BC470

Form Printing With SAP Smart Forms

mySAP Technology

Date _____

Training Center _____

Instructors _____

_____

Education Website _____

Participant Handbook

Course Version: 2003 Q2
Course Duration: 2 Day(s)
Material Number: 50066260

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

Copyright

Trademarks

Disclaimer

About This Handbook

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

Typographic Conventions

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

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

Icons in Body Text

The following icons are used in this handbook.

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

Contents

Course Overview

In this course, you will learn how to modify and create forms using the three main tools of SAP Smart Forms:

- SAP Form Builder
- Style Builder
- Text Module Maintenance tool

You will learn how to incorporate different node types, such as text, tables, and graphics into forms. You will also learn to use variables for database communication. You will understand how SAP Smart Forms are integrated into application programs.

Target Audience

This course is intended for the following audiences:

- Project team members, developers, and consultants responsible for form printing

Course Prerequisites

Required Knowledge

- BC400 (ABAP Workbench: Basics)
- ABAP programming knowledge

Recommended Knowledge

- Standard Level 2 application training courses are an advantage

Course Goals

This course will prepare you to:

- Understand SAP Smart Forms and the interaction between the graphic tools
- Create and maintain SAP Smart Forms
- Integrate SAP Smart Forms into application programs
- Use Smart Styles

Course Objectives

After completing this course, you will be able to:

- Create and change SAP Smart Forms using SAP Form Builder tools
- Create and change form styles using Style Builder
- Explain the technical implementation of forms and their integration into application programs

SAP Software Component Information

The information in this course pertains to the following SAP Software Components and releases:

Unit 1

SAP Smart Forms

Unit Overview

In this unit, you will learn about the areas of application of SAP Smart Forms. You will also get an overview of integration of the generated function module into application programs.

Unit Objectives

After completing this unit, you will be able to:

- List the areas of application of SAP Smart Forms
- Explain the basics of creating documents

Unit Contents

Lesson: Overview of SAP Smart Forms

Lesson Overview

In this lesson, you will learn about the areas of application of SAP Smart Forms. In addition, you will learn about basics for creating documents and forms.

Lesson Objectives

After completing this lesson, you will be able to:

- List the areas of application of SAP Smart Forms
- Explain the basics of creating documents

Business Example

The Fly & Smile travel agency constitutes a number of independent units located in different parts of the world. Each unit is headed by a business unit head and operates as an independent entity under the common parent company. You are responsible for sending invoices of the booked flights to the respective customers. To create invoices using SAP Smart Forms, you want to learn about the areas where SAP Smart Forms can be used. You also need to get an overview of how these forms are created.

Application Areas of SAP Smart Forms

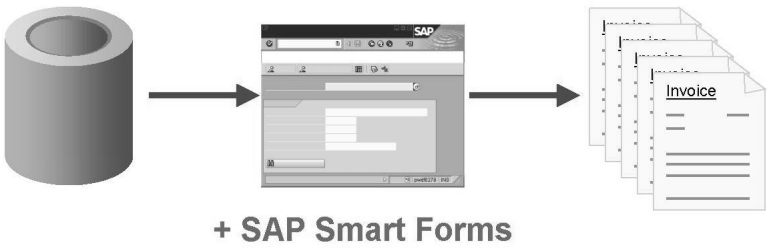


Figure 1: Form Printing

Every company regularly needs to print large numbers of documents with a consistent design, such as invoices or delivery notes. To do this, the companies must use theirbusiness application software. Documents can be output to a printer, a fax device, or as e-mails.

As of SAP R/3 4.6C, SAP provides a tool for form processing - **SAP Smart Forms**. This tool includes utilities for designing forms and defining the interface to the application programs that use forms for their data output.

With the SAP Web Application Server (WAS), SAP Smart Forms can also be generated in the HTML format and made available in the intranet or Internet for user entries.

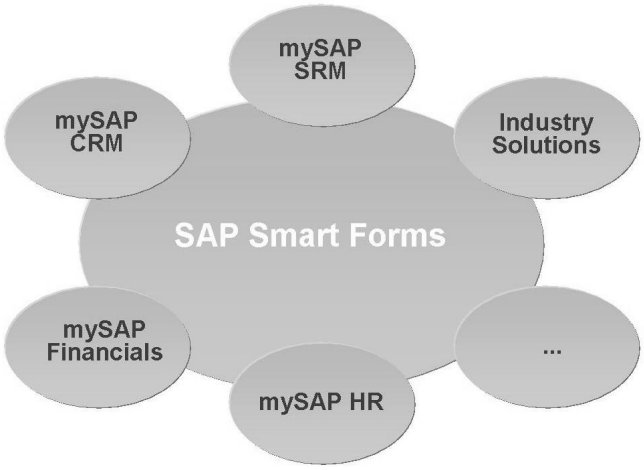


Figure 2: SAP Smart Forms in Applications

In addition to the tool itself, the mySAP.com solutions comes with a number of forms for central business processes. These include forms for industry solutions, mySAP Customer Relationship Management (CRM), mySAP Supplier Relationship Management (SRM), mySAP Financials, and mySAP HR.

For an overview of the forms available and their integration in applications, see the SAP note 430621. This note is continually updated and does not therefore claim to be exhaustive.

Note: This training course does **not** teach you any **application-specific knowledge**. You rather learn the basics of form development, which will enable you to make changes to all forms and/or application programs once the course is completed.

Advantages of SAP Smart Forms

The advantages of SAP Smart Forms can be listed out in brief as:

- Minimum time required for creating and maintaining forms
- Form customizing without programming knowledge, thanks to complete graphical user interface
- Integration into SAP products
- High performance when printing in large quantities
- Link to a transport system
- Platform independence
- Multilingual capacity

Since SAP Smart Forms are directly integrated into the applications, no time is lost due to data exchange with external systems. In addition, documents can be automatically created in the background, which is useful in case of extensive dunning runs.

SAP Smart Forms objects are linked to the transport system so that you can easily test your forms and subsequently transfer them to your production system.

SAP Smart Forms can be used on all platforms supported by SAP. To edit the forms, you need the graphical user interface SAP GUI for Windows.

SAP Smart Forms provides multiple language support. You need to define the layout of a form only once and can translate the texts using the translation tools. This ensures that your documents have a consistent design on an international level.

Difference Between Smart Forms and SAPscripts

The advantages of SAP Smart Forms as compared to SAPscript are:

- Minimum time required for form customizing
- Consistent use of graphical tools
- Separation of data retrieval and form logic
- No special scripting language
- Integration in Internet applications
- Migration of SAPscript forms and styles is supported
- SAPscript texts can be inserted into SAP Smart Forms

Compared to SAPscript, which is a classic form processing tool, SAP Smart Forms provide considerable advantages:

Customizing forms is considerably easier as SAP Smart Forms offer more functions, such as you can create tables with the Table Painter of SAP Smart Forms. Also, the interface between the forms and the application programs has been redesigned.

Specific tasks such as retrieving additional data within forms, do not require you to use special scripting language commands as you had to with SAPscript. You can also insert normal ABAP code.

As of SAP Web Application Server 6.10, SAP Smart Forms can be output in HTML.

The migration of SAPscript forms and styles to SAP Smart Forms is supported. You can also convert SAPscript styles into Smart Styles.

SAPscript texts can be used directly in SAP Smart Forms.

Depending on the application, either SAP Smart Forms or SAPscript forms can be used. In some cases, you can choose which you want to use. This means that SAP will provide continued support and maintenance for SAPscript forms.

Creating Documents

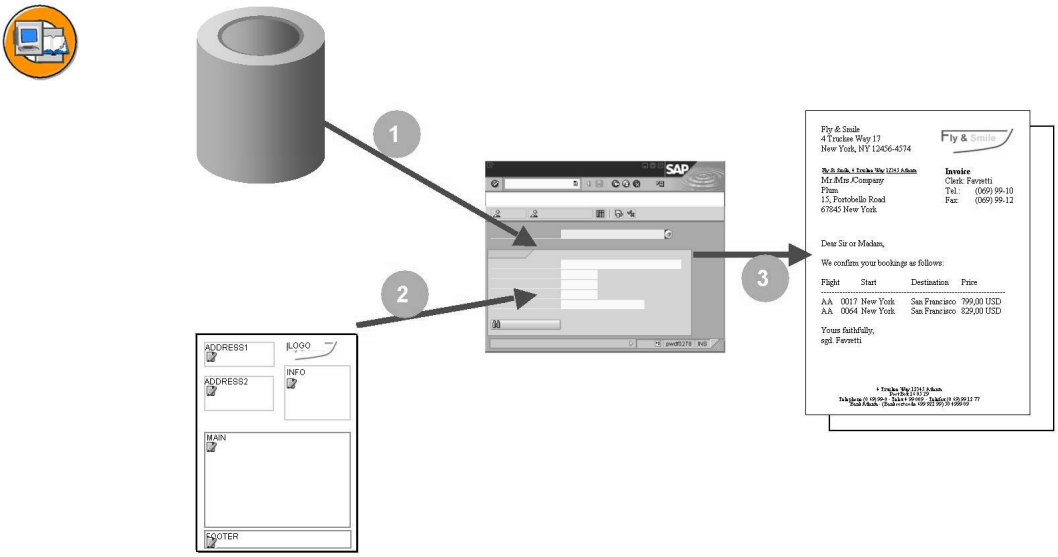


Figure 3: Creating Documents: Simple View

To allow an application program to output a document, two basic steps are required:

1. Data retrieval: The data retrieval step is the central element of the program and can have any degree of complexity with user interactions.
2. Starting form processing: During this step, the data read is written into the form.

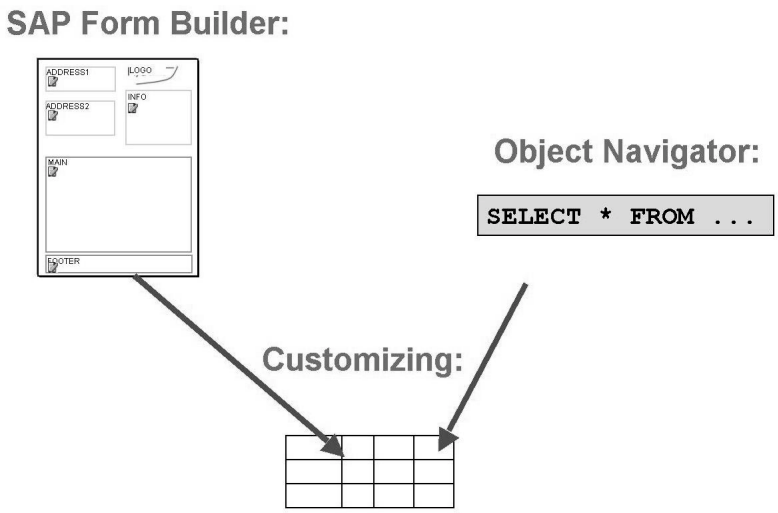


Figure 4: Tools for SAP Smart Forms

Maintaining SAP Smart Forms comprises the following tasks:

1. Customizing the form: This is your main task.
2. Customizing the application program: This is only necessary in special cases, such as if you want to modify spool settings or read additional data that is to be output in all documents of a printing operation.
3. Saving the changes in Customizing.

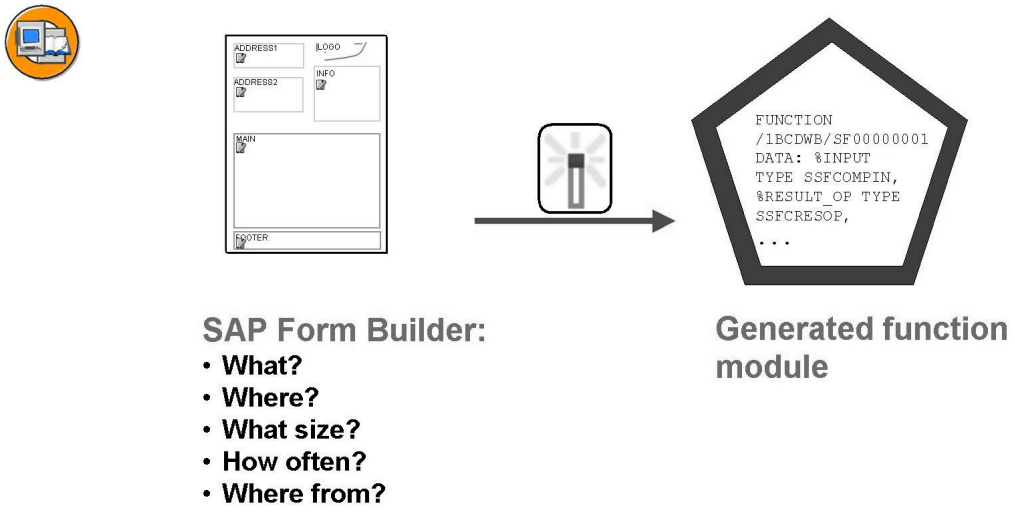


Figure 5: Creation of SAP Smart Forms

You use the SAP Form Builder to create and adjust forms (transaction: SMARTFORMS). There you define the layout, such as the position and the size of texts or graphics, the processing sequence of the form elements, and the interface, that is, the application data you want to output in the form.

After making necessary adjustments, you must activate the form. During this process, the system first checks if the form contains errors and saves it automatically. The main step is the generation of a function module. A function module is an encapsulated piece of ABAP code that can be compared to a subroutine. The interface of the function module is the same as the one you defined for the form in the Form Builder. Since the function module is generated automatically, no ABAP knowledge is required.

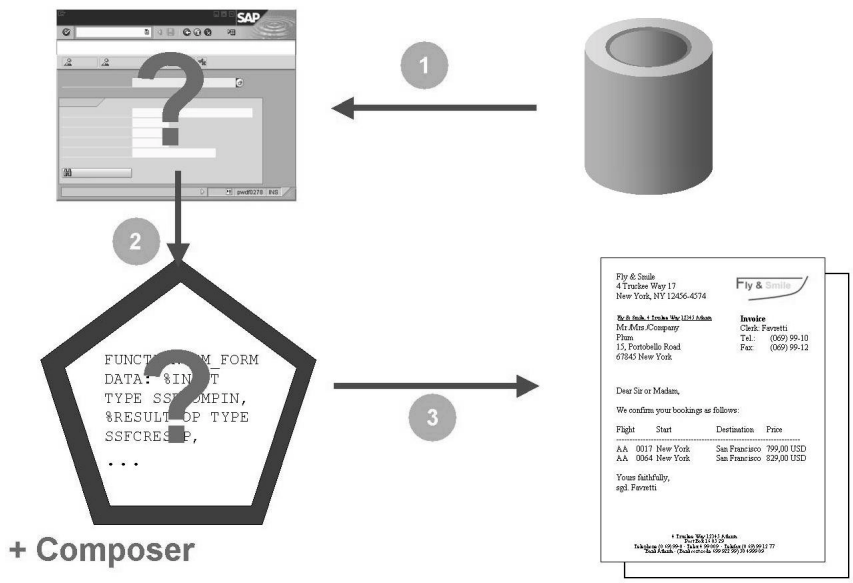


Figure 6: Creating Documents: More Complex View

The processes involved in document creation are now explained in more detail:

1. The transaction checks in Customizing which program to call. This program then reads the data.

2. The transaction checks in Customizing which SAP Smart Form to use for the scenario chosen, calls the appropriate function module generated, and triggers the form processing process. The interface is filled with the data read.

3. When the form processing process is started, the form processor, that is, Composer is automatically called in the background. The composer is responsible for formatting the texts according to the layout information stored in the form, filling fields with values at runtime, controlling the page breaks, and placing the completed document in the spool.

Lesson Summary

You should now be able to:

- List the areas of application of SAP Smart Forms
- Explain the basics of creating documents

Unit Summary

You should now be able to:

- List the areas of application of SAP Smart Forms
- Explain the basics of creating documents

Test Your Knowledge

1. Identify the forms that come with mySAP.com solutions for central business processes.

2. What is a function module?

3. The _____ is responsible for formatting the text according to the layout information stored in the form.

   Fill in the blanks to complete the sentence.

Answers

1. Identify the forms that come with mySAP.com solutions for central business processes.

   Answer: The forms that come with mySAP.com solutions include Industry solutions, mySAP Customer Relationship Management (CRM), mySAP Supplier Relationship Management (SRM), mySAP Financials, and mySAP HR.

2. What is a function module?

   Answer: A function module is an encapsulated piece of ABAP code.

3. The composer is responsible for formatting the text according to the layout information stored in the form.

   Answer: composer

Unit 2

SAP Form Builder

Unit Overview

In this unit, you will learn about tree, maintenance screen, and Form Painter. You will understand the form attributes, pages and their processing, windows, background graphics, and predecessors and sequence of node processing in successors and subnodes. You will also learn how to copy, save, check, activate, test, rename, and create forms.

Unit Objectives

After completing this unit, you will be able to:

- Use SAP Form Builder to edit SAP Smart Forms
- Identify the attributes of an SAP Smart Form
- Identify and create pages in forms and documents
- Define page structure and page attributes
- Explain the different types of windows
- Identify the use of the Form Painter
- Activate and test forms

Unit Contents

Lesson: Creating and Maintaining SAP Smart Forms

Lesson Overview

In this lesson, you will learn about using SAP Smart Form Builder for editing SAP Smart Forms. You will also be able to identify the attributes of an SAP Smart Form and its output options.

Lesson Objectives

After completing this lesson, you will be able to:

- Use SAP Form Builder to edit SAP Smart Forms
- Identify the attributes of an SAP Smart Form

Business Example

The Fly & Smile travel agency constitutes a number of independent units located in different parts of the world. Each unit is headed by a business unit head and operates as an independent entity under the common parent company. You are responsible for creating invoices, which is a step-wise procedure, of the booked flights for the respective customers.

You need to know how to create and modify tables, containing customer information and booking details, in SAP Smart Forms and how to modify these forms as per your requirement. The data from these forms is then used for creating invoices.

Creating SAP Smart Forms

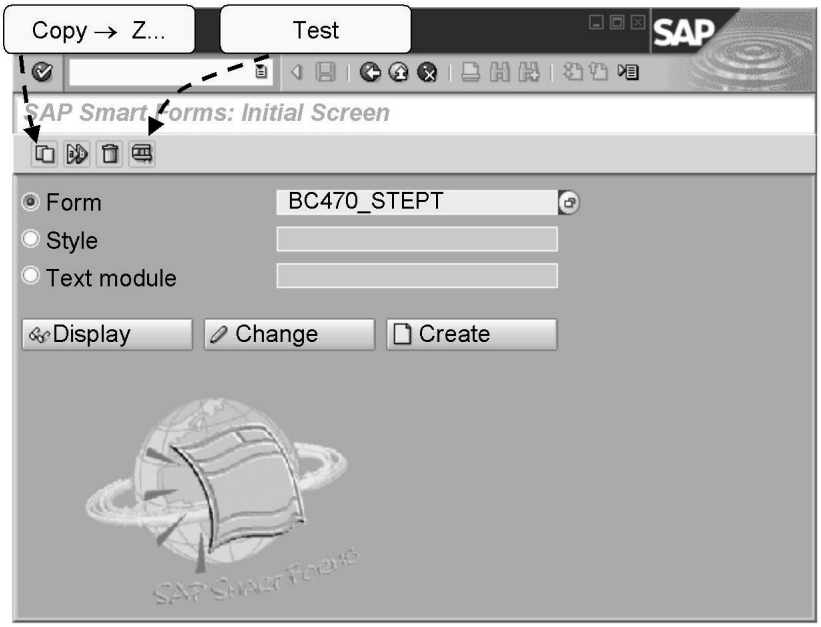


Figure 7: Initial Screen: Maintaining Forms

To call the initial screen of the SAP Smart Forms maintenance transaction, either enter *SMARTFORMS* into the OK code field, or choose *Tools → Form Printout → Smart Forms.* Then, choose the type of SAP Smart Form objects you want to edit by selecting the relevant radio button.

- Forms
- Styles
- Text modules

To work on a form, choose the *Form* radio button and enter the name of the form.

You can create, display, and change forms. The system takes you to the graphical editing tool, which is called the SAP Form Builder.

Never change the original SAP forms to prevent your modifications from being lost during the next upgrade. Instead, copy the original form into your customer namespace, starting with Y or Z, and modify the form copied as required.

You can also rename, delete, or test forms. To do this, choose the relevant pushbuttons or the options in the *Smart Forms* menu. Testing a form from the initial screen requires that it is activated in advance in the SAP Form Builder. You can also make settings specific to the SAP Form Builder. You can create SAP Smart Forms based on existing SAPscript forms by choosing *Utilities → Migrate SAPscript Form*.

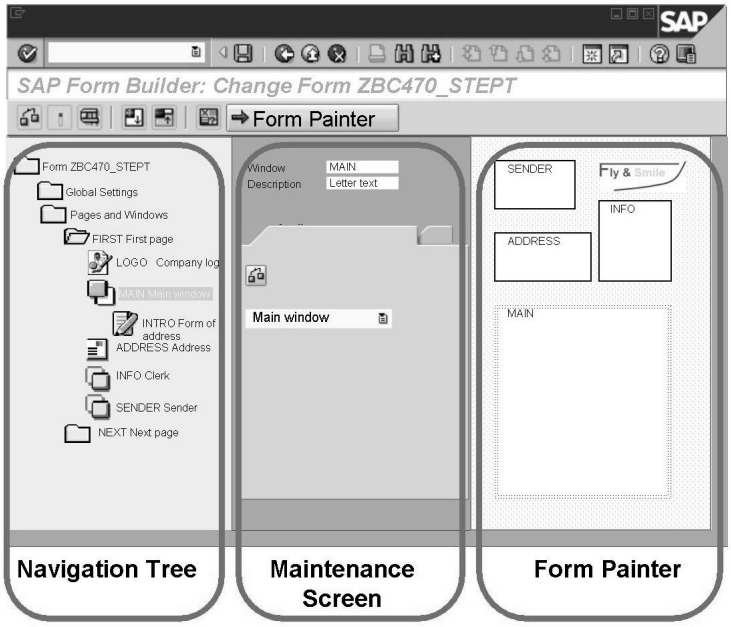


Figure 8: Areas of the SAP Form Builder

You edit forms using the graphical SAP Form Builder.

The SAP Form Builder is divided into three areas:

- On the left-hand side is the **navigation tree**. This tree graphically displays the hierarchy of the SAP Smart Form. The individual form elements, such as pages or graphics, are represented by nodes. You can also display the field list with variables below the navigation tree.
- In the middle is the **maintenance screen**. This screen has several tabs on which you set and change the attributes of the node currently selected. You can also enter text with the editor or use the Table Painter to determine the layout of a table. Any error messages occurring when the form is checked are displayed at the bottom of the maintenance screen.
- On the right-hand side is the **Form Painter**. The Form Painter is used to define the layout of a page, such as the position and size of text windows and graphics. You can hide the Form Painter by choosing *Utilities → Form Painter on/off*.

You can select nodes for editing by double-clicking these in the navigation tree or the Form Painter.

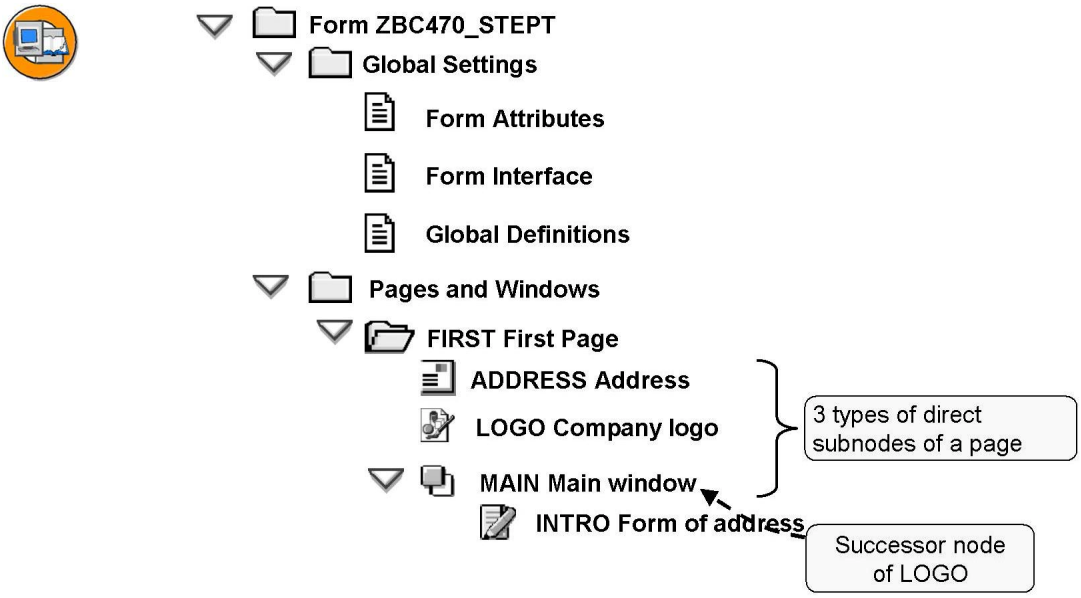


Figure 9: The Navigation Tree

All elements of a form are represented by a specific node in the navigation tree.

Subnodes "inherit" the attributes of higher-level nodes, such as the style. If a node is not processed, neither are its subnodes. A **successor node** of a node is independent. It is processed after the predecessor node.

If a node has subnodes, you can expand its structure by clicking the triangle symbol beside the node icon. You can select a node for editing by double-clicking it. The system displays the node in the maintenance screen and in the Form Painter, provided the Form Painter is switched on.

Below the top node, you always find the following two nodes:

- Global Settings. These include:
  - Form Attributes: These can be administrative information and basic formatting settings.
  - Form Interface: Here, you must define the fields to be filled by the application program or returned to the application program.
  - Global Definitions: Here, you can define additional fields to be used in the form.
- Pages and Windows

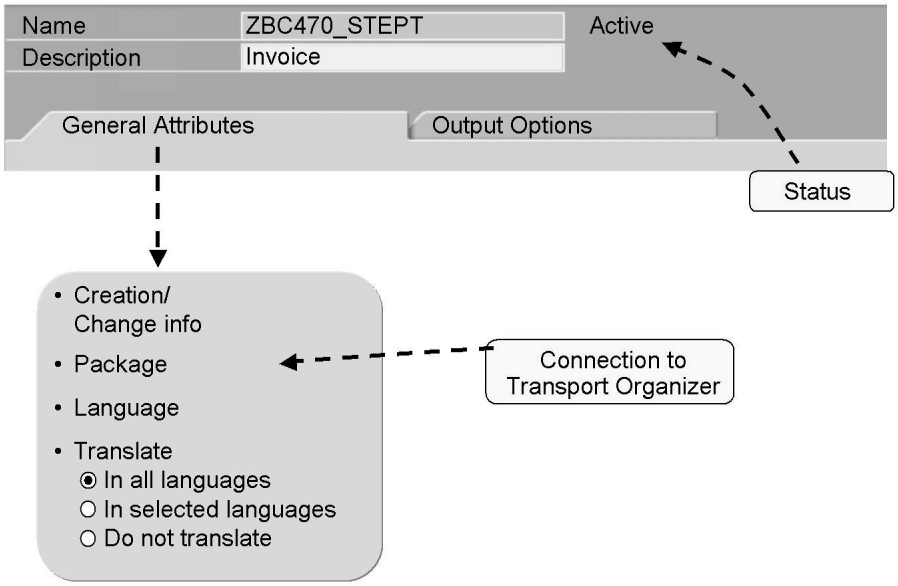


Figure 10: Form Attributes: General Attributes

The form attributes include not only the name and description of the form but also its current status: *Active* or *inactive*. A form can exist in any of these two versions. Application programs always use the active version. This means that you can provisionally save your changes without directly affecting application processing. To activate a form, choose *Form → Activate*. Note that when you copy a form, the status of the copy is always set to *inactive*.

Since SAP Smart Forms are connected to the transport system of SAP R/3 Enterprise, they have to be assigned to a package (SAP R/3 4.6C: development class). You do this when you first save your form. You can subsequently change the package assigned from within the SAP Smart Forms initial screen by choosing *Goto → Object Directory Entry.* Be sure to use the customer namespace that begins with Y or Z.

Each form has an original language. The *General Attributes* tab allows you to determine if you want to translate the form into other languages, and if so, which ones.

As of SAP Web Application Server 6.20, you can indicate in the *Restricted Languages* field that the logon language is not to be used for missing texts.

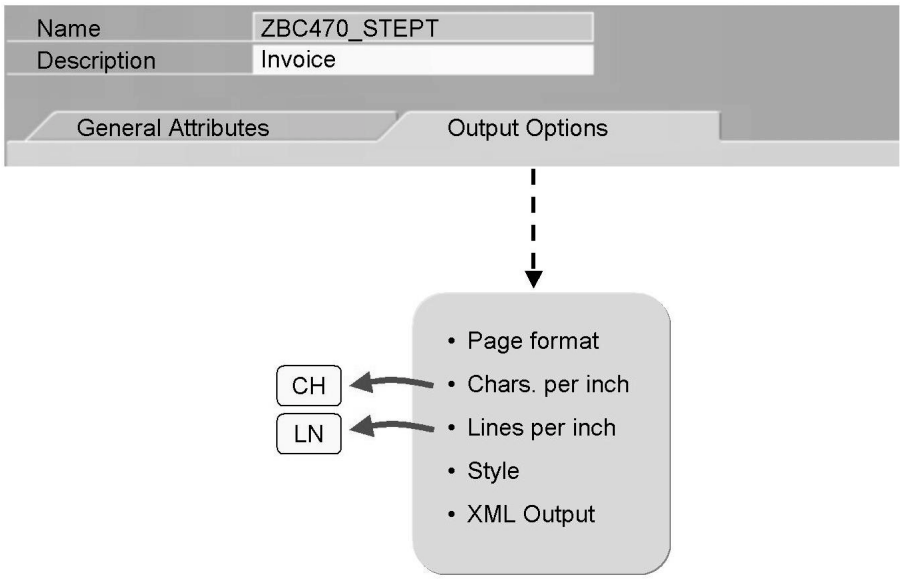


Figure 11: Form Attributes: Output Options

The page formats available include the page formats provided in spool administration. The orientation, whether portrait or landscape, is set individually for each page.

You can specify a default page format for new forms. To do this, choose *Utilities* → *Settings* → *General* tab from the initial screen of transaction SMARTFORMS.

You must assign a style to each form. A style is a collection of different character and paragraph formats, which are then used in the form. However, you can specify a separate style for most subnodes, which then overrides the default setting of the form.

Characters per inch (CPI). This field allows you to determine the *CH* unit of measure that you can use for horizontal length specifications, such as window widths, in the form. If you enter the default value 10, 1 CH is equivalent to one tenth of an inch, that is, approximately 2.5 mm.

Similarly, the *Lines per inch* field allows you to determine the *LN* unit of measure that you can use for vertical length specifications, such as window lengths, in the form.

The standard output format for printing is OTF (Output Text Format). A certified XML interface is provided, called the SAP Smart Forms XML Interface (XSF). The XSF data stream does not contain formatting specifications.

As of SAP Web AS 6.10, HTML output is also possible, for example for Web forms. The XDF format, a special XML format that also contains specifications for the dictionary types used for the interface parameters, can also be used.

Lesson Summary

You should now be able to:

- Use SAP Form Builder to edit SAP Smart Forms
- Identify the attributes of an SAP Smart Form

Lesson: Overview of Form Elements

Lesson Overview

In this lesson, you will be able to identify and create pages in forms and documents. You will learn to define the structure and the attributes of pages. In addition, you will learn about the different window types and identify the uses of the Form Painter. Finally, you will be able to activate and test an SAP Smart Form.

Lesson Objectives

After completing this lesson, you will be able to:

- Identify and create pages in forms and documents
- Define page structure and page attributes
- Explain the different types of windows
- Identify the use of the Form Painter
- Activate and test forms

Business Example

The Fly & Smile travel agency constitutes of a number of independent units located in different parts of the world. Each unit is headed by a business unit head and operates as an independent entity under the common parent company. You are responsible for creating invoices, which is a step-wise procedure, of the booked flights for the respective customers.

You need to know how to work with SAP Smart Forms so that you can enter the relevant data, such as customer information and booking details in your invoice forms and customize them as per your requirements.

Creating Pages

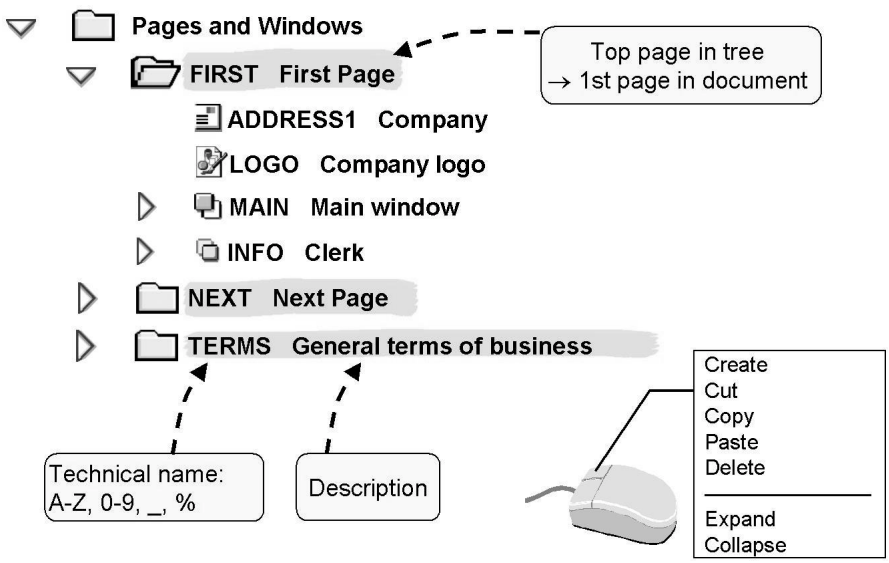


Figure 12: Pages

Each form consists of at least one page.

A page is represented by a node in the navigation tree. As with any other node types, such as texts or tables, right-clicking the mouse on an existing page opens a context menu with the available options:

- Create or delete in change mode only. When you create a new page, the system proposes a unique technical name, which you can change if required. Note that when you delete a node, all subnodes on the respective page are also deleted.
- Copy to clipboard, cut and insert into clipboard, paste from clipboard. All subnodes are also affected.
- Expand or collapse the page in the tree.

As of SAP Web AS 6.10, the keyboard can also be used for actions in the navigation tree: Delete, clipboard (Ctrl+C, Ctrl+V, and Ctrl+X), expand (arrow pointing right) and collapse (arrow pointing left), first and last node (Pos1 and End), page up/down.

You can also call these functions from the menu by choosing *Edit* → *Node* or *Subtree*. Each page, like all subnodes, has a technical name and a description. The following naming conventions apply: Only letters (without umlauts), numbers and underscores are permitted. The first character must be a letter. As a special case, the percentage sign is allowed as the first character. The percentage sign is used by the SAP Form Builder to generate names automatically.

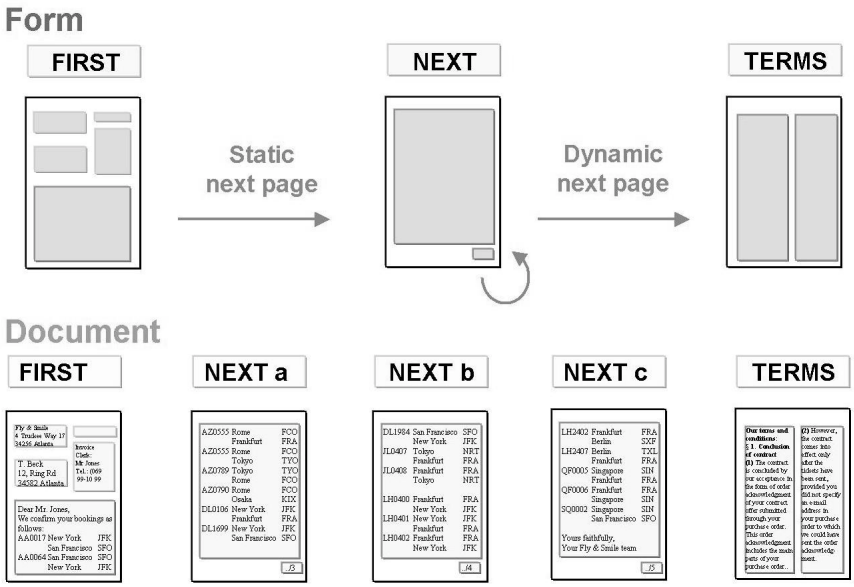


Figure 13: Pages in Forms and Documents

You can define one or more pages with different layouts for a form.

The top page of the navigation tree is processed first. You control the processing sequence by specifying the next page on the *General Attributes* tab for the page, which is processed automatically after the top page. Alternatively, you can have the next page determined dynamically based on conditions. For example, you can process a page with line items as many times as needed to output all data records and force the system to change to the page containing the general terms and conditions of business.

Depending on the amount of data to be processed, a form page can be used more than once in a document.

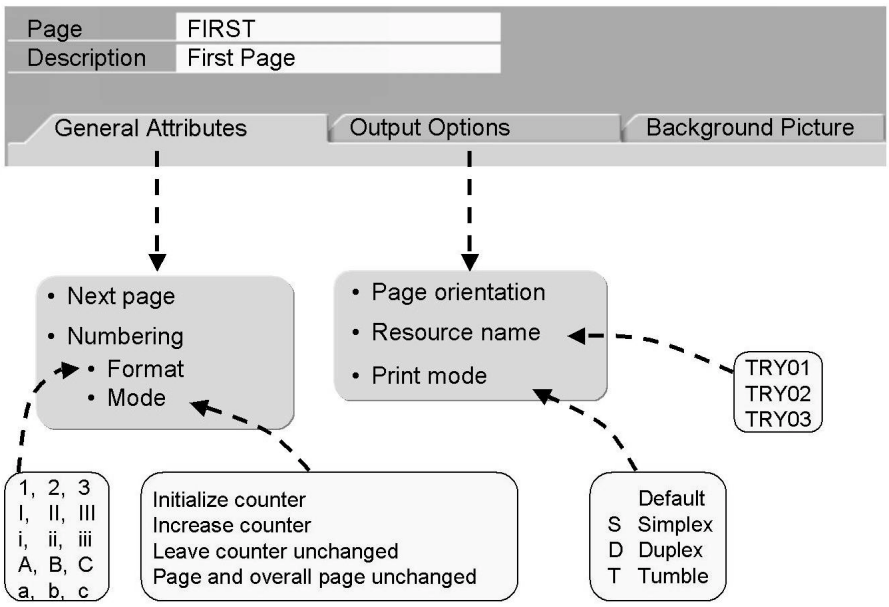


Figure 14: Page Attributes

You can make settings on the following tabs:

General Attributes

- The next page. The default value is the page itself.
- The type of automatic page numbering. You can choose between Roman and Arabic digits and uppercase and lowercase format and determine the behavior of the page counter. Note that if you make page number settings, this does not mean that the pages are automatically numbered. To have your pages numbered, you must output the variable SFSY-PAGE in a text window.

Output Options

- While the page format you specify applies to the entire form, you set the orientation (portrait or landscape) at the page level only.
- You can assign different paper trays to pages - provided your printer supports this feature. This makes sense in cases, such as if you want to use your company letter paper for the first page of a form and normal typing paper for all other pages.
- You can set double-sided print mode. Prerequisite: Your printer supports this feature.

Background Picture Prerequisite: The required picture already exists in the system after import using the transaction SE78. Text is printed over the picture.

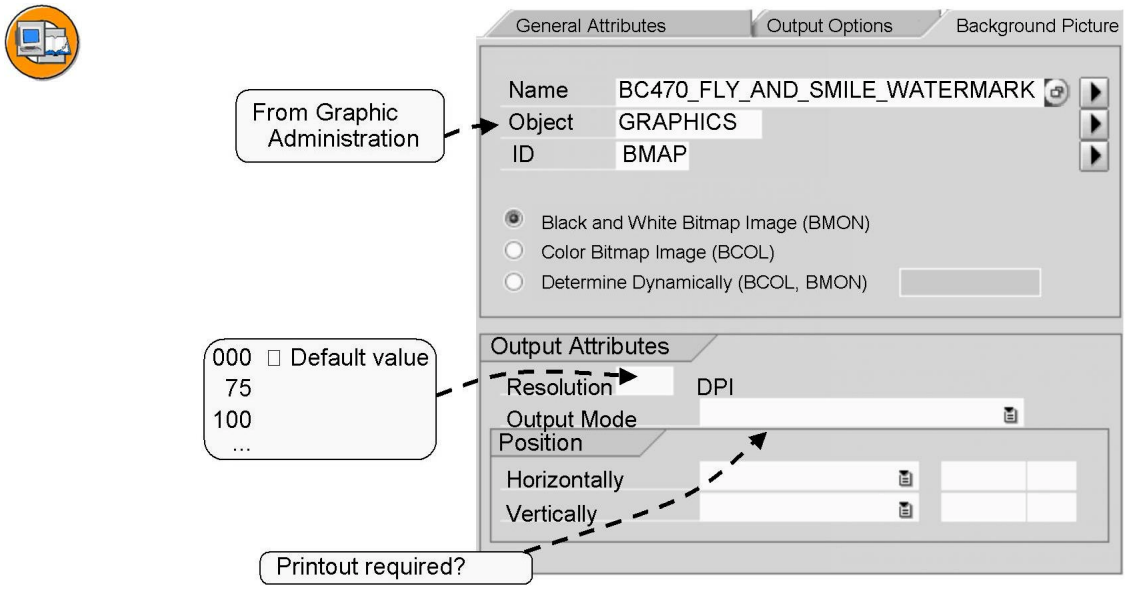


Figure 15: Background Picture

Specify the name of the graphic and its description, that is, object and ID. The default values are GRAPHICS and BMAP. The F4 Help lists the graphics that are available in the system.

Determine whether it is a black and white picture or a color picture.

Note: You can also select the background picture dynamically by entering fields instead of static names, which must have a value assigned to them at runtime.

Under *Output attributes*, you can specify the *resolution* in dpi (dots per inch). The smaller the value, the larger the graphic is displayed in the form. If you leave the field blank or enter *000*, the default setting of the graphic is used.

The *output mode* allows you to determine whether the background picture should only be displayed in the Form Builder. This makes sense if you redraw a scanned form that you do not want to print or whether the picture should be included in the actual form processing process. You can choose between *Print preview* or *Print preview and print*. You can decide from within the print preview, before the request is sent to the spool whether you want to print the background picture or not.

You must also determine the horizontal and the vertical position of the background picture with regard to the page border.

Update the preview of the Form Painter by choosing *Enter* in the maintenance screen.

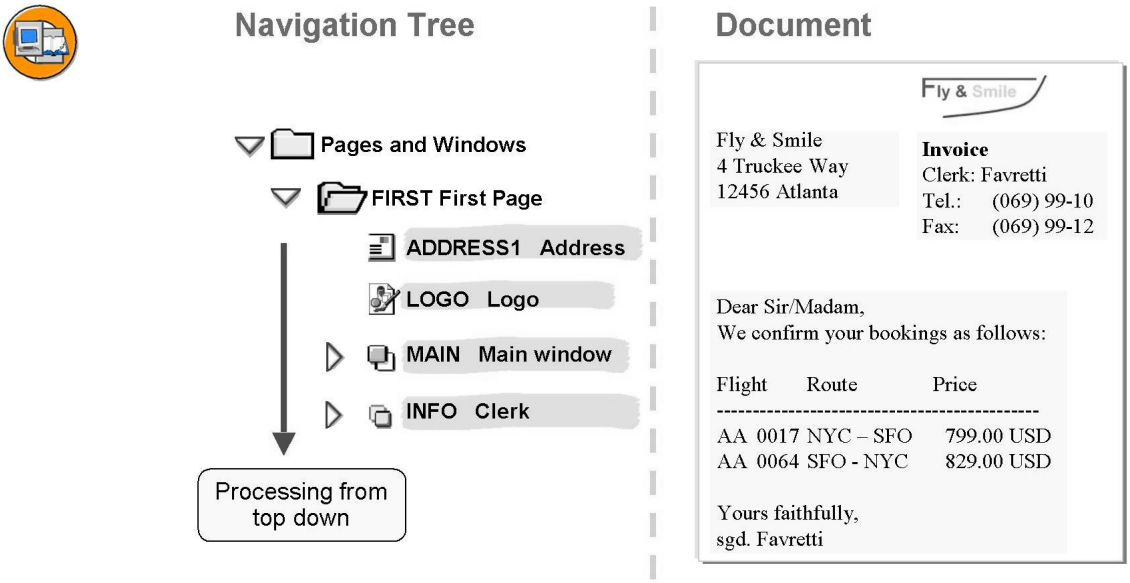


Figure 16: Page Structure with Output Areas I

To output information in a form, you must create suitable output areas, that is, windows on the relevant page. The following output areas are available:

Windows: Subnodes of windows are used to output text and data.

