.

Unit Objectives

After completing this unit, you will be able to:

- create secondary indexes for SAP tables without making modifications
- add additional fixed values to SAP domains without making modifications

Unit Contents

Internal Use SAP Partner Only

Lesson: Enhancements in ABAP Dictionary

Lesson Overview

This lesson deals with the new ABAP Dictionary enhancement options as of *SAP NetWeaver 7.0*. This includes *Extension Index* and *Fixed Value Append*.

Lesson Objectives

After completing this lesson, you will be able to:

- create secondary indexes for SAP tables without making modifications
- add additional fixed values to SAP domains without making modifications

Business Example

You want to use the new ABAP Dictionary enhancement options as of *SAP NetWeaver 7.0* correctly.

Extension Index for SAP Tables

Creating a secondary index for an SAP table is a modification even though the customer name space is retained. As of *SAP NetWeaver 7.0*, you have the option of creating secondary indexes without making modifications. This type of index is called an *Extension Index*.

The following figure shows how to create an extension for an SAP table.

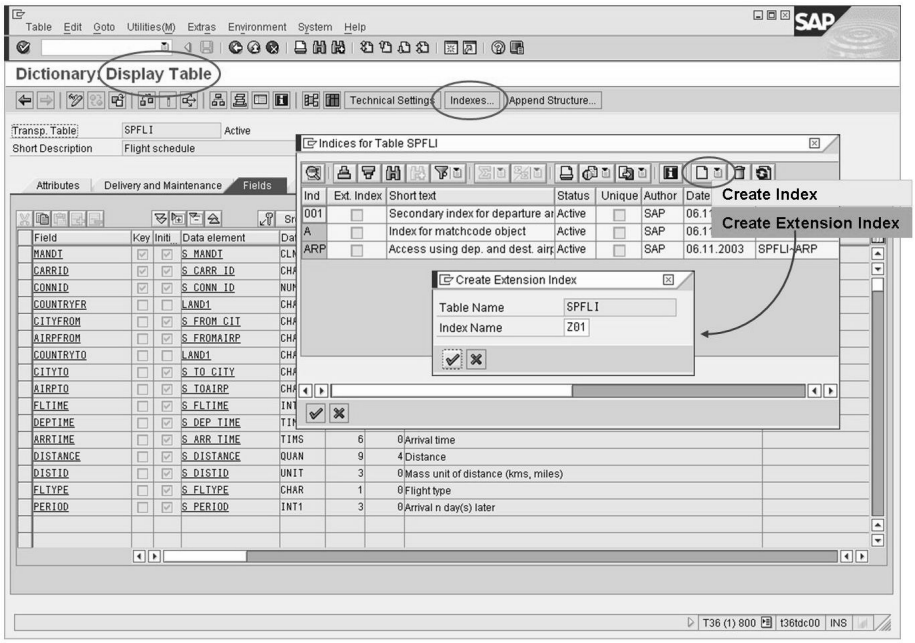

Figure 14: Extension index

Extension indexes are automatically copied over during upgrades and no modification adjustment takes place.

Fixed Value Append for SAP Domains

Previously, you had to use modifications to add additional fixed values to SAP domains As of *SAP NetWeaver 7.0*, you can use fixed value appends to add additional fixed values and you do not have to use modifications.

The following two figures illustrate how to create a fixed value append for an SAP domain.

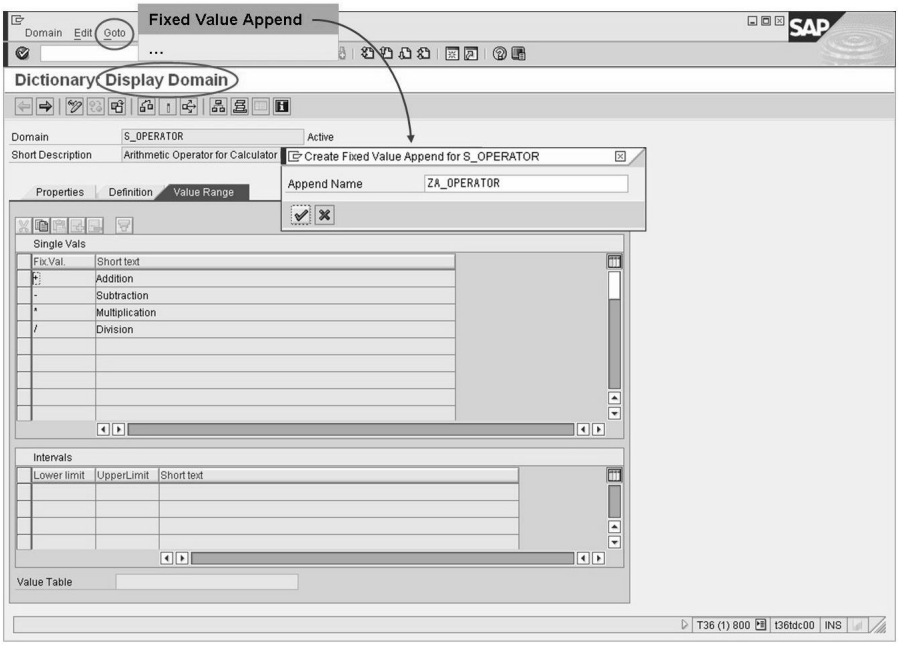

Figure 15: Creating a fixed value append

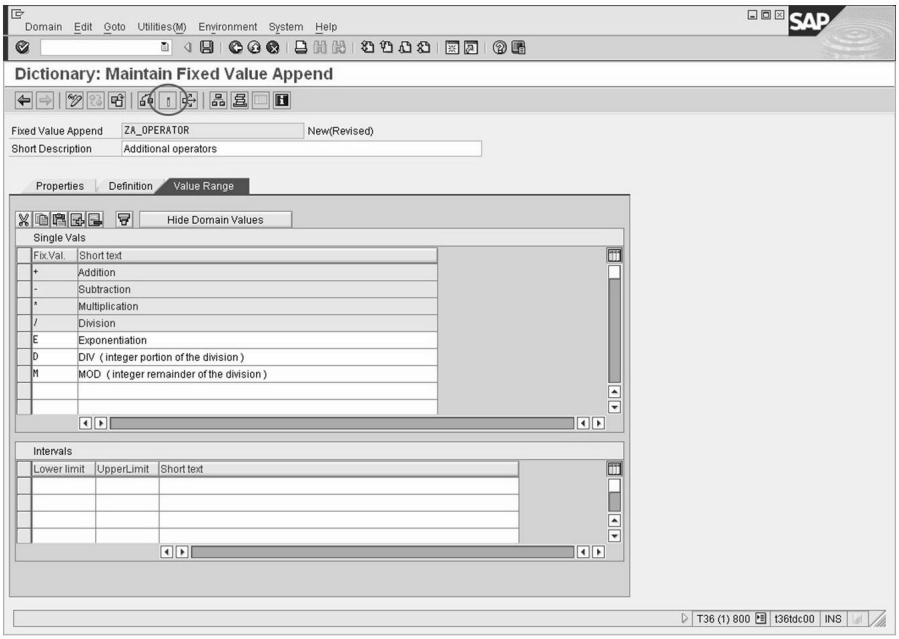

Figure 16: Maintaining additional fixed values

The append must be activated at the end of the maintenance.

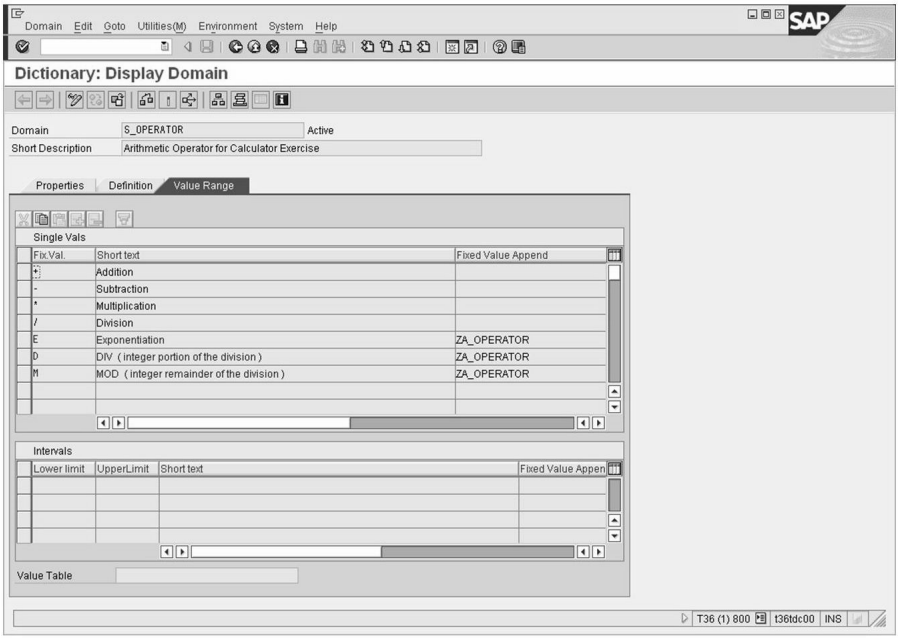

Figure 17: A domain with an activated fixed value append

In all input fields that are defined using the enhanced domain, the user can select the appended fixed values in addition to the original fixed values when they choose F4.

Due to structure appends in the dictionary the fixed value append is hold as a domain in the dictionary.

Internal Use SAP Partner Only

Exercise 1: Fixed Value Append

Exercise Objectives

After completing this exercise, you will be able to:

- use the fixed value append technology to enhance the fixed values specified in the SAP domain without making modifications

Business Example

For an SAP domain, you want to enhance the specified fixed values without making modifications.

Task:

Enhance the fixed values in SAP domains

1. Create package **ZBC427_##** (## = group number) and **assign all your exercise objects for this training course to that package**.
2. Call program **BC427_##_EPS** (## = group number). On the selection screen, determine with which domain the parameter **WEEKDAY** is defined and which fixed values are specified there.
3. Add the values **'6' (Saturday)** and **'7' (Sunday)** to the specified fixed values.

   Name your fixed value append **ZA_WEEKDAY_##** .
4. Call the program again to check the enhanced input help (F4) on the selection screen. What is the program output when you enter '6' or '7' ?

Solution 1: Fixed Value Append

Task:

Enhance the fixed values in SAP domains

1. Create package **ZBC427_##** (## = group number) and **assign all your exercise objects for this training course to that package**.

   a)

2. Call program **BC427_##_EPS** (## = group number). On the selection screen, determine with which domain the parameter **WEEKDAY** is defined and which fixed values are specified there.

   a) You can use the F1 help in the input field to determine the data element used for the definition and, therefore, also the domain.

   The domain used is called **S_WEEKDAY_##**.

   It has the fixed values '1', '2', '3', '4', '5'.

3. Add the values **'6' (Saturday)** and **'7' (Sunday)** to the specified fixed values.

   Name your fixed value append **ZA_WEEKDAY_##** .

   a) Use the menu *Goto* → *Fixed Value Append* in the domain display.

4. Call the program again to check the enhanced input help (F4) on the selection screen. What is the program output when you enter '6' or '7' ?

   a) When you enter '6' and '7', the system issues an information message saying 'Invalid weekday input'.

Lesson Summary

You should now be able to:

- create secondary indexes for SAP tables without making modifications
- add additional fixed values to SAP domains without making modifications

Internal Use SAP Partner Only

Unit Summary

You should now be able to:

- create secondary indexes for SAP tables without making modifications
- add additional fixed values to SAP domains without making modifications

Internal Use SAP Partner Only

Internal Use SAP Partner Only

Unit 3

Enhancement Points/Options and Enhancement Sections

Unit Overview

This unit provides information about using the enhancement points, enhancement options and the enhancement sections.

Unit Objectives

After completing this unit, you will be able to:

- Describe which enhancement options are available in the Enhancement Framework
- List the points in SAP objects where implicit enhancement points and options are available and use them to enhance the SAP functions without making modifications
- Find implemented explicit enhancement points and sections in SAP objects and use them to enhance or replace SAP functions

Unit Contents

Lesson: Enhancement Points/Options and Enhancement Sections

Lesson Overview

This lesson describes the new enhancement options as of *SAP NetWeaver 7.0*. This includes the implicit enhancement points and options and the explicit points and sections.

Lesson Objectives

After completing this lesson, you will be able to:

- Describe which enhancement options are available in the Enhancement Framework
- List the points in SAP objects where implicit enhancement points and options are available and use them to enhance the SAP functions without making modifications
- Find implemented explicit enhancement points and sections in SAP objects and use them to enhance or replace SAP functions

Business Example

You want to use the new enhancement options in the Enhancement Framework to enhance your SAP functions without making modifications.

Overview of the Enhancement Framework

This section gives an overview of the enhancement options included in the Enhancement Framework.

Internal Use SAP Partner Only

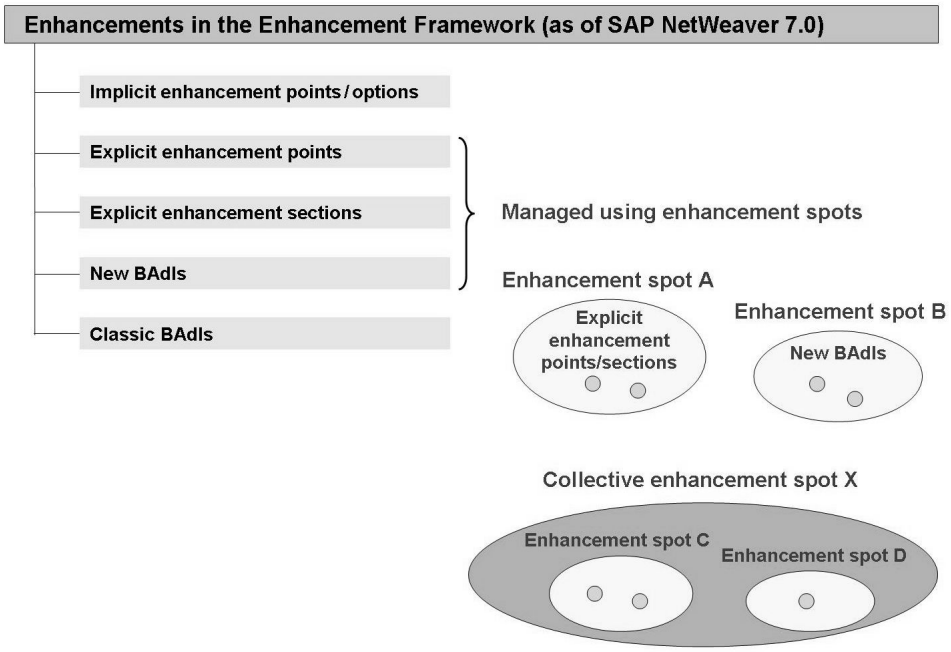

Figure 18: Detailed overview of enhancements in the Enhancement Framework

In the **Enhancement Framework**, there are some new enhancement options as of *SAP NetWeaver 7.0*: **Enhancement points/options** and **enhancement sections**. It is particularly interesting that, unlike enhancements implemented using previous enhancement technology, enhancements implemented using implicit enhancement points require no preparation from SAP.

SAP implemented the new BAdI technology for performance reasons, and other reasons described in the next unit.

The figure above illustrates how enhancement points and sections, and BAdIs that have been created using the new technology, are grouped together and managed using enhancement spots.

Composite enhancement spots comprise both simple enhancement spots and other composite enhancement spots. They serve to semantically bundle enhancement spots.

Previous BAdIs (classic BAdIs) are still available in the system as before. However, SAP will only use the new technology to implement future BAdI enhancements.

