

# Changing the SAP Standard (BC)



**Release 4.6C**



## Copyright

© Copyright 2001 SAP AG. All rights reserved.

No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

Some software products marketed by SAP AG and its distributors contain proprietary software components of other software vendors.

Microsoft®, WINDOWS®, NT®, EXCEL®, Word®, PowerPoint® and SQL Server® are registered trademarks of Microsoft Corporation.

IBM®, DB2®, OS/2®, DB2/6000®, Parallel Sysplex®, MVS/ESA®, RS/6000®, AIX®, S/390®, AS/400®, OS/390®, and OS/400® are registered trademarks of IBM Corporation.

ORACLE® is a registered trademark of ORACLE Corporation.

INFORMIX®-OnLine for SAP and Informix® Dynamic Server™ are registered trademarks of Informix Software Incorporated.

UNIX®, X/Open®, OSF/1®, and Motif® are registered trademarks of the Open Group.






HTML, DHTML, XML, XHTML are trademarks or registered trademarks of W3C®, World Wide Web Consortium, Massachusetts Institute of Technology.

JAVA® is a registered trademark of Sun Microsystems, Inc.

JAVASCRIPT® is a registered trademark of Sun Microsystems, Inc., used under license for technology invented and implemented by Netscape.

SAP, SAP Logo, R/2, RIVA, R/3, ABAP, SAP ArchiveLink, SAP Business Workflow, WebFlow, SAP EarlyWatch, BAPI, SAPPHIRE, Management Cockpit, mySAP.com Logo and mySAP.com are trademarks or registered trademarks of SAP AG in Germany and in several other countries all over the world. All other products mentioned are trademarks or registered trademarks of their respective companies.

## Icons

Icon	Meaning
	Caution
	Example
	Note
	Recommendation
	Syntax

## Inhalt

<b>Changing the SAP Standard (BC)</b>	<b>7</b>
<b>Changing the SAP Standard</b>	<b>8</b>
Customizing	9
Personalization	10
Enhancements to the Standard	13
Modifying the Standard	14
Customer Development	15
<b>Personalizing the Application</b>	<b>17</b>
Hiding Screen Elements	18
Defining Default Values for Input Fields	19
Revoking an Element's Ready for Input Status	20
Adjusting Table Controls	21
Simplifying Selection Screens	25
Personalizing the Possible Entries Help	26
Moving Fields on a Screen	29
Changing Field Texts	30
Including Graphics and Text	31
Inserting Pushbuttons for Frequently-used Functions	32
Changing Possible Entries into Radio Buttons	33
Defining Additional Elements on a Screen	34
Simplifying Screens for Specific Users	35
<b>Tools for Personalizing the Application</b>	<b>36</b>
Transaction Variants and Screen Variants	37
Variant Transactions	38
GuiXT	39
Enhancements	40
Modification Assistant	41
Application-specific User Exits	42
Application-specific Field Selection	43
SET/GET Parameters	44
Parameter Transactions	45
Selection Variants	46
<b>Transaction Variants and Screen Variants</b>	<b>47</b>
<b>Transaction Variants</b>	<b>49</b>
Maintenance	50
Requirements	52
Dialog Box	55
Additional Features	57
Transport	59
<b>Starting Transaction Variants</b>	<b>60</b>
<b>Screen Variants</b>	<b>62</b>
Maintenance	63
Additional Features	65
Calling Screen Variants	67

Table Control Layout in Variants .....	68
Transaction Variants and Screen Variants: Restrictions .....	70
Transaction Variants and Screen Variants: Special Features .....	72
GuiXT in Transaction Variants and Screen Variants .....	76
<b>Business Add-Ins .....</b>	<b>77</b>
Business Add-Ins: Architecture .....	79
A Comparison of Different Enhancement Techniques .....	81
Defining Business Add-Ins .....	82
Calling Add-Ins from Application Programs .....	85
Implementing Business Add-Ins .....	87
Filter-Dependent Business Add-Ins .....	89
Multiple Use Business Add-Ins .....	93
Menu Enhancements .....	95
Business Add-Ins: Import Procedure .....	97
<b>Customer Exits .....</b>	<b>99</b>
Enhancements to the SAP Standard with Customer-Exits .....	100
Types of Exits .....	101
Locating Applications that Have Exits .....	102
Creating an Add-On Project .....	104
Activating and Deactivating a Project .....	106
Transporting Add-on Projects .....	107
Creating Customer-Specific Menus .....	108
Creating Customer-Specific Subscreens .....	109
Creating Customer-Specific Function Modules .....	111
Creating Customer-specific Documentation .....	114
<b>The Modification Assistant .....</b>	<b>116</b>
Modifications in Programs .....	118
Modifications in the Screen Painter .....	120
Modifications in the Menu Painter .....	123
Modifying Text Elements .....	127
Modifying and Adding Function Modules .....	128
Modifications in the ABAP Dictionary .....	130
Modifying Tables and Structures .....	131
Modifying Data Element Attributes .....	132
Modifying Documentation .....	133
Disabling the Modification Assistant .....	135
Resetting to the Original .....	136
The Modification Browser .....	137
<b>Upgrade Procedure/R/3 Support Packages .....</b>	<b>140</b>
Adjusting ABAP Dictionary Objects .....	142
Prerequisites .....	143
Procedure .....	144
Preparing to Run Transaction SPDD .....	145
Transaction SPDD .....	146
Adjusting R/3 Repository Objects .....	149

Preparing to Run Transaction SPAU .....	150
Transaction SPAU: General Functions .....	151
Adjustment Category: With Modification Assistant .....	156
Adjusting Programs .....	157
Adjustments in the Screen Painter .....	159
Adjustments in the Menu Painter .....	162
Adjusting Text Elements .....	164
Adjusting Function Modules .....	165
Adjusting Documentation .....	166
Adjustment Category: Without Modification Assistant .....	167
<b>Upgrading from Release 4.5A or Less .....</b>	<b>168</b>
<b>Modifications Made by Several Different People .....</b>	<b>169</b>
<b>Adjusting Other SAP Systems .....</b>	<b>171</b>
<b>Upgrading the Development System .....</b>	<b>172</b>
<b>Upgrading the Production System .....</b>	<b>173</b>
<b>Handling Change Requests During Adjustment .....</b>	<b>174</b>
<b>Choosing a Change Request for Modifications .....</b>	<b>175</b>
<b>Local and Transportable Change Requests .....</b>	<b>176</b>
<b>Releasing Tasks at the End of Modification Adjustment .....</b>	<b>177</b>
<b>Transferring Modification Adjustments to Other SAP Systems .....</b>	<b>178</b>
<b>Notes (Troubleshooting) .....</b>	<b>179</b>
<b>The SAP Software Change Registration Procedure (SSCR) .....</b>	<b>180</b>
<b>What Is Registered? .....</b>	<b>181</b>
<b>When Are Objects Registered? .....</b>	<b>182</b>
<b>How Do You Get a Key in the Service System? .....</b>	<b>184</b>
Registering a Development User .....	185
Registering an Object .....	186
Displaying an Overview .....	187
<b>How Can Registration Take Place Without SAPNet Access? .....</b>	<b>188</b>
<b>Who Can Use Registration? .....</b>	<b>189</b>
<b>Request for Expansion of Customer Exits .....</b>	<b>190</b>

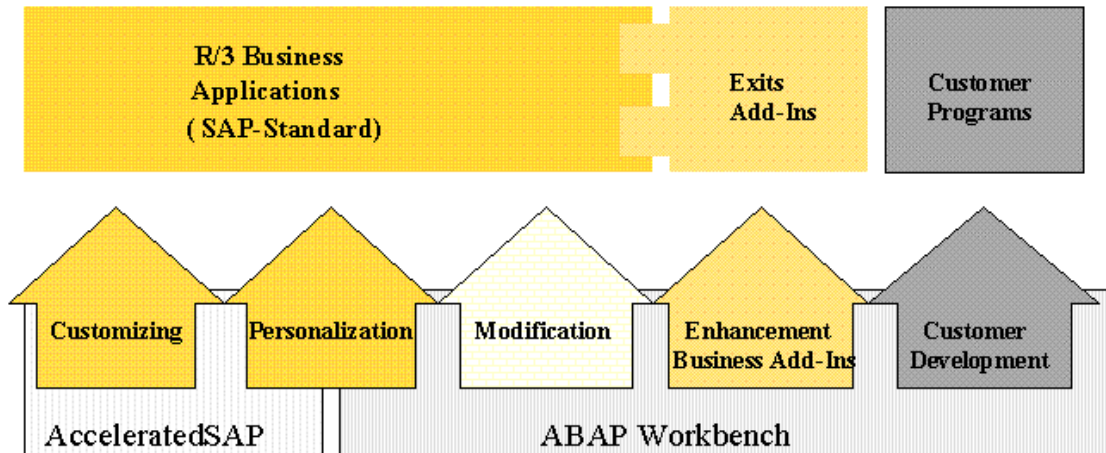
## Changing the SAP Standard (BC)

## Changing the SAP Standard

## Changing the SAP Standard

The SAP System provides a comprehensive infrastructure for business computing. To streamline business processes, however, you may be required to make modifications to the standard. You can enhance, modify or reduce the functions provided for a specific environment.

When making changes to the SAP standard, you must first determine which type of modification best suits your needs. There are several ways of how you can modify the standard system:



ABAP Workbench tools are used to modify and enhance the system, and for customer development as well. Customizing and most Personalization take place in the context of the AcceleratedSAP standard implementation method.

For information about the modification levels see:

[Customizing \[Page 9\]](#)

[Personalization \[Page 10\]](#)

[Modifications \[Page 14\]](#)

[Business Add-Ins/Enhancements \[Page 13\]](#)

[Customer Development \[Page 15\]](#)



## Customizing

Customizing is the setting of system parameters via SAP's own interface. Possible changes to the standards have been thought out and organized by SAP. Customizing is an obligatory part of the R/3 implementation process.

Additional information can be found in the online documentation under [BC - Customizing \[Ext.\]](#)

---

**Personalization**

## Personalization

Personalization means to adjust the SAP System to meet the work requirements of specific users or user groups.

Personalization is aimed at accelerating and simplifying the business transactions that the SAP System processes. Based on the "What you see is what you need" slogan, application Personalization refers to two sub-areas:

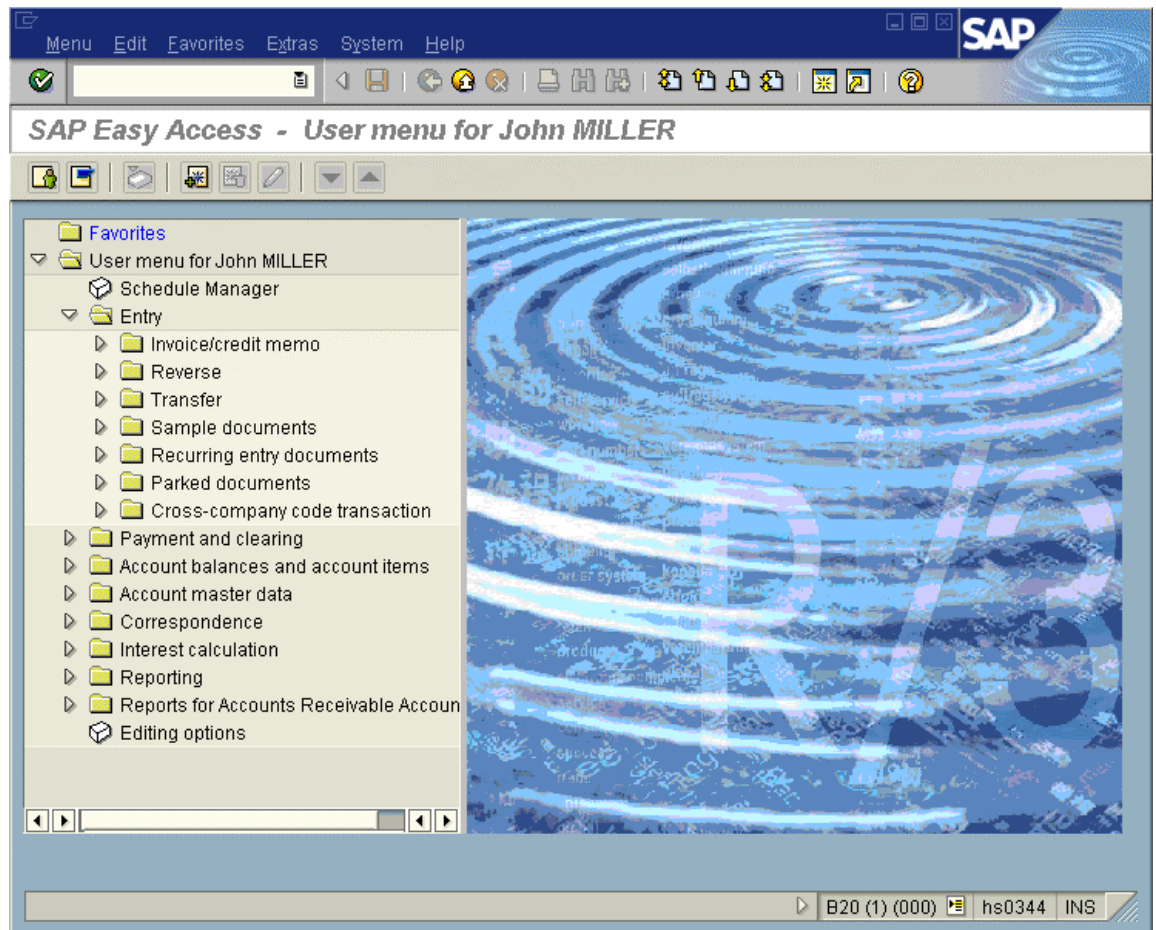
- Simplifying navigation
- Simplifying transactions

## Simplifying Navigation

As of Release 4.6A, the standard point of entry into the system is the [SAP Easy Access \[Ext.\]](#) user menu. Each user of the SAP System can be assigned a user menu tailored to the individual activities of that user that appears when the user logs on to the system. System administrators can choose from more than 1200 pre-defined [standard roles \[Ext.\]](#) and assign these roles to one, several or all users of a company.



Employee John Miller works as an accountant in the accounts receivable department, posting and maintaining a set of customer accounts. The user menu would look like this:



If necessary, the user could add his own reports, Intranet or Internet links, and so on, to the user menu.

Since the user menu contains only those functions that are relevant to the user, navigation is much quicker.

For more information, see [Simplifying Navigation \[Ext.\]](#)

## Simplifying Transactions

You can adjust not only the menus but also the transactions of the SAP System to the business environment of your company. In many cases, the fields and options contained in the standard transactions are not needed for specific process flows. Besides other tools, you can use [transaction and screen variants \[Page 47\]](#) to adjust the transactions of the SAP System. Using transaction and screen variants you can:

- Hide fields and even entire screens
- Preassign values to fields
- Change the ready for input status of fields
- Change the properties of table control columns and hide specific columns
- Hide menu options

---

## Personalization

For a description of the other tools for personalizing transactions in the SAP System, see [Personalization of an Application \[Page 17\]](#).

## Enhancements to the Standard

Enhancements represent potential customer requirements that have not been developed in the standard. Instead, the standard provides for further development of such exits at the customer site, using logic specific to the customer. Upward compatibility is assured since SAP guarantees that the call of an enhancement from the standard software and the calling interface will remain valid in future releases.

You can make enhancements to the standard using [customer exits \[Page 99\]](#).

From Release 4.6A, you can implement enhancements for the system standard using a new technique called [Business Add-Ins \[Page 77\]](#).

---

Modifying the Standard

## Modifying the Standard

Before you modify your system or develop your own solution, make sure that your requirements cannot be fulfilled by either Customizing or Personalization, or by the exits or business add-ins provided in the system.

Before making modifications to the standard, refer to the section on the [Modification Assistant \[Page 116\]](#).



For information on modifying SAP tables, consult the portion of the *ABAP Dictionary* documentation called [Making Changes to Tables \[Ext.\]](#). You can also create append structures for tables and structures with the Modification Assistant (see [Modifying Tables and Structures \[Page 131\]](#)).

SSCR (**S**AP **S**oftware **C**hange **R**egistration) is a procedure for registering all modifications to SAP source code and SAP Dictionary objects. For more information on this subject, see the section on the [SAP Software Change Registration Procedure \(SSCR\) \[Page 180\]](#).

You should also consult the sections [General development organization \[Ext.\]](#) and [Notes on modifying the standard \[Ext.\]](#) before modifying the standard.

## Customer Development

During customer development, customer-specific objects are created in the customer namespace.



For further information about creating programs, data modeling, and related topics, consult the ABAP Workbench section of the SAP online documentation.

SAP has reserved namespaces for customer objects. Using these namespaces ensures that your objects are not be overwritten by SAP objects when new objects are imported into your system or during a release upgrade. For more information on naming conventions, see [Customer Namespaces \[Ext.\]](#).

Pay attention to the following when developing your own programs and modules:

### Dividing Up Development Tasks

How might authorizations be distributed amongst your developers?

Four areas of responsibility come into play here:

- Maintaining ABAP Dictionary elements such as tables, data elements, and domains
- Maintaining database tables
- Maintaining objects such as module pools, function modules, screens, and so on
- Writing documentation

Authorizations relate both to the object type (program, table, development class, and so on) and to the activity (change, display, create, activate, and so on).

You need to find a way to allow your developers some freedom while ensuring system consistency.

You might assign authorizations as follows:

- Give developers authorization to edit all programs within an application (this authorization is assigned using the corresponding development classes) and to display all ABAP Dictionary objects. Also allow them to create and activate structures and views (data in the database cannot be changed by structures or views).
- Give the person(s) responsible for the ABAP Dictionary authorization to create and activate all Dictionary objects. Give the person(s) responsible for the database authorization to create tables in the database.
- Documentation developers should have appropriate authorization for writing user documentation for the objects (developers may, however, choose to do this themselves).

### Languages

All SAP objects have an original language; this is the language in which the object was created. To simplify the customer development process, you should agree on a development language in which all objects are to be created.

---

**Customer Development**

You must enter and maintain texts in the original language for each object in your development. You may also translate these into other languages by choosing *Tools → ABAP Workbench → Utilities → Translation → Short/long texts*.

**Ergonomic Guidelines**

Your own applications should look and feel like SAP standard applications and have similar screen layout and function key assignments.

For information on how to design your applications, see the documentation *SAP Style Guide (BC)*.



## Personalizing the Application

The following sections describe how you can simplify the SAP user interface to meet the work requirements of specific users or user groups. Application [Personalization \[Page 10\]](#) can often be undertaken using simple methods (outside the ABAP Workbench). Modifications to the standard are usually not necessary. Modifications are only mentioned if they do not create a lot of extra work at upgrade, or if they are unavoidable.

In the following, this documentation provides various example tasks from the working world. Each example introduces the tools used to adjust the SAP user interface to the desired task and discusses the restrictions you need to pay attention to when using these utilities. The documentation then goes on to discuss the rules that apply when you combine certain functions.



When choosing tools, make sure that you use the same ones consistently in similar situations (when installing the R/3 System or within the same R/3 component) for tailoring the interface of the SAP System.

The following examples are designed to give you an overview of the various Personalization possibilities available. Each section contains helpful references to additional literature within the SAP System.

[Hiding Screen Elements \[Page 18\]](#)

[Defining Default Values for Input Fields \[Page 19\]](#)

[Revoking an Element's Ready for Input Status \[Page 20\]](#)

[Adjusting Table Controls \[Page 21\]](#)

[Simplifying Selection Screens \[Page 25\]](#)

[Personalizing the Possible Entries Help \[Page 26\]](#)

[Moving Fields on a Screen \[Page 29\]](#)

[Changing Field Texts \[Page 30\]](#)

[Including Graphics and Text \[Page 31\]](#)

[Inserting Pushbuttons for Frequently-used Functions \[Page 32\]](#)

[Displaying Possible Entries for Fields \[Page 33\]](#)

[Defining Additional Elements on a Screen \[Page 34\]](#)

[Simplifying Screens for Specific Users \[Page 35\]](#)

## Hiding Screen Elements

## Hiding Screen Elements

Functions:	Description:	Restrictions:
<a href="#">Transaction Variants and Screen Variants [Page 37]</a>	You can hide screens, menu options, table controls, subscreens, tab strip elements, and so on	May only be used with dialog transactions
<a href="#">Selection Variants [Page 46]</a>	Hiding selection fields	May only be used with selection screens
<a href="#">GuiXT [Page 39]</a>	Hiding fields, pushbuttons, frames, and so on	Cannot be used to hide subscreens
<a href="#">Parameter Transactions [Page 45]</a>	The initial screen of a parameter transaction can be hidden (switched off) if all necessary entries for that screen have already been stored as parameters.	May only be used with dialog transactions
<a href="#">Application-Specific Field Selection [Page 43]</a>	Field attributes <i>Input</i> , <i>Display</i> , <i>Required field</i> , <i>Highlighted</i> and <i>Invisible</i> can be enabled or disabled at runtime.	<ul style="list-style-type: none"> <li>No assignment according to user possible</li> <li>Field selection must be implemented by an application developer</li> </ul>

## Combinations of the Methods Listed Above

The following rule applies when combining the functions listed above:

Screen elements that have been hidden by a function remain hidden. This is also true if the element is set to *Display only* by another function.

## Defining Default Values for Input Fields

Functions:	Restrictions:
<a href="#">SET/GET Parameters</a> <a href="#">[Page 44]</a>	Application developers must use the Screen Painter to set the attributes <i>Set Parameter</i> and <i>Get Parameter</i> for all input fields for which you want to define default values.
<a href="#">Transaction Variants and Screen Variants</a> <a href="#">[Page 37]</a>	May only be used with dialog transactions
<a href="#">Selection Variants</a> <a href="#">[Page 46]</a>	May only be used with selection screens
<a href="#">Parameter Transactions</a> <a href="#">[Page 45]</a>	The default values selected are only valid on the initial screen of the dialog transaction
<a href="#">GuiXT</a> <a href="#">[Page 39]</a>	None

### Combinations of the Methods Listed Above

The following rules apply when combining the functions listed above:

- Transaction variants and SET/GET parameters:

Default values from transaction or screen variants are given priority if the variant hides a field or revokes its ready for input status. These values also have priority if SET parameter is used to set an initial default value (SPACE, for example). As far as fields are concerned that are ready for input, however, default values of SET parameters have priority since these may be based on user entries, for example.

- Selection variants and SET/GET parameters:

Default values from selection variants always have priority as long as they are not initial.



With initial values you can select *Switch off SET parameter/GET parameter* during variant maintenance. This inserts the initial value from the variant, otherwise SET parameter values retain priority.

- GuiXT combined with other functions:

Default values are only set using GuiXT if the field in question has an initial value. This is true for regardless of whether the value has been set using a transaction, SET/GET parameters, or a variant.

## Revoking an Element's Ready for Input Status

## Revoking an Element's Ready for Input Status

Functions:	Restrictions:
<a href="#">Transaction Variants and Screen Variants [Page 37]</a>	May only be used with dialog transactions
<a href="#">Selection Variants [Page 46]</a>	May only be used with selection screens
<a href="#">Application-Specific Field Selection [Page 43]</a>	<ul style="list-style-type: none"><li>• No assignment according to user possible</li><li>• Field selection must be implemented by an application developer</li></ul>

## Adjusting Table Controls

You can adjust the layout of a table control by:

- Changing the column sequence
- Changing the column width
- Hiding columns
- Revoking the ready for input status of individual columns, or
- Preassign values to table control fields

You can adjust the layout of a table control at two levels:

- User level, and
- Client level

### User Level

The *Table controls* function in the upper right corner of the table control allows you to personalize the basic table settings delivered with the system. For more information, see [Table Controls \[Ext.\]](#) in the *Getting Started* documentation.

Table control adjustments made at user level take priority over adjustments made at client level. Note the following exceptions:

- Hidden columns remain invisible
- Columns not ready for input remain locked
- Screen variant default values are adopted

The first two points are important, otherwise columns could be seen or changed by user groups that they should not be available to.

### Client Level

There are two ways of how you can adjust the layout of a table control on a client-wide basis:

- Using the administrator function for table control settings, or
- Using a transaction or screen variant

You use the administrator functions for table control settings if you want to make the following changes in one client only:

- Change the column sequence
- Change the column width
- Hide columns

If you want to revoke the ready for input status of individual columns or preassign values to table control fields, you must create a screen variant since the administrator function for table controls does not support this. You should also use a screen variant for adjusting table controls if you want to copy the table control settings to other clients. Since screen variants can be transported, their corresponding table control layouts can be transported as well.

## Adjusting Table Controls

### Administrator Function for Table Control Settings

Users with authorization 'TCTR' in authorization object 'S\_ADMI\_FCD' can define a layout variant of a table control as a standard setting. This setting (variant) then becomes the table control's global setting for all users whenever they start the transaction. If you do not define a systemwide variant, the system interprets the screen definition as the global setting.

In order to define a table control for the entire system, proceed as follows:

1. Start the transaction in question and navigate to the appropriate table control.
2. Use the *Table settings* symbol to activate the *Global settings* variant (if this is not already the case).
3. Adjust the table control to your specifications (column width and column position can be adjusted; all columns you do not need can be hidden by setting their width to zero).



One column of the table control must remain visible.

4. Choose *Administrator* from the dialog box that appears when you activate the *Table settings* icon.

## Adjusting Table Controls

Airline carrier  Lufthansa  
 Flight number

Depart.city   
 Arrival city   
 Flight time   
 Departure

Date	Plan	C:
29.11.1999	DC-10-10	3E
02.12.1999	DC-10-10	3E
09.12.1999	DC-10-10	3E
29.12.1999	DC-10-10	3E
31.12.1999	DC-10-10	3E

Edit System Settings

Current settings

Colu	Fld	Pos.	Leng	Invi
Date	SFLIGHT-FLDATE	1	10	<input type="checkbox"/>
FlgtPrice	SFLIGHT-PRICE	2	0	<input checked="" type="checkbox"/>
Curr.	SFLIGHT-CURRENCY	3	0	<input checked="" type="checkbox"/>
Plane type	SFLIGHT-PLANETYPE	4	10	<input type="checkbox"/>
Capacity	SFLIGHT-SEATSMAX	5	10	<input type="checkbox"/>
Occupied	SFLIGHT-SEATSOCC	6	10	<input type="checkbox"/>

Program

Control

☐ Available sys. variant

Further settings

☒ Horizontal separator  
☒ Vertical separator  
 No. of fixed columns

The table control attributes are displayed in the table *Current settings*. No value appears in the field *Length* for those columns whose width has been set to zero; you can deactivate them by selecting the setting *Invisible*.

You can also change other attributes of the current table control configuration. You can:

- Hide additional columns using the *Invisible* checkbox
  - Configure the number of fixed columns
  - Hide or display the gridlines in the table control
5. Choose *Activate* to transfer your settings to the rest of the system. The settings activated are the basic settings for each user.



If you close the dialog box without choosing *Activate*, the settings remain unchanged.

6. Call the transaction a second time and check your new settings.

### Adjusting Table Controls

Repeat the steps listed above if you want to alter the table control settings yet again. Each time you activate new settings, this variant overwrites the existing one.

### Adjusting a Table Control Using a Screen Variant

For information on adjusting table controls using screen variants, see [Table Control Layout in Variants \[Page 68\]](#).



If administrator settings are available for a table control and a layout is set using a screen variant, then the layout of the screen variant takes effect.



## Simplifying Selection Screens

Function:	Description:
<a href="#">Selection Variants [Page 46]</a>	<p>You can:</p> <ul style="list-style-type: none"> <li>• Preassign selection values for selection screens In addition to static defined values (such as '0001' for company code), variable selections are also allowed (for example, a variable billing date for date fields).</li> <li>• Revoke the ready for input status of selection criteria</li> <li>• Hide screen elements</li> <li>• Hide the entire selection screen</li> </ul>

## Personalizing the Possible Entries Help

## Personalizing the Possible Entries Help

### Use

The possible entries help is a standard function available in the SAP System. Using the possible entries help, users can display a list of all possible values for a screen field.



In the SAP System, the possible entries help process flow is based on a [search help \[Ext.\]](#) from the ABAP Dictionary. Search helps are assigned to the screen fields for which they are to be available.

For example, the possible entries help for the *Sold-to party* field in transaction *Create Sales Order* (VA01) would be displayed as follows:

SearchTerm	Zip code	City	Name 1
HALLMANN	60270	GOUVIEUX	HALLMANN S.A.
HARLEY	68766	HOCKENHEIM	HARLEY SALES HOCK
HARLEY	69121	HEIDELBERG	BIKER GMBH
HAVERS	59425	UNNA	HAVERS BAUMARKT
HEATHROW	DE2 8UP	LONDON	HEATHROW AIRPORT
HEIMBAU	45470	MUEHLHEIM AN DER RUHR	HEIMBAU
HENDERSON	50944	ATLANTA	HENDERSON EQUIPM
HENDERSON	60804	CHICAGO	HENDERSON INC.
HERSHAY	26436	HAMBURG	HERSHAY FOODS - HA
HERSHAY	32821	ORLANDO	HERSHAY FOODS - OF
HERSHAY	33131	MIAMI BEACH	HERSHAY FOODS COI
HERSHAY	60000	FRANKFURT	HERSHAY FOODS - FR
HEVER	HP12 3TL	HIGH WYCOMBE	HEVER INDUSTRIAL U
HIER	GA30328	ATLANTA	SMITH SOUTH
HIER	M2P 2B8	NORTH-YORK	SMITH NORTH
HIER	PA19113	LESTER	SMITH CENTRAL BUYI

You can adjust a list and save settings that:

- Modify the behavior of all possible entries helps
- Modify the behavior of a single possible entries help

### Procedure

To adjust a possible entries help, proceed as follows:

1. Call the possible entries help for any field.

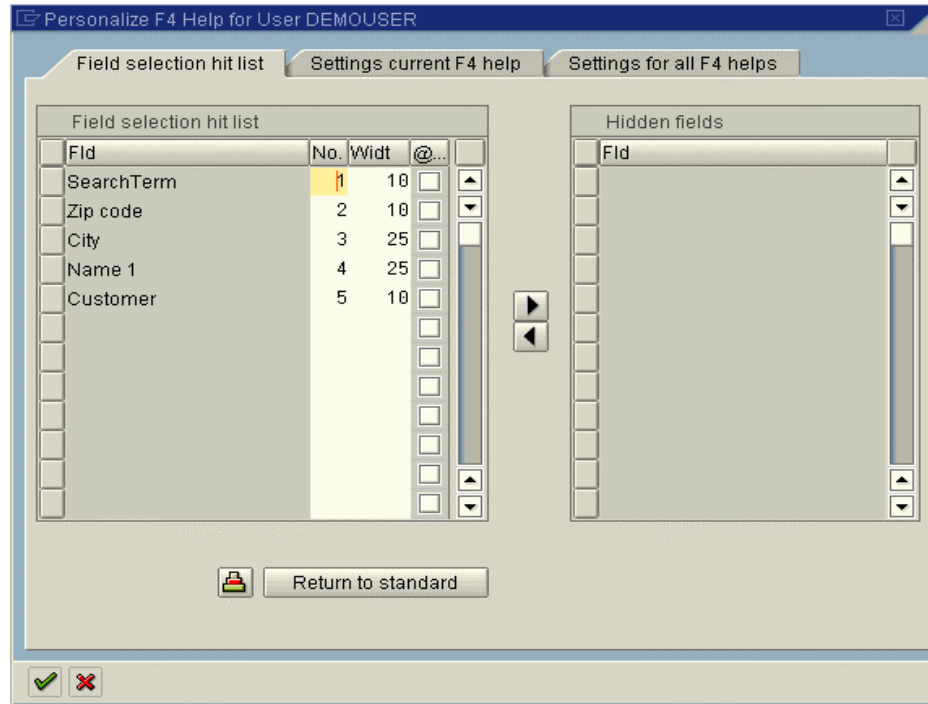
## Personalizing the Possible Entries Help

- Click the right mouse-button or choose the Ctrl-F4 key combination. Choose *Personalize* from the context menu.