Address windows: If the application program uses Business Address Services (BAS), you can easily output formatted addresses in address nodes. Note that BAS was called *Central Address Management* (*CAM*) in SAP R/3 4.6C.

Graphic windows: The output areas of a form are represented as nodes in the navigation tree. The icon helps you to identify the three different node types, such as address, graphic, or window.

The order of page subnodes in the navigation tree does not affect their position in the form, except their processing. They are processed from top to bottom for each page. It is useful to imagine that all nodes are expanded. If necessary, you must move subnodes using Drag and Drop using the left mouse button. The processing sequence is particularly important if you use fields, that is, variables that are only filled at runtime.

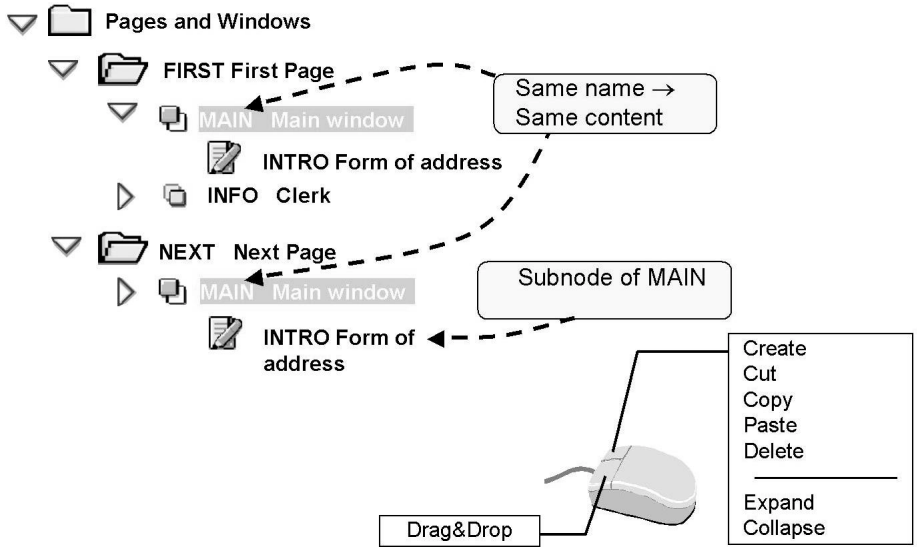


Figure 17: Page Structure with Output Areas II

You create output areas in the same way as you create any other node: from the context menu (right mouse click on a page). The system proposes a unique technical name which you can change if required.

You can use Drag and Drop to move using the left mouse button or copy (Ctrl key and left mouse button) subtrees, including nodes with subnodes. Alternatively, you can use the clipboard using the right mouse button: *cut, copy, and paste*. For example, you can move or copy windows or text nodes between pages.

As of SAP Web Application Server 6.10, you can also use the keyboard to copy (Ctrl+C), cut (Ctrl+X), and paste (Ctrl+V) nodes.

If you drag node A onto node B, node A is inserted after node B. In some cases, you can also insert the node A to be moved as a subnode of node B. The system then displays a dialog box where you can choose to insert the node *below the node...* or *after the node...* If you choose the second option, the node to be moved is inserted at the same level as node B but after it.

If you place output areas on **several pages** of a form, all changes made to the node contents (including the deletion of subnodes) affect all pages since the technical names of the nodes are identical. Although the output areas have the same contents, their position might differ from page to page.

If you place output areas several times **on the same page**, the system creates copies with the same contents but with different technical names than the originals. Changes to the node contents therefore affect only the relevant area.

Creating Windows

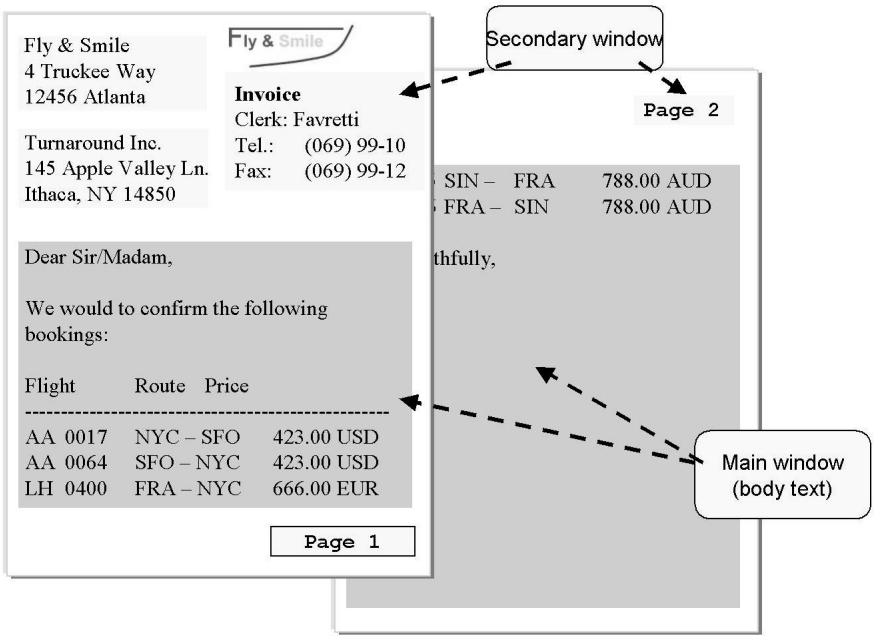


Figure 18: Main Window and Secondary Windows

There are two types of windows, main and secondary.

In the subnodes of the **main window**, you enter content that may span over several pages (called the body text), such as the bookings of a customer. As soon as the main window is filled with content, all secondary windows on the page that have not yet been processed are processed. There is an automatic page break on the next page. The nodes are processed again in the sequence in which they are arranged in the navigation tree and the body text is continued in the main window.

You can only define one window in the form as the main window.

The main window must have the same width on each page. You can choose the height and position.

A page without a main window might not refer to itself as the next page since this would cause an endless loop.

In the subnodes of a **secondary window**, you output text and data in a predefined output area. This means that the text is not displayed as a body text with page breaks.

Text that does not fit into the secondary window is truncated and not output.

The height, width, and position of a secondary window might be different for each page.

Graphics are automatically set to the correct size.

As of SAP Web AS 6.10, there are two window types: *Copy windows* and *Final Windows*. These behave in the same way as secondary windows with regard to the body text, size, and position.

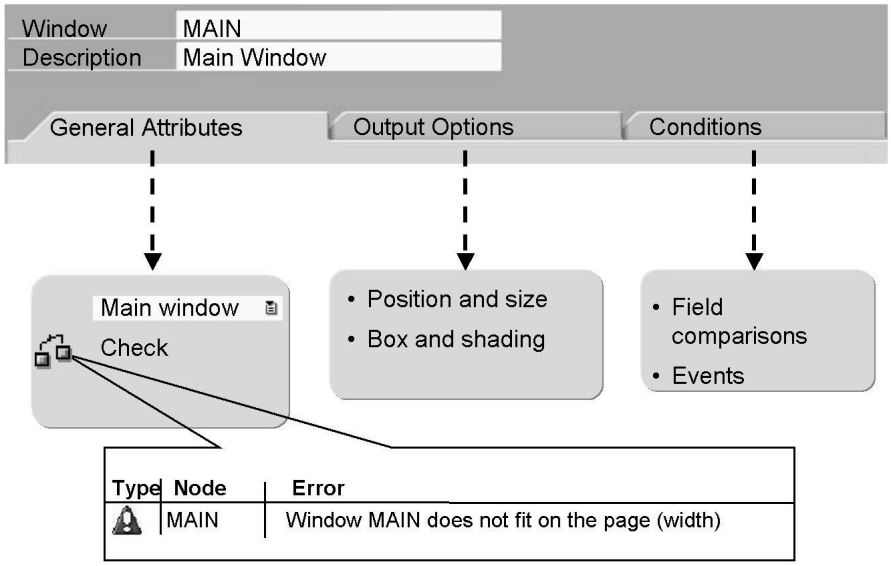


Figure 19: Window Attributes

You can execute checks on different levels, from the bottom node to the entire form. To ensure that all windows are free of errors, they must fit onto the respective page, and the main window must have the same width on all pages. All error messages are displayed in the bottom part of the maintenance screen. By clicking the name of a node you can directly go there.

You set the position and the appearance of a window using the output options.

As with most node types, you can use conditions for windows to determine the time they are processed. You can choose from a number of processing events, such as *not on first page* or *only on first page,* and also control processing by means of specific values. For example, you might want to print text A for certain customers only and text B for all other customers. If the conditions set for a window are not fulfilled, neither the window nor its subnodes are processed. The same is true for all other nodes and subnodes for which conditions have been specified.

If you use identical window nodes on different pages, each node has its own *Output options* and *Conditions* tabs.

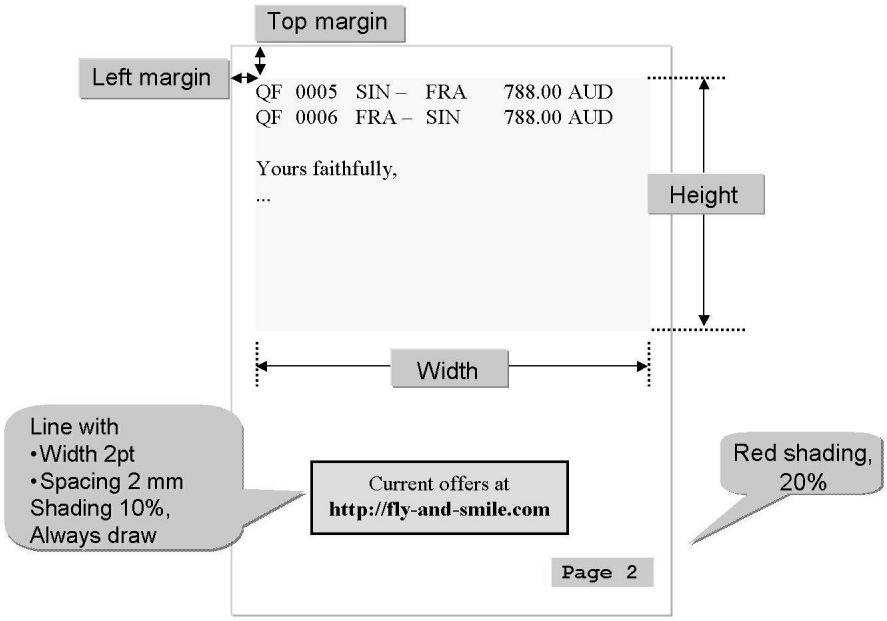


Figure 20: Windows: Output Options

You determine the position of a window by specifying the upper-left margin and its size by entering its height and width. If you draw a window in the Form Painter, the values you set are automatically copied to the maintenance screen and vice versa.

As with all other nodes that allow the output of text, you can define a box and a shading for windows.

As of SAP Web Application Server 6.10, you can make detailed settings for the four window borders (top, bottom, left, and right). You can define lines with different widths and colors. The distance between the border and text can also be set for each window border individually. You can also choose the color for shading and its intensity for each window.

With SAP R/3 4.6C, the chosen line width applies to each of the four borders of a window. Only the color black is available for shading but in all intensity. You set the distance between the text and the window in the *Vertical Spacing* or *Horizontal Spacing* field.

If you select the *Always draw box and shading* checkbox, the window is output in the format chosen even if it does not have any contents.

You can use the following units of measure:

CM, MM, IN (inch = approximately 2.54 cm), PT (point = 1/72 inches), and TW (twip = 1/20 points). For vertical length specifications, you can also use LN, and for horizontal length specifications CH. You define these units in the form attributes.

Using the Form Painter

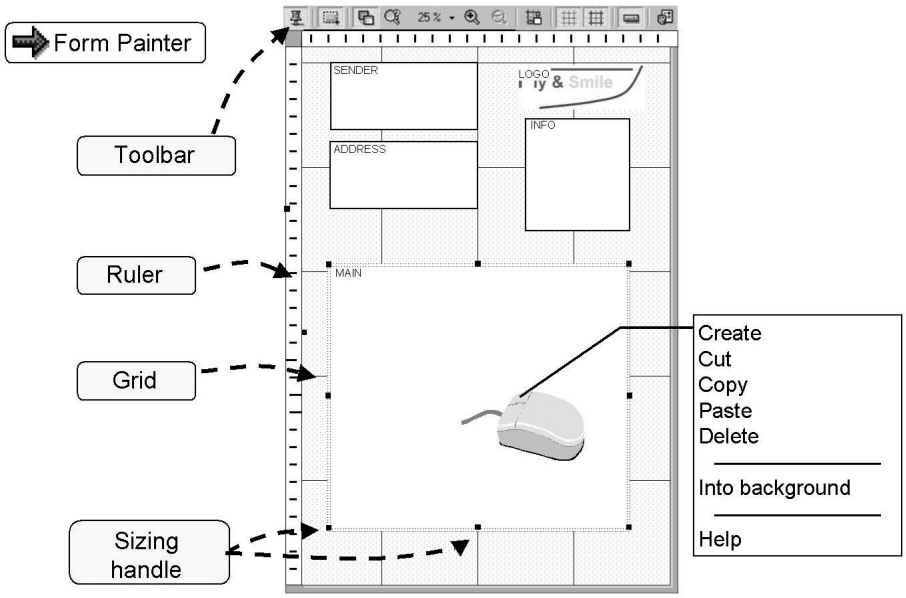


Figure 21: The Form Painter

You use the Form Painter to check/change the layout of a form. You can show or hide the Form Painter in the Form Builder by clicking the corresponding pushbutton or by choosing *Utilities → Form Painter on/off* from the menu. The Form Painter always displays the page selected in the navigation tree, including all output areas (windows, graphic windows, and address windows) and the background picture, provided there is one.

To edit an output area, select it with a mouse click. The corresponding node is also displayed on the maintenance screen. You can change the size of a window by clicking one of the sizing handles situated at the corners and the sides of the selection rectangle and dragging the handle to its new position while keeping the left mouse button pressed. If you want to reposition an output area, click the area and move it while keeping the left mouse button pressed (Drag and Drop). All size- and position-related changes that you make are automatically copied to the maintenance screen.

The context menu (right-mouse button) is also available in the Form Painter. You can use this menu to create or delete output areas and perform normal clipboard functions (cut, copy, and paste). Choose the *Into background* option if a small window is completely hidden by a larger one and you want to edit the small one. Using this option has no effect on the actual print output. In general: If windows, graphics, or texts overlap, they are printed one by one over another.

As of SAP Web AS 6.10, the keyboard can also be used for editing in the Form Painter, for example, Ctrl+C, Ctrl+X, and Ctrl+V for temporary storage operations.

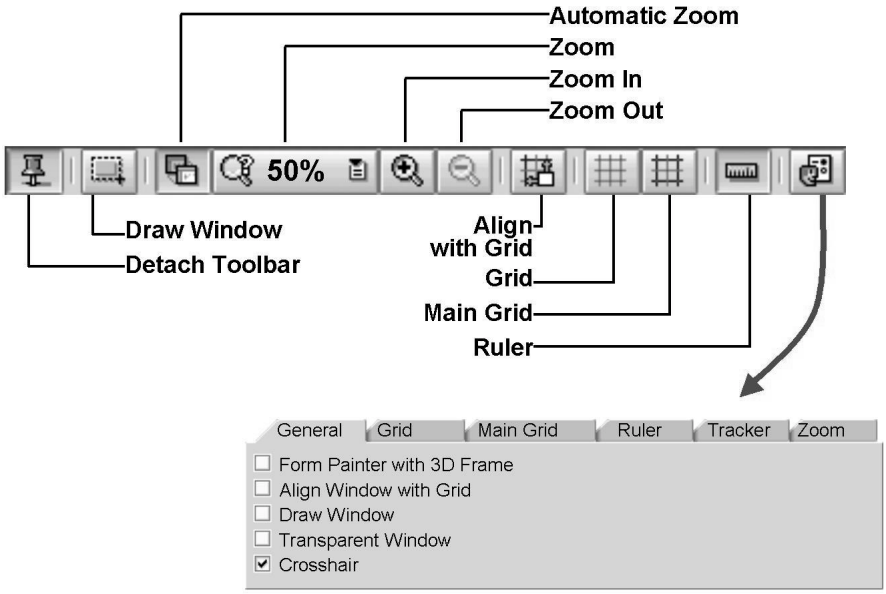


Figure 22: Form Painter: Settings

The detachable toolbar of the Form Painter contains pushbuttons for the most important settings. To display further options, choose *Utilities* → *Settings* or click the right pushbutton of the toolbar.

If the *Draw window* option is selected, you can draw a window directly by clicking on a free area in the Form Painter and dragging the mouse with its button pressed until the window has reached the desired size.

Several zoom options are available to adjust the display. The most comfortable option is the *Automatic zoom*.

To ensure that your output areas are correctly aligned, you can display a detail grid and/or the main grid. You can also make a setting in the Form Painter to ensure that output areas are automatically aligned with the grid when you move them with the mouse. You can set the step size of both grids. Using a crosshair cursor instead of the normal mouse pointer helps you to align the nodes correctly. You activate the crosshair cursor on the *General* tab of the *Settings* menu.

The *Tracker* tab of the Form Painter settings allows you to determine how the window that is currently selected is to be highlighted.

Other Settings of SAP Smart Forms

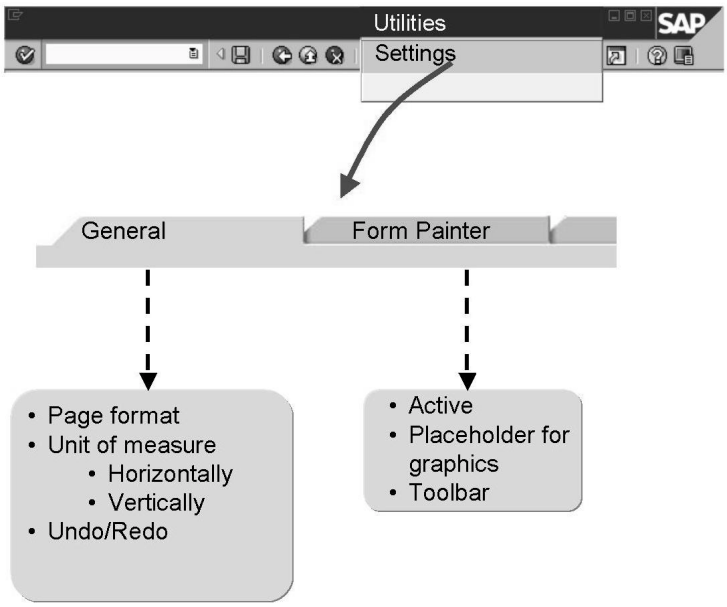


Figure 23: Other Settings

If you choose *Utilities* → *Settings*, you can make the following settings:

General tab

- Default value for the *page format* of new pages
- Default values for horizontal and vertical *units of measure*

As of SAP Web Application Server 6.10, you can also set whether it is possible to undo or redo changes in the Form Builder.

Form Painter tab

- Activate/deactivate the Form Painter (*Active* checkbox) *Placeholder for graphics.* This setting is sometimes useful to increase performance, such as if your workstation computer is too slow. If you select this checkbox, graphics are represented in the Form Painter simply as frames that are of the same size.
- Activate/deactivate the *toolbar* of the Form Painter

Activating and Testing Forms

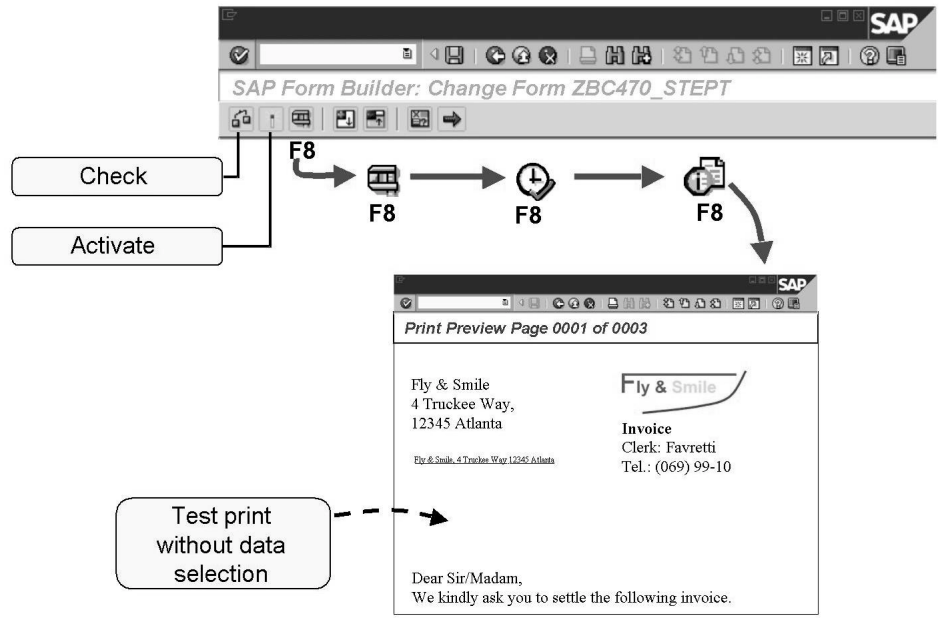


Figure 24: Activating and Testing Forms

You should test your form after making changes. You can make local checks at node level or check the entire form.

Before you can use a form in programs, you have to activate it. To do this, choose *Form → Activate* or click the second pushbutton in the toolbar of the Form Builder. Activating a form means that the entire form is checked and saved and the function module is generated.

If you have changed and saved your form but want to revert to the active form version, choose *Utilities → Return to active version in the Form Builder*. Note that this deletes the inactive form version that you have edited.

To test your form from within the Form Builder by using the print preview or by printing the form, choose *Form → Test* or click the third pushbutton in the toolbar of the Form Builder. The system takes you to the Function Builder, which is the development environment for function modules. The name of the function module generated is already entered here. For testing purposes, you can also enter values in the interface of the function module. Click *Test* again (menu: *Function Module → Test → Single Test*). Choose *Execute*. In the *Print Attributes* dialog box, enter your printer in the *Output Device* field and choose *Print Preview* or *Print*. Tip: The quickest way to go to the print preview is to press the function F8 four times in the SAP Form Builder.

You can only test inactive forms as of SAP Web Application Server 6.10. With SAP R/3 4.6C, you have to activate a form even just to test it.

Exercise 1: First Steps with the SAP Form Builder

Exercise Objectives

After completing this exercise, you will be able to:

- Copy forms
- Navigate in the SAP Form Builder
- Determine output options for forms, pages, and windows
- Create pages and windows
- Check, activate, and test forms

Business Example

The Fly & Smile travel agency constitutes of a number of independent units located in different parts of the world. Each unit is headed by a business unit head and operates as an independent entity under the common parent company. You are responsible for creating invoices, which is a step-wise procedure, of the booked flights for the respective customers. Your need to copy an existing form and make some changes to that form using the SAP Form Builder. The data from this form is to be used for creating invoices.

Task 1:

The following applies to all exercises:

Printer

Your course instructor tells you the printer to use as the output device. We recommend that you enter this printer as a default value in your user profile so that you do not have to type in the printer over and over again.

Development Class

Create a development class of your own called ZBC470_## (## is your two-character group number). You will save all objects that you create during the course to this development class. Ask your instructor for help if necessary.

Start transaction SE80 (Object Navigator). Choose *Workbench → Edit object* and go to the *More* tab. Enter the name of your development class. Choose the page icon at the bottom of the dialog box to create your development class. Enter a description and choose the disk icon. On the dialog box that appears next, enter the request (*Own requests* pushbutton) that your instructor created for you.

Naming Convention

Please observe the naming conventions specified in the exercises. This makes it easier for you (and the instructor) to track down errors.

Note: Copy template for the form: BC470_STEPT

Package/Development class (for all exercises): ZBC470_##

Name of the form to be created: ZBC470_##_STEPS

Model solution: BC470_STEPS

Application program for testing purposes: SAPBC470_DEMO

Get an Overview

1. The best thing to do at the beginning of the exercise is to run the program SAPBC470_DEMO to get an idea of the current layout of the template form BC470_STEPT.

Task 2:

Copy template

1. Copy the template form BC470_STEPT to ZBC470_##_STEPS (## is your two-character group number).

Continued on next page

Task 3:

Set general attributes

1. Enter a description and specify that the form should only be translated into German, French, and Spanish.

Task 4:

Choose page Format.

1. Change the page format to DINA4. This requires that you adjust the width of certain windows. Check the form and make any necessary corrections.

Task 5:

Set background picture.

1. Use the black and white image BC470_FLY_AND_SMILE_WATER-MARK, object GRAPHICS, ID BMAP as the background image for the page FIRST. Set a resolution of 300 dpi.

   Delete the existing graphic in the top right corner.

   Note: To ensure that the graphic is not hidden by a window in the Form Painter, you should set the *Transparent windows* option for the Form Painter.

   The preview of the Form Painter is very rough. For a more realistic display, you need to generate the function module. With SAP R/3 4.6C, the form has to be activated for testing.

Task 6:

Create new window.

1. On the page FIRST, create a new window with the name INFO and the description *Clerk*. The window should be at the same height as the existing window ADDRESS2. Define a box for the window so that the window is always output even if it does not have any contents.

Task 7:

Create additional page.

1. Create an additional page with the name NEXT.
2. NEXT should be the next page of FIRST and of NEXT itself.

Continued on next page

3. On the NEXT page, create the main window at a height of 25 cm and the windows PAGE and FOOTER. Copy all three windows, including all subnodes, from page FIRST to page NEXT.
4. **Optional:** Page numbering should start on page NEXT at II in Roman format.

Task 8:

Test form.

1. Check and activate your form. Test the form by executing the function module that was generated automatically. What is the name of this function module?
2. Test your form again using the program SAPBC470_DEMO.

Solution 1: First Steps with the SAP Form Builder

Task 1:

The following applies to all exercises:

Printer

Your course instructor tells you the printer to use as the output device. We recommend that you enter this printer as a default value in your user profile so that you do not have to type in the printer over and over again.

Development Class

Create a development class of your own called ZBC470_## (## is your two-character group number). You will save all objects that you create during the course to this development class. Ask your instructor for help if necessary.

Start transaction SE80 (Object Navigator). Choose *Workbench → Edit object* and go to the *More* tab. Enter the name of your development class. Choose the page icon at the bottom of the dialog box to create your development class. Enter a description and choose the disk icon. On the dialog box that appears next, enter the request (*Own requests* pushbutton) that your instructor created for you.

Naming Convention

Please observe the naming conventions specified in the exercises. This makes it easier for you (and the instructor) to track down errors.

Note: Copy template for the form: BC470_STEPT

Package/Development class (for all exercises): ZBC470_##

Name of the form to be created: ZBC470_##_STEPS

Model solution: BC470_STEPS

Application program for testing purposes: SAPBC470_DEMO

Get an Overview

1. The best thing to do at the beginning of the exercise is to run the program SAPBC470_DEMO to get an idea of the current layout of the template form BC470_STEPT.

Continued on next page

a) Choose *System → Services → Reporting*. Enter the name SAPBC470_DEMO, and execute the program using the function key F8. Enter BC470_STEPT as the form name, and execute the program using function key F8.

Task 2:

Copy template

1. Copy the template form BC470_STEPT to ZBC470_##_STEPS (## is your two-character group number).

   a) Start transaction SMARTFORMS by entering the transaction code into the OK code field or by choosing *Tools → Form Printing → Smart Forms*, in the SAP menu. Enter BC470_STEPT in the *Form* field, and choose the *Copy* icon in the toolbar. Enter ZBC470_##_STEPS as the new form name and press *Enter*. You will be prompted for a package (SAP R/3 4.6C: development class) in which to save the form. Choose ZBC470_##, press *Enter*, and choose the request created by your instructor by clicking *Own Requests* on the next screen. The name of the form copied is automatically transferred into the *Form* field. Choose the *Change* pushbutton. The system then takes you to the SAP Form Builder. When prompted, enter ZBC470_## as the package (SAP R/3 4.6C: development class) and on the following screen the request created by your instructor.

Task 3:

Set general attributes

1. Enter a description and specify that the form should only be translated into German, French, and Spanish.

   a) You make these settings on the maintenance screen, which forms the middle part of the SAP Form Builder. Enter the description in the second line, directly below the form name. To set the language attributes, select the *To selected languages* option in the *Language attributes* group box. Click the icon to the right and select the languages on the dialog box that appears next.

Continued on next page

Task 4:

Choose page Format.

1. Change the page format to DINA4. This requires that you adjust the width of certain windows. Check the form and make any necessary corrections.

   a) The page format is defined centrally for the entire form with the exception of the orientation, which can be set at page level. Go to the *Output options* tab and choose the DINA4 page format.

   Check the form by clicking the pushbutton furthest to the left in the toolbar. The system checks the entire form. The error message is displayed in the bottom part of the maintenance screen. The message indicates that the width of the window FOOTER is too large for the new paper format chosen. You can directly go to this window by clicking the name of the node in the error display. Change the width of the window FOOTER in the *Position and Size* group box on the *Output Options* tab. Alternatively, you can also activate the Form Painter by clicking the pushbutton furthest to the right in the toolbar and adjust the width of the window using the mouse. To do this, click one of the sizing handles and drag the window border to the desired size while keeping the left mouse button pressed. Check your form again. There should not be any errors.

Task 5:

Set background picture.

1. Use the black and white image BC470_FLY_AND_SMILE_WATERMARK, object GRAPHICS, ID BMAP as the background image for the page FIRST. Set a resolution of 300 dpi.

   Delete the existing graphic in the top right corner.

   Note: To ensure that the graphic is not hidden by a window in the Form Painter, you should set the *Transparent windows* option for the Form Painter.

Continued on next page

The preview of the Form Painter is very rough. For a more realistic display, you need to generate the function module. With SAP R/3 4.6C, the form has to be activated for testing.

a) Double-click the page FIRST in the navigation tree. The attributes of that page are then displayed on the maintenance screen. Go to the *Background picture* tab and enter the values specified for the name, the object, and the ID. Select *Black and White Bitmap Image (BMON)* and determine the output attributes for the background picture. 300 dpi, *Print preview and print* output mode and a vertical and horizontal position of your choice. Update the screen by choosing *Enter*.

You can delete the existing company logo by selecting the logo in the Form Painter or in the navigation tree and choosing *Delete* from the context menu using the right mouse button.

Task 6:

Create new window.

1. On the page FIRST, create a new window with the name INFO and the description *Clerk*. The window should be at the same height as the existing window ADDRESS2. Define a box for the window so that the window is always output even if it does not have any contents.

   a) From the context menu (right mouse button) of the page FIRST, choose *Create → Window*. The system displays a window with a default size and a default name in the top left corner of the Form Painter. Move this window with the mouse to the desired position. To ensure that this window is at exactly the same height as the window INFO, you can choose the *Align with grid* option in the toolbar of the Form Painter, or enter an identical value for both windows in the *Upper margin* field on the *Output options* tab. Change the name of the new window to INFO on the maintenance screen and enter a description. You set the box on the *Output options* tab. Select the *Always draw box and shading* checkbox.

Task 7:

Create additional page.

1. Create an additional page with the name NEXT.

   a) From the context menu of the page FIRST, choose *Create → Page*. Change the name of the new page to NEXT on the maintenance screen and enter a description (“Next page”).

Continued on next page

2. NEXT should be the next page of FIRST and of NEXT itself.

   a) On the *General attributes* tab, enter NEXT as the next page. Repeat these steps for the page FIRST.

3. On the NEXT page, create the main window at a height of 25 cm and the windows PAGE and FOOTER. Copy all three windows, including all subnodes, from page FIRST to page NEXT.

   a) From the context menu of the MAIN window, choose *Copy*. This automatically copies the contents of the entire window, including all subnodes, to the clipboard. From the context menu of the page NEXT, choose *Paste*. This inserts the contents of the clipboard, that is the main window, into the page NEXT. Repeat these steps for the windows PAGE and FOOTER. Now, change the height of the window MAIN on the page NEXT.

4. **Optional:** Page numbering should start on page NEXT at II in Roman format.

   a) The page numbers are output in the window PAGE. The simplest way to suppress the output of page numbers on the page FIRST is to delete the entire window PAGE on the page FIRST. Do not delete the contents of the window PAGE. Since the contents of this window are identical on all pages, the page numbers would then also be lost for the page NEXT. Go to the *General attributes* tab of the page NEXT and set Roman digits for the page numbers and the mode *Increase counter*.

Task 8:

Test form.

1. Check and activate your form. Test the form by executing the function module that was generated automatically. What is the name of this function module?

   a) Perform a check as before by choosing the left pushbutton in the application toolbar. If the check does not return any errors, you should be able to activate the form. You can find the *Activate* pushbutton directly to the right of the *Check* pushbutton. The name of the function module is unique in your respective system. You can find out the name by choosing *Test* or pressing the function key F8. The quickest way to go to the print preview is to press the function F8 another three times. Make sure that the output device entered is the printer your instructor told you.

2. Test your form again using the program SAPBC470_DEMO.

   a) See task 1.

Lesson Summary

You should now be able to:

- Identify and create pages in forms and documents
- Define page structure and page attributes
- Explain the different types of windows
- Identify the use of the Form Painter
- Activate and test forms

Unit Summary

You should now be able to:

- Use SAP Form Builder to edit SAP Smart Forms
- Identify the attributes of an SAP Smart Form
- Identify and create pages in forms and documents
- Define page structure and page attributes
- Explain the different types of windows
- Identify the use of the Form Painter
- Activate and test forms

Test Your Knowledge

1. You can select nodes for editing by double clicking them in the _____ or the _____.

   Fill in the blanks to complete the sentence.

2. The _____ graphically displays the hierarchy of an SAP Smart Form.

   Fill in the blanks to complete the sentence.

3. Define the term *Style*.

   _____
   _____
   _____
   _____

4. The _____ format is a special XML format that contains specifications for the dictionary types used for the interface parameters.

   Fill in the blanks to complete the sentence.

5. A form page can be used only once in a document.

   Determine whether this statement is true or false.

   □ True
   □ False

6. What all settings can be made on the General Attributes tab of a page in an SAP Smart Form?

   _____
   _____
   _____
   _____

7. How do you ensure that all windows are free of errors?

   _____
   _____
   _____
   _____

8. In the Form Painter, the most comfortable zoom option to adjust the display is the _____.

   Fill in the blanks to complete the sentence.

9. How do you test a form?

   _____
   _____
   _____
   _____

Answers

1. You can select nodes for editing by double clicking them in the navigation tree or the Form Painter.

   Answer: navigation tree, Form Painter

2. The navigation tree graphically displays the hierarchy of an SAP Smart Form.

   Answer: navigation tree

3. Define the term *Style*.

   Answer: A *style* is a collection of different character and paragraph formats, which are then used in the form.

4. The XDF format is a special XML format that contains specifications for the dictionary types used for the interface parameters.

   Answer: XDF

5. A form page can be used only once in a document.

   Answer: False

   Depending on the amount of data, a form page can be used more a once in a document.

6. What all settings can be made on the General Attributes tab of a page in an SAP Smart Form?

   Answer: The following settings can be made on the General Attributes tab:

   - The next page. The default value is the page itself.
   - The type of automatic page numbering. You can choose between Roman and Arabic digits and uppercase and lowercase format and determine the behavior of the page counter.

7. How do you ensure that all windows are free of errors?

   Answer: To ensure that all windows are free of errors, these windows must fit onto the respective page and the main window must have the same width on all pages.

8. In the Form Painter, the most comfortable zoom option to adjust the display is the *Automatic zoom* .

   Answer: *Automatic zoom*

9. How do you test a form?

   Answer: You test a form by clicking F8 four times.

Unit 3

Texts, Addresses, and Graphics

Unit Overview

In this unit, you will learn how to enter text using the inplace editor, include fields into texts, create text modules with the text module maintenance tool, and include text modules in forms. You will also learn how to include include text (SAPscript texts), use formatting options, create nodes that work with Business Address Services, and import graphics.

Unit Objectives

After completing this unit, you will be able to:

- Create text nodes
- Insert fields using the field list
- Format fields
- Create addresses from Business Address Services
- Import graphics in SAP R/3 Enterprise
- Create graphics as separate windows or as subnodes of a window

Unit Contents

Lesson: Text Nodes

Lesson Overview

In this lesson, you will learn about creating text nodes and inserting fields in forms using field list. You will also learn about formatting fields that are inserted in forms.

Lesson Objectives

After completing this lesson, you will be able to:

- Create text nodes
- Insert fields using the field list
- Format fields

Business Example

The Fly & Smile travel agency constitutes a number of independent units located in different parts of the world. Each unit is headed by a business unit head and operates as an independent entity under the common parent company. You are responsible for sending invoices of the booked flights to the respective customers. To include text in your invoice forms, you need to create text nodes in the invoice forms.

Creating Text Nodes

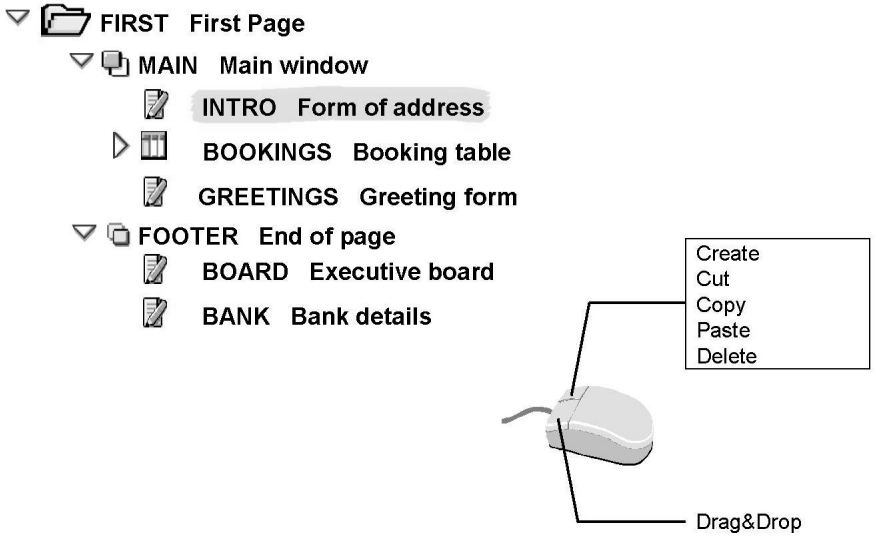


Figure 25: Text Nodes

You enter all texts in the form using text nodes (with the exception of address nodes). Text nodes are subnodes of windows and their subnodes, such as folders or templates.

For existing text nodes, you can use the context menu (right mouse button). You use this, for example, to create a further text node directly afterwards, on the same level. Alternatively, you can create text nodes using the context menu of the superordinate node. If you use this method, the new node is entered as the top subnode of this node.

You can also call these functions from the menu by choosing *Edit → Node.*

Text nodes themselves cannot have subnodes.

Note that text in secondary windows that no longer fits will be cut off. This is also the case for copy windows and final windows, as of SAP Web Application Server 6.10. On the next page, further processing only occurs in the main window.

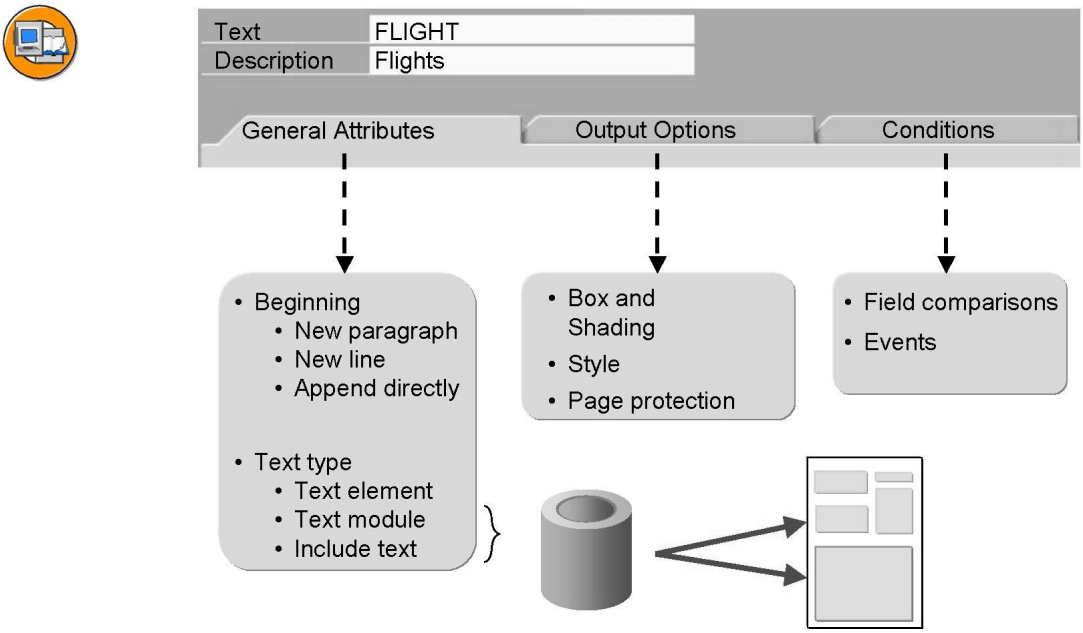


Figure 26: Text Nodes: Attributes

If you select a text node in the navigation tree or in the Form Painter, its attributes are displayed on the maintenance screen.

The three tabs are similar to those for windows.

General attributes:

The text type you choose allows you to determine whether the text should be saved and edited within the form (as a text element) or outside the form. In the latter case, you can choose between a SAP Smart Forms text module and a SAPscript include text.

You also determine how two directly successive text elements should be combined.

The remaining fields you see on the tab vary depending on the text type.

Output options: The previous information on *boxes and shading* for windows is also applicable to text nodes.

In addition to boxes and shading, you can also *assign a style* to the text node. A style is a collection of different character and paragraph formats.

If the text node is in the main window, you can choose *Page protection*. This option prevents text from being separated by page breaks. If the protected text does not fit onto the current page, it is output on the next page.

Conditions: See the conditions for windows.

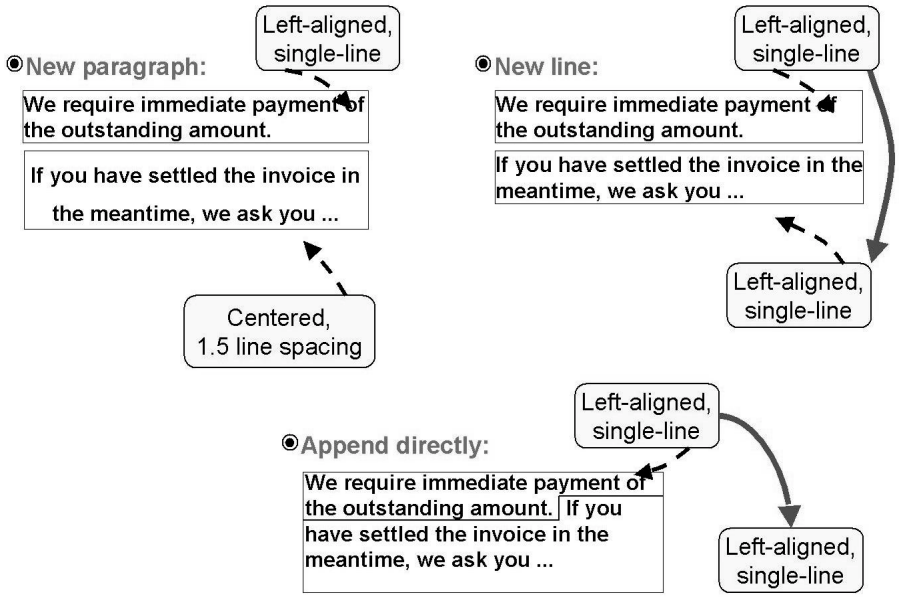


Figure 27: Linking Text Nodes

On the *General attributes* tab page, you determine how two successive text nodes are combined in the same window.

If you choose *New paragraph,* the text of the second node begins in a new line based on the paragraph format that you specified for this paragraph. This means that the two text nodes are completely independent of each other. *Enter* within **one** text node creates a new paragraph.

If you select *New line*, the text of the second node also begins in a new line. However, the format of the last paragraph of the first text node is used for the first paragraph of the second text. *Shift+Enter* within **one** text node creates a new line.

You can also choose *Append directly*. In this case, two successive text nodes are combined without blanks or blank lines. The resulting paragraph is assigned the format of the first text element.

If the first node has a box and/or shading, the second node is always appended using the *New paragraph* option, irrespective of which option you select.