Explicit Enhancement Points and Sections

This section shows how to search for and use explicit enhancement points and enhancement sections in SAP programs.

Explicit enhancement points and enhancement sections

Option to insert or replace source code in SAP programs, function modules, and methods (on the basis of explicit preparation by *SAP)* without making modifications

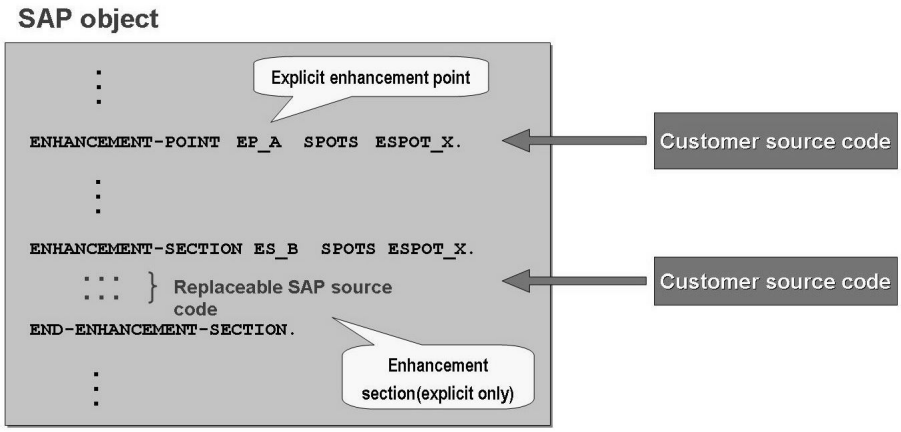

Figure 19: Explicit enhancement points and sections

An **explicit enhancement point** is an option provided by SAP to allow you to enhance the SAP source code without making modifications. It is implemented using the syntax "`ENHANCEMENT-POINT ...`".

An **explicit enhancement section** is an option provided by SAP to allow you to replace the SAP source code section without making modifications. The replaced SAP source code is enclosed by the statements "`ENHANCEMENT-SECTION ...`" and "`END-ENHANCEMENT-SECTION ...`".

Explicit enhancement points and sections are always embedded in enhancement spots.

Explicit enhancement points and sections that allow you to enhance or replace **source codes** are called **dynamic**.

Explicit enhancement points and sections that allow you to enhance or replace **declarations** are called **static**.

To use explicit enhancement points and sections, implement an **enhancement implementation** (an implementation of the higher-level enhancement spot). The following steps describe how to use explicit enhancement points and enhancement sections:

1. Display the SAP object (program, function module, method).
2. Search for the required enhancement point/section.
3. In the GUI status, choose the enhancement button.
4. Create the enhancement implementation using the context menu of the enhancement point/section.
5. Specify the name of the enhancement implementation (comply with customer namespace Y* / Z*).
6. Enter the source code.
7. In the GUI status, choose *Activate Enhancements*.

Searching for enhancement spots

For a global search it is not possible to search for explicit enhancement points and sections, but for spots. Bear in mind that in general the spot name of a spot that contains all explicit enhancements of a program is mostly ES_<Name of SAP Program>.

For a search, proceed as follows:

1. Start the Repository Information System (transaction SE84).
2. In the left window, choose *Enhancements → Enhancement Spots*.
3. Specify the search criteria for the enhancement spots you are looking for and execute to display a list of results.
4. Pick one of the results to display. You will see the listed enhancement points and sections of this spot.
5. On the *Enhancement Implementations* tab you can see all enhancement implementations of the spot. Choose one to display it.
6. The result shows all enhancement implementation elements related to the enhancement implementation name.
7. Select, for example, a dynamic enhancement implementation element and choose *Source* to see the implementation code at the bottom. The *ABAP Editor* button shows you the source code where the enhancement is offered.

Note: You can also use the **enhancement browser** (transaction SPAU_ENH) to show enhancement spots and their implementations. The functionality is the same as in the Repository Information System.

Rep. Info. System; specify search criteria for enh. spot, execute and display; choose Enh. Implementations and double-click one to see the implementation elements

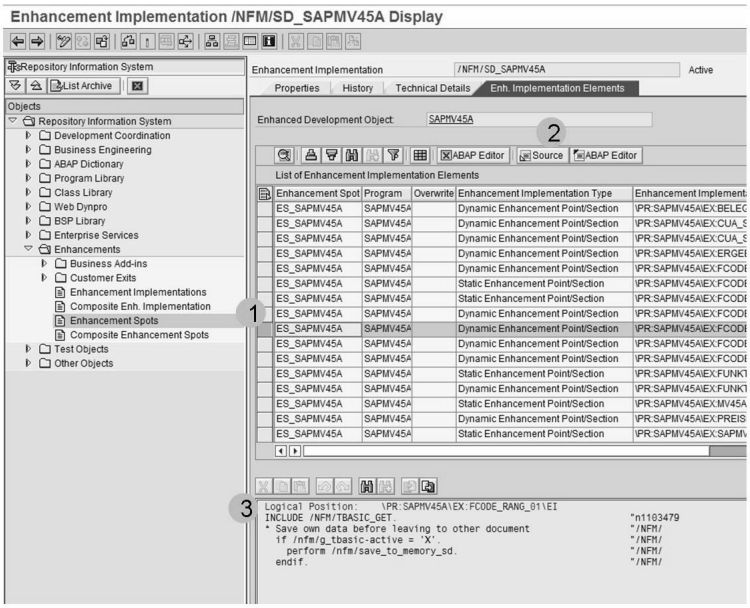

Figure 20: Information on an Enhancement Spot

As of *SAP NetWeaver 7.02* it is possible to offer another enhancement point/section within an enhancement implementation. To do so, click the right mouse button while in the implementation mode and choose *Enhancement Operations → Create Option*. Specify a point or a section name and name the spot (comply with customer namespace Y* / Z* for the spot name). Enter and activate it with the implementation activation.

Hint: Note that enhancement implementations are done on a separate software layer (such as modifications that are performed with the modification assistant). You cannot see them in the debugger directly. To debug them, step to the defined enhancement point/section and then choose *single step* (F5) to get into the enhancement implementation.

The following figure illustrates the terms **(composite) enhancement spot** and **(composite) enhancement implementation**.

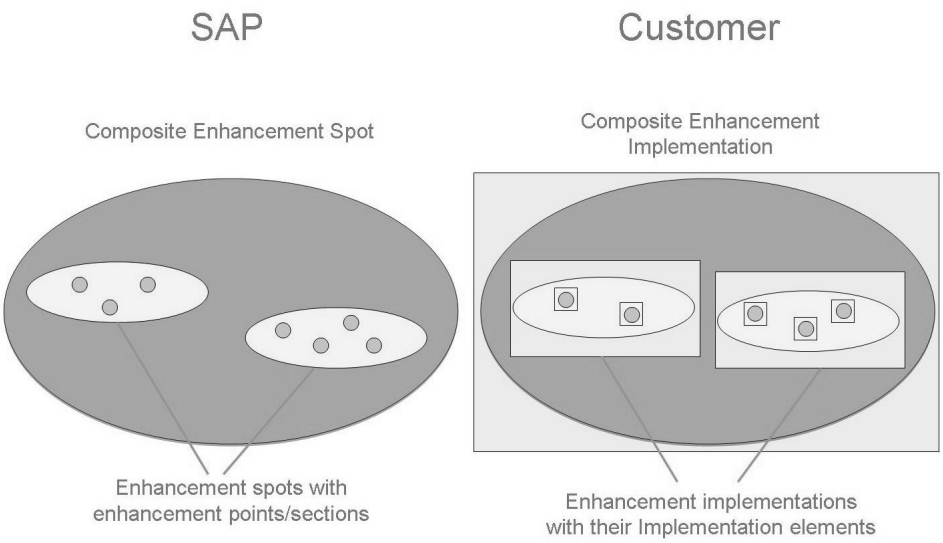

Figure 21: Enhancement Spot and Enhancement Implementation

SAP offers enhancement points/sections within enhancement spots. The enhancement spots can be combined by SAP into composite enhancement spots. This provides a better structure for the global search of enhancement spots.

Customers can implement the enhancement points/sections within enhancement implementations of the related enhancement spot. Usually, the customer implements all enhancement points/sections of one enhancement spot in one enhancement implementation. The enhanced elements are numbered automatically.

The customer also has the option to combine their enhancement implementations to a composite enhancement implementation. This makes it easier to find the enhancement implementations. However, this does not necessarily mean that a composite enhancement spot has to be implemented by a composite enhancement implementation.

Implicit Enhancement Points and Options

This section shows which implicit enhancement points and enhancement options are available in SAP programs, subroutines, function modules, and methods, and how you can use them to enhance the corresponding SAP objects without making modifications.

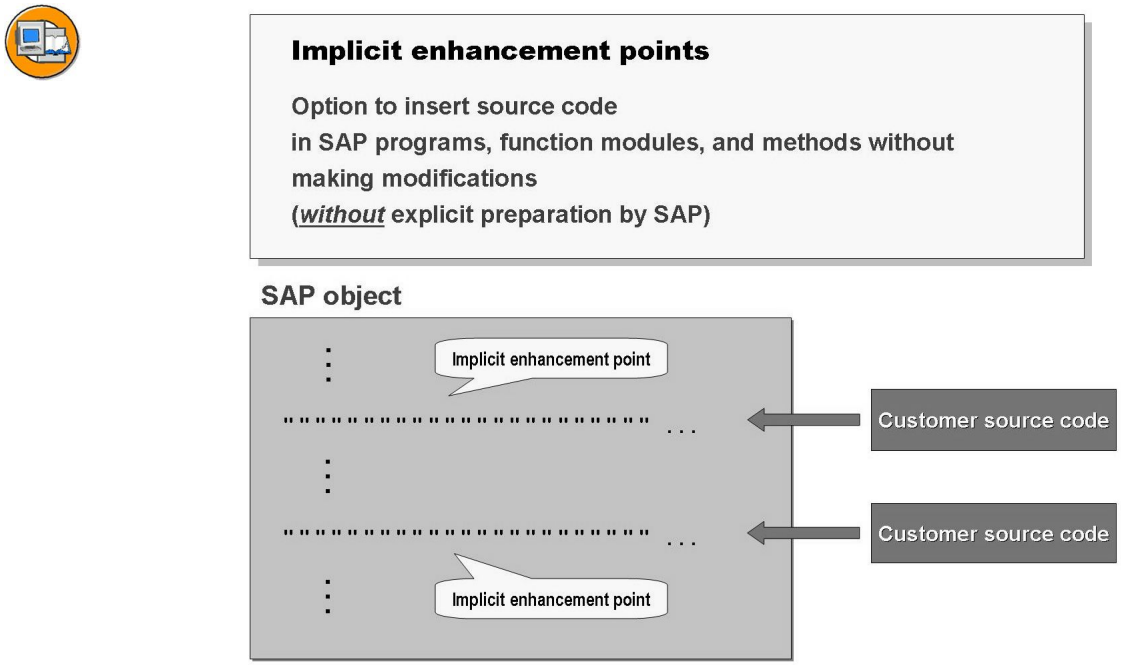

Figure 22: Overview of implicit enhancement points

To implement additional functions, customers can use the *Enhancement Framework* (as of *SAP NetWeaver 7.0*) to insert additional source code in certain, **generally** specified points without making modifications. The corresponding SAP object may be derived from earlier releases and it may **not** require any preparation by SAP application programmers (**implicit enhancement point**).

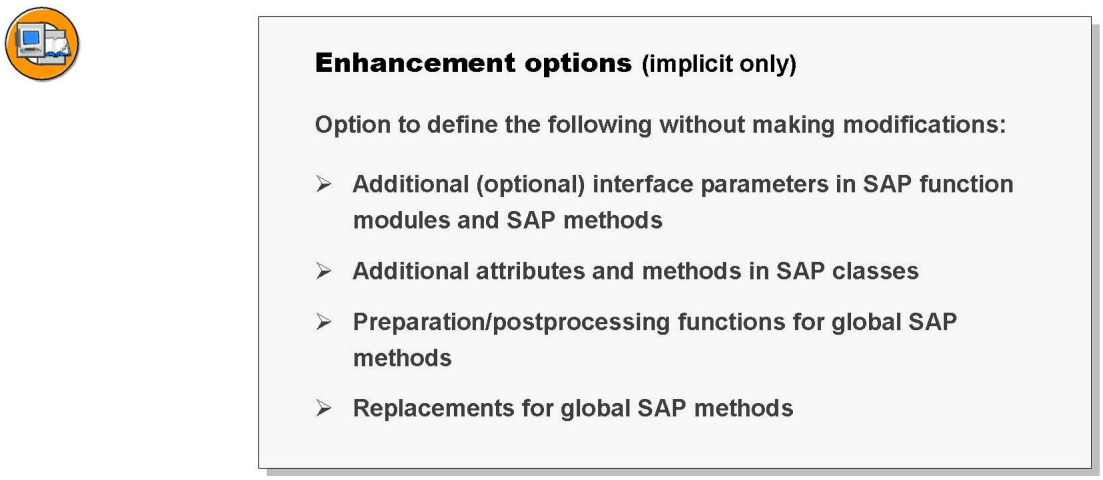

Figure 23: Overview of implicit enhancement options

The *Enhancement Framework* also allows you to enhance interfaces for SAP function modules and methods without making modifications, and to enhance attributes and methods in SAP classes (**implicit enhancement option**).

To use implicit enhancement points and options, implement an **enhancement implementation**. This procedure is described in more detail later in this section.

The following figure shows which implicit enhancement points and options are available in SAP function modules.

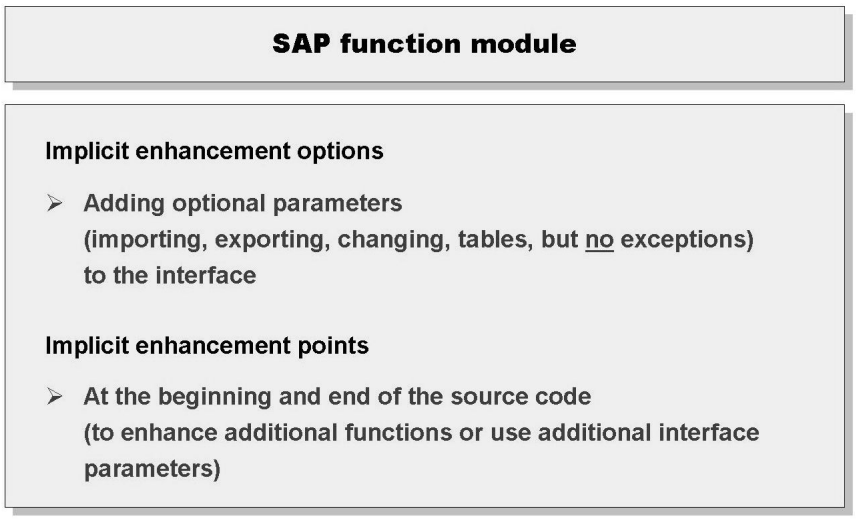

Figure 24: Implicit enhancement points and options in SAP function modules

Enhancing interfaces for SAP function modules:

1. **Display** the function module in the Function Builder.
2. Choose *Function Module → Enhance Interface* and specify an enhancement implementation to add a new interface parameter with type assignment.
3. In the GUI status, choose *Activate Enhancements*.

These newly added interface parameters are generally **optional** and they can be activated in the source code enhancements of the corresponding function modules.