The *Personalize* option of the context menu is only available on the hit list.

The following dialog box appears:



- Make your settings for the current possible entries help or for all possible entries helps. Note the following information.

#### Settings you can make globally for all possible entries helps or for the current possible entries help:

- Display of personal value list:* You can specify if your personal value list should be displayed automatically if one exists.
- System behavior if only one hit:* If the system finds one hit only for the possible entries help, you can either display this hit on the hit list or have it returned directly to the screen.
- Maximum width of hit list:* You can determine the maximum width (number of characters) of the hit list in order to adjust the possible entries help to small monitors.
- Maximum number of hits to be displayed:* You can determine the maximum number of lines that the hit list should contain when called for the first time.

#### Settings you can make additionally for a single possible entries help only:

- Hide and swap columns:* You can hide columns in the hit list. The list of hidden columns is saved. These columns are not displayed any more when you call up the possible entries help the next time. This allows you to hide columns that you do not need and make the hit list easier to read. You can also swap columns on the hit list.

---

**Personalizing the Possible Entries Help**

- *Skip selection screen:* You can specify if the hit list is to be displayed immediately or if a selection screen should be displayed where you can further restrict the values.

**Setting you can make globally for all possible entries helps only:**


- *Display:* You can determine whether the possible entries help is to be displayed in a search help control, in an R/3 dialog box or in accordance with the system default.

For more information on how your settings affect the behavior of the possible entries help, see the F1 help for the individual parameters.

## Moving Fields on a Screen

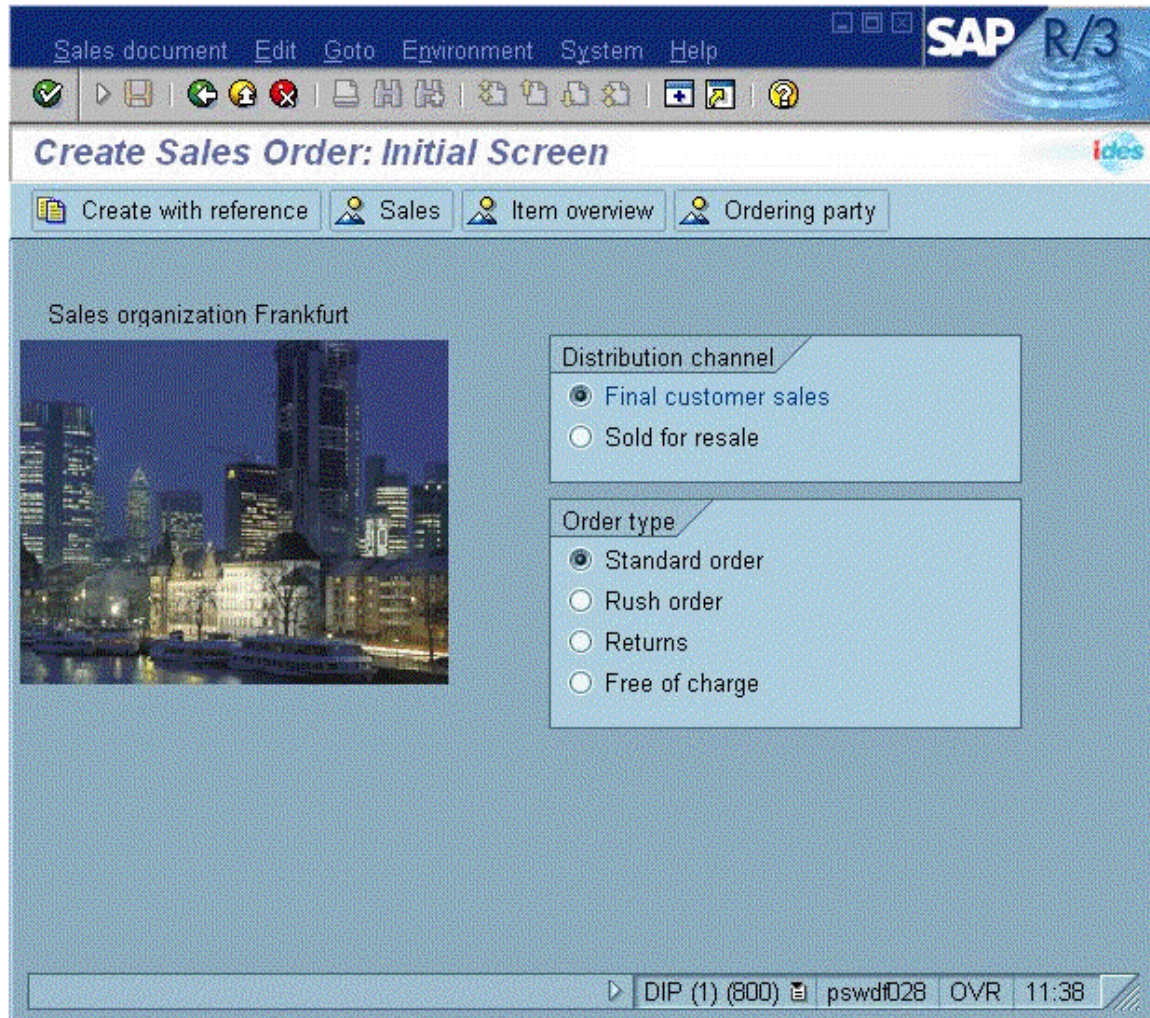
[GuiXT \[Page 39\]](#) allows you to move individual fields and groups of fields around on your screen. You can even move fields from a subscreen to the actual screen or vice versa without affecting program logic (modifying the screen in the ABAP Workbench would alter the program logic). By contrast, GuiXT only alters a screen's layout.

## Changing Field Texts

Function:	Description:
<a href="#">Enhancements (SMOD/CMOD) [Page 40]</a>	<p>You can use the enhancement concept to create your own keywords for data elements from the ABAP Dictionary. Keywords are used to identify fields on screens. Data elements can have up to three keywords of varying lengths.</p> <p>To change a screen's field texts, choose <i>Utilities</i> → <i>Enhancements</i> → <i>Project management</i> (transaction CMOD) from the initial screen of the <i>ABAP Workbench</i>. The <i>Text enhancements</i> menu contains all of the enhancement functions for keywords and data element documentation.</p>  <p>Text enhancements do not have to be assigned to an enhancement project.</p>
<a href="#">GuiXT [Page 39]</a>	You can change field texts in a GuiXT script using the appropriate commands.
<a href="#">Modification Assistant [Page 41]</a>	You can also change field texts using the Modification Assistant. For further information, please refer to the section on <a href="#">Modifying Data Element Attributes [Page 132]</a> . All such modifications must be registered in SSCR.

## Including Graphics and Text

[GuiXT \[Page 39\]](#) allows you to add graphics and text to a screen.



Pay close attention to the guidelines found in the section [GuiXT in Transaction Variants and Screen Variants \[Page 76\]](#).



## Inserting Pushbuttons for Frequently-used Functions

## Inserting Pushbuttons for Frequently-used Functions

Function:	Description:
<a href="#">GuiXT [Page 39]</a>	With GuiXT you can create pushbuttons for frequently-used menu functions and position them on your screen or in the application toolbar.
<a href="#">Modification Assistant [Page 41]</a>	<p>The Modification Assistant allows you to create and delete pushbuttons for frequently-used menu functions on your screen or in the application toolbar.</p> <p>Even though these changes are technically modifications, they can be adopted automatically at upgrade. This is the case unless the SAP standard contains a new function code with the same name in the new release (which is highly unlikely). Further information can be found in the section <a href="#">Modifications in the Menu Painter [Page 123]</a>.</p>




## Changing Possible Entries into Radio Buttons

[GuiXT \[Page 39\]](#) allows you to convert an input field into a group of radio buttons.

You can transform each possible entry into a radio button and thus avoid additional dialog involving the possible entries help (F4).

For example, if an employee wants to create a sales order in the application SD, substituting radio buttons for the possible entries pushbutton next to the *Order type* field might look like this:


Order type CS  Cash Sale

Organizational data

Sales organization	1000	Germany Frankfurt
Distribution channel	10	Final customer sales
Division	00	Cross-division
Sales office		
Sales group		

Conversion using GuiXT:

Sales organization Frankfurt



Distribution channel

☒ Final customer sales

☐ Sold for resale

Order type

☒ Standard order

☐ Rush order

☐ Returns

☐ Free of charge

---

Defining Additional Elements on a Screen

## Defining Additional Elements on a Screen

Enhancements (SMOD/CMOD) make it possible for you to add your own elements to application logic, screens, and menus.

Menu enhancements and screen enhancements allow you to define additional elements for a screen.

Menu enhancements are predefined menu options that you can activate and name. The function code transmitted when you select the menu option calls a corresponding function module exit.

Screen enhancements are areas on a screen that customers can fill with their own developments. They must be predefined by an application developer. You can display additional information or allow users to enter additional data in these areas.

You can find an overview of the types of standard enhancements contained in the SAP System in the ABAP Workbench under *Utilities* → *Enhancements* → *Project management* (transaction CMOD). Use the possible entries help (F4) for the field *Project* or choose *Utilities* → *SAP enhancements* to display a list of the exits available.

The corresponding documentation describes the functionality of each enhancement.



Upward compatibility of customer exits is assured. SAP guarantees that the call of the exit from SAP standard software, as well as the interface which calls the exit, will both remain valid in future releases.

## Simplifying Screens for Specific Users

Functions:	How Simplification is Implemented:
<a href="#">SET/GET Parameters [Page 44]</a>	Desired default value can either be entered in a user's master record or individually determined for a particular input field
<a href="#">Transaction Variants and Screen Variants [Page 37]</a>	Alternate transaction is called by the user
<a href="#">GuiXT [Page 39]</a>	GuiXT script is stored locally on user's PC

---

**Tools for Personalizing the Application**

## **Tools for Personalizing the Application**

You can use the following functions when making changes to user interfaces in the SAP System:

[Transaction Variants \[Page 37\]](#)

[Variant Transactions \[Page 38\]](#)

[GuiXT \[Page 39\]](#)

[Enhancements \(SMOD/CMOD\) \[Page 40\]](#)

[Modification Assistant \[Page 41\]](#)

[Application-Specific User Exits \[Page 42\]](#)

[Application-Specific Field Selection \[Page 43\]](#)

[SET/GET Parameters \[Page 44\]](#)

[Parameter Transactions \[Page 45\]](#)

[Selection Variants \[Page 46\]](#)

## Transaction Variants and Screen Variants



<b>Type (Behavior at Upgrade, Transport)</b>	Customizing
<b>Description of Function</b>	<p>Transaction variants simplify transaction flow by:</p> <ul style="list-style-type: none"> <li>• Inserting default values in fields</li> <li>• Changing the ready for input status of fields</li> <li>• Hiding various screen elements and menu functions, or even entire screens</li> <li>• Adjusting table control settings</li> </ul> <p>When you create a transaction variant, you run through the transaction in a special mode. In this mode, you decide which elements you no longer need in your transaction and which values should be inserted where. If you choose to hide an entire screen, the system will no longer display that screen in the transaction and will proceed on to the next screen in the series. You can also adjust table control settings and determine the width of columns and the order in which you want them displayed. In tab strips, you can suppress tabs that you no longer need by setting them to invisible.</p> <p>You may also create as many variants for a transaction as you desire. One of these variants can be defined as a standard variant. This variant is then valid across all clients. You may also assign different variants to specific users.</p> <p>This is done using <a href="#">Variant Transactions [Page 38]</a>. Variant transactions are then placed in those users' user menus instead of the standard transaction.</p>
<b>Restrictions</b>	May not be used with selection screens
<b>Range (Validity)</b>	Transactions, both client-dependent and cross-client
<b>Access in the System</b>	<i>Tools → Accelerated SAP → Personalization</i> (transaction SHD0)
<b>Further Information</b>	More information on creating transaction variants can be found under <a href="#">Transaction Variants and Screen Variants [Page 47]</a>

## Variant Transactions

## Variant Transactions

Type (Behavior at Upgrade, Transport)	Modification
Description of Function	In order to assign transaction variants to specific users, you must first define a variant transaction. When defining a variant transaction you must enter the name of the transaction and the name of the variant. The new transaction code then allows you to call this specific variant of your transaction.
Restrictions	May only be used with dialog transactions
Range (Validity)	Throughout the entire system
Access in the System	<p>Transaction SHD0, from there you can navigate to the ABAP Workbench using <i>Goto</i> → <i>Create vari. trans.</i></p> <p>Or you can call the transaction from the ABAP Workbench using <i>Development</i> → <i>Other tools</i> → <i>Transactions</i>. Enter a new transaction code in the appropriate field and choose <i>Variant transaction</i> in the subsequent dialog box.</p>
Further Information	More information on creating variant transactions can be found under <a href="#">Starting Transactions Variants [Page 60]</a>

## GuiXT

Type (Behavior at Upgrade, Transport)	Add-on (no transport possible)
<b>Description of Function</b>	<ul style="list-style-type: none"> <li>You can insert default values in fields</li> <li>You can hide fields and groups of fields</li> <li>You can move fields</li> <li>You can add and change texts</li> <li>You can add input help</li> <li>You can add new screen elements like checkboxes, pushbuttons, graphics, or documentation</li> </ul>  <p>GuiXT scripts can also be integrated into transaction variants. Further information can be found under <a href="#">Transaction Variants and Screen Variants [Page 47]</a></p>
<b>Restrictions</b>	May not be used with selection screens or lists Windows 32 bit is a requirement
<b>Range (Validity)</b>	Throughout the entire system or user-specific on your PC
<b>Further Information</b>	You can find more information about GuiXT by adding the alias <i>/GuiXT</i> to your SAPNet address.
<b>Availability</b>	<p>Available as an add-on in releases 3.0, 3.1, 4.0, and 4.5; starting with release 4.6A integrated into R/3.</p>  <p>Those customers with release 3.0/3.1 interested in purchasing GuiXT should contact Synactive GmbH in Mannheim; customers with release 4.0 or greater can contact SAP directly.</p>

## Enhancements

## Enhancements

<b>Type (Behavior at Upgrade, Transport)</b>	Customizing
<b>Description of Function</b>	Enhancements allow you to add additional functions to the SAP standard.
<b>Restrictions</b>	<ul style="list-style-type: none"><li>• The interfaces must have been implemented.</li><li>• Changes to keywords in SMOD/CMOD affect all of a data element's screen fields.</li></ul>
<b>Range (Validity)</b>	Throughout the entire system
<b>Access in the System</b>	Choose <i>ABAP Workbench</i> → <i>Utilities</i> → <i>Business Add-Ins (SE18)</i> or <i>Utilities</i> → <i>Enhancements</i> → <i>Definition</i> or <i>Project Management (SMOD/CMOD)</i> .
<b>Further Information</b>	For more information on enhancements, see <a href="#">Business Add-Ins [Page 77]</a> and <a href="#">Customer Exits [Page 99]</a> .



## Modification Assistant

Type (Behavior at Upgrade, Transport)	Modification
Description of Function	<p>In order to use the Modification Assistant to simplify the upgrade process, you branch to a special modification mode whenever you are modifying objects from the standard in an ABAP Workbench editor. Originals are initially protected in this mode and can only be changed with the help of the additional pushbuttons that are placed at your disposal.</p> <p>All changes that you make to the system are logged with the help of the Modification Assistant. This provides you with a detailed overview of modifications that is easy to read and that dramatically reduces the amount of effort needed to upgrade your system.</p> <p>The Modification Assistant offers support in the following areas:</p> <p>ABAP Editor, Screen Painter, Menu Painter, text element maintenance, Function Builder, ABAP Dictionary (table and structure enhancements using append structures, data element attribute modification) and documentation (modification of documents from specific classes).</p>
Range (Validity)	Throughout the entire system
Access in the System	ABAP Workbench
Further Information	For further information, refer to the section on the <a href="#">Modification Assistant [Page 116]</a> .

## Application-specific User Exits

## Application-specific User Exits

Type (Behavior at Upgrade, Transport)	Modification
Description of Function	<p>User exits allow you to add additional functions to the SAP standard.</p> <p>Programs with user exits contain subroutine calls at certain points in their syntax that are identified by the prefix USEREXIT. The actual user exits are located in an include that has been assigned to a module pool. This is where customers can include any changes (enhancements) that they want to make to the system. These includes are always processed during program flow.</p> <p><b>Advantage:</b> In principle, customers can modify anything they want that is found in the include (tables, structures, and so forth).</p> <p><b>Disadvantage:</b> SAP cannot check the individual enhancements themselves which often leads to errors in the enhancement process.</p>
Range (Validity)	Throughout the entire system
Access in the System	SAP Reference IMG
Further Information	User exits are primarily used in the sales and distribution component. You can find an overview of user exits as well as a description of those user exits that exist in SD in the SAP Reference IMG under <i>Sales and Distribution</i> → <i>System Modification</i> → <i>User exits</i> .

## Application-specific Field Selection

<b>Type (Behavior at Upgrade, Transport)</b>	Customizing
<b>Description of Function</b>	<p>The <i>Field selection</i> function allows you to change screen field attributes dynamically at runtime. You should use this special function whenever you need different field attributes for the same screen at different points within an application.</p> <p>Field selection allows you to alter the attributes <i>Input</i>, <i>Display</i>, <i>Required field</i>, <i>Highlighted</i>, and <i>Invisible</i> at runtime.</p> <p>You can also have your field attribute assignments change dynamically according the definition of other, influencing fields.</p>
<b>Restrictions</b>	Field selection must be implemented by an application developer
<b>Range (Validity)</b>	The client
<b>Access in the System</b>	Can be found in the SAP Reference IMG in various components. (Example from the IMG: <i>Project System</i> → <i>Confirmation</i> → <i>Define Field Selection for Confirmation</i> )

## SET/GET Parameters

## SET/GET Parameters

Type (Behavior at Upgrade, Transport)	Customizing
Description of Function	<p>You can fill fields on screens with default values from SAP memory using parameter IDs.</p> <p>For example, a user only has authorization for company code 0001. By entering the value '0001' in field <i>COCD</i> in the <i>Parameter</i> register in this user's master record (SU01), the system automatically fills the field <i>Company code</i> with the value '001' on all screens he or she calls. If this company code is not predetermined using a parameter ID in the user master record, the system automatically adopts the first value entered by the user at the beginning of the transaction for the rest of the current terminal session. However, this value has to be re-entered the next time the user logs on to the system.</p>
Restrictions	Fields on screens are only ever automatically filled with the value saved under the parameter ID of a data element if the <i>Set Parameter/Get Parameter</i> attributes for the corresponding fields have been explicitly set in the Screen Painter.
Range (Validity)	Per end user
Access in the System	Choose <i>Tools</i> → <i>Administration</i> , then <i>User maintenance</i> → <i>Users</i> . Enter the IDs you want in the <i>Parameter</i> register during user maintenance.
Further Information	For more information about setting up parameter IDs, refer to <a href="#">Maintaining User Profiles and Options [Ext.]</a>

## Parameter Transactions

Type (Behavior at Upgrade, Transport)	Modification
Description of Function	<p>In parameter transactions, you can preallocate values for the screen fields on the initial screens of specific transactions.</p> <p>The initial screen of a parameter transaction can be hidden (switched off) if all necessary entries for that screen have already been stored as parameters.</p>
Restrictions	<p>May only be used with dialog transactions</p> <p>The default values selected are only valid on the initial screen of the transaction</p>
Range (Validity)	Throughout the entire system
Access in the System	<p>Can be found in the ABAP Workbench under <i>Development</i> → <i>Other tools</i> → <i>Transactions</i>. Enter a new transaction code in the appropriate field and choose <i>Transaction with parameters (Parameter transaction)</i> in the subsequent dialog box. For more information, see <a href="#">Creating a parameter transaction [Ext.]</a></p>

## Selection Variants

## Selection Variants

<b>Type (Behavior at Upgrade, Transport)</b>	Customizing
<b>Description of Function</b>	<p>Whenever you start a program for which selection screens are defined, the system prompts you with input fields for database-specific and report-specific selections. If you only want to work with a specific dataset, its corresponding values must be entered here.</p> <p>If you often use the same program with identical selections, for example, a program for generating monthly sales statistics, this process of repeatedly entering the same selection criteria can become quite tedious. In this case, it is helpful to bundle your selection criteria together into a selection variant.</p> <p>Selection variants allow you to set default values for selection criteria. The input status of fields can be changed and fields can also be hidden.</p>
<b>Restrictions</b>	May only be used with selection screens
<b>Range (Validity)</b>	Selection variants can be client-dependent or cross-client
<b>Access in the System</b>	Accessed from a report's selection screen by choosing <i>System</i> → <i>Services</i> → <i>Reporting</i> (SA38)
<b>Further Information</b>	Further information on creating selection variants can be found in the section <a href="#">Maintaining Variants [Ext.]</a> in the online documentation under <i>BC - Basis Components</i> → <i>ABAP Workbench: Tools</i> .

## Transaction Variants and Screen Variants

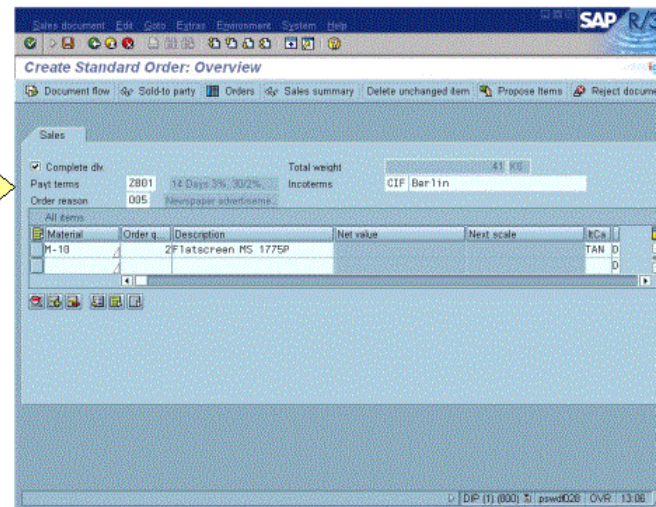
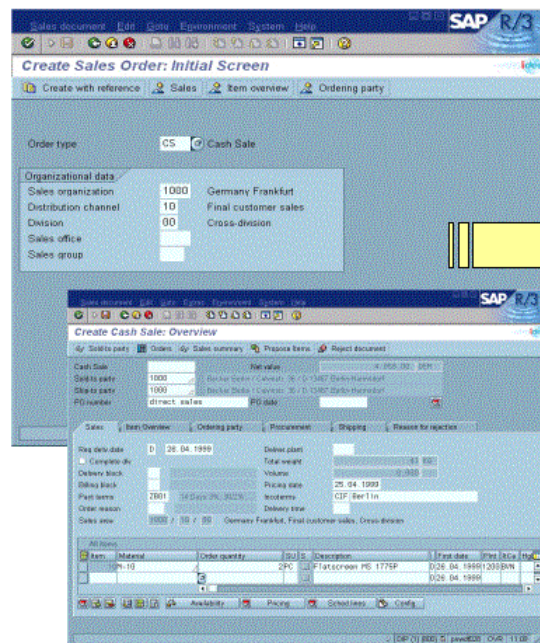
Whenever you use a transaction in the SAP system to process specific business transactions, it often makes sense to adjust processing flow to mirror these business activities. This can be done by hiding all information not pertinent to the business transaction and by displaying especially important information more prominently.



The transaction *Create sales order* (VA01) allows you to create export orders and returns as well as normal sales orders. You are provided with a specific view of this transaction's flow according to the task at hand. Sometimes only a handful of dialogs steps are actually relevant.

For example, you want to create a direct customer sale (used with those customers that come to your production site, pay directly, and take their merchandise with them). All fields on the initial screen can have default values inserted into them according to transaction. Since no additional input is necessary here, this screen can now be hidden. Your second screen also contains a lot of data that can be set to a certain default value and hidden, for example customer data, sales data, and purchasing data.

In this case you will want to hide the initial screen. The second screen can be simplified by hiding the fields, tabs, pushbuttons, and menu functions contained in the standard transaction.



Creating a transaction variant only alters the layout of the screen. Business processes delivered by SAP retain their integrity in any case.

Transaction variants are actually made up of a series of screen variants. The field values and field attributes for each screen in a transaction variant are stored in [screen variants \[Page 62\]](#).

## Transaction Variants and Screen Variants

You can create as many variants of a specific transaction as you like. These variants are started by entering a transaction code that you have selected into the R/3 command field. Your system administrator can assign your transaction codes to user menus. This allows the users in the departments that use your transaction to call the transaction directly. For more information, see [Creating Roles \[Ext.\]](#).

Transaction variant functionality is complemented by the use of [GuiXT \[Page 76\]](#). This new tool allows transactions to be enhanced using graphics, texts, and HTML pages. Screen fields can also be moved around on the screen and important new menu functions can be added to the user interface or application toolbar in the form of pushbuttons.

For more information, see:

[Transaction Variants \[Page 49\]](#)

[Variant Transactions, Setting Transaction Variants \[Page 60\]](#)

[Screen Variants \[Page 62\]](#)

[Table Control Layout in Variants \[Page 68\]](#)

[Transaction Variants and Screen Variants: Restrictions \[Page 70\]](#)

[Transaction Variants and Screen Variants: Special Features \[Page 72\]](#)

[GuiXT \[Page 76\]](#)



## Transaction Variants

Transaction variants simplify transaction flow by:

- You can insert default values for fields
- Hiding and changing the ready for input status of fields
- Hiding and changing the attributes of table control columns
- Hiding individual menu functions
- Hiding entire screens

Transaction variants are actually made up of a series of screen variants. The field values and field attributes for each screen in a transaction variant are stored in [Screen Variants \[Page 62\]](#). Each variant is assigned to a transaction. Variants may, however, contain values for screens in multiple transactions, if transaction flow makes this necessary. The transaction the variant is assigned to serves as its initial transaction, whenever you start the variant.

Both client-specific and cross-client transactions exist. Screen variants are almost always cross-client; they may, however, be assigned to a client-specific transaction.

A specific namespace has been designated for cross-client transaction variants and screen variants and they are both automatically attached to the Change and Transport System. Client-specific transaction variants can be transported manually.

Transaction and screen variants may be created for all dialog and reporting transactions, however, there are certain [Restrictions \[Page 70\]](#) that apply to their use, depending on how their corresponding transactions have been realized internally.

Transaction variants may not be created for transactions already containing pre-defined parameters (parameter transactions and [Variant Transactions \[Page 60\]](#)).

The following sections contain additional information on how to create and maintain transaction variants:

[Maintenance \[Page 50\]](#)

[Additional Functions \[Page 57\]](#)

[Transport \[Page 59\]](#)

## Maintenance

## Maintenance

You can only create transaction variants for dialog transactions and reporting transactions. Only "normal" screens, dialog boxes, and subscreens can be included in the variant.

[Screen variants \[Page 62\]](#) are automatically created anytime you create a transaction variant.

To create a transaction variant, proceed as follows:


1. Choose *Tools* → *AcceleratedSAP* → *Personalization* to call the transaction variant maintenance transaction or use transaction code SHD0.
2. Enter the name of the transaction and the transaction variant on the initial screen.

The system creates a cross-client transaction variant. If you want to create a client-specific transaction variant, choose *Goto* → *Client-specific transaction variants* to branch the client-specific transaction variant maintenance transaction.

Client-specific transaction variants only exist in the client in which they are created. The field contents of the transaction variant must be available in this client. Cross-client transaction variants are available throughout the system, regardless of the client currently being used. The field contents of these transaction variants must be available in all clients.

3. Choose *Create* to create a variant.

The system calls the application transaction that you want to create a variant for.

4. Enter the values you want to use in the input fields. Each time an action is completed (choosing  for example), a [Dialog Box \[Page 55\]](#) appears listing the fields of the current screen with their current values. The kind of dialog box called depends on the kind of screen currently being processed ( see [Requirements \[Page 52\]](#)).
5. Choose the options you want. The following options checkboxes can be selected at the top of the dialog box:

<i>Adopt field values</i>	Saves the field values you have inserted on the current screen Resetting this field (deselecting it) allows you to delete all values saved for this screen in your transaction variant.
<i>Do not display screen</i>	Hides screen This is only possible if settings are copied to your variant ( <i>Adopt field values</i> ).
<i>With contents</i>	Field contents are saved with it
<i>Output only</i>	Field is no longer ready for input (display mode only)
<i>Invisible</i>	Hides field
<i>Mandatory</i>	Required field

You may or may not be able to select each of these checkboxes for every field depending on the field's type (see [Requirements \[Page 52\]](#)).

6. Enter a screen variant name and corresponding short text if necessary.

Screen variants are created automatically for each screen where values have been saved (copied). Enter screen variant names in the *Name of screen variant*: field. These names must be unique. If the system is able to find a unique name for a screen variant it

is automatically inserted into this field. The convention reads like so:  
 <name\_of\_transaction\_variant>\_(<client>)\_<screen\_number>.

The following pushbuttons are available from this dialog box:

<i>Cancel</i>	Displays current application transaction screen again. Here you can make changes to your settings.
<i>Deactivate menu functions</i>	Deactivates menu functions in an additional dialog box.
<i>GuiXT</i>	Allows user to edit a GuiXT script for the current screen ( <a href="#">GuiXT [Page 76]</a> ).
<i>Exit and Save</i>	Exits and saves application transaction.

7. Choose the function *Continue* to proceed to the next screen in your application transaction. Repeat steps 3-5 for all further screens in the transaction.
8. When you have reached the last screen of the transaction and inserted the values you want to insert, choose *Exit and Save*. The system exits the application transaction and saves your entries. A list appears containing all of the screens in the application transaction that you want to save entries for (that is, all screens for which screen variants will be created).



You can also branch to this list using the *Change values* function during the function selection process.

9. Enter a short text for your transaction variant.  
 Display settings can be changed as needed from this list in the future. Settings that require information at application transaction runtime (field values, table control columns) cannot be changed from this list.
10. Choose *Save*. The system saves your transaction variant and the corresponding screen variants. The Workbench Organizer dialog box is displayed for the transaction variant and for each screen variant. Use it to assign each of these objects to a development class.

## Deleting Preassigned Values

You can delete all of the values you assigned to fields of a single screen in a transaction variant by resetting (deselecting) the *Adopt field values* checkbox. This deletes the screen from your variant, even those entries that were transferred to the variant during previous processing. If a screen variant has already been created for this screen, then the system simply deletes the screen variant's transaction variant assignment; the screen variant itself is not actually deleted.