Creating Text Elements

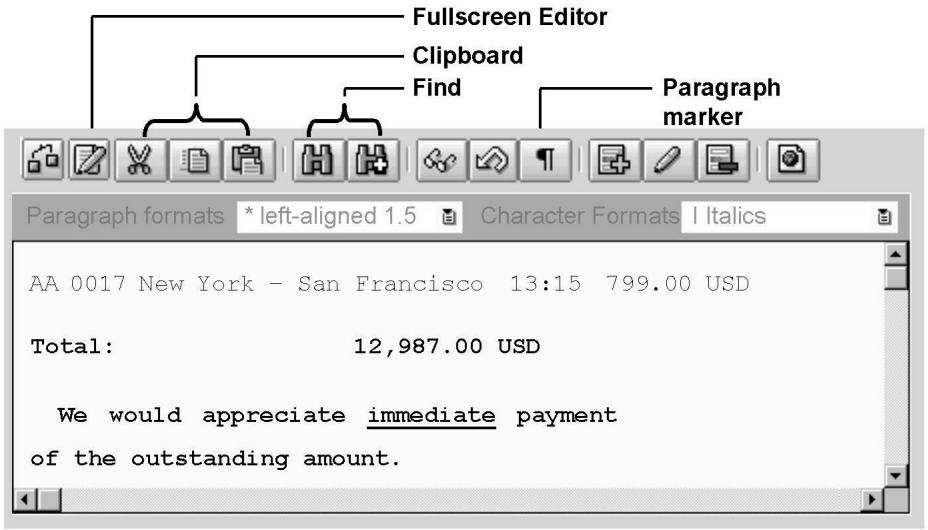


Figure 28: Text Elements in the Editor

If you choose *text element* as the text type, the inline editor is displayed on the *General attributes* tab. You can enter text in the usual way as you would do in any common word-processing system. Alternatively, you can switch to the full-screen mode by choosing the *Text editor* pushbutton.

You can use the clipboard by selecting text blocks with the mouse and clicking the *Cut*, *Copy*, or *Paste* pushbutton. This way you can copy text sections between different windows or forms.

Lines in text nodes are broken automatically depending on the window width. You can also use the *Enter* key in the editor to create a new paragraph that might have a different format than the preceding one. *Shift-Enter* allows you to create a line break within a paragraph.

The *Paragraph mark on/off* pushbutton allows you to determine whether you want to display non-printing characters such as blanks, tabulators, paragraph marks, and line breaks.

Formatting Text

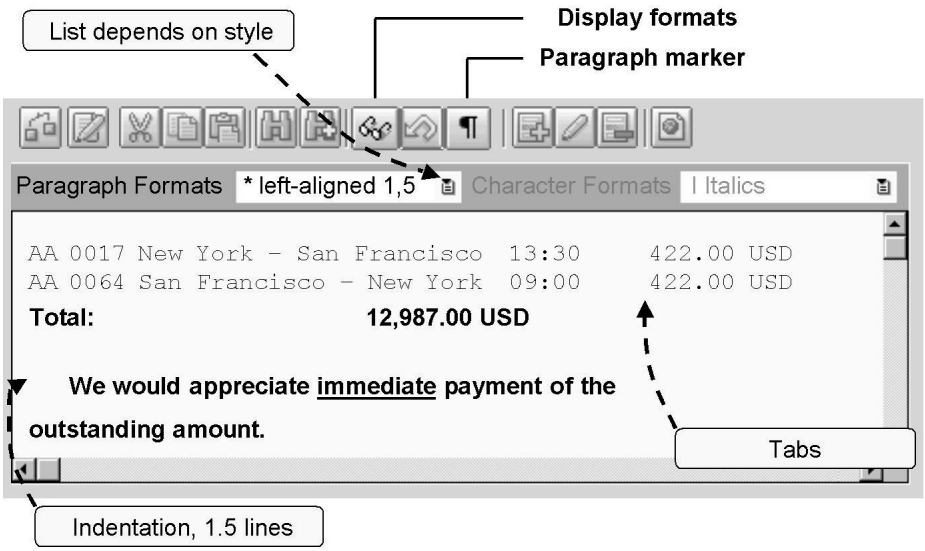


Figure 29: Formatting Texts: Paragraph Formats

You can format selected text sections. These text sections are displayed as they appear when printed (WYSIWYG = **W**hat **Y**ou **S**ee **I**s **W**hat **Y**ou **G**et).

For each paragraph, you can choose a paragraph format from the selection list in the editor. A paragraph format is a collection of format settings, such as tabs, type of justification, and so on. The paragraph formats available for selection in the list depend on the style you have chosen. If you have entered a style for more than one node, for example, for the form attributes and for a table, the following applies:

The style of a node overrides the style of the form attributes.

The style of a lower-level node overrides the style of a higher-level node. For text nodes, this means that the style of the text node is used first; if there is none, the style of the next higher node is used, and so on.

If you do not choose a paragraph format, the standard paragraph format of the style is used.

As of SAP R/3 4.6C, support package 9, the system automatically displays the format set at the current cursor position in the paragraph and character format list. You only need the *Display formats* function if you want to obtain detailed information on the format, or if text has been formatted using several character formats. See also the SAP note 327636.

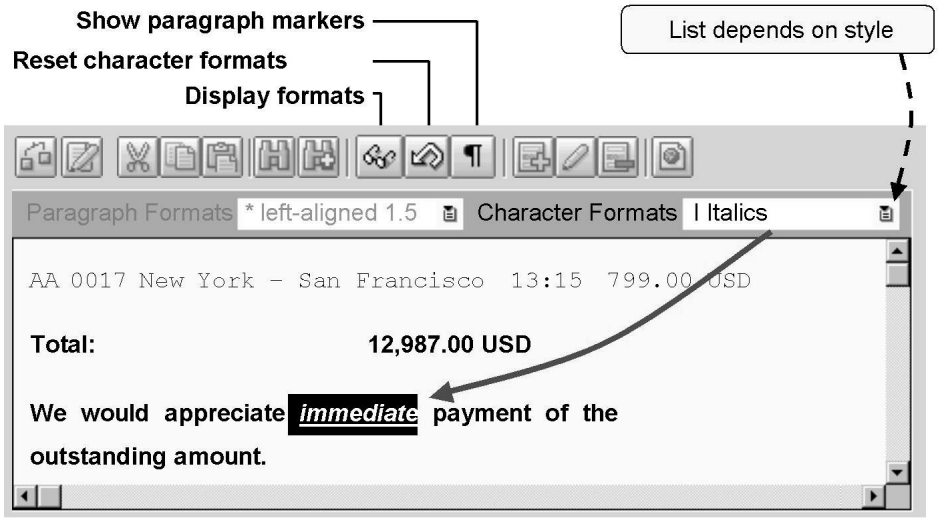


Figure 30: Formatting Texts: Character Formats

You can also assign one or more character formats to the selected text. A character format is a collection of format settings such as the *font type* or *superscript*. The same is true for the character formats available as for the paragraph formats. The *Reset character formats* pushbutton resets all character formats for the selected text to the formats of the paragraph format.

If you want to know which paragraph and character formats are valid at a specific cursor position, click the *Display formats* pushbutton. The system displays a dialog box containing the formats. For more information on a specific paragraph or character format, double-click the respective format.

Using Field List

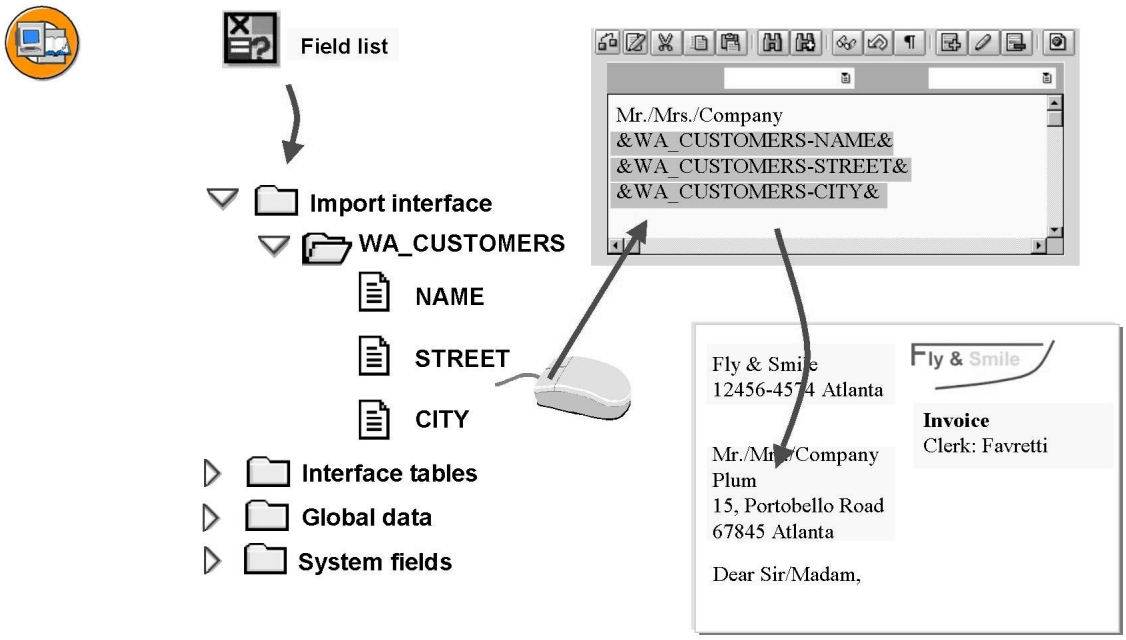


Figure 31: Field List

Normally, your form contains not only static text but also variable data, referred to as fields, which are read from the database at application runtime or are entered by the user.

The simplest way to insert fields is to use the field list. You can hide or show the field list by clicking the corresponding pushbutton or by choosing *Utilities* → *Field list on/off* from the menu. The following types of fields are available for selection:

All fields that are recognized by the form as import, export or table parameters through the form interface that implies they come from the application program.

All global data and field symbols you have created in the form in the global definitions.

System fields that are filled automatically during program execution.

You insert the fields using drag and drop. Drag the name of a field from the field list to the desired position in your text element.

If a field is structured, click the triangle icon to the left of the corresponding folder to access the individual subfields.

Defining System Fields

&SFSY-DATE&	Date
&SFSY-TIME&	Time
&SFSY-PAGE&	Number of current print page
&SFSY-FORMPAGES&	Total number of pages in document currently being edited
&SFSY-JOBPAGES&	Total number of pages of all documents in current print request
&SFSY-WINDOWNAME&	Name of current window
&SFSY-PAGENAME&	Name of current page

As of SAP Web Application Server 6.10:

&SFSY-XSF&	Indicator for XSF output
&SFSY-COPYCOUNT0&	Copy counter (0 = original, 1 = 1st copy)
&SFSY-COPYCOUNT&	Copy counter (1 = original, 2 = 1st copy)
&SFSY-USERNAME&	User name

Figure 32: System Fields

DATE: Date display. The display format is set in the user master record.

TIME: Time in the form HH:MM:SS (HH: hours, MM: minutes, SS: seconds).

Page numbers:

PAGE: Number of current print page. You determine the format of the page number, such as Arabic, or numeric and the mode, that is, increase, initialize, or leave unchanged on the *General attributes* tab of the page node.

FORMPAGES: Total page number for document currently being edited. You can print the page number in the form “Page x of y”.

JOBPAGES: Total number of pages of all documents in the current print request.

WINDOWNAME: Name of current window

PAGENAME: Name of current page

New as of SAP Web Application Server 6.10:

XSF: This indicator is set if the form is output in the XSF format or HTML format.

COPYCOUNT and COPYCOUNT0: Query whether the original or the copy is output.

SUBRC: 0 if text module or include text exists, 4 if not.

USERNAME: Logon name of the user who is printing the form.

For internal use (SAP R/3 4.6C only):

MAINEND: Is set if the processing of the main window ends on the current page.

PAGEBREAK: Is set if a page break occurs on the current output page.

EXCEPTION: Is set if error messages occur in the form that lead to termination of output.

Formatting Fields

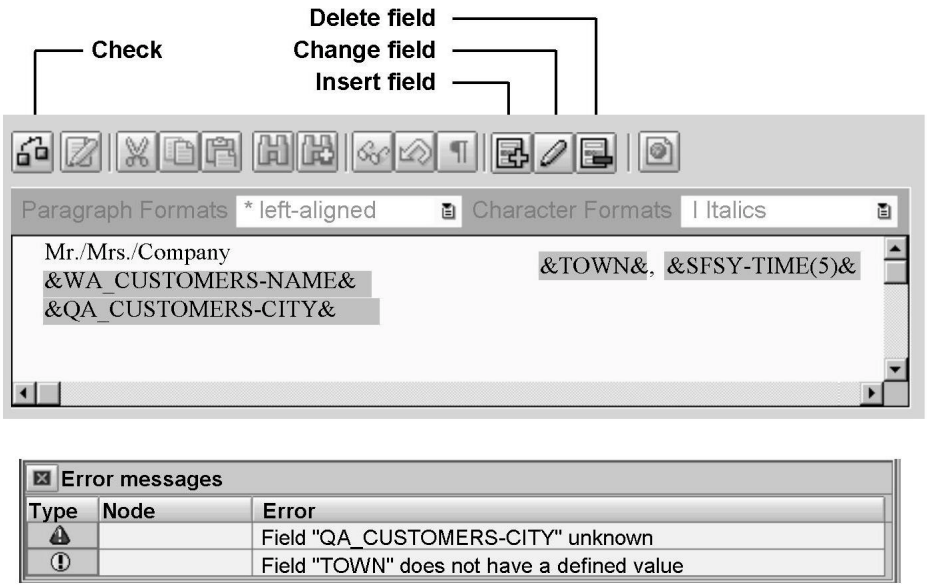


Figure 33: Editing Fields in Texts

You can also use the *Insert field* pushbutton to add fields to your text element. Enter the name of the field enclosed in ampersands (&). This also allows you to access the ABAP system fields of the structure SYST, such as &sy-uname& (user name). Field names are not case-sensitive.

The fields are grayed out to distinguish them from normal text. They cannot be directly changed or deleted. To delete a field, select it and click the *Delete field* pushbutton. To change a field, place your cursor on the field and click the *Change field* pushbutton. You need this function, for example, to determine formatting options such as the output length for a field.

To ensure that your field entries are correct, perform a check by clicking the corresponding pushbutton in the editor. The system notifies you of any errors that might exist by means of an exclamation mark in a red triangle. If errors exist, you cannot activate the form.

Only if you use the check function of the Form Builder (first pushbutton in the toolbar), does the system check if all fields used have been assigned a value when they are processed or if they are still initial. If the fields are initial, the system issues a warning (exclamation mark in a yellow circle) but you can nevertheless activate the form. Fields which are initial at application program runtime will be ignored. For more information, see the online documentation.

`&field+6&`	Offset (only for character fields; here: 6)
`&field(9)&`	Output length (here: 9)
`&field(S)&`	Suppress +/- signs
`&field(<)&`	Display +/- signs to the left of the number
`&field(8.2)&`	Decimal form. (here: 8 output places, 2 after dec. point)
`&field(T)&`	Suppress thousands separator
`&field(Z)&`	Suppress leading zeros of numbers
`&field(I)&`	Suppress output of initial values
`&field(R)&`	Right-justified (only in combination with output length)
`&field(F<Filler>)&`	Replace left-justified blanks with fillers

Figure 34: Formatting Options

Formatting options allow you to adjust the value of a field before it is output. You enter the corresponding shortcuts directly after the field name and always in **uppercase**. Some of the options can be combined, for example +6(9) or (8.2).

The formatting options are not appropriate for all data types of a field, for example, you cannot display numbers with an offset. The difference between character and numeric fields is described below:

Character fields:

By default, the value of a field name is displayed in full. However, blank spaces at the end of the value are cut off.

Evaluation sequence: Suppress blanks (C), <offset> and (<length>), right-justified display (R), insert filler (F).

Numeric fields:

The closing blank is interpreted as a plus sign. To suppress it, use the formatting options S.

Evaluation sequence: (<length>), +/- sign to the left (<), Japanese date (L), suppress blanks (C), right-justified display (R), insert filler (F).

For information on further formatting options, see the online documentation.

Creating Text Modules

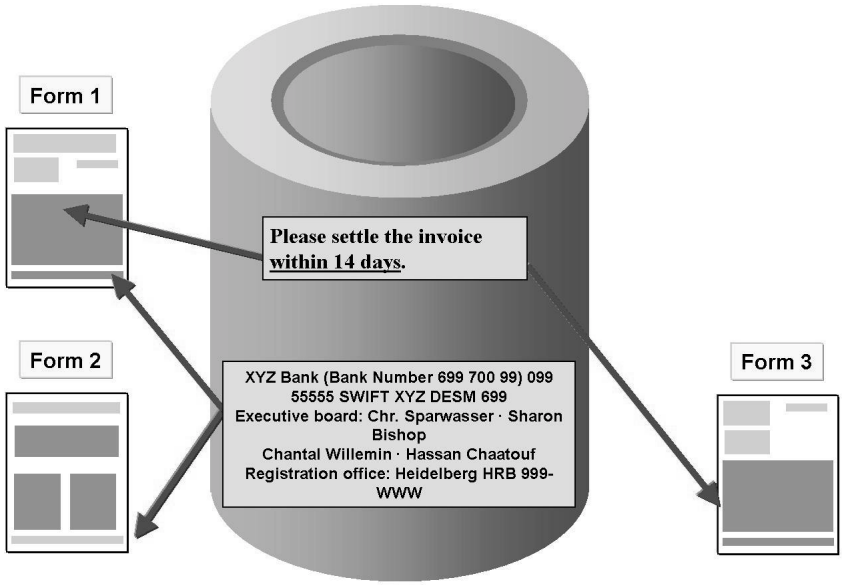


Figure 35: Text Type Text Module or Include Text

In some cases, it makes sense to store text not in the form itself but centrally in the database. The form contains only a reference, indicating which text should be used at application program runtime. This method has the following advantages:

You need to create the texts only once and can reuse these as required.

You make changes centrally only once without having to modify the actual forms. The reference in the form remains unchanged. Example: Your bank details change.

You can use text modules of SAP Smart Forms as well as include texts as you can use them in SAPscript. You set the text type on the *General attributes* tab of the text node. If you change the text type, the system displays a warning, because text already entered would be lost.

The names of the paragraph and character formats used are saved in the text module or include text. In the form, however, they are interpreted as they are defined in the style of the text node.

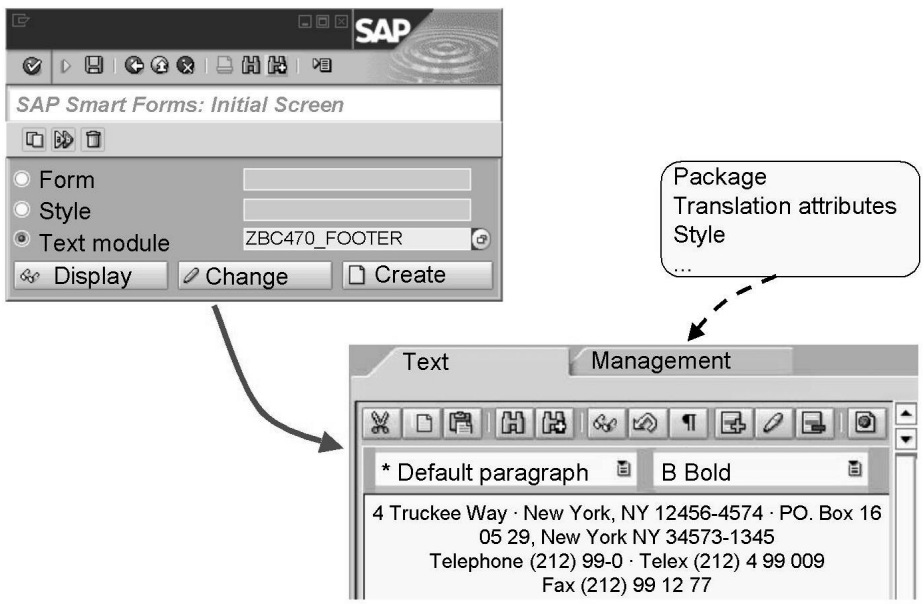


Figure 36: Creating Text Modules

You go to the maintenance transaction for text modules by choosing the *Text module* radio button on the SAP Smart Forms initial screen and then *Display*, *Change*, or *Create* depending on what you want to do. From this screen, you can also copy, rename, or delete existing text modules. To do this, choose the appropriate pushbutton from the toolbar or use the *Text module* menu.

The name of a text module that you create must be in the customer namespace and begin with Y or Z.

Since text modules are integrated with the SAP transport system (like SAP Smart Forms), you must assign them to a package. You do this when you first save your text module. In SAP R/3 4.6C, subsequent changes are not possible.

You work with the text module editor in the same way as you work with the inline editor of the form maintenance screen. However, neither the field list nor the check function is available since text modules can be inserted into various SAP Smart Forms with different interfaces. You can insert, delete, and edit fields using the corresponding pushbuttons. Note that the fields must be defined and filled in all forms that use the text module.

You must assign a style to each text module on the *Management* tab. The default style is the style *System*.

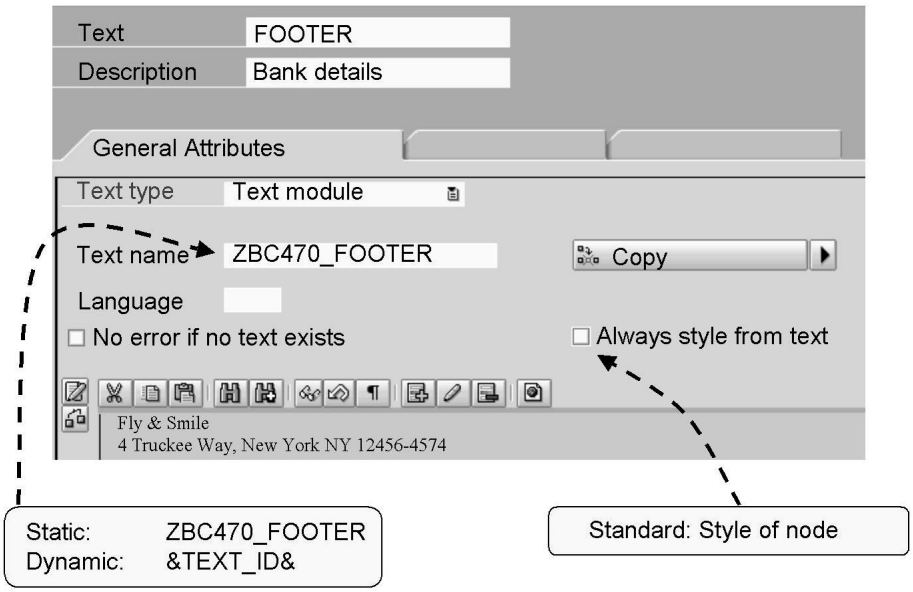


Figure 37: Text Modules in Forms

To insert a text module into a form, create a text node, choose the text type *Text module*, and enter a name.

You can also determine the name of the text module dynamically at application program runtime. To do this, enter the name of a field recognized in the form, enclosed in ampersands (&). If the space is not sufficient to enter the full name, click *Return*, or the triangle to the right of the *Copy* pushbutton.

As of SAP Web AS 6.10, you can determine the language of the text module statically or dynamically. In SAP R/3 4.6C, the text module is always entered in the same language as the form.

The text of a text module is always grayed out in the inline editor of the Form Builder because you cannot change it directly from within form maintenance.

If you determine the text module dynamically, its text cannot be displayed in the editor. Likewise, you cannot check from within form maintenance whether the field is filled with an appropriate value at application program runtime. In SAP R/3 4.6C, the current output request is terminated if the text module that you determined dynamically is not found when the program is executed. As of SAP Web AS 6.10, there is an additional field: *No error if no text exists.* If this field is selected, the application is not terminated at runtime. SFSY-SUBRC contains the value 4 if a text module is not found. Otherwise it contains 0.

If you choose the *Copy* pushbutton, the text of the text module is inserted into the form as a text element. Later, there is no link to the text module.

By default, the system uses the style of the text node or of a higher-level node. For a text module you can also choose *Always copy style from text*.

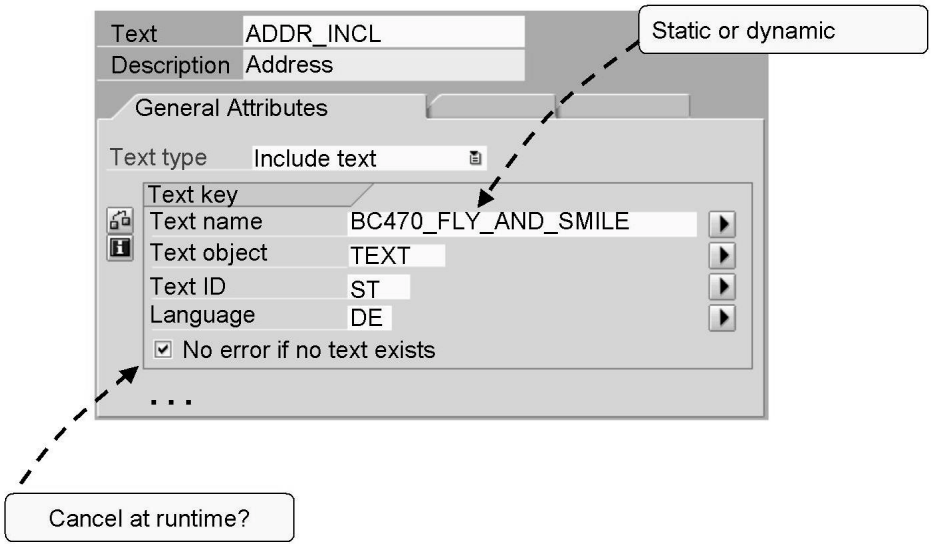


Figure 38: Include Texts

You can also insert existing SAPscript text. To do this, choose *Include text* as the text type of the text node and enter the necessary selection information in the *Text key* group box. You can also make these specifications dynamically (name of a field, enclosed in ampersands (&)).

If you select *No error if no text exists*, the function module generated does not terminate if the text specified is not found at runtime. As of SAP Web Application Server 6.10, there is the system variable SFSY-SUBRC. You can query a value for this variable in a program line node. If the text is found, SFSY-SUBRC has the value 0. If an error occurs, it has the value 4.

If you create new texts for use in SAP Smart Forms, you should always create SAP Smart Forms text modules rather than SAPscript texts. The reasons for this are:

You cannot maintain or preview SAPscript texts from SAP Smart Forms. As before, you can use transaction SO10 to maintain standard texts.

You cannot check whether fields of an inserted SAPscript text are actually defined in the form.

SAPscript texts are not automatically integrated with the transport system.

SAPscript texts are client-specific.

Note that SAP Smart Forms ignore all SAPscript commands in SAPscript texts. If required, you must convert these commands into SAP Smart Forms logic.

- Formats in include texts are interpreted in the same way as they are defined in the Smart Style of the node.
- Unknown formats in include texts are ignored.
- SAPscript styles in include texts are ignored.
- It is possible to override formats in SAPscript texts:

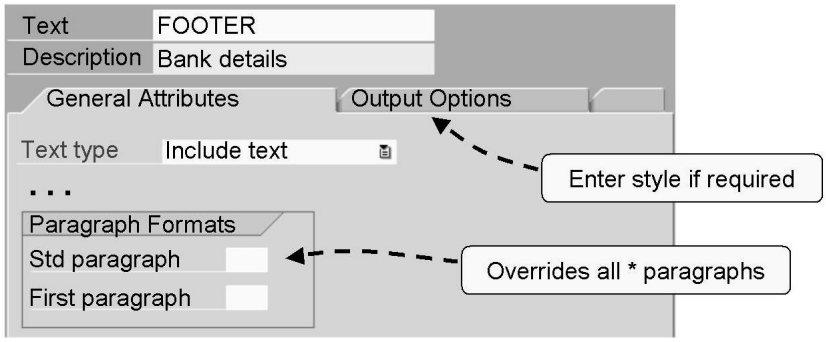


Figure 39: Formats in Include Texts

All paragraph formats and character formats of an include text are interpreted as they are defined in the style (Smart Style) that applies for the text node. You can determine this style on the *Output Options* tab page of the text node. If you do not enter anything here, the style of the superordinate node is used, such as a table or the style that you entered in the form attributes.

Example: The paragraph format B in a SAPscript text means *Bold*, but in the Smart Style it means *Small*. Each paragraph with format B in the text is therefore printed in small text.

You can override the paragraph formatting of include texts:

In the *Standard Paragraph* field, you can select a paragraph format of the Smart Style that applies for the current node. This format is then used for all paragraphs of the SAPscript text that are formatted using the standard paragraph (*). If you are familiar with SAPscript commands: An entry in this field corresponds to the PARAGRAPH addition of the INCLUDE command.

You use the *First Paragraph* attribute to set a paragraph format for the first paragraph of the include text - independent of how the paragraph is actually formatted in the include text (corresponds to the NEW PARAGRAPH addition of the SAPscript command INCLUDE). If the *Standard Paragraph* field remains empty, all standard paragraphs in the include text also adopt this paragraph format.

Overriding formats is only possible as of Support Package 23 for SAP R/3 4.6C. For preliminary corrections, see the SAP note 422120.

Formats in include texts are interpreted in the same way as they are defined in the Smart Style of the node.

Unknown formats in include texts are ignored.

SAPscript styles in include texts are ignored.

It is possible to override formats in SAPscript texts:

Lesson Summary

You should now be able to:

- Create text nodes
- Insert fields using the field list
- Format fields

Lesson: Addresses

Lesson Overview

In this lesson, you will learn about how to create addresses from Business Address Services (BAS).

Lesson Objectives

After completing this lesson, you will be able to:

- Create addresses from Business Address Services

Business Example

The Fly & Smile travel agency constitutes a number of independent units located in different parts of the world. Each unit is headed by a business unit head and operates as an independent entity under the common parent company. You are responsible for sending invoices of the booked flights to the respective customers. To print addresses of your customers in the invoice forms, you need to create address node in the invoice forms. You also want to include your company's logo in the background of invoices. To do this, you need to embed graphics in your invoice forms.

Creating Addresses

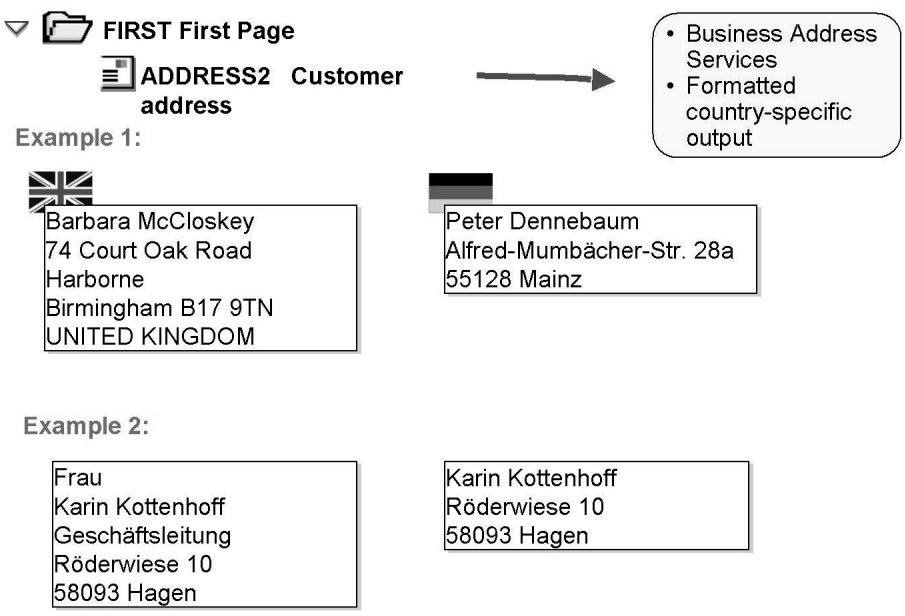


Figure 40: Addresses: Business Address Services

Instead of using their own tables for address information, many applications now access the Business Address Services (BAS, which before the SAP Web Application Server 6.10 were known as Central Address Management - CAM). In the BAS, addresses are identified by means of numbers.

SAP Smart Forms allow you to use the BAS. You do not need to know the technical details of BAS and the correct formatting of the addresses. The addresses are formatted in accordance with country-specific conventions, based on ISO 11180 and the guidelines of the Universal Postal Union. If the space in the form is not sufficient, some fields might not be displayed. For detailed information, see the documentation on the function module ADDRESS_INTO_PRINTFORM.

You normally create addresses as direct subnodes of a page, that is, as address windows. You can position them as required in the Form Painter. Alternatively, you can also create an address node as the subnode of a main or secondary window, a loop, or a template.

If you want to output formatted addresses without using the Business Address Services, you need to call the function module ADDRESS_INTO_PRINTFORM in a program line node.

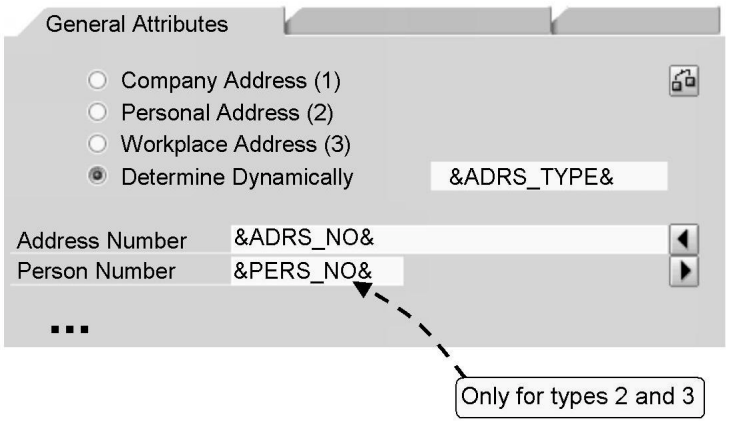


Figure 41: Addresses: Attributes I

You make the basic settings for the address node on the *General Attributes* tab page.

Address type:

Company addresses: Typical examples are delivery addresses or company codes. These addresses are uniquely identified by their address number.

Personal addresses: Addresses of this type are assigned to one natural person, along with other associated attributes, such as the form of address. Because a person can have more than one address, you enter both the address number and the person number for identification.

Workplace addresses: These are personal addresses in companies, which means they have additional attributes such as the department or the room number. You identify such an address by means of the address number and the person number.

Determine Dynamically: If you want to determine the address type at runtime, enter the name of the field (enclosed in ampersands) that must be filled at runtime with 1, 2, or 3.

You can also determine the address and the person number dynamically. If the length of the input field is not sufficient, click on the small triangle to the right of the field to make it larger. If the system does not find an address with the number specified in the Business Address Services at runtime, the function module of the form terminates with an error message.

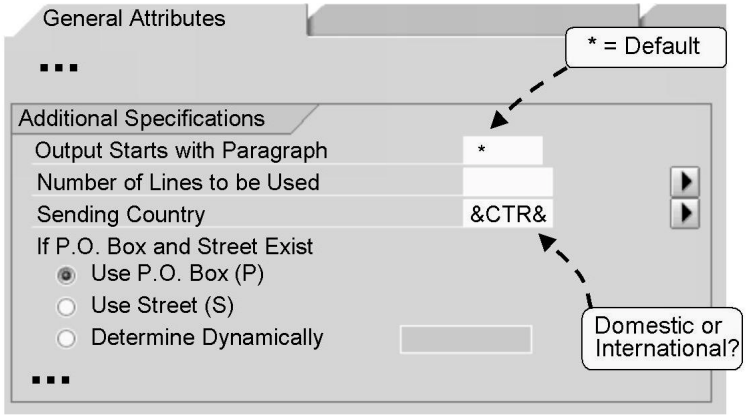


Figure 42: Addresses: Attributes II

In the *Output Starts with Paragraph* field, you set the paragraph format for the address node. This must be defined in the style of the address node or if you have not defined one in the output options, in the style of a higher-level node or the form. If you enter an asterisk, the system uses the default paragraph of the style.

If in the *Number of Lines to be Used* field, you set fewer lines than are required to output the full address, the Business Address Services suppresses address parts of reduced importance, such as the form of address or the function of the person in the company. The same applies if the window area you reserve for the address node is too small.

To determine whether the address is a domestic address or a foreign address, you should always enter the ID of the sending country (if required, dynamically using a field name). In this case, the address country or its respective ID is printed in international addresses, but not in domestic ones.

For addresses that have both a P.O. box number and a street address, use the radio buttons in the group box to determine which one to use.

As of SAP Web AS 6.10:

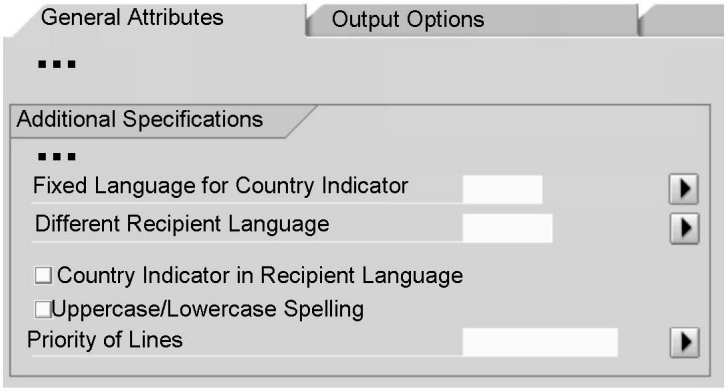


Figure 43: Addresses: Attributes III

As of the SAP Web Application Server 6.10, the lower section of the *General Properties* tab page contains an additional range of parameters.

In the *Priority of Lines* field, you can determine which part of the address is first suppressed if there is not enough space for the address node. For example, A stands for form of address, P for obligatory empty line, and D for department. For more information, see the documentation for the function module ADDRESS_INTO_PRINTFORM. Like other nodes, addresses have the *Output options* tab where you can determine the style and the box and shading. Also on this tab, you can set the values for the position and size of the output area. Prerequisite for this is that you have created the address node as a separate address window, that is, as a direct subnode of a page.

Lesson Summary

You should now be able to:

- Create addresses from Business Address Services

Lesson: Graphics

Lesson Overview

You will learn about how to import existing graphics in SAP R/3 Enterprise. You will also learn about how to create graphics as separate windows or as subnodes of a window.

Lesson Objectives

After completing this lesson, you will be able to:

- Import graphics in SAP R/3 Enterprise
- Create graphics as separate windows or as subnodes of a window

Business Example

The Fly & Smile travel agency constitutes a number of independent units located in different parts of the world. Each unit is headed by a business unit head and operates as an independent entity under the common parent company. You are responsible for sending invoices of the booked flights to the respective customers. To print addresses of your customers in the invoice forms, you need to create address node in the invoice forms. You also want to include your company's logo in the background of invoices. To do this, you need to embed graphics in your invoice forms.

Creating Graphics

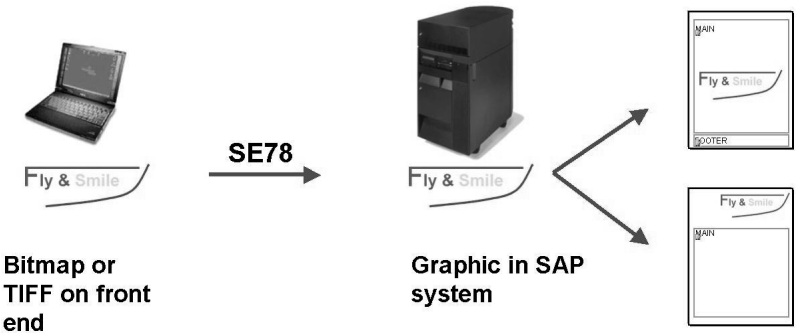


Figure 44: Additional Information: Graphics Administration I

Inserting a graphic, either as the background picture of a page or as graphic node, requires that the graphic has been imported from your front-end computer into the system using graphics administration. Graphics that you can use for SAP Smart Forms are stored in the Business Document Server.

You can display the graphics administration using transaction code SE78.

You can import graphics in the TIFF or bitmap format.

A graphic that has been imported once can be integrated in any number of forms in the system.

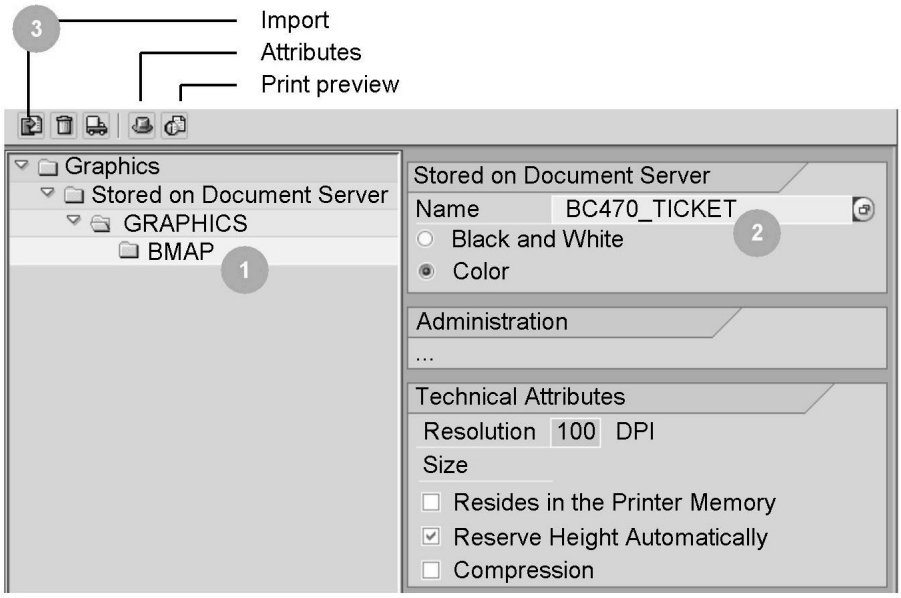


Figure 45: Additional Information: Graphics Administration II

The storage paths in the system are displayed on the left of the screen in the transaction SE78. For SAP Smart Forms, you can only use the *Stored on Document Server* option. You can determine which subfolders are displayed using transaction SE75.

On the right of the screen, information is displayed for the picture or a preview is displayed after you have selected a screen or after successful import.

To transfer a graphic to the document server, follow the steps below:

1. In the left screen area, select a subfolder.
2. In the field on the right, enter a name for the graphic, and decide whether you want black and white or color.
3. Click the left pushbutton in the toolbar. In the dialog box that appears, enter a description and determine whether the graphic should reside in the printer memory. If you select this checkbox, the graphic is stored in the printer memory during printout when it is used for the first time, so that it can be retrieved from there when needed again in the same print request. This might increase performance considerably. As of SAP Web AS 6.10, it is possible to compress graphics.
4. Start the import by clicking the *Enter* button.

After importing a graphic, you can delete it using the Trash can pushbutton or you can add it to a transport request using the Truck pushbutton. You can also display the graphic attributes or the graphic itself in the print preview on the right side of the screen.

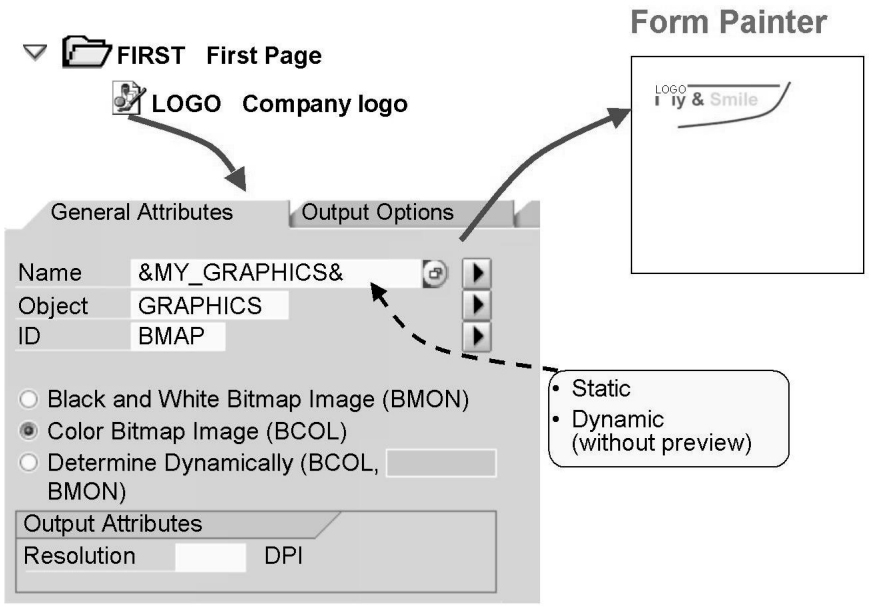


Figure 46: Graphics

You can embed graphics not only as background pictures, but also as separate graphic windows (direct subnodes of a page) or as subnodes of a window. A prerequisite for this is that the graphic already exists in the system.