Enhancing source code for SAP function modules:

1. **Display** the source code of the function module.
2. In the GUI status, choose the enhancement button
3. Choose the menu options *Edit → Enhancement Operations → Show Implicit Enhancement Options* to show the implicit enhancement options.
4. In the editor, use the context menu of one of the implicit enhancement points displayed to create an enhancement implementation.
5. Insert the source code.
6. In the GUI status, choose *Activate Enhancements*.

The following figure shows which implicit enhancement points and options are available in SAP classes.

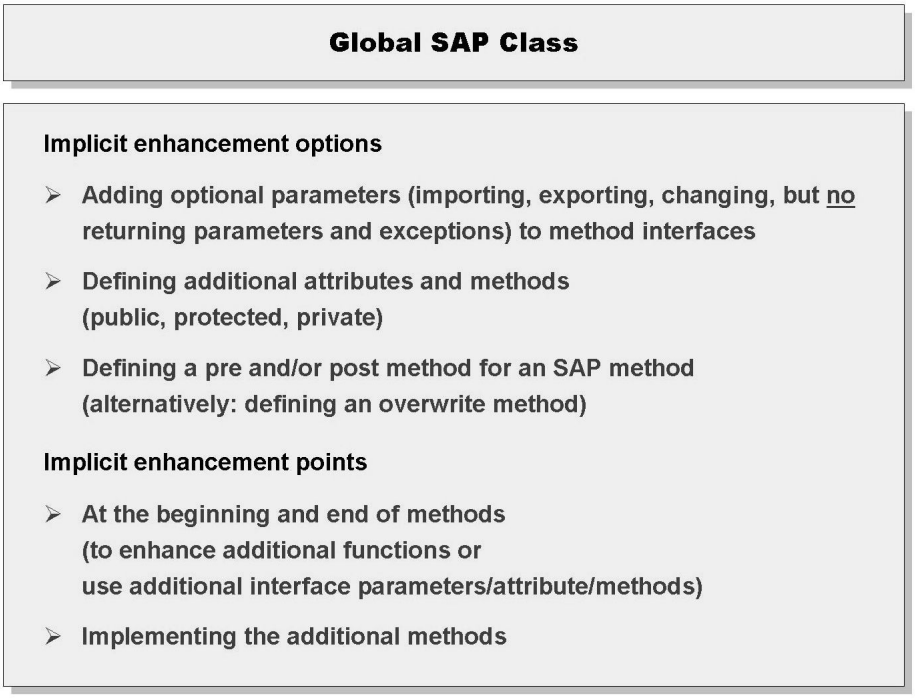

Figure 25: Implicit enhancement points and options in SAP classes

Enhancing interfaces for global SAP methods and

Defining additional attributes/methods for global SAP classes:

1. **Display** the class in the Class Builder.
2. Choose *Class* → *Enhance* and specify an enhancement implementation to add new attributes, methods, and interface parameters to existing methods.
3. In the GUI status, choose *Activate Enhancements*.

Double-click an additional method to branch to the method editor for the implementation.

These newly added interface parameters are generally **optional** and they can be activated in the source code enhancements of the corresponding methods.

New additional attributes and methods can be activated in the source code enhancements of methods of the same global class.

Defining a pre or post or overwrite method for the method of a global SAP class:

1. **Display** the class in the Class Builder
2. Choose *Class* → *Enhance* to specify or to create an enhancement implementation.
3. Use the cursor to mark the required SAP method.
4. Choose *Edit* → *Enhancement Operations* and then choose one of the menu entries: *Insert Pre-Method*, *Insert Post-Method*, or *Add Overwrite Method*
5. Choose the new button in the column “Pre-/Post-/Overwrite-Exit” to implement the corresponding method. To call components of the global class in the method, you have to use the reference variable *me->core_object*, which is set in the local constructor above.
6. Save the method.
7. In the GUI status, choose *Activate Enhancements*

For each SAP method, you can define a pre-method or a pre-method and post method. As an alternative, you can create an overwrite method, which replaces the SAP method.

These methods are automatically called before, after, or instead of the SAP method. They are instance methods of an automatically generated, local class and are available via an instance attribute named CORE_OBJECT. This attribute is a reference to the corresponding current instance in the application program that this SAP class is being used.

Enhancing source code in methods of global SAP classes:

1. **Display** the source code of the method
2. In the GUI status, choose the enhancement button
3. Choose the menu options *Edit → Enhancement Operations → Show Implicit Enhancement Options* to show the implicit enhancement options.
4. In the editor, use the context menu of one of the implicit enhancement points displayed to create an enhancement implementation.
5. Insert the source code.
6. In the GUI status, choose *Activate Enhancements*.

The following figure shows which implicit enhancement points and options are available in local SAP classes.

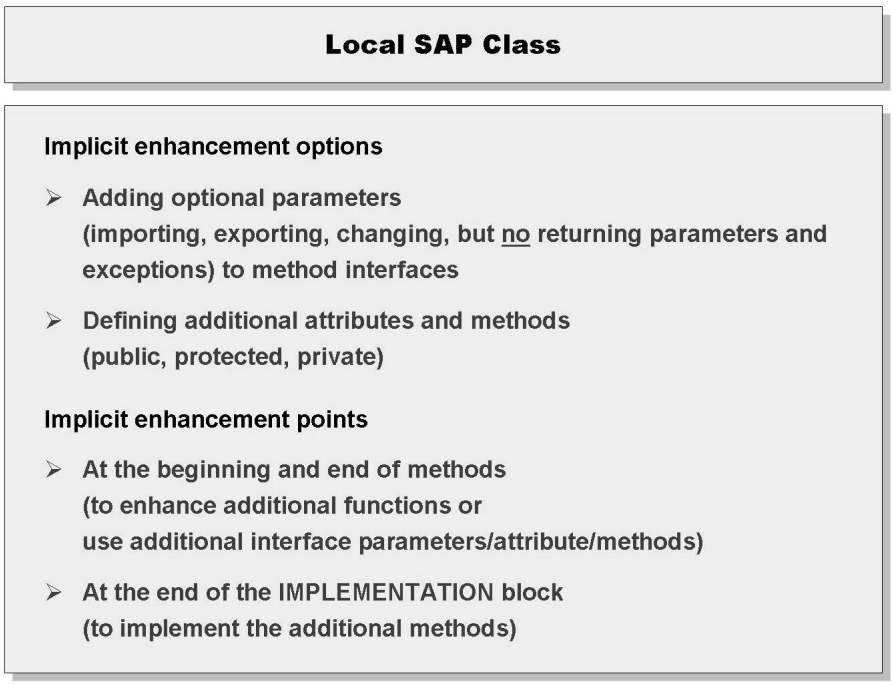

Figure 26: Implicit enhancement points and options in local SAP classes

To use implicit enhancement points and options in local SAP classes, implement an enhancement implementation. To do so, proceed as follows:

Enhancing local SAP classes:

1. **Display** the source code of the local class.
2. In the GUI status, choose the *enhancement* button.
3. Choose *Edit* → *Enhancement Operations* → *Show Implicit Enhancement Options* to show the implicit enhancement options.
4. In the editor, use the context menu of one of the implicit enhancement options/points displayed to create an enhancement implementation.

Hint: You have to scroll to the right end of the editor to see what implicit enhancements are given. If you use an enhancement implementation for public section, for example, you have to choose the right quotation marks for that. Then do not write `PUBLIC SECTION` in the enhancement implementation. Proceed in the same way in all other enhancement possibilities.

5. Insert the source code.
6. In the GUI status, choose “Activate Enhancements”.

For the enhancement of a method interface, the introductory additions IMPORTING, EXPORTING and CHANGING are **not** to be specified, even if they are missing from the original method declaration.

The following figure shows which other implicit enhancement points are available in SAP objects.

Other implicit enhancement points

- at the end of a structure (type) declaration (before `"END OF ...`")
  (to include additional fields)
- at the beginning and end of subroutines
  (to enhance additional functions)
- at the end of includes
  (to implement additional functions and subroutines)

Figure 27: Other implicit enhancement points

Using the other implicit enhancement points:

1. **Display** the source code of the corresponding SAP object.
2. In the GUI status, choose the *enhancement* button.
3. Choose the menu options *Edit → Enhancement Operations → Show Implicit Enhancement Options* to show the implicit enhancement options.
4. In the editor, use the context menu of one of the implicit enhancement points displayed to create an enhancement implementation.
5. Insert the source code.
6. In the GUI status, choose *Activate Enhancements*.

Important: When you enhance a structure (type) declaration (before "END OF ...") you **must** use the syntax

`DATA:` <Additional field1> `TYPE` <Type1>, <Additional field2> `TYPE` <Type2>, ... **.**

because otherwise the program would be syntactically incorrect.

Hint: Note that implicit enhancement points and sections are not possible for elements of the central SAP Basis.

What has to be done after an upgrade?

The enhancement implementations are done without modifications. However, after the enhanced programs are newly released in an upgrade, you have to make an adjustment to "return" the enhancement implementations back to the program. To do that, use the enhancement browser transaction SPAU_ENH. Please notice that one implementations can have several elements. In the enhancement browser list double-click the enhancement implementation which has to be adjusted to see the element list (conflict list). Each element with status yellow or red light has to be adjusted tool-aided or manually separately. To do that change to change mode. Use the *Long Text* button to get information about what has to be done. The green lighted elements where adjusted automatically. After all elements are adjusted you can adjust the complete implementation by using the *Adjust Enhancement* Button. Don't forget to activate your work.

Exercise 2: Explicit Enhancement Points and Sections

Exercise Objectives

After completing this exercise, you will be able to:

- use explicit enhancement points and enhancement sections to enhance or replace SAP source code

Business Example

You want to use explicit enhancement points and enhancement sections to enhance or replace SAP source code without making modifications.

Task:

Enhancing and replacing SAP source code

1. Analyze the source code of the program **BC427_##_EPS** (## = group number).
2. Use the explicit enhancement point BC427_##_EP1 to react to the input values '6' and '7' the same way as the SAP program does.

   Name the enhancement implementation to be created **ZBC427_##_EPS**.
3. Use the explicit enhancement section BC427_##_ES1 to output the corresponding text in the list (instead of the user message).

Solution 2: Explicit Enhancement Points and Sections

Task:

Enhancing and replacing SAP source code

1. Analyze the source code of the program **BC427_##_EPS** (## = group number).

   a) -

2. Use the explicit enhancement point BC427_##_EP1 to react to the input values '6' and '7' the same way as the SAP program does.

   Name the enhancement implementation to be created **ZBC427_##_EPS**.

   a) When you enter '6' and '7', the system should output 'Saturday' and 'Sunday'.

      Proceed as described in this lesson.

3. Use the explicit enhancement section BC427_##_ES1 to output the corresponding text in the list (instead of the user message).

   a) Proceed as described in this lesson.

Exercise 3: Implicit Enhancement Points and Options in SAP Function Modules

Exercise Objectives

After completing this exercise, you will be able to:

- use implicit enhancement options and enhancement points to enhance SAP function modules.

Business Example

You want to use implicit enhancement options and enhancement points to enhance SAP function modules without making modifications.

Task:

Enhancing SAP function modules

1. Familiarize yourself with the function of SAP function module **BC427_##_CALC_PRICE** .

   The function module calculates the gross price for the imported net price and returns it as the EXPORT parameter.

2. Without making modifications, define an additional (optional) IMPORT parameter named **im_discount** (type I) that can be used to transfer a discount (in %) to the function module. Name your enhancement implementation **ZBC427_##_CALC_PRICE_INTERFACE** (## = group number).

   Define an EXPORT parameter named **ex_discprice** (type BC427_PRICE) that the function module can use to return the discounted gross price.

   Save your enhancements.

3. Enhance the source code of the function module to ensure that the discounted price can be calculated into ex_discprice using the IMPORT parameter im_discprice. Name your enhancement implementation **ZBC427_##_CALC_PRICE_SOURCE** (## = group number).

4. Activate your two enhancement implementations.

5. Copy the program **BC427_IEP_FM_TEMPLATE** (name of the target program: **ZBC427_##_IEP_FM** (## = group number)).

   In your copied program, call your enhanced function module to ensure that it also returns the discounted gross price in addition to the standard gross price. (The user input option for discount has already been implemented using the parameter **discount**.)

   The system should then output the two gross prices.

Solution 3: Implicit Enhancement Points and Options in SAP Function Modules

Task:

Enhancing SAP function modules

1. Familiarize yourself with the function of SAP function module **BC427_##_CALC_PRICE** .

   The function module calculates the gross price for the imported net price and returns it as the EXPORT parameter.

   a) -

2. Without making modifications, define an additional (optional) IMPORT parameter named **im_discount** (type I) that can be used to transfer a discount (in %) to the function module. Name your enhancement implementation **ZBC427_##_CALC_PRICE_INTERFACE** (## = group number).

   Define an EXPORT parameter named **ex_discprice** (type BC427_PRICE) that the function module can use to return the discounted gross price.

   Save your enhancements.

   a) Proceed as described in this lesson.

3. Enhance the source code of the function module to ensure that the discounted price can be calculated into ex_discprice using the IMPORT parameter im_discprice. Name your enhancement implementation **ZBC427_##_CALC_PRICE_SOURCE** (## = group number).

   a) Proceed as described in this lesson.

4. Activate your two enhancement implementations.

   a) For example, you can use the activation button in the function module editor to do this.

5. Copy the program **BC427_IEP_FM_TEMPLATE** (name of the target program: **ZBC427_##_IEP_FM** (## = group number)).

   In your copied program, call your enhanced function module to ensure that it also returns the discounted gross price in addition to the standard gross price. (The user input option for discount has already been implemented using the parameter **discount**.)

   The system should then output the two gross prices.

   a) -

Exercise 4: Implicit Enhancement Points in SAP Structure Definitions and Subroutines

Exercise Objectives

After completing this exercise, you will be able to:

- use implicit enhancement points for enhancing SAP structure definitions and subroutines.

Business Example

You want to use implicit enhancement points for enhancing SAP structure definitions and subroutines.

Task:

Enhancing SAP structure definitions and subroutines

1. Analyze the source code of the program **BC427_##_IEP_STRFORM** (## = group number).
2. Use the corresponding implicit enhancement point to add the fields **distance** and **distid** to the structure definition **wa_conn** (flight distance/unit, type assignment using the SPFLI fields of the same name).

   Name your enhancement implementation **ZBC427_##_IEP_STRFORM** (## = group number).
3. In the subroutine **display_conn** , the formal parameter **f_conn** now contains the two additional fields distance and distid due to its type assignment via wa_conn. Use the corresponding implicit enhancement point to output these fields.

Solution 4: Implicit Enhancement Points in SAP Structure Definitions and Subroutines

Task:

Enhancing SAP structure definitions and subroutines

1. Analyze the source code of the program **BC427_##_IEP_STRFORM** (## = group number).

   a) -

2. Use the corresponding implicit enhancement point to add the fields **distance** and **distid** to the structure definition **wa_conn** (flight distance/unit, type assignment using the SPFLI fields of the same name).

   Name your enhancement implementation **ZBC427_##_IEP_STRFORM** (## = group number).

   a) The implicit enhancement point to be used is at the end of the structure definition (before "`END OF ...`").

      Proceed as described in this lesson.

3. In the subroutine **display_conn** , the formal parameter **f_conn** now contains the two additional fields distance and distid due to its type assignment via wa_conn. Use the corresponding implicit enhancement point to output these fields.

   a) The implicit enhancement point to be used is at the end of the subroutine.

      Proceed as described in this lesson.