Individual fields can be deleted from transaction and screen variants by resetting (deselecting) their corresponding checkboxes.

## Requirements

# Requirements

This section provides you with answers to the following questions:

- Which screens can I include in a variant?
- How many screens can I include in a variant?
- What settings can be copied for which fields?
- Under what conditions are transaction variant settings adopted?

Information about possible error dialogs that may occur during transaction variant maintenance can be found at the end of this section.

## Which Screens Can I Include in a Variant?

Only "normal" screens, dialog boxes, and subscreens can be included in the variant.

The following screens may not be included:

- Help and possible entries help (F1, F4)
- List display (and selection screens)
- Confirmation dialog boxes
- Various system functions (status display, save list...)

You can only create screen transactions for those types of screens that can be included in a transaction variant. The [Dialog Box \[Page 55\]](#) allowing you to copy values for the variant only appears for these screens.

If a screen is composed of numerous subscreens, this [Dialog Box \[Page 55\]](#) appears for each subscreen, as well as for the entire screen.

## How Many Screens Can I Include in a Variant?

In principle, every screen in a transaction that fulfills the requirements listed above can be included in a transaction variant.

A screen may only be included in a transaction variant once per transaction (however it may be included in multiple transaction variants for different transactions).

If a screen is processed more than once in this transaction, the same predefined values are inserted each time. Unless initial, field contents altered by the user or within the transaction are not overwritten with the values from the variant.

Changed field display attributes are always accepted by the variant.

## What Settings Can be Copied For Which Fields?

Field values can usually be copied for input fields. Their display attributes may be changed as well.

Exception: No field values can be copied for input fields with user-specific formatting (except date fields).

Initial values cannot be copied into the variant. Display settings may, however, be changed for these fields.

Function entries are not copied into the variant.

Exception: If a screen is meant to be hidden, this function entry is transferred to the variant for this screen.

Radio buttons may not be altered in any way.

No values can be transferred for pushbuttons.

All other fields may be hidden.

Special rules exist for:

- **Step loops and table controls**  
 You can set individual values for each input field in a step loop / table control. Display attributes (Invisible, No input) can only be set once per column (for the first field in each column); these attributes are then valid for all fields in the column.  
 All step loop and table control fields for which a value has been entered are displayed in the field list. The initial field is always displayed even if it does not contain a value so that you can use it to determine the display attributes for that column.  
 Several [Restrictions \[Page 70\]](#) apply when saving settings for step loop and table control fields.
- **Table Controls**  
 The same rules apply to table control fields and table control columns that exist for step loop fields and columns.  
 In addition, the entire table control can be hidden, simply by setting it to *Invisible*. Other table control display attributes may not be changed.  
 If a table control's column sequence or column width has been changed on a screen, these settings may also be copied to the transaction variant (*Adopt column sequence / Adopt column width*).  
 For more detailed information see [Table Control Layout in Variants \[Page 68\]](#).
- **Tabstrip Control**  
 None of the tabstrip control's display attributes may be changed. You can, however, set the tabstrip control's tabs to "Invisible". (Be careful, this may lead to errors. Pay close attention to the guidelines found in the section [Transaction Variants and Screen Variants: Special Features \[Page 72\]](#))
- **Subscreens**  
 A separate dialog box allowing you to copy field values is sent for each subscreen. For technical reasons, the *Cancel* function has been deactivated on these dialog boxes.

## Under What Conditions Are Transaction Variant Settings Adopted?

In principle, all values present in a variant are adopted each time its transaction ([Variant Transactions \[Page 60\]](#)) is started if their corresponding fields are initial. If one of the fields in question already contains a value (inserted by a user or set internally by a transaction), this value is not overwritten with the value found in the transaction variant.

Exception: If the field has been reset to initial (by a user or internally by a transaction). In this case, the variant cannot tell that the initial value found in the has been inserted explicitly. If the variant contains a value for this field that differs from the initial value, then this second value is inserted. This means that fields that have a value in the variant cannot be reset to initial because this initial value will always be overwritten by the variant.

For more information about conflicts between user entries, SPA and GPA parameters, and values set by variants, see [Transaction Variants and Screen Variants: Special Features \[Page 72\]](#).

## Requirements

Display attributes are not set when you change a variant *With processing*. The [Dialog Box \[Page 55\]](#) for determining values shows which display attributes are valid in that variant.

## Error Dialogs

From a technical point of view, there are two types of error messages:

- (a) Error messages output by the screen itself (for example, when an entry is incompatible with its field type)
- (b) Error messages output by the program.

During transaction processing, no difference is apparent between error messages of type (a) and (b). This is not the case during variant creation.

Type (a) messages are displayed BEFORE the dialog box for determining values is sent. This ensures that errors that trigger type (a) messages can be corrected before the dialog box is sent.

Type (b) messages are displayed AFTER the dialog box for determining values has been sent. The system automatically replaces incorrect values with correct ones if they are to be adopted into the variant. This is also the case with corrections that have been made due to warnings. Exception: If the "Exit and Save" function has been chosen from the dialog box, the system exits the transaction before it can send a message (b). Here, the entries can no longer be corrected.

## Dialog Box

You can use the dialog box displayed after each interaction during the transaction variant creation phase to determine values and display attributes for the current screen.

### Displaying Data

The name of the current screen (screen number and program name) is displayed in this dialog box along with the screen type for subscreens.

Enter a name and, if applicable, a short text for your screen variant here (name default: <name\_of\_transaction\_variant>\_<(client)>\_<screen\_number>).

The individual screen fields appear in the dialog box in table form. The sequence of the fields corresponds to their order in the Screen Painter field list. If a text has been stored for a field, this text is displayed in the dialog box. If no text has been stored, the technical name of the field appears.

With pushbuttons, radio buttons, checkboxes, borders, table controls, and tab strips, the field type is displayed as well.

Special rules exist for:

- **Step loops**  
All step loop fields for which a value has been entered in the transaction are included in the field list. In addition to the field name, the step loop line is displayed as well.  
The fields from the first line of a step loop are always displayed, even when no value has been entered so that you can use it to determine the display attributes.
- **Table Controls**  
The same rules apply to table control fields that exist for step loop fields.  
Table controls are segregated from other fields in the dialog box by separator lines. The attributes that can be set for table controls differ from those that can be entered for normal fields. The list of fields that belong to a table control appear after the line for the table control itself and are also set off by another separator line.
- **Subscreens**  
Since subscreens are their own screen, subscreen fields do not appear in the initial dialog box, but instead in subsequent dialog boxes of their own.
- **Tab strip controls**  
From a technical point of view, tab strip controls consist of subscreens and their corresponding pushbuttons (tabs). These tabs are part of the main screen and are therefore displayed in the field list of the dialog box for the main screen. The fields of each subscreen are displayed in separate dialog boxes as per normal for subscreens.

### Checkboxes

The checkboxes *Adopt field values* and *Do not display screen* refer to the entire screen.

- *Adopt field values*  
If *Adopt field values* is selected, your screen will accept values. If *Adopt field values* is not selected, this means that your screen will accept no values whatsoever, even if checkboxes for individual fields have been selected.
- *Do not display screen*

## Dialog Box

If *Do not display screen* is selected, this screen will not be displayed when its transaction processed (Caution, errors are possible here. For further information, see [Special Features \[Page 72\]](#)).

All other checkboxes refer to the field or screen element (table control) in their particular list line.

- *With contents*  
If the checkbox *With contents* is selected, the contents of that particular field will be transferred to the variant as long as the field is not initial.
- *Output only*  
If the *Output only* checkbox is selected, the corresponding field is no longer ready for input.
- *Invisible*  
If *Invisible* is selected, the corresponding field is not displayed.  
If the fields *Invisible* and *Output only* are both selected, the corresponding field is not displayed.
- *Mandatory*  
If the checkbox *Mandatory* is selected, the corresponding field becomes a required entry field. This is only useful for input fields whose input status (ready/not ready for input) is not changed by a transaction variant.

Table controls also have, in addition to *Invisible*, the switches

- *Adopt column sequence / Adopt column width*  
If the column sequence and/or column width of a table control has been changed, you can transfer these values to a variant by selecting the checkboxes *Adopt column sequence* and/or *Adopt column width*. For information on adopting table control settings, see [Table Control Layout in Variants \[Page 68\]](#).

Depending on their field type, checkboxes may or may not be ready for input (see [What Settings Can Be Copied for Which Fields? \[Page 52\]](#)).



## Additional Features

The following is a list of additional features available for maintaining and administering transaction variants:

Function:	Description:
<i>Create</i>	Use the <i>Create</i> function to create a new transaction variant (for more information, see <a href="#">Maintenance [Page 50]</a> ).
<i>Display</i>	Use the <i>Display</i> function to output a list of all field values stored in a transaction variant.
<i>Change</i>	<p>There are two possible ways to change transaction variants:</p> <ol style="list-style-type: none"> <li>1. Changing values without processing the transaction Use the <i>Change</i> function to branch to the list of all values stored in the transaction variant. You can change the display attributes of the fields in the list. The field values, as well as those parameters determined at transaction runtime, cannot be changed. You can save the variant on this screen.</li> <li>2. Changing values with transaction processing Use the <i>Change with processing</i> function to call the transaction whose variant you want to maintain. Enter the values you want (display attributes and field values). Those values that already exist in the variant will be displayed as defaults. (<a href="#">Maintenance [Page 50]</a>, <a href="#">Special Features [Page 72]</a>) If, after choosing <i>Change with processing</i>, the system does not run through all screens that settings have been made for (that is, through all screens with screen variants), you can determine whether or not these missing screens should also be adopted into the transaction variant using a dialog box that appears before you save the variant. If you choose not to adopt them, their corresponding screen variants still exist; only their assignment to the transaction variant is deleted.</li> </ol>
<i>Test</i>	Starts a transaction with variant. The transaction should run in call mode. Further information can be found under <a href="#">Variant Transactions, Starting Transaction Variants [Page 60]</a> .
<i>Copy</i>	Use the <i>Copy</i> function to copy a transaction variant. The new variant must not already exist. It is possible to copy client-specific variants to client independent ones and vice versa. Only the assignments are copied and not the actual screen variants.
<i>Delete</i>	Use the <i>Delete</i> function to delete a transaction variant. With client-specific transaction variants, this function applies only to the current client. Only the assignments are deleted and not the actual screen variants.

**Additional Features**

<i>Assign screen variants</i>	<p>Use the <i>Assign screen variants</i> function to assign existing screen variants to transaction variants or to delete an existing assignment. Only one screen variant can be assigned per screen.</p> <p>The system does not check to see if the screens of the transaction that belong to the screen variants are actually run through. No value conflict checks with the screens' default values are performed either.</p>
<i>Activate/Deactivate standard variant</i>	<p>Each transaction variant can be used as a standard variant (Further information can be found under <a href="#">Variant Transactions, Starting Transaction Variants [Page 60]</a>). Use the <i>Activate</i> and <i>Deactivate</i> functions to use a specific transaction variant as a standard variant or to deactivate it.</p>
<i>Client-specific/Cross-client transaction variants</i>	<p>This function allows users to branch from cross-client transaction variant maintenance to client-specific transaction variant maintenance and vice versa.</p>
<i>Screen Variants</i>	<p>The <i>Screen variants</i> function takes you from transaction variant maintenance into screen variant maintenance.</p>
<i>Create variant transaction</i>	<p>Use the <i>Create variant transaction</i> function to create variant transactions (Further information can be found under <a href="#">Variant Transactions, Starting Transaction Variants [Page 60]</a>). Enter the transaction name. Transaction type variant transaction is already selected. Enter a text for the transaction on the next screen. All other entries should already be filled with default values.</p>
<i>Catalog</i>	<p>Use the <i>Catalog</i> function to display all variants of a given transaction. You can also select this function via the <i>Possible entries</i> help for the variant field. You can use the <i>Catalog</i> function to create a list of all variants for a single or for multiple transactions.</p>
<i>Standard variants</i>	<p>Displays those transactions for which standard variants exist.</p>
<i>Transport (client-specific variants only)</i>	<p>Use the <i>Transport</i> function to transport variants to other SAP Systems. A transport request is created containing the variant(s) in question.</p> <p>Corrections and transport requests are created automatically for cross-client transaction variants when you <i>Create</i> or <i>Change</i> them (see <a href="#">Transport [Page 59]</a>).</p>
<i>Client copy (client-specific variants only)</i>	<p>Use the function <i>Client copy</i> to copy a variant to another client or to make it cross-client. (For more information, see <a href="#">Transport [Page 59]</a>). Only the screen variant assignments are copied; screen variants themselves are always cross-client.</p>
<i>Fetch from client (client-specific variants only)</i>	<p>Use the <i>Fetch from client</i> function to copy a client-specific variant from another client to a variant in the current client or to a cross-client variant.</p> <p>Only the screen variant assignments are copied; screen variants themselves are always cross-client.</p>

## Transport

### Transport into Different Clients

Transaction variants may be client-specific. You can copy a variant from another client using the function *Client copy* in variant maintenance.

Enter the name of the transaction and the variant on the initial variant maintenance screen and select the function *Client copy*. The system displays a dialog box where you can enter the following:

- the *target client* (If the variant is to be copied into all available clients, choose *All clients*)
- whether or not existing variants with the same name should be overwritten in the target client (Deselect *Overwrite in target client* if this is the case).

Choosing *Continue* copies the variant.

### Transport to Different Systems

Cross-client transaction variants and screen variants are attached to the Change and Transport Organizer.



Whenever a transaction variant is transported, only the screen variant assignments are transported; screen variants themselves are not automatically transported with the transaction variant.

Client-specific transaction variants must be transported manually using the *Transport* function.

When you call the *Transport* function, a dialog box is displayed where you can enter a transport request. The variant you specified on the initial screen is then automatically entered in this transport request. The variant name can also be entered generically. All variants that match this generic name are then included in the transport request.

## Starting Transaction Variants

### Starting Transaction Variants

A transaction can be started with a variant:

- Using the *Test* function from the maintenance transaction
- By a program that includes a call to function module RS\_HDSYS\_CALL\_TC\_VARIANT



Function module RS\_HDSYS\_GET\_TC\_VARIANT checks to see if a transaction variant is active and returns its name and (if possible) its current values (screen variants, inactive functions).

- Using its own transaction code as a variant transaction.



When a transaction with variant is called using the *Test* function, for technical reasons it is started in call mode. Some transactions behave differently in call mode than when called directly.

A variant transaction is started either in call mode or LEAVE TO TRANSACTION, depending to how it is called.

If you call a transaction with variant from a program using the function module RS\_HDSYS\_CALL\_TC\_VARIANT, you can use the function module to determine which mode is called (call mode or LEAVE TO TRANSACTION).

### Variant Transactions

A transaction with a variant can have its own transaction code, which you can then use to execute the transaction with the appropriate variant.

To create a variant transaction, proceed as follows:

1. Choose *Goto* → *Create vari. trans.* from the initial transaction variant maintenance screen.
2. Enter a transaction name and corresponding short text. Transaction type *Variant transaction* is already set. The following screen already contains default values for the name of the calling transaction, the variant name, and the client-specific or cross-client variant checkbox.
3. Save your changes.

#### Example:

Variant ABC has been defined for transaction Z000. We now want to create the variant transaction ZVAR.

- Name of the new transaction: ZVAR
- Name of the calling transaction: Z000
- Variant name: ABC

Transaction ZVAR starts transaction Z000 with variant ABC.

If the variant specified for a variant transaction does not exist (in the current client), then the calling transaction will be started without variant.

## The Standard Transaction Variant

Standard transaction variants have a special purpose. You can use any transaction variant as a standard variant. Use the *Activate* and *Deactivate* functions to use a specific transaction variant as a standard variant or to deactivate it.

If a standard variant is active for a transaction, the system searches for the variant automatically whenever the transaction is started and adopts its values.

If a transaction with an active standard variant is started with another variant, the values of the standard variant are not imported.

If a transaction with an active standard variant is started in batch input mode, the values of the standard variant are not imported either.

## Variant Transactions and Parameter Transactions

- Variant transactions can be client-specific (this means that their corresponding variants can also be client-specific).
- In a variant transaction, several screens can be given pre-assigned values or suppressed. In a parameter transaction this can only be done for the initial screen.
- You can suppress as many screens as you like in variant transactions; in parameter transactions you can only suppress the initial screen. If a screen is suppressed in a variant transaction, it is never displayed. An initial screen that has been suppressed in a parameter transaction is only suppressed when the transaction is initially called.

You can use the function *Change transaction type* during transaction code maintenance to convert a parameter transaction to a variant transaction. The system creates a transaction variant and a screen variant for its initial screen containing the values of the parameter transaction. The system suggests the name CV\_P\_<name of the variant transaction> for these variants. The default name may be changed. The variant is created for the called transaction. It can be changed in [Transaction Variant Maintenance \[Page 50\]](#) and [Screen Variant Maintenance \[Page 63\]](#). The transaction variant can be created in the current client or as a cross-client variant.

---

**Screen Variants**

## Screen Variants

Screen variants allow you to simplify screen editing by:

- Inserting default values in fields
- Hiding and changing the ready for input status of fields
- Hiding and changing the attributes of table control columns

A screen variant contains field values and attribute for exactly one screen. A screen variant may be assigned to multiple transaction variants, however. Screen variants are always cross-client; they may, however, be assigned to a client-specific transaction. They can also be called at runtime by a program. The different possibilities for calling screen variants guarantee great flexibility of use.

A specific namespace has been designated for screen variants and they are automatically attached to the Change and Transport System.

For more information, see:

[Maintenance \[Page 63\]](#)

[Additional Functions \[Page 65\]](#)

[Calling Screen Variants \[Page 67\]](#)

## Maintenance

Choose *Goto* → *Screen variants* from the initial transaction variant maintenance screen to maintain screen variants.

With the functions

- *Create*
- *Change with Processing*
- *Test*

enter the name of an initial transaction and the name of the screen variant. Only the name of the screen variant is necessary with all other functions available on the screen.

With the functions

- *Create*
- *Change with Processing*

you may choose to edit multiple screen variants, if you so desire.

If you only want to edit one screen variant, you must enter the program name and screen number in addition to the name of the initial transaction and name of the screen variant. Here the dialog box listing field values is only sent for the specific screen you want. The name of the screen variant that you entered on the initial screen appears in the dialog box.

If you want to edit multiple screen variants, leave program name and screen number initial. The dialog box for adopting field values is sent for each screen that is processed in the transaction. Here you can decide whether or not you want to create a screen variant for the current screen. No default names are suggested for screen variants in the dialog box.

The name of the screen variant is the only required entry for all other functions available on the screen.

To create a screen variant, proceed as follows:

1. Enter the name of a transaction and the name of the screen variant.
2. Choose *Create*.

The application transaction is called and default values may be entered. A [Dialog Box \[Page 55\]](#) appears listing the current screen's fields and their current values. This dialog box appears once for every screen variant you create (see above).

3. Choose the options you want. The following options checkboxes can be selected at the top of the dialog box:

<i>Adopt field values</i>	Saves the field values you have inserted on the current screen Resetting this field (deselecting it) allows you to delete all values saved for this screen in your transaction variant.
<i>Do not display screen</i>	Hides screen This is only possible if settings are copied to your variant ( <i>Adopt field values</i> ).
<i>With contents</i>	Field contents are saved with it
<i>Output only</i>	Field is no longer ready for input (display mode only)

**Maintenance**

<i>Invisible</i>	Hides field
<i>Mandatory</i>	Required field

You may or may not be able to select each of these checkboxes for every field depending on the field's type (see [Requirements \[Page 52\]](#)).

- Enter a name and a short text for your screen variant here (name default:<name\_of\_transaction\_variant\_subsequent\_number\_screen\_number>).

The following pushbuttons are available from this dialog box:

<i>Continue</i>	Application transaction continues (proceeds)
<i>Cancel</i>	Displays current application transaction screen again. Here you can make changes to your settings.
<i>Deactivate menu functions</i>	Deactivates menu functions in an additional dialog box.
<i>GuiXT</i>	Allows user to edit a GuiXT script for the current screen ( <a href="#">GuiXT [Page 76]</a> ).
<i>Exit and Save</i>	Exits the application transaction. A list of all of the screens in the application transaction that have screen variants is displayed. The display attributes can be changed from the list. The field values can, however, no longer be changed.

- Save your screen variants. The Change and Transport Organizer's dialog box is displayed for each screen variant. Use it to assign each of these objects to a development class.

Further information can be found under [Transaction Variant Maintenance \[Page 50\]](#).



## Additional Features

Function:	Description:
<i>Create</i>	Use the <i>Create</i> function to create new screen variants (Further information can be found under <a href="#">Screen Variants: Maintenance [Page 63]</a> ).
<i>Change with Processing/Change</i>	<p>There are two possible ways to change screen variant values:</p> <ol style="list-style-type: none"> <li>1. Changing values without processing the transaction. Use the <i>Change</i> function to branch to the list of all values stored in the screen variant. You can change the display attributes of the fields in the list. The field values, as well as those parameters determined at transaction runtime, cannot be changed.</li> <li>2. Changing values with processing the transaction. Use the <i>Change with processing</i> function to call the transaction specified on the initial screen. The dialog box for the screen associated with the screen variant entered is processed, and you can change the values (display attributes and field values). Those values that already exist in the variant will be displayed as defaults (for more information, see <a href="#">Screen Variants: Maintenance [Page 63]</a>).</li> </ol>
<i>Display</i>	Use the <i>Display</i> function to output a list of all field values stored in a screen variant.
<i>Check</i>	<p>Use the <i>Check</i> function to check screen variants containing table control layout for consistency.</p> <p>If the column sequence of a table control has been changed in a screen variant, the variant must contain all table control columns. If this is not the case, the column sequence is automatically corrected at runtime. This is, however, undesirable. When these kinds of variants are tested, a message is displayed urging the user to adjust the variant.</p> <p>The <i>Check</i> function allows you to check all of a screen's screen variants for consistency. A list is generated of all screen variants that need to be adjusted. You can select individual variants from this list and adjust them using the <i>Adjust and Save</i> function.</p>
<i>Test</i>	The <i>Test</i> function allows you to test a screen variant. The transaction entered on the initial screen is started and values inserted for the screen variant's corresponding screen. The transaction should run in call mode. For more information, see <a href="#">Starting Transaction Variants [Page 60]</a> .
<i>Where-used list</i>	Use the function <i>Where-used list</i> to display a list of which transaction variants your screen variant is used in. Screen variants called by a program (see <a href="#">Calling Screen Variants [Page 67]</a> ) do not appear.

**Additional Features**

<i>Delete</i>	The <i>Delete</i> function allows you to delete a screen variant. The variant may then no longer be used in a transaction variant. The function <i>Where-used list</i> allows you to determine which transaction variants your screen variant is used in. Use the <i>Assign screen variants</i> function (see <a href="#">Transaction Variants: Additional Features [Page 57]</a> ) to delete a screen variant from a transaction variant.
<i>Copy</i>	The <i>Copy</i> function allows you to copy a screen variant. The new variant must not already exist.
<i>Transaction variants</i>	The <i>Transaction variants</i> function takes you from screen variant maintenance back to transaction variant maintenance.

## Calling Screen Variants

Screen variants are called:

- Automatically whenever you start the transaction variant that they are assigned to
 

The field values and attributes stored for a screen are technically saved in the screen variant. When a transaction variant is processed (see [Starting Transaction Variants \[Page 60\]](#) ), these values are inserted on the transaction's screens from those screen variants that are both available in the system and assigned to that particular transaction variant. You cannot set different screen variants according to case at this juncture.
- By a program that includes a call to function module RS\_HDSYS\_SET\_SC\_VARIANT
 

This technique allows you to set different field values and attributes for a screen according to the current context.

When is it helpful to call screen variants using a program?
- When the same screen is used by multiple transactions
 

Transaction A and transaction B have the same initial screen. However, the subsequent screens of these two transactions differ from one another. Screen variant A or B can be set for the initial screen depending on whether transaction A or B is being processed. By calling the appropriate screen variant with a program, you avoid having to create two different transaction variants.
- When different settings are necessary with the save screen within a transaction
- When a subscreen has several different settings
 

Screen variants can also be created for subscreens. This allows you to send the same subscreen with different values and attributes.
- When different table control settings exist within a screen
 

You can adopt table control layout guidelines in screen variants (see [Table Control Layout in Variants \[Page 68\]](#)). You can also create multiple screen variants for a table control, each with a different layout, and call these variants according to context.

Function module RS\_HDSYS\_GET\_SC\_VARIANT checks if a screen variant is active for the current screen and returns with the variant's name and current values.

---

**Table Control Layout in Variants**

## Table Control Layout in Variants

This section contains information on the following topics:

- The Administrator Function and Table Controls in Screen Variants
- Adopting Table Control Layout in Screen Variants
- Setting and Resetting Table Control Layout in Screen Variants
- Inserting Additional Columns in Table Controls

### The Administrator Function and Table Controls in Screen Variants

You can adopt layout guidelines for table controls in a screen variant.

The same functions are available here as found in the administrator function in a table control:

- Determining column sequence
- Determining column width
- Hiding columns

In addition, you can also:

- Revoke the ready for input status of individual columns
- Define default values for table controls in certain circumstances

Since screen variants can be transported, their corresponding table control layouts can also be transported as well.

If table control administrator settings exist and the layout has been defined using a screen variant, the screen variant's layout takes precedence (screen variant layout is set after PBO and thus overwrites the administrator settings).

A table control's user settings are not overwritten by screen variant settings except in the following cases:

- Hidden columns remain invisible
- Columns not ready for input remain locked
- Screen variant default values are adopted

The first two points are important, otherwise columns could be seen or changed by user groups that they should not be available to.

### Adopting Table Control Layout in Screen Variants

You can define specific table control options when creating or changing a screen variant just as you can with other screen elements.

Table controls are displayed in the following manner: Those attributes that are valid for the entire table control are displayed first-- set off from the others by underscores. They are:

- Adopt column sequence
- Adopt column width
- Invisible (that is, hide entire table control)

## Table Control Layout in Variants

If *Adopt column sequence* or *Adopt column width* is selected, the column sequence or column width that was just set is saved to your screen variant.

After all of the attributes for the table control as a whole are displayed, all of the columns are displayed in the sequence in which they were defined in the Screen Painter (NOT in the sequence set at runtime).

All fields containing a value are displayed; in addition to the field names, the line number is displayed as well.



Since the system cannot always tell when display masks are initial, their fields are sometimes displayed as well, even though they are initial.

You can adopt values in your variant for all table control lines except the selection field (certain [Restrictions \[Page 70\]](#) apply).

The first field of each column is always displayed so that the display attributes for the column (*Invisible*, *No entry*) can be set.

Display attributes can be set only once per column and apply to the whole column.

## Setting and Resetting Table Control Layout in Screen Variants

When [Calling Screen Variants \[Page 67\]](#) that contain table controls, the layout of the table control is adopted automatically.

The layout settings are changed directly in the application program. These changes retain validity until they are overwritten, either by another screen variant or by the application program itself.

Layout settings remain active, even after screen variants have been explicitly reset and are no longer active.

You can alter the table control layout by calling a new screen variant containing the original table control layout settings.

## Inserting Additional Columns in Table Controls

Whenever you attach additional columns to a table control for which screen variants have already been created, the screen variants can no longer determine the column sequence since this could lead to multiple column indexes (all other options remain unchanged).

In this case all of the screen variants belonging to your table control must be adjusted. All new columns should be inserted at the beginning of the screen variant.

If you do not adjust these variants, the message *Column sequence could not be set* is displayed at runtime.

## Transaction Variants and Screen Variants: Restrictions

### Leave to Transaction/Call Transaction

If a LEAVE TO TRANSACTION XYZ or CALL TRANSACTION XYZ is processed in an application transaction where XYZ is the transaction and not the variant transaction, the reference to the variant is lost. This means that from this point on, values can no longer be imported from the variant.

Using a standard variant is one solution to this problem. No new transaction codes are necessary with standard variants. However, be aware that in this case all transaction users will automatically be provided with the standard variant's values and attributes.

### Differing Display Attributes in Different Step Loop and Table Control Lines

Transaction variants can only change step loop and table control display attributes column for column.

If step loop or table control fields from different lines have different display attributes (for example, the initial line is ready for input and the subsequent line is display only), the following may occur:

1. When adopting a new value in a step loop or table control field, the display attributes for the entire column may change. Whenever you adopt fields that have different display attributes in different lines of the same step loop or table control, the entire column takes on the display attributes that were valid in the last line.
2. Certain step loop/table control fields should be made into display only fields; under certain circumstances, these fields may, however, be hidden. Maximizing windows or choosing *Next page* can have the same effect. This also happens with certain window sizes. The number of fields displayed does not change.

### Step Loops and Table Controls Extending over Multiple Pages

If a table control extends over several pages on your screen, no values can be set for it since the system cannot determine where (in which line) the value should be set. The same is true for step loops.

### Re-Sorting Entries in Step Loops and Table Controls

If step loop or table control fields are re-sorted internally in an application transaction (alphabetically, for example), the transaction variant's values appear more than once and overwrite those entries entered manually.

The transaction variant's values can only be determined statically for specific line numbers in the step loop. If you want to allow further changes to be made, no field contents can be adopted here.

### Consolidating Screens

You cannot use transaction variants to hide fields in various screens and subsequently consolidate these screens into a single new screen.

Technical background: One of the advantages of transaction variants is that the program logic of the application transaction in question does not need to be altered. Consolidating multiple

## Transaction Variants and Screen Variants: Restrictions

screens into a single screen would make it necessary to alter flow logic and SAP discourages this.

### Additional Screen Elements

You cannot use transaction variants and screen variants to add additional elements to a screen. Variants should be used to simplify transaction flow by reducing the complexity of screens and transactions. You may, however, add certain elements to a screen by creating a corresponding GuiXT script (see [GuiXT \[Page 76\]](#)).

### No Screen Sequence Control in Transaction Variants

Function codes are only stored in transaction variants if a screen is to be hidden using a variant. In all other cases, function codes are not saved, which means that the screen sequence control is not recorded. ([What Settings Can Be Copied for Which Fields? \[Page 52\]](#)) Only the field values and field attributes for specific screens are saved. This allows a transaction variant to be used in different transaction flow. When a screen from the variant is processed, those values stored in the variant are inserted at the appropriate spots. No values are inserted for those screens contained in the variant but not processed at runtime. This does not lead to errors.

Be aware of the fact that each screen can only be saved once per variant. (see also: [Which Screens are Included in the Variant? \[Page 52\]](#) ). This means that you cannot create a transaction variant for the following transaction flow: Say the user wants to select different menu entries, one right after the next. In doing so, the user branches once from screen A to screen B, and from there to screen C. If the user chooses another menu entry, he or she branches from screen A to screen B, and then to screen D. In this example, the user wants to create different values for screen B. This is not possible, because values last entered for screen B are saved and overwrite those entered before them.

## Transaction Variants and Screen Variants: Special Features

### Using Variants to Suppress Screens

Whenever a screen for a particular transaction is to be suppressed using a transaction variant, the function code is saved in the variant so that the transaction can resume with the following screen. It is especially easy to use the function *Exit and Save* to exit the dialog box for value creation and mistakenly save the function code that keeps you on the 'invisible' screen. When this happens, the program is terminated with the message

MS419: *Variant error: Screen XXX nn was processed more than 100 times.*

If the invisible screen possesses a function code for leaving screens, it may be not be possible to leave the next screen.