You create graphic nodes like you create any other node by using the context menu of the navigation tree, that is, right mouse button or by choosing *Edit* → *Node* → *Create* from the menu.

If you create a separate graphic window for the graphic, the graphic is visible in the Form Painter. Make sure that you have not selected the *Placeholder for graphics* checkbox in the Form Painter settings. You can easily position the graphic on the page using drag and drop.

If you create the graphic as the subnode of an existing node, for example, of a window or a template, it is not displayed in the Form Painter. You cannot position the graphic using Drag & Drop. The graphic is output depending on higher-level nodes.

You make the settings for the name of the graphic, the object and ID, the type of the graphic (black and white or color) and the resolution in the same way as for a background picture. In particular, you can determine the name of the graphic dynamically at runtime by entering a field.

If you do not create the graphic in a separate graphic window of the form but rather as a subnode, you must also determine its horizontal position on the *Output Options* tab.

Exercise 2: Creating Text, Addresses, and Graphics

Exercise Objectives

After completing this exercise, you will be able to:

- Create text nodes with different text types
- Use the field list
- Embed graphics

Business Example

You are an employee of the Fly & Smile travel agency. You are responsible for sending invoices of the booked flights to the respective customers. To include text in your invoice forms, you need to create text nodes for different text types. You need to use field list to include customer addresses in the invoice. In addition, you also need to place your company's logo at the top right corner of the invoice form.

Task 1:

Copy template

Note: Copy template for the form: BC470_TEXTT

Package/Development class (for all exercises): ZBC470_##

Name of the form to be created: ZBC470_##_TEXTS

Model solution: BC470_TEXTS

Application program for testing purposes: SAPBC470_DEMO

1. Copy the template form BC470_TEXTT to ZBC470_##_TEXTS (## is your two-character group number). You cannot use the form for the previous exercise.

Task 2:

Create text nodes

1. Create the following text nodes of the text type *Text element* in the main window:

   -INTRODUCTION - Write a short introduction for your letter.

   -DUMMY_TABLE - Add a few lines as a placeholder for the table that will be inserted here later on.

Continued on next page

-GREETINGS - The greeting form

Task 3:

Use field list for customer address

1. The field WA_CUSTOMERS, which is filled by the application program, contains all the important information about the customer.

   Create a text node (of the text type *Text element*) in the window ADDRESS2 and use the field list to output the following fields of the import parameter WA_CUSTOMERS: FORM, NAME, STREET, POSTCODE, REGION if required, CITY. Do not format the address in accordance with ISO standards at this point.

Task 4:

Define shading for window

1. Define shading for the INFO window with a gray value of 10 percent.

Task 5:

Create text module

1. Create the text module ZBC470_##_FOOTER preferably in a second session. Assign the style BC470. Enter the bank details of the travel agency in a small font size in the text module.
2. Save the text in your package (SAP R/3 4.6C: your development class) ZBC470_##.
3. Embed this text module in the FOOTER window of the form.

Task 6:

Company logo

1. Insert the color graphic BC470_FLY_AND_SMILE, object GRAPHICS, and ID BMAP in a separate graphic window in the top-right corner of the page FIRST.

Task 7:

Optional: Insert include text

1. Insert the SAPscript text BC470_FLY_AND_SMILE, text object TEXT, text ID ADRS as a new text node in the window ADDRESS2. This is a small text for the transparent window of the envelope. Call this text node ADDR_INCL.

Continued on next page

Task 8:

Optional: Determine page numbers

1. Output the current page and the total page number in the form "Page X of Y" in the window PAGE of the page NEXT.

Task 9:

Test result

1. Activate your form. Test your form using the program SAPBC470_DEMO. If required, modify your form so that the page NEXT is also processed.

Solution 2: Creating Text, Addresses, and Graphics

Task 1:

Copy template

Note: Copy template for the form: BC470_TEXTT

Package/Development class (for all exercises): ZBC470_##

Name of the form to be created: ZBC470_##_TEXTS

Model solution: BC470_TEXTS

Application program for testing purposes: SAPBC470_DEMO

1. Copy the template form BC470_TEXTT to ZBC470_##_TEXTS (## is your two-character group number). You cannot use the form for the previous exercise.

   a) Copy template

      See exercise of the previous unit.

Task 2:

Create text nodes

1. Create the following text nodes of the text type *Text element* in the main window:

   -INTRODUCTION - Write a short introduction for your letter.

   -DUMMY_TABLE - Add a few lines as a placeholder for the table that will be inserted here later on.

Continued on next page

-GREETINGS - The greeting form

a) Create text nodes

From the context menu of the main window, choose *Create* → *Text*. On the maintenance screen, change the name to read INTRODUCTION and enter a description. Enter some text in the editor, which is displayed in the bottom part of the maintenance screen if you select the text node to edit it: “Dear...”

So that the other two text nodes in the tree appear after the node INTRODUCTION, create the node DUMMY_TABLE using the context menu of INTRODUCTION and the node GREETINGS using the context menu of DUMMY_TABLE. Alternatively, you can also move the nodes to the desired position using Drag and Drop, which means you keep the left mouse button pressed while moving the nodes. Enter text in the text editor for DUMMY_TABLE and GREETINGS.

Task 3:

Use field list for customer address

1. The field WA_CUSTOMERS, which is filled by the application program, contains all the important information about the customer.

   Create a text node (of the text type *Text element*) in the window ADDRESS2 and use the field list to output the following fields of the import parameter WA_CUSTOMERS: FORM, NAME, STREET, POSTCODE, REGION if required, CITY. Do not format the address in accordance with ISO standards at this point.

   a) Use field list for customer address

   From the context menu of the text node you have just created, choose *Create* → *Text*. On the maintenance screen, change the name and enter a description.

   Show the field list in the bottom left corner of the screen, for example by choosing *Utilities* → *Field list on/off* from the menu.

   Expand the *Import interface* folder by clicking the small triangle to its right with the mouse. Use the same procedure to open the structure WA_CUSTOMERS and move the required fields to the editor using Drag and Drop. Add some paragraphs.

Continued on next page

Task 4:

Define shading for window

1. Define shading for the INFO window with a gray value of 10 percent.

   a) Define shading for window

      Select the INFO window (by double clicking it in the navigation tree or single clicking it in the Form Painter).

      If you are using the SAP Web Application Server 6.10, on the *Output Options* tab page, group box *Box and Shading*, either enter the color black with a saturation of, for example, 10 percent, or choose an appropriate gray level with a saturation of 100 percent.

      If you are using SAP R/3 4.6C, on the *Output Options* tab page, group box Shading, choose the gray level 10 percent.

Task 5:

Create text module

1. Create the text module ZBC470_##_FOOTER preferably in a second session. Assign the style BC470. Enter the bank details of the travel agency in a small font size in the text module.

   a) Create text module

      Open a new session and start the SAP Smart Forms maintenance transaction. Select the *Text module* radio button, enter ZBC470_##_FOOTER, and choose *Create*. The system displays the editor for the new text module. Enter the bank details. The default style SYSTEM, which is initially assigned to each new text module, does not use small font sizes. You must therefore assign another style, that is, BC470. You do this on the *Management* tab page. If you now return to the editor (Text tab), you can select the text with the mouse and use the *S Key Word (Small)* character format.

2. Save the text in your package (SAP R/3 4.6C: your development class) ZBC470_##.

   a) Save your text module by clicking the disk icon. Enter your package (SAP R/3 4.6C: development class).

Continued on next page

3. Embed this text module in the FOOTER window of the form.

   a) Now, return to the first session and choose *Create → Text* from the context menu of the FOOTER window. *Text*. On the maintenance screen, change the name to read FOOTER_TEXT and enter a description. On the *General Attributes* tab page of FOOTER_TEXT, change the text type to *Text module*. Choose Yes in response to the confirmation prompt. The *General Attributes* tab changes so that you can enter ZBC470_##_FOOTER in the *Text name* field. Press Return - this displays the text module in the editor. You cannot directly modify the text module from within the SAP Form Builder.

Task 6:

Company logo

1. Insert the color graphic BC470_FLY_AND_SMILE, object GRAPHICS, and ID BMAP in a separate graphic window in the top-right corner of the page FIRST.

   a) Company logo

      From the context menu of the page FIRST, choose *Create → Graphic*. Enter a name and a description. On the *General Attributes* tab of the graphic window created, enter the name, the object, and the ID. Select *Color Grid Screen (BCOL)*. Press *Return* to update the preview in the Form Painter. Then, move the graphic with the mouse to the top right corner.

Task 7:

Optional: Insert include text

1. Insert the SAPscript text BC470_FLY_AND_SMILE, text object TEXT, text ID ADRS as a new text node in the window ADDRESS2. This is a small text for the transparent window of the envelope. Call this text node ADDR_INCL.

   a) **Optional:** Insert include text

      From the context menu of the ADDRESS2 window, choose *Create → Text*. On the maintenance screen, change the name to read ADDR_INCL and enter a description. On the *General Attributes* tab of ADDR_INCL, change the text type to *Include text*. Choose Yes in response to the confirmation prompt. The *General Attributes* tab changes so that you can enter the text name, text object and text ID in the *Text key group* box. Note that you cannot preview include texts from within the SAP Form Builder.

Continued on next page

Task 8:

Optional: Determine page numbers

1. Output the current page and the total page number in the form "Page X of Y" in the window PAGE of the page NEXT.

   a) **Optional:** Determine page numbers

   Go to the page NEXT. Use the context menu of the window PAGE to create a new text node. Change the name and enter a description. The text type should remain *Text element*. From the field list choose *System fields* → *SFSY*. Then, drag the fields PAGE and FORMPAGES into the editor of the text node. Add the text "Page ... of".

Task 9:

Test result

1. Activate your form. Test your form using the program SAPBC470_DEMO. If required, modify your form so that the page NEXT is also processed.

   a) Test result

   Activate your form by choosing *Form* → *Activate* from the menu.

   To test the form, choose *System* → *Services* → *Reporting*. Enter the name SAPBC470_DEMO, and execute the program (function key F8). On the selection screen, enter the name of your form and execute the program (function key F8).

   If you enter less text in the text node DUMMY_TABLE, the page NEXT is not processed at all. In this case, you must either enter more text or reduce the height of the window MAIN on the page FIRST.

Lesson Summary

You should now be able to:

- Import graphics in SAP R/3 Enterprise
- Create graphics as separate windows or as subnodes of a window

Related Information

- [Enter an optional reference using the URL or CrossReference tag to additional information that learner may find useful. Examples include websites or whitepapers. Delete if not used.]

Unit Summary

You should now be able to:

- Create text nodes
- Insert fields using the field list
- Format fields
- Create addresses from Business Address Services
- Import graphics in SAP R/3 Enterprise
- Create graphics as separate windows or as subnodes of a window

Test Your Knowledge

1. Which of the following options specifies that the text of the second node begins in a new line and the two text nodes are completely independent of each other?

   Choose the correct answer(s).

   - □ A New Line
   - □ B New Paragraph
   - □ C Append Directly
   - □ D Include Text

2. You can insert fields from a field list using _____.

   Fill in the blanks to complete the sentence.

3. The corresponding commands for formatting options for a field are entered directly after the _____ and always in _____.

   Fill in the blanks to complete the sentence.

Answers

1. Which of the following options specifies that the text of the second node begins in a new line and the two text nodes are completely independent of each other?

   Answer: B

   You choose the "New Paragraph" option to specify that the text of the second node begins in a new line. When you choose this option, the two text nodes are completely independent of each other.

2. You can insert fields from a field list using drag and drop.

   Answer: drag and drop

3. The corresponding commands for formatting options for a field are entered directly after the field name and always in uppercase.

   Answer: field name, uppercase

Unit 4

Data in Forms

Unit Overview

In this unit, you will focus on the integration of data into forms. In addition, you will get an overview of the different types of data used in SAP Smart Forms and create global data and its types. Finally, you will be able to define the interface of a form.

Unit Objectives

After completing this unit, you will be able to:

- Define the form interface
- Create global data and global types
- Create field symbols

Unit Contents

Lesson: Integrating Data in Forms

Lesson Overview

In this lesson, you will learn about the form interface, global data and definitions, and field lists. You will be able to explain the concept of field symbols and types.

Lesson Objectives

After completing this lesson, you will be able to:

- Define the form interface
- Create global data and global types
- Create field symbols

Business Example

The Fly & Smile travel agency constitutes a number of independent units located in different parts of the world. Each unit is headed by a business unit head and is operated as an independent entity under the common parent company. You are creating an invoice for a booked flight. You need to enhance the existing invoice form and make it more flexible by inserting fields.

Defining Interface

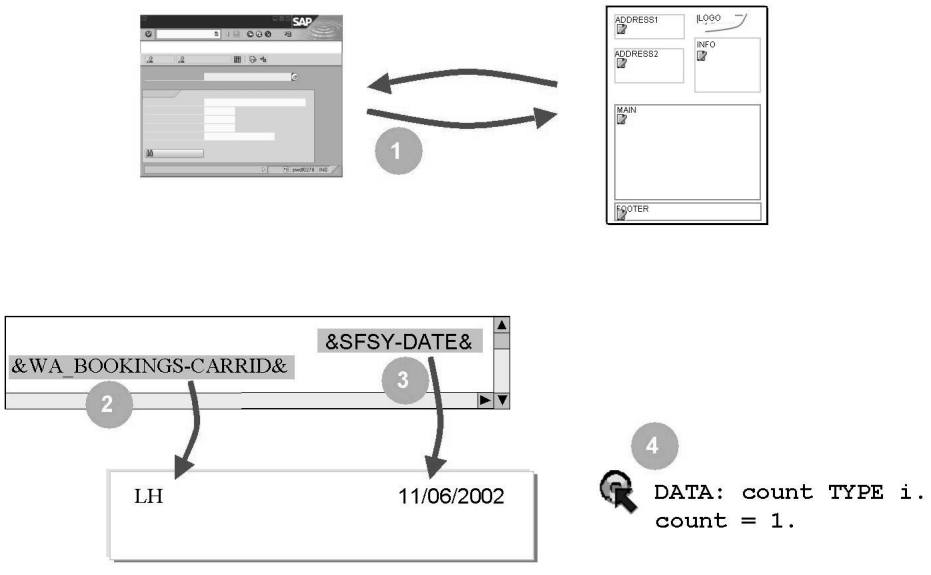


Figure 47: Data in Forms

You have seen several times that you always need fields, that is, variable data in forms, for example, to output today's date, integrate values from a database table into the form, or define conditions for the processing sequence of nodes.

The following types of data exist in an SAP Smart Form:

1. Interface data called parameters. They are passed to the form by the application program and vice versa.
2. Global data, which is known in all nodes of the form.
3. System fields like the page number or the current time.
4. Local data that you create in program line nodes using ABAP statements. This type of data is not dealt with in this training course because it requires general ABAP knowledge not specific to forms.

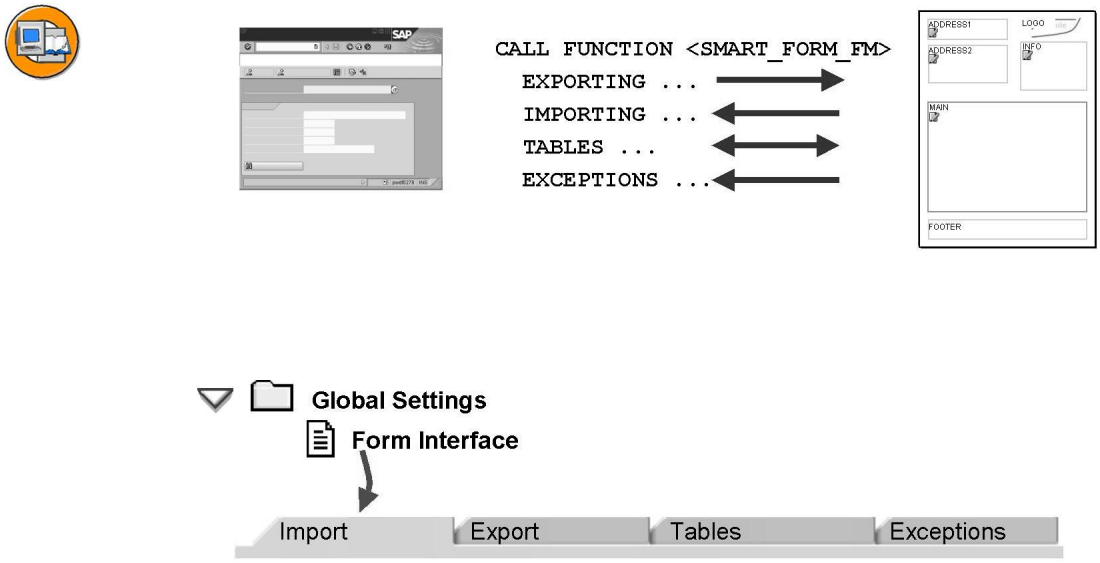


Figure 48: Interface Parameters I

If an application program calls an SAP Smart Form (or more precisely, the generated function module of the form), there must be a way for the program to communicate with this function module. Data must be passed to the form and further returned by the form to the calling program. All data is exchanged through the form interface. You define the interface in the global settings of the Form Builder.

All parameters of the interface are global, which means they are known in all nodes of the form.

The interface of the form/generated function module has the following parameters:

- Import
- Export
- Tables
- Exceptions

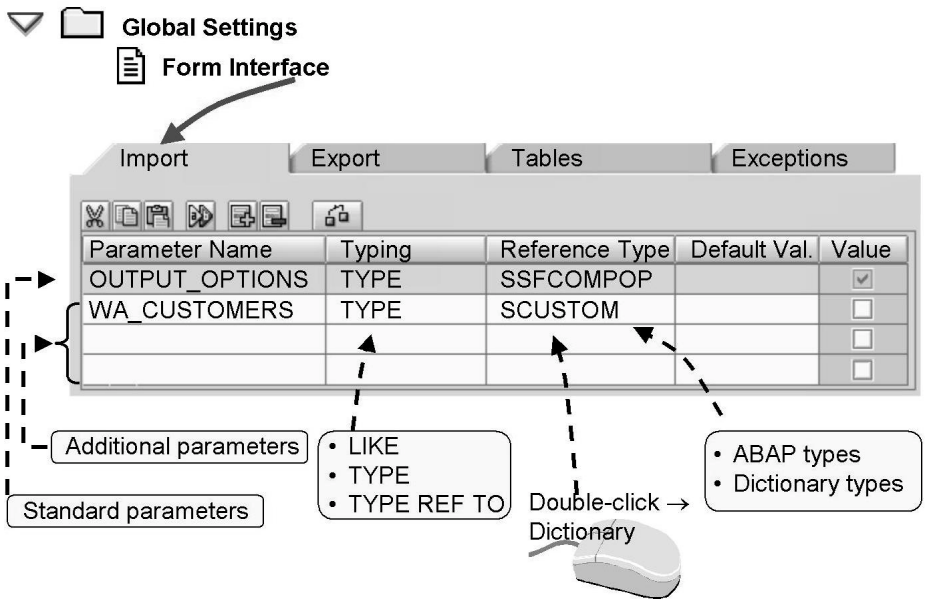


Figure 49: Interface Parameters II

Import parameters are read from the application program. There is a number of default parameters and an undefined number of others of any type. Their fields are not ready for input in the interface. All other parameters differ from form to form and contain values provided by the application program, which means basically everything you want to output in the form. They are particularly important for the correct processing of the form.

Assign names to all other parameters that comply with the ABAP naming conventions, that is, do not contain umlauts, blank spaces, or special characters.

Import parameters must be typed. To do this, you can use TYPE, LIKE, and (for ABAP Objects) TYPE REF TO - similarly as with the DATA statement of an ABAP program.

You can choose one of the following *associated types* (SAP R/3 4.6C: *reference types*):

ABAP types (C, N, D, T, X, I, P, F, STRING, and XSTRING)

Dictionary types such as data elements, structure types, and internal table types

You can set a default value for import parameters in the *Default value* field to make sure they have a value even if they are not filled in the application program. This is particularly useful if you want to test a form without using an application program.

If you check *Pass value*, the system passes a copy of the parameter to the form and not the parameter itself.

As of SAP Web AS 6.10, you can set import parameters as optional.

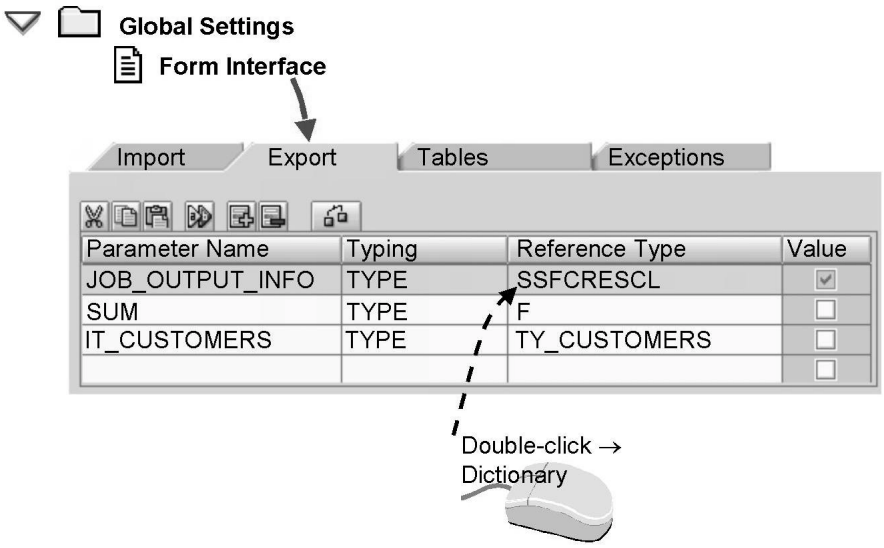


Figure 50: Interface Parameters III

Export parameters are returned to the application program. Again, there is a number of default parameters and an undefined number of others of any type. What has been said about the *parameter names* and the *type assignment* of import parameters is also true for the export parameters.

If you check *Pass value*, the value is only passed at the end of the form function module to the parameter in the application program. However, if you do not check *Pass value*, the value is passed by reference: If export parameters are changed within the form, the original values of the application program are then modified **directly**. If form processing terminates for any reason, any value changes made so far to export parameters are also visible in the application program.

If values are passed by reference, an export parameter is turned into an import export parameter, which means that values can be passed in both directions: From the application program to the form and vice versa.

Tables: You can pass internal tables to the form. For typing, you can refer to a flat table type or (with LIKE) a flat structure type. You can pass nested tables using appropriate export parameters. The values are always passed by reference, which means that value changes in the form directly affect the values of the application program.

Exceptions: Exception parameters are queried in the application program so that the program can respond to errors that occur during form processing.

Creating Global Data and Types

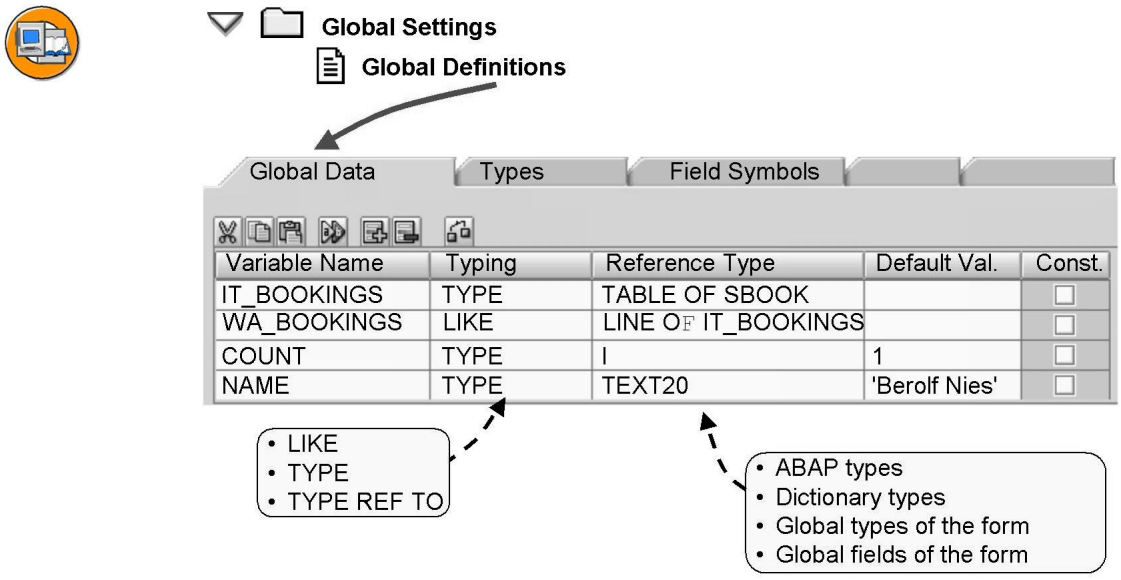


Figure 51: Global Data

Variables that you enter on the *Global data* tab of the global definitions are known in the entire form and can be used as work areas of tables or loops, for example. The best way to insert them into text nodes is to use the field list.

The *associated types* (SAP R/3 4.6C: *reference types*) for typing the global data provide the same possibilities as the interface parameters and additionally:

Global types of the form

Global fields of the form (if referenced with LIKE)

You can set an initial value for variables in the *Default value* field.

If you select the *Constant* checkbox, you generate a global constant instead of a variable, that is, you must specify an initial value that you must not change in the form.

If you have entered a Dictionary reference type, a double-click on the type or element takes you to the ABAP Dictionary where you can obtain information on the definition of the type.

Check your entries with the *Check* pushbutton.

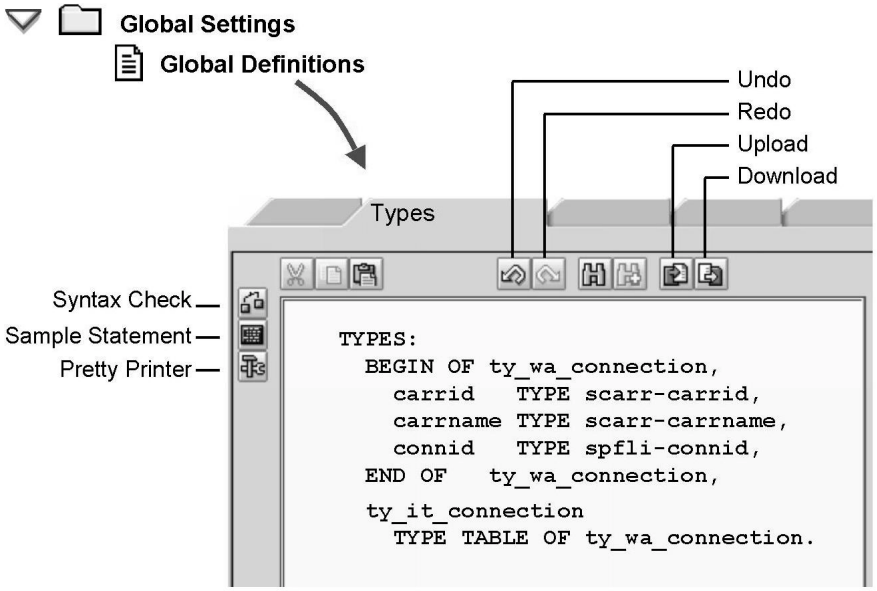


Figure 52: Global Types

The *Types* tab of the global definitions contains an ABAP editor that you can use to define types. You can then use these types in the entire form. In particular, you can type the global data with reference to these types.

Types are defined with the ABAP statement TYPES.

Creating Field Symbols

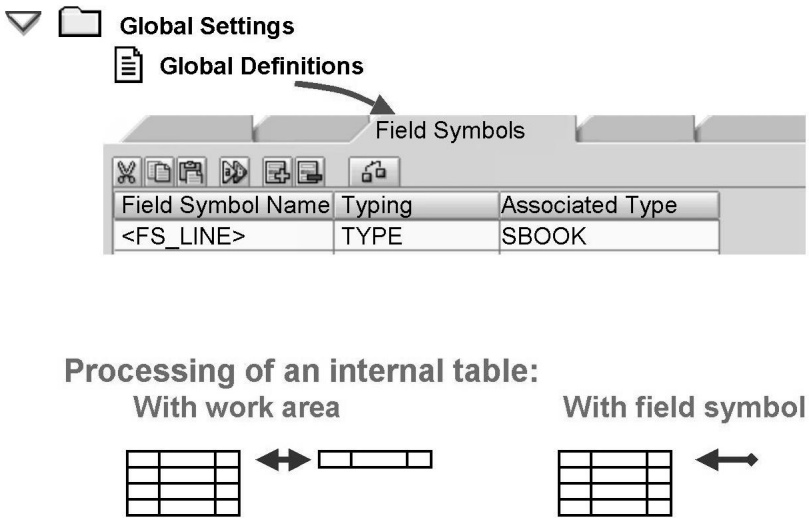


Figure 53: Field Symbols

In the global settings, you can create field symbols. The syntax corresponds to the ABAP command FIELD-SYMBOLS. This means you have to choose a name with an opening and closing angle bracket, a type assignment (TYPE, LIKE, TYPE REF TO), and an associated type (SAP R/3 4.6C: a reference type).

Field symbols are placeholders for data objects. To be more precise, they are de-referenced pointers to data objects. Field symbols must be assigned to a data object at runtime and then contain the value of that object. This allows greater programming flexibility - however, at the expense of a restricted syntax and security check. This might result in runtime errors or wrong data assignments. You should therefore use field symbols only if other ABAP statements do not provide an adequate solution for implementing the operation you want.

A good way to use field symbols is in loops over internal tables. The ABAP command LOOP AT <itab> INTO <workarea> copies each line of the internal table into the work area. If you use field symbols instead (LOOP AT <itab> ASSIGNING <<fieldsymbol>>, the rows need not be copied since the field symbol directly accesses the row contents of the internal table. Using field symbols for large tables in particular can therefore be better for runtime than work areas. SAP Smart Forms support field symbols for loops and tables.

Exercise 3: Data in Forms

Exercise Objectives

After completing this exercise, you will be able to:

- Create interface parameters
- Create global variables and types

Business Example

You are working with the Fly & Smile travel agency.

As an employee, you are responsible for creating invoices for booked flights and send them to the respective customers. You are in the process of creating an invoice form. You need to extend your existing invoice form and make it more flexible by inserting fields. You need to perform the necessary preparatory steps for the output of the booking table.

Task 1:

Copy template

Note: Copy template for the form: BC470_TEXTS

Package/Development class (for all exercises): ZBC470_##

Name of the form to be created: ZBC470_##_DATAS

Model solution: BC470_DATAS

Application program for testing purposes: SAPBC470_DEMO

1. Copy the form that you used in the last task (ZBC470_##_TEXTS) to ZBC470_##_DATAS. Alternatively, copy the copy template (BC470_TEXTS).

Task 2:

Enhance interface

1. On the selection screen of the print program, you can decide whether you want to embed the company logo in color or in black and white. The parameter is called COLOR and is a four-digit character field.

   Add an import parameter called COLOR of the Dictionary type CHAR4 to the interface of your form.

   Use this parameter for the company logo and make the necessary change on the General attributes tab of the graphic.

Continued on next page

2. Perform the necessary preparatory steps to output a real booking table in the next exercise: Create a table called IT_BOOKINGS in the interface.

   Refer to the Dictionary type TY_BOOKINGS.

   Perform a local check.

   TY_BOOKINGS is a table type. What is the name of the underlying structure? Determine the name of the field in this structure which contains the price of the booking in foreign currency.

   Hint: Double-clicking TY_BOOKINGS takes you to the ABAP Dictionary.

Task 3:

Create global variable

1. Also in preparation for the next exercise, create an appropriate work area for the table called WA_BOOKINGS. Type this work area with reference to the structure that you have determined earlier (2.2.3).

Task 4:

Create and use global constant

1. Create a global constant called CLERK and type this constant with reference to the Dictionary data element CHAR15. Assign a fine-sounding employee name to this constant.
2. Use this constant on page FIRST, window INFO, and text node INFO_TEXT instead of the clerk entered there.
3. Add the name of the clerk to the greeting form using text node GREETINGS of the main window by inserting the constant.

Task 5:

Optional: Create global type

1. Create a global type called CHAR_FIFTEEN for a 15-digit character field. Perform a local check. Type the constant CLERK with reference to that type instead of to CHAR15.

Continued on next page

Task 6:

Test

1. Activate your form and test it using the program SAPBC470_DEMO. Verify if the clerk is displayed correctly and if the color/black and white selection for the graphic on the selection screen functions properly.

Hint: If the program terminates, you might not have used the correct interface specifications. Both, the names and types for interface parameters must be identical in the form and the application program. If required, read the text of the error message to track down the error in your program.

Solution 3: Data in Forms

Task 1:

Copy template

Note: Copy template for the form: BC470_TEXTS

Package/Development class (for all exercises): ZBC470_##

Name of the form to be created: ZBC470_##_DATAS

Model solution: BC470_DATAS

Application program for testing purposes: SAPBC470_DEMO

1. Copy the form that you used in the last task (ZBC470_##_TEXTS) to ZBC470_##_DATAS. Alternatively, copy the copy template (BC470_TEXTS).

   a) See the exercise in the unit SAP Form Builder.

Task 2:

Enhance interface

1. On the selection screen of the print program, you can decide whether you want to embed the company logo in color or in black and white. The parameter is called COLOR and is a four-digit character field.

   Add an import parameter called COLOR of the Dictionary type CHAR4 to the interface of your form.

   Use this parameter for the company logo and make the necessary change on the General attributes tab of the graphic.

   a) Choose *Global Settings* → *Form interface* and go to the *Import* tab. Add COLOR TYPE CHAR4 at the end of the list.

      In the navigation tree of the page FIRST, select the graphic LOGO. On the maintenance screen, click the General attributes tab and select the Determine dynamically (BMON, BCOL) radio button. Enter `&COLOR&`. At the runtime of the function module generated, this variable is assigned the value of the selection screen. Depending on the option you choose, the company logo is printed in color or in black and white.

2. Perform the necessary preparatory steps to output a real booking table in the next exercise: Create a table called IT_BOOKINGS in the interface.

Continued on next page

Refer to the Dictionary type TY_BOOKINGS.

Perform a local check.

TY_BOOKINGS is a table type. What is the name of the underlying structure? Determine the name of the field in this structure which contains the price of the booking in foreign currency.

Hint: Double-clicking TY_BOOKINGS takes you to the ABAP Dictionary.

a) In the navigation tree, choose *Global settings* → *Form interface*. Click the *Tables* tab. Add the following at the end of the list: IT_BOOKINGS TYPE TY_BOOKINGS.

Click the Check pushbutton of the maintenance screen.

The structure is called SBOOK. The name of the field which contains the price of the booking in foreign currency is FORCURAM.

Task 3:

Create global variable

1. Also in preparation for the next exercise, create an appropriate work area for the table called WA_BOOKINGS. Type this work area with reference to the structure that you have determined earlier (2.2.3).

   a) In the navigation tree, choose *Global settings* → *Global definitions*. Enter the following on the Global data tab: WA_BOOKINGS TYPE SBOOK.

Task 4:

Create and use global constant

1. Create a global constant called CLERK and type this constant with reference to the Dictionary data element CHAR15. Assign a fine-sounding employee name to this constant.

   a) In the navigation tree, choose *Global settings* → *Global definitions*. Enter the following on the *Global data* tab: CLERK TYPE CHAR15. Select the *Constant* checkbox at the end of the line.

Continued on next page

2. Use this constant on page FIRST, window INFO, and text node INFO_TEXT instead of the clerk entered there.

   a) Select the text node INFO_TEXT in the navigation tree. Delete the existing clerk in the editor and insert the global field CLERK using the field list. The field is inserted as `&CLERK&`.

3. Add the name of the clerk to the greeting form using text node GREETINGS of the main window by inserting the constant.

   a) Repeat these steps for the text node GREETINGS. This node is in the MAIN window.

Task 5:

Optional: Create global type

1. Create a global type called CHAR_FIFTEEN for a 15-digit character field. Perform a local check. Type the constant CLERK with reference to that type instead of to CHAR15.

   a) In the navigation tree, choose *Global settings* → *Global definitions*. Enter the following on the *Types* tab: TYPES CHAR_FIFTEEN(15).

   In the navigation tree, choose *Global settings* → *Global definitions*. Change the type on the Global data tab: CLERK TYPE CHAR_FIFTEEN.

Task 6:

Test

1. Activate your form and test it using the program SAPBC470_DEMO. Verify if the clerk is displayed correctly and if the color/black and white selection for the graphic on the selection screen functions properly.

Hint: If the program terminates, you might not have used the correct interface specifications. Both, the names and types for interface parameters must be identical in the form and the application program. If required, read the text of the error message to track down the error in your program.

   a) Test: See previous exercise.

Lesson Summary

You should now be able to:

- Define the form interface
- Create global data and global types
- Create field symbols

Unit Summary

You should now be able to:

- Define the form interface
- Create global data and global types
- Create field symbols

Test Your Knowledge

1. Identify the parameters of a generated function module or a form interface.
   Choose the correct answer(s).
   - □ A Import
   - □ B Export
   - □ C Global Data
   - □ D Tables

2. The best way to insert global definitions into text nodes is by using the _____.
   Fill in the blanks to complete the sentence.

3. Global data from a form can be accessed from outside the form (e.g. a program).
   Determine whether this statement is true or false.
   - □ True
   - □ False

Answers

1. Identify the parameters of a generated function module or a form interface.

   Answer: A, B, D

   The interface of a form or a generated function module has the following parameters: Import, Export, tables, and Exceptions.

2. The best way to insert global definitions into text nodes is by using the field list.

   Answer: field list

3. Global data from a form can be accessed from outside the form (e.g. a program).

   Answer: False

   Only the form interface can be used for a data exchange between a form and a program.

Unit 5

Tables and Templates

Unit Overview

In this unit, you will learn how to use Table Painter to create tables and templates. You will also learn how to work with tables and templates.

Unit Objectives

After completing this unit, you will be able to:

- Identify and create tables and their line types
- Process tables
- Output data in tables and sort a table
- Create control levels in a table
- Perform calculations in tables
- Identify the template node type of an SAP Smart Form
- Draw table templates using a Table Painter
- Define template layout and output contents in templates
- Create templates

Unit Contents

Lesson: Tables

Lesson Overview

In this lesson, you will learn about tables, which are one of the nodes of an SAP Smart Form. You will be able to identify the line types of tables by defining different attributes of table cells. In addition, you will learn how to process and output data in tables. Finally, you will learn about creating control levels and performing calculations in tables.

Lesson Objectives

After completing this lesson, you will be able to:

- Identify and create tables and their line types
- Process tables
- Output data in tables and sort a table
- Create control levels in a table
- Perform calculations in tables

Business Example

The Fly & Smile travel agency constitutes a number of independent units located in different parts of the world. Each unit is headed by a business unit head and operates as an independent entity under the common parent company. You are responsible for creating invoices, which is a step-wise procedure, of the booked flights for the respective customers. You need to create tables containing various data, such as customer information and details of bookings. This data is used for setting up an invoice form.

Introducing Tables

The table function has existed since SAP R/3 4.6C but was changed for SAP Web Application Server 6.10. You can still use and edit forms with old table types, but not create them. Tables of the old type have their own icon.

This lesson only describes the procedure as of SAP Web Application Server 6.10. The procedure with SAP R/3 4.6C is described in the Appendix.

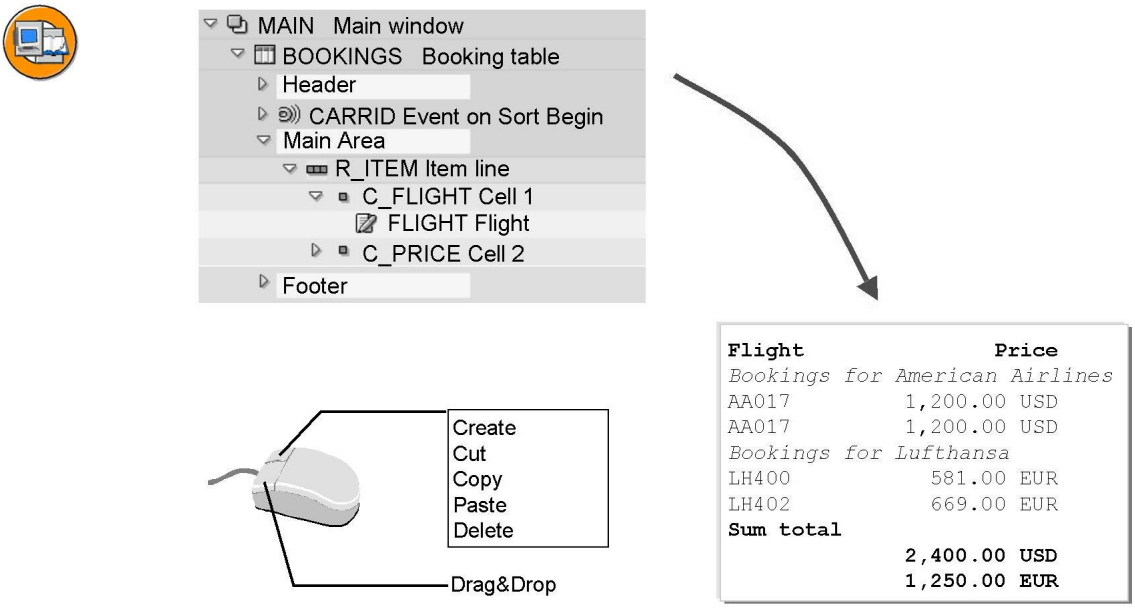


Figure 54: Table: Overview

Forms are frequently used to output data in tables. Tables in SAP Smart Forms are subnodes of windows and are created like all other subnodes using the context menu (right mouse button) of the navigation tree.

Because the length of tables is dynamic, you should only use them in main windows. Tables might get truncated in secondary windows.

Tables provide functions to output headers and footers, sort the table, and calculate subtotals.

Creating Tables

For each line type:

- **Definition of number and width of cells**
- **Height dynamic**
- **Box/Shading (also cell by cell)**

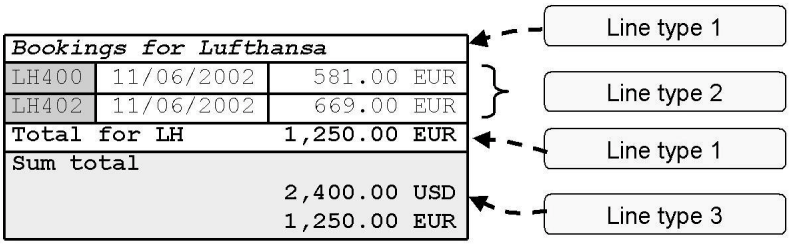


Figure 55: Line Types

Before you can fill tables with text, you must determine the table width and define line types on the *Table* tab of the maintenance screen. You also specify how many cells are to be in one table line and the width of these cells. The height is determined automatically at runtime. For simple applications, a single line type is sufficient. However, you can also create different types for hierarchical or multi-level tables. You do this, for example, if you want to print the different bookings for a flight in the next lines or if you want to use subtotals.

In the output options of the table lines, you specify which line types are to be used when.

You can maintain line types graphically or alphanumerically. If maintained alphanumerically boxes and/or shading cannot be defined.

For each line type, define the following parameters:

- Number and width of cells
- Height dynamic
- Box/Shading (also cell by cell)

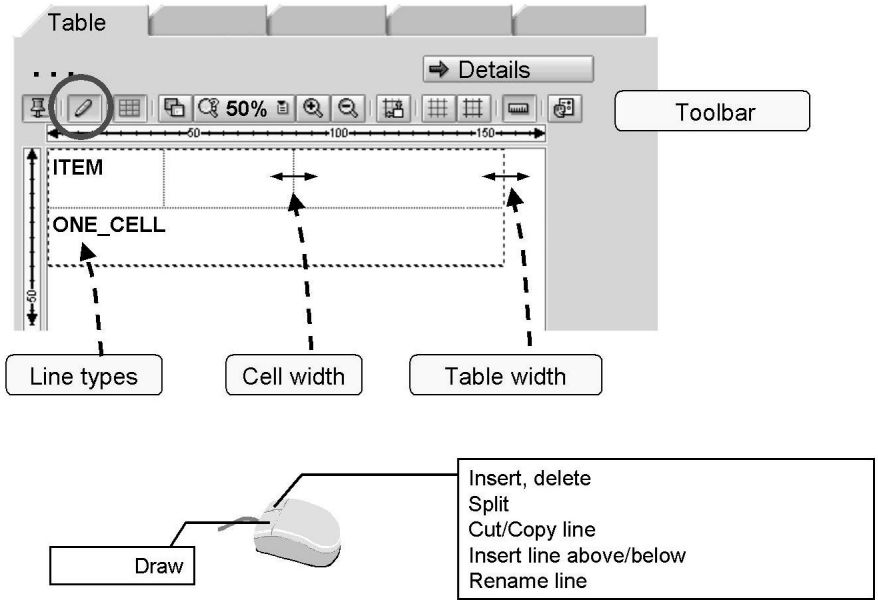


Figure 56: Line Types in the Table Painter

Line types are maintained on the Table tab page using the Table Painter as standard.