Lesson Summary

You should now be able to:

- Describe which enhancement options are available in the Enhancement Framework
- List the points in SAP objects where implicit enhancement points and options are available and use them to enhance the SAP functions without making modifications
- Find implemented explicit enhancement points and sections in SAP objects and use them to enhance or replace SAP functions

Unit Summary

You should now be able to:

- Describe which enhancement options are available in the Enhancement Framework
- List the points in SAP objects where implicit enhancement points and options are available and use them to enhance the SAP functions without making modifications
- Find implemented explicit enhancement points and sections in SAP objects and use them to enhance or replace SAP functions

Internal Use SAP Partner Only

Internal Use SAP Partner Only

Unit 4

Business Add-Ins (BAdIs)

Unit Overview

This unit provides information about the runtime architecture and the call syntax. It also provides information about searching for and using classic and new BAdIs.

Unit Objectives

After completing this unit, you will be able to:

- Describe the integration of BAdIs into the Enhancement Framework
- Describe the idea and the architecture of classic and new BAdIs
- Explain the implementation of the new BAdI technology by SAP
- Find classic and new BAdIs in SAP programs and use them for the enhancement

Unit Contents

Lesson: Business Add-Ins (BAdIs)

Lesson Overview

This lesson provides information about how to search for and use classic and new BAdIs.

Lesson Objectives

After completing this lesson, you will be able to:

- Describe the integration of BAdIs into the Enhancement Framework
- Describe the idea and the architecture of classic and new BAdIs
- Explain the implementation of the new BAdI technology by SAP
- Find classic and new BAdIs in SAP programs and use them for the enhancement

Business Example

You want to use Business Add-Ins (BAdIs) to enhance your SAP software without making modifications.

Overview

This section describes why SAP implemented the new BAdI technology as of *SAP NetWeaver 7.0* and gives an overview of how the classic and new BAdIs are integrated into the Enhancement Framework.

The Business Add-In technology (BAdI technology) has existed since Release 4.6 and is a modern enhancement technology for SAP developers who want to provide the option to branch to customer functions in their programs. Enhancements implemented by SAP using the BAdI technology are called BAdIs.

As of *SAP NetWeaver 7.0*, SAP implemented the new BAdI technology and intends to use it for future enhancements. The following figure shows the reasons for implementing the new BAdI technology.

Why new BAdI Technology?

- To improve performance
- To implement additional functions
  - Enhanced filter concept
  - Option to inherit attributes from sample implementation classes (selective method redefinition)
  - ...
- To integrate into a new Enhancement Framework (together with enhancement points and enhancement sections)

Figure 28: Reasons for the new BAdI technology

In the Enhancement Framework available as of *SAP NetWeaver 7.0*, the classic and new BAdIs are managed together with the enhancement options, points and sections.

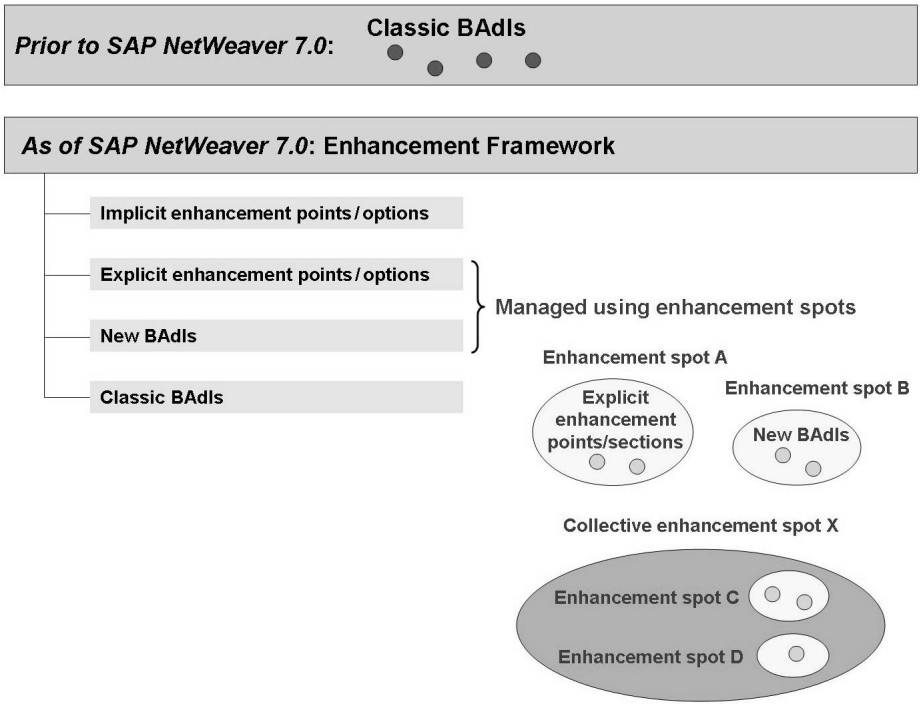

Figure 29: Overview: BAdIs in the Enhancement Framework

Classic BAdIs are not fixed, but new BAdIs always belong to enhancement spots and they are managed by these spots.

Since there are still so many enhancements that are implemented using classic BAdI technology, these classic BAdIs are still very important. Therefore, we want to describe this technology in more detail in the following section to ensure that you can search for and use these enhancements in SAP programs.

Classic BAdIs

The following figure shows how a program exit that is implemented using the classic BAdI technology works.

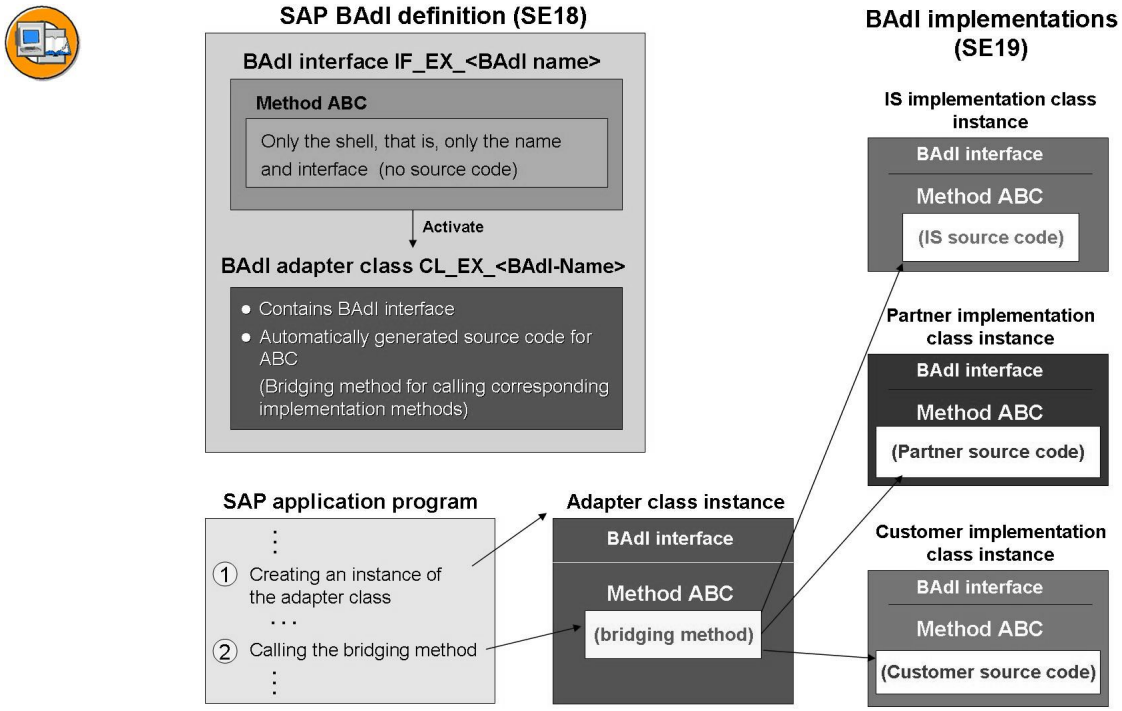

Figure 30: Classic BAdI program exit: Architecture

A BAdI definition and a BAdI interface is specified in transaction SE18. This automatically generates a BAdI adapter class and the corresponding bridging method, which the SAP application calls to branch to the relevant program. The only task of this bridging method is to call existing active implementation methods of industry solutions, partners, and customers one after the other. The sequence in which they are called is not specified.

The following figure shows the syntax that is used to prepare and call the bridging method in the SAP program.

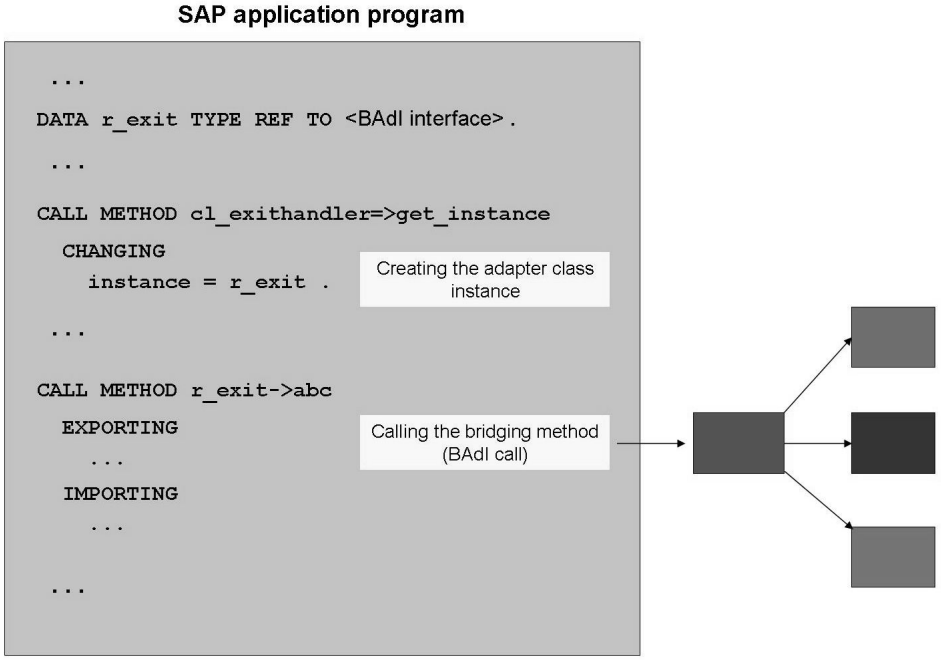

Figure 31: Classic BAdI program exit: Call syntax in the SAP program

In the SAP program, after you define the BAdI and generate the adapter class including the bridging method, a type is assigned to a reference variable using the BAdI interface.

When the static method GET_INSTANCE of the standard class CL_EXITHANDLER is called, an instance of the adapter class (BAdI instance) is created.

The bridging method in the BAdI instance is then called and it branches to the relevant program.

The following two figures show how to find the classic BAdI in SAP programs and how to use them for enhancements.

Free search: SE84

⇨ List of freely selected BAdIs

Application-related search: SE81 → SE84

⇨ List of application-related BAdIs

Program-related search:

1. Program global search for 'GET_INSTANCE'
2. Double-click the reference variable to navigate to the variable definition.
3. Double-click the BAdI interface to navigate to the interface.
4. Where-used list in classes → CL_EX_<BAdI-Name>
5. Read BAdI documentation in SE18.

It may also be necessary to perform step 1 in the function modules and methods called.

Alternative to 1. + 2.:

In the debugger, set a breakpoint for method GET_INSTANCE → F8.
In the program, search for definitions of the reference variables determined

Figure 32: Searching for classic BAdI program exits

Steps to create a BAdI implementation (BAdI use):

1. Display the BAdI definition.
2. Choose the menu options *Implementation* → *Create* (→ SE19).
3. Specify a name for the BAdI implementation.
4. Double-click methods of the implementation class to implement them.
5. Activate the methods and the implementation class.
6. Activate the BAdI implementation.

Figure 33: Using classic BAdI program exits

New BAdIs

The new BAdI technology works in the same way as the classic technology. However, adapter classes are no longer created, which means the SAP application program does not have to instantiate them. Instead, at the runtime of the application program, the system generates a BAdI handle in the kernel, which performs the same function as the adapter class, but which calls the available implementation methods much more efficiently.

The following figure shows the architecture of a program exit that is implemented using the new BAdI technology.

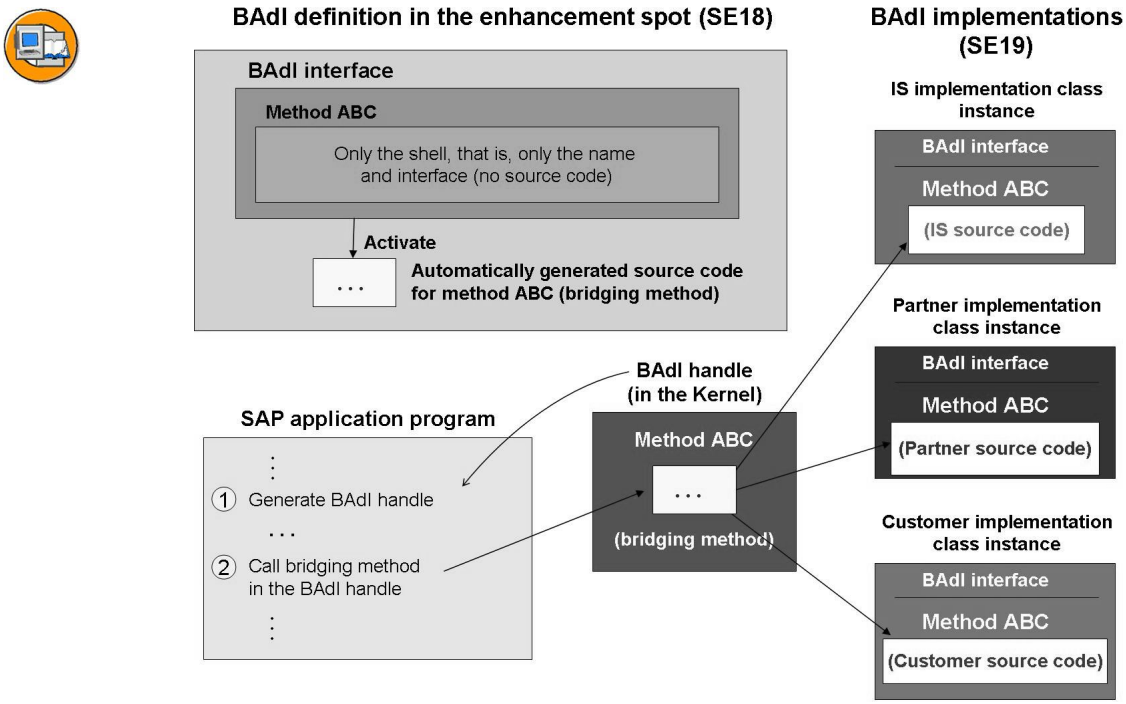

Figure 34: New BAdI program exit: Architecture

The following figure shows the relevant syntax of the SAP program calling the BAdI.