Example: A transaction is made up of two screens: Initial screen A and the next screen B. On screen B, the function *Back* allows the user to go back to the initial screen A. Suppose that a variant is used to make screen A invisible: Now, whenever the transaction is processed, screen A is processed invisibly, immediately followed by B, the first visible screen in this transaction. If *Back* is chosen on screen B, then A is once again processed invisibly, followed by B. To the user it looks as if screen B has never been left.

### No Screen Sequence Control in Transaction Variants

Function codes are only stored in transaction variants if a screen is to be hidden using a variant. In all other cases, function codes are not saved, which means that the screen sequence control is not recorded. (see also: [What Settings Can Be Copied for Which Fields? \[Page 52\]](#)) Only the field values and field attributes for specific screens are saved. This allows a transaction variant to be used in different transaction flow. When a screen from the variant is processed, those values stored in the variant are inserted at the appropriate spots. No values are inserted for those screens contained in the variant but not processed at runtime. This does not lead to errors.

Be aware of the fact that each screen can only be saved once per variant. (See also: [Which Screens are Included in the Variant? \[Page 52\]](#) ). This means that you cannot create a transaction variant for the following transaction flow: Say the user wants to select different menu entries, one right after the next. In doing so, the user branches once from screen A to screen B, and from there to screen C. If the user chooses another menu entry, he or she branches from screen A to screen B, and then to screen D. In this example, the user wants to create different values for screen B. This is not possible, because values last entered for screen B are saved and overwrite those entered before them.

### Tabstrip Controls

Tab strip controls are made up of pushbuttons (the tabs) and several subscreens. Each tab has its own subscreen. These tabs are part of the main screen and are therefore displayed in the field list of the dialog box for the main screen. Subscreen fields are displayed in other dialog boxes.

According to how a tabstrip has been constructed internally:

- Either a sole dialog box is displayed for the subscreen that belongs to the tab that is currently active, or
- Dialog boxes are displayed for all subscreens that belong to the tabstrip control.



## Transaction Variants and Screen Variants: Special Features

If an invisible tab is set to *active* by the application transaction, different subscreens are selected for those tabs remaining that actually belong to other tabs. If the subscreens are invisible, it is also possible that an empty screen will be displayed.

Technical Background:

The transaction variant sets another tab (the remaining tab furthest left) to active. This additional active tab is, however, unknown in the application transaction, which assumes that its tab is active and selects the corresponding subscreen. If this subscreen has been faded out using the transaction variant, then an empty screen is displayed.

### Checkboxes

Information on this topic can be found under [What Settings Can Be Copied for Which Fields? \[Page 52\]](#).

#### ***With contents is Not Ready for Input with Input Fields***

No values can be transferred for numeric fields since the formatting (decimal representation) for these fields is user-specific and due to technical reasons no user-specific settings can be tampered with during variant processing. Date fields can, however, be transferred. The user-specific setting here is pulled at the beginning of variant processing.

#### ***With contents is Ready for Input with Non-Input Fields***

If a field only becomes an input field during runtime, all switches in the dialog box are ready for input, but only the switch *Invisible* takes effect.

### Screen Compression

In principle, screen compression works if all screen elements are hidden using a variant. Screen compression can be explicitly switched off for certain application transactions (for example, FB01). Variants are not valid in this case.

### Interaction between Application-Specific Field Selection Control, SPA and GPA Parameters, GuiXT, and Transaction and Screen Variants

There are several different ways to simplify user interface layout in the R/3 System (application-specific field selection control, SPA/GPA parameters, screen and transaction variants, and GuiXT). For details, see [Designing User Interfaces in the R/3 System \[Page 17\]](#).

The following rules apply when using more than one of these techniques concurrently:

- Hiding screen elements
  - Screen elements that have been hidden by a function remain hidden. This is also true if the element is set to *Display only* by another function.
- Default values
- SPA/GPA parameters and screen and transaction variants
  - Default values for SPA/GPA parameters may be overwritten by screen and transaction variant values sometimes.
  - Default values from screen and transaction variants are given priority if the variant hides a field or revokes its ready for input status. These values also have priority if SET parameter is used to set an initial default value (SPACE, for example). On the other hand, SET parameter default values have priority in ready for input fields since the

## Transaction Variants and Screen Variants: Special Features

variant cannot tell the difference between values set by SET parameters and user entries.

- GuiXT combined with other functions:

GuiXT values are only set if the field in question has an initial value. This also applies if the value was set by the transaction, SPA/GPA, or a variant.

- Hidden screens

All screens hidden using transaction variants or screen sequence control remain hidden.

## Upgrade Procedures

### Overwriting Variants

- Up to Release 4.5B

Up to and including Release 4.5B there was no namespace for transaction variants and they were not attached to the Change and Transport Organizer. Normally, transaction variants were not delivered.

- From Release 4.6A

From Release 4.6A a namespace exists for cross-client transaction and screen variants and they are attached to the Change and Transport Organizer. This means that customers' cross-client transaction and screen variants are not overwritten by variants delivered by SAP.

### Changing Screens: Possible Problems

Transaction variants are usually tolerant of screen changes. In certain cases, however, it may be necessary to adjust your screen or transaction variant.

- Additional screen elements or deleted screen elements

If a screen contains additional screen elements not contained in the screen or transaction variant, the variant ignores these fields. The same is true for screen elements contained in the variant that have been deleted. In this case the settings save for the element cannot be set (adopted). This does not lead to errors.

Exception: If a table control's column sequence has been saved in the variant and columns have been subsequently added, then the variant must be adjusted.

- Renamed screen elements, screen elements whose type and/or length has been changed

If screen elements present in the variant have a new name, length, or type, this can lead to errors at runtime. In this case, you must adjust the variant.

- Deleted screens

If a variant contains screens that have been deleted, the variant ignores these screens. This does not lead to errors.

### Batch Input - Standard Variants

If a standard variant is active for a transaction and the transaction is to be executed during batch input, the values from the standard variant are not inserted.

## Batch Input - Variant Transactions

You can create and run batch input folders for variant transactions.

## GuiXT in Transaction Variants and Screen Variants

GuiXT allows you to alter transaction screen layout to fit your or your company's specific needs. Specifically, GuiXT allows you to move fields, change texts, and add new elements (texts, images, pushbuttons, radio buttons, etc.).

Layout modification is realized using a script that is stored on your frontend. The GuiXT process interprets these scripts and modifies screen layout accordingly; your transaction's logic remains untouched.

Use the *GuiXT* function during transaction variant maintenance to edit a GuiXT script. The script is then stored together with its corresponding screen variant in the database. The GuiXT process downloads the script to the frontend if no current version is already on the frontend. GuiXT scripts are transported along with their corresponding screen variants.

If you want to display GuiXT images on your screens, you should familiarize yourself with the guidelines contained in the section of [Storing Images for Transaction Variants \[Ext.\]](#).

## Business Add-Ins

Business Add-Ins are a new SAP enhancement technique based on ABAP Objects. They can be inserted into the SAP System to accommodate user requirements too specific to be included in the standard delivery. Since specific industries often require special functions, SAP allows you to predefine these points in your software.

As with customer exits ([SMOD/CMOD \[Page 40\]](#)), two different views are available:

- In the definition view, an application programmer predefines exit points in a source that allow specific industry sectors, partners, and customers to attach additional software to standard SAP source code without having to modify the original object.
- In the implementation view, the users of Business Add-Ins can customize the logic they need or use a standard logic if one is available.

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

SAP guarantees the upward compatibility of all Business Add-In interfaces. Release upgrades do not affect enhancement calls from within the standard software nor do they affect the validity of call interfaces. You do not have to register Business Add-Ins in SSCR.

The Business Add-In enhancement technique differentiates between enhancements that can only be implemented once and enhancements that can be used actively by any number of customers at the same time.

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

All ABAP sources, screens, GUIs, and table interfaces created using this enhancement technique are defined in a manner that allows customers to include their own enhancements in the standard.

A single Business Add-In contains all of the interfaces necessary to implement a specific task. In Release 4.6A, program and menu enhancements can be made with Business Add-Ins.

The actual program code is enhanced using ABAP Objects. In order to better understand the programming techniques behind the Business Add-In enhancement concept, SAP recommends reading the section on [ABAP Objects \[Ext.\]](#).

More information about Business Add-Ins is contained in the following sections:

[Business Add-Ins: Architecture \[Page 79\]](#)

[A Comparison of Different Enhancement Techniques \[Page 81\]](#)

[Defining Business Add-Ins \[Page 82\]](#)

[Calling Add-Ins from Application Programs \[Page 85\]](#)

[Implementing Business Add-Ins \[Page 87\]](#)

[Filter-Dependent Business Add-Ins \[Page 89\]](#)

[Multiple Use Business Add-Ins \[Page 93\]](#)

---

**Business Add-Ins**

[Menu Enhancements \[Page 95\]](#)

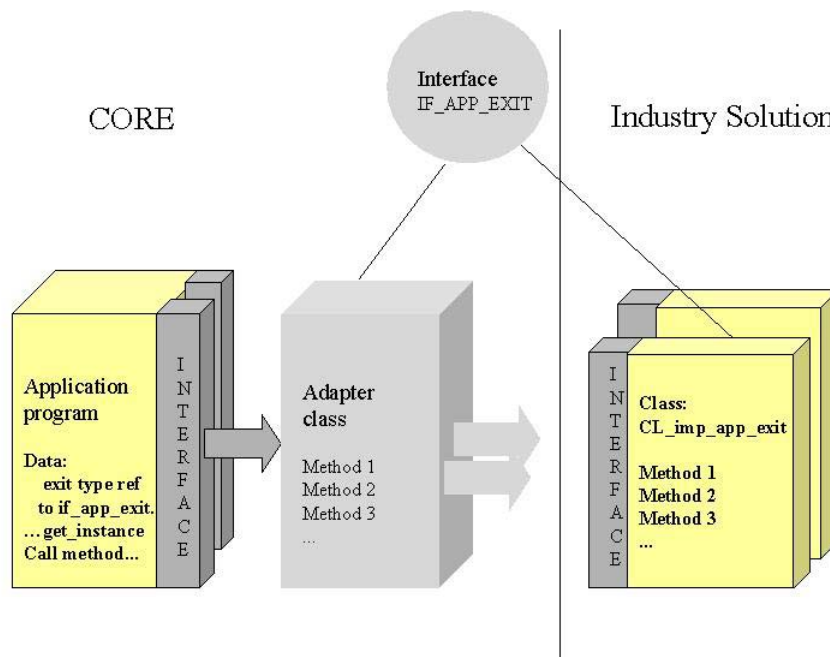
[Business Add-Ins: Import Procedure \[Page 97\]](#)

## Business Add-Ins: Architecture

In order to enhance a program, a Business Add-In must first be defined. Application developers create an interface for the add-in. Enhancement management takes this interface and generates an adapter class for implementing it, thus opening a path for implementations created by partners or customers. Your developer then creates an instance of the adapter class in the application program and calls the corresponding method at the appropriate time.

Customers can find the enhancements present in their system in the IMG and in the component hierarchy. Whenever they want to use a Business Add-In, they must create their own implementation of the add-in. Customers must first implement methods and user interface enhancements, and then activate their implementations of the enhancement. The enhancement's active components are then called at runtime.

Normally, a Business Add-In contains an interface and other additional components such as function codes for menu enhancements. Starting with the next release, Business Add-Ins will also include enhancements for screens and tables. The enhancement, interface, and associated classes generated all lie in the appropriate application development namespace. Business Add-In implementations lie in the respective namespaces of the people who created them.



The following sections contain examples of both program enhancements and menu enhancements.





## A Comparison of Different Enhancement Techniques

Due to the necessity of adjusting R/3 to meet the specific needs of a variety of customers, several different enhancement techniques were developed in the past. A short description of each of the various enhancement techniques follows.

### Business Transaction Events (Open FI)

The Open FI enhancement technique was developed in the Financial Accounting component. Open FI is based upon the following principles:

Application developers must define their interface in a function module, an assignment table is read in the accompanying (generated) code, and the customer modules assigned are called dynamically.

This technique differentiates between enhancements that are only allowed to have one implementation and enhancements that can call multiple implementations in any sequence desired. Both industry-specific and country-specific enhancements may be defined.

The concepts behind the Business Add-Ins enhancement technique and Open FI are basically the same. However, the two enhancement techniques do differ from each other in the following points:

- Open FI can only be used to make program enhancements, that is, you can only enhance source code using Open FI. You cannot enhance user interface elements with Open FI like you can with Business Add-Ins.
- Open FI assumes that enhancement will only take place on three levels (SAP - partners - customers), whereas with Business Add-Ins you can create and implement enhancements in as many software layers as you like.
- Open FI uses function modules for program enhancements. With Business Add-Ins, ABAP Objects are used enhance programs.

### Enhancements in Transactions SMOD/CMOD

Making enhancements using the transactions [SMOD/CMOD \[Page 40\]](#) has the following disadvantages:

- This enhancement technique assumes a two-tiered system infrastructure (SAP – customers).
- The naming conventions in effect do not tolerate name extension.

### Taking Stock of Your Options:

None of the techniques mentioned above can easily be extended to fulfill the requirements of a system infrastructure containing country versions, industry solutions, partners, and customers.

Business Add-Ins should be considered generalized Business Transaction Events that can be used to bundle menu and program enhancements into a single add-in (and in the future, screen and table enhancements as well). Business Add-Ins can be created and employed in each of the various software levels.

## Defining Business Add-Ins

## Defining Business Add-Ins

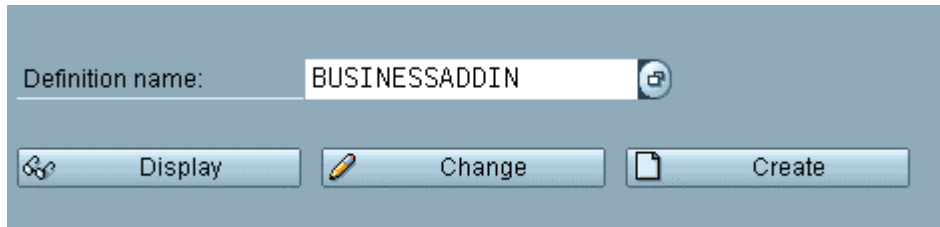
In order for application developers to include Business Add-Ins in their programs, they must define an interface for the enhancement in the SAP menu under *Tools* → *ABAP Workbench* → *Utilities* → *Business Add-Ins* (transaction SE18) and call this interface at the appropriate point in their application program. Customers can then select the add-in and implement it according to their needs.

**Example:**

You want to be able to convert strings in your application program. You also want users to determine how their strings are converted themselves. As the application developer, you define an enhancement consisting of an interface with a method. A changing parameter is used to transfer strings.

In order to create an add-in like this, proceed as follows:

1. Choose *Tools* → *ABAP Workbench* → *Utilities* → *Business Add-Ins* (transaction SE18) from the SAP menu.



2. Enter a name for your Business Add-In containing a maximum of 20 characters.
3. Choose *Create*.
4. Enter a short description for your Business Add-In on the following screen.



If this enhancement is going to be used multiple times or if you want its implementation to depend on a specific filter value, first choose *Management* and then select the appropriate checkbox in the category *Type*. More information about multiple use and filter dependency can be found in the sections [Multiple Use Business Add-Ins \[Page 93\]](#) and [Filter-Dependent Business Add-Ins \[Page 89\]](#). This example only deals with basic enhancements, therefore neither of these checkboxes should be selected.

5. Choose the *Interface* tab.  
The name of the interface is generated automatically and now appears along with the name of its corresponding class. At this time, you can change the name of the interface if you so desire.

Defining Business Add-Ins

Definition name	BUSINESSADDIN
Definition short text	Word conversion add-in
<div> <div>Attributes</div> <div>Interface</div> <div>FCodes</div> </div>	
Interface name	IF_EX_BUSINESSADDIN
Name of generated BAdI class:	CL_EX_BUSINESSADDIN



You can use the *Fcodes* tab to create menu enhancements. For more information, refer to the section on [Menu Enhancements \[Page 95\]](#).

6. Double-click on the interface's name field to assign a method to the interface.
7. The system branches to the Class Builder. A dialog box appears, asking you if you want to save the entries you have made. Save your entries. Assign your add-in to a development class.
8. Use the Class Builder to assign a method to the interface.

Interface		IF_EX_BUSINESSADDIN		Implemented / Inactive	
<div> <div>Attributes</div> <div>Interfaces</div> <div>Attributes</div> <div>Methods</div> <div>Events</div> </div>					
Parameters		Exceptions		<div> <div>Parameters</div> <div>Exceptions</div> <div>Methods</div> <div>Events</div> </div>	
Methods	Level	Mo...	C...	Description	
Method		<input type="checkbox"/>		Word conversion add-in	

9. Now define a parameter with the following attributes:

Interface		IF_EX_BUSINESSADDIN		Implemented / Inactive(revised)	
<div> <div>Attributes</div> <div>Interfaces</div> <div>Attributes</div> <div>Methods</div> <div>Events</div> </div>					
Method parameters		METHOD			
Methods		Exceptions		<div> <div>Parameters</div> <div>Exceptions</div> <div>Methods</div> <div>Events</div> </div>	
Parameter	Type	P...	O...	Typing	Reference type
PARAMETER	Changi...	<input type="checkbox"/>	<input type="checkbox"/>	Type	C
				Default value	Description
					Conversion

10. Save and activate your changes. Use the pushbutton *Back* to navigate back to Business Add-In definition.



If you do not activate your attributes in the Class Builder, the system will not allow you to proceed with Business Add-In definition.

A table control now appears on the definition screen containing the method you have assigned to the interface.

## Defining Business Add-Ins



Whenever you assign a method to an interface, the corresponding executing class is generated. The code generated cannot be altered in the initial expansion phase.

11. Save your entries and use the *Def.-Docu.* pushbutton to create a description for your new Business Add-In. Be aware that this documentation is of great importance in helping end users understand the purpose of your add-in.



### **Changes made to the interface and changes made to the Business-Add-In definition are always incompatible!**

If implementations already exist for a Business-Add-In definition, they are invalidated if you make changes to the interface. This means that their syntax is no longer correct. No statements can be given on the runtime behavior. Try to absolutely avoid making changes to the interface or the Business-Add-In definition after the transport has taken place.

If changes to the interface are inevitable, navigate to the Class Builder (transaction SE19, tab *Interface*, field *Name of implementing class*) for all implementing classes, that is, all classes for which Business-Add-In implementations are used, and clean up the method includes of these classes (*Utilities* → *Clean up* → *Method includes*).

## Default and Sample Code

In the BAdI Builder, you can choose the *Goto* menu entry to create, display, change and delete default or sample code.

The default implementation is only executed if no other active implementation is available. This applies also to filter-dependent Business Add-Ins.



When you create default or sample code, do not forget to save your entries to ensure that the link between the class and the implementation can be established.

## Calling Add-Ins from Application Programs

When you define a Business Add-In, enhancement management generates a class that implements your interface. Application developers use factory methods to create instances of these adapter classes in their application programs and call the corresponding method if necessary.

The adapter class methods generated by add-in management decide if multiple active implementations should be called. If necessary, these implementations are subsequently executed. The application program itself simply calls the adapter class methods; it does not know which implementations are actually being called.

### Example:

Having created a string conversion Business Add-In, you would program the call of the Business Add-In into your ABAP source code as follows:

```
Report businessaddin.
class cl_exithandler definition load.           "declaration
data exit type ref to if_ex_businessaddin.    "interface reference
data word(15) type c value 'Business Add-in'.  "string you want to
change

start-of-selection.
  call method cl_exithandler=>get_instance      "factory method call
    changing instance = exit.
  write:/ 'Please click here'.

at line-selection.
  write:/ 'Original word: ',word.

  call method exit->methode                      "add-in call
    changing parameter = word.

  write:/ 'Changed word: ',word.
```

In order to be able to call static methods, you must declare the corresponding class in ABAP Objects. This is why the `class ... definition load` statement is necessary for the factory class.

A variable for object reference is also necessary when calling the method. Use `data` to create it and type it to the interface.

Application developers use factory methods to create instances of adapter classes during initialization in lines 7 and 8. The instance methods are then called at the appropriate time.

### Notes on Usage

The instance generated through the factory method should be declared as globally as possible or generally be passed as a parameter to ensure that the initialization process must be run through as rarely as possible - one time would be best. In no case should you discard the instance as soon as it is generated or repeatedly run the initialization process in a loop.

Within the instance of the adapter class, required database accesses are buffered locally, so that each access is executed only once. However, repeated initialization makes the buffer useless and dramatically reduces performance.

---

**Calling Add-Ins from Application Programs**

Due to the local buffering, you can call Business-Add-In methods without having to expect considerable performance restrictions even if no active implementations exist.

In particular, you are not required to use a function module to verify if active implementations exist at all.

Also, if the definition of the Business-Add-In is filter-dependent, a single instance is sufficient.

However, you should not do without initialization at all. Although it would be possible to call static methods of the implementing class of the Business-Add-In implementation without an instance, you lose the benefit of performance improvement through the Business-Add-Ins and the possibility of repeated use. Also, if you switch the method type in the interface from static to instantiatable at any time in the future, many code adjustments are required. In addition, you can no longer use default code provided.

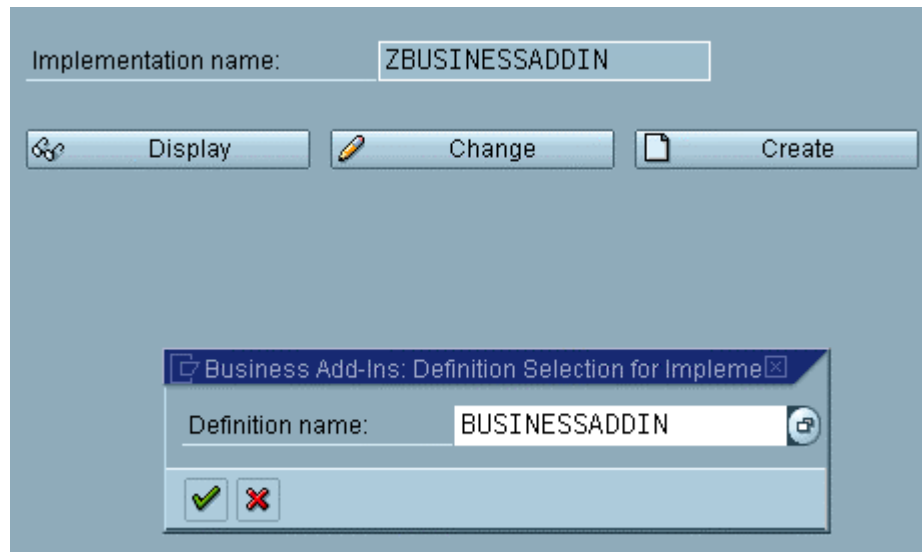
## Implementing Business Add-Ins

A list of the Business Add-Ins present in your system can be found either in the IMG or in the component hierarchy. The enhancements' names and corresponding documentation should help you decide which add-in you want to create an implementation for. During implementation creation, a class for implementing the enhancement's interface is also created. Implementations are discrete transport objects and lie within the namespace of the person or organization implementing them.

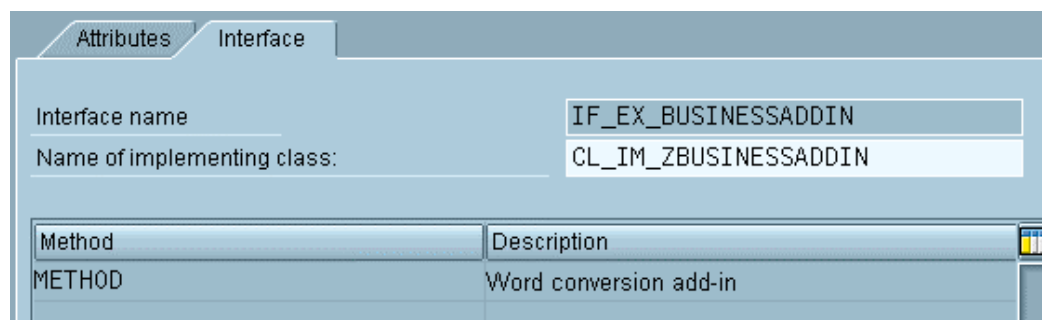
In order to create an implementation for the string conversion example, the add-in (in this case, the interface's method) needs to be filled with logic that converts the string. This logic will be run through every time the add-in is called from the application program.

To create an implementation, proceed as follows:

1. Choose *ABAP Workbench* → *Utilities* → *Business Add-Ins* (transaction SE19) from the SAP menu or double-click on the corresponding activity in the Implementation Guide.
2. Enter a name for the implementation and choose *Create*.
3. Enter the name of the add-in you want to create an implementation for in the dialog box that appears.



4. Enter a short text describing your implementation on the following screen.
5. Choose the *Interface* tab.



**Implementing Business Add-Ins**

6. In order to implement a method for your add-in, double-click on the method to branch to the Class Builder. You must first enter a development class before the Class Builder is displayed.
7. Insert the desired source code for the implementation between the `method`  
`if_ex_businessaddin~method.` and `endmethod.` statements automatically provided to you by the system.



Enter the statement `translate parameter to upper case.` for the string conversion example.

8. Save your entries and return to the *Change Implementation* screen.
9. Choose *Activate*. You may now use this implementation when the application program is executed.



Numerous implementations may exist for a Business Add-In that cannot be used on a multiple basis. However, only one implementation can be active for these kinds of Business Add-Ins at any one time.

What is also important is that you must declare the instance generation of the implementing class (*Attributes* tab) as public and **not** as private or even abstract. If you do this, the system will return short dumps at runtime.



## Filter-Dependent Business Add-Ins

Business Add-Ins may be implemented depending on a specific filter value. If the standard allows for an enhancement for, for example, country-specific versions, it is likely that various partners will want to implement this enhancement. Distinct implementations can then be created and activated according to country.

Enter a filter type when defining your enhancement (a country or industry sector, for example). All methods created in the enhancement's interface have filter value 'FLT\_VAL' as their import parameter. The application program provides the filter value to the enhancement method. The method then selects the active implementation for that value.

A description follows of how a filter-dependent Business Add-In works in the context of the string conversion example. In the following example, different implementations will be called using different filter values.

### Defining a Filter-Dependent Business Add-In

To define a filter-dependent Business Add-In, first create a normal Business Add-In and select the *Filter* checkbox.

Definition name: BADI1  
Definition short text: Sample Business Add-In with Filter Function

Attributes | Interface

General data

Development class: SECE\_TESTOBJECTS  
Language: DE German  
Changed by: BRUEGGEMANN  
Last change: 15.12.1998 10:25:16

Type

☐ Multiple use  
☒ Filter  
Filter type: DE\_LAND

Enter the data element you want as a filter type or select a filter type using the possible entries help.

Filter types are data elements and must fulfill the following criteria:

- The data element's domain may contain a maximum of 30 characters and must be of type *Character*.
- The data element must 1). either have a search help with a search help parameter of the same type as the data element and this parameter must serve as both the import and export parameter or 2). the element's domain must have fixed domain values or a value table containing a column with the same type as the data element.

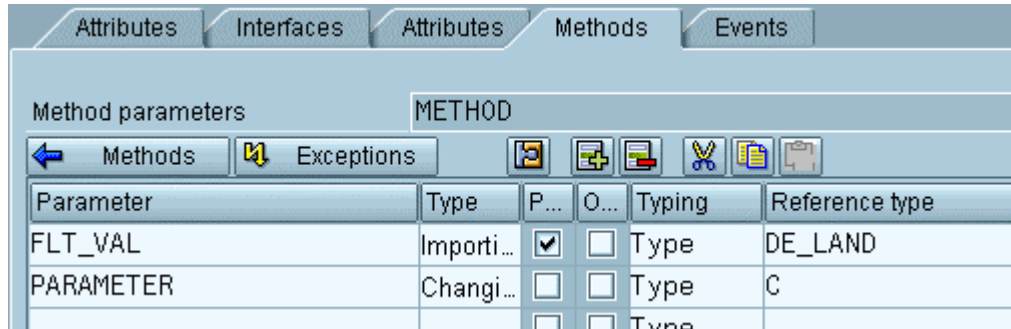
If need be, you can create such data elements yourself.

Now create an interface with a method. Be aware that for each method you create in the interface of a filter-dependent enhancement, the appropriate filter value must be defined as the

## Filter-Dependent Business Add-Ins

import parameter so that the application program can provide the filter value to the enhancement method. The method then selects the active implementation for that value.

The filter value is declared using parameter *flt\_val* and is preset in the list of parameters.



## Calling a Filter-Dependent Business Add-In from an Application Program

As previously discussed, application developers create an instance of the generated class in their application programs and call the corresponding method at the appropriate time. The filter value is passed to the method as export parameter.

```
Report businessaddin.
class cl_exithandler definition load.
data flt type usa_land.
data exit type ref to if_ex_businessaddin.
data word(15) type c value 'Business Add-in'.
```

```
start-of-selection.
  perform formatlist.
  call method cl_exithandler=>get_instance
    changing instance = exit.
  write:/ 'Please click here'.
```

```
at line-selection.
  new-page.
  write:/ 'Original word: ',word.
```

```
    call method exit->method
      exporting
        flt_val = flt.
    Changing
      parameter = word
```

```
  write:/ 'Changed word: ',word.
```

The subroutine *formatlist* looks like this:

```
form formatlist.
write:/ 'USA          -> Conversion to upper case'.
flt = 'USA'.
hide flt.
write :/'Ireland      -> Conversion to lower case'.
flt = 'Ireland'
hide flt.
```

```
write :/'Italy          -> Conversion to...'
flt = 'Italy'.
hide flt.
endform.
```

## Implementing a Filter-Dependent Business Add-In

If you want to use a filter-dependent Business Add-In, you will need an implementation for each relevant filter value. Multiple filter values may use the same implementation, however.

When implementing a filter-dependent Business Add-In, proceed as follows:

1. Create an implementation by referring to the corresponding Business Add-In definition.
2. Enter a characteristic filter value for the implementation, or choose F4 and select a value from the list of possible entries displayed. In principle, it is possible to define multiple characteristic filter values for each implementation.
3. Use the Class Editor to fill the interface method.

In the string conversion example, you would make the following entries for each country:

```
BRD:
translate parameter to upper case.

Ireland:
translate parameter to lower case.

Italy:
translate ...
```

4. Repeat steps 1-3 for each implementation that you create.
5. Activate your implementations.

Now, whenever you execute the report program described above, different country-specific implementations are executed.

## Extendible Filter Types

If you flag a filter type as being *extendible*, it is also possible to create implementations for filter values that did not exist so far. When you assign the *extendible* attribute to a filter type, the *Implementation* menu option in the BAdI Builder is replaced by *Filter value*. If you choose *Filter value* → *Create* and enter a filter value that did not yet exist (you do not need to specify the name of the implementation), the system takes you to the transaction for implementing Business Add-Ins. You can also use transaction SE19, as usual.