To edit line types, the **Draw Lines and Columns** pushbutton must be activated.

You insert a **new line** by horizontally dragging the mouse pointer, which is in the shape of a pencil, at the desired position. Alternatively, you can use the context menu (right mouse button *Insert → Empty line above/below)*. To assign a new name to a line, position the mouse on it and choose *Rename Line* from the context menu.

You insert a **new cell boundary** by vertically dragging the mouse pointer at the desired position while keeping the left mouse button pressed. Alternatively, you can divide the cell in which the mouse pointer is positioned using the context menu (right mouse button *Split → Cell*).

You change the **width of a cell** by placing the mouse on a cell boundary and dragging it to the desired position while keeping the left mouse button pressed. Notice that the mouse pointer assumes the shape of a double arrow.

You change the **table width** by placing the mouse on the right table border and dragging it to the desired position while keeping the left mouse button pressed.

To **position the whole table** within the window, deactivate the *Draw Lines and Columns* pushbutton and drag the table to the required position with the left mouse button. Notice that the mouse pointer is in the shape of a double arrow.

You cannot set the **height of individual line types** because the height is determined dynamically at application program runtime, depending on the data that is output.

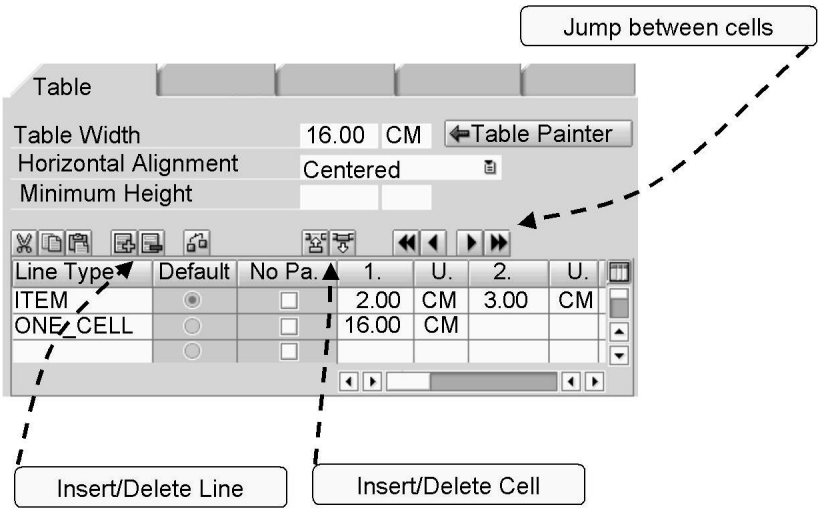


Figure 57: Line Types in the Detail View

Instead of entering the line types in the Table Painter, you can define them numerically in the detail view. The entries you make in the Table Painter are automatically copied to the alphanumeric overview and vice versa.

The following information is required for line types:

- Name.
- Protection against page breaks. Since the height of a line type is dynamic, a page break might occur at runtime. If you select this option, all nodes of a table line of this line type are output together on one page.
- Number and width of the cells.
- Default type: Without relevance.
- The table width must be identical to the total width of all cells for each line type.

You use the *horizontal alignment* to determine how the table is to be aligned with reference to the window margin. You can choose between *Left*, *Right*, and *Centered*. If you choose *Left* or *Right*, you can optionally specify a distance from the respective window margin.

As of SAP Web Application Server 6.20, there is another input field: *Minimum Height*. If you enter a value in this field, the table is started on the next page if the minimum height is no longer available in the current window.

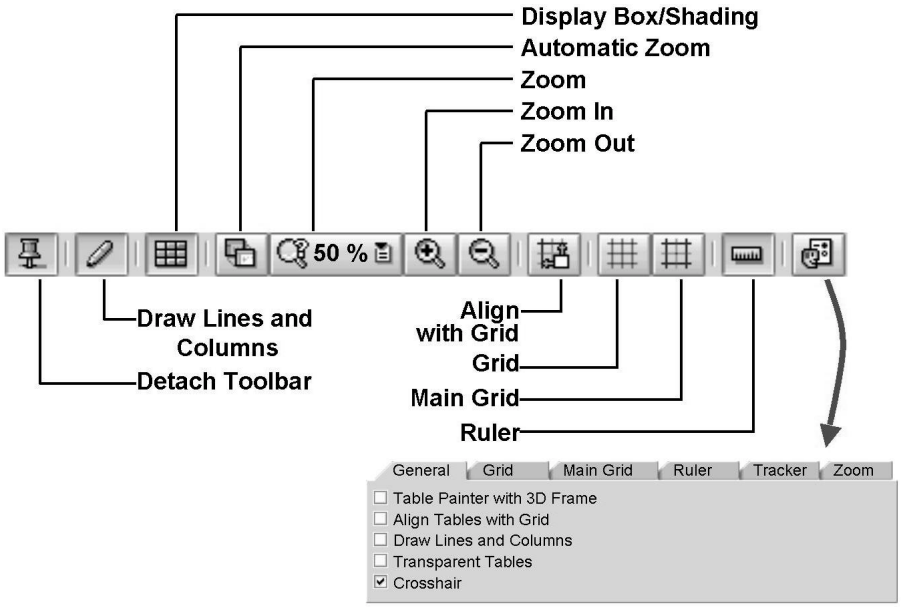


Figure 58: The Table Painter: Toolbar

The most important settings, which are basically the same as those in the Form Painter, are displayed in the detachable toolbar of the Table Painter. To display further options, choose *Utilities* → *Settings* → *Table Painter* tab, or choose the right pushbutton in the toolbar.

If the *Draw lines and columns* checkbox is selected, you can draw the cells of the line types directly with the mouse. If you do not select this checkbox, you can move the table to the left or to the right within the window width - provided you have not chosen *Centered* as the horizontal alignment of the table in the detail view.

Several zoom options are available to adjust the display. The most comfortable option is the *Automatic zoom*.

To draw cells easily, you can display a grid and/or the main grid. You can also make a setting in the Table Painter that ensures that vertical cell boundaries are automatically aligned with the grid when you move them with the mouse. You can set the step size of both grids. The crosshair cursor, which you can set instead of the normal mouse pointer on the *General* tab of the Table Painter settings, also facilitates the drawing of cells.

The *Tracker* tab of the Table Painter settings allows you to determine how the table is to be highlighted against the background.

To hide the toolbar, choose *Utilities* → *Settings* → *Table Painter* tab, and deselect the *Toolbar* checkbox.

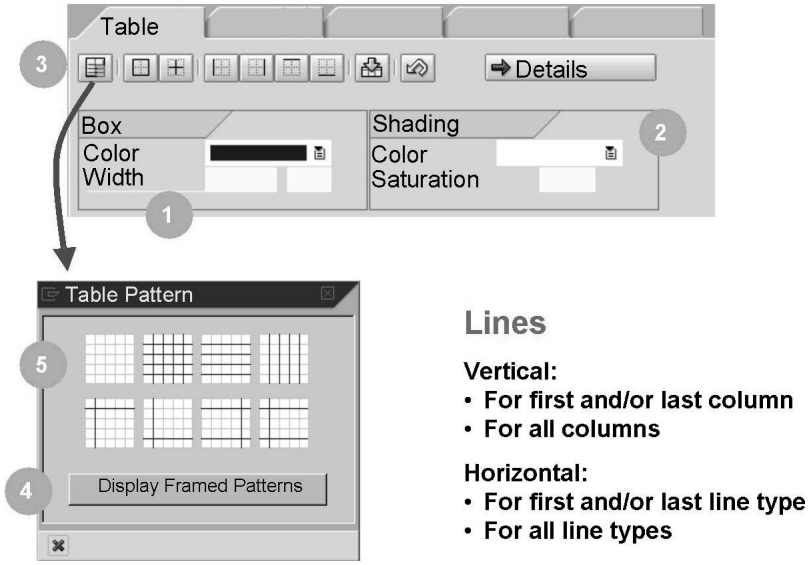


Figure 59: Box and Shading: Pattern

If you want to frame and/or shade your table, you can choose a pattern on the Table tab page in the Table Painter. To do this, proceed as follows:

1. Choose the required **box color** from the list of available colors and set the **width**.
2. Alternatively, you can choose a **color** and a **degree of saturation** for the **shading**. This color is used for the whole table.
3. Choose the **Select Pattern** pushbutton.
4. Decide whether you want the whole table to be framed or not (**Display Framed Pattern** pushbutton).
5. Select the pattern you want to use by clicking it with the mouse. You can choose whether the first, the last, or all columns are to be separated by vertical gridlines. You can also choose whether the first, the last, or all line types are to be separated by horizontal gridlines.

If you use patterns, you should arrange your line types in the sequence in which they are to be output, that is, first line type for the header, then the main area, and finally the footer.

If you want to use tables with the old style (SAP R/3 4.6C), note that table patterns refer to actual table lines and not to line types, as described above.

Lines, that is, gridlines can be used as:

Vertical gridlines:

- For first and/or last column
- For all columns

Horizontal gridlines:

- For first and/or last line type
- For all line types

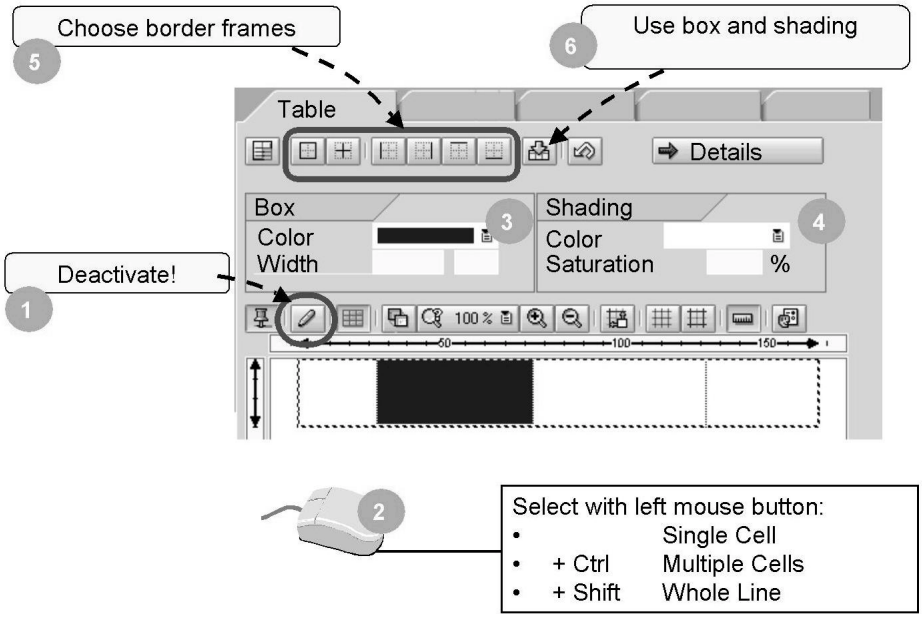


Figure 60: Box and Shading: Single Cells

To add lines and/or shading to individual cells and lines of tables, proceed as follows:

1. Deactivate the **Draw Lines and Columns** pushbutton.
2. In the Table Painter, **select** with the left mouse button the cell(s) for which you want to set a frame and/or shading. To select multiple cells, click the left mouse button in all required cells while holding down the Control button (Ctrl) on your keyboard. To select a whole line, position the mouse pointer in the line and click with the left mouse button whilst holding down the Shift key on your keyboard. Press Ctrl-A to select the whole table.
3. Choose the required **box color** from the list of available colors and set the **width**.
4. Alternatively, you can choose a **color** and a **degree of saturation** for the **shading**.
5. Choose the **page(s)** of the selected cells to which a line is to be added. Repeated clicking switches between the line being shown/hidden. Note that shading is not visible until you deselect.
6. If you want to add the same color and frame to other cells, all you have to do is select the cell as described above and choose the *Use Shading* pushbutton.

Select with left mouse button:

- Single Cell
- + Ctrl Multiple Cells
- + Shift Whole Line

Working With Tables

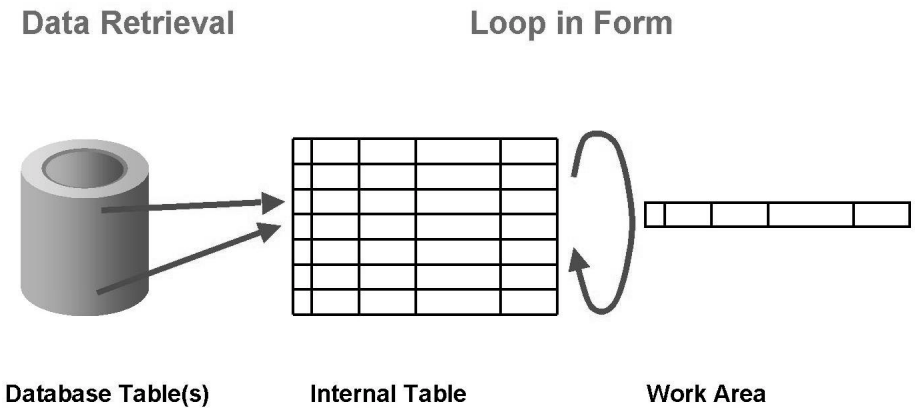


Figure 61: Tables: Technical Background

From a technical point of view, a table in an SAP Smart Form is filled by processing an internal table on a line-by-line basis. This is referred to as a *loop*. The respective lines are copied into a work area that has the same structure as the table. The internal table must be filled beforehand in the application program or in the form. The data is taken from database tables. If the data is read in the application program, the internal table must be defined in the interface of the SAP Smart Form.

If the internal table is very large, it is recommended that you process the table using field symbols instead of a work area since field symbols can directly access the individual lines and therefore the lines do not need to be copied.

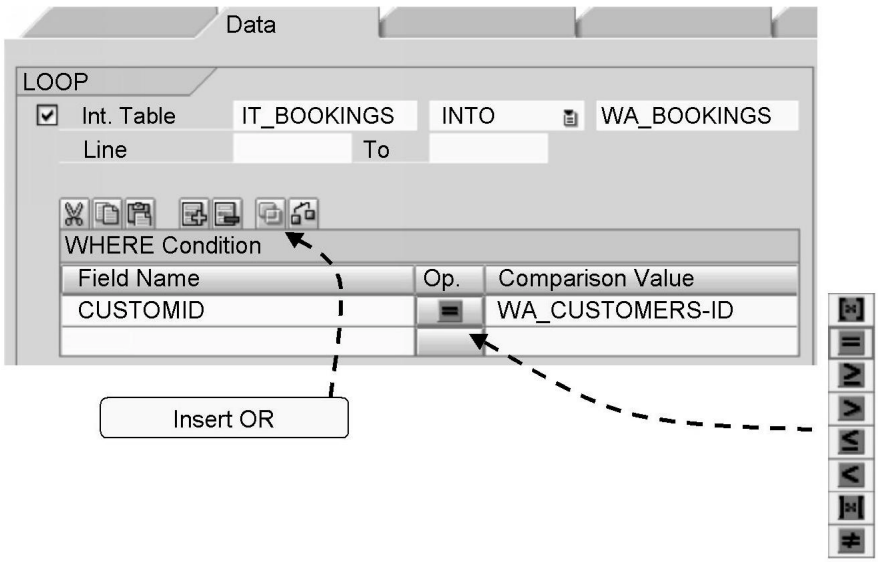


Figure 62: Filling Tables

After defining the table design on the *Table* tab, you should determine how the table is to be processed. You do this on the *Data* tab.

Select the *Internal table* checkbox and enter a name for the table and for the work area. Both the internal table and the work area must be known in the form. This means that they must have been defined from the interface or as a global field. If you do not set the *Internal table* indicator, no loop processing takes place. This might be useful, for example, if you want to output text in parallel columns.

Possible assignment types are *into* and *assigning*. If you use *into*, the lines are copied from the table into the work area. If you use *assigning*, the lines are assigned to a field symbol. If you want to use tables with header lines, enter the name of the table as the work area.

It is possible to use only a specific line range of the internal table. To do this, specify the lines in the fields *Line ... to...*

You can also use logical conditions to determine the lines of the internal table that need to be processed. This corresponds to the WHERE clause of the ABAP command LOOP AT <itab>. Enter the name of a field of the work area, a relational operator, and the comparison value or comparison field. You can use all relational operators that you know from normal selection screens: *With/without pattern, Equal to, Not equal to, Greater than or equal to, Greater, Less than or equal to, and Less.* If you do not enter an operator, *Equal to* is used automatically. You link multiple conditions with AND, but you can also use the OR pushbutton.

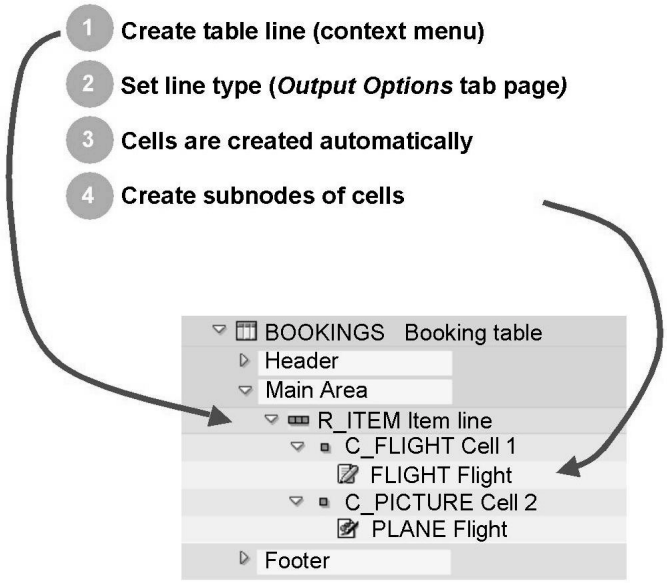
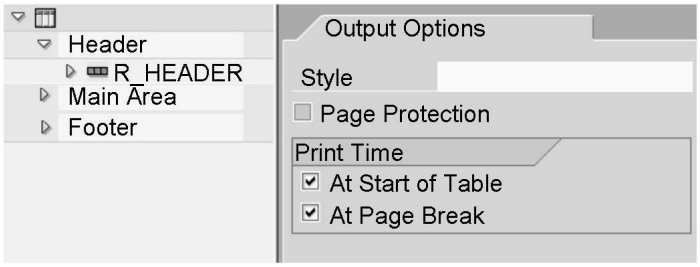


Figure 63: Contents in Table Lines

To output contents in a table, proceed as follows:

1. Click with the right mouse button on the relevant area (header, main area, or footer) and choose **Create Table Line** from the context menu.
2. On the *Output Options* tab page for the inserted line, enter a title and a description. Choose an appropriate **line type** from the list.
3. The number of cells that you defined for the selected line type are inserted automatically in the navigation tree.
4. You now create the actual contents, for example text or graphics, as normal **subnodes of cells**. Since the height of cells is dynamic, you can create more than one subnode for a cell. Page breaks might then occur automatically.



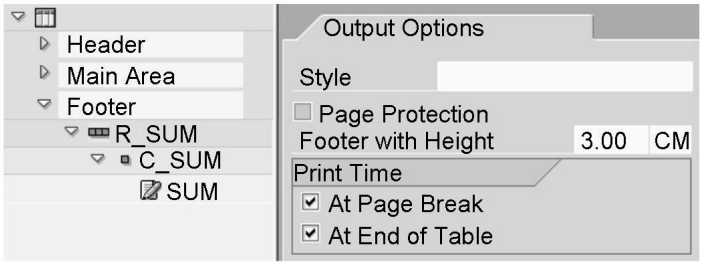


Figure 64: Headers and Footers

You can output contents for headers and footers in the same way as for the main area of a table. Whereas the main area might be used for items, for example, the header might be used for column headers and the footer for page totals and sum totals. The subnodes for the three areas are created automatically as soon as you create the table.

On the *Output Options* tab page for the header, you specify whether and where the header is to be output: At the start of the table and/or after a page break.

The same applies for the footer: You determine on the *Output Options* tab page whether the footer is to be processed at the end of the table and/or before a page break. You must also specify a height for the footer to enable the form processor to reserve sufficient space.

If all three areas appear on one page, they are processed from top to bottom, that is, first the header, then the main area, and finally the footer.

Note: Page protection for multiple table lines is only possible as of SAP Web AS 6.20. You have to create a folder, set page protection for this folder (on the *Output Options* tab page), and move the relevant table lines to it.

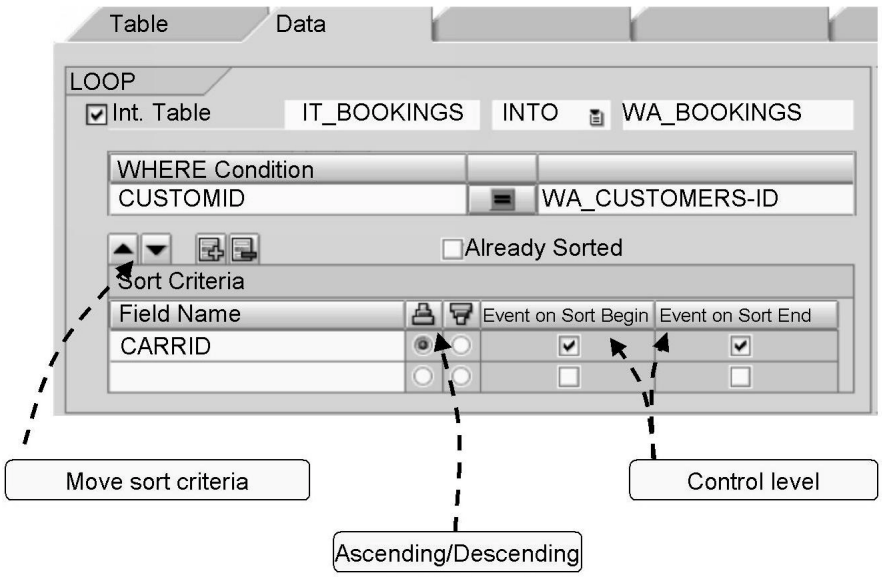


Figure 65: Sorting a Table

You can also sort the internal table within the form. To do this, enter the names of the fields to be used as the *Sort criteria*. The order of the fields in this list determines the sort sequence. You can subsequently change the sort sequence by placing your cursor on a field and moving it up or down a line by clicking one of the two black triangles displayed above the sort criteria. To the right of the field, you can choose whether the table is to be sorted in ascending or descending order.

For technical reasons, the system cannot recognize whether the internal table has already been sorted, such as in the data retrieval program. If this is the case, you still have to enter the sort criteria, but select the *Already sorted* checkbox otherwise the table will be sorted again.

Sorting is required for subtotals and subheadings.

Data Records in Internal Table:

AA	017	12/16/2002	1,200.00 USD
AA	017	01/07/2003	1,200.00 USD
AA	026	03/31/2003	1,400.00 USD
LH	400	11/17/2002	581.00 EUR
LH	400	12/19/2002	581.00 EUR

Table in Form:

Bookings for AA

```
017  12/16/2002  1,200.00 USD
017  01/07/2003  1,200.00 USD
026  03/31/2003  1,400.00 USD
Total for AA     3,800.00 USD
```

Bookings for LH

```
400  11/17/2002    581.00 EUR
400  12/19/2002    581.00 EUR
Total for LH     1,162.00 EUR
```

Figure 66: Control Levels I

Tables are often not output in exactly the same structure in which they are filled. For example, it should be possible to group data records and to output subheaders or subtotals. Grouped data records that have certain identical values are called *control levels*. SAP Smart Forms enable you to create any number of control levels in a table. In the above example, there is a control level for airlines.

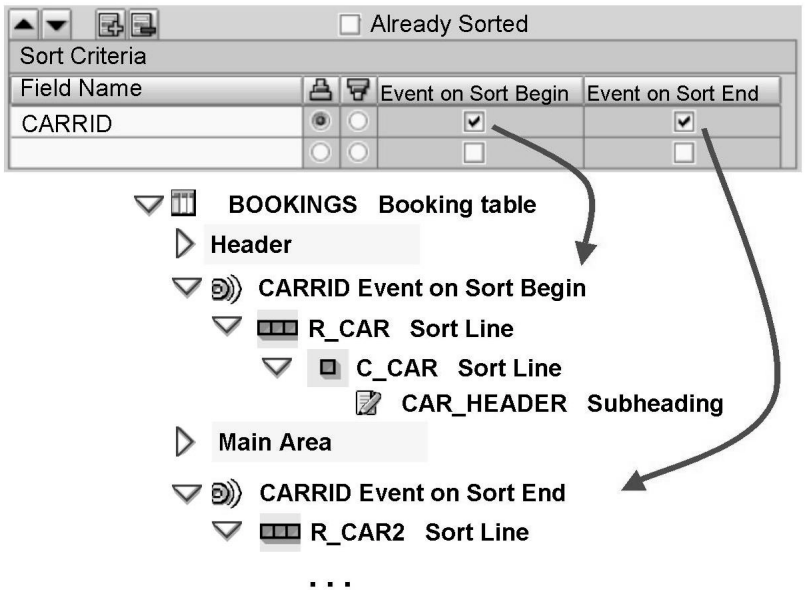


Figure 67: Control Levels II

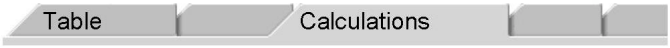
If you select *Event on Sort Begin* and/or *Event on Sort End* for a sort criterion, the corresponding control levels are inserted into the navigation tree of the table. A control level contains all records of the internal table that have the same value in the sort field. In the example above, all records of an airline carrier belong to one control level.

You decide what is output in the control levels. You can insert table lines as subnodes of the sort levels and fill the cells with contents for output, such as program lines for complex subtotal calculations or text nodes.

The node of a control level, called an event node, only has one tab page on the maintenance screen, the *Output options* tab. You can only set a style on this tab page.

You can define control levels for all sort fields. This means that you can set up a hierarchical table that contains, such as one control level for airline carriers and one for flight connections.

You cannot create control levels directly as nodes in the navigation tree. You must always follow the procedure described: Determine the sort criteria and then select the *Event on Sort Begin* and/or *Event on Sort End* checkbox.



Flight	Booking	Luggage
AA 0017	47001	13.2 KG
AA 0017	47002	25.0 KG
AA 0017	47003	17.2 KG
AA 0017	47004	11.7 KG
AA 0017	47005	19.5 KG
AA 0017	47006	22.9 KG
AA 0017	47007	10.0 KG
Total Number:		7
Total Luggage:		119.5 KG
Average Luggage:		17.1 KG

Figure 68: Calculations in Tables I

As of SAP Web Application Server 6.10, there is another tab page for tables: *Calculations.* On this tab page, you can specify that different calculations are to be performed when the table is processed, without having to enter code for them.

You can configure the system so that all fields of a table are automatically counted and totals and average values are calculated from all numeric fields.

Note: Calculations do not take account of currencies or units of measure. If you have prices in different currencies for flight bookings, such as Euros and US Dollars, you cannot use the *Calculations* tab page for the prices since there is no automatic conversion. You need to perform the calculations manually using the program line nodes.

Operation	Field Name	Target Field Name	Event	For Field Name

Initialization	Reset	For Field Name
☐		

Operation:
- Mean value
- Number
- Total

Field Name: From where is the total/average to be calculated?

Target Field Name: Contains variable for result

Event:
- Before loop (at start of main area)
- After loop (at end of main area)
- Before sorting
- After sorting

Initialization: Target field (once before table processing)

Reset: Target field (for a new level)

Figure 69: Calculations in Tables II

There are three possible operations for calculations:

- *Number*: The target field is increased by 1 for each table line.
- *Total*: The value of the field entered in the *Field Name* column is added to the value of the target field.
- *Mean value*: The values of the field entered in the *Field Name* column are used line-wise to calculate the mean value, which is put in the target field.

You choose the events *before loop* or *after loop* if a calculation is to be performed for every processed line of the table.

You use the events *before sorting* and *after sorting* primarily to count sublevels. If you choose one of these events, you have to enter the sort criterion, that is, the variable of the control level in the *For Field Name* field without the name of the output area, for example, CARRID and not WA-CARRID.

If you check the *Initialization* checkbox for calculation, the target field is reset at the start of the table.

Normally, all values in a column are used for calculations. If you choose the entry *Sort Criterion* for *Reset* and then enter a field in the *For Field Name* column, the corresponding counter variables are reset each time a change occurs as soon as the content of the sort criterion changes - therefore at every control level change. If you want to use the *Reset* option, make sure to have checked *Event on Sort Begin* on the *Data* tab page.

Operation:

- Mean value
- Number
- Total

Field Name: From where is the total/average to be calculated?

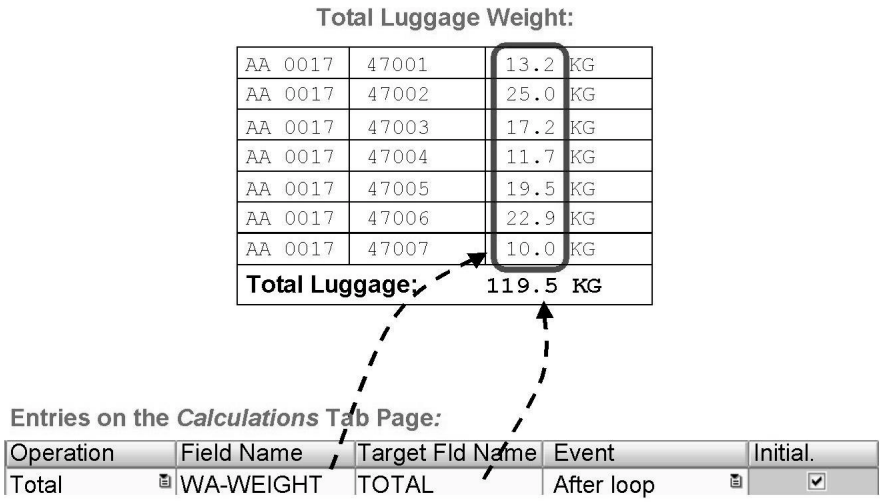
Target Field Name: Contains variable for result

Event:

- Before loop (at start of main area)
- After loop (at end of main area)
- Before sorting
- After sorting

Initialization: Target field (once before table processing)

Reset: Target field (for a new level)



→ Output of TOTAL, for example, after table

Figure 70: Calculations in Tables: Example I

Example: Calculation of total luggage weight:

- Prerequisite: You loop in a work area (here: WA) that has a field for the luggage weight (here: WEIGHT). You require a global help variable that is suitable for the luggage weight of the work area (here: TOTAL).
- Settings on the *Calculations* tab page, see above. If you select the *Initialization* checkbox, the field TOTAL is set to zero before processing.

Number of bookings for each airline carrier:

AA	017	12/16/2002	13.2 KG
AA	017	12/31/2002	25.0 KG
AA	017	01/07/2003	17.2 KG
AA	026	02/12/2003	11.7 KG
AA	026	03/31/2003	19.5 KG
Bookings for AA:			**5**
LH	400	11/17/2002	22.9 KG
LH	400	12/19/2002	10.0 KG
Bookings for LH:			**2**

Entries on the *Calculations* Tab Page:

Operation	Trgt Fld Name	Event	Reset	For Fld Name
Number	CNT	Before loop	Sort criterion	CARRID

→ Output of CNT in node *CARRID Event on Sort End*

Figure 71: Calculations in Tables: Example II

Example: Number of bookings for each airline carrier:

- Prerequisite: You require a global counter variable, for example, type I. Here: CNT.
- Your table must be sorted by CARRID on the *Data* tab page.
- It does not matter whether you choose *before* or *after loop* for the event because you do not output the counter until the end of each control level and not for each line.
- The counter variable has to be reset at each level change, that is, every time there is a change in the CARRID field.

Exercise 4: Tables

Exercise Objectives

After completing this exercise, you will be able to:

- Create tables
- Create line types
- Output text in tables
- Define control levels for tables

Business Example

The Fly & Smile travel agency constitutes of a number of independent units located in different parts of the world. Each unit is headed by a business unit head and operates as an independent entity under the common parent company. You are responsible for creating invoices, which is a step-wise procedure, of the booked flights for the respective customers. You need to enhance your existing invoice form and create a "real" table to output actual customer bookings.

Task 1:

Note: Copy template for the form: BC470_DATAS

Package/Development class (for all exercises): ZBC470_##

Name of the form to be created: ZBC470_##_TABLS

Model solution: BC470_TABLS

Application program for testing purposes: SAPBC470_DEMO

Copy template

1. Copy the form that you used in the last task (ZBC470_##_DATAS) to ZBC470_##_TABLS. Alternatively, copy the copy template BC470_DATAS.

Continued on next page

Task 2:

Define table

Note: You do not need the Form Painter for tables – you can hide it to allow more space for the maintenance screen.

Output the table of bookings for each customer in the main window instead of the existing text node DUMMY_TABLE. The application program passes the data as an internal table (IT_BOOKINGS) to the form.

1. Create table and define data for table.

   In the MAIN window, create a table node called BOOKINGS after the node INTRODUCTION. The system fills this form table by reading each line of the internal table IT_BOOKINGS into the work area WA_BOOKINGS. You defined WA_BOOKINGS as a global variable in the previous exercise.

   Note that IT_BOOKINGS contains the data for **all** customers selected on the selection screen. In the WHERE condition, define that the field CUSTOMID of the internal table is identical to WA_CUSTOMERS-ID. Remember, WA_CUSTOMERS contains all important information on the customer for whom the invoice is to be created. It is an interface parameter, which is filled in the application program.

   Perform a local check.

2. Determine table layout

   The table should have the following structure:

Flight	***Flight Date***	***Price***
Bookings for AA		
AA0017	10/16/2002	1,200.00 EUR
AA0017	10/17/2002	1,200.00 EUR
AA0026	11/18/2002	700.00 USD
Number of bookings for AA: 3		

3. Create the four line types required using the Table Painter.

Continued on next page

- Line type HEADING with three cells. This is required for the column heading. Use the table above to determine the cell widths. It is useful to display the ruler.

- Line type SUBHEADING with one shaded cell (for the subheadings).

- Line type ITEM with three cells (for the items).

Line type SUBTOTAL with a cell (for the end of a control level).

4. View the result in the detail view. If you like even numbers, you can adjust the cell widths as follows, for example:

   - Set the table width to 15 cm.

   - The cells widths are to be 2.5 cm, 4.0 cm, and 8.5 cm for the line types HEADING and ITEM.

5. Prevent page breaks from occurring within each line type.

6. Perform a local check.

Task 3:

The following is to be output for each booking:

- Carrier ID and connection number

- Flight date

- Price and currency

1. Create a table line for the main area of the table and choose the type ITEM. In the process, three cells should be automatically inserted in the navigation tree. Assign meaningful names for the table line and cells. Create a text element FLIGHT in the first cell, a text element DATE in the second cell, and PRICE in the third cell.

2. Include the carrier ID and the connection number in FLIGHT. Use the field list and drag the fields CARRID and CONNID of the global field WA_BOOKINGS into the text node. Select the paragraph format TB (“Cell in table body”).

3. Output the flight date (WA_BOOKINGS-FLDATE) in DATE. Choose the paragraph format TB.

4. Output a tabulator as well as the price and currency WA_BOOKINGS-FORCURAM and WA_BOOKINGS-FORCURKEY in PRICE. Choose the paragraph format TB. The decimal tabulator is already contained in paragraph format TB.

   Define the following formatting options for WA_BOOKINGS-FORCURAM:

Continued on next page

- Output length: 13

- Decimal places: 2

Result

When completely expanded, the table BOOKINGS should appear as follows at the end of task 3:

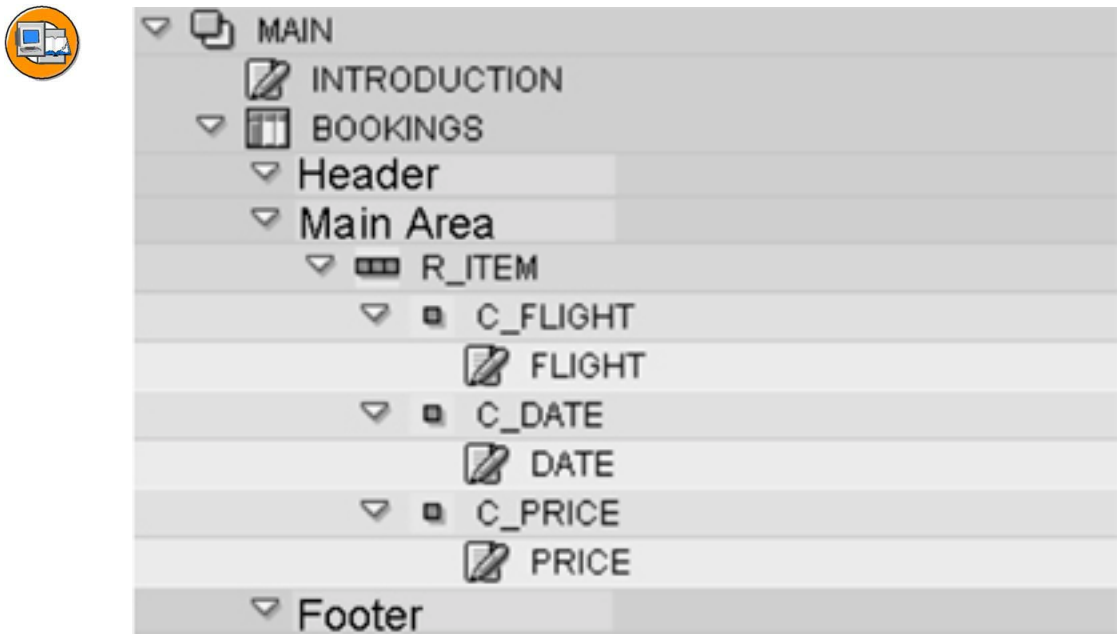


Figure 72: When completely expanded, the table BOOKINGS should appear as follows at the end of task 3:

Task 4:

Define column headings

The table is to have the column headings “Flight”, “Flight date” and “Price”.

1. Create a table line for the table header and choose the type HEADING. In the process, three cells should be automatically inserted in the navigation tree. Assign meaningful names for the table line and cells. Create a text element H_FLIGHT in the first cell, a text element H_DATE in the second cell, and H_PRICE in the third cell.

Continued on next page

2. Enter “Flight” in the text node H_FLIGHT, “Flight date” in the text node H_DATE, and “Price” in the text node H_PRICE. Choose the paragraph format TH (“Cell in table header”) for all three text nodes.

Hint: Collapse the column headings in the navigation tree again to obtain an overview.

Result

When completely expanded, the table BOOKINGS should appear as follows at the end of task 4:

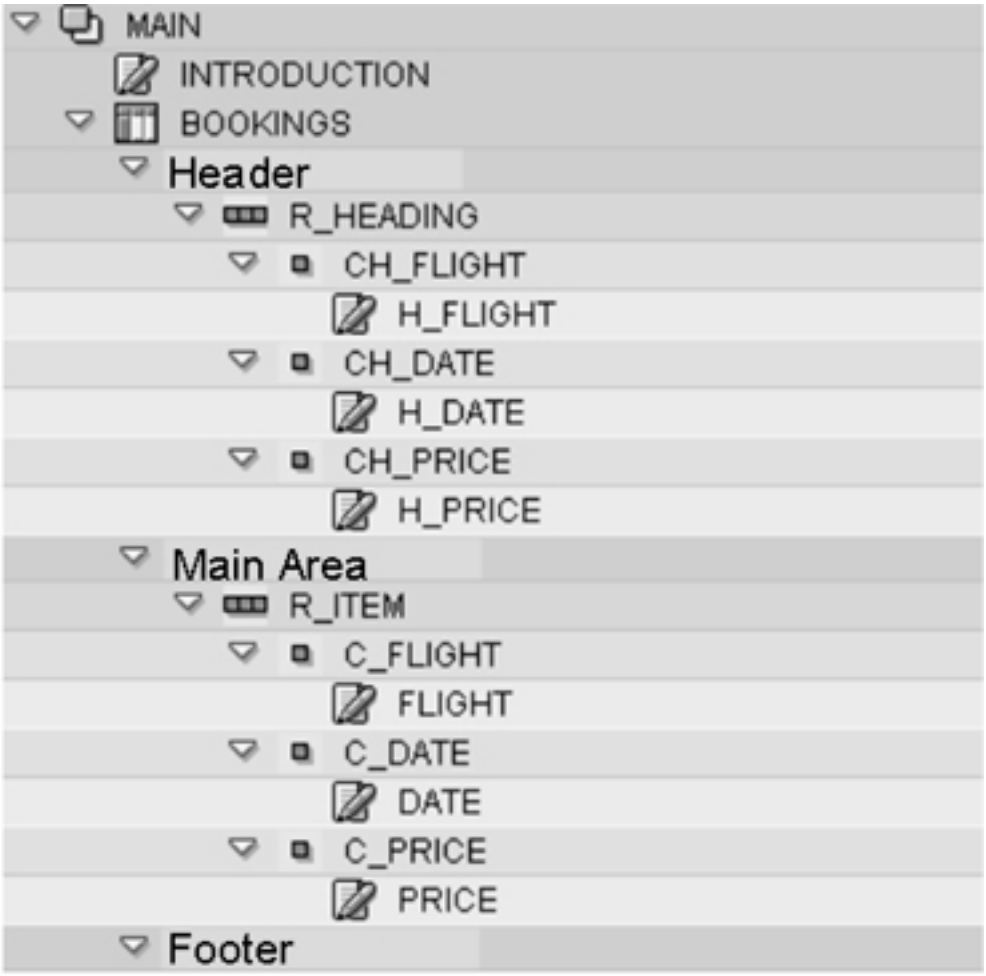


Figure 73: When completely expanded, the table BOOKINGS should appear as follows at the end of task 4:

Continued on next page

Task 5:

Define control levels

The table is to have control levels for the airline carriers. Select the beginning and the end of the bookings for each carrier.

1. Make the necessary entries on the *Data* tab to ensure that the table is sorted by the carrier (CARRID) and event nodes are created for the beginning and the end of the control level in the navigation tree.
2. Insert subheading

   To output a subheading for each airline carrier, create a table line for "CARRID Event on Sort Begin" and choose the type SUBHEADING. In the process, a cell should be automatically inserted in the navigation tree. Assign meaningful names for the table line and cell.
3. Create a text node called H_CARRIER_HEADING in the new cell. Enter the text "Bookings for <carrier ID>". Use the field list to insert the appropriate ID (WA_BOOKINGS-CARRID). Choose the Italic character format and the TO paragraph format for the subheading.

Result

When completely expanded, the table BOOKINGS should appear as follows at the end of task 5:

Continued on next page

Figure 74: When completely expanded, the table BOOKINGS should appear as follows at the end of task 5:

Continued on next page

Task 6:

Optional: Insert intermediate counter

The relevant number of booked flights is to be output after each airline carrier.

1. Create a global counter variable CNT as a whole number. On the *Calculations* tab page of the table, make appropriate entries so that CNT increases by 1 with each booking item but the airline carrier is reset after each change.
2. Create a table line for "CARRID Event on Sort End" and choose the type SUBTOTAL. In the process, a cell should be automatically inserted in the navigation tree. Assign meaningful names for the table line and cell.
3. Insert a text node with the name CARRIER_SUBTOTAL as a subnode of the new cell and output (in paragraph format TO) the intermediate counter.

Task 7:

1. Delete the old text node DUMMY_TABLE since it is no longer required.

Task 8:

1. If you reduced the height of the main window on the page FIRST in one of the previous exercises and now use this form template, increase the height again.

Task 9:

1. Test your form

Solution 4: Tables

Task 1:

Note: Copy template for the form: BC470_DATAS

Package/Development class (for all exercises): ZBC470_##

Name of the form to be created: ZBC470_##_TABLS

Model solution: BC470_TABLS

Application program for testing purposes: SAPBC470_DEMO

Copy template

1. Copy the form that you used in the last task (ZBC470_##_DATAS) to ZBC470_##_TABLS. Alternatively, copy the copy template BC470_DATAS.

   a) See the exercise in Unit 3.

Task 2:

Define table

Note: You do not need the Form Painter for tables – you can hide it to allow more space for the maintenance screen.

Output the table of bookings for each customer in the main window instead of the existing text node DUMMY_TABLE. The application program passes the data as an internal table (IT_BOOKINGS) to the form.

1. Create table and define data for table.

   In the MAIN window, create a table node called BOOKINGS after the node INTRODUCTION. The system fills this form table by reading each line of the internal table IT_BOOKINGS into the work area WA_BOOKINGS. You defined WA_BOOKINGS as a global variable in the previous exercise.