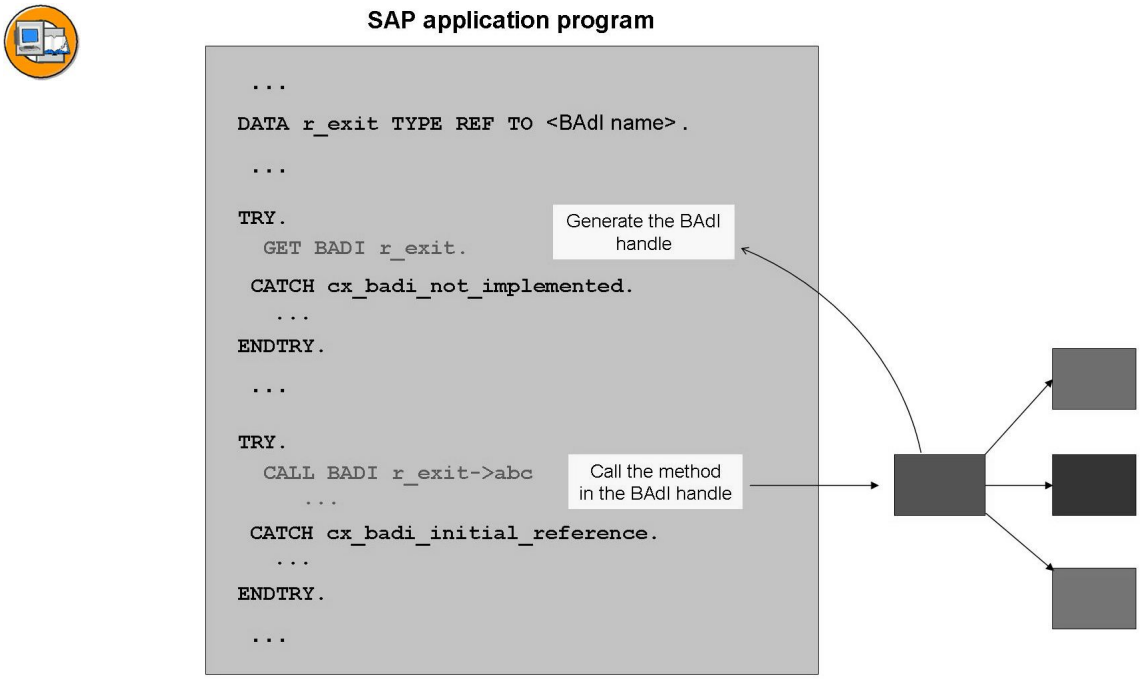

Figure 35: New BAdI program exit: Call syntax in the SAP program

In contrast to the classic BAdI technology, a type is assigned to the reference variable in the SAP program using the BAdI instead of the BAdI interface.

The BAdI handle is generated using the statement `GET BADI` specifying the reference variables. If no active implementation of the BAdI is found, the exception *cx_badi_not_implemented* is triggered.

The system branches to the relevant program when the bridging method in the BAdI handle is called (`CALL BADI`). If an initial handle reference is used for this (for example, because `GET BADI` was not successful), the exception *cx_badi_initial_reference* is triggered.

Knowing the syntax required in the SAP program allows you to perform the program-related search for new BAdIs. The following two figures show you how to search for and use the new BAdIs with program exits.

Free search: SE84

⇨ List of freely selected BAdIs or enhancement spots

Application-related search: SE81→ SE84

⇨ List of application-related BAdIs or enhancement spots

Program-related search:

1. Program global search for 'GET BADI'.
2. Double-click the reference variable to navigate to the variable definition.
3. Double-click the BAdI name to navigate to the display for the corresponding enhancement spot.

It may also be necessary to perform step 1 in the function modules and methods called.

Alternative to 1. + 2.:

In the debugger, set a breakpoint for the statement GET_BADI → F8.
In the program, search for definitions of the reference variables determined.

Figure 36: Searching for new BAdI program exits

The following figure shows you an enhancement spot with a (new) BAdI Definition.

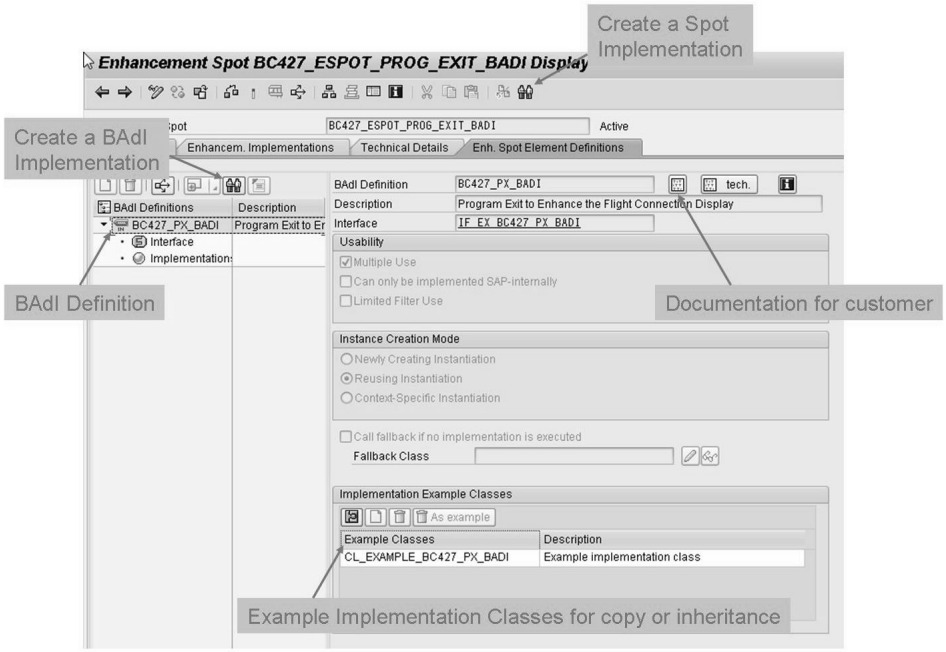

Figure 37: Enhancement Spot with BAdI Definition

To use a BAdI that you have found, you must implement an **enhancement implementation** (an implementation of the higher-level enhancement spot). This implements a **BAdI implementation** for each BAdI in the enhancement spot.

Steps to create a BAdI implementation (BAdI use):

1. **Display the corresponding enhancement spot.**
2. **Choose *Implement Enhancement Spot* (F6) and create the enhancement implementation.**
3. **Specify a name for the enhancement implementation.**
4. **Specify a name for the BAdI implementation(s).**
5. **Maintain attributes for the BAdI implementation(s).**
6. **In the navigation area for the required BAdI, choose the component *Implementing class*.**
7. **Specify the name of the implementing class and choose *Change***
   (This may be inherited from the sample class, or you can copy it.)
8. **Double-click the method(s) to implement or adjust them.**
9. **Activate the method and all corresponding data, including the enhancement implementation.**

Figure 38: Using new BAdI program exits

The following figure shows you an enhancement spot implementation with a (new) BAdI implementation.

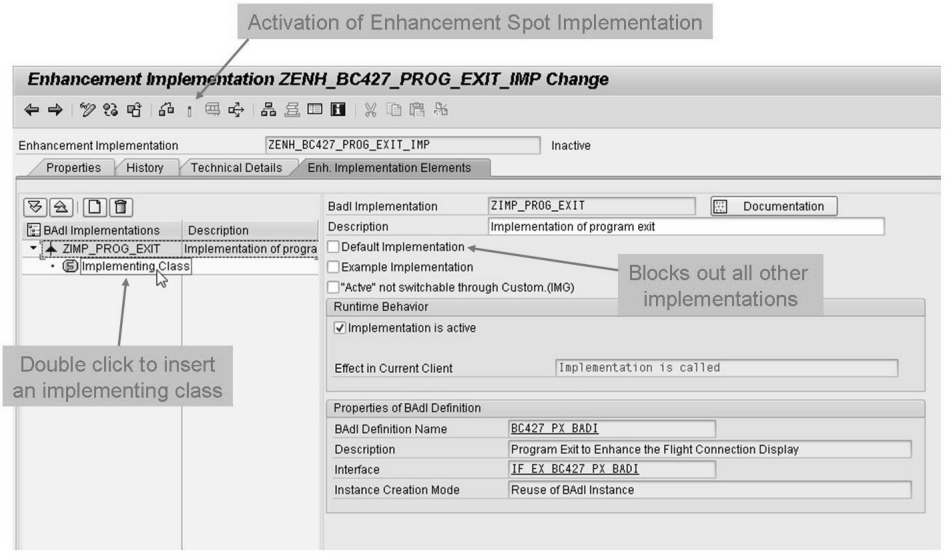

Figure 39: Enhancement Spot Implementation with BAdI Implementation

Filter-dependent BAdIs

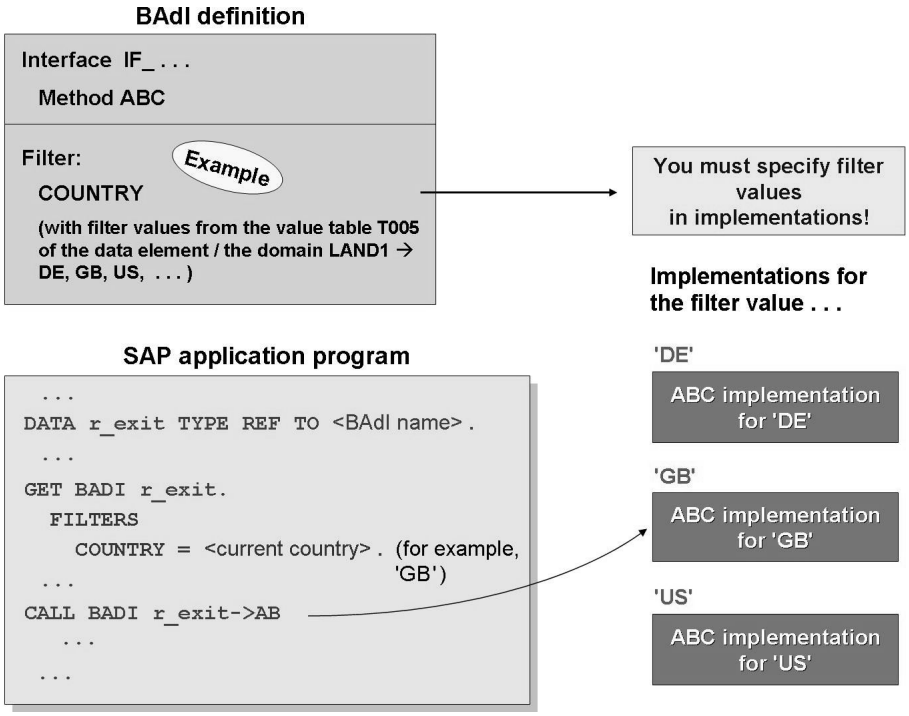

Figure 40: Filter-dependent BAdIs

The figure above illustrates the concept of filter-dependent BAdIs, which corresponds to the classic BAdI concept. However, the functions have been enhanced. Numeric filters can now also be used. For implementations, you can now specify not only single filter values, but also filter conditions by using the operators <>, >, <, >=, <=, CP, NP. You can also define multiple filters for a BAdI.

The following figure shows the principle of a menu exit that is implemented using the new BAdI technology. Exits of this type can be used only once.

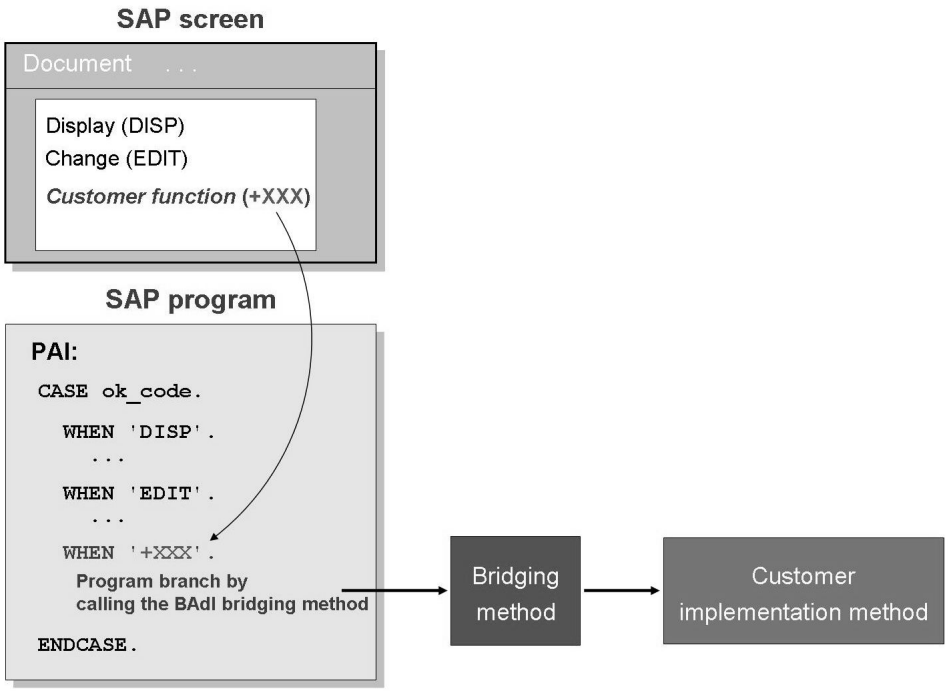

Figure 41: BAdI menu exits: Principle

Menu exits provide customers with the option of implementing additional menu entries including customer functions in SAP screens without making modifications. In the corresponding GUI status, the developer of the SAP screen prepared a menu option, which is not yet visible, with a specified function code (always begins with +) that can already be caught in the PAI of the screen using a BAdI program exit.

When a customer implements a BAdI of this type, in addition to specifying the text for the additional menu option, they must also specify the corresponding customer function in the form of a program exit. This activates the additional entry in the menu of the SAP screen and makes it visible. If the user chooses an entry, the customer function is then processed.

The following figure provides details about the situation described above.