Note that the assignment of the *extendible* attribute is subject to the following restrictions:

The domain to which the extendible filter type refers must have the following properties:

- The domain is linked to a cross-client value table. This value table has exactly one key field which has the data element of the filter type as its field type.
- The domain has a text table with two key fields. A key field has the filter type as its field type, and a key field is a language field. To mark a field as a text field, a field must exist in this table that contains the string 'TEXT' or 'TXT' as a partial string. In the Dictionary, the text table must be assigned to the value table.
- The delivery class of both tables must be "E" or "S".

## Filter-Dependent Business Add-Ins

All filter values that are created in the context of an extendible filter-dependent Business Add-In must not yet occur in the value table and are added to the value table when the data is saved. Analogously, the values are removed from the value table when the implementation or the entire Business Add-In is deleted. The same applies to the text table.

## Special Characteristics of Country-Specific Business Add-Ins

The standard system contains a Business Add-In which provides an interface for integrating additional postal checks of the address data through external tools. This Business Add-In is called *Address-Check*.

This example uses the data element INTCA (ISO code of the country). Since SAP recommends complying with the international ISO standard for country-specific queries, you should use the data element INTCA also for Business Add-Ins. The ISO code of a country key would be, for example, US instead of USA, or DE instead of BRD.

Before the Business Add-In is called from within the application program, you must determine the ISO code of the country by submitting a SELECT statement against table T005 (field INTCA). Then you can pass the filter value as exporting parameter.

```
*-----External interface through Business Add-In ADDRESS_CHECK
* Instantiate object
  if g_obj_ex_address_check is initial.
    call method cl_exithandler=>get_instance
      changing instance = g_obj_ex_address_check.
  endif.
* Call Business Add-In
  data: ch_error_table type addr_error_tab.
  data: l_message_type_single like addr_error_msg_type.
  data: l_message_type_all like addr_error_msg_type.
  call method g_obj_ex_address_check->address_postal_check
    exporting im_dialog_allowed = x_dialog_allowed
              flt_val          = x_t005-intca
    changing  ch_adrc_struct   = x_adrc_struct
              ch_t_error_table = ch_error_table.

* Error handling
  l_message_type_all = '0'.
  loop at ch_error_table into error_table.
```

## Multiple Use Business Add-Ins

You can differentiate between single and multiple use Business Add-Ins. Single use add-ins are based on procedures, whereas multiple use add-ins have characteristics similar to those of events. In the first case, the program waits for the enhancement to return something, usually a return code. Benefit calculation in HR is a good example of this type of enhancement. Here, different calculations can be performed according to whichever implementation is active. With multiple use add-ins, an event is processed in program flow that may be of interest for other components. These components can then use this event as a hook to hang their own additional actions on.



Since the call sequence cannot be predicted, it does not make sense to allow return values for this second type.



In addition to importing parameters, you can also use changing parameters for multiple use Business Add-Ins.

### Example:

You want your application to continue processing indexes with a different component after you have saved (in other words, the system should allow you to use an add-in after saving). Since this is a good callup point for numerous different functions, you want to create an enhancement at this juncture that can be used by multiple subscribers.

To create a multiple use Business Add-In, proceed as follows:

1. Define an add-in and select the *Multiple use* checkbox from the *Administration* tab.
2. Define an interface with method 'OBJECT\_SAVED' and the import parameter 'OBJECTNAME'.

Calling your enhancement in the application program

**program event.**

```
...
data exit_obj type ref to if_ex_event.
...
call method cl_exithandler =>get_instance
      changing instance = exit.
```

```
form save_object using obj_name type c.
```

```
...
update ...
call method exit_obj->object_saved
      exporting objectname = obj_name.
endform.
```

The number of subscribers that subsequently call the event and hang their own additional actions on it is of no importance to the application program calling the add-in. Active implementations are called in the adapter method.

---

**Multiple Use Business Add-Ins**

In addition to importing parameters, you can also use changing parameters for multiple use Business Add-Ins.

## Menu Enhancements

SAP allows you to enhance menus in its user interfaces using function codes. These function codes must adhere to the form /namespace/+, just like in SMOD/CMOD enhancements. They are assigned to a specific enhancement and only appear in their corresponding menus once an implementation of this enhancement has been activated.

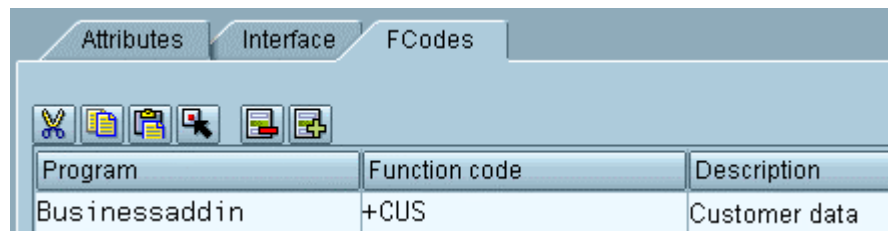
Application developers reserve specific function codes for customers when defining a Business Add-In. They use the Menu Painter to include these codes in the appropriate menu lists. Application developers must also ensure that these menu options are called in their applications and that the corresponding add-in methods are also retrieved. Customers can take advantage of menu enhancements by creating a new implementation, choosing a text for the menu option, and then programming the method used to determine what action is performed when the menu enhancement is called.



Menu enhancement is only possible using single use add-ins (not multiple use add-ins) that are not filter-dependent. Currently, menu enhancements can only be created in conjunction with program enhancements (interfaces).

To create a menu enhancement, proceed as follows:

1. Create an add-in and define its interface.
2. Choose *Fcodes* from the tabstrip.
3. Enter the name of your program, the function code, and a description.



4. Call the Menu Painter or double-click on your program name or function code to branch to user interface maintenance in the Menu Painter. Enter your function code in the appropriate menu list. If you have accessed the Menu Painter directly during add-in definition, you can call your menu lists by choosing *Goto* → *Object lists* → *Menu list* instead.

## Calling a Menu Enhancement from an Application Program

Your programming should look like this:

```
(...)
case fcode.
  when 'SAP'.
    (...)
  when '+CUS'
    call method ...
```

## Implementing a Menu Enhancement

When implementing menu enhancements, proceed as follows:

**Menu Enhancements**

1. Create an implementation and choose *Fcodes*. All data adopted from your Business Add-In's definition is displayed here. You can make entries for the implementation on the right. You can also double-click on the first input field. The following dialog box appears:

Program	Function code	Description	Function text
BUSINESSADDIN	+CUS		

Business Add-Ins: Text Maintenance for Function Code +CUS

Program: BUSINESSADDIN

Function code: +CUS

Text: Display customer data

Icon name: ICON\_DISPLAY

Icon text:

Info. text:

Texts

OK Cancel

Here you may enter a text for your function code, the name of an icon and a text for the icon, and a short informational text.

The actions that you want the system to perform after the pushbutton is chosen must be programmed in the appropriate method, either manually or using default source code that has been provided to you.

Menu enhancements only become visible after the implementation has been activated and the application program that calls the Business Add-In has been executed.














## Business Add-Ins: Import Procedure

Conflicts can occur at release upgrade or when transporting a Business Add-In within a system infrastructure containing multiple levels (country versions, industry solutions, partners, etc.).

Possible collisions include:

Case 1	Multiple active implementations exist for Business Add-Ins that have not been designed for multiple use (for the same filter value, if add-in is filter-dependent) .
Case 2	Identical interfaces were assigned during Business Add-In definition.
Case 3	Identical function codes were assigned during Business Add-In definition.

Whenever such collisions occur, corresponding error messages and warnings are created in the transport log at import. Their long texts provide you with information on how to proceed.

	Processing subsequent to import for Change/transport request: B20K8A
	Subsequent processing AFTER_IMP_SXCI for SXCI L started at 00:08:59
	There are multiple active implementations for singular definition EX
	Errors occurred during post-handling AFTER_IMP_SXCI for SXCI L
	The errors affect the following R/3 components:
	BC-DWB-CEX (Customer Enhancements)
	Subsequent processing AFTER_IMP_SXSD for SXSD L started at 00:09:06
	Processing subsequent to import of change/transport request B20K8A07
	Start of subsequent processing ... 19990319000858
	End of subsequent processing... 19990319000907
	Execute reports for change/transport request: B20K8A07TK
	There are no reports to execute
	Execution of programs after import (XPRA)
	End date and time : 19990319000908
	Ended with return code: ===> 8 <===

If the collision described in case 1 occurs in your system, proceed as follows:

1. Choose *Utilities* → *Adjustment* → *Multiple active implementations* (in transaction SE18).  
A list appears displaying your Business Add-Ins. A red traffic light indicates that multiple active implementations exist for that add-in.
2. Deactivate these implementations using the appropriate pushbutton or double-click on the name of the implementation to branch to the corresponding transaction where you can then correct the problem

In cases 2 and 3, proceed as follows:

1. Choose *Utilities* → *Adjustment* → *Multiple assigned interfaces* or *Multiple function codes assigned*.  
Both of these menu options display an overview; the add-ins in question are marked with a red traffic light.
2. The people responsible for those add-in definitions where identical interfaces and function codes occur must now decide how to proceed. To delete a function code or change an

---

**Business Add-Ins: Import Procedure**

interface name, call the ABAP Workbench and use the appropriate pushbutton to call the tool you need. A new transport may be necessary.

## Customer Exits

This section explains how you can use customer exits to add your own functionality to standard R/3 applications.

[Enhancements to the SAP Standard with Customer Exits \[Page 100\]](#)

[Types of Exits \[Page 101\]](#)

[Locating Applications that Have Exits \[Page 102\]](#)

[Creating an Add-on Project \[Page 104\]](#)

[Activating and Deactivating a Project \[Page 106\]](#)

[Transporting Add-on Projects \[Page 107\]](#)

[Creating Customer-specific Menus \[Page 108\]](#)

[Creating Customer-specific Subscreens \[Page 109\]](#)

[Creating Customer-specific Function Modules \[Page 111\]](#)



More information on this subject can be found in the online help of the Project management screen for SAP enhancements.

## Enhancements to the SAP Standard with Customer-Exits

The R/3 enhancement concept allows you to add your own functionality to SAP's standard business applications without having to modify the original applications. SAP creates customer exits for specific programs, screens, and menus within standard R/3 applications. These exits do not contain any functionality. Instead, the customer exits act as hooks. You can hang your own add-on functionality onto these hooks.

If you want to enhance the functionality of your SAP System, you should take advantage of the exits available in standard R/3 applications. There are two main reasons why you should use exits rather than modifying SAP software yourself. Add-ons attached to exits have the advantage that:

- They do not affect standard SAP source code

When you add new functionality to your SAP System using SAP's exits, you do not alter the source code of standard SAP programs in any way. The code and screens you create are encapsulated as separate objects. These customer objects are linked to standard applications, but exist separately from SAP's standard software package.

- They do not affect software updates

When you add new functionality to your SAP System using SAP's exits, your objects (called customer objects) must adhere to strict naming conventions. When it comes time to upgrade a to a new software release, customer objects' names ensure that they will not be affected by any changes or new additions to the standard software package.

Customer exits are not available for all programs and screens found in the SAP System. You can only use customer exits if they already exist in the SAP System. You find more information about locating applications with pre-defined exits in [Locating Applications that Have Exits \[Page 102\]](#).

## Types of Exits

There are several different types of customer exits. Each of these exits acts as hooks where you can attach or “hang” your own add-ons.

- **Menu Exits**

Menu exits add items to the pulldown menus in standard SAP applications. You can use these menu items to call up your own screens or to trigger entire add-on applications.

SAP creates menu exits by defining special menu items in the Menu Painter. These special entries have function codes that begin with “+” (a plus sign). You specify the menu item’s text when activating the item in an add-on project.

- **Screen Exits**

Screen exits add fields to screens in R/3 applications. SAP creates screen exits by placing special subscreen areas on a standard R/3 screen and calling a customer subscreen from the standard screen’s flow logic.

- **Function Module Exits**

Function module exits add functions to R/3 applications. Function module exits play a role in both menu and screen exits. When you add a new menu item to a standard pulldown menu, you use a function module exit to define the actions that should take place once your menu is activated. Function module exits also control the data flow between standard programs and screen exit fields.

SAP application developers create function module exits by writing calls to customer functions into the source code of standard R/3 programs. These calls have the following syntax: CALL CUSTOMER-FUNCTION ‘001’.



From Release 4.5A, you can change field texts on screens and add data element documentation using the Modification Assistant.

## Locating Applications that Have Exits

### Locating Applications that Have Exits

You can only take advantage of exits if SAP has added them to one of its standard applications. For this reason, you need to know how to locate the exits available in the SAP System. SAP organizes its exits in packages are called SAP enhancements. Each SAP enhancement can contain many individual exits.

Choose *Utilities* → *Enhancements* → *Project management* from the *ABAP Workbench* menu. From here, choose *Utilities* → *SAP enhancements* to call a selection screen that lets you look for the exits available in standard applications. If you do not know the name of a specific enhancement package, you can search for enhancements by development class. To list all enhancements in the entire system, choose the *Execute* function without specifying any selection criteria.

The system displays a list of all enhancement packages. From within this list, you can display the documentation for each enhancement. You can also display a list of each enhancement's different exit types. One enhancement package, for example, might contain several menu exits and function module exits, and a single screen exit.

### Searching in the Application Hierarchy

You can also use the Application Hierarchy to search for exits that are available for a particular application area. You can access the application hierarchy by choosing *ABAP Workbench* → *Overview* → *Application hierarchy*

The SAP Application Hierarchy lists all of the standard SAP applications and their component parts. You can use the hierarchy with the Repository Information System to locate exits available for specific applications or programs. To locate the exits associated with Materials Management, for example, proceed as follows:

1. Place the cursor on the *Materials Management* branch of the Application Hierarchy tree and choose *Sel./desel. sub-tree*.

The system marks the application area.

2. Choose *Repository Infosys*.

The system lets you search for any objects assigned to a particular application.

3. Expand the *Environment* sub-tree in the Repository Information System.
4. Expand the *Exit Techniques* sub-tree. Choose either *Enhancements* or *Projects* for specific customer exit by double-clicking.

The system offers a selection screen that allows you to narrow down your search. You can leave the fields on this screen empty.

5. Choose *Execute*.

The system will retrieve all the enhancements assigned the category you marked in the Application Hierarchy.

To display the individual exits that each enhancement package contains, select the enhancement and choose the *Display* icon. You can also double-click on any enhancement in the list to go to the transaction that SAP uses to manage its enhancements.



---

Creating an Add-On Project

## Creating an Add-On Project

To take advantage of the exits available in standard R/3 applications, you need to create an add-on project. This project lets you organize the enhancement packages and exits you want to use. The add-on project also allows you to hang add-on functionality onto the exit hooks contained with SAP enhancements.

## Managing an Add-On Project

An add-on project contains a series of exits, as well as the add-ons that you develop to attach to these exits (like menu entries or function modules). To create an add-on project from within the *ABAP Workbench* menu, choose *Utilities* → *Enhancements* → *Project management*.

Before you begin defining a project, you need to decide which application, application component, or specific standard transaction you would like to add your own functionality to. Give your project a name that indicates the type of functions it contains and that shows which transactions are affected. It may be useful to agree upon a company-wide naming convention for enhancement projects.

There are two factors you need to keep in mind when you create an add-on project. First, you can include an SAP enhancement package and the customer exits it contains in one project only. The same SAP enhancement may not appear in two separate customer projects.

Second, in order to make your add-ons, such as menu items or screen fields, appear in standard R/3 applications, you must activate your add-on project. When you activate a project, all of the add-ons created within this project are activated as well. For this reason, you should ensure that the exits you include in your project contain functions that can be activated simultaneously. It is not possible to activate exits individually.

After you have specified the name of your new project, proceed as follows:

1. Choose *Create*.
2. Describe the nature of the project by providing a short descriptive text.
3. Choose *Save*.

The system then prompts you to assign a change request. This assignment allows you to transport the project and its components into a productive system once you have completed it.

4. Specify which SAP enhancement packages you want to include in your project by choosing *SAP enhancements*.
5. Enter the names of these SAP enhancements in the spaces provided.

You should choose enhancement packages that logically belong together. For example, if you are planning to use several enhancements that deal with Materials Management modules, you can include all of these enhancements in the same project. If you are working on enhancements that deal with different applications, or if the enhancements are not logically related, you should include these enhancements in separate projects.

Once you have identified the SAP enhancements you want to include in your project, you can begin adding your own functions to the exits offered in the enhancements. To display the individual components of the SAP enhancements, return to the main screen of the *Project Management* transaction and choose *Enhancement components*. The system displays all of the exits included in the enhancements assigned to your project.





---

**Activating and Deactivating a Project**

## Activating and Deactivating a Project

After you have attached add-on functionality to the exits in your project, you need to activate the project. Activating a project turns on all your add-ons. You also need to activate your project after you have transported it from a development system into a productive system.

When you activate your project, the system turns on all add-ons that the project contains. You cannot activate individual add-ons separately. To activate an add-on project, proceed as follows:

1. Call the *Project management* transaction.
2. Choose *Activate project*.

The system displays a message confirming that the project was activated.

You can now call up any standard SAP transaction that contains an exit you used in your add-on project. The newly added add-on function(s) should appear.

If you need to make changes to any of your add-ons, you must first deactivate the project that contains that add-on. Deactivating a project turns off all of that project's add-ons. To deactivate an add-on project, proceed as follows:

1. Call the *Project management* transaction.
2. Choose *Deactivate*.

The system confirms that the project was deactivated.

Once the project is turned off, you can make changes to the project's add-ons or build new functions and attach them to other exits in the project.

## Transporting Add-on Projects

To take advantage of exits in the SAP System, you must create an add-on project. This project contains at least one SAP enhancement package and all of the user exits assigned to this package. Your add-on project also encompasses all add-on functions you create and attach to the exit “hooks”.

When you first create an add-on project, you must assign the project to a change task. You also need to assign all of your add-on components (include programs, subscreens, and menu texts) a change task number. If your project is spread out over more than one change task, you should assign these tasks to the same change request.

Once you have completed your project, release your change tasks. As a final step, release the change request that contains all the change tasks for your project. The system will export your add-on project and coordinate its import into either a consolidation system or productive system. Check to make sure that your add-on project is active in this system.

---

Creating Customer-Specific Menus

## Creating Customer-Specific Menus

Menu exits allow you to add your own functions to the pulldown menus in standard R/3 transactions. To take advantage of a menu exit, you first need to create a project as described in the section [Creating an add-on project \[Page 104\]](#). Then, you must include the SAP enhancement package that contains the menu exit you want to use in your project. From the main screen of the *Project management* transaction, proceed as follows:

1. Select *Enhancement components* and choose *Change*.

The system lists all customer exits contained in the enhancements included in your project.

2. Place the cursor on the menu exit you want to add your own function to.
3. Choose *Edit component*.

The system displays technical information about the menu entry. It also displays two input fields where you can specify a language and the text you want your menu item to have.

4. Enter the name of your menu function (such as *Special order method*) in the field *New text*.
5. Choose *Save text*.

Your special menu function will not appear in the pulldown menu of the corresponding standard transaction until you activate your enhancement project.

Specifying a menu text and activating your project are not the only steps you need to take to make your own menu item work. After you carry out these steps, your menu item will appear, but it will not call a corresponding function. To attach your menu entry to its corresponding function, you need to create a function module for the function module exit associated with your menu text. This function module exit is part of the same SAP enhancement that contains the menu exit you used. [Creating Customer-Specific Function Modules \[Page 111\]](#) tells you more about how to use function module exits.

## Creating Customer-Specific Subscreens

Screen exits allow you to add your own fields to specified screens in standard R/3 transactions. To take advantage of customer-specific subscreens, you first need to create projects as described in the section [Creating an add-on project \[Page 104\]](#). Then, you must include the SAP enhancement that contains the screen exit you want to use in your project. From the main screen of the *Project management* transaction, proceed as follows:

1. Select *Enhancement components* and choose *Change*.

The system lists all customer exits contained in the enhancements included in your project.

2. Place the cursor on the screen exit you want to create your own subscreen for.

3. Choose *Edit component*.

The system prompts you to enter a development class for your subscreen. Use the development class that contains all of the objects created for this specific enhancement project. The system then transfers you to the Screen Painter.

4. Create your subscreen using the Screen Painter.

5. Add modules to the screen flow logic as needed.

Enter the name of the module in the flow logic editor, then double-click on the module name and the system automatically creates the module in the corresponding function module program.

6. Generate your screen and choose *Back* (the green arrow) to return to the *Project management* transaction.

Once you activate your enhancement project, the fields defined in your subscreen will appear in the standard R/3 transaction.

## Using Predefined Screen Data

When you use a screen exit, you need to be aware of how your subscreen and the fields it contains relate to the data on the standard R/3 screen. The global data defined in the standard SAP program is not known to the program that controls your customer subscreen. In the same way, the fields you use in your subscreen are not known to the SAP program in which they appear. To compensate for this problem, SAP creates special flow logic modules when defining screen exits.

SAP creates a special Process Before Output module (PBO module) with a function module exit. You can use this function module exit to transfer specific field values from the SAP program to your customer subscreen. SAP also includes a Process After Input (PAI) module that has a function module exit for transferring values back from your subscreen to the SAP program.

The function module exits in these PBO and PAI modules are only activated if you create actual modules for each exit. For more information on how to create customer-specific function modules, refer to the section [Creating customer-specific function modules \[Page 111\]](#).



The import and export parameters for the PBO and PAI function module exits are predefined by SAP. As a result, you can only transfer values to and from your

---

**Creating Customer-Specific Subscreens**

subscreen if they correspond to the parameters determined in the existing function module exit interface.

## Creating Customer-Specific Function Modules

Function module exits allow you to implement your own functions and processing logic in standard SAP programs. To take advantage of a function module exit, you first need to create a project as described in the section [Creating an Add-on Project \[Page 104\]](#). Then, you must include the SAP enhancement that contains the function module exit you want to use in your project. From the main screen of the *Project management* transaction, proceed as follows:

1. Select *Enhancement components* and choose *Change*.

The system lists all the customer exits contained within the enhancements included in your project.

2. Place the cursor on the function module exit you want to use.
3. Choose *Edit component*.

The system displays the exit's corresponding function module, which has been prepared by an SAP application developer. You can fill this module with functions you want by calling the include program (called ZXnnnU01) contained within the module and inserting your own source code into it. Both the function module and the include program lie within the namespace reserved for customers.

4. Call the function module's include program by double-clicking on it.

The system asks you if you want to create the include program.

5. Confirm that you want to create the include program.
6. Enter your function's source code in the editor.
7. Save your include program.

## Using Predefined Interface Parameters

When you use a function module exit, you need to be aware of how your function module fits into the program logic of the standard R/3 program that calls it. When defining function module calls in its standard program, SAP specifies exactly what data can be transferred to and from the standard program. If the customer function call includes import and export parameters, the customer function module will itself have the same predefined interface.

You can not make changes to the general framework or to the interface of a function module exit's function module. You are only allowed to use the special include program it contains to add your own programming logic. When you create the source code for your include program, be aware of the restrictions set by the function module exit's interface. You can only transfer values to and from your include program if these values are included in the function module's predefined interface.

## Using Sample Code Provided by SAP

SAP sometimes supplies sample coding for the function module exits it provides. You can copy this sample code into your include program. To use sample code, proceed as follows:

1. Create and save a customer include program as described in steps 1 - 6 of the section on [Creating Customer-specific Function Modules \[Page 111\]](#).
2. Return to the *Project management* transaction.

## Creating Customer-Specific Function Modules

3. Display the exits in your project and place the cursor on the function modules exit you created an include program for.
4. Choose *Edit* → *Use SAP source code*.

If sample code exits, the system copies this code to your include program. If you have already written your own code in the include program, the system appends the sample to your code. You are free to alter the sample code as needed.

## Declaring Global Data

SAP often supplies function module exits with predefined global data. Global data is required if, for example, SAP provides subroutines with a function module exit. All global data defined by SAP is stored in an include called LXnnnTOP, where **nnn** stands for the other characters in the function group. You can define your own global data in any function module exit.

Take advantage of the ABAP Workbench's navigational features when creating global data definitions:

1. Enter the name of the field or internal table you want to define in the editor.
2. Double-click on the field or table name.

The system asks you if you want to define the object, since no data definition exists for it yet.

3. Confirm that you want to create a new data definition.

The system automatically creates a new include called ZXnnnTOP. Use this include to store all of the data definitions for your function module exit(s).

## Creating Subroutines

SAP may include pre-programmed subroutines in certain function module exits. These predefined subroutines are stored in an include called LXnnnF00, where **nnn** stands for the other characters in the function group. You may call these routines from within your customer include program.

You can also create your own subroutines. To create new subroutines, use forward navigation in the ABAP Workbench and proceed as follows: Proceed as follows to create new subroutines:

1. Open the editor and write the keyword **PERFORM**, followed by the name of your subroutine.
2. Double-click on the subroutine's name.

Since the routine does not yet exist, the system asks if you wish to create it.

3. Confirm that you want to create a new subroutine.

The system automatically creates an include called ZXnnnZZZ, where **nnn** stands for the other characters in the function group. This include should contain all subroutines used in your customer function modules.



The central include program (called ZXnnnU01) contains the commands **FUNCTION** and **ENDFUNCTION**. Therefore, you should not use the keywords **FORM** and **ENDFORM** in this include.



## Calling Your Own Screens

You can call your own screens from within a customer function module. For example, you might want to call a screen that lets users carry out certain processes not available in a standard R/3 transaction. Use forward navigation in the ABAP Workbench to create your own screens and their flow logic. Proceed as follows:

1. Write the command `CALL SCREEN xxx` in the editor, where xxx stands for the screen number.
2. Double-click on the screen number.

The system asks if you want to create the screen and then branches to the Screen Painter.

To create PBO and PAI modules, proceed as follows:

1. Enter the modules' names in the Flow Logic Editor.
2. Double-click on each name.

The system creates the modules. At the same time, the system creates include programs that adhere to SAP's naming standards.

Use Workbench navigation to create any other program objects (like events) that you want to include in your function module.

---

Creating Customer-specific Documentation

## Creating Customer-specific Documentation

SAP's enhancement concept allows you to create your own keywords and online help documentation for all data elements stored in the R/3 Repository. These documentation exits differ from the other exits described in this document in several ways:

- SAP application developers do not need to prepare special exits for data elements. You can customize the texts of all standard data elements in your SAP System.
- You do not need to create an enhancement project to create your own documentation for data elements.
- Your own documentation texts are not affected when you activate and deactivate enhancement projects.

## Creating Your Own Keywords and Short Texts

You can use the enhancement concept to create your own keywords for data elements stored in the ABAP Dictionary. The system uses keywords on screens to identify entry fields. Data elements can have up to three keywords of varying lengths.

The data element BBBNR, for example, identifies the first part of a customer's international company number. The standard keywords for this data element in the ABAP Dictionary are:

- Company no. 1
- Comp. no.1
- Company number 1

You might want to change these keywords to reflect your company's own terminology. Instead of 'Company number 1,' you might want to identify the field as 'Corporate ID.' If you change the keywords using the enhancement concept, your new texts will appear in all SAP screens where any field directly refers to the data element you altered. If SAP deliberately modifies a keyword on a standard screen, direct reference to the original data element is dissolved. Any changes you make to a keyword with the enhancement concept will not appear in standard screens that do not retain a direct reference to the modified data element.

You can identify such screen-specific modifications by looking in the Screen Painter. Display the field's screen attributes and examine the field *Modific*. If an 'x' appears, the SAP developer modified the data element's key word. Your own keyword will not appear.

The enhancement concept also allows you to alter a data element's short description (short text). This brief text appears whenever a user calls up online help for that field with F1. To create your own keyword and short text documentation, proceed as follows:

1. Enter the *Project management* transaction by choosing *Utilities* → *Enhancements* → *Project management* from the *ABAP Workbench* menu
2. Choose *Text enhancements* → *Key words* → *Change*.

The system displays a dialog box.

3. Enter the name of data element.

If you want to see a list of all the screens where this data element appears, choose *Ref. to screen* before proceeding to the next step. The reference list shows you how many standard R/3 screens your new documentation will affect.

## Creating Customer-specific Documentation

4. Choose *Change* from the application toolbar.
5. Edit the data element's keywords and short text.
6. Save your changes.

If you want to list all the data elements in your system that you have created customer-specific key words for, return to the main screen of the *Project management* transaction and choose *Text enhancements* → *Key words* → *Display*. The system displays all altered data elements and shows which changes were made.

## Restoring SAP and Customer Versions

At some point you may want to restore all standard SAP keywords and short texts for data elements that you may have altered. You can restore the standard texts from the main screen of the *Project management* transaction. Choose *Text enhancements* → *Key words* → *Restore SAP version*. Specify the SAP release from which the texts are to be restored.

The restoration function is carried out by a special program that runs in the background. This program goes through all data elements in your SAP System and restores the standard texts where required.

## Creating Your Own Online Documentation

The enhancement concept allows you to create your own online documentation for all data elements in the ABAP Dictionary. The system displays this documentation whenever a user chooses F1 to get more information about a screen field.

The standard documentation for the R/3 data element BBBNR reads: 'Here, you enter the first 7 digits of the international location number'. You can enhance this documentation with additional information that your users may need. To write your own field documentation, proceed as follows:

1. Call the *Project management* transaction by choosing *Utilities* → *Enhancements* → *Project management* from the *ABAP Workbench* menu.
2. Choose *Text enhancements* → *Data elements* → *New DE cust. docu*.  
The system displays a dialog box.
3. Specify the data element's name.  
If you want to see a list of all the screens where this data element appears, choose *Ref. to screen* before proceeding to the next step. The reference list shows you how many standard R/3 screens your new documentation will affect.
4. Choose *Change* from the application toolbar.
5. On the next screen, ensure that the checkbox labelled *Make available SAP documentation in the form of an INCL* is activated.
6. Choose *Create*.
7. Write your own documentation.
8. Save your changes.

## The Modification Assistant

### The Modification Assistant

The SAP system is modified to meet the needs of individual working environments at several different levels:

- Customizing
- Personalization
- Customer exits
- Modifications to the standard
- Customer development

Customizing, Personalization, customer exits, and customer developments are all supported by corresponding tools such as the IMG, Business Add-Ins, transaction CMOD, or the development environment. The Modification Assistant offers you support in the other area listed above, modification to the standard.

In order to use the Modification Assistant to simplify the upgrade process, you branch to a special modification mode whenever you are modifying objects from the standard in an ABAP Workbench editor. Originals are initially protected in this mode and can only be changed with the help of the additional pushbuttons that are placed at your disposal.

All changes that you make to the system are logged with the help of the Modification Assistant. This provides you with a detailed overview of modifications that is easy to read and that dramatically reduces the amount of effort needed to upgrade your system.

The Modification Assistant offers support in the following areas:

ABAP Editor, Screen Painter, Menu Painter, text element maintenance, Function Builder, ABAP Dictionary (table, collective search help, and structure enhancements using appends, data element attribute modification ) and documentation (modification of documents from specific classes).

Global interfaces, classes, and their components cannot be changed using the Modification Assistant.