   Note that IT_BOOKINGS contains the data for **all** customers selected on the selection screen. In the WHERE condition, define that the field CUSTOMID of the internal table is identical to WA_CUSTOMERS-ID. Remember, WA_CUSTOMERS contains all important information on the customer for whom the invoice is to be created. It is an interface parameter, which is filled in the application program.

Continued on next page

Perform a local check.

a) Choose *Create* → *Table* from the context menu of the text node INTRODUCTION. Change the name to BOOKINGS and enter a description. Select the *Internal table* checkbox on the *Data* tab and enter the following data into the fields on the right side: IT_BOOKINGS INTO WA_BOOKINGS. Enter the following for the WHERE condition: CUSTOMID = WA_CUSTOMERS-ID. Choose the *Check* pushbutton on the maintenance screen.

2. Determine table layout

The table should have the following structure:

Flight	***Flight Date***	***Price***
Bookings for AA		
AA0017	10/16/2002	1,200.00 EUR
AA0017	10/17/2002	1,200.00 EUR
AA0026	11/18/2002	700.00 USD
Number of bookings for AA: 3		

a) You define the layout on the *Table* tab.

3. Create the four line types required using the Table Painter.

- Line type HEADING with three cells. This is required for the column heading. Use the table above to determine the cell widths. It is useful to display the ruler.

- Line type SUBHEADING with one shaded cell (for the subheadings).

- Line type ITEM with three cells (for the items).

Continued on next page

Line type SUBTOTAL with a cell (for the end of a control level).

a) Make sure that the *Lines and Columns* option is activated pencil pushbutton in the bottom toolbar of the Table Painter. A line type is automatically created. Create three other line types by drawing horizontal lines whilst holding down the left mouse button in the Table Painter. To create the required names HEADING, SUBHEADING, ITEM and SUBTOTAL, choose *Rename Line* from the context menu of a line type. Draw two vertical lines whilst holding down the left mouse button for the two line types HEADING and ITEM to create the required cells. To add lines to the table, deactivate the *Lines and Columns* option (pencil pushbutton in the bottom toolbar of the Table Painter). Choose an appropriate box color and line width, and then choose black with an intensity of 0 percent for the shading. Select all line types (with Ctrl-A) and choose the *Outer Frame* and *Inner Frame* buttons in the top toolbar of the Table Painter.

   To shade the line type SUBHEADING, position the cursor on the line type SUBHEADING and choose a shading. Choose the *Use Shading* pushbutton (in the top toolbar of the Table Painter).

4. View the result in the detail view. If you like even numbers, you can adjust the cell widths as follows, for example:

   - Set the table width to 15 cm.

   - The cells widths are to be 2.5 cm, 4.0 cm, and 8.5 cm for the line types HEADING and ITEM.

   a) Choose *Details* (top right in the Table Painter) to go to the detail view and change the entries that have been copied by the Table Painter, if necessary.

5. Prevent page breaks from occurring within each line type.

   a) In the detail view, select the *No page break* checkbox for all four line types.

6. Perform a local check.

   a) Choose the *Check* pushbutton on the maintenance screen.

Continued on next page

Task 3:

The following is to be output for each booking:

- Carrier ID and connection number

- Flight date

- Price and currency

1. Create a table line for the main area of the table and choose the type ITEM. In the process, three cells should be automatically inserted in the navigation tree. Assign meaningful names for the table line and cells. Create a text element FLIGHT in the first cell, a text element DATE in the second cell, and PRICE in the third cell.

   a) Use the context menu of the main area of the table to create a table line and call it R_ITEM. On the *Output Options* tab page, choose ITEM as the line type. Name the resulting cells C_FLIGHT, C_DATE, and C_PRICE, in this case. Using the context menu, create a text node for each of these cells and call them FLIGHT, DATE, or PRICE.

2. Include the carrier ID and the connection number in FLIGHT. Use the field list and drag the fields CARRID and CONNID of the global field WA_BOOKINGS into the text node. Select the paragraph format TB (“Cell in table body”).

   a) Select the text node FLIGHT and go to the *General attributes* tab. Drag the fields WA_BOOKINGS-CARRID and WA_BOOKINGS-CONNID with the mouse from the field list into the editor of the text node. Choose the paragraph format TO from the list. To assign paragraph formats, you simply have to place the cursor in the paragraph that is to be formatted. You do not need to select anything in particular.

3. Output the flight date (WA_BOOKINGS-FLDATE) in DATE. Choose the paragraph format TB.

   a) Repeat these steps for the text node DATE and WA_BOOKINGS-FLDATE. Choose the paragraph format TO from the list.

4. Output a tabulator as well as the price and currency WA_BOOKINGS-FORCURAM and WA_BOOKINGS-FORCURKEY in PRICE. Choose the paragraph format TB. The decimal tabulator is already contained in paragraph format TB.

   Define the following formatting options for WA_BOOKINGS-FORCURAM:

   - Output length: 13

Continued on next page

- Decimal places: 2

a) Repeat these steps for the text node PRICE and WA_BOOKINGS-FORCURAM and WA_BOOKINGS-FORCURKEY. Choose the paragraph format TO from the list.

Set a tab by pressing the tab button on your keyboard.

Place your cursor on WA_BOOKINGS-FORCURAM. Choose the *Change Field* pushbutton (the pencil pushbutton in the Table Painter). On the subsequent dialog box, change & WA_BOOKINGS-FORCURAM& into &WA_BOOKINGS-FORCURAM(13.2)&.

Result

When completely expanded, the table BOOKINGS should appear as follows at the end of task 3:

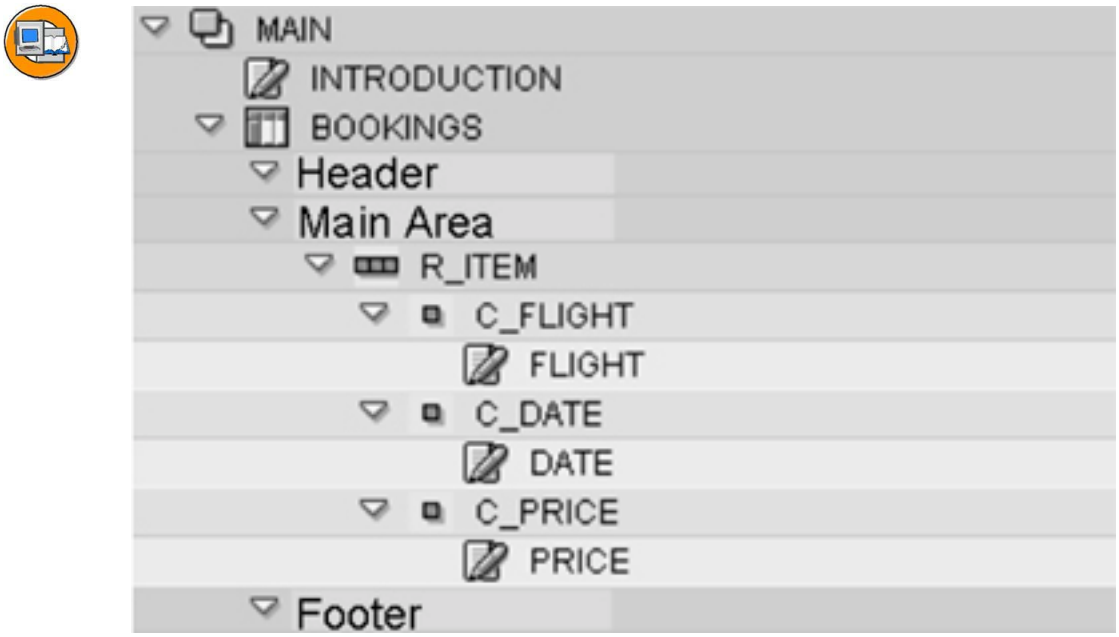


Figure 75: When completely expanded, the table BOOKINGS should appear as follows at the end of task 3:

Continued on next page

Task 4:

Define column headings

The table is to have the column headings “Flight”, “Flight date” and “Price”.

1. Create a table line for the table header and choose the type HEADING. In the process, three cells should be automatically inserted in the navigation tree. Assign meaningful names for the table line and cells. Create a text element H_FLIGHT in the first cell, a text element H_DATE in the second cell, and H_PRICE in the third cell.

   a) Use the context menu of the header (right mouse click) to create a table line, and call it R_HEADING. On the Output Options tab page, choose HEADING as the line type. Name the resulting cells, such as CH_FLIGHT, CH_DATE, and CH_PRICE.

   Using the context menu, create a text node for each of these cells and call them H_FLIGHT, H_DATE, or H_PRICE.

2. Enter “Flight” in the text node H_FLIGHT, “Flight date” in the text node H_DATE, and “Price” in the text node H_PRICE. Choose the paragraph format TH (“Cell in table header”) for all three text nodes.

   Hint: Collapse the column headings in the navigation tree again to obtain an overview.

   a) Enter the texts in the editor for the new text elements (*General attributes* tab page). Choose the paragraph format TH from the list.

Result

When completely expanded, the table BOOKINGS should appear as follows at the end of task 4:

Continued on next page

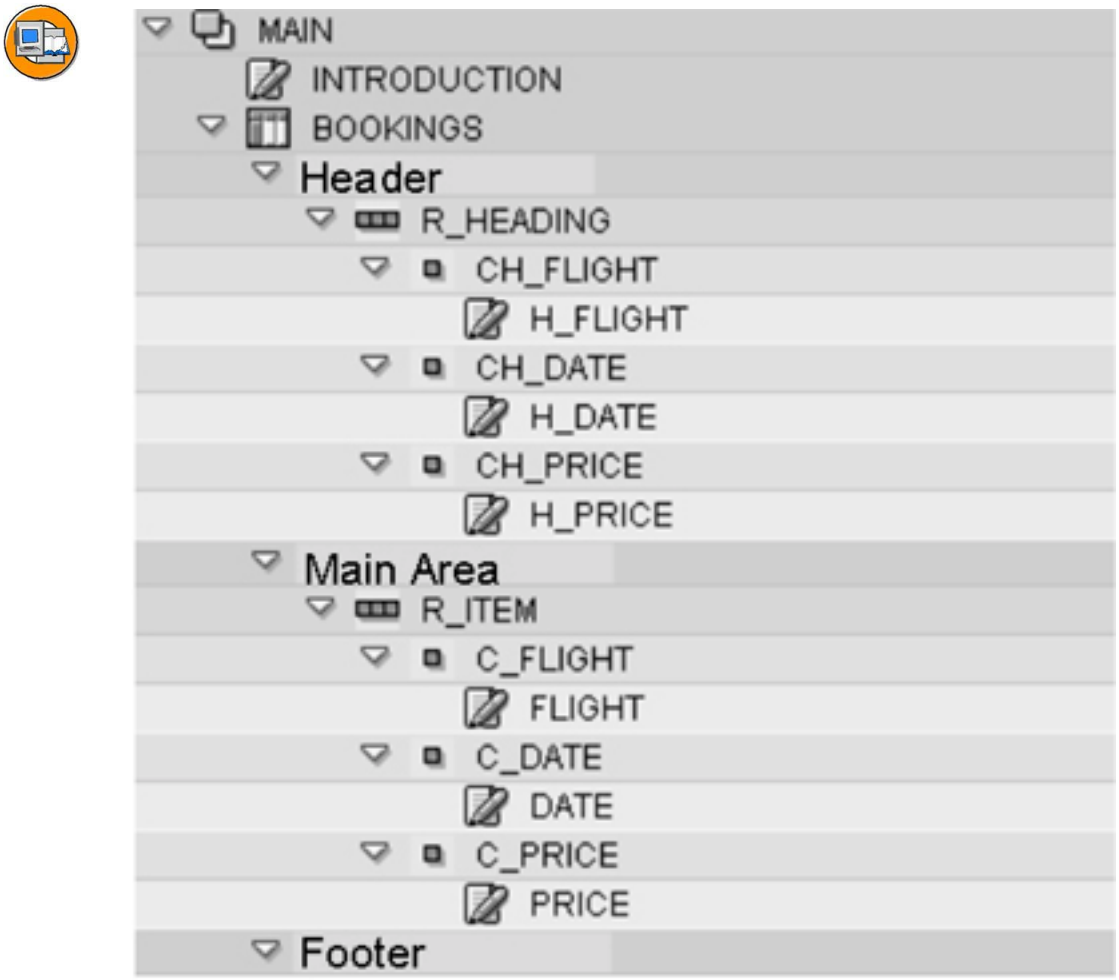


Figure 76: When completely expanded, the table BOOKINGS should appear as follows at the end of task 4:

Task 5:

Define control levels

The table is to have control levels for the airline carriers. Select the beginning and the end of the bookings for each carrier.

1. Make the necessary entries on the *Data* tab to ensure that the table is sorted by the carrier (CARRID) and event nodes are created for the beginning and the end of the control level in the navigation tree.

   a) On the *Data* tab page of the table BOOKINGS, enter CARRID as the field name for the sort criteria and select the *Event on Sort Begin* and *Event on Sort End* checkboxes. This automatically inserts the nodes for the control levels in the navigation tree.

Continued on next page

2. Insert subheading

   To output a subheading for each airline carrier, create a table line for "CARRID Event on Sort Begin" and choose the type SUBHEADING. In the process, a cell should be automatically inserted in the navigation tree. Assign meaningful names for the table line and cell.

   a) Use the context menu (right mouse button) of the event node "CARRID Event on Sort Begin" to create a table line. Call it R_CARRIER_SUB_HEADING, for example, and choose SUBHEADING as the line type on the *Output Options* tab page.

3. Create a text node called H_CARRIER_HEADING in the new cell. Enter the text "Bookings for <carrier ID>". Use the field list to insert the appropriate ID (WA_BOOKINGS-CARRID). Choose the Italic character format and the TO paragraph format for the subheading.

   a) Create a text node called CARRIER_SUBHEADING in the cell that has now been automatically created. Enter your text in the editor of the text node CARRIER_HEADING (*General attributes* tab page).

Result

When completely expanded, the table BOOKINGS should appear as follows at the end of task 5:

Continued on next page

Figure 77: When completely expanded, the table BOOKINGS should appear as follows at the end of task 5:

Continued on next page

Task 6:

Optional: Insert intermediate counter

The relevant number of booked flights is to be output after each airline carrier.

1. Create a global counter variable CNT as a whole number. On the *Calculations* tab page of the table, make appropriate entries so that CNT increases by 1 with each booking item but the airline carrier is reset after each change.

   a) Create the variable CNT in the global definitions by entering the following: CNT TYPE I.

      On the *Calculations* tab page of the table, choose or enter:

      - Operation: Number

      - Target field name: CNT

      - Event: Before loop

      - Reset: Sort criterion

      - For field name: CARRID

2. Create a table line for "CARRID Event on Sort End" and choose the type SUBTOTAL. In the process, a cell should be automatically inserted in the navigation tree. Assign meaningful names for the table line and cell.

   a) You have already created the event node "CARRID Event on Sort End" in exercise 5-1. Create a table line with its context menu (right mouse button), specify a name (for example, R_CARRIER_SUBTOTAL), and define SUBTOTAL as the line type on the tab page. This automatically inserts a new cell in the navigation tree. Name it C_CARRIER_SUBTOTAL.

3. Insert a text node with the name CARRIER_SUBTOTAL as a subnode of the new cell and output (in paragraph format TO) the intermediate counter.

   a) Create a text node called CARRIER_SUBTOTAL as a subnode of the cell C_CARRIER_SUBTOTAL. Using the field list, assign the intermediate counter CNT.

Continued on next page

Task 7:

1. Delete the old text node DUMMY_TABLE since it is no longer required.

   a) Delete the text node DUMMY_TABLE using its context menu (right mouse button).

Task 8:

1. If you reduced the height of the main window on the page FIRST in one of the previous exercises and now use this form template, increase the height again.

   a) If required, increase the size of the window MAIN on the page FIRST using the Form Painter.

Task 9:

1. Test your form

   a) Test your form using the program SAPBC470_DEMO.

Lesson Summary

You should now be able to:

- Identify and create tables and their line types
- Process tables
- Output data in tables and sort a table
- Create control levels in a table
- Perform calculations in tables

Lesson: Templates

Lesson Overview

In this lesson, you will learn about the template node of an SAP Smart Form. You will also be able to draw templates using Table Painter. In addition, you will be able to define template layout and output contents in templates. Finally, you will learn about creating templates by redrawing.

Lesson Objectives

After completing this lesson, you will be able to:

- Identify the template node type of an SAP Smart Form
- Draw table templates using a Table Painter
- Define template layout and output contents in templates
- Create templates

Business Example

The Fly & Smile travel agency constitutes a number of independent units located in different parts of the world. Each unit is headed by a business unit head and operates as an independent entity under the common parent company. You are responsible for creating invoices, which is a step-wise procedure, of the booked flights for the respective customers.

You need to draw a template for printing the data, such as customer information and booking details, on predefined forms. This data is to be used for setting up an invoice form.

Introducing Templates

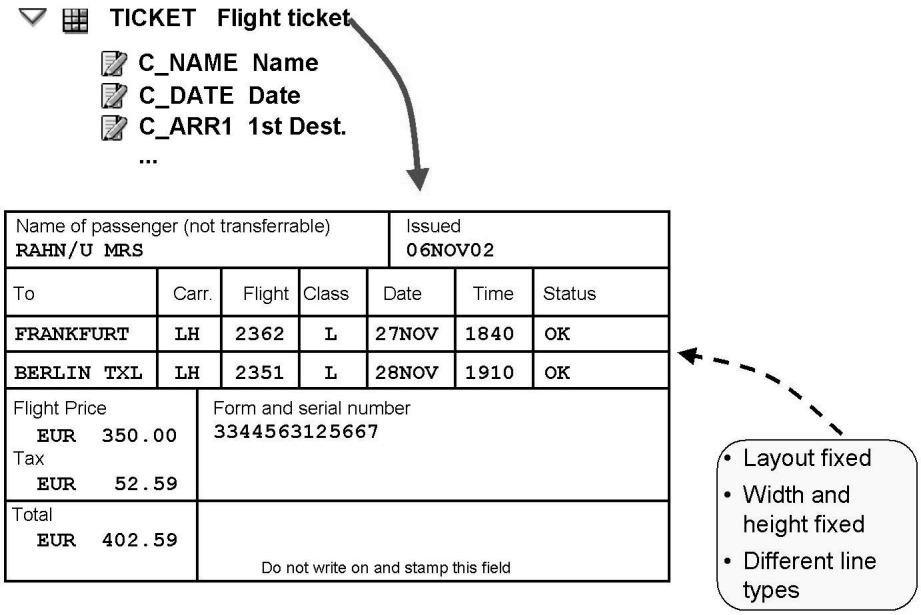


Figure 78: Templates

You use the *Template* node type to output tables with a fixed layout and size. Templates are used, for example, for printing data on predefined forms, such as flight tickets (see above) or tax forms.

Like all other nodes, templates are created as subnodes of windows, that is, using the context menu (right mouse button) in the navigation tree.

Templates cannot be nested.

You can create different node types as subnodes of templates.

Note: Text that does not fit into the cell selected is not output because the layout of the template is fixed.

Graphics that you create as subnodes of templates are not visible in the Form Painter. You only see these in the print preview.

Comparing Tables and Templates

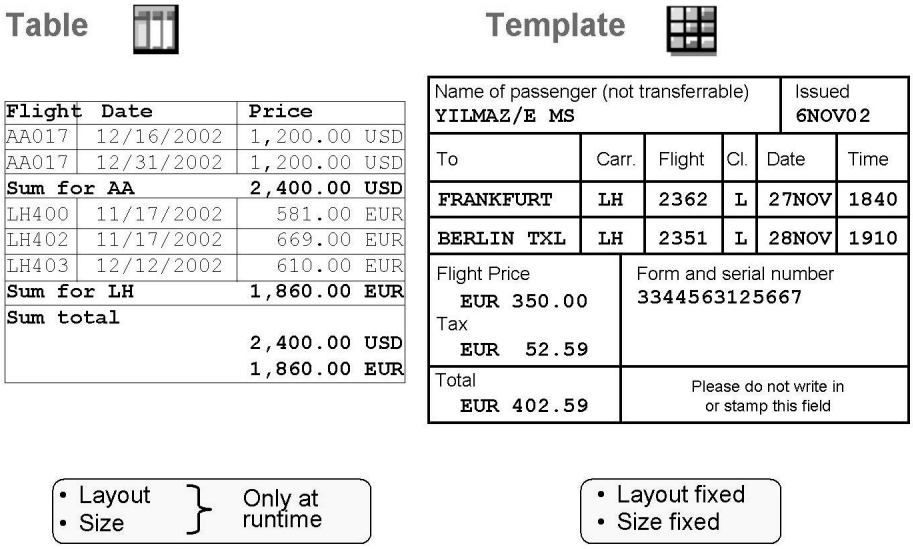


Figure 79: Tables and Templates

This unit will introduce you to two other types of nodes, tables and templates. Tables and templates have several things in common. For example, they are both designed with the Table Painter and they use different line types.

The most important difference between them is how their layout is determined:

- The precise layout and the length of a table can only be determined at runtime, depending on the type and the number of records read by the application program from the database.
- Templates are defined completely in the Form Painter. This means that the type and the number of their cells cannot be modified at application program runtime. You use templates primarily for external forms.

Drawing Templates

Usage: Same as with tables

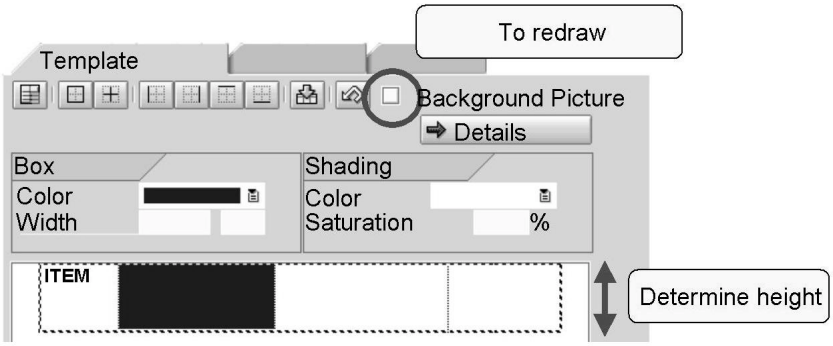


Figure 80: Drawing Templates: Table Painter

The use of the Table Painter is almost the same for templates and tables. You draw your lines and cells with the mouse and add frames and possibly also shading.

One difference is that you can show or hide the background picture for the page, if you assume that there is only one background picture. This is useful if you have scanned in a template and you want to redraw it with the Table Painter. The prerequisite is that you set *Absolute (from top)* for the vertical alignment of the template in the detail view.

You also have to specify the height of the individual lines because it is not dynamic, unlike with tables.

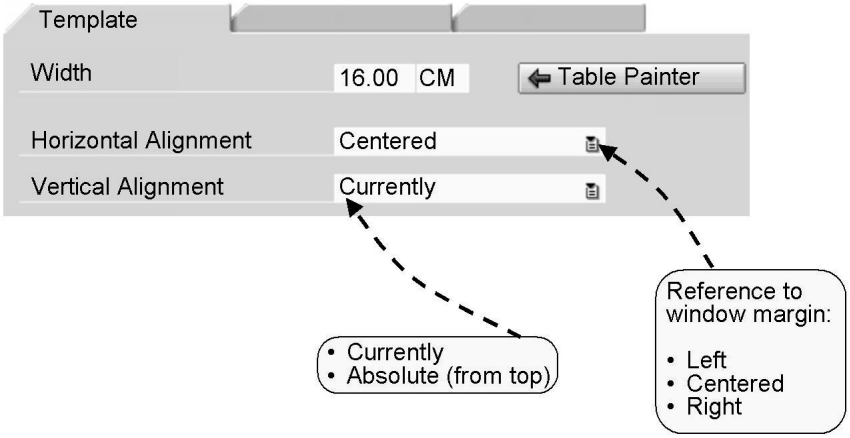


Figure 81: Drawing Templates: Details I

If you choose *Details* on the *Template* tab page, you can make the following settings in the upper area:

Width. The width of the template must not exceed the width of the window into which the template is embedded.

Horizontal alignment. You can choose *Left*, *Centered*, or *Right* as the horizontal alignment of the template. These values refer to the window margin. If you choose *Left* or *Right*, the system displays an additional input field where you can enter the distance to the window margin. If you do not enter a value here, the template is placed directly on the margin.

Vertical alignment. The vertical alignment option allows you to determine the distance of the template from the top window margin. Choose *Absolute (from top)* and enter the desired distance in the input field that appears on the right side. This enables you to place several templates side by side in the same window. This is useful when you want to print labels or put several templates on top of one another to define complex line structures. You can also choose *Currently* as the vertical alignment. This places the template in the window directly underneath the node that precedes the template in the navigation tree. The vertical position of the template in the form is determined by the number of nodes processed before the template at the time of output.

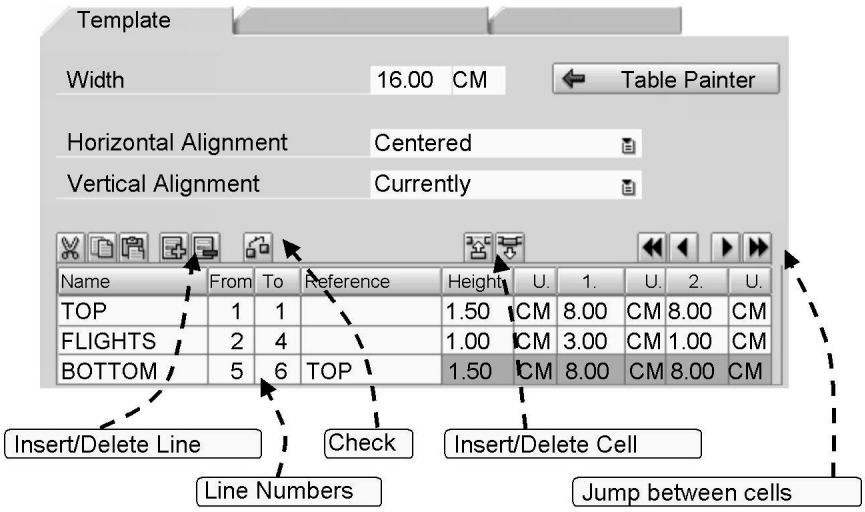


Figure 82: Drawing Templates: Details II

The lines and cells that you can draw with the mouse in the Table Painter can also be created and edited in the detail view. The procedure is similar to that with tables.

Define a unique symbolic name and the range of lines that use this line type. If several lines that are not successive use the same line type, you only need to define the line type once and can then specify it in the

Reference field each time it is used. In the above example, lines 1, 5, and 6 have the same type. The line type BOTTOM refers to the type TOP which has already been declared. This means that the fields for the line height and the width of its cells are not ready for input.

In the *Height* field, you set the height for the entire line.

You can create any number of cells for each line. Enter the width of these cells. The sum of the values for the width of the cells must be the same as the width set for the template.

Template Layout

Name	From	To	Reference	Height	U.	1.	U.	2.	U.	3.	U.
TOP	1	1		1.50	CM	8.00	CM	8.00	CM		
FLIGHTS	2	4		1.00	CM	3.00	CM	1.00	CM	12.00	CM
BOTTOM	5	6	TOP	1.50	CM	8.00	CM	8.00	CM		

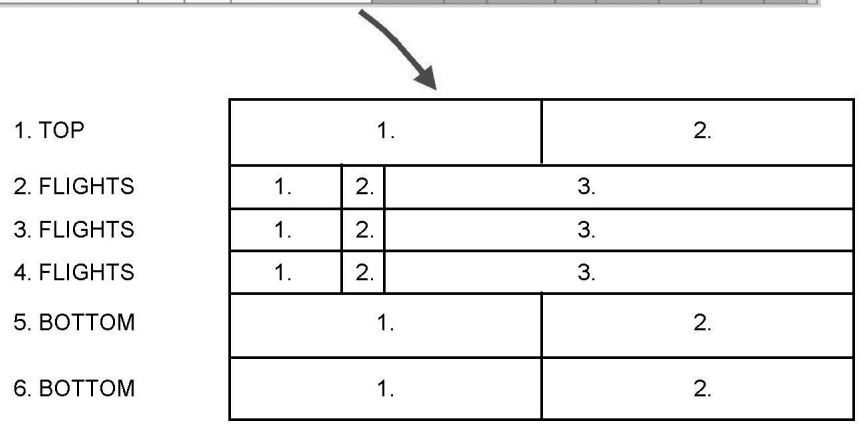


Figure 83: Template Layout: Example

The above graphic shows a possible layout definition and the result of the print preview.

You need the line numbers and cell numbers to output the contents in the cells.

Hint: Since you cannot define cells with different heights in one line, you can put two or more tables on top of each other. To do this, enter absolute values for the vertical and the horizontal position.

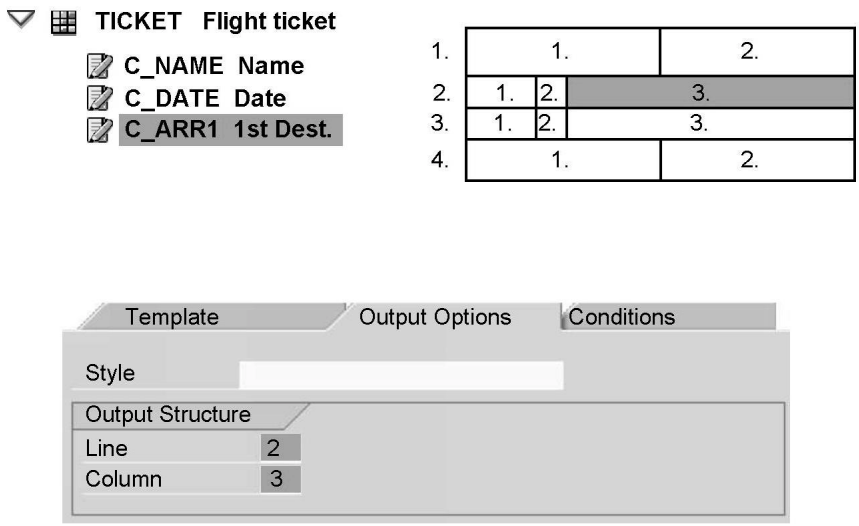


Figure 84: Outputting Contents in Templates

After defining the layout of the template, you can use the context menu, that is, right mouse button on the template to create subnodes in which contents are output. Alternatively, you can choose *Edit → Node → Create* from the menu and select the *Under* radio button to create subnodes for the contents.

In the *Output structure* group box on the *Output options* tab of the new nodes created, you determine the template line and cell in which the node is to be inserted. If you do not make an entry here, the node is output in the current cell, that is, the cell in which data was last output. Note that text that does not fit into a cell is not output.

You can also assign multiple nodes to a cell. The output order in the cell is determined by the order of the nodes in the navigation tree.

Note: No settings are permitted for frames or shading on the *Output Options* tab page. You have to make these settings on the *Template* tab page.

Creating Templates

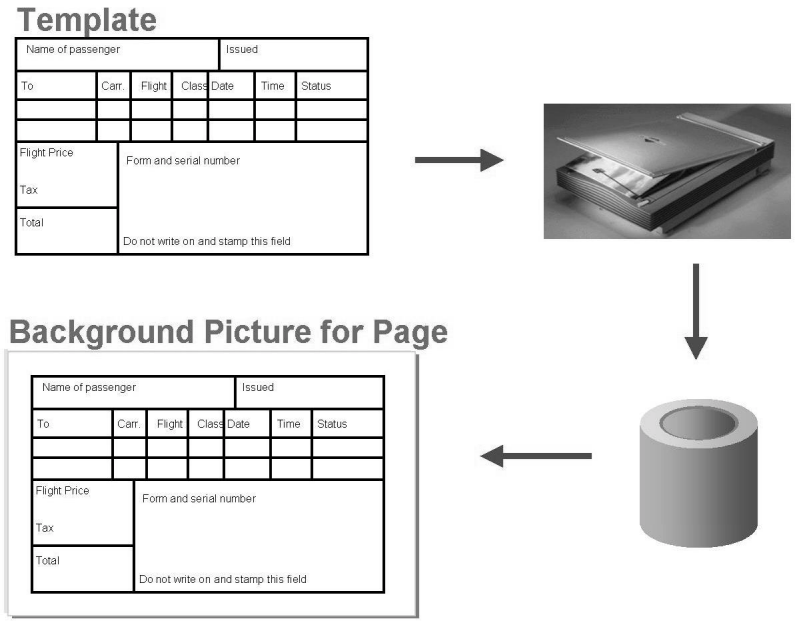


Figure 85: Creating Templates by Redrawing I

You can create templates following the procedure just described. An alternative procedure - which is useful in the case of complex templates - is to scan an external form, import it using graphics administration (transaction SE78), set the graphic as the background picture for a page and redraw the form in the Table Painter.

To do this, follow the steps described in the next graphics. It is assumed that the form has already been imported into the SAP R/3 Enterprise as a graphic.

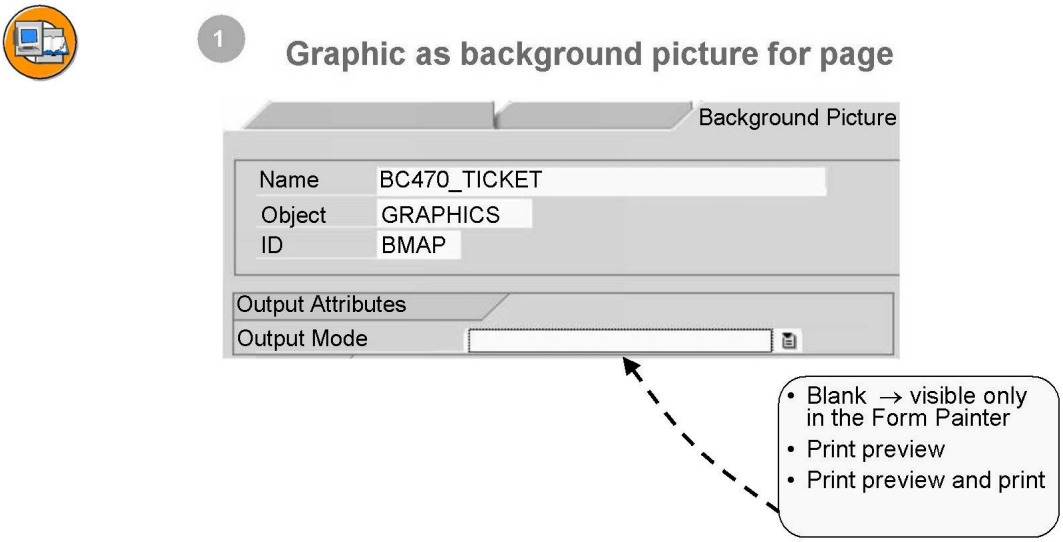


Figure 86: Creating Templates by Redrawing II

On the *Background picture* tab of the page on which you want to position the template, select the scanned form, such as graphic. Determine in the *Output mode* combo box whether you want to print the scanned graphic or not. If you want to print your data in an existing form, leave the field initial. If you want to print the background picture on blank paper, choose *Print preview and print*.

Update the preview of the Form Painter by choosing *Enter* in the maintenance screen.

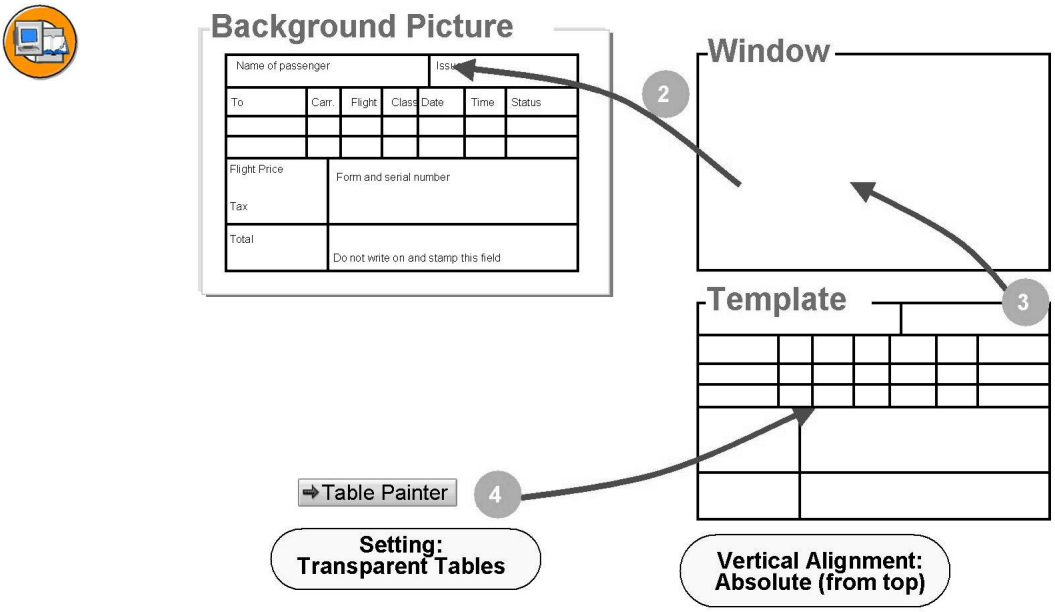


Figure 87: Creating Templates by Redrawing III

Use the Form Painter to place a window exactly over the graphic.

Create a template in this window. The template width should be the same as the window width. On the *Template* tab, choose *Absolute (from top)* as the *Vertical alignment*. Start the Table Painter from this tab page. Ensure that *Display background picture* is selected. In addition, the *Transparent tables* checkbox must be selected on the *General* tab of the Table Painter. You can navigate to this tab from the Form Painter settings. Click the last icon in the toolbar of the Form Painter.

Now, redraw the original form you scanned. To make redrawing easier, you can deselect the *Align tables with grid* checkbox in the Table Painter settings.

If the original form is very complex, you might need to create several templates above or next to each other.

Lesson Summary

You should now be able to:

- Identify the template node type of an SAP Smart Form
- Draw table templates using a Table Painter
- Define template layout and output contents in templates
- Create templates

Unit Summary

You should now be able to:

- Identify and create tables and their line types
- Process tables
- Output data in tables and sort a table
- Create control levels in a table
- Perform calculations in tables
- Identify the template node type of an SAP Smart Form
- Draw table templates using a Table Painter
- Define template layout and output contents in templates
- Create templates

Test Your Knowledge

1. Tables in SAP Smart Forms are subnodes of windows and are created like all other subnodes using the context menu of the _____.

   Fill in the blanks to complete the sentence.

2. What all information is required for line types?

   _____
   _____
   _____
   _____

3. What are the two assignment types possible while processing tables?

   _____
   _____
   _____
   _____

4. How are the header, the main area, and the footer of a table processed if they all appear on one page?

   _____
   _____
   _____
   _____

5. To sort the internal table within the form, the names of the fields to be used are entered as the _____.

   Fill in the blanks to complete the sentence.

6. If you select _____ and/or _____ for a sort criterion, the corresponding control levels are inserted into the navigation tree of the table.

   Fill in the blanks to complete the sentence.

7. You use the events before loop or after loop to count sublevels.

   Determine whether this statement is true or false.

   □ True
   □ False

8. What is the use of a template node type of an SAP Smart Form?

9. Choosing _____ as the vertical alignment places the template in the window directly underneath the node that precedes the template in the navigation tree.

   Fill in the blanks to complete the sentence.

10. Settings for frames or shading are permitted on the *Output Options* tab page.

    Determine whether this statement is true or false.

    □ True

    □ False

11. To make redrawing the original form easier, you can deselect the _____ checkbox in the Table Painter settings.

    Fill in the blanks to complete the sentence.

Answers

1. Tables in SAP Smart Forms are subnodes of windows and are created like all other subnodes using the context menu of the navigation tree.

   Answer: navigation tree

2. What all information is required for line types?

   Answer: The following information is required for line types: -

   - Name
   - Protection against page breaks
   - Number and width of the cells
   - Default type: Without relevance
   - The table width must be identical to the total width of all cells for each line type.

3. What are the two assignment types possible while processing tables?

   Answer: The two possible assignment types are *into* and *assigning*. In case of *into*, the lines are copied from the table into the work area while in *assigning*, the lines are assigned to a field symbol.

4. How are the header, the main area, and the footer of a table processed if they all appear on one page?

   Answer: If all three areas of a table appear on one page, they are processed from top to bottom, that is, first the header, then the main area, and finally the footer.

5. To sort the internal table within the form, the names of the fields to be used are entered as the Sort criteria.

   Answer: Sort criteria

6. If you select Event on Sort Begin and/or Event on Sort End for a sort criterion, the corresponding control levels are inserted into the navigation tree of the table.

   Answer: Event on Sort Begin, Event on Sort End

7. You use the events before loop or after loop to count sublevels.

   Answer: False

   You use the events before loop or after loop if a calculation is to be performed for every processed line of the table.

8. What is the use of a template node type of an SAP Smart Form?

   Answer: The Template node type of an SAP Smart Form is used to output tables with a fixed layout and size. Templates are used for printing data on predefined forms, such as flight tickets or tax forms.

9. Choosing Currently as the vertical alignment places the template in the window directly underneath the node that precedes the template in the navigation tree.

   Answer: Currently

10. Settings for frames or shading are permitted on the *Output Options* tab page.

    Answer: False

    No settings are permitted for frames or shading on the *Output Options* tab page. You have to make these settings on the *Template* tab page.

11. To make redrawing the original form easier, you can deselect the *Align tables with grid* checkbox in the Table Painter settings.

    Answer: *Align tables with grid*

Unit 6

Flow Control

Unit Overview

In this unit, you will get an overview of the various types of nodes, such as node conditions and alternative nodes used for flow control. In addition, you will be able to use the various flow control nodes.

Unit Objectives

After completing this unit, you will be able to:

- Explain the use of various flow control nodes
- Create node conditions
- Use alternative nodes
- Create program line nodes
- Create folders
- Create command nodes
- Create loops

Unit Contents

Lesson: Flow Control Nodes

Lesson Overview

In this lesson, you will obtain an overview of the various flow control nodes provided by SAP Smart Forms. In addition, you will learn about the use and creation of the different flow control nodes.

Lesson Objectives

After completing this lesson, you will be able to:

- Explain the use of various flow control nodes
- Create node conditions
- Use alternative nodes
- Create program line nodes
- Create folders
- Create command nodes
- Create loops

Business Example

The Fly & Smile travel agency constitutes a number of independent units located in different parts of the world. Each unit is headed by a business unit head and operates as an independent entity under the common parent company. You are responsible for creating invoices, which is a step-wise procedure, of the booked flights to the respective customers. The tables are already populated with customer data.

Now, you need to explore the various flow control options provided by SAP Smart Forms for adding more detail to your invoice.

Flow Control Nodes: Overview

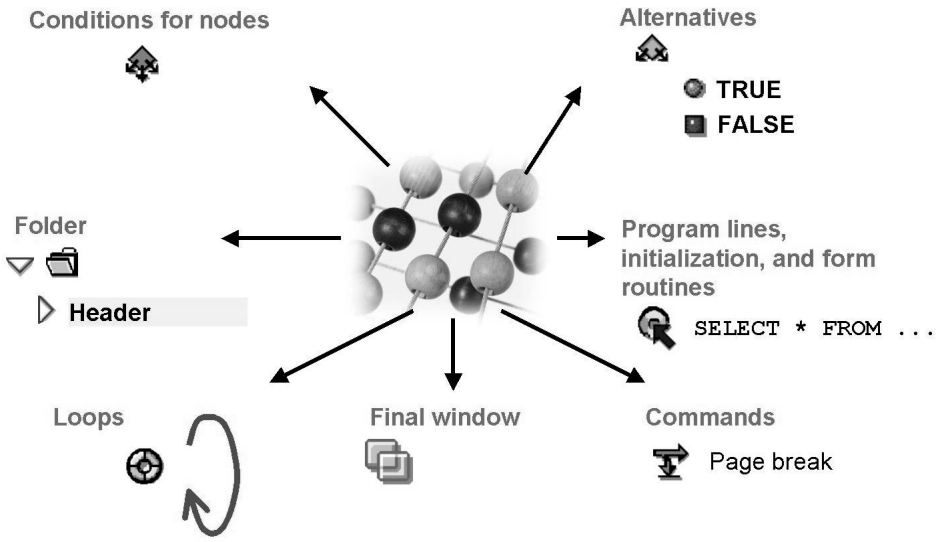


Figure 88: Flow Control: Overview

The form elements presented to you up to this point are processed in a predefined order. Starting with the first page, the nodes of the tree structure are processed from top to bottom. It is helpful to imagine that all nodes are expanded.

In some cases, however, the system can only determine at runtime which parts of a form should be processed. An example of this is tables: The table length and the order of the line types are determined by the data records read.