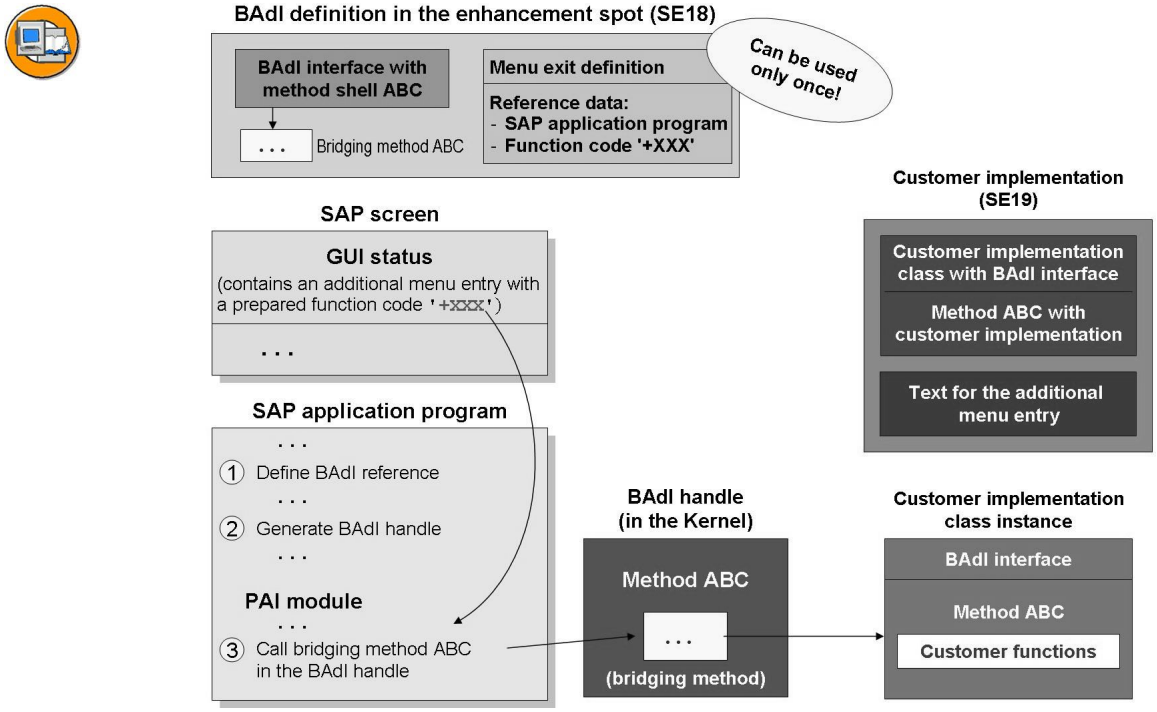

Figure 42: BAdI menu exits: Architecture and details

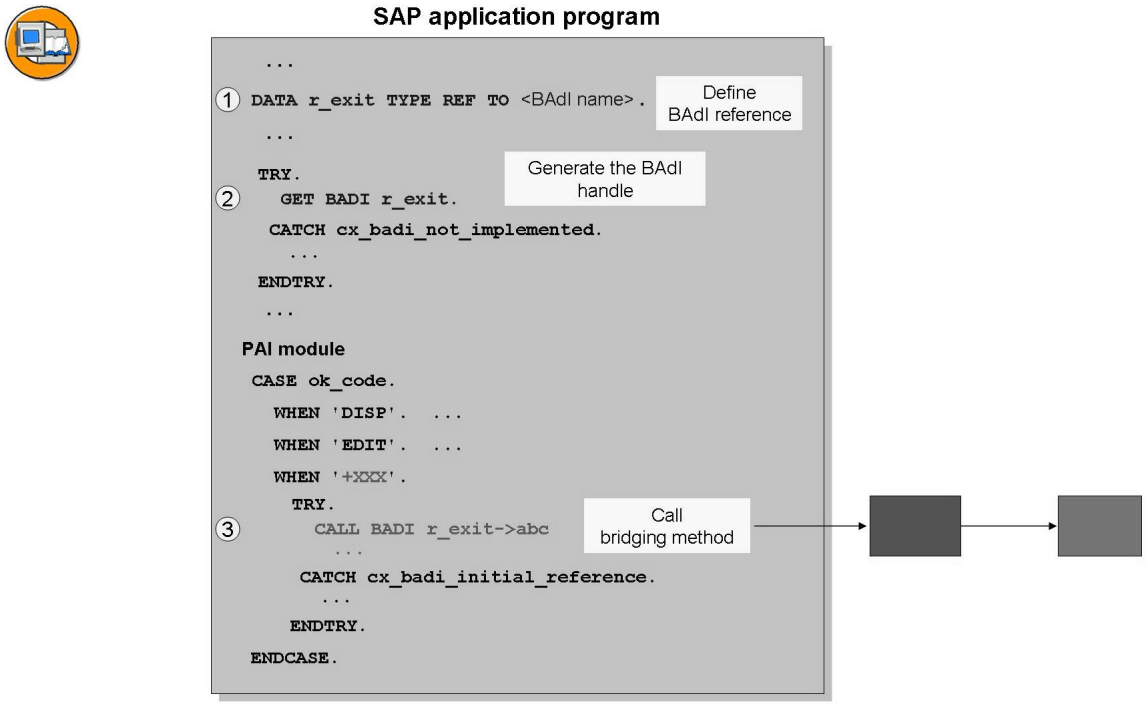

Figure 43: BAdI menu exits: Call syntax in the SAP program

The figure above shows the call syntax required in the corresponding SAP program. It is identical to the call syntax described above for program exits because a program exit plays the deciding role here.

Searching for and using BAdI menu exits in an SAP screen works as follows:

1. In the GUI status of the screen, search for the function code that begins with +.
2. In the PAI modules of the screen, search for the corresponding function code.
3. In the CALL-BADI command implemented there, double-click the specified reference variable to navigate to its definition.
4. Double-click the BAdI used for the type assignment to navigate to the display of the corresponding enhancement spot including the BAdI.
5. Read the BAdI document and, if necessary, create the spot and BAdI implementation as described above.
6. Implement the program exit available in the BAdI as described above.
7. Double-click the menu exit available in the BAdI to implement it; specify text for additional menu entries (see the figure below).
8. Activate all objects.

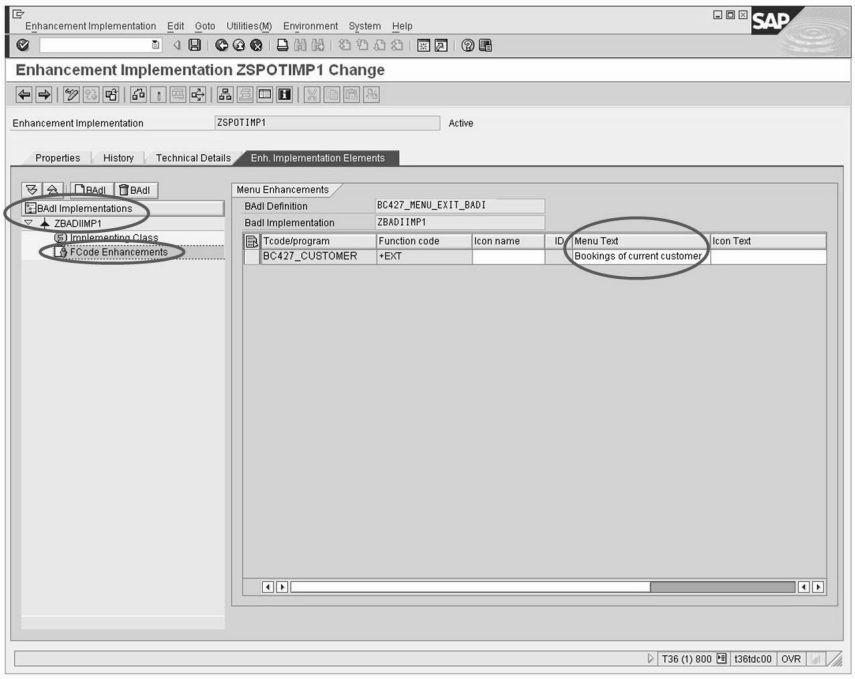

Figure 44: BAdI menu exits: Implementation

The following figure shows the principle of a screen exit that is implemented using the new BAdI technology. Similar to menu exits, exits of this type can be used only once.

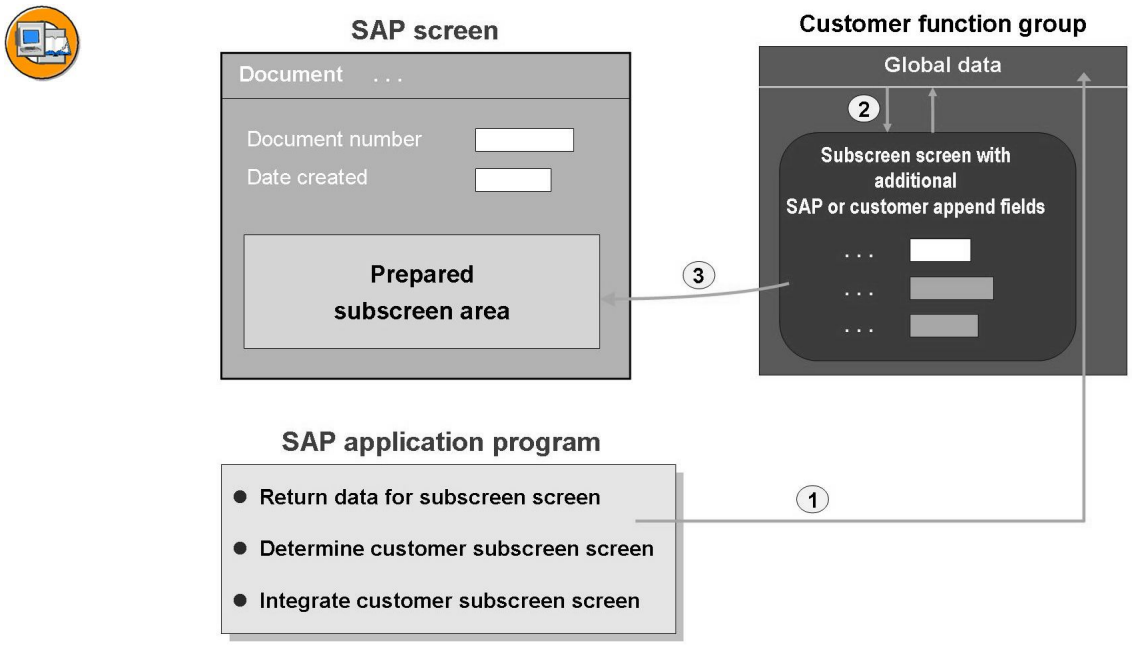

Figure 45: BAdI screen exits: Principle

Screen exits provide customers with the option to also display additional fields for the displayed data record in the SAP screen without making modifications. Using the screen exits for the additional display of the customer append fields for the corresponding data record is particularly interesting.

The developer of the SAP screen reserved a subscreen area to store a customer subscreen screen. The customer subscreen screen must be defined in a customer function group and it should contain the additional fields to be displayed. In the corresponding customer implementation of the BAdI, the reference to the subscreen screen is specified and this subscreen screen is stored in the PBO of the SAP screen in the subscreen area provided.

The following figure shows further details.

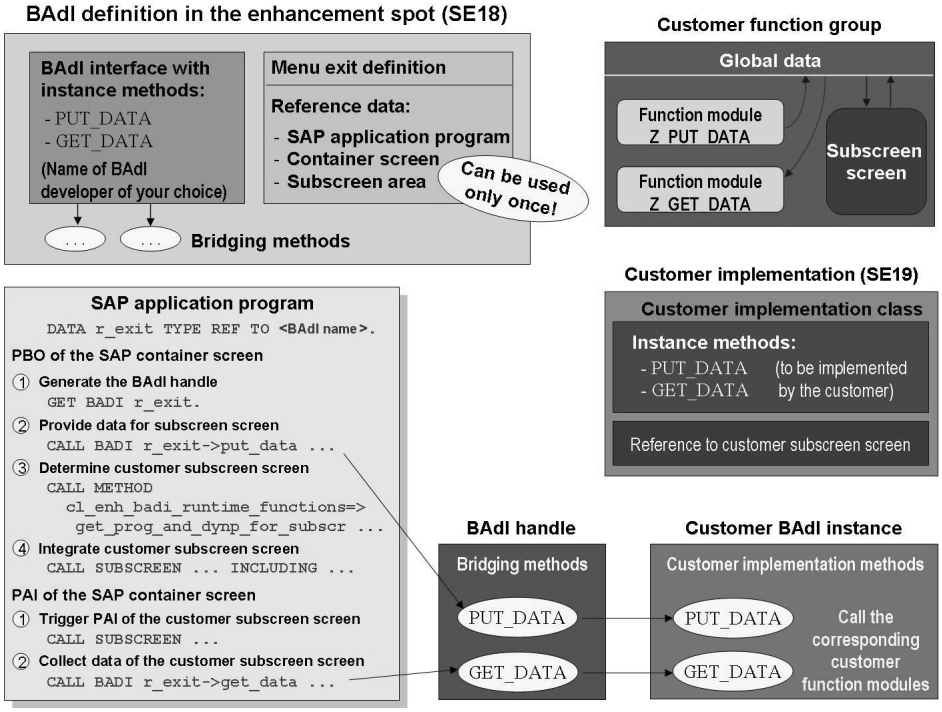

Figure 46: BAdI screen exits: Architecture and call syntax

To ensure that the customer subscreen screen displays corresponding data, the data is transferred from the PBO of the SAP screen (by calling the corresponding bridging method) to the customer *putter* method. The customer must implement this in the BAdI implementation in such a way that the corresponding customer function module of the customer function group is called when the data is transferred, to ensure that this transfers the data to the global data of the function group to provide the subscreen fields. This means that in the customer function group, in addition to the subscreen screen, a corresponding TABLES structure (in the TOP include) for the data exchange and a corresponding function module to be called by the customer puffer method must be implemented to provide the TABLES structure.

If the ready-for-input status of the subscreen fields is provided, a predefined *get* method must also be implement in the customer's BAdI implementation in such a way that it calls the corresponding customer function module of the customer function group, in order to obtain the data transferred from the subscreen screen to the TABLES structure (user entries) and to export it to the individual caller. The SAP program then calls this customer *get* method in the PAI of its screen (using the bridging method) to load and manage the subscreen user entry.

Searching for and using BAdI screen exits works as follows:

1. On the screen, search for a subscreen area (optional pre-search).

2. In the program, search for the GET-BADI command.

3. Double-click the reference variable specified there to navigate to its definition.

4. Double-click the BAdI used for the type assignment to navigate to the display of the corresponding enhancement spot including the BAdI.

5. Read the BAdI document and, if necessary, perform the subsequent steps for the BAdI implementation.

6. Create the function group including the subscreen screen, the TABLES structure (in the TOP include) and the corresponding function modules (see the figure and the description above).

7. Create the spot implementation and the BAdI implementation as described above.

8. In the BAdI implement the available program exits (the *put* and *get* methods) as described above.

9. In the BAdI, double-click the available screen exit to implement it; specify the subscreen screen and the main program of the function group (see the figure below). The name of the main program is the name of the function group with the prefix “SAPL”.

10. Activate all objects.

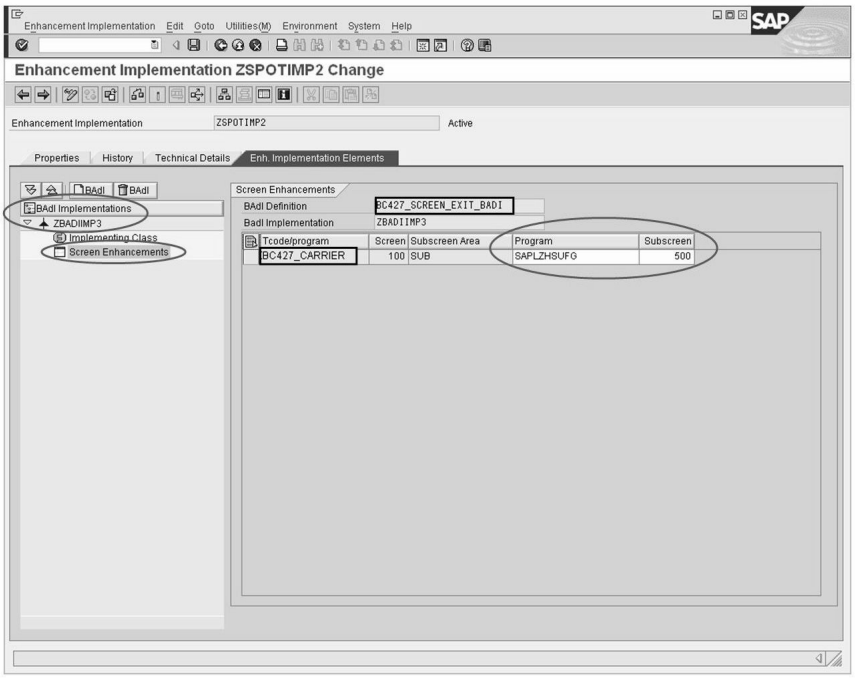

Figure 47: BAdI screen exits: Implementation

Exercise 5: Classic BAdI Program Exits

Exercise Objectives

After completing this exercise, you will be able to:

- find program exits in SAP programs that were implemented using the classic BAdI technology and use them for enhancements.

Business Example

You want to find classic BAdI program exits in SAP programs and use them for enhancements.

Task:

Finding and using classic BAdI program exits

1. You want to add additional columns to the list output of the SAP program **BC427_##_CBD_PX** (## = group number) without making modifications.

   Find out if the SAP program contains a classic BAdI program exit for this purpose.

2. If applicable, use the BAdI program exit that you found, in order to add any additional columns of your choice to the program output.

   Name you implementation **ZBC427_##_CBD_PX**.