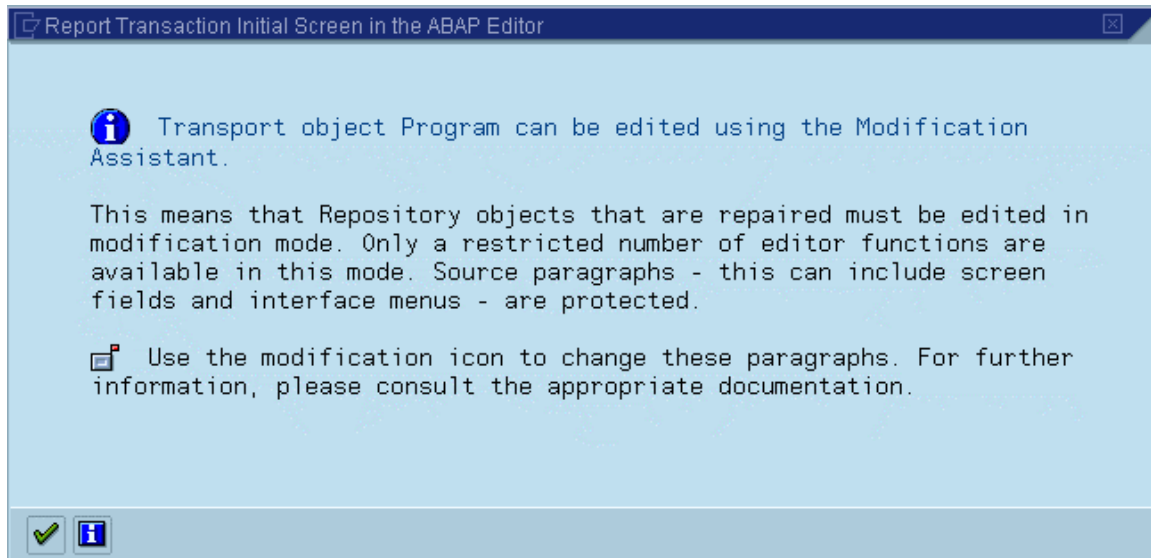
Modifications made to those ABAP Workbench objects not supported by the Modification Assistant are logged and displayed in an overview. Modifications made to these objects are adjusted in the system in the conventional manner at upgrade.



The [Inactive Sources \[Ext.\]](#) concept is not currently used with the Modification Assistant.

If an object can be edited using the Modification Assistant, a dialog box appears the first time that you attempt to edit that object informing you that editing functions are limited in modification mode. This dialog box appears exactly once per user for each of the various different kinds of transport objects.

## The Modification Assistant



Modifications should only be made to the standard when they are necessary for the optimization of specific business processes. Before modifying the standard, it is essential that you are well acquainted with an application's construction and flow logic. Such knowledge is necessary if you want to evaluate modification possibilities and decide on a sensible modification design.

The responsibility for a modification's semantic correctness ultimately lies with the person making the modifications.

For more information, see:

[Modifications in Programs \[Page 118\]](#)

[Modifications in the Screen Painter \[Page 120\]](#)

[Modifications in the Menu Painter \[Page 123\]](#)

[Modifying Text Elements \[Page 127\]](#)

[Modifying and Adding Function Modules \[Page 128\]](#)

[Modifications in the ABAP Dictionary \[Page 130\]](#)

[Modifying Documentation \[Page 133\]](#)

[Disabling the Modification Assistant \[Page 135\]](#)

[Resetting to the Original \[Page 136\]](#)

[The Modification Browser \[Page 137\]](#)

[Upgrade Procedure/R/3 Support Packages \[Page 140\]](#)

## Modifications in Programs

## Modifications in Programs

You can make modifications to programs from the ABAP Editor using the *Insert*, *Replace*, and *Delete* pushbuttons. To facilitate such changes, programs are organized into modularization units called modules, subroutines, and events. Those lines of program code not contained within these units, such as comment inserted after a form routine or the code found at the beginning of a program, are also classified as modularization units (unnamed or external modularization units). Each modification you make is assigned to its corresponding unit and logged for both the modification overview and upgrade.

```

10  MODULE USER_COMMAND_0100 INPUT.
11      CASE OK_CODE.
12          WHEN 'SHOW'.
13              CLEAR OK_CODE.
14              SELECT SINGLE * FROM SPFLI WHERE CARRID = SPFLI-CARRID
15  *{  REPLACE          MODK920030                      1
16  *\                                AND  CONNID = SPFLI-CONNID.
17                                  AND  CONNID  = SPFLI-CONNID
18                                  AND  CITYFROM = SPFLI-CITYFROM
19                                  AND  CITYTO  = SPFLI-CITYTO.
20  *}  REPLACE
21  *{  INSERT          MODK920030                      2
22      WHEN 'CHNG'.
23      PERFORM CHANGE_SPFLI.
24  *}  INSERT
25      WHEN SPACE.
26      WHEN OTHERS.
27          CLEAR OK_CODE.
28          SET SCREEN 0. LEAVE SCREEN.
29      ENDCASE.
30  ENDMODULE.                                " USER_COMMAND_0100  INPUT

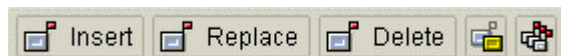
```



In the ABAP Editor, changes based on the Modification Assistant are currently made in the table control mode. The edit control mode is switched off.

If no modularization units that SAP delivered are altered, then your modifications can be automatically adopted at upgrade.

In addition to *Insert*, *Replace*, and *Delete*, the functions *Modification overview* and *Undo modification* are also available for modifying programs in the Editor.



*Insert*

## Modifications in Programs

To insert ABAP code into a program, place your cursor on the appropriate line in the Editor and choose *Insert*. A new line is inserted at the point where your cursor was positioned. This line is ready for input. Start your modification here. Whenever you choose *Enter* at the end of a line, the system inserts an additional line that is ready for input. All standard ABAP Editor editing functions are available for use in these lines.



In command mode, you place your cursor on the line after which you want to insert additional code. If you enter a command such as **I(nsert)**, more ready for input lines are added.

Any code you insert is framed (set off from the other code) by comment lines describing what kind of modification is being made (Insert, Replace, or Delete). The repair number assigned to your modification is also displayed.



Modifications are numbered (at far right) so as to aid their internal administration.

### *Replace*

To replace a line in the Editor, place your cursor on the appropriate line and choose *Replace*. The line you are replacing is changed to a comment line and then copied into a line that is ready for input where you can then alter it any way you want to. Additional ready for input lines can also be inserted here, allowing you to make as many new entries as necessary.

If you want to replace multiple lines all at once, select these lines before using *Replace*.



If you want to add a subroutine, use the *Insert* pushbutton to add the FORM/ENDFORM statements, for example. Do not replace the ENDFORM statement with another FORM/ENDFORM statement.

### *Delete*

You can delete whole lines by either placing your cursor on a single line or by selecting an entire passage and then choosing *Delete*.

### *Modification overview*

All modifications from the corresponding include are displayed on an overview screen.

### *Undo modification*

In order to undo a modification you have made, place your cursor on a line within that modification and choose *Undo modification*.



Whenever you add a new modularization unit to an SAP program, you should assign the unit to a customer include. You should also move all sections of code that are inserted in the program to the subroutines of a customer include. This reduces upgrade time and is helpful when trying to fix any conflicts that arise. For further information, please refer to the section on [Adjusting program parts \[Page 157\]](#).

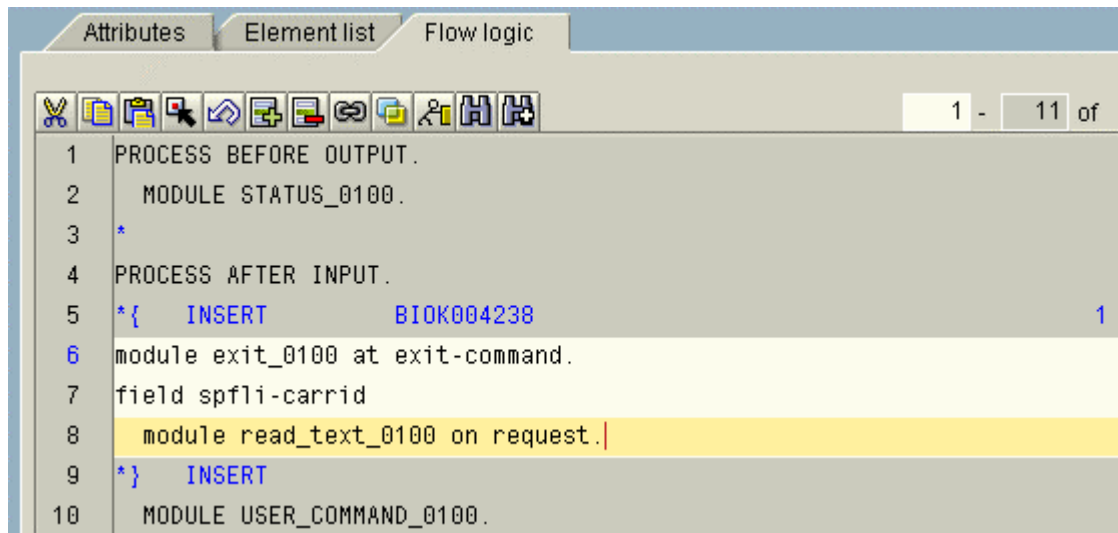
## Modifications in the Screen Painter

## Modifications in the Screen Painter

In the Screen Painter, you can make modifications to screen attributes, the flow control of a screen, its layout or in the element list.

Screen attributes can be modified in any way that you like, including, for example, changing the screen size maintenance value.

In a screen's flow logic, you can modify existing events or add new ones. Just as in the ABAP Editor, flow logic statements can be added, replaced, or deleted.



In the Layout Editor, all sorts of screen element types can be added. You can change screen element attributes and make modifications to screen layout.

You are, however, not allowed to:

- Delete an existing screen element



You can, however, make the element *Invisible* by selecting the appropriate checkbox from among its attributes. This allows value transport to and from the screen element to continue at runtime, so that unwanted changes to flow are minimized.

- Modify the name of a screen element
- Change an element's type (for example, from a checkbox to a radio button).
- Change an element's graphical element (container) assignment (for example, from a step loop to a table control).

You can only perform the above actions in the Modification Assistant if you have created the element in question yourself.



However, if you still want to execute any or all of the functions listed above, you can always disable the Modification Assistant. Use *Edit → Modifications → Disable*



## Modifications in the Screen Painter

*assistant* to edit screens without the help of the Modification Assistant. For further information, please refer to the section on [Disabling the Modification Assistant \[Page 135\]](#).

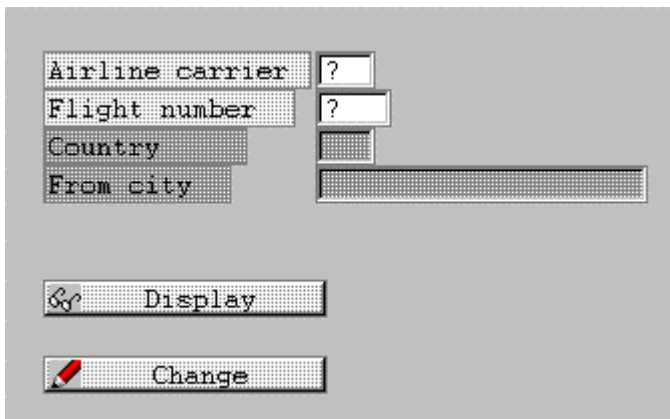
If you want to undo modifications you have made in the Screen Painter, no pushbutton is available for this function at the time being. This is due to the fact that if screen elements have been moved and new ones put in their place, two elements might collide if you try to undo specific modifications. In the Layout Editor, modifications can only be manually reset. For example, you can move a screen element back to its original position. Afterwards, it is no longer classified as a modified element and the system once again treats it as an original element.

There are, however, two different ways in which screen modifications can be undone. One involves the Modification Browser - where you can reset the attributes of whole screens (see [The Modification Browser \[Page 137\]](#)) - the other, the initial screen of the Screen Painter. From the latter screen, use the *Edit → Reset to original* function to reset attributes of whole screens.

## Modifications in the Graphical Layout Editor

When making changes to existing screens, no special functions are necessary when using the graphical Layout Editor in the Modification Assistant. All changes are made using normal Layout Editor functions.

You want to insert two new input fields on a screen, for example, to more precisely define your selections and, as a result of their addition, you have to move two pushbuttons further down the page. Make these changes just as you normally would in the graphical Layout Editor.



The modifications will be highlighted with different shades of gray in modification mode. Changes made to new objects will be displayed in dark gray, changes to pre-existing objects in light gray.



Choose *Overview* for an overview of the modifications made to the current screen.





















The *Edit → Modifications → Display* function displays all modifications for a selected object.

Use the *Overview* symbol (upper right) during screen attribute display or modification to display modifications that have been made to that element.

## Modifications in the Screen Painter

El. type	Pushbutton 		
Name	PUSH1		
Text	Anzeigen_____		
Icon name	ICON_DISPLAY 		
Quick info	Display		
Line	8	DefLg	19
Column	3	VisLg	17

The same function is available from the element list (to the left of the field name).

Name		General attr	Text
	 SPFLI-CARRID		Airline_carrie
	 SPFLI-CARRID		_____
	 SPFLI-CONNID		Flight_numb
	 SPFLI-CONNID		_____
	 SPFLI-CITYFROM		Depart.city
	 SPFLI-CITYFROM		_____
	 SPFLI-CITYTO		Arrival_city
	 SPFLI-CITYTO		_____
	 PUSH1		Anzeigen_____

If you want to make your modifications in the Alphanumeric Screen Painter, read the section on [Modifications in the Alphanumeric Screen Painter \[Ext.\]](#).

## Modifications in the Menu Painter

Menu enhancements can also be made in the Modification Assistant. Menu entries, pushbuttons, icons, and so on, can be both added and deleted. All changes made to the standard are highlighted in color.

The following is an example of how modifications are made in the Menu Painter using the Modification Assistant.

Assume that you want to add a new function to a program's menu bar. In addition, you want to alter a menu bar entry that already exists.

In order to change an original object in the Menu Painter, select the object and choose the Change sub-object pushbutton to make the field ready for input.

If you want to add a new entry to a menu bar, you can also use the *Insert entry* function to create a new entry. Here you can either create a new menu or refer to a pre-existing one (F4). New menu creation takes place in the same way as in the 'normal' editor mode.

All modified sub-objects are highlighted in color.

[illegible]

## Modifications in the Menu Painter



Consult *Utilities* → *Help texts* → *Color legend* for a more comprehensive explanation of how colors are used in modification mode.



Whenever you want to add a menu to a menu bar, the new entry is highlighted in color. However, if the new entry is an original menu, then it will not be highlighted at all.

In our example, the pre-existing menu entry *Exit* in the *Flight data* menu is going to be assigned a different function type. Double-clicking on the entry produces the following dialog box:

The screenshot shows the 'Function attributes' dialog box. On the left, a list of menu items includes 'EXIT' which is highlighted. The dialog box contains the following fields and buttons:

- Function code:** EXIT
- Functional type:** E Exit command
- Modification:** Original
- Buttons:** Change sub-object
- Static function texts section:**
  - Function text:** Exit
  - Icon name:** (empty)
  - Icon text:** (empty)
  - Info. text:** (empty)
  - Fastpath:** X
  - Modification:** Original
  - Buttons:** Change sub-object
- Bottom buttons:** Change text type, (Close button)

All changeable fields are ready for input after the corresponding pushbutton has been chosen (*Change sub-object*).

Now, the new function *Edit* should appear in the application toolbar. First, assign the function code to an F-key. Place your cursor on the short description of the function key setting and choose *Change sub-object*. All non-assigned function keys are now ready for input. Make your entry. For the application toolbar entry use *Insert entry*.



Use *Cut* to remove entries from sub-objects. Whenever you remove an original entry from a menu, it is highlighted in color.

In addition to the *Change sub-object* function, several other functions are available to you in modification mode in the Menu Painter.

Choose *Overview* for an overview of the modifications made.

**Modifications in the Menu Painter**

You can display the original version of any object selected by using the *Display original* function (for example, a menu bar, a menu, a function text, or even the original version of the entire status).

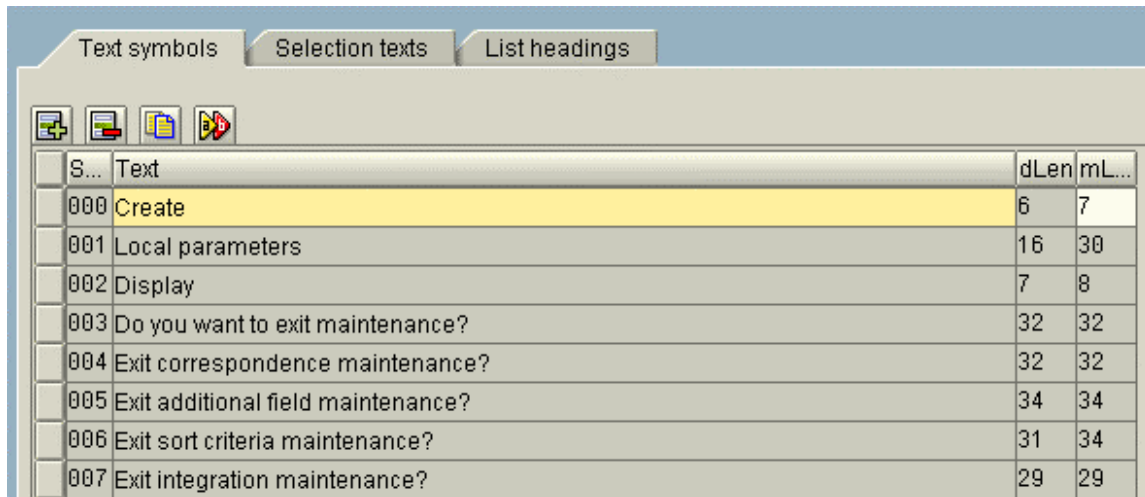
*Reset to original* allows you to reset all objects that have been modified by a customer back to their original form. For further information, see [Resetting to the Original \[Page 136\]](#).



## Modifying Text Elements

Choose *Development* → *Programming environment* → *Text elements* in the ABAP Workbench to branch to text element maintenance. When making a modification, choose the corresponding type of text and choose *Change*.

For example, you want to change the text elements for a program:



	S... Text	dLen	mLen
000	Create	6	7
001	Local parameters	16	30
002	Display	7	8
003	Do you want to exit maintenance?	32	32
004	Exit correspondence maintenance?	32	32
005	Exit additional field maintenance?	34	34
006	Exit sort criteria maintenance?	31	34
007	Exit integration maintenance?	29	29

Place the cursor on the line you want to modify. Choose the *Change original text* pushbutton. The addition of new text elements is also supported in the Modification Assistant.

The procedure for modifying selection texts and list headers is much the same. When changing column headers, be aware that the text can be modified and reset line for line.

In addition to the *Change original text* function, the functions *Display original*, *Reset to original*, and *Modification overview* are available to you here.

## Modifying and Adding Function Modules

## Modifying and Adding Function Modules

You can add function modules and modify existing function modules in the Modification Assistant. You can:

- Add modules from the customer namespace to function groups in the standard



During preliminary corrections, the name of the function module added can also come from the SAP namespace.

Create the module and assign it to an existing function group. Define the function module interface by determining its parameters and exceptions. Then write the function module's code. At this point, the pushbuttons that you are accustomed to using in the ABAP Editor are put at your disposal. For further information, see [Modifications in Programs \[Page 118\]](#).

- Modify the process type of a function module

The screenshot shows the 'Attributes' tab of the Modification Assistant. The 'Classification' section has 'Function group' set to 'STZ1' and 'Application' set to 'S'. The 'Processing type' section has 'Normal function module' selected with a radio button. Other options are 'Remote-enabled module', 'Update module', 'Start immed.', 'Immediate start, no restart', 'Start delayed', and 'Coll.run'.



Be certain that this type of change is compatible with your parameters and the contents of the function module.

- Enhance the interface of function modules and guarantee compatibility

You can add new parameters. If you do not enter a *Default* for newly added parameters, they automatically receive the attribute *Optional*.

You can always change the reference fields and defaults of those parameters that already exist.



## Modifying and Adding Function Modules

Function module CALCULATE_NEW_PRICE Active					
Attributes Import Export Changing Tables Exceptions					
Para	Type	Refe	Defa	Opti	Pa...
VALUE	TYPE	CKMLCR-SALKV		<input type="checkbox"/>	<input type="checkbox"/>
PRICE_UNIT	TYPE	CKMLCR-PEINH		<input type="checkbox"/>	<input type="checkbox"/>
STOCK	TYPE	CKMLPP-LBKUM		<input type="checkbox"/>	<input type="checkbox"/>
OLD_PRICE	TYPE	CKMLCR-PVPRS		<input checked="" type="checkbox"/>	<input type="checkbox"/>
CURRENCY_TYPE	TYPE	CKMLCR-CURTP		<input checked="" type="checkbox"/>	<input type="checkbox"/>
OUTPUT_PRICE	TYPE	CKMLCR-CUROU		<input checked="" type="checkbox"/>	<input type="checkbox"/>
				<input type="checkbox"/>	<input type="checkbox"/>

- Modify the logic of a function module

The procedure for modifying function module code is much the same as the procedure for modifying programs.

Use *Modification overview* to display an overview of the modifications you have made to the module.

---

**Modifications in the ABAP Dictionary**

## **Modifications in the ABAP Dictionary**

[Modifying Tables and Structures \[Page 131\]](#)

[Modifying Data Element Attributes \[Page 132\]](#)



[Collective Search Helps \[Ext.\]](#) delivered by SAP can be enhanced to include customer-specific search paths.

## Modifying Tables and Structures

Use the append technique to attach fields to transparent tables and structures.

To do so enter the name of the table or structure that you want to modify on the initial screen of the ABAP Dictionary. The remainder of this procedure is described in the section [Attaching Append Structures \[Ext.\]](#) in the ABAP Dictionary documentation.



Append structures can also be created in the ABAP Dictionary in display mode. In this case, no key from the SAP Service System is necessary.

## Modifying Data Element Attributes

## Modifying Data Element Attributes

In the Modification Assistant, you can modify all data element attributes. Place the cursor on the field that you want to modify and choose *Create*.

The screenshot shows the SAP Modification Assistant interface. At the top, there is a toolbar with various icons and a 'Documentation' button. Below the toolbar, the 'Data element' is set to 'S\_CARR\_ID' and its status is 'Active'. The 'Short text' is 'Airline carrier ID'. The 'Attributes' tab is selected, displaying a table with columns for 'Short', 'Length', and 'Field label'.

Short	Length	Field label
Short	7	Airline
Medium	16	Airline carrier
Long	16	Airline carrier

A 'Change attribute' dialog box is open for the 'Medium' attribute. It shows the 'Medium field label' 'Airline' highlighted in yellow. At the bottom of the dialog, there are 'OK' (green checkmark) and 'Cancel' (red X) buttons.

All changes that you make are subsequently highlighted in color.

Use *Modification overview* to display an overview of the modifications you have made to the current data element.

You can use *Display original* to display the original of an attribute you have selected with your cursor.

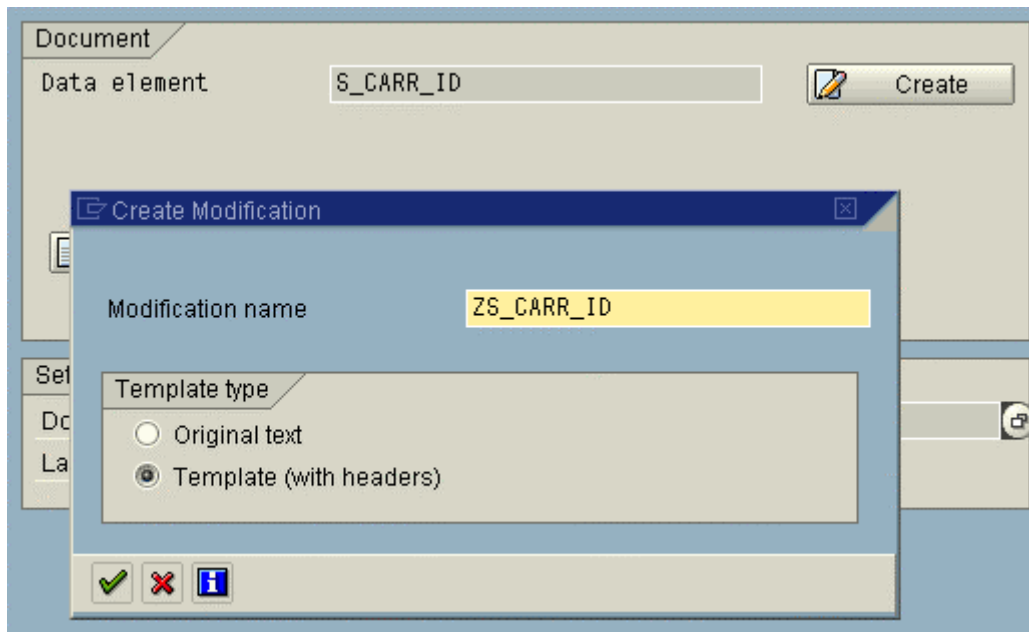
*Reset to original* allows you to reset all objects that have been modified by a customer back to their original form.

## Modifying Documentation

The following document classes can be modified:

Document class	Maintenance transaction
Data element	SE61, SE11
Text in dialog	SE61
Data element supplement	SE61, SE11
Function module documentation	SE37
Message	SE61, SE91
Report/module group, Logical database	SE61, SE38
General text	SE61

For example, you want to change the documentation for a data element. Choose *Utilities* → *Documentation* (SE61) from the ABAP Workbench menu. On the lower portion of the screen, enter *Data element* as your document class and the name of your data element in the *Data element* field. Now choose *Change* and make the changes you want. The following dialog box appears:



Now assign a name within the customer namespace and decide if you want to modify or enhance a text that already exists (choose the *Original text* radio button) or if you simply want to use one of the headers stored for this document class in the standard. If you decide on the latter course of action, choose *Template (with headers)*.

Choosing *Change* from the *Utilities* → *Documentation* menu displays originals only if no modified text is available.

You can reset modifications to their original form by deleting the document from the customer namespace.

---

**Modifying Documentation**

You can modify documentation in the Function Builder by choosing the *Function module doc.* pushbutton in change mode. A dialog box like the one pictured above appears. Function module parameter documentation is not supported in the Modification Assistant.

## Disabling the Modification Assistant

In the Modification Assistant, you can make all customary modifications to Repository objects. If, however, it should ever become necessary to disable the Modification Assistant (for example, when uploading a program to the Editor), choose the *Edit → Modifications → Disable assistant* menu function for the appropriate Repository object.

Only make this sort of change after considering whether or not it is absolutely necessary and thinking about the consequences that such a change will have during the adjustment process. The system will register the fact that there are modifications, however detailed display of the modifications will not be possible since the Modification Assistant will have been disabled at the time they were made. Modification adjustment can only take place during an upgrade using version management functionality.

## Resetting to the Original

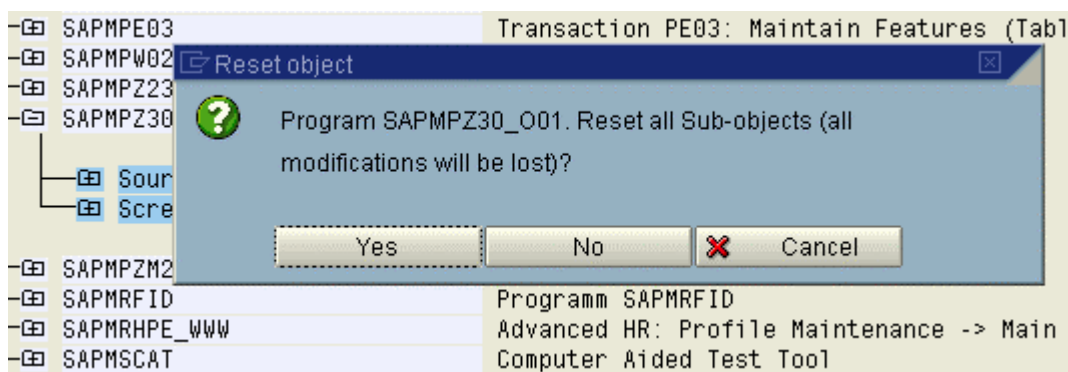
## Resetting to the Original

If you want to undo the modifications you have made to certain Repository objects, you can do this on the initial screen of the corresponding ABAP Workbench tool using the *Edit* → *Modifications* → *Reset to original* function. You may also undo modifications in the Modification Browser (SE95) by selecting the Repository object and choosing *Reset to original*. This will reset the entire Repository object to its version in the standard.



Includes that belong to a program cannot be individually reset.

Whenever an object, for example a program name, is chosen, any modifications made to its associated sub-objects are reset to their original form as well.



After you have reset a modification, there is no way in which this modification can be retrieved.

In order to maintain consistency, modifications can only be reset for entire Repository objects. To undo individual modifications (for example on a particular status), you have to navigate to the modification directly using the appropriate tool. If you navigate to where the modification was made, you can reset selected sub-objects (for example a menu bar) to their original appearance and functionality.



The section on [Modifications in the Screen Painter \[Page 120\]](#) describes peculiarities that you should pay attention to when resetting modifications in the Screen Painter.



## The Modification Browser

An overview of all modifications and enhancements found in your system can be displayed from the ABAP Workbench choosing the function *Overview → Modification Browser (SE95)*. You can also display an overview from the Repository Browser using *Environment → Modification Browser*.

The following selection screen appears, where you can input selections that limit what the Modification Browser outputs:

The screenshot shows the SAP Modification Browser: Object selection screen. The title bar includes 'Selections', 'Edit', 'Goto', 'System', and 'Help' menus, along with standard SAP window controls and the SAP logo. The main area is titled 'Modification Browser: Object selection'. It features a 'Selections' section with three input fields: 'Last changed by' (with a yellow highlight), 'Last transport request', and 'Development class', each with a right-pointing arrow button. Below this is a 'Modified objects and enhancements' section with three checked checkboxes: 'With Modification Assistant', 'Without Modification Assistant', and 'Business Add-Ins'. At the bottom is a 'Grouping in the display according to' section with four radio buttons: 'Objects' (selected), 'Last changed by', 'Last transport requests', and 'Development classes'. The status bar at the bottom right shows 'BJA (1) (000)', 'ds0037', and 'OVR'.

Standard selections can be made on the upper half of the selection screen by entering the development class or who the object was last changed by. If you insert the request number in the field *Request/Task*, then all modifications and enhancements that belong to this request's tasks are displayed in an overview. You can restrict display further by entering the number of a specific task.

Standard objects that are not supported by the Modification Assistant during modification or upgrade can be displayed using the *Without Modification Assistant* checkbox. This category also

## The Modification Browser

displays all objects that were modified in the current SAP System before the Modification Assistant was introduced. *Business Add-Ins* provide you with an overview of all enhancements and modifications undertaken in your system. This checkbox is set in the standard. Further information on this way to create enhancements can be found in the section on [Business Add-Ins](#) [Page 77].

On the lower part of the selection screen, you can use the *objects yet to be adjusted* radio button to display a list of those objects that have yet to be adjusted since the last upgrade or patch. You can perform adjustments in transaction SPAU. Call this transaction using the menu path *Utilities* → *Maintenance* → *Upgrade utilities* → *Program compare*. For further information, see [Adjusting R/3 Repository Objects](#) [Page 149].