This unit deals with other flow control options provided by SAP Smart Forms:

- **Output conditions** for nodes.
- **Alternatives:** An alternative is a condition that controls two nodes. One node is processed if the condition is fulfilled, the other is processed if the condition is not fulfilled.
- **Program lines, initialization nodes, and global form routines** allow you to integrate ABAP statements into your form without having to adjust the application program.
- **Final windows** are only processed in a second repetition. **Command nodes** are used for dynamic page breaks, for example.
- Subnodes of **loops** are run through several times.
- **Folders** are used for grouping nodes.

Using Flow Control Nodes

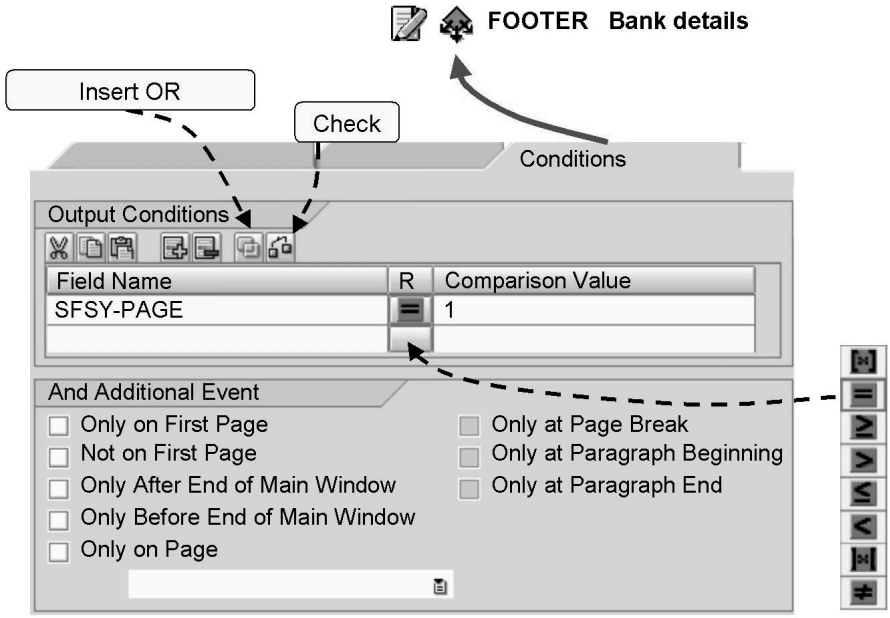


Figure 89: Conditions for Nodes

Most nodes have the *Conditions* tab. You can define two types of conditions for processing the respective node and all of its subnodes:

Field comparisons:

Enter a field name without ampersands for each line. Then, select a relational operator, the default operator is *Equal to*, by clicking the pushbutton between the two columns, and enter a comparison value. This value can be a field or a fixed value.

The fields must be defined in the form interface or in the global definitions, or they must be system fields of SAP Smart Forms (SFSY-...). In final windows, you can only use the system fields FORMPAGES and JOBPAGES in conditions because otherwise their value is not set.

If you insert several conditions, these are linked by a logical AND. Using the pushbutton highlighted above, you can also define an OR relationship.

Specific events: The options available depend on which node is selected. *Only at paragraph beginning* is, for example, only available for headers or footers of tables, and for complex sections. Multiple events are always linked with OR.

Check your entries using the *Check* pushbutton on the tab.

In the navigation tree, the icon shown above is added to the respective node if you have defined a condition for it.

If you use identical windows, graphic windows, or address windows on different pages, then each has its own *Conditions* tab.

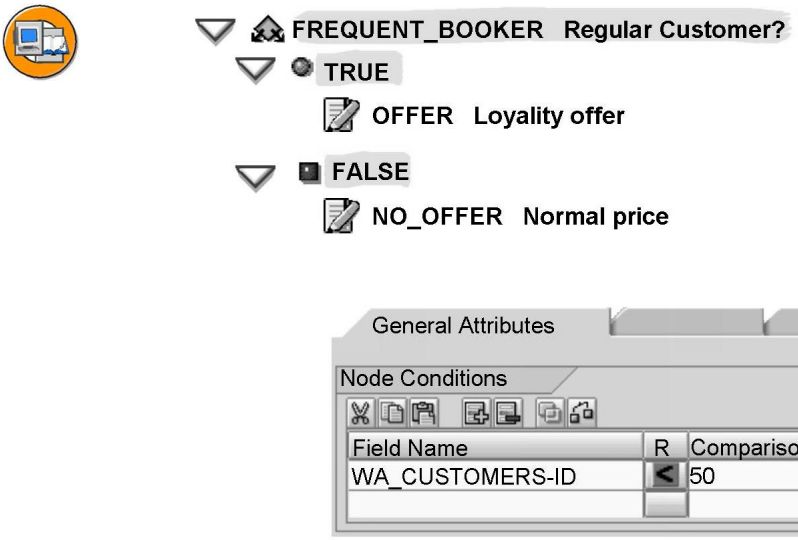


Figure 90: Alternatives

An alternative is a node with two subnodes, which each contain further subnodes. The condition(s) you enter on the *General attributes* tab of the alternative determine which of the two subnodes is processed. If the condition is fulfilled, the TRUE node is processed including all of its subnodes. If the condition is not fulfilled, the FALSE node is processed together with its subnodes. This query is similar to that of the ABAP commands IF and ELSE.

You create alternatives in the same way as any other node either by using the context menu of the navigation tree or by choosing *Edit → Node → Create*.

You can enter the same types of conditions as on the *Conditions* tab of other nodes, such as field comparisons or specific events.

Alternatives can be nested. This allows you to define complex queries.

Please note the following important difference:

On the *General Attributes* tab, you set the conditions that determine whether the node TRUE or the node FALSE is processed.

On the *Conditions* tab, however, you set the conditions that must be fulfilled for the alternative to be processed at all.

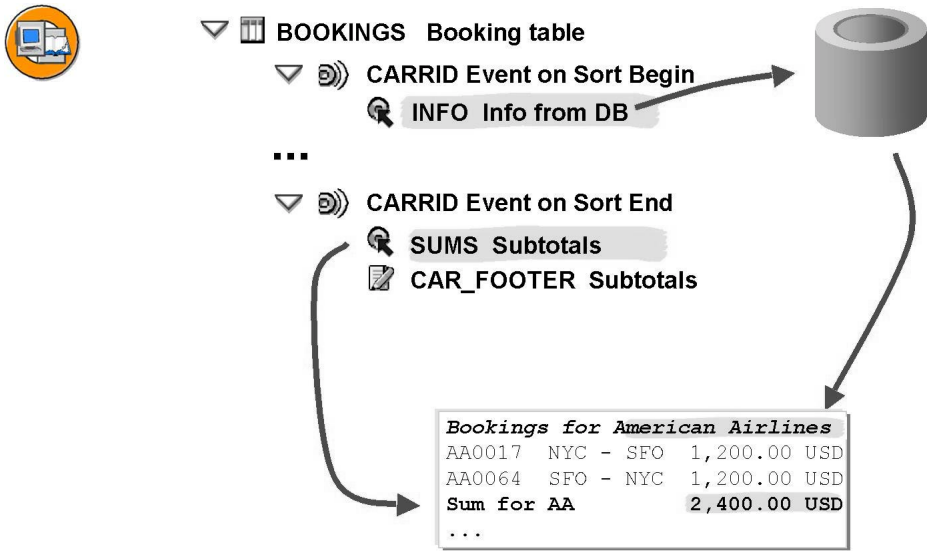


Figure 91: Program Lines I

Nodes of the type *Program lines* allow you to integrate ABAP code into your form. There are many situations where this is helpful, for example:

You need data that is not provided by the application program actually responsible for data retrieval. You could indeed modify the program, but this can be a very complex task, and you would not be able to benefit from enhancements made to this program during an upgrade.

You want to reset counter variables.

You want to calculate complex subtotals and totals, for example, within a table.

You create a program line node using the context menu or by choosing *Edit → Node → Create.* Note the processing sequence. Nodes are processed from top to bottom in the navigation tree. The results of the program lines are therefore only available for those nodes that are processed afterwards.

Since program line nodes cannot generate output and cannot have subnodes, they do not have an *Output options* tab.

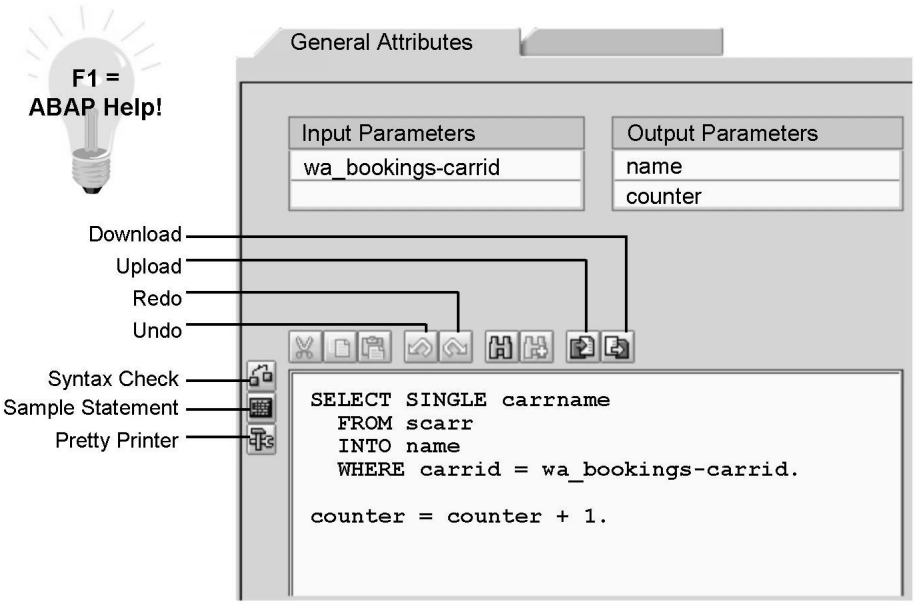


Figure 92: Program Lines II

Program lines in SAP Smart Forms are similar to subroutines in ABAP programs. This means that you must determine the interface but you can also work with local variables that you create with the DATA statement. The input and output parameters must be globally recognized in the form, that is, they must be defined in the interface or in the global definitions.

The distinction between input parameters and output parameters is only made for structuring purposes. It has no effect on the potential for modifying parameters, as both input and output parameters are transferred to the program line node for reference. As a consequence, changes to values in input parameters are permanent and do not only apply within the program lines.

SAP Smart Forms system fields (sfsy-...) or system fields of the ABAP system structure SYST (sy-...) do not need to be declared, but can be used directly in the coding. System fields should be read-only.

In the program lines, you can use form routines that you have created in the global definitions of the form.

If you prefer to work with the old line-oriented ABAP editor, you can set it by choosing *Utilities* → *Settings* → tab page *Editor* → radio button Table Control Editor.

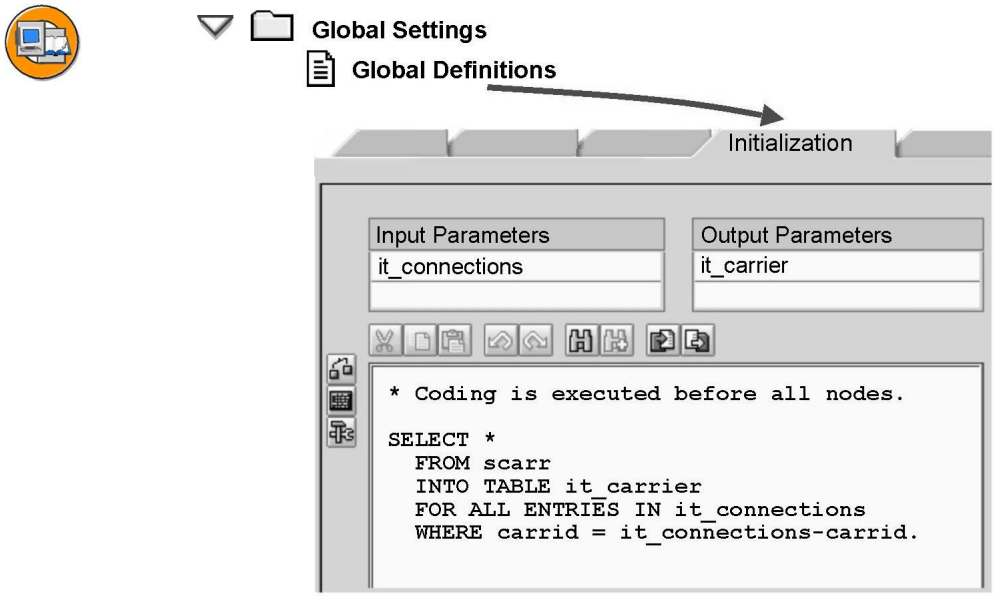


Figure 93: Initialization

The *Initialization* tab in global definitions allows you to enter ABAP code of your choice that you want to execute before the start page is processed. In particular, you can assign values to global data before the actual form formatting process starts.

The handling is identical to that for normal program lines.

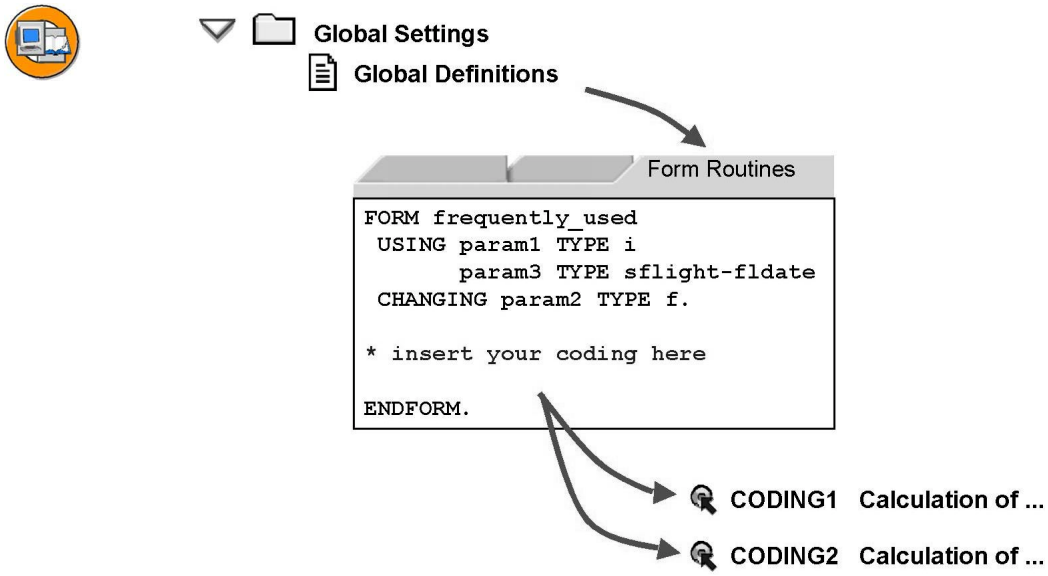


Figure 94: Global Form Routines

If you have coding that is used in different program line nodes, it makes sense to move it into global subroutines (forms) and then call them as required. You create these subroutines in the editor of the *Form Routines* tab of *Global Definitions*.

The syntax is normal ABAP syntax. You define the routines using FORM `<form> [TABLES ...][USING ...] [CHANGING ...]`. You also use normal ABAP syntax for calling a defined form using `perform <form>`. For more information, see the ABAP documentation.

Subroutines that you create in program lines are only known there and cannot be used in other program lines.

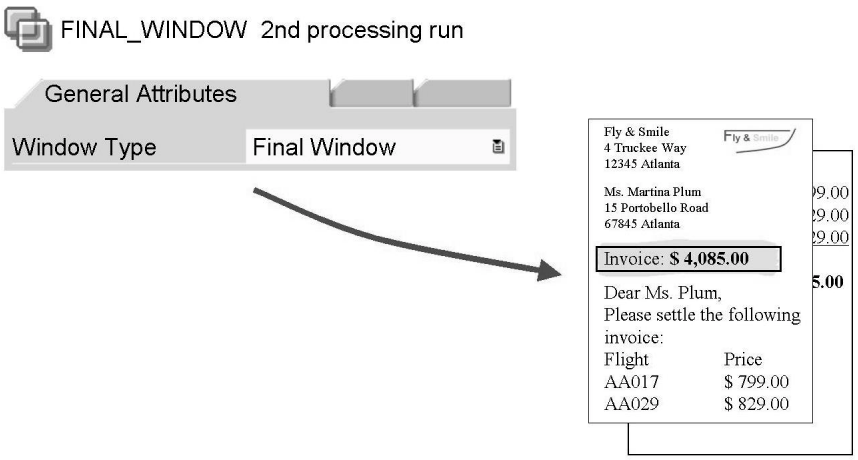


Figure 95: Final Windows

You may sometimes need to use a condition on the first page to query the total number of pages of the form being processed, for example, for bar codes in machines for filling envelopes. The variable SFSY-FORMPAGES does indeed already exist, however, its value is not determined until the end of form processing, and it is subsequently inserted by the form processor in a second processing run.

You may also want to display a total on a page although this total is not calculated until the form is processed. The problem is that fields for which a value has not yet been determined cannot be evaluated in a normal window.

As of SAP Web AS 6.10, there is a new window type, *Final Window*. Windows of this type are only processed in a second processing run, by which time the values for all form variables are already known (including SFSY-FORMPAGES).

Apart from the time of processing, final windows are identical to secondary windows. This means that text that doesn't fit into the window is cut off, and each page in a final window can have an individual height, width, and positioning.

Note: Using SFSY-FORMPAGES (and also SFSY-JOBPAGES) has a negative effect on performance, because all output pages are saved in the main memory until the end of the form, or until the end of the whole print output.

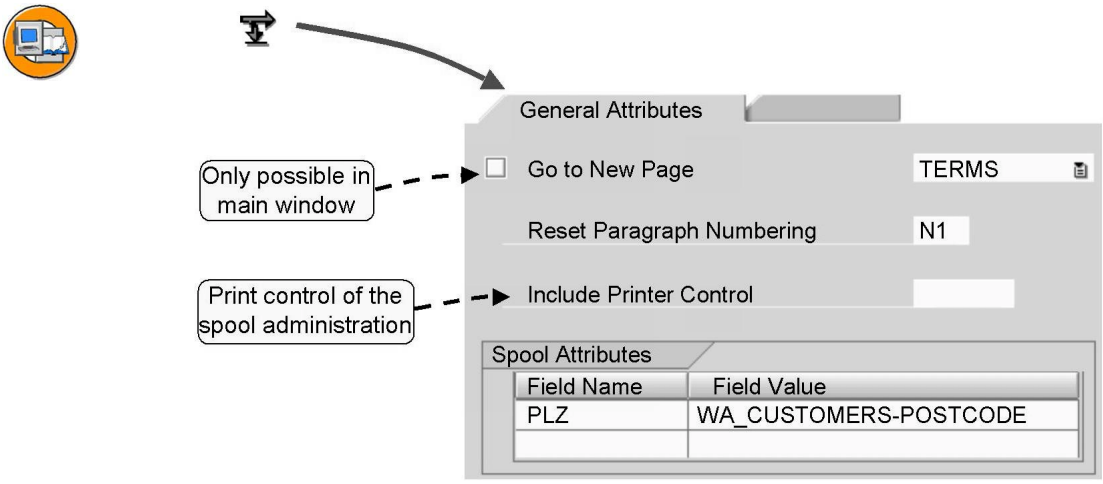


Figure 96: Command Nodes

A command node enables you to do the following:

Go to New Page: A page break normally occurs if the main window of a page is full. The next page processed is the page that you entered on the *General Attributes* tab of the page. In some cases, however, you may want to process a different next page, possibly based on conditions. This is the case, for example, if a page is output several times, that is, is its own next page, but another page is to be processed afterwards. You then use this option and specify which page the system should process next. Note: This option is only allowed within main windows. Otherwise, the function module issues an error message. Furthermore: All nodes after a manual page break in the main window are not processed on the current page. All secondary windows of the current page are, however, still processed before the page break.

Reset Paragraph Numbering: If you enter an outline paragraph here, which must exist in the style used, the numbering of this paragraph and all associated paragraphs at lower-level outline depth is reset to initial. Paragraph formats without outline attributes are ignored.

Include Printer Control: Here, you can send a print control to the output device. This allows you to use special features of your printer. Print controls are managed in spool administration and are converted into printer-specific escape sequences during output.

You can also define free attributes for the spool request with a value of your choice. These are evaluated using the table TSP02A or, for example, using the report RSOPRNT. For more information, see the SAP note 359379.

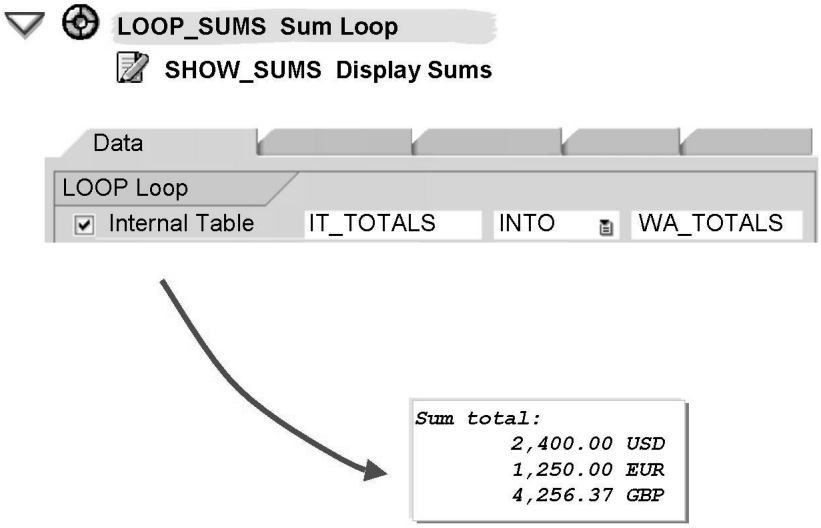


Figure 97: Loops

A loop is very similar to a table because internal tables are also read line-by-line in loops. You do not, however, determine where the data is output, which means there are also no line types. Content nodes, for example, texts are therefore direct subnodes of a loop, both for SAP R/3 4.6C and for the SAP Web Application Server 6.10.

Data tab: Enter the name of the internal table over which the loop is to be executed and the associated work area (assignment type *into*) or the field symbol (assignment type *assigning*). You can specify a line range, determine one or more WHERE conditions, and sort the internal table before it is processed. Sorting the internal table is a prerequisite for sort levels (control levels).

Events tab: If you select *Header* and/or *Footer*, event nodes are automatically created in the navigation tree. You can, for example, enter text as a direct subnode of these event nodes. You can output headers at the beginning of the loop and/or after a page break. Similarly, footers can be output at the end of the loop and/or before a page break. You must specify a height for the footer so that the form processor reserves sufficient space.

Loops can be nested. It is also possible to use tables or templates in loops.

Examples for the use of loops:

Output of an internal table containing amounts sorted by currency

Output of all bookings of all customers in one form

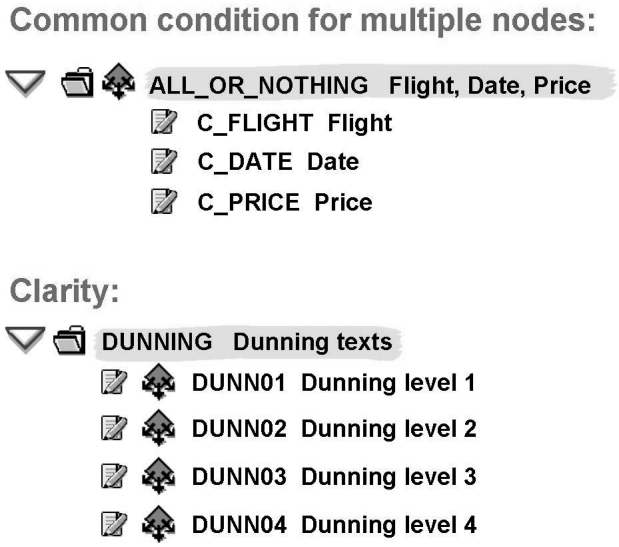


Figure 98: Folder

The larger the form, the more complex is its node hierarchy. For a clearer overview, you can create folders and group nodes in these folders.

Examples of the usage of folders:

Nodes of a template that are assigned to a specific cell or line can be more easily identified or moved in the navigation tree if one folder exists for each cell or line.

A dunning form has different dunning texts of which only one is to be output depending on the reminder days exceeded. Create these texts, with conditions, and group them into folders.

Several nodes share the same condition. Instead of setting this condition individually for each node, you can create a folder and then assign the condition to this folder.

Folders can also be used to output text with a footer and/or header. Like loops, folders have the *Events* tab. If you select the *Header* and/or *Footer* checkbox on this tab, one or two event nodes are displayed in the navigation tree. Create your text nodes as subnodes of these event nodes.

Exercise 5: Flow Control

Exercise Objectives

After completing this exercise, you will be able to:

- Create node conditions
- Use alternative nodes
- Create program nodes
- Create folders
- Create command nodes
- Create loops

Business Example

You are an employee of the Fly & Smile travel agency. You need to add the following details to the invoice that you are preparing:

Award a discount for some circumstances, transfer the English-speaking customers to a separate agent, and avoid inconvenient page breaks.

Optionally, you can also attach the terms and conditions of business and display the total sum of the invoice listed in order of currency.

Task 1:

Copy template

Note: Copy template for the form: BC470_TABLS

Package/Development class (for all exercises): ZBC470_##

Name of the form to be created: ZBC470_##_FLOWS

Model solution: BC470_FLOWS2

Application program for testing purposes: SAPBC470_DEMO

1. Copy the form that you used in the last task (ZBC470_##_TABLS) to ZBC470_##_FLOWS. Alternatively, copy the copy template (BC470_TABLS).

Continued on next page

Task 2:

Award discount

Note: You do not need the Form Painter for this exercise. You can therefore hide it to allow more space for the maintenance screen.

In the customer table read by the application program, there is an entry for discounts. If the customer has negotiated a discount, the following text is displayed at the end of the form: "A discount of ... percent has been taken into account"

1. As a subsequent node to GREETINGS, create a text node with the name DISCOUNT. Enter the reminder text detailed above.
2. Define the following condition for the text node: WA_CUSTOMERS-DISCOUNT > 0.
3. Activate your form and test it.

Task 3:

Choose *clerk*

Up to this point, only one clerk was responsible for creating the invoices. The travel agency Fly & Smile has now hired a second clerk. Clerk A is responsible for the US and Canada while clerk B is responsible for all other countries.

1. Convert the global constant CLERK into a variable.

   Note: You do not have to make any changes in text nodes because they use CLERK.

2. Create an alternative node called WHICH_CLERK at the correct position. If the customer lives in the US or Canada (WA_CUSTOMERS-COUNTRY has the value 'US' or 'CA'), the system should process the node TRUE. In all other cases, the system should process the node FALSE.
3. Create a program node called CLERK_A in the node TRUE and a program node called CLERK_B in the node FALSE. Specify a fine-sounding employee name each for the global field CLERK.

   Hint: To test whether your alternative actually works, look in the Data Browser (transaction code SE16), in the table SCUSTOM, and check for which customers the COUNTRY field contains, such as the value DE or US.

Continued on next page

Task 4:

Prevent page break

1. Ensure that the greeting form and the discount notification are on the same page. To do this, create a folder called PROTECT with the appropriate output option.

Task 5:

Optional: Create page for general terms and conditions of business

1. Create an additional page called TERMS for the general terms and conditions of business. Copy the main window onto this page. The TERMS page should be processed after (not under) the folder PROTECT. Create a suitable command node called SHOW_TERMS. After the command node, create a text node CONDITIONS, in which you can formulate business conditions as required.

Task 6:

Optional: Output totals by currency

For every booking line of the table BOOKINGS, collect the amounts in an internal table and output the content of this internal table before the greeting form in a loop.

1. Create a global variable WA_SUM of the type BC470_FORCUR, and an associated internal table IT_SUM of the type IT_BC470_FORCUR. For information: The structure type BC470_FORCUR contains the two fields FORCURAM (price of the booking in foreign currency) and FORCURKEY (currency), in accordance with the fields of the same name in the table SBOOK.
2. In the main area of the form table, such as after the text node in which you output the flight price, create a program line node with the name ADD_PRICES.

   The currency and amount of each booking should be inserted into the internal table IT_SUM as appropriate.

   Hint: This is best achieved using the syntax:

   ```
   move-corresponding wa_bookings to wa_sum
   collect wa_sum into it_sum
   ```

3. Before the closing GREETINGS, create a text node with the name SUMS and the text: “Totals by currency:”.

Continued on next page

4. After the text node SUMS, create a loop called SUM_LOOP for the totals by currency. Use the internal table IT_SUM to loop into the work area WA_SUM. The output should be sorted by currency.
5. Create a text node called SHOW_SUM as a subnode of the loop, and output the total and the currency.

Task 7:

Activating the invoice form

1. Activate your form and test it.

Solution 5: Flow Control

Task 1:

Copy template

Note: Copy template for the form: BC470_TABLS

Package/Development class (for all exercises): ZBC470_##

Name of the form to be created: ZBC470_##_FLOWS

Model solution: BC470_FLOWS2

Application program for testing purposes: SAPBC470_DEMO

1. Copy the form that you used in the last task (ZBC470_##_TABLS) to ZBC470_##_FLOWS. Alternatively, copy the copy template (BC470_TABLS).

   a) See the exercise in the Unit on the SAP From Builder.

Task 2:

Award discount

Note: You do not need the Form Painter for this exercise. You can therefore hide it to allow more space for the maintenance screen.

In the customer table read by the application program, there is an entry for discounts. If the customer has negotiated a discount, the following text is displayed at the end of the form: “A discount of ... percent has been taken into account”

1. As a subsequent node to GREETINGS, create a text node with the name DISCOUNT. Enter the reminder text detailed above.

   a) Using the context menu, create a text node with the name DISCOUNT as a subsequent node to GREETINGS. In the editor (tab page General Attributes for the text node), enter the text (“Note:...”). To ensure the correct percentage is output, drag the import parameter WA_CUSTOMERS-DISCOUNT to the appropriate point in the text node.

Continued on next page

2. Define the following condition for the text node: WA_CUSTOMERS-DISCOUNT > 0.

   a) Enter the following on the *Conditions* tab of the text node:

      Field Name: WA_CUSTOMERS-DISCOUNT

      Comparison operator: >

      Comparison value: 0

3. Activate your form and test it.

   a) Activate your form and test it using the program SAPBC470_DEMO.

Task 3:

Choose *clerk*

Up to this point, only one clerk was responsible for creating the invoices. The travel agency Fly & Smile has now hired a second clerk. Clerk A is responsible for the US and Canada while clerk B is responsible for all other countries.

1. Convert the global constant CLERK into a variable.

   Note: You do not have to make any changes in text nodes because they use CLERK.

   a) In the global definitions of the form, deselect the Constant field for the CLERK variable on the *Global Data* tab.

Continued on next page

2. Create an alternative node called WHICH_CLERK at the correct position. If the customer lives in the US or Canada (WA_CUSTOMERS-COUNTRY has the value 'US' or 'CA'), the system should process the node TRUE. In all other cases, the system should process the node FALSE.

   a) Your task is to create an alternative with program lines. The clerk should be determined within the program line. It is important to place the program lines at the correct position in the navigation tree as it does not make sense to output the clerk earlier. Since the clerk is output for the first time in the text node GREETINGS of the main window, you must insert the alternative with the program lines at an earlier point in the navigation tree. For example, you can create the alternative using the context menu of the main window. Call it WHICH_CLERK.

      On the *General Attributes* tab of the alternative, enter the two conditions: WA_CUSTOMERS-COUNTRY = 'US' and WA_CUSTOMERS-COUNTRY = 'CA'. Note the uppercase and lowercase for the entry, and the single quotation marks. Link these two conditions by an OR. To do this, place your cursor in the second condition line and then click the OR pushbutton, which is the second pushbutton on the maintenance screen.

3. Create a program node called CLERK_A in the node TRUE and a program node called CLERK_B in the node FALSE. Specify a fine-sounding employee name each for the global field CLERK.

   Hint: To test whether your alternative actually works, look in the Data Browser (transaction code SE16), in the table SCUSTOM, and check for which customers the COUNTRY field contains, such as the value DE or US.

   a) Use the context menu of the TRUE node, which was generated automatically when you created the alternative to create a program lines node called CLERK_A. Determine a value for the variable CLERK in the ABAP editor of the General Attributes tab, for example: CLERK = 'Mr. Miller'. Make the variable CLERK known to the program lines node by adding CLERK to the list of output parameters. Repeat these steps for the FALSE node and assign another name to this clerk.

      Hint: Now, collapse the Alternatives node for a better overview of the navigation tree.

Continued on next page

Task 4:

Prevent page break

1. Ensure that the greeting form and the discount notification are on the same page. To do this, create a folder called PROTECT with the appropriate output option.

   a) Use the context menu of the text node GREETINGS to create a folder called PROTECT. Drag the text nodes GREETINGS and DISCOUNT into the folder. (When you use Drag and Drop, the system asks whether to insert the nodes under or after PROTECT. Choose under). Select the *Page Protection* checkbox on the *Output Options* tab of the folder.

Task 5:

Optional: Create page for general terms and conditions of business

1. Create an additional page called TERMS for the general terms and conditions of business. Copy the main window onto this page. The TERMS page should be processed after (not under) the folder

Continued on next page

PROTECT. Create a suitable command node called SHOW_TERMS. After the command node, create a text node CONDITIONS, in which you can formulate business conditions as required.

a) Use the context menu of the page NEXT to create the page TERMS. Copy the main window from the page NEXT to the page TERMS. You can do this using Drag and Drop and holding down the CTRL key. After (not under) the PROTECT folder, create a command node with the name SHOW_TERMS, activate the field Go to New Page, and enter TERMS as a new page.

Hint: It does not matter whether you create the command node on the page FIRST, NEXT, or TERMS because the content of the main window is the same on all pages.

Hint: If you create the command node using the context menu of the folder PROTECT, it is created under the folder as standard. Therefore, you need to move it after the folder. Create the text node CONDITIONS after the command node, in which to enter your terms and conditions of business.

Hint: If you have used another procedure and your TERMS page has no main window, you need to ensure that no subsequent page is entered on the *General Attributes* tab page.

Task 6:

Optional: Output totals by currency

For every booking line of the table BOOKINGS, collect the amounts in an internal table and output the content of this internal table before the greeting form in a loop.

1. Create a global variable WA_SUM of the type BC470_FORCUR, and an associated internal table IT_SUM of the type IT_BC470_FORCUR. For information: The structure type BC470_FORCUR contains the two fields FORCURAM (price of the booking in foreign currency) and FORCURKEY (currency), in accordance with the fields of the same name in the table SBOOK.

   a) In the navigation tree, choose *Global Settings* → *Global Definitions*. On the *Global Data* tab page, enter WA_SUM TYPE BC470_FORCUR and IT_SUM TYPE IT_BC470_FORCUR.

Continued on next page

2. In the main area of the form table, such as after the text node in which you output the flight price, create a program line node with the name ADD_PRICES.

   The currency and amount of each booking should be inserted into the internal table IT_SUM as appropriate.

   Hint: This is best achieved using the syntax:
   ```
   move-corresponding wa_bookings to wa_sum
   collect wa_sum into it_sum
   ```

   a) In the main area of the table, use the context menu of the text node with the flight price output to create a program line node with the name ADD_PRICES. Enter the import parameter WA_BOOKINGS, and the output parameters WA_SUM and IT_SUM. Enter the following coding:

   ```
   move-corresponding wa_bookings to wa_sum
   collect wa_sum into it_sum
   ```

3. Before the closing GREETINGS, create a text node with the name SUMS and the text: “Totals by currency:”.

   a) Use the context menu of the PROTECT folder to create a text node, and call it SUMS. Enter the text.

4. After the text node SUMS, create a loop called SUM_LOOP for the totals by currency. Use the internal table IT_SUM to loop into the work area WA_SUM. The output should be sorted by currency.

   a) Use the context menu of the text node SUMS to create a loop with the name SUM_LOOP. On the Data tab, enter: Internal table IT_SUM INTO WA_SUM. In the lower section of the same Data tab page, enter FORCURKEY as a sort criterion.

5. Create a text node called SHOW_SUM as a subnode of the loop, and output the total and the currency.

   a) Use the context menu of the loop to create a text node, and call it SHOW_SUM. Now, output the booking amount and the booking currency (WA_SUM-FORCURAM and WA_SUM-FORCURKEY).

Continued on next page

Task 7:

Activating the invoice form

1. Activate your form and test it.

   a) Activate your form and test it using the program SAPBC470_DEMO.

Lesson Summary

You should now be able to:

- Explain the use of various flow control nodes
- Create node conditions
- Use alternative nodes
- Create program line nodes
- Create folders
- Create command nodes
- Create loops

Unit Summary

You should now be able to:

- Explain the use of various flow control nodes
- Create node conditions
- Use alternative nodes
- Create program line nodes
- Create folders
- Create command nodes
- Create loops

Test Your Knowledge

1. Forms are always processed in a predefined order.

   Determine whether this statement is true or false.

   □ True
   □ False

2. The fields used for comparisons in the Conditions tab must be either defined in the form interface, be a part of global definitions, or they must be system fields of the SAP Smart Forms.

   Determine whether this statement is true or false.

   □ True
   □ False

3. An alternative node consists of two subnodes, processed in a manner similar to that of the ABAP commands IF and ELSE.

   Determine whether this statement is true or false.

   □ True
   □ False

4. Coding that is used in different program line nodes can be moved into global subroutines and then called as and when required.

   Determine whether this statement is true or false.

   □ True
   □ False

5. Several nodes share the same condition. Instead of setting this condition individually for each node, you can create a folder and assign the condition to this folder.

   Determine whether this statement is true or false.

   □ True
   □ False

6. Identify the functions of a command node.

   Choose the correct answer(s).

   □ A Go to a new page
   □ B Reset paragraph numbering
   □ C Define free attributes for the spool request
   □ D Create subroutines

7. Loops are similar to tables.

Determine whether this statement is true or false.

- □ True
- □ False

Answers

1. Forms are always processed in a predefined order.

   Answer: False

   In some cases, the system determines which parts of a form should be processed at runtime. For example, in the case of forms with tables.

2. The fields used for comparisons in the Conditions tab must be either defined in the form interface, be a part of global definitions, or they must be system fields of the SAP Smart Forms.

   Answer: True

   You can use fields for comparisons only if they are defined in the form interface, a part of the global definitions, or they are system fields of SAP Smart Forms.

3. An alternative node consists of two subnodes, processed in a manner similar to that of the ABAP commands IF and ELSE.

   Answer: True

   An alternative node consists of two subnodes, processed in a manner similar to that of the ABAP commands IF and ELSE. The TRUE node of an alternative node is processed if the condition entered in the *General Attributes* tab is fulfilled, else the FALSE node is processed.

4. Coding that is used in different program line nodes can be moved into global subroutines and then called as and when required.

   Answer: True

   Coding that is used in different program line nodes can be moved into global subroutines and called as and when required. These subroutines can be created in the *Global Definitions* → *Form Routines* tab editor.

5. Several nodes share the same condition. Instead of setting this condition individually for each node, you can create a folder and assign the condition to this folder.

   Answer: True

   If several nodes share the same condition, you can create a folder and assign the condition to the folder.

6. Identify the functions of a command node.

   Answer: A, B, C

   A command node enables you to go to a new page, reset paragraph numbering, send print control to the output device, and define free attributes for spool requests.

7. Loops are similar to tables.

   Answer: True

   Like tables, data in loops is read line-wise. Therefore, loops are similar to tables.

Unit 7

Application Programs

Unit Overview

In this unit, you will be able to identify the various components of an application program and explain their interaction with SAP Smart Forms. In addition, you will be able to explain the significance, different parameters, and interfaces of the generated function module.

Unit Objectives

After completing this unit, you will be able to:

- Obtain an overview of an application program
- Explain the generated function module
- Customize application programs

Unit Contents

Lesson: Integration of Application Programs with SAP Smart Forms

Lesson Overview

In this lesson, you will be able to focus on the various components of an application program and explain its integration with SAP Smart Forms. In addition, you will be able to explain the significance, different parameters, and interfaces of the generated function module. Finally, you will be able to customize application programs.

Lesson Objectives

After completing this lesson, you will be able to:

- Obtain an overview of an application program
- Explain the generated function module
- Customize application programs

Business Example

The Fly & Smile travel agency constitutes a number of independent units located in different parts of the world. Each unit is headed by a business unit head and operates as an independent entity under the common parent company.

You have created an invoice for a booked flight. Now, you need to add appropriate lines of code to the body of an application program so that the invoice is processed.

Application Programs: Overview

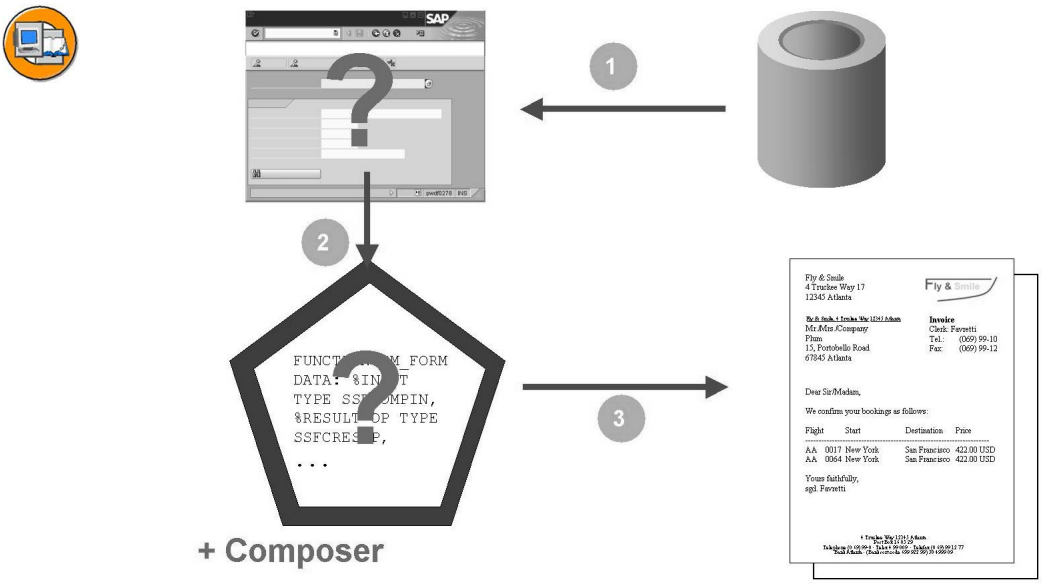


Figure 99: Recap: Creating Documents

Here, is a recap of the document creation process:

1. The transaction checks in Customizing the program the needs to be called. This program reads the data.
2. The transaction checks in Customizing which SAP Smart Form to use for the scenario chosen, calls the appropriate function module generated, and triggers the form processing process. The interface is filled with the data read.

When the form processing process is started, the form processor, that is, Composer is automatically called in the background. The Composer is responsible for formatting the texts according to the layout information stored in the form, filling fields with values at runtime and controlling the page breaks.

a) Data retrieval

b) Name of the generated function module?

c) Calling the function module

```
PROGRAM ...
DATA:
   ssf_name       TYPE tdsfname,
   func_mod_name  TYPE rs38L_fnam.

SELECT ... FROM ...
...

CALL FUNCTION 'SSF_FUNCTION_MODULE_NAME'
   EXPORTING
      formname = ssf_name
   IMPORTING
      fm_name  = func_mod_name.


LOOP AT ...
   CALL FUNCTION func_mod_name
      EXPORTING ...
      IMPORTING ...
ENDLOOP.
```

Figure 100: Components of the Application Program

From the perspective of the SAP Smart Forms, the application program consists of three parts:

1. The data is selected from the database, which is by far the most comprehensive part.
2. The name of the function module generated for the form must be determined. Background: The name differs depending on the form and system.

   Call the function module SSF_FUNCTION_MODULE_NAME and pass the form name to it. This name is typed as TYPE tdsfname. The return value (import parameter) you get is the name of the generated function module as TYPE rs38L_fnam.
3. Actual form processing starts. The generated function module is called once for each document to be created, that is, for example, once for each customer for which you want to create an invoice.