Solution 5: Classic BAdI Program Exits

Task:

Finding and using classic BAdI program exits

1. You want to add additional columns to the list output of the SAP program **BC427_##_CBD_PX** (## = group number) without making modifications.

   Find out if the SAP program contains a classic BAdI program exit for this purpose.

   a) See the description in the lesson.

2. If applicable, use the BAdI program exit that you found, in order to add any additional columns of your choice to the program output.

   Name you implementation **ZBC427_##_CBD_PX**.

   a) See the description in the lesson.

Exercise 6: New BAdI Program Exits

Exercise Objectives

After completing this exercise, you will be able to:

- find program exits in SAP programs that were implemented using the new BAdI technology and use them for enhancements.

Business Example

You want to find new BAdI program exits in SAP programs and use them for enhancements.

Task:

Finding and using new BAdI program exits

1. You want to add additional columns to the list output of the SAP program **BC427_##_NBD_PX** (## = group number) without making modifications.

   Find out if the SAP program contains a new BAdI program exit for this purpose.

2. If applicable, use the BAdI program exit that you found, in order to add any additional columns of your choice to the program output.

   Name your

   - enhancement implementation (spot implementation) **ZBC427_##_PXBADI_SPOT**

   - BAdI implementation **ZBC427_##_NBD_PX**

   - implementation class **ZCL_BC427_##_NBD_PX** .

Solution 6: New BAdI Program Exits

Task:

Finding and using new BAdI program exits

1. You want to add additional columns to the list output of the SAP program **BC427_##_NBD_PX** (## = group number) without making modifications.

   Find out if the SAP program contains a new BAdI program exit for this purpose.

   a) See the description in the lesson.

2. If applicable, use the BAdI program exit that you found, in order to add any additional columns of your choice to the program output.

   Name your

   - enhancement implementation (spot implementation) **ZBC427_##_PXBADI_SPOT**

   - BAdI implementation **ZBC427_##_NBD_PX**

   - implementation class **ZCL_BC427_##_NBD_PX** .

   a) See the description in the lesson.

Exercise 7: New BAdI Menu Exits

Exercise Objectives

After completing this exercise, you will be able to:

- find menu exits in SAP programs that were implemented using the new BAdI technology and use them for enhancements.

Business Example

You want to find new BAdI menu exits in SAP programs and use them for enhancements.

Task:

Finding and using new BAdI menu exits

1. The SAP program **BC427_##_NBD_MX** (## = group number) shows the required customer master record on a screen after you enter a customer ID. From the customer display, you want to use an additional menu entry to list the flight bookings of the customer displayed.

   Find out if a BAdI menu exit exists for this.

2. If applicable, use the menu exit that you found to call the program **BC427_CUSTOMER_BOOKINGS** to display the relevant flight bookings of the displayed customer. (On a selection screen, this program provides the parameter **CUSTOMID** to enter a customer ID and list the respective bookings.)

   Name your

   - enhancement implementation (spot implementation) **ZBC427_##_MXBADI_SPOT**

   - BAdI implementation **ZBC427_##_NBD_MX**

   - implementation class **ZCL_BC427_##_NBD_MX** .

Solution 7: New BAdI Menu Exits

Task:

Finding and using new BAdI menu exits

1. The SAP program **BC427_##_NBD_MX** (## = group number) shows the required customer master record on a screen after you enter a customer ID. From the customer display, you want to use an additional menu entry to list the flight bookings of the customer displayed.

   Find out if a BAdI menu exit exists for this.

   a) See the description in the lesson.

2. If applicable, use the menu exit that you found to call the program **BC427_CUSTOMER_BOOKINGS** to display the relevant flight bookings of the displayed customer. (On a selection screen, this program provides the parameter **CUSTOMID** to enter a customer ID and list the respective bookings.)

   Name your

   - enhancement implementation (spot implementation) **ZBC427_##_MXBADI_SPOT**

   - BAdI implementation **ZBC427_##_NBD_MX**

   - implementation class **ZCL_BC427_##_NBD_MX** .

   a) The syntax for calling the program to display the bookings is as follows:

```
SUBMIT bc427_customer_bookings
  WITH customid = im_customid
  AND RETURN.
```


Exercise 8: New BAdI Screen Exits

Exercise Objectives

After completing this exercise, you will be able to:

- find screen exits in SAP programs that were implemented using the new BAdI technology and use them for enhancements.

Business Example

You want to find new BAdI screen exits in SAP programs and use them for enhancements.

Task:

Finding and using new BAdI screen exits

1. The SAP program **BC427_##_NBD_SX** (## = group number) shows details of the specified airline on a screen after you enter an airline. You also want to display the local currency of the airline there.

   Find out if a BAdI screen exit exists for this.

2. If applicable, use the screen exit that you found to also display the field **SCARR-CURRCODE** (local currency of airline).

   Name your

   - enhancement implementation (spot implementation) **ZBC427_##_SXBADI_SPOT**

   - BAdI implementation **ZBC427_##_NBD_SX**

   - implementation class **ZCL_BC427_##_NBD_SX** .

Solution 8: New BAdI Screen Exits

Task:

Finding and using new BAdI screen exits

1. The SAP program **BC427_##_NBD_SX** (## = group number) shows details of the specified airline on a screen after you enter an airline. You also want to display the local currency of the airline there.

   Find out if a BAdI screen exit exists for this.

   a) See the description in the lesson.

2. If applicable, use the screen exit that you found to also display the field **SCARR-CURRCODE** (local currency of airline).

   Name your

   - enhancement implementation (spot implementation) **ZBC427_##_SXBADI_SPOT**

   - BAdI implementation **ZBC427_##_NBD_SX**

   - implementation class **ZCL_BC427_##_NBD_SX** .

   a) See the description in the lesson.

Lesson Summary

You should now be able to:

- Describe the integration of BAdIs into the Enhancement Framework
- Describe the idea and the architecture of classic and new BAdIs
- Explain the implementation of the new BAdI technology by SAP
- Find classic and new BAdIs in SAP programs and use them for the enhancement

Internal Use SAP Partner Only

Unit Summary

You should now be able to:

- Describe the integration of BAdIs into the Enhancement Framework
- Describe the idea and the architecture of classic and new BAdIs
- Explain the implementation of the new BAdI technology by SAP
- Find classic and new BAdIs in SAP programs and use them for the enhancement

Internal Use SAP Partner Only

Internal Use SAP Partner Only

Internal Use SAP Partner Only

Unit 5

Switch Framework

Unit Overview

This unit discusses the general concept and how to use the Switch Framework for activating and deactivating enhancement implementations.

Unit Objectives

After completing this unit, you will be able to:

- Describe what the Switch Framework is and for what it is used
- Use the Switch Framework to activate and deactivate your enhancement implementations

Unit Contents

Lesson: Switch Framework

Lesson Overview

This lesson provides information about how to use the Switch Framework to activate and deactivate your enhancement implementations.

Lesson Objectives

After completing this lesson, you will be able to:

- Describe what the Switch Framework is and for what it is used
- Use the Switch Framework to activate and deactivate your enhancement implementations

Business Example

You want to learn more about the general concept of the Switch Framework, and in particular, you want to be able to use this tool to activate and deactivate your enhancement implementations.

Switch Framework

This lesson provides information about the idea and general concept of the **Switch Framework**. It also provides information on how to use this tool to activate and deactivate your enhancement implementations.

The idea of the Switch Framework is that customers receive all industry solutions as a complete package and that they are able to activate those they want to use. All the other solutions are available but they cannot be used. SAP also decided to use enhancement packages to deliver future developments. Customers may then choose which new function to activate.

The following figure illustrates the concept of the Switch Framework.

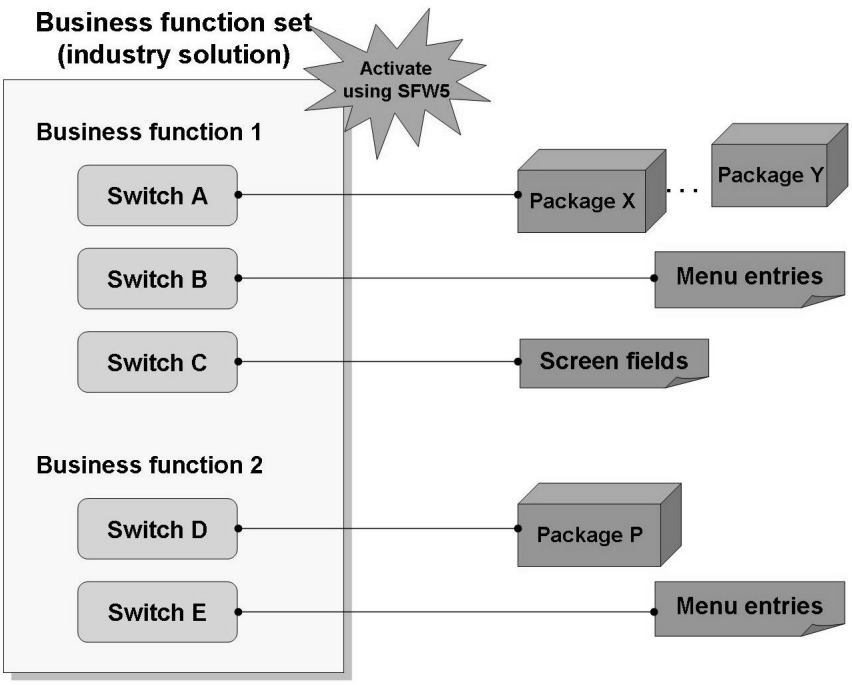

Figure 48: Overview and concept of the Switch Framework

You can use the Switch Framework to create switches and to assign packages, screen elements, and menu entries to them. You can use a business function to group these switches.

One industry solution from SAP is a collection of business functions called a "business function set". You can use transaction SFW5 to activate only one business function set, and/or to activate and deactivate business functions.

Customers can also use the Switch Framework to activate and deactivate enhancement implementations. The following figure illustrates this:

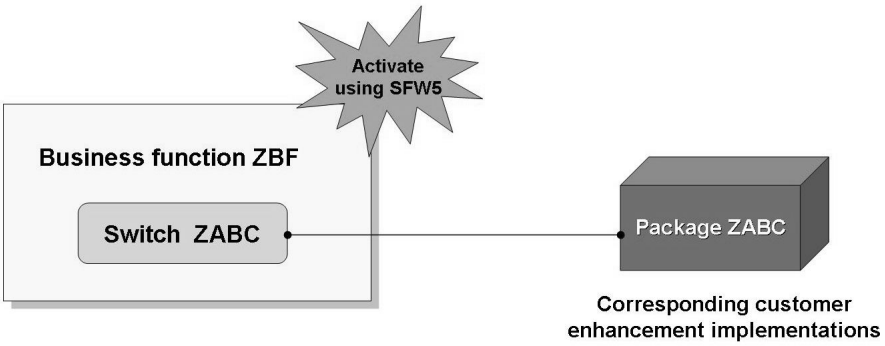

Figure 49: Using the Switch Framework for enhancement implementations

The customer defines a switch to which he assigns the package with the implementations to be activated. The customer also defines a business function, to which the switch is assigned. This can be done in transaction SFW1 and SFW2.

The business function can be activated and deactivated using transaction SFW5. When it is deactivated, all package objects that can be activated (this includes enhancements) become ineffective although they are still available in the system **For the deactivation, it is prerequisite** that the business function is defined as ***reversible*** and that the package does not contain **any dictionary objects**.

It does not make sense to deactivate industry solution components after they were activated, because due to activation all related programs were generated and the related dictionary objects were created. That is why industry solutions and enhancement packages are not reversible.

The following figures show the procedures for the actions described above.

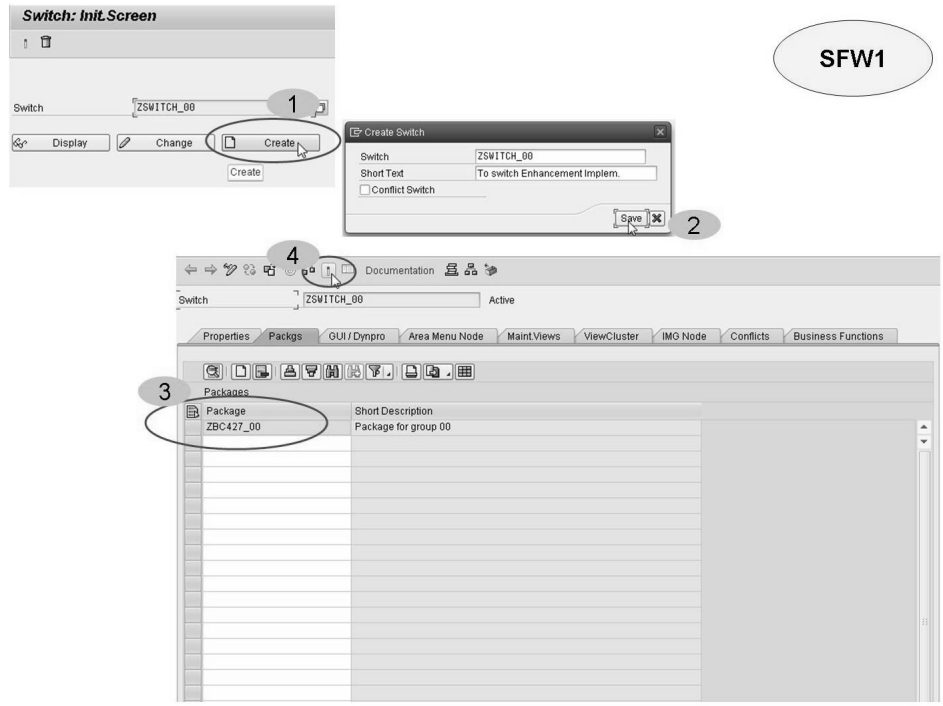

Figure 50: Defining a switch and assigning packages

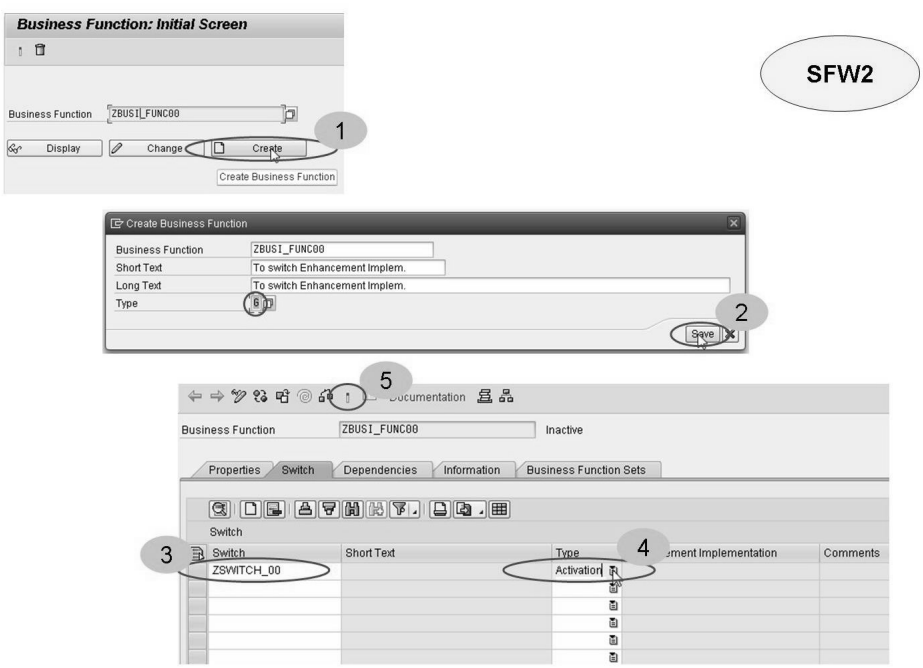

Figure 51: Defining a business function and assigning switches

Because the new business function does not contain any components belonging to the industry solution, choose business function **type G**.