An overview of all changes to your system made *With Modification Assistant* resembles the following illustration:

Modifications and enhancements		
With Modification Assistant		
Data Elements		
Database Tables		
Structures		
Programs		
CATSSHOW		Display Time Sheet Data
CFW_WORKSHOP_TEMPLATE		CFW Workshop: HTML Demo Program for Calling Methods
CM_WORKBENCH		Program CM_Workbench
CORIAPI0		Copy confirmations of BAPI and update in AFRU
FICCOI00		Consolidation of Investments
FICSHY10		Hierarchy Maintenance for CG/CU
GFW_DEMO_MAPSERVER		Is Oil Internetszenario
IWBPREP		IWB: Prepare training system
MP000200		Module Pool: Infotype 0002 Personal Data
OFX_ACCOUNT_STMT_GET		Request Account Statement via OFX
OPENPS		Select activities with LDB
PP_ORDER_PROGRESS		Order progress report
PR_RECRUIT_WEB		Recruitment interface to the internet
RABUCH00		Depreciation Posting Run

The *Color key* pushbutton displays a list of what certain highlighting colors mean within the Modification Browser.



Modifications to programs that cannot be assigned to a modularization unit (for example additional fields or comment lines preceding a subroutine) are displayed in the node *External modularization units*.

Sources		
Outside of modularization units		
- IWBPREP	RFISCHER	02.10.1998

You can branch directly to ABAP Workbench tools from Modification Browser displays by placing your cursor on a specific object and choosing *Display* or *Change*.



Changes cannot be made to objects currently being adjusted.

**The Modification Browser**

You can also undo modifications from the Modification Browser by placing your cursor on a specific object and choosing *Reset to original*. Further information can be found in the section [Resetting to the Original \[Page 136\]](#).

The *Reset to original* function can also be used with objects that were modified without the help of the Modification Assistant. Using this function causes the object to be deleted from the modification overview. Since no original is available for the object, it is now treated as an SAP original (modifications may, however, be lost during upgrade).

Objects that have yet to be adjusted also show up in the Modification Browser and are highlighted in color.

## Upgrade Procedure/R/3 Support Packages

During an upgrade or an import of R/3 Support Packages, existing objects of the SAP standard are overwritten by new objects delivered. In order to help customers keep those objects that have been modified in a previous release, SAP now offers upgrade adjustment for all objects being upgraded in the form of transactions SPAU and SPDD. These transactions allow customers to enter their modifications into the corresponding new objects being delivered at upgrade. The Modification Assistant supports this process of adopting customer modifications.

In the past, all objects modified by a customer and delivered again by SAP, an IBU or an SAP partner, had to be manually re-modified during an upgrade. Now, modifications are either automatically adopted or the system provides you with an assistant for adjusting your modifications to the newly upgraded configuration.

In general, objects altered using the Modification Assistant can now be automatically accepted into the upgraded system if the modifications undertaken in the original version do not directly overlap those made in the customer version. If collisions occur between the two versions at upgrade (naming collisions, or if SAP has deleted an object modified by a customer), the system offers semi-automatic adjustment support. In some cases, however, you may still have to manually adjust objects using ABAP Workbench tools.

Objects modified according to the old system used prior to the advent of the Modification Assistant must be manually maintained after an upgrade has been run.

Use transactions SPDD and SPAU during an upgrade to select these objects and modify them manually.

- Transaction SPDD allows you to adjust modifications to ABAP Dictionary objects during an upgrade.

For more information, refer to:

[Adjusting ABAP Dictionary Objects \[Page 142\]](#)

[Prerequisites \[Page 143\]](#)

[Procedure \[Page 144\]](#)

[Preparing to Run Transaction SPDD \[Page 145\]](#)

[Transaction SPDD \[Page 146\]](#)



ABAP Dictionary objects modified using the Modification Assistant can be adjusted in transaction SPDD (see [Modifications in the ABAP Dictionary \[Page 130\]](#)).

- Transaction SPAU allows you to adjust programs, function modules, screens, interfaces, documentation, and text elements after an upgrade.

In addition, transaction SPAU allows you to adjust ABAP Dictionary objects where no data loss can occur, specifically:

- Search helps
- Views
- Lock objects
- Data elements (Texts)

**Upgrade Procedure/R/3 Support Packages**

Additional information about the modification adjustment procedure using transaction SPAU can be found in [Adjusting R/3 Repository Objects \[Page 149\]](#).

Additional information about upgrade procedure can be found in:

[Upgrading From Release 4.5A or Less \[Page 168\]](#)

[Modifications Made by Several Different People \[Page 169\]](#)

[Adjusting Other SAP Systems \[Page 171\]](#)

[Upgrading the Development System \[Page 172\]](#)

[Upgrading the Production System \[Page 173\]](#)

[Handling Change Requests During Adjustment \[Page 174\]](#)

[Choosing a Change Request for Modifications \[Page 175\]](#)

[Local and Transportable Change Requests \[Page 176\]](#)

[Releasing Tasks at the End of Modification Adjustment \[Page 177\]](#)

[Choosing a Change Request to Transfer Modification Adjustments to Other SAP Systems \[Page 178\]](#)

[Notes \(Troubleshooting\) \[Page 179\]](#)

---

Adjusting ABAP Dictionary Objects

## Adjusting ABAP Dictionary Objects

This section describes how ABAP Dictionary objects are adjusted. An adjustment is carried out using transaction SPDD, and must be performed as soon as you are prompted to carry out a modification adjustment for ABAP Dictionary objects.

This section deals with the following objects:

- Domains
- Data elements
- Tables (structures, transparent tables, pooled and cluster tables including their technical settings)



If you adjust data elements that have been changed with the Modification Assistant in an earlier release, the changes can be copied automatically.

Changes to other ABAP Dictionary objects such as lock objects and views cannot result in the loss of data. They are not handled by transaction SPDD, but by [transaction SPAU \[Page 149\]](#) when the upgrade has taken place.

You need not perform adjustments with transaction SPDD if:

- you have made no changes whatsoever to the SAP standard objects of the ABAP Dictionary
- you have only added your own development work to the SAP System, that is new domains, data elements, or table structures in the customer name range. Only changed SAP objects require further processing.

## Prerequisites

In the adjustment procedure, ABAP Dictionary objects before the upgrade (old version) are compared with objects after the upgrade (new version). Transaction SPDD guides you through the adjustment process and provides help. If you still need more information on the objects, this is provided in [Version Management of Repository Objects \[Ext.\]](#).

To prevent data from being lost, you have to adjust the new and old versions of ABAP Dictionary objects. You are presented with a list of all ABAP Dictionary objects that have been modified in the SAP System or changed by a customer transport into the SAP System (for example, <SID>K9\*). At the beginning of the upgrade, the current version of objects that have been changed by your transports is recorded, to save the old status. Before you can modify newly imported SAP objects, their current state is also stored in the versions database.

---

**Procedure**

## Procedure

The system automatically recognizes during upgrade if modification adjustment is necessary. You are then prompted to perform the modification adjustment.

You can run a preliminary analysis of necessary adjustments that shows you if and in which phase adjustments will be necessary before an upgrade is run. As of Release 3.0, those objects needing adjustment can already be displayed in transaction SPDD after you run the analysis. If you have a release older than 3.0, the UMODPROT.<SID> file located in the upgrade directory 'log' contains a list of the objects you need to adjust.

An adjustment is required for all objects that were both modified by you and delivered by SAP. If you have not made any or only very few ABAP Dictionary changes, you are generally not prompted to perform an ABAP Dictionary adjustment.

You must perform one of the following actions for all objects presented for adjustment:

- Retain modification:  
Maintain the change with the appropriate maintenance transaction.
- Cancel modification:  
Choose *Return to SAP standard*.

If you do not carry out any of these actions, the new SAP standard becomes active in your SAP System and your modifications are therefore overwritten. In a later upgrade, you will be prompted once again to adjust these modifications. It is therefore important that you perform one of the two actions to minimize the work involved in future upgrades.



## Preparing to Run Transaction SPDD

During the adjustment of ABAP Dictionary objects, the ABAP Workbench already has the new release (however, not the applications).

Before you begin adjustment, set the system change option for the SAP System to “modifiable” for all namespaces:

1. Start transaction SE03 as user DDIC and choose *Administration → Set system change option → Global setting...*
2. Select *Modifiable* in the *Global setting* dialog box.
3. Choose *Continue*.
4. Choose *Edit → Select all*.
5. Save your entries.
6. Then log on as a normal user, since user DDIC may not perform any repairs.



When you make adjustments keep the following in mind:

- Work under your normal user name and not as user DDIC.
- Whenever an object is modified, a request query dialog box appears asking you to specify the change request where modifications should be recorded. For the first object you modify, create a new change request with the function *Create request*.

You should use the same request for all further changes you make in connection with SPDD.

- Store your work in the ABAP Dictionary (SE11) but **do not activate any objects**. Activation is carried out automatically after the adjustment.

For the adjustment of future Release upgrades or other SAP Systems (such as production systems), it is important to observe the following. Create exactly one request in the Workbench Organizer for all corrections carried out in connection with transaction SPDD. For further information read the sections [Adjusting Other SAP Systems \[Page 171\]](#) and [Handling Change Requests During Adjustment \[Page 174\]](#).

**Transaction SPDD**

## Transaction SPDD

Processing in transaction SPDD is based on a Release upgrade that is stopped by the R3up upgrade control program. SPDD provides the following functions:

- Display
- Comparing old and new versions
- Maintenance
- Returning to SAP Standard
- Marking objects as processed
- Refresh
- Marking for Transport

### Display

The first screen presents you with a hierarchical list of objects that have been changed by you and SAP. You can arrange the display by development class or by correction. Switch to a different display with *Settings* → *Switch view*.

Within individual development classes or corrections, the objects are displayed according to type. To view a changed object, select it with a double-click. You are automatically shown an overview of all existing versions of the selected object.

Objects in the hierarchical list are highlighted with different colors that reflect their status. To find out the meaning of the colors, choose *Utilities* → *Color key*.

### Comparing Old and New Versions

Selecting an individual object lets you view all existing versions of the object stored in [Version Management \[Ext.\]](#). The display is the same as the one you are presented with when you compare two objects in version management. The information reflects the status of the objects after their activation. For example, a comment in a table field indicating deletion means that the field will be deleted after activation, provided no maintenance is carried out beforehand.

It shows you:

- the version of the object before the upgrade (second to last version)
- the new SAP standard (last version).

The last version has just been created by user R3trans or SAP, whereas the second to last version was created by a developer working in your SAP System.

### Maintenance

Transaction SPDD lets you compare the most important attributes of old (customer) and new (SAP) versions of domains and data elements. This helps you to decide how to proceed. Perform the adjustments required in the ABAP Dictionary maintenance transaction (SE11), preferably in a second window.



If you adjust data elements that have been changed with the Modification Assistant in an earlier release, the changes are copied automatically.

To assist you in selecting tables, SPDD not only presents you with a comparison but also recommends the sequence of the fields that need to be maintained. You can automatically accept the recommendation, which takes all new features of the ABAP Dictionary introduced in the new Release into account. When you adjust tables and structures, SPDD recommends that you place inserted fields in an append structure to ensure that your changes are not overwritten by SAP in the future. Newly created append structures are in the customer namespace.

To correctly handle table includes, you need to adhere to the sequence of the objects given for each object type. Existing includes are always displayed before the tables themselves.



At this point it is imperative to adjust the include structures, otherwise you could lose data.

Make sure that you just save your activities in the ABAP Dictionary (SE11). **Do not activate the objects.** Activation is carried out automatically after the adjustment.

## Returning to SAP Standard

If after comparing the results of the analysis, you conclude that you do not wish to retain previous modifications, choose *Return SAP stand.* for the object in question. For example, you may need this function if you have patched an object by hand. When SAP delivers the same patch, together with other corrections in the next upgrade, you can revert to the SAP standard because you no longer need your own modification.

The *Return to SAP standard.* function lets you indicate that you no longer wish to retain the changes made to the object and want to revert to the SAP standard. An object marked in this way is no longer displayed in future upgrades by transaction SPDD, even though it was included in a repair.



If you have defined a customer append structure for a table and consequently not altered the table itself, it is essential that you afterwards select the function *Return to SAP Standard.* This ensures that transaction SPDD will not present the table for adjustment during one of the next upgrades.

After an object has been reset to original, it is treated as an unmodified object during the next upgrade and can be overwritten without its previous form being saved.

## Marking Objects as Processed

To facilitate work with the generated object list, you can assign the status *Completed* to each object. Selecting the function changes the status of the object and indicates this by highlighting it with a different color. The completed flag is only meant to assist you in organizing your work and has no other internal function.

## Refresh

When you select this function, the information in the database is reread and the display updated. Only the status of objects indicated by various colors is changed. The number of objects displayed remains the same.

---

Transaction SPDD**Marking for Transport**

This function is only active for the upgrade and not when applying patches.

The *Select for transport* function helps you to adjust other SAP Systems. For further information, please refer to the section on [Adjusting Other SAP Systems \[Page 171\]](#).

## Adjusting R/3 Repository Objects

The adjustment of R/3 Repository Objects is carried out with transaction SPAU. Only those objects that have been modified by you and are being redelivered by SAP in an upgrade are presented for adjustment. If the objects you modified are not being delivered by SAP with the current upgrade, then they do not appear in the display.

The system automatically recognizes during upgrade if modification adjustment is necessary. You are then prompted to perform the modification adjustment.



You can run a preliminary analysis of necessary adjustments that shows you if and in which phase adjustments will be necessary before an upgrade is run. As of Release 3.0, those objects needing adjustment can already be displayed in transaction SPAU after you run the analysis. If you have a release older than 3.0, the UMODPROT.<SID> file located in the upgrade directory 'log' contains a list of the objects you need to adjust.

You must decide whether the objects contained in the adjustment list should retain your modifications or if they should revert to the standard. This is mandatory and must be carried out for each object visible in the adjustment transaction. Otherwise, the new SAP standard stays active in your system. The objects remain visible in transaction SPAU as not adjusted. Therefore, you should always carry out one of these two actions.

The following sections contain additional information about the modification adjustment procedure in transaction SPAU .

[Preparing to Run Transaction SPAU \[Page 150\]](#)

[Transaction SPAU: General Functions \[Page 151\]](#)

[Adjustment Category: With Modification Assistant \[Page 156\]](#)

[Adjustment Category: Without Modification Assistant \[Page 167\]](#)

Please consult the following documentation if you are upgrading from Release 4.0B or less to a higher release ( 4.5A or greater) number for the first time: [Upgrading From Release 4.5A or Less \[Page 168\]](#).

## Preparing to Run Transaction SPAU

## Preparing to Run Transaction SPAU

Before you begin adjustment, set the SAP System change option to “All objects (with Workbench Organizer)”:

1. Start transaction SE03 as user DDIC and choose *Tools → Administration → Set system change option*.
2. Select *Modifiable* in the *Global setting* dialog box.
3. Choose *Continue*.
4. Choose *Edit → Select all*.
5. Save your entries.
6. Then log on as a normal user, since user DDIC may not perform any repairs.



When you make adjustments keep the following in mind:

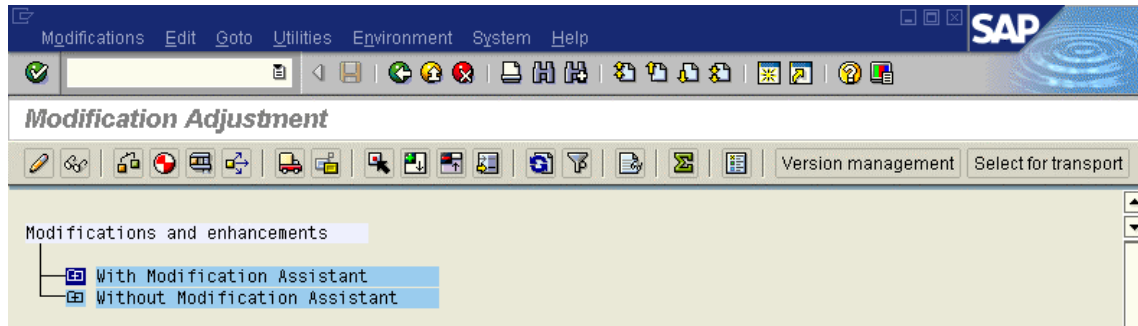
- Work under your normal user name and not as user DDIC.
- Whenever an object is modified, a request query dialog box appears asking you to specify the change request where modifications should be recorded. For the first object you modify, create a new change request with the function *Create request*.

You should use the same request for all further changes you make in connection with SPAU.

For the adjustment of future Release upgrades or other SAP Systems (such as production systems), it is important to observe the following. Create exactly one request in the Workbench Organizer for all corrections carried out in connection with transaction SPAU. For further information read the sections [Adjusting Other SAP Systems \[Page 171\]](#) and [Handling Change Requests During Adjustment \[Page 174\]](#).

## Transaction SPAU: General Functions

Transaction SPAU allows you to process Repository objects needing adjustment. An overview shows all objects that have been modified by you, SAP, an IBU or an SAP partner.




The *Without Modification Assistant* category displays objects that were modified in the current SAP System before the Modification Assistant was installed. This category also displays all objects from those areas in the standard where modifications are not supported by the Modification Assistant.

You can display a modified object by double-clicking on it.








In addition to the common functions for displaying the structure tree, such as *Expand/Collapse subtree*, and so on, other functions are available for modification adjustments that are explained in more detail below:

	<p>With <i>Select subtree</i> you can select several objects or a subtree and simultaneously change the status of several objects.</p> <p>Selecting multiple objects is only supported for the <i>Adjust modifications</i>, <i>Reset to original</i> and <i>Delete from display</i> functions.</p>
	<p>With <i>Update list</i> you remove objects that were reset to original from the display. Tree nodes that list only adjusted objects are highlighted in color.</p>
	<p>With <i>Filter</i>, you can call the selection screen of transaction SPAU again. This makes sense if the node <i>No object with selected filter settings found</i> appears in the tree display. If this is the case, choose the basic settings on the selection screen so that all objects are displayed.</p>
	<p>With <i>No. of log entries</i>, you get an overview of how many objects have been modified and how many objects need to be adjusted. The system displays a list with the number of log entries that are below a tree node. The number of log entries shown in transaction SPAU, and the modification adjustment status are displayed separately.</p> <p>For one transport object, several log entries can exist for each modularization unit. They are grouped together for the tree display and are represented as a single object. The name of the user that has made the last change, and the associated transport request are displayed. If several log entries exist, the adjustment mode with the highest priority is shown during modification adjustment. If the user name, the transport request number or the adjustment mode are not compatible with the selection criteria of the subtree, no object is represented.</p>




**Transaction SPAU: General Functions**

	With <i>Color legend</i> , you can display explanatory information on the colors used in the tree structure. Depending on their status, the objects are highlighted in different colors.
---	--

The following icons indicate the status of the individual Repository objects:

	<p><b>Automatic adjustment</b></p> <p>The modification of the customer can be adopted automatically. Clicking on the icon causes the system to automatically adjust the object.</p> <p>This icon only appears in the <i>With Modification Assistant</i> category.</p> <p>You can select a subtree or several objects, or you can place the cursor on a single object.</p>
	<p><b>Semi-automatic adjustment</b></p> <p>Semi-automatic means that each tool will individually offer you support during the adjustment process. When adjusting programs, the splitscreen editor is called, whereas in the other tools any entries made in the collision dialog box lead to the necessary adjustments being made automatically.</p> <p>Also read the information provided in the following sections.</p> <p>As with the green traffic light, the semi-automatic adjustment icon only appears in the <i>with Modification Assistant</i> category.</p>
	<p><b>Manual adjustment</b></p> <p>Objects in the <i>Without Modification Assistant</i> subtree can only be post-processed manually after the adjustment process. Manual adjustment means that you must make modifications without any special support from the system. Use the log () as a help. Using <i>Version Management</i>, you can retrieve old versions or use your recordings to process the newly imported objects.</p> <p>In rare cases, the red traffic light may also appear in the <i>With Modification Assistant</i> category.</p>
	<p><b>Unknown adjustment mode</b></p> <p>The adjustment mode (manual, semi-automatic, automatic) for at least one of the objects in question could not be determined for modification adjustment with the Modification Assistant. If this is the case and you start transaction SPAU, a dialog box informs you that you can start a background process by choosing the appropriate pushbutton which determines the adjustment modes for all objects.</p>
	<p><b>Object adjusted</b></p> <p>After the adjustment has taken place, the traffic light symbols are replaced with either a green checkmark or a stop sign. For objects supported by the Modification Assistant, you can display the upgrade log by clicking the checkmark or choosing the  pushbutton.</p>



	<p><b>Adjustment problems</b></p> <p>The stop sign indicates adjustment problems. In most cases, you must solve these problems manually.</p> <p>You can display the log by clicking the stop sign or choosing the  pushbutton. This function informs you what problems occurred at upgrade and what actions are necessary to correct them. Pay special attention to the <i>Open (unresolved) problems</i> category. If not all objects in the <i>With Modifications Assistant</i> category can be successfully adjusted, then a dialog box is displayed informing you that unresolved problems still exist.</p>
	<p><b>Reset to original</b></p> <p>If you choose <i>Reset to original</i> for an object displayed in the overview, no modifications are adopted for this object. The original is the version that was last imported into the SAP System during an upgrade or the application of an R/3 Support Package. Also read the following section on this subject.</p> <p>For the <i>Reset to original</i> function, you can select a subtree or several objects, or you can place the cursor on a single object.</p>

For one transport object, several log entries can exist for each modularization unit. They are grouped together for the tree display and are represented as a single object. The name of the user that has made the last change, and the associated transport request are displayed. If several log entries exist, the adjustment mode with the highest priority is shown during modification adjustment. If the user name, the transport request number or the adjustment mode are not compatible with the selection criteria of the subtree, no object is represented.

Choose *Goto* to get a list of the deleted objects. It may take several minutes to find the objects. These objects were previously modified by the customer and deleted during an upgrade or the import of a Support Package. Using *Version Management* or your own recordings, you can restore the modifications. Afterwards, you should remove the log entries of the objects deleted. To do this, use the delete function of the list display.

Before leaving transaction SPAU, you should delete all objects from the display for which modification adjustment is complete. To do this, you can select a subtree, or you can place the cursor on a single object. Using the *Modifications* menu, you can delete the selected objects from the display.

## Reset to Original

If you no longer want to maintain modifications that you have made to an object, then call the *Reset to original* function for that object. This function is particularly useful if you have manually imported preliminary corrections from SAP, an IBU or an SAP partner. After preliminary corrections have been delivered by SAP, an IBU or an SAP partner with the next upgrade, your initial preliminary correction phase should be concluded.

If you choose *Reset to original*, you accept the version that was imported during the upgrade or applied with the R/3 Support Package as the new original. Originals deleted from modification adjustment although they have been added to a transport request are not displayed in transaction SPAU after future upgrades. An object reset to original is no longer displayed in the Modification Browser.



## Transaction SPAU: General Functions

After an object has been reset to original, it is treated as an unmodified object during the next upgrade or the application of an R/3 Support Package and can be overwritten without its previous form being saved.

If the object that you want to reset has not yet been adjusted the imported active version is confirmed as the original. You can also reset objects to original that have already been adjusted. If the objects concerned are supported by the Modification Assistant, the modified active version is replaced with the original version that was saved in the modification logs. If the objects concerned are not supported by the Modification Assistant, you receive a warning that no original exists. The modified active version is confirmed as the original version.

If you choose the *Reset to original* pushbutton, the system requires you to enter a transport request. The transport request is necessary to ensure that the modification information for an object is available throughout the entire transport process of the customer. However, this means also that an object that was reset to original must be transported into all subsequent systems. In the case of a consolidation transport (transport type K) or a transport of copies (transport type T), the modification information is included in the transport. In the target system, the object reset to original is no longer displayed in the Modification Browser. It is not included in the modification adjustment process when the next upgrade is carried out or the next R/3 Support Package is imported.

## Version Management

You need version management if you want to adjust objects that are not supported by the Modification Assistant. Place your cursor on the object concerned and choose *Version management*.

Further information can be found under [Version Management of Repository Objects \[Ext.\]](#).

## Marking for Transport

The *Select for transport* function helps you to adjust other SAP Systems. In this case, do not use the release transport in the Workbench Organizer. If you choose the *Select for transport* function, the adjustment process needs not be executed again in the subsequent systems (for example, the production system).

Create only one transport request that you use to record all modifications.



If you have created several transport requests already, you must consolidate your object lists into one request. For example, if you have adjusted different objects in your development and your consolidation system, you must add the objects to a common transport request to include it in the adjustment process of the production system. For more information, see [Choosing a Change Request to Transfer Modification Adjustments to Other SAP Systems \[Page 178\]](#).

To automatically transfer modification adjustments to subsequent systems, choose *Select for transport* after releasing the tasks for the request. When an upgrade is carried out for the subsequent system, the marked transport is imported. A new adjustment is not necessary. For detailed information on transporting adjusted objects, see the following sections:

[Adjusting Other SAP Systems \[Page 171\]](#)

[Upgrading the Development System \[Page 172\]](#)

[Upgrading the Production System \[Page 173\]](#)

[Handling Change Requests During Modification Adjustment \[Page 174\]](#)

[Choosing a Change Request for Modifications \[Page 175\]](#)

[Local and Transportable Change Requests \[Page 176\]](#)

[Releasing Tasks at the End of Modification Adjustment \[Page 177\]](#)

[Choosing a Change Request to Transfer Modification Adjustments to Other SAP Systems \[Page 178\]](#)

## Removing Adjusted Objects From Display

Before you exit transaction SPAU, you should choose *Modifications* → *Delete from display* to remove all objects from display for which modification adjustment is complete. To do this, you can select a subtree or several objects, or you can place the cursor on a single object.

**Adjustment Category: With Modification Assistant**

## Adjustment Category: With Modification Assistant

Objects modified *With the Modification Assistant* are adjusted automatically or in the individual ABAP Workbench tools using either corresponding dialog boxes or the splitscreen editor. In rare cases, direct maintenance of adjustment objects may be necessary.



Test all objects after an upgrade! Even those objects modified automatically can cause unforeseen errors when subsequently used in an area different from where they were modified.

### Automatic Adjustment

A green traffic light icon next to a Repository object in the adjustment list means that this object's modifications can be adopted automatically. Clicking on the icon causes the system to automatically adjust the object.

To automatically adjust multiple objects, place your cursor on a subtree and choose *Select subtree*. Now click on one of the traffic light icons for the selected objects or choose the menu option *Modifications → Adjust modifications*. After you have chosen a change request, a dialog box appears confirming that the system has adopted the customer modifications. The objects are subsequently marked with the *Object adjusted* symbol.

### Semi-automatic Adjustment

The yellow spotlight stands for semi-automatic adjustment. When you click on a yellow traffic light (if a change request has already been entered), you either branch to the splitscreen editor or, in other ABAP Workbench tools, a dialog box appears asking you to correct any collisions that have occurred. In rare cases, direct maintenance of adjustment objects may be necessary. Choose *Display log* to display a log of unresolved problems.

Examples of semi-automatic adjustment can be found in the following sections:

[Adjusting Parts of Programs \[Page 157\]](#)

[Adjustments in the Screen Painter \[Page 159\]](#)

[Adjustments in the Menu Painter \[Page 162\]](#)

[Adjusting Text Elements \[Page 164\]](#)

[Adjusting Function Modules \[Page 165\]](#)

[Adjusting Documentation \[Page 166\]](#)



Use transaction SPDD to adjust modifications to ABAP Dictionary objects.

## Adjusting Programs

Whenever you alter a program using the Modification Assistant, most changes can be adopted automatically because source comparison now takes place at the modularization unit level (that is routine, module, or event level) and not at the transport object level.

Modifications to modularization units are automatically adopted at release upgrade if:

- They have only been changed by the customer, but not in the original version
- They have been newly created by the customer

Automatic adoption of modifications at release upgrade is not possible if both the original version of the modularization unit and the customer version have been changed (for example, if interface changes have been made that affect either function modules or subroutines, or if the source code has been altered). For these kinds of adjustments the Modification Assistant does, however, offer support in the form of the splitscreen editor.

You must edit these programs manually using the editor whenever the part of program changed by the customer no longer exists. They can be found in the adjustment log marked as unresolved problems. In transaction SPAU, a stop sign icon denotes these problems. Click on the icon to display the log file.

### Example:

A customer has changed the name of a subroutine. This subroutine is meant to perform authorization checks in a manner that differs from the standard. The modifications to the subroutine can be adopted automatically, if SAP has not changed the subroutine in its new release. However, since the modularization unit in the new original differs from the one in the old original, automatic adjustment is not possible in this case and the splitscreen editor is called. The system accepts all modifications to the SAP original that can be adopted automatically before it calls the editor.

The window on the left displays the new SAP original, while the window on the right displays the source code modified by the customer.

5	form safety_check using rcode.	5	*( REPLACE
6		6	*\FORM SAFETY_CHECK USING RCODE.
7		7	form customer_safety_check using r
8		8	
9		9	*( REPLACE
10		10	
11		11	
12	local ok_code.	12	local ok_code.
13	rcode = 'EXIT'.	13	rcode = 'EXIT'.
14	check spfli ne old_spfli.	14	check spfli ne old_spfli.
15	clear ok_code.	15	clear ok_code.

The splitscreen editor is called in comparison mode. The positioning pushbutton *Next difference* allows you to scroll to the next source code difference between the two programs. At this point, you must decide whether you want to transfer your modifications the new original or if the original should remain unchanged. To make changes to the original, proceed as follows:

### Adjusting Programs

- Place your cursor on the appropriate passage in the modified source (right side) or select a portion of the source code and then choose *Copy*.
- Place your cursor on the corresponding passage in the original (left side) or select the block of code you want to replace and choose either *Insert* or *Replace*. This inserts the code you copied to the buffer into the original.

If further differences occur, choose *Next difference* to scroll to the next modification that was not automatically adopted by the system.

You may also use the combination *Next similarity* and *Next difference* to navigate forward.

After having made all necessary modifications, save your changes. A dialog box appears, asking you to confirm that you have adjusted all modifications. The following statistics are also displayed:

- Units to be modified
- Units automatically adjusted
- Units manually adjusted

If you answer 'yes', all modifications not yet adopted are lost. These are lost forever as soon as you leave the splitscreen editor.



If your new original version contains code that you would like to replace with unmodified code from the previous version, use *Block/buffer* menu to copy it to the buffer and *Insert/Replace* to insert it into the original.

Further information about how to use the editor can be found in the section on [The Splitscreen Editor \[Ext.\]](#).

## Adjustments in the Screen Painter

Certain adjustments are automatically adopted by the system:

- Newly created screen elements (as long as there is room for them).
- Changes to screen attributes including their short texts (changes to screen attributes are even automatically inserted if SAP has modified the attributes)
- Changes to original elements such as position, size, and text, as long as no name or type collisions occur, no elements overlap, and no screen limits are exceeded
- Flow logic modifications, as long as the modularization unit has only been changed by customers and not in the original version, or if it was added by the customer. In this case, PBO, PAI, POV, and POH all count as separate modularization units for the Modification Assistant.