When you assign a switch to a business function, you must specify the assignment type. ***Activation*** and ***Enabling/Standby*** are possible options.

The option *Activation* assigns all switch or package objects to the business function, but the option *Enabling/Standby* assigns only the dictionary objects. Therefore, you should choose *Activation*.

SFW2

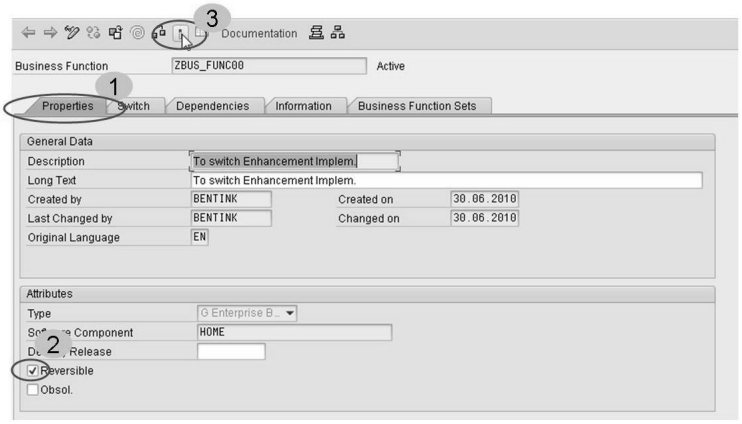

Figure 52: Reversible Business Function

To **deactivate** the business function, select the *reversible* checkbox on the *Properties* tab page. All business functions can be switched on, however, only reversible business functions can be switched off later.

SFW5

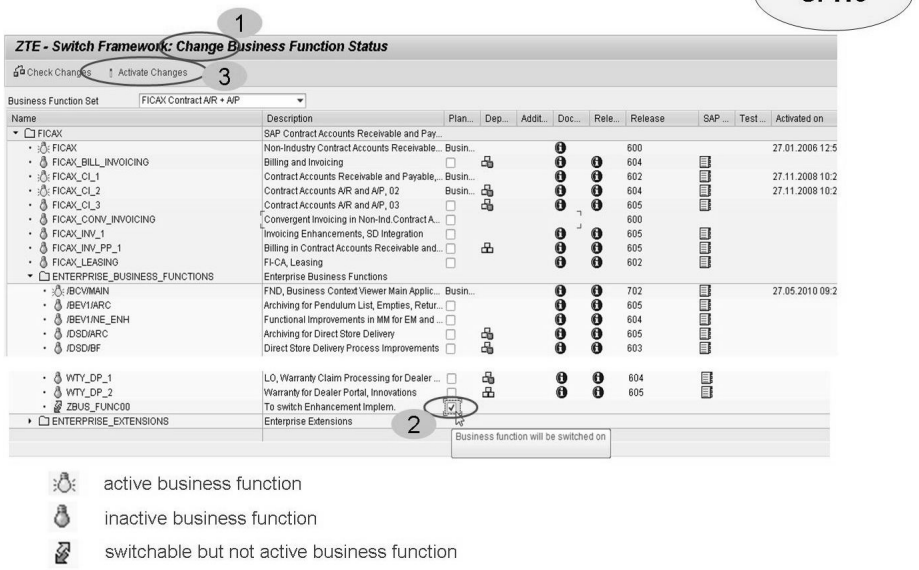

Figure 53: Activating and deactivating business functions

To activate or deactivate a business function, make sure that you have switched to **change mode**.

As you have seen, all enhancement implementations assigned to the enhancement framework are switchable. That also means that all other SAP enhancements, such as Customer Exits, classical Business Add-Ins, User Exits, and Business Transaction Events, are not affected.

However, there is one way to admit these enhancement techniques into the switch. You have to use the implicit enhancement framework technique while implementing.

This will be demonstrated due to the classic Business Add-Ins. Use transaction SE19 to implement the BAdI in the usual manner.

Create the intended customer class as usual.

In the Class Builder switch from editing to display the interface method.

Now use implicit enhancements to enhance the interface method implementation at the beginning or at the end of the method.

You can proceed with all other SAP enhancement techniques that are not related to the enhancement framework in the same way.

This technique should not be used in general but only if you would like to switch these enhancement implementations along with other enhancement implementations.

Is it possible to derive from the coding whether an enhancement implementation is switched off?

After being activated in the coding, the enhancement implementation is always marked as active in the coding. So you cannot see directly in the coding whether the enhancement implementation runs or not. However, you can see in the enhancement implementation whether it is assigned to a switch and what the setting of this switch is. For this, double-click the enhancement implementation name in the coding and choose *properties*.

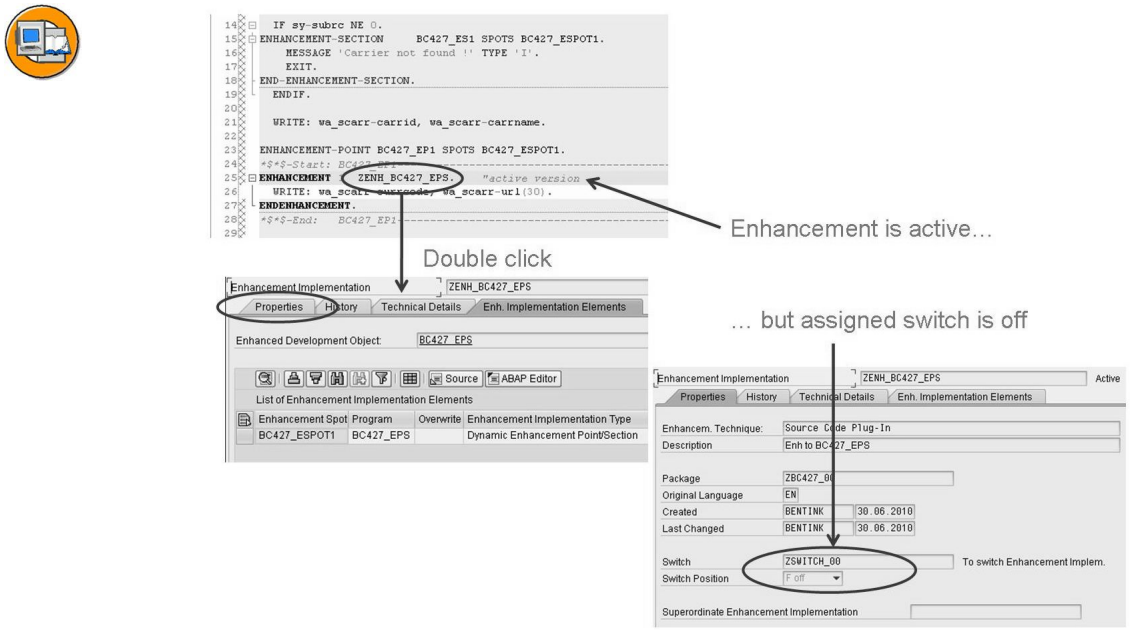

Figure 54: Switched off Enhancements

Exercise 9: Switch Framework

Exercise Objectives

After completing this exercise, you will be able to:

- Use the Switch Framework to activate and deactivate your enhancement implementations

Business Example

You want to be able to activate and deactivate your enhancement implementations at any time.

Task:

Activating and deactivating enhancement implementations

1. Create a switch named **ZSWITCH_##** (## = group number).
2. Assign your package **ZBC427_##** to your switch.
3. Activate your switch.
4. Create a business function named **ZBF_##**.
5. Assign your switch to your business function.
6. Set your business function to *reversible* and activate it.
7. Check whether or not the enhancement implementations stored in your package are still effective.
8. Turn on your business function and activate your settings.
9. Check again whether or not your enhancement implementations are effective.

Solution 9: Switch Framework

Task:

Activating and deactivating enhancement implementations

1. Create a switch named **ZSWITCH_##** (## = group number).
   a) See the information for SFW1.
2. Assign your package **ZBC427_##** to your switch.
   a) See the information for SFW1.
3. Activate your switch.
   a) See the information for SFW1.
4. Create a business function named **ZBF_##**.
   a) See the information for SFW2.
5. Assign your switch to your business function.
   a) See the information for SFW2.
6. Set your business function to *reversible* and activate it.
   a) See the information for SFW2.
7. Check whether or not the enhancement implementations stored in your package are still effective.
   a) -
8. Turn on your business function and activate your settings.
   a) See the information for SFW5.
9. Check again whether or not your enhancement implementations are effective.
   a) -

Lesson Summary

You should now be able to:

- Describe what the Switch Framework is and for what it is used
- Use the Switch Framework to activate and deactivate your enhancement implementations

Unit Summary

You should now be able to:

- Describe what the Switch Framework is and for what it is used
- Use the Switch Framework to activate and deactivate your enhancement implementations

Internal Use SAP Partner Only

Internal Use SAP Partner Only

Course Summary

You should now be able to:

- Provide an overview of the classic enhancement technology and the new enhancement concept *(Enhancement Framework)*
- Use the new options as of *SAP NetWeaver 7.0* to enhance the *ABAP Dictionary*
- Use enhancement points and options and enhancement sections effectively to enhance SAP software
- Find and use enhancements that are determined using BAdI technology
- Use the Switch Framework to activate and deactivate enhancement implementations

Appendix 1

Introduction into ABAP OO

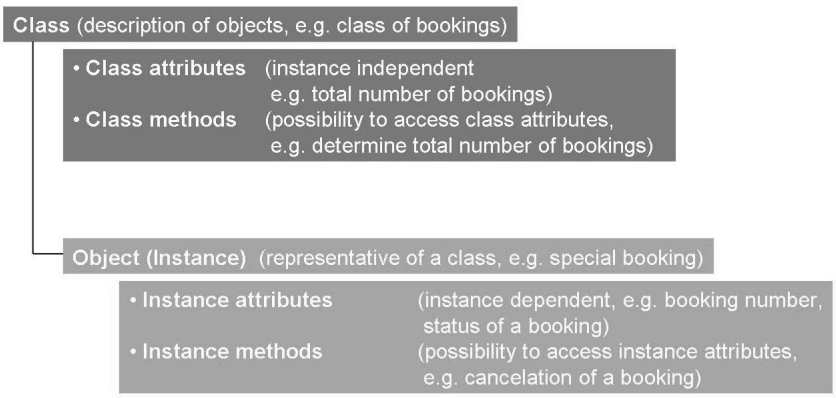

Figure 55: Classes and Objects

The basic approach of object oriented programming is to think in objects. For example a special booking with all the attributes and their special functionality which only work on that attributes.

The technical description of an object is given by its class and the object is an instance of a class. Each object only exists while the program is running.

There are not only object dependent descriptions in the class, but also components which are valid in general and have no special dependence to a concrete object. These components are called static components (class attributes, class methods).

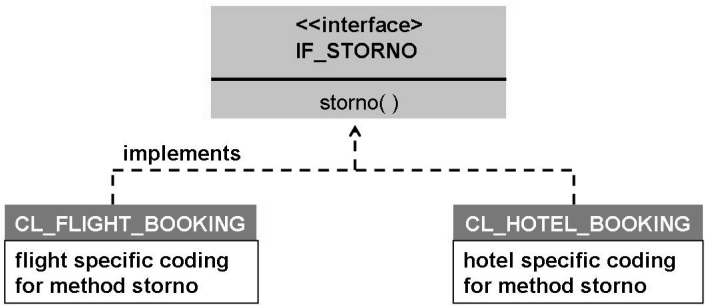

Figure 56: Interfaces

An interface is a collection of formal defined methods without any coding, that means name of methods and their signature.

An interface can be integrated to a class. The class is then called as the implementing class of the interface. In the class you are able to implement all interface methods with their class specific implementation.

If you have several interface implementing classes with their class specific coding you have reached the following: there is only one method definition (in the interface), but several different functionalities (in the implementing classes). In the next but one slide we will see how we can get a special effect using this feature.

```
* Definition of a reference variable
DATA my_pointer TYPE REF TO cl_flight_booking.

* Creating an object
CREATE OBJECT my_pointer [EXPORTING …].

* Calling an instance method using reference variable
CALL METHOD my_pointer->get_flight_booking […].

* Calling a class method using class prefix
CALL METHOD cl_flight_booking=>get_n_o_bookings […].
```

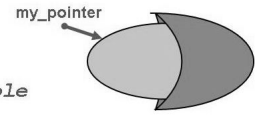

Figure 57: Reference variable

Instances were created during runtime and therefore they have no name. That means you have to call an instance over its reference variable (pointer) which refers to the instance.

First of all you have to define a reference variable in the declaration block of a program with the additional specification REF TO <class>. That means that this reference variable shows during runtime to a specific instance of this class which is not yet created. It doesn't mean that the pointer shows to the class, because a class is not anything physical. It is only a type description.

To create an object of the class, use the defined reference variable. The reference variable then refers to this object automatically.

Now you can call (instance) components of the object using the reference variable as prefix.

To call class components (for example class methods), you don‘t need an object, because the class components are independent from any object of this class. Use therefore the class prefix to call a class method for example.

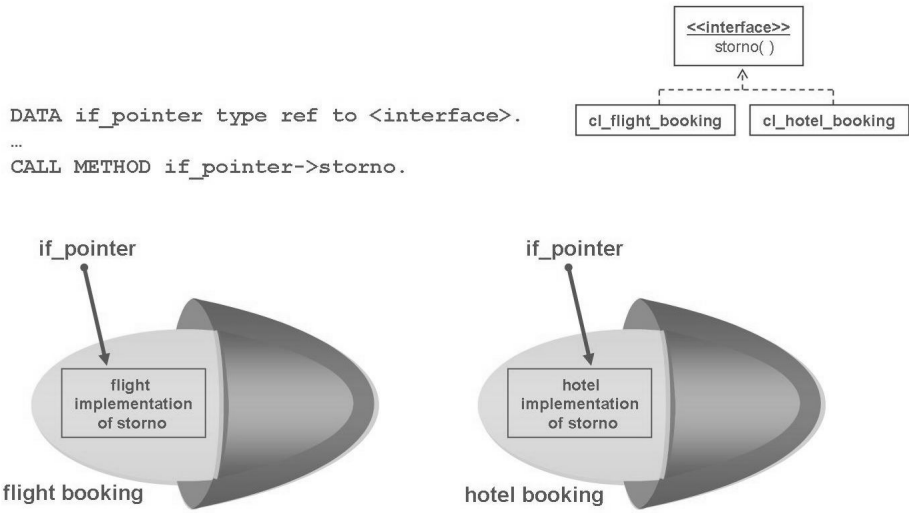

Figure 58: Interface reference

There is no way to instantiate an interface, because interfaces doesn‘t have any coding. That‘s why the statement CREATE OBJECT to a reference that is defined to an interface doesn't make sense.

The implementation of an interface is given by a class which is hierarchically under the interface. So a reference which is defined to an interface is able to refer to an object of the implementing class.

Interface references are only able to call the interface components of an object of the implementing class. It depends on their type definition (static type). The functionality then runs in the context of this implementing class. So, if you have several implementing classes of an interface, you can use the same semantic of calling an interface method for the objects of these classes. But for each object runs different functionality.

Internal Use SAP Partner Only

Index

Internal Use SAP Partner Only

Internal Use SAP Partner Only

Feedback

SAP AG has made every effort in the preparation of this course to ensure the accuracy and completeness of the materials. If you have any corrections or suggestions for improvement, please record them in the appropriate place in the course evaluation.

Internal Use SAP Partner Only