However, you must adjust these kinds of objects if:

- Customers have modified a screen element and SAP has changed that element's type or container assignment.
- Screen size has been changed, cutting off certain elements
- Screens or screen elements changed by the customer have been deleted by SAP
- The maximum number of lines (200) is exceeded when repositioning overlapping fields
- Customers and SAP have created screen elements with the same name
- The Modification Assistant has been disabled
- Collisions occur in a screen's flow logic

If automatic adjustment is not possible, screens must be manually adjusted according to the following procedure:

- Edit screen attribute changes first, followed by screen element changes, and changes to flow logic.
- Before making individual manual changes, make any adjustments that can be adopted automatically
- Eliminate any screen element name conflicts using the appropriate dialog box (see example)
- If SAP has deleted a screen element modified by the customer, a dialog box informs you that the modification cannot be made. To rename an element, you must delete it and create a new one
- When screen elements overlap, a dialog box appears. Clear up positioning conflicts of this nature using the following strategy:
- Screen elements modified by customers should be moved down to an unoccupied line on the screen. Place modified table control columns at the end of the list of columns.
- If you have to move multiple screen elements from a single line, transfer them all to the same new line together.
- Elements from a single container (for example, a group of radio buttons) should be moved as a group

## Adjustments in the Screen Painter

You must use the Screen Painter to edit objects if:

- The Modification Assistant has been disabled
- The maximum number of lines (200) is exceeded when repositioning overlapping fields

### Example:

A customer has changed the flow logic and layout of a screen. The statement `module hugo.` has been added at the beginning of PBO and the statement `module exit.` and has been added at the beginning of PAI in the flow logic. In addition, the screen elements SPFLI-COUNTRYFR, SPFLI-CITYFR, SPFLI-COUNTRYTO, and SPFLI-CITYTO (each as both text and input/output fields) have been added.

SAP has also changed the flow logic and layout of this screen. SAP has replaced the statement `module status_0100.` at PBO with the statement `module status_0101.` and has added the screen elements SPFLI-COUNTRYFR and SPFLI-COUNTRYTO (each as both text and input/output fields).

At upgrade, semi-automatic adjustment of this screen takes place. This is due to the fact that:

- Modifications have been made at PBO both by the customer and in the SAP standard, and
- Screen objects with the same name have been added (a naming conflict has occurred)

During adjustment a dialog box appears, asking you to rename the screen object, so that the naming conflict can be cleared up.

The screenshot shows a SAP dialog box titled "Name conflicts". It contains the following text:

There are conflicts with the following screen elements:

Customer modification contains screen element as  
new element that has been added to the former original screen.

The new original screen contains an element with the same name, however.

The customer modification screen element must be renamed

Please adjust program accordingly

The dialog box has a table with two columns: "Previous name" and "new name". The "Previous name" column contains the following entries:

Previous name	new name
SPFLI-COUNTRYFR	
SPFLI-COUNTRYFR	
SPFLI-COUNTRYTO	
SPFLI-COUNTRYTO	

Then the splitscreen editor is called so that you can resolve the conflict occurring at PBO. The customer modification `module exit.` has already been adopted automatically at PAI.



Adjustments in the Screen Painter

1	process before output.	1	process before output.
2	module status_0101.	2	*{ INSERT
3		3	* module hugo.
4		4	*} INSERT
5		5	module status_0100.
6	*	6	*
7	process after input.	7	process after input.
8	*{ INSERT	8	*{ INSERT
9	module exit.	9	module exit.
10	*} INSERT	10	*} INSERT
11	module user_command_0100.	11	module user_command_0100.

**Adjustments in the Menu Painter**

## Adjustments in the Menu Painter

New interfaces can be automatically adjusted at upgrade if:

- There are no name conflicts between function codes, statuses, and titles
- Those texts modified were not changed in the original
- Menus, menu bars, F keys, and pushbuttons have been modified by both the customer and in the original, but these changes do not lead to any conflicts (see below).



If both the original and the customer version have been changed in an identical manner, then this text is no longer considered a modification after upgrade.

In the following cases, modifications cannot be automatically adjusted at upgrade. However, the system offers support when making the adjustments manually.

- Name conflicts (Same name has been chosen for a function code, status, or title in both the original and a customer modification)
- In both the original and a customer modification, the same menu, function, or title text has been altered in different ways
- In both the original and a customer modification, a link to the same menu bar, function key, or pushbutton setting has been altered in different ways
- In both the original and a customer modification a function type has been altered differently

You must use the Editor to edit objects if:

- If a menu bar or pushbutton setting has been altered both in the original and by a customer, and this modification will lead to there being more than 15 menu entries, 6 menu points, or 35 pushbuttons.
- In both the original and a customer modification, the same open pushbutton has been given different settings
- Objects modified by the customer have been deleted in the new original (function codes, menus, and so on)
- Objects used by customers in their own objects have been deleted in the new original (for example, a customer menu bar contains an SAP menu that has been deleted)



If any of these problems occur, they are marked as an unresolved problem and entered in the log.

If automatic adjustment is not possible, objects must be manually adjusted in the Menu Painter according to the following procedure:

- Eliminate name conflicts using the appropriate dialog box (see example). Rename customer objects everywhere they are used in the customer version.
- Before making individual manual changes, make any adjustments that can be adopted automatically.

### Adjustments in the Menu Painter

- Conflicts that arise when attributes have been altered ( for example, text changes) should be resolved using the appropriate dialog box.
- When saving new entries, you will be alerted to any new conflicts that arise from these entries (if any) and directed to the log in category *Open (unresolved) problems*, where you can then remove them manually.



The adjustment log gives you an overview of the modifications that have been made (both the new and old versions are displayed). From it, you can branch to the appropriate editor.

#### Example:

Both a customer and SAP have defined a new function code that overlaps.

In the dialog box below, the customer is prompted at upgrade to rename his function code so that no collision occurs.

In addition, the dialog box displays the parts of the program that have to be adjusted as a result of this change (if any).

 Naming conflict with function code Please rename the objects to avoid naming conflicts.		<input type="button" value="PRINT"/>
 Adjust the corresponding function code requests		
New function code original	<input type="text" value="PRINT"/> <input type="text" value="Printout"/>	
Customer function code	<input type="text" value="?"/> <input type="text" value="Print request"/>	

## Adjusting Text Elements

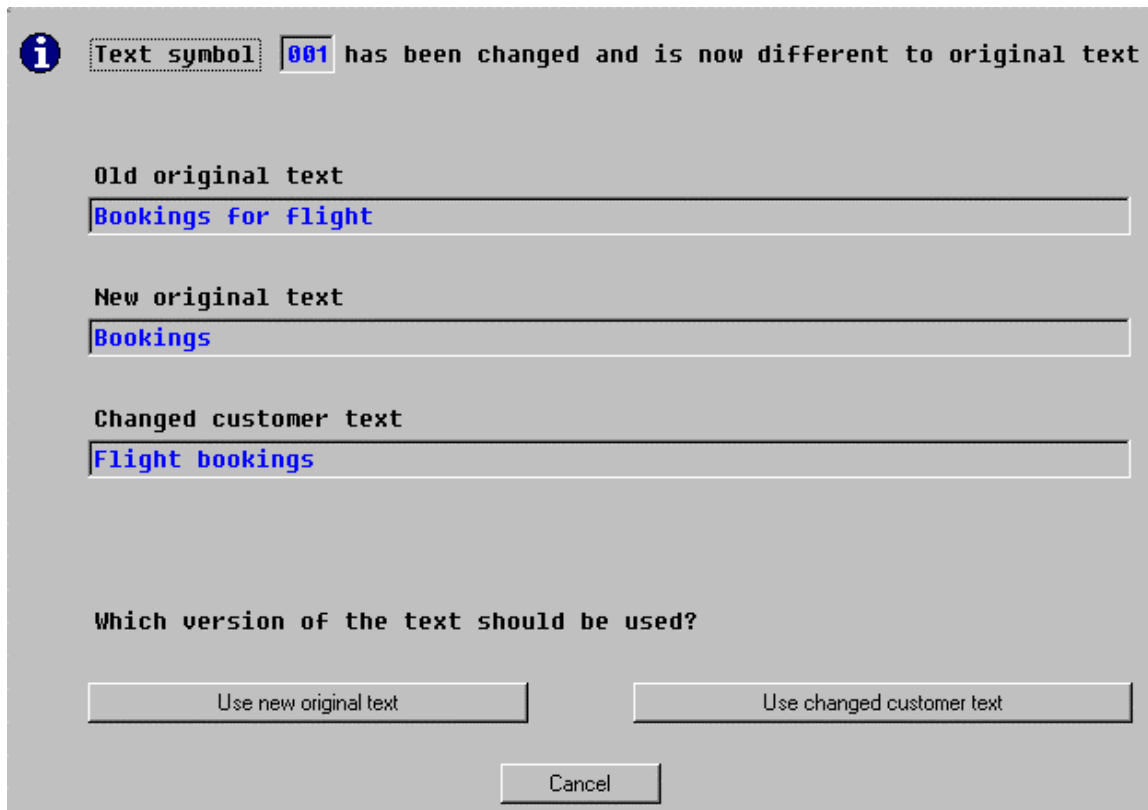
## Adjusting Text Elements

Texts are automatically adjusted if:

- Their number is different than that of the original texts
- The corresponding texts were not changed in the original

The system offers you manual adjustment support if:

- In both the original and a customer modification, the same text has been altered in different ways



The dialog box has a grey background. At the top left is an information icon (a blue circle with a white 'i'). To its right, the text 'Text symbol' is followed by a small box containing '001'. Further right, the text 'has been changed and is now different to original text' is displayed. Below this, there are three text input fields. The first is labeled 'Old original text' and contains the text 'Bookings for Flight'. The second is labeled 'New original text' and contains the text 'Bookings'. The third is labeled 'Changed customer text' and contains the text 'Flight bookings'. Below these fields, the question 'Which version of the text should be used?' is asked. At the bottom, there are three buttons: 'Use new original text' on the left, 'Use changed customer text' on the right, and 'Cancel' in the center.

**i** Text symbol **001** has been changed and is now different to original text

Old original text  
Bookings for Flight

New original text  
Bookings

Changed customer text  
Flight bookings

Which version of the text should be used?

Use new original text      Use changed customer text      Cancel

## Adjusting Function Modules

The system automatically adopts the following at release upgrade:

- Modules created by customers
- Parameters added by customers, as long as they do not conflict with any existing parameters
- Process types that have modified by customers, as long as they are compatible with their modules' parameters

The system offers you support during the adjustment process if:

- A new parameter in the original has the same name as a modified parameter
- The reference type of a parameter has changed in the original

The customer can choose pushbuttons in a dialog box to determine if the modifications are to be adopted or not.

If changes to the contents of an original function module are not compatible with a customer's modifications, see the section on [Modifications in Programs \[Page 118\]](#).



If problems still exist after adjustments have been made, a dialog box appears asking you to refer to the adjustment log for additional information. Unresolved problems can occur during process type modification if, for example, a function module is designated as “updateable” and export parameters have to be deleted.

## Adjusting Documentation

## Adjusting Documentation

In this area, you can adopt modifications whenever you want. Choose the corresponding pushbutton. You can view both the original and the modifications before adjustment.

A new version of the document was imported at update. An enhancement document was created for the old version which overwrote the original document.

The version before the upgrade is lost.

You can decide if the modification should overwrite the new version, or if the modification should be removed.

was imported at

New original

Modification

The screenshot shows a SAP dialog box with a light gray background. On the left, there is a text area with three paragraphs of text. The first paragraph is enclosed in a dashed border. To the right of the text area, there is a label 'was imported at' followed by a rectangular box containing two buttons: 'New original' and 'Modification'.

## Adjustment Category: Without Modification Assistant

To adjust objects *without the Modification Assistant*, use version management wherever possible (see [Version Management of Repository Objects \[Ext.\]](#)). When modifying objects where version management cannot be used, carefully document any changes that you make. This documentation can be of great assistance the next time the object needs to be adjusted.

Choose *Change* from the maintenance transactions of the individual ABAP Workbench tools to adjust objects *without the Modification Assistant*.

For more information, read [Upgrading from Release 4.5A or Less \[Page 168\]](#).

## Upgrading from Release 4.5A or Less

## Upgrading from Release 4.5A or Less

The first time that you upgrade from Release 4.5A or less to Release 4.6C, all objects that you have modified, as well as all objects delivered by SAP, IBU or an SAP partner for the first time, are displayed in transaction SPAU in the category *Without Modification Assistant*. Objects for which no version management is possible and which have never been previously managed using transaction SPAU are also displayed here.

Objects in the category *Without Modification Assistant* must be adjusted manually.

Transaction SPAU divides objects into three different categories:

- Objects where no version management is possible (without Modification Assistant)

Changes to these objects are displayed in the category *without Modification Assistant*. Information about adjusting these kinds of objects can be found in the documentation created by the last person who modified them. Transaction codes are examples of objects where no version management is possible and that are changed without Modification Assistant support.
- Objects where version management is possible (without Modification Assistant)

Changes to these objects are displayed in the category *without Modification Assistant*. However, version management can be used when adjusting them. ABAP Dictionary objects are examples of objects where version management is possible, but that are changed without Modification Assistant support.
- Objects where version management is possible (with Modification Assistant)

These objects initially appear in the category *Without Modification Assistant* when you upgrade to Release 4.6C. You then adjust them using version management or manually with the tools (the Modification Assistant's logging process is enabled at this time). From now on, these objects will be displayed in the category *With Modification Assistant*.

Objects that have been modified with the help of the Modification Assistant appear in the category *With Modification Assistant* at upgrade and may be adjusted in the manner described in the preceding sections.

In version management, the new version imported from SAP, an IBU or an SAP partner (the active 4.6C version) is called the *Version in the development database*. In the category *Version in the development database*, the active 4.6C version appears first and has exactly the same contents as the version in the development database. This version is not identified by a number, but instead with the text *Upgrade 4.6C*. The version after it is from the previous release and reflects the state of the system before upgrade. This version is identified by a number. Compare these two versions with each other in order to determine which changes you made in the past. Then adjust the objects using the appropriate maintenance transaction --- if possible in a second, parallel window.



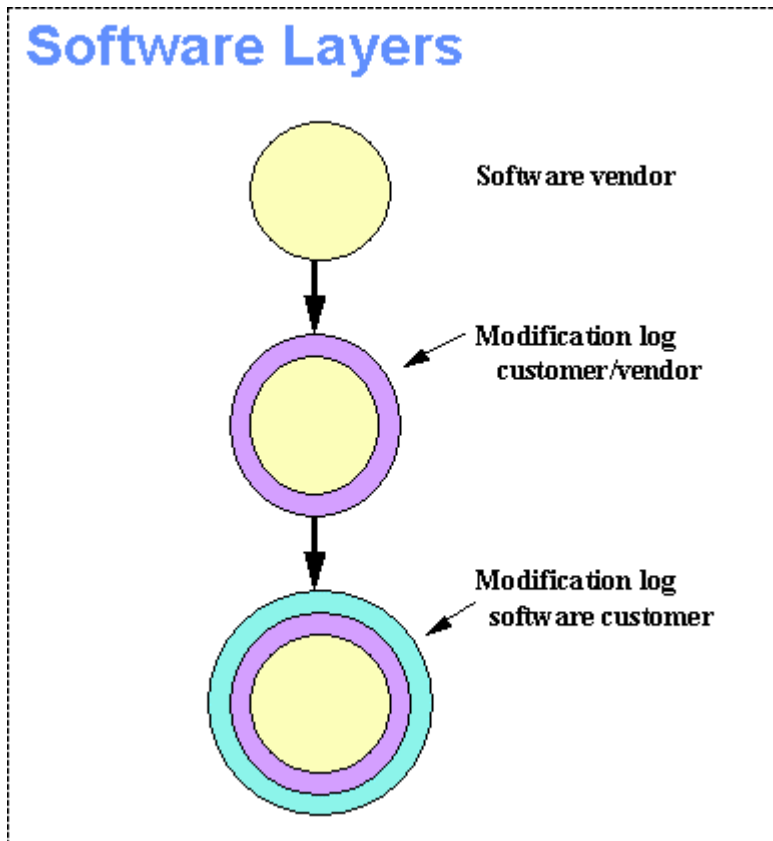
## Modifications Made by Several Different People

Often SAP Systems are not delivered directly to the “end customer”, but initially to a software vendor. This vendor makes modifications to the system and then delivers it to the customer, who subsequently makes another set of modifications. It is also conceivable, that a large company might implement important additional functions centrally for all of its sales offices and that these offices, in turn, might make their own modifications to the system in order to streamline specific work processes.

The Modification Assistant maintains a log of changes made for each system group (development systems, production systems, and so on). This means that a set of modification logs organized according to system group exists for the software vendor, as well as for the customer and for any set of other users in between. The original SAP System plus the modifications made to it (by the software vendor, for example) make up the original for the next generation of users (the customer, for example).

Thus it is possible for a customer’s system to contain several rounds of modifications by the time it is delivered.

The source of these layers of modifications should be visible in the naming norms and standard comments inserted into the system. The source of the current, uppermost layer of modifications can be displayed using the log function from the overview.



**Modifications Made by Several Different People**

Change requests and copies always include all modification data when they are transported.

For example, you are making modifications within a system group in a development system. All transportable change requests and copy transports will transport modification data into the next system (test systems, production systems, and so on). Hence, change requests and copy transports should only be imported into a system when you want to adopt the modification data from these requests as well. Adjustment support using the Modification Assistant is only available at upgrade or when applying a patch.

## Adjusting Other SAP Systems



The functions described below are only active for the upgrade and not when applying patches.

If you have a production and development system, the maintenance level is first imported into the development system where quality tests are performed. After testing, it is imported into the production system or other SAP Systems.

The process of adjusting objects in the production system is the same as in the development system. If your development system and production system have the same technical status, you can use a simplified adjustment procedure.

The same technical status means that:

- Before the upgrade was imported, the development system and the production system were running on the same version of a Release
- All objects modified in the development system are the same in the production system or will be transferred there.

If your systems meet the above requirements, you need not use the transactions SPDD/SPAU to adjust the second system by hand. You can simply transport the changes made from the development system to the production system. If you do this, you are not asked to run the transactions SPDD/SPAU during the upgrade, even if objects have been modified. In this way, the time required for upgrading the production system can be reduced.

A description of the adjustment procedure must take both the [upgrade of the development system \[Page 172\]](#) and the [upgrade of the production system \[Page 173\]](#) into account.

## Upgrading the Development System

## Upgrading the Development System

1. Perform the adjustment in the development system using the procedure described. Whenever possible, use the same change request to record your changes.

If this is not possible, read the section [Handling Change Requests During Adjustment](#) [Page 174].

2. In the Workbench Organizer, release all the tasks (corrections and repairs) that are assigned to the change request.



Do not release the change request, but indicate that it is intended for transport into another SAP System by choosing the *Select for transport* function.

Only mark one change request with the *Select for transport* function.

When you mark a second request, this deselects the first request. Only the last marked request is taken into account when other SAP Systems are adjusted.

The upgrade control program R3up exports the transport that you have marked in a later upgrade phase.

## Upgrading the Production System

During the test phase of the upgrade, the R3up upgrade control program checks whether there are any change requests registered for transport from the development system to the production system. If this is the case, R3up offers to import the transport automatically, instead of you carrying out adjustments with transactions SPDD/SPAU. If you choose this procedure, you still have the option of stopping SPDD/SPAU to check the changes accepted automatically before they are activated.

If you decide to import a transport automatically, a subsequent analysis checks whether there is an entry in the transport you specified for each modified object found in this SAP System. Only if this is the case is the transport marked as suitable.

If there are more modified objects in the production system than in the test system, they cannot be dealt with by the transport. This means that you have to adjust these objects.

If you had more modified objects in the test system than in the production system and these are also contained in the transport, you import these into the production system with the transport.

---

**Handling Change Requests During Adjustment**

## **Handling Change Requests During Adjustment**

When you carry out modification adjustment during and after an upgrade, the functions of the Workbench Organizer are active. During the adjustment after the upgrade, questions are frequently asked about the Workbench Organizer functions. Typical questions are:

- Which change request should I choose to record changes made during modification adjustment?
- What are local and transportable change requests?
- How are tasks (repairs and corrections) released when modification adjustment has been completed?
- How do I select a change request to transfer modification adjustment from the development system to other SAP Systems?

The following section enables you to carry out modification adjustment for the upgrade without having an in-depth knowledge of Workbench Organizer functions.

## Choosing a Change Request for Modifications

When you modify ABAP Dictionary objects (SPDD) or R/3 Repository objects (SPAU) during modification adjustment, a request query dialog box of the Workbench Organizer appears.

To create a change request, proceed as follows:

1. The first time you are prompted to enter a request in SPDD or SPAU, create a new request by choosing *Create request*.
2. In the dialog box which appears next, enter a meaningful short description and save the request.
3. When you choose *Continue* (green check mark), the object is included in the change request.

Generally, when you edit other objects, the same change request is automatically proposed in the request query dialog box. Then choose *Continue*.

If no request is suggested, choose *Own requests* to access an [overview \[Ext.\]](#) of your requests. Double-click on the request that is to be used for SPAU or SPDD.



If your SPDD or SPAU request is not proposed and does not appear in the overview with *Own requests*, you can create a second request.

**In the above case it is essential to refer to the section [Choosing a Change Request to Transfer Modification Adjustments to Other SAP Systems \[Page 178\]](#).**

However, this only occurs in certain exceptional situations.

---

**Local and Transportable Change Requests**

## Local and Transportable Change Requests

You can access the [request overview \[Ext.\]](#) in the Workbench Organizer by choosing *Own requests* in the request query dialog box or by choosing:

*Tools → ABAP Workbench → Overview → Workbench Organizer*

The request overview is displayed in the form of a hierarchical list and is organized according to the categories “Transportable” and/or “Local”. If you only have change requests belonging to one of the categories, only this category will appear.

Local change requests were introduced in Release 3.0. Unlike transportable change requests, local change requests are not transported (no export) when they are released.

The configuration of the transport routes in the SAP System determines whether changes to objects during modification adjustment are recorded in a local or transportable change request. You cannot and must not change these settings during the upgrade. In all cases, the change request can still be used for automatically transferring modifications to a subsequent system. For this procedure, the change requests are not released in the normal way, but handled specially.



Do not change the configuration of the transport routes in your system group during the upgrade.



## Releasing Tasks at the End of Modification Adjustment

When modification adjustment has been completed, release your tasks. The tasks are the repairs and/or corrections that were automatically assigned to you when the change request was created.

To do this, proceed as follows:

1. Choose *Tools* → *ABAP Workbench* → *Overview* → *Workbench Organizer* → *Display* (left half of the screen).

The request overview in the Workbench Organizer appears.

2. Click on the expansion symbol (+) before the relevant change request.

The change requests are displayed together with the tasks that are assigned to them.

3. Position the cursor on each of your tasks (normally only one per change request) and choose *Release*.

You are then requested to enter documentation.

If you have created more than one change request, you need to release the tasks for each of these requests.



Do not release the change request at this point, if you want to use it to automatically transfer the modification adjustments to subsequent systems.

During the adjustment of the ABAP Dictionary objects, releasing requests is locked until the edited objects have been activated. Do not release these requests until after the upgrade.

## Transferring Modification Adjustments to Other SAP Systems

## Transferring Modification Adjustments to Other SAP Systems

When you have completed modification adjustment, to automatically transfer modifications to a subsequent system use the *Select for transport* function in SPDD or SPAU.

A dialog box appears asking you to specify the change requests that contain your modifications.

If you have only created one change request for all modifications during the entire adjustment, enter its number here.



If your modifications are recorded in several change requests, you first have to place all entries in a single request. Several requests cannot be marked for transfer.

To do this, proceed as follows:

1. First call the request overview of the Workbench Organizer (transaction SE09):  
*Tools → ABAP Workbench → Overview → Workbench Organizer → Display* (left half of the screen).
2. Make sure that all tasks under the requests used during modification adjustment have been released.
3. Position the cursor on the request number of the selected request.
4. Choose *Request/task → Object list → Include objects*.
5. Select the option *Object list from request*, enter the number of another request used during adjustment in the field provided, and include its object list.
6. Repeat steps 4 and 5 until the object lists of all requests used during the adjustment have been included in the request selected in step 3.
7. Now enter the request in which the object lists were included in the SPDD or SPAU query window.

## Notes (Troubleshooting)

Before the upgrade:

- You are not sure whether all modifications to SAP objects were carried out with the system change option "All objects (with correction system)".

Check whether the objects you modified are recorded in repairs. To do this, you can use analysis report *Search for objects in requests/tasks* in transaction SE03.

- No repairs or customer transports are displayed by the Transport Organizer (SE01) and the Workbench Organizer (SE09), although you know that objects have been modified in the SAP System:

Indicate the objects that have been modified and differ from the SAP standard by including them in a repair and subsequently releasing it.

During the adjustment:

- No differences are found when you compare the second to last and third to last versions:

During the last upgrade you did not perform adjustments to retain changes you had made to objects.

Indicate that modifications to these objects are no longer needed with the *Reset to original* function.

- The last and second to last versions were created by user SAP(R3TRANS):

During the last upgrade you did not perform adjustments to retain changes you had made to objects.

Indicate that modifications to these objects are no longer needed with the *Reset to original* function.

---

The SAP Software Change Registration Procedure (SSCR)

## The SAP Software Change Registration Procedure (SSCR)

SSCR (**S**AP **S**oftware **C**hange **R**egistration) is a procedure, starting in Release 3.0A, for registering all manual changes to SAP source code and Dictionary objects. Tuning measures, such as the configuration of database indexes and buffers, are exempt from the registration process.

This procedure further improves the availability and reliability of R/3 production installations.

The procedure is designed to support and accelerate SAP's service and support functions.

At the same time, the procedure supports customer projects.



Modifications severely reduce SAP's warranty obligations concerning the software it delivers. Despite this reduced liability, SAP will continue to try and support customers who have modified their systems themselves.

[What Is Registered? \[Page 181\]](#)

[When Are Objects Registered? \[Page 182\]](#)

[How Do You Get a Key in the SAP Service System? \[Page 184\]](#)

[Who Can Use Registration? \[Page 189\]](#)

[Request for Expansion of Customer Exits \[Page 190\]](#)

[How Can Registration Take Place Without SAPNet Access? \[Page 188\]](#)

## What Is Registered?

The registration process involves every **development user** who creates, changes, or deletes objects (including customer objects) in your R/3 System. Each development user is registered once.

Independent of development user registration, the system also registers ABAP Repository objects.

However, it does not register which object was modified by which user. To trace this connection, use the Workbench Organizer.

## When Are Objects Registered?

## When Are Objects Registered?

### Registration Takes Place When:

- An SAP source is changed
- Manual changes are made to ABAP Dictionary elements
- Code from SAP notes is included

### Validity Period of the Registration

- Developer registration

Once a developer has been registered, the corresponding key is saved locally and accessed automatically during future changes. These keys remain valid indefinitely, even across release updates and upgrades.
- Object registration

Objects are registered according to release. Registered objects' keys become invalid after a release upgrade.

### How to Deal with Modifications at Upgrade

Registered objects' keys become invalid after a release upgrade.

To copy modifications into the new release, use the SAP Patch Manager (transaction SPAM). No new object keys are necessary.

If you have to modify objects again after a successful upgrade, you must request a new key for each object you want to alter in the current release.

### Registration Does Not Take Place for

- Database indexes
- Buffer settings
- Customer objects (objects not lying within the SAP namespace).
- Objects changed due to automatic generation (for example, from Customizing)
- Customer development implemented using exits



Use the following procedure when configuring database indexes and buffer settings to prevent the system from prompting you to enter a key:

1. Enter the name of the object you want to configure on the initial screen of the ABAP Dictionary. Choose *Display* and then *Goto → Technical settings or Indexes* on the next screen.
2. Choose *Change* from the application toolbar.

The system only requires key entry if change mode is selected directly.

### Automatic Registration

If SAP provides you with code for solving a problem in your system (for example, in the form of a note) the objects in question will be registered automatically. You will receive the necessary keys together with the note.

### Registering Modifications to SAP Objects

If you modify any SAP objects subject to registration, the system prompts you for the required key in a dialog box. Enter the SSCR key in the dialog box.

---

How Do You Get a Key in the Service System?

## How Do You Get a Key in the Service System?

[Registering a Development User \[Page 185\]](#)

[Registering an Object \[Page 186\]](#)

[Displaying an Overview \[Page 187\]](#)



## Registering a Development User

Each development user must request a key, and is thus registered automatically. The system prompts development users for their key the first time they attempt to change one of the objects listed above.

To register yourself as a developer, proceed as follows:

1. Logon to the SAPNet - R/3 frontend.

If you do not have an user for the SAPNet - R/3 frontend, please contact an employee at your company who does. This person can register you in SAPNet - R/3 frontend and will subsequently be logged as your initiator.

2. Choose *Registration*.

The screen for *Registering Changes to SAP Objects* is displayed.

3. Enter the R/3 user name of the developer you want to register.

The system displays a 20-digit key.

4. When the SAP System prompts you to enter this key, either enter the key manually or cut and paste it into the appropriate field.



From Release 4.6A, SSCR will also be available in the SAPNet - Web frontend by adding the extension /SSCR to your SAPNet address.

---

Registering an Object

## Registering an Object

Object registration takes place the first time an object is changed in your SAP System by a registered development user. If the same user changes the object again at a later time, you do not need to re-enter your key.

To register an object, proceed as follows:

1. Logon to the SAPNet - R/3 frontend.

If you do not have an user for the SAPNet - R/3 frontend, please contact an employee at your company who does. This person can register you in SAPNet - R/3 frontend and will subsequently be logged as your initiator.

2. Choose *Registration*.

The screen for *Registering Changes to SAP Objects* is displayed.

3. Choose *Object* to change from development user registration to object registration.
4. Select the appropriate checkbox if you are making advance corrections.
5. Enter the object type and the name of the object. This information is displayed in the Change and Transport Organizer any time you attempt to change an object.

The system displays a 20-digit key.

6. When the SAP System prompts you to enter this key, either enter the key manually or cut and paste it into the appropriate field.

## Displaying an Overview

To display an overview of all registered objects in your system and their initiators, choose *Overview* on the *Changes to SAP Objects* screen.

Double-click on a line in the list to display additional information (such as the developer's key, for example).

---

How Can Registration Take Place Without SAPNet Access?

## How Can Registration Take Place Without SAPNet Access?

In exceptional cases - for example, if the SAPNet - R/3 frontend connection of the customer is down - the **SAP Local Support Center** can carry out registration for you.

## Who Can Use Registration?

Each user of the SAPNet -R/3 frontend can register objects and development users except if their authorization to do so has been revoked by the system administrator.

Define one of your users as an administrator. This user can then revoke administrator authorization for all other SAPNet - R/3 frontend users (authorization is granted automatically when a user is created).

To revoke an authorization, proceed as follows:

1. From the SAPNet - R/3 frontend screen *Process Customer Messages*, choose *Goto* → *Administration*.
2. To revoke a user's administrator authorization, deselect *Administrator*; to revoke authorization to register objects or development users, deselect *Object registration*.

---

**Request for Expansion of Customer Exits**

## **Request for Expansion of Customer Exits**

If you would like to have additional functionality at certain places in the SAP System, enter a problem message in SAPNet in which you name the specific application. SAP will process your request and decide whether the functions you desire can be implemented in the next functional release and whether or not it makes sense to provide a customer exit at that point in the source code.