Generated Function Module

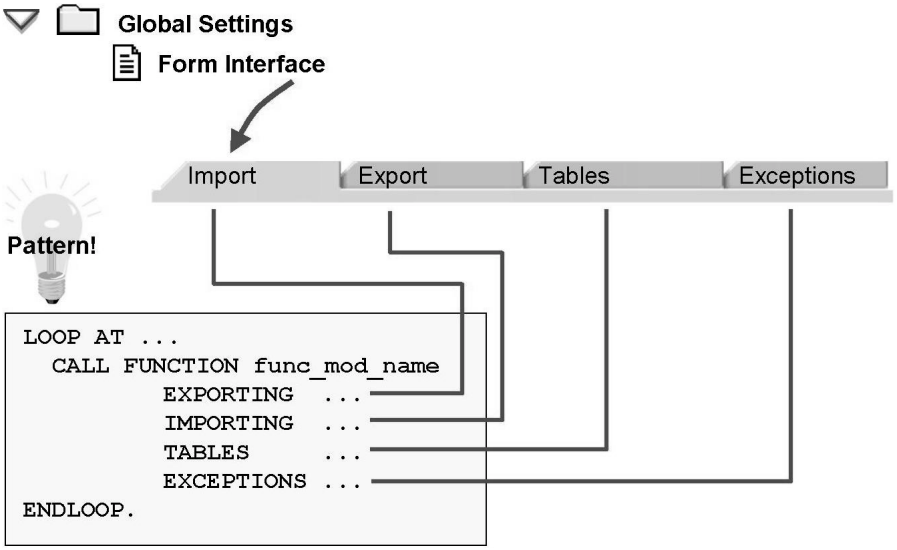


Figure 101: The Generated Function Module

Each generated function module has the interface that you defined on the four tabs of the global settings for the SAP Form Builder.

To integrate function modules into programs, you can generally use the sample statement of the ABAP Editor. However, since the name of the function module differs depending on the form and system, you must use a workaround:

In the Form Builder choose *Environment → Function module name* and then use CTRL+Y and CTRL+C to copy the name to the clipboard.

Use CTRL+V to paste the name of the function module into the sample statement for CALL FUNCTION. The call of the function module with the correct interface is then inserted at the cursor position.

Replace the name after CALL FUNCTION with the variable, which is filled by calling SSF_FUNCTION_MODULE_NAME and contains the current name of the generated function module at runtime.

```
CALL FUNCTION func_mod_name
  EXPORTING
*   control_parameters    =
*   output_options        =
*   user_settings         = 'X'
    wa_customers          =
*   color                 =

* IMPORTING
*   job_output_info       =
*   job_output_options    =

  TABLES
    it_bookings           =

* EXCEPTIONS
*   formatting_error      = 1
*   others                = 2.

* ... Error handling
IF sy-subrc <> 0.   … ENDIF.
```

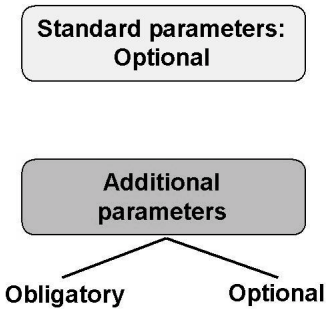


Figure 102: Interface Parameters

The generated function module has both required and optional parameters:

Optional parameters. All standard parameters, available in every form, are optional. Your name and typing cannot be changed. They include:

- control_parameters: General output control
- output_options: Output options (see structure ssfcompop in the Dictionary)
- user_settings: If set to **X**, the user defaults for spool control are used. Otherwise, the output_options values for the printer, immediate output, and spool retention period are evaluated.
- Parameters for archiving
- Parameters of the Business Communication Interface for sending forms as e-mails
- document_output_info: Number of pages output (field tdfpages)
- job_output_info, job_output_options: Structures with information on the output, such as with XML output

As of SAP Web AS 6.10, you can also mark those separate importing parameters of the form without default values as **optional.**

Exporting parameters of the form are always **optional.**

Required parameters: All table parameters. For SAP R/3 4.6C, all importing parameters of the form are also obligatory – they have one default value.

See the SAP note 433988 on the standard parameters.

You handle errors as with other function modules by querying the return code (sy-subrc) directly after the call of the function module.

CONTROL_PARAMETERS
(Export parameters of the function module generated)
Type: `ssfctrlop`

`no_open`	No new spool request
`no_close`	Do not close spool request
`device`	Output device (`'PRINTER'`, `'TELEFAX'`, `'MAIL'`)
`no_dialog`	No dialog box for output
`preview`	Print preview
`langu`	Language
`startpage`	Start page ≠ default

Figure 103: Control Structure CONTROL_PARAMETERS

One of the most important parameters of the generated function module is control_parameters. The following fields are available:

no_open and no_close: These parameters allow you to add several forms to a spool request. To do this, set the parameters as follows:

- First call: no_open = space, no_close = “X”.
- All subsequent calls: no_open = “X”, no_close = “X”.
- Last call: no_open = “X”, no_close = space.
- device: Output device (**PRINTER**, **TELEFAX**, **MAIL**). The default value is **PRINTER**.
- no_dialog: No dialog box for output.
- preview: Print preview
- langu: Language in which you want to print the form
- replangu1, replangu2, replangu3: Alternative languages if the form does not exist in langu
- startpage: Start page other than the top page in the navigation tree of the SAP Form Builder
- getotf: No printout, display or faxing, but OTF (Output Text Format) output to the table job_output_info-otfdata.

Customizing Application Programs

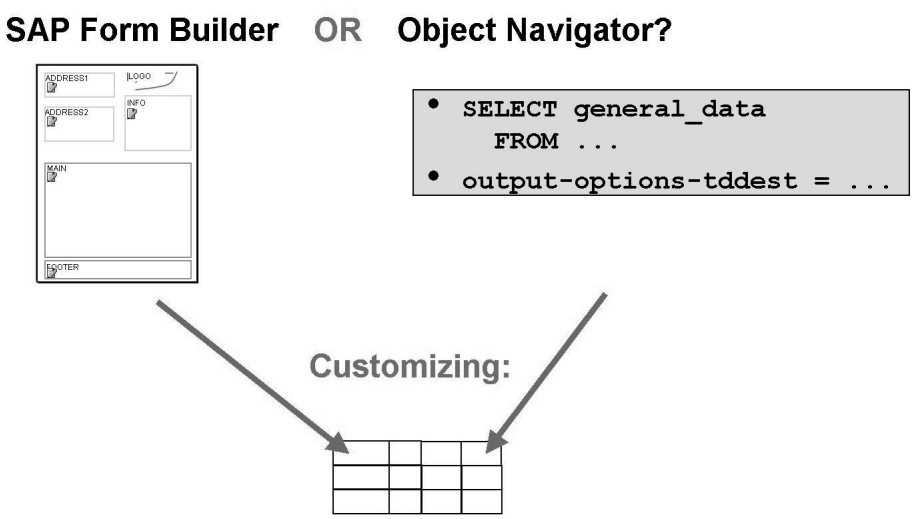


Figure 104: Changes to the Application Program

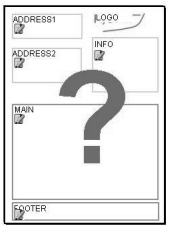
When do you need to modify the application program?

You want to retrieve additional data to be used in all documents of the print run. For performance reasons, you should not select this data in the form since the data would be selected multiple times.

You want to make output-related settings, such as disable the print preview.

The transaction determines which part of the application program you actually need to change, such as a function module or a subroutine. As a rule, you should change original SAP programs in exceptional cases only.

If you want to use your own modified copies instead of the SAP originals, that is, programs, forms, or texts you must make the appropriate entries and settings in Customizing. Again, the application determines which settings you have to make in Customizing and in which part of Customizing you have to make them.



Customizing: When do you use which form?

Application	Scenario	Program	Form
XYZ	1	Z_PROG1	Z_SMARTIE_1
	2	Z_PROG1	Z_SMARTIE_2
	3	Z_PROG1	Z_SMARTIE_3
ABC	1	Z_MY_PROG2	Z_MY_SMARTIE_1
	2	Z_MY_PROG2	Z_MY_SMARTIE_2

Figure 105: Customizing

The settings you have to make in Customizing depend on the application you use.

Depending on the application, you have to decide between SAPscript or SAP Smart Forms.

You also determine the form is to be used for the application.

The reason behind this is that you should not directly modify the forms delivered by SAP but always copy them to your customer namespace and change this copy. You must inform the transaction about the name of your own form - you do this in Customizing.

Second, some applications allow you to choose between various forms for different scenarios. For example, you can determine in Customizing that for the dunning procedure an individual form should be used for each dunning level, amount and duration of late payment.

Finally, you have to add the print program to Customizing. For example, this can be a function module (in dunning) or a report with specification of the relevant subroutine (in the delivery note). For examples on Customizing, see the appendix. Also, see the Customizing documentation on the relevant applications and SAP notes, particularly, SAP note 430621.

Exercise 6: Optional: Integration of Application Programs with SAP Smart Forms

Exercise Objectives

After completing this exercise, you will be able to:

- Insert the generated function module for the invoice form into a program
- Set output options for the function module

Business Example

You are an employee of the Fly & Smile travel agency. You need to add the appropriate lines of code to the body of an application program to process the invoice form BC470_FLOWS2.

Task:

Integrating into Application Programs

Note: Copy template for the program: SAPBC470_PROGT

Name of the program to be created: ZBC470_##_PROGS

Package/Development class (for all exercises): ZBC470_##

Name of the form to be used: BC470_##_FLOWS2 (You can of course also use your own form)

Model solution: SAPBC470_PROGS

We recommend that you work with two sessions in this exercise so that you can switch between the ABAP Workbench and the SAP Form Builder.

1. **Copy template**

   In the ABAP Workbench, copy the template program SAPBC470_PROGT to ZBC470_##_PROGS. This program provides a selection screen and retrieves the data. You are responsible for the remaining parts.

   Determine the function module name

Continued on next page

The program must know the name of the function module generated for the form BC470_FLOWS2. Call the function module SSF_FUNCTION_MODULE_NAME at the end of the program code. To do this, use the sample statement. This ensures that the interface is correct.

Create a variable called FUNC_MOD_NAME of the type RS38L_FNAM. This variable should contain the name of the generated function module for the form after the function module is called.

Call the generated function module in a loop.

The customer data is stored in the internal table IT_CUSTOMERS. Create a loop in the program at IT_CUSTOMERS into the existing work area WA_CUSTOMERS.

Call the generated function module in this loop. To do this, follow these steps:

In the SAP Form Builder, determine the name of the generated function module for the form BC470_FLOWS2 by choosing Function module name and then use CTRL+Y and CTRL+C to copy the name to the clipboard.

Use CTRL+V to pass the name of the function module in the sample statement for CALL FUNCTION. The call of the function module with the correct interface is then inserted at the cursor position.

Fill the required interface parameters WA_CUSTOMERS, COLOR, and IT_BOOKINGS with the identically named parameters of the program. These already exist and are filled with values.

Test your program

Optional: Set output options

You will have noticed that the system displays the printer settings dialog box for each customer. Avoid this by filling the parameters CONTROL_PARAMETERS and OUTPUT_OPTIONS with suitable values. Create a variable called CONTROL_PARAMETERS of the type SSFCTRLOP and a variable called OUTPUT_OPTIONS of the type SSFCOMPOP. Go to Dictionary to get a description of the relevant fields or look in your course materials. To ensure that your options are used you must set the parameter USER_SETTINGS to space.

Optional: Use only one spool request.

Ensure that all invoices are output within a single spool request. Assign appropriate values to the fields NO_OPEN and NO_CLOSE of the interface parameter CONTROL_PARAMETERS.

Solution 6: Optional: Integration of Application Programs with SAP Smart Forms

Task:

Integrating into Application Programs

Note: Copy template for the program: SAPBC470_PROGT

Name of the program to be created: ZBC470_##_PROGS

Package/Development class (for all exercises): ZBC470_##

Name of the form to be used: BC470_##_FLOWS2 (You can of course also use your own form)

Model solution: SAPBC470_PROGS

We recommend that you work with two sessions in this exercise so that you can switch between the ABAP Workbench and the SAP Form Builder.

1. **Copy template**

In the ABAP Workbench, copy the template program SAPBC470_PROGT to ZBC470_##_PROGS. This program provides a selection screen and retrieves the data. You are responsible for the remaining parts.

Determine the function module name

The program must know the name of the function module generated for the form BC470_FLOWS2. Call the function module SSF_FUNCTION_MODULE_NAME at the end of the program code. To do this, use the sample statement. This ensures that the interface is correct.

Create a variable called FUNC_MOD_NAME of the type RS38L_FNAM. This variable should contain the name of the generated function module for the form after the function module is called.

Call the generated function module in a loop.

The customer data is stored in the internal table IT_CUSTOMERS. Create a loop in the program at IT_CUSTOMERS into the existing work area WA_CUSTOMERS.

Continued on next page

Call the generated function module in this loop. To do this, follow these steps:

In the SAP Form Builder, determine the name of the generated function module for the form BC470_FLOWS2 by choosing Function module name and then use CTRL+Y and CTRL+C to copy the name to the clipboard.

Use CTRL+V to pass the name of the function module in the sample statement for CALL FUNCTION. The call of the function module with the correct interface is then inserted at the cursor position.

Fill the required interface parameters WA_CUSTOMERS, COLOR, and IT_BOOKINGS with the identically named parameters of the program. These already exist and are filled with values.

Test your program

Optional: Set output options

You will have noticed that the system displays the printer settings dialog box for each customer. Avoid this by filling the parameters CONTROL_PARAMETERS and OUTPUT_OPTIONS with suitable values. Create a variable called CONTROL_PARAMETERS of the type SSFCTRLOP and a variable called OUTPUT_OPTIONS of the type SSFCOMPOP. Go to Dictionary to get a description of the relevant fields or look in your course materials. To ensure that your options are used you must set the parameter USER_SETTINGS to space.

Optional: Use only one spool request.

Ensure that all invoices are output within a single spool request. Assign appropriate values to the fields NO_OPEN and NO_CLOSE of the interface parameter CONTROL_PARAMETERS.

a)

```
The optional parts are printed in bold.
*********************************************************************
*  Report  SAPBC470_PROGS
*********************************************************************
*  Solution for the exercise of unit 8 of BC470
*********************************************************************

REPORT  sapbc470_progs.
TABLES: spfli

DATA:
  it_bookings         TYPE ty_bookings,
  it_customers        TYPE ty_customers,
```

Continued on next page


```
  wa_customers        TYPE scustom,
  color(4).

DATA:
    func_mod_name     TYPE rs38l_fnam.

DATA:
  output_options      TYPE ssfcompop,   " optional part of exercise
  control_parameters  TYPE ssfctrlop.   " optional part of exercise


* selection-screen
SELECTION-SCREEN COMMENT 1(45) text-se1.

SELECTION-SCREEN SKIP 1.
SELECT-OPTIONS:
  so_cust FOR wa_customers-id DEFAULT 1    TO 3,
  so_carr FOR spfli-carrid    DEFAULT 'AA' TO 'LH'.

* printing options
SELECTION-SCREEN SKIP 1.
PARAMETERS:
  pa_prnt  TYPE tsp03-padest DEFAULT 'P280' OBLIGATORY
  VISIBLE LENGTH 4.

* graphics
SELECTION-SCREEN SKIP 2.
SELECTION-SCREEN COMMENT 1(30) text-se2.
PARAMETERS:
  pa_col     RADIOBUTTON GROUP col,
  pa_mon     RADIOBUTTON GROUP col DEFAULT 'X'.
*********************************************************************
START-OF-SELECTION.

* set color for company logo
  IF pa_col = 'X'.
    color = 'BCOL'.
  ELSE.
    color = 'BMON'.
  ENDIF.


  SELECT * FROM  scustom
    INTO TABLE it_customers
    WHERE id IN so_cust
```

Continued on next page


```
    ORDER BY PRIMARY KEY.

  SELECT * FROM  sbook
    INTO TABLE it_bookings
    WHERE customid IN so_cust AND
          carrid   IN so_carr
    ORDER BY PRIMARY KEY.




**********************************************************************
**********************************************************************
* Your Coding here:

* find out the name of the generated function module
  CALL FUNCTION 'SSF_FUNCTION_MODULE_NAME'
       EXPORTING
            formname           = 'BC470_FLOWS2'
       IMPORTING
            fm_name            = func_mod_name
       EXCEPTIONS
            no_form            = 1
            no_function_module = 2
            OTHERS             = 3.
  IF sy-subrc <> 0.
    MESSAGE ID sy-msgid TYPE sy-msgty NUMBER sy-msgno
           WITH sy-msgv1 sy-msgv2 sy-msgv3 sy-msgv4.
  ENDIF.


* set output options (optional)
  output_options-tddest        = pa_prnt.

  control_parameters-no_dialog = 'X'.
  control_parameters-preview   = 'X'.


* process the form for every customer in it_customers
  LOOP AT it_customers
    INTO wa_customers.

* make sure only one spool request is used
    AT FIRST.
      control_parameters-no_close = 'X'.
```

Continued on next page

```
    ENDAT.
    AT LAST.
      control_parameters-no_close = space.
    ENDAT.
* call the generated function module
    CALL FUNCTION func_mod_name
       EXPORTING
* The following three parameters belong to the optional
* part of the exercise.
            control_parameters = control_parameters
            output_options     = output_options
            user_settings      = space

            wa_customers       = wa_customers
            color              = color
       TABLES
            it_bookings        = it_bookings
       EXCEPTIONS
            formatting_error   = 1
            internal_error     = 2
            send_error         = 3
            user_canceled      = 4
            OTHERS             = 5.
    IF sy-subrc <> 0.
      MESSAGE ID sy-msgid TYPE sy-msgty NUMBER sy-msgno
              WITH sy-msgv1 sy-msgv2 sy-msgv3 sy-msgv4.
    ENDIF.
    control_parameters-no_open = 'X'.
  ENDLOOP.
```

Lesson Summary

You should now be able to:

- Obtain an overview of an application program
- Explain the generated function module
- Customize application programs

Unit Summary

You should now be able to:

- Obtain an overview of an application program
- Explain the generated function module
- Customize application programs

Test Your Knowledge

1. You can directly modify the forms delivered by SAP.

   Determine whether this statement is true or false.

   □ True

   □ False

2. _____ and _____ parameters allow you to add several forms to a spool request.

   Fill in the blanks to complete the sentence.

3. The generated function module needs to be called only once for all the forms to be processed.

   Determine whether this statement is true or false.

   □ True

   □ False

Answers

1. You can directly modify the forms delivered by SAP.

   Answer: False

   You should not directly modify the forms delivered by SAP but always copy them to your customer namespace and change this copy.

2. no_open and no_close parameters allow you to add several forms to a spool request.

   Answer: no_open, no_close

3. The generated function module needs to be called only once for all the forms to be processed.

   Answer: False

   The generated function module needs to be called every time a form needs to be processed.

Unit 8

Smart Styles

Unit Overview

In this unit, you will get an overview on the various smart styles and use them in different nodes. In addition, you will be able to use the style builder to create or modify new or existing smart styles.

Unit Objectives

After completing this unit, you will be able to:

- Use styles and Smart Styles in forms
- Use the Style Builder
- Create paragraph and character formats
- Use formats in text

Unit Contents

Lesson: Form Styles

Lesson Overview

In this lesson, you will identify the use of styles and Smart Styles in forms. In addition, you will learn to use the Style Builder to create or modify new or existing Smart Styles. You will also learn to create paragraph and character formats and use formats in text.

Lesson Objectives

After completing this lesson, you will be able to:

- Use styles and Smart Styles in forms
- Use the Style Builder
- Create paragraph and character formats
- Use formats in text

Business Example

The Fly & Smile travel agency constitutes a number of independent units located in different parts of the world. Each unit is headed by a business unit head and operates as an independent entity under the common parent company. You have already created an invoice for a booked flight. Now, you need to copy and enhance the existing style of the invoice form.

Using Styles in Forms: Overview

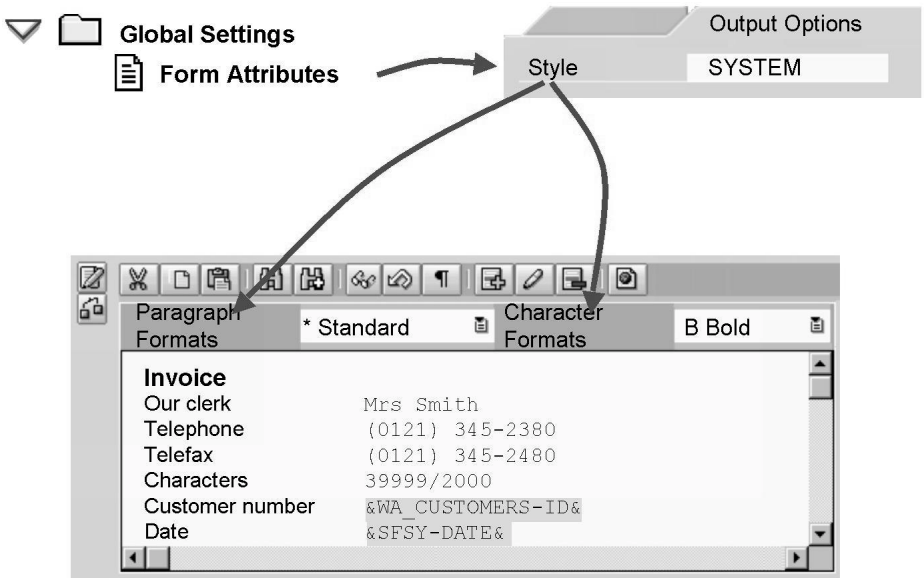


Figure 106: Styles: Usage in Forms

If you want to format texts in a form, select a character and/or paragraph format from the format lists in the editor. The formats offered in this list depend on the style, that is, Smart Style chosen. The following rules apply:

Each form must be assigned a style using the *Output options* tab of the form attributes. The default for new forms is the style *System*.

You can assign a different style to most nodes using the *Output options* tab. This style then applies to all lower-level nodes.

Styles are collections of character and paragraph formats that are similar to format templates in common word-processing programs. You cannot choose a format that has not been added to a style. This ensures that all forms using the same style have a consistent text design.

Note that SAPscript styles of include texts are not considered irrespective of whether they are set as dynamic or static. If formats are used in include texts that are not defined in the Smart Style that applies to the include text, then these formats are ignored.

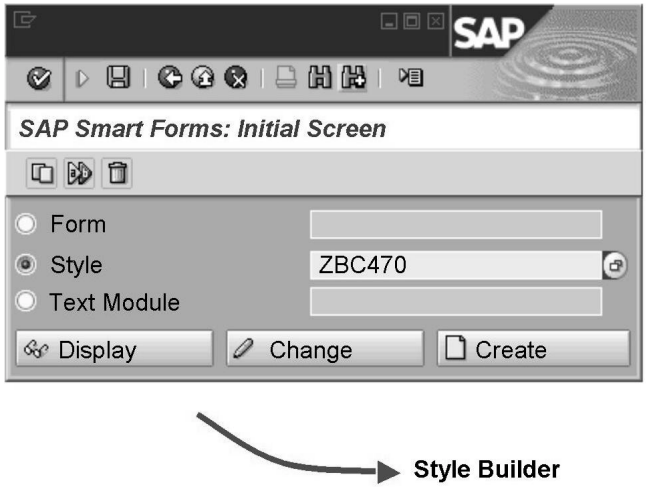


Figure 107: Smart Styles: Initial Screen

To create a Smart Style, you can either select the *Style* radio button on the initial screen of the SAP Smart Forms transaction, or start transaction SMARTSTYLES directly. Enter the name of the style. It is recommended that you only change styles in your customer namespace, that is, styles beginning with Y or Z. If required, copy the SAP styles to your customer namespace. To do this, click the corresponding pushbutton or choose *Smart Styles → Copy from the menu.*

Like forms, styles are integrated with the SAP R/3 transport system. This is why the system prompts you for a development class when you first save a style.

You can see the development class assigned - and other style-related information such as who created or changed the style - on the *Administration* tab of the style header data. It makes no sense to enter a style variant for SAP R/3 4.6C here.

Using Style Builder

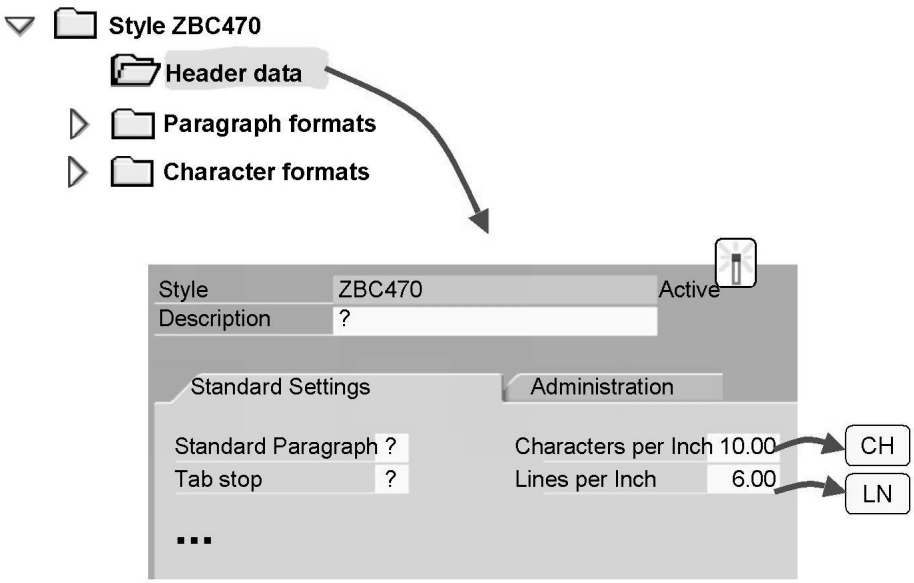


Figure 108: The Style Builder/Header Data I

The maintenance tool for Smart Styles, called the Style Builder, consists of two areas.

The navigation tree is on the left-hand side.

Detailed information on the element you select in the tree is displayed on the right-hand side. You can also see a preview of the element there.

Like forms, styles can exist in both active and inactive versions. You activate a style by clicking the corresponding pushbutton or by choosing *Style → Activate*.

The header data has two tabs: *Standard settings* and *Administration*. Part of the *Standard settings* can be seen on the above graphic.

Standard paragraph: Here, you specify which paragraph format is to be used to format the text if the text has no explicit formatting. In the format list of the editor, this standard paragraph is marked with an asterisk (*). You must specify the standard paragraph and also the description of the style if you want to activate the style. Before you can make an entry here, you must have defined at least one paragraph format.

Tab stop: Here, you set the distance between the standard tabs. These are always left-aligned tab stops.

In the *Characters per inch* field, you enter the size of the CH unit of measure, for horizontal size specifications, in the style. Similarly, you determine the LN unit of measure, for vertical size specifications, in the *Lines per inch* field.

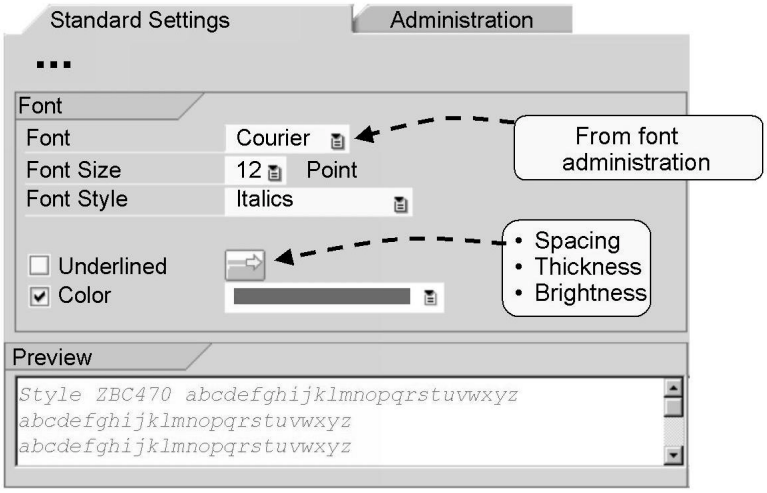


Figure 109: Header Data II

The lower part of the *Standard settings* tab contains the remaining options. The selections you make in the *Font* group box apply to all paragraph and character formats of the style - unless you explicitly specify individual font attributes for these formats.

The *font families* available depend on the font administration settings.

The *font size* determines the height of the font in points (PT). The default size is 12.

You can choose between the following *font styles*: *Bold*, *italic*, *bold and italic*, or none.

If you select the *Underlined* checkbox, you can make further settings by clicking the pushbutton on the right. Note that not all printers support this option.

With *Spacing*, you determine the distance between the underline and the base line. The default is 0, which means that the underline is on the base line. If you enter a negative value, you do not underline but strike through the text.

Thickness: The default is 1 point.

Brightness: Enter a percentage value. A brightness of 0 percent means the full shade.

For technical reasons, the various underlining types cannot be differentiated from each other in the preview of the Style Builder or the editor. This is only possible in the print preview of the application program or in the printout itself.

The selection field *Standard formats permitted* for SAP R/3 4.6C does not have a function.

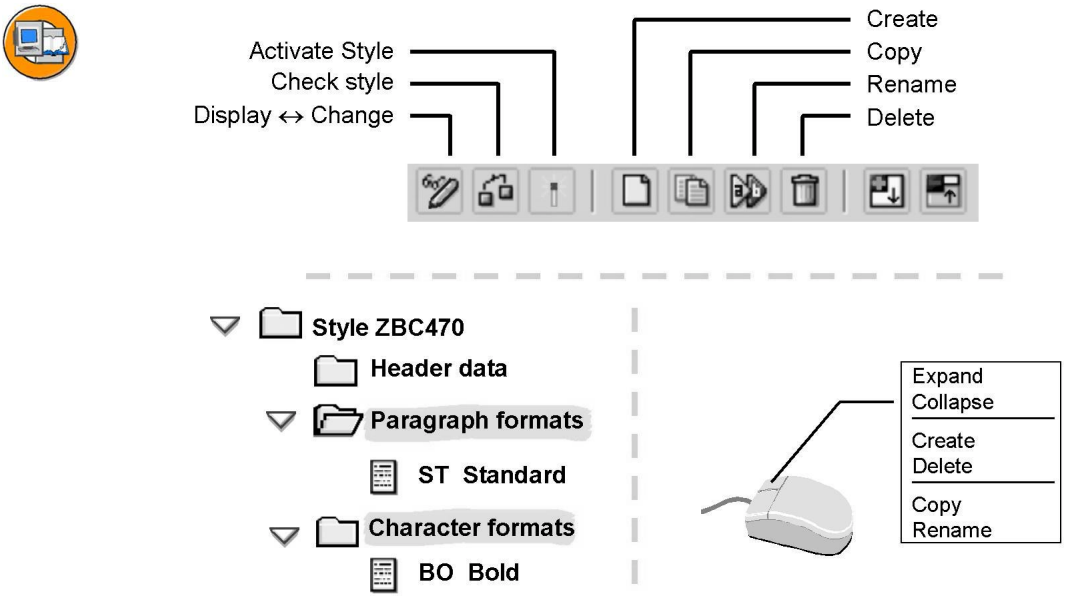


Figure 110: Editing Character and Paragraph Formats

Character and paragraph formats are displayed in the navigation tree. They always have a two-character technical name and a description.

To create, copy, or rename a character or paragraph format, use the *Edit* menu, the pushbuttons, or the context menu (right mouse button). The action you choose refers to the node currently selected in the tree. You can select nodes with a double-click.

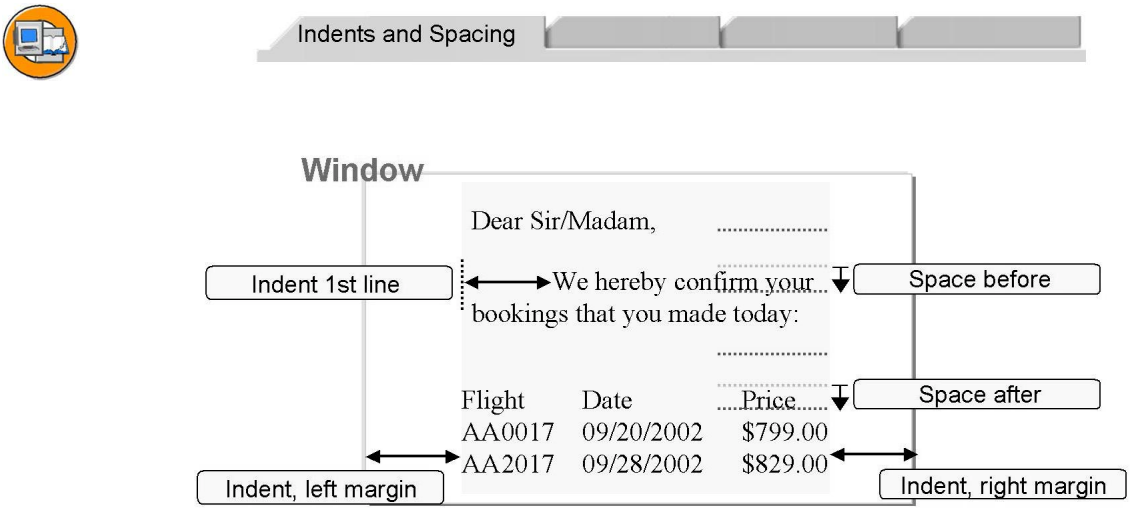


Figure 111: Paragraph Formats: Indents and Spacing

On the *Indents and spacing* tab of a paragraph format, you can make the following entries:

Indent:

Left/right margin of indent: Amount of space between the window and the text margin.

In addition to the left margin, you can determine a greater indent for the first line of a paragraph.

Spacing:

Space before: This is the space by which the current paragraph is moved down. Similarly, the *space before* defines the space by which the next paragraph is moved down.

Line spacing: Note that the line spacing is not adjusted automatically if you use a larger font size. To avoid overlappings, you must therefore change the line spacing if required.

Text flow:

You can determine that text within a paragraph should always be printed on one page (*Page protection* checkbox). If a paragraph does not fit onto a page, the system automatically inserts a page break before this paragraph.

If you select the *Next paragraph same page* checkbox, the next paragraph is printed on the same page as the current paragraph.

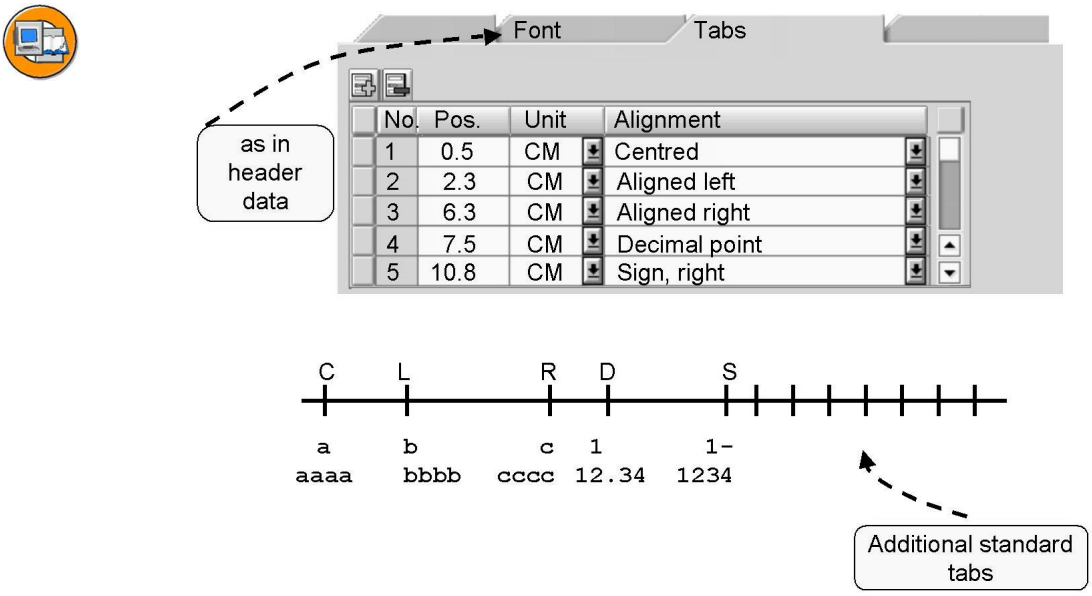


Figure 112: Paragraph Formats: Tabs

The *Font* tab has the same fields as the *Standard settings* tab of the style. If you do not make any settings, the system uses the settings of the header data: *Font family, Font size, Font style, Underlined, and Color*. You can

override the header data entries by specifying a different font. Note that standard settings such as the color or the font family are not displayed in the preview of a paragraph format.

On the *Tabs* tab page, you define individual tab stops for the paragraph format.

The *Sign* alignment type lets you define numbers right aligned at the tab stop position, considering the minus sign or implied blank space at the end of the number.

The *Alignment with decimal point* alignment type lets you align numbers with decimal points printed at the tab stop position.

After the last tab stop, the standard tab stops of the standard settings, that is, header data are used.

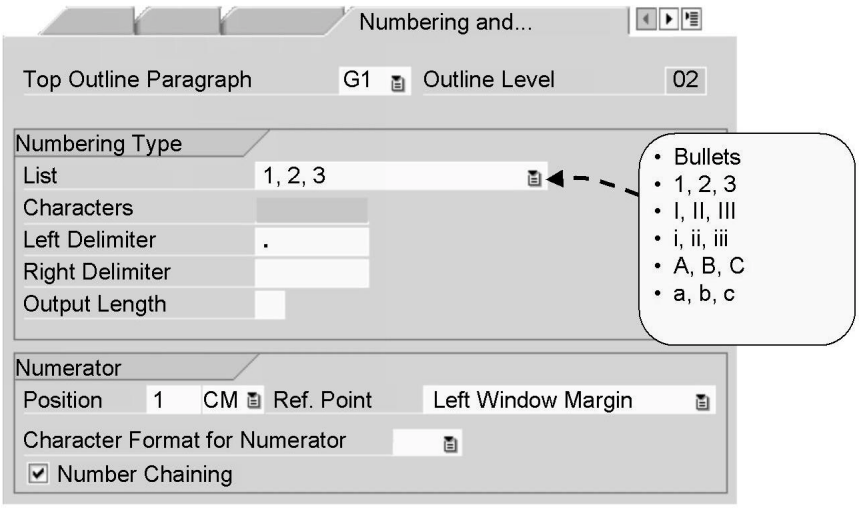


Figure 113: Paragraph Formats: Numbering and Outline I

You can define paragraphs that have outline characters, for example, or are numbered automatically. To do this, go to the *Numbering and outline* tab.

If you want to use a multi-level numbering or outline such as 1, 1.1, 1.2, 1.2.1, and so on, you must create a separate paragraph for each level. It is recommended that you use the *Copy* function to do this. As the *Top outline paragraph,* you enter the top paragraph in the hierarchy, which controls all other outline or numbering levels. The individual levels are inserted as subnodes of the top outline paragraph into the navigation tree, and the outline level is automatically entered in the corresponding field on the tab. An example is shown on the next page.

Choose a numbering type:

List character: The list character printed at the beginning of the paragraph is the one that you enter in the *Character* field. You can use eight digits at the most.

Arabic numbers, Roman numerals (lowercase or uppercase) or letters (lowercase or uppercase). You can also specify a *left* and a *right delimiter*, such as an angle bracket to define numberings of the form "a), b), c)", and so on.

In the *Position* field, you enter the space between the numbering/outline character and the left window margin. Make sure that the paragraph margin is not affected by the numerator margin. To avoid overlapping, define a paragraph margin that is large enough.

If you select *Number chaining*, the system uses the previous numerator for multi-level numbering. Example: If level G1 = 1, then level G2 (with number chaining) = 1**.2**.

Example: Paragraph Definition

	List	Left Delimiter	Right Delimiter	Number Chaining	Position
G1	1, 2, 3				
G2	1, 2, 3	.		☑	0.5 cm
G3	a, b, c	.	)	☑	1 cm

Text

	Outline Level	Paragraph	Top Paragraph
1	1	G1	G1
1.1	2	G2	G1
1.1.a)	3	G3	G1
1.1.b)	3	G3	G1
1.2	2	G2	G1
2	1	G1	G1

Figure 114: Paragraph Formats: Numbering and Outline II

This is an example of a three-level outline. G1 is the top level.

Outlines and numbering are not displayed in exact WYSIWYG mode in the preview of the editor. For a real WYSIWYG display, you need to test the function module of the form.

If you want to reset the numbering of a paragraph to its initial value, create a command node in your form and use the command *Reset paragraph numbering*. See *Flow Control*.

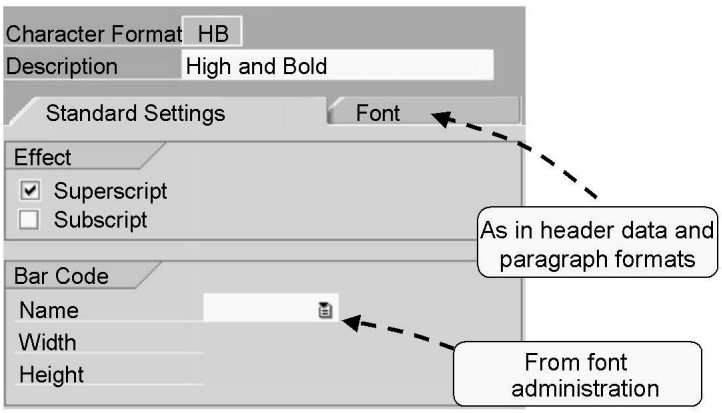


Figure 115: Character Formats

A character format has two tabs: *Standard settings* and *Font*. The *Font* tab has the same fields as the *Standard settings* tab of the style or the *Font* tab of a paragraph. The entries you make here override the settings of the header data or the paragraph format used in the text: *Font family, Font size, Font style, Underlined, and Color*. Note that font-related settings made in the standard settings or a paragraph format are not displayed in the preview of a character format.

On the *Standard settings* tab, you can set the attributes *Superscript* and *Subscript*.

You can also choose a *barcode*.

The height and width of a barcode are automatically adopted from the font administration.

You cannot combine the *Superscript* or *Subscript* attribute with a barcode.

Barcodes are not displayed in the preview of the Style Builder. The print preview of the application program roughly displays them in their correct size as a pattern of lines.

Exercise 7: Smart Styles

Exercise Objectives

After completing this exercise, you will be able to:

- Maintain Smart Styles

Business Example

You are working for the Fly & Smile travel agency. As an employee, you are responsible for creating invoices of booked flights for all customers. You have already created the invoice forms. Now, you need to copy and enhance the existing style and use it in the forms.

Task 1:

Copy template.

Note: Copy template for the style: BC470

Name of the style to be created: ZBC470_##_STYLS

Package/Development class (for all exercises): ZBC470_##

Name of the form to be used: ZBC470_##_FLOWS

Only use your own form ZBC470_##_FLOWS if you have worked through the optional task in unit 7 where you had to create the TERMS page. Else, copy the form BC470_FLOWS2 to ZBC470_##_STYLS

Model solution for the style: BC470_STYLS

Application program for testing purposes: SAPBC470_DEMO

1. Copy the style BC470 to ZBC470_##_STYLS.

Task 2:

Change standard settings.

1. Set Times, 12 pt as the standard font type.

Task 3:

Change and create paragraph formats.

1. Specify for the paragraph format AS that the lines of a paragraph should not be separated by a page break.

Continued on next page

2. Use the standard font family for the paragraph format AS.
3. For the paragraph format TO, change the positions for the two tab stops to 13 and 14 cm.
4. **Optional:** Create the paragraph formats G1 and G2. These formats should outline paragraphs as follows:

   I This paragraph has format G1

   II This paragraph has format G1

   II a) This paragraph has format G2

   II b) This paragraph has format G2

   III This paragraph has format G1

    **Note:** The preview does not support WYSIWYG.

Task 4:

Creating character formats.

1. Create at least one character format of your choice.

Task 5:

Activating styles.

1. Activate your style.

Task 6:

Assigning styles.

1. Assign your style to the text on the page TERMS of your form.

   Test your formats.

   Optional: Format several paragraphs using the outline paragraphs G1 and G2 you created. If necessary, add a few lines of text.

Task 7:

Testing your form.

1. Test your form using the program SAPBC470_DEMO.

Solution 7: Smart Styles

Task 1:

Copy template.

Note: Copy template for the style: BC470

Name of the style to be created: ZBC470_##_STYLS

Package/Development class (for all exercises): ZBC470_##

Name of the form to be used: ZBC470_##_FLOWS

Only use your own form ZBC470_##_FLOWS if you have worked through the optional task in unit 7 where you had to create the TERMS page. Else, copy the form BC470_FLOWS2 to ZBC470_##_STYLS

Model solution for the style: BC470_STYLS

Application program for testing purposes: SAPBC470_DEMO

1. Copy the style BC470 to ZBC470_##_STYLS.

   a) On the initial screen of transaction SMARTFORMS, select the *Style* radio button and enter BC470 as the name.

      Alternatively, you can start the separate maintenance transaction SMARTSTYLES and enter the name of the style there. It does not matter which point of entry you choose since the subsequent steps are all identical.

      Copy the style by choosing the *Copy* pushbutton, which is the first pushbutton in the application toolbar. Enter the new name, ZBC470_##_STYLS, as the target style, and assign your package (development class) on the dialog box that is displayed next.

Task 2:

Change standard settings.

1. Set Times, 12 pt as the standard font type.

   a) Select the *Header data* node in the navigation tree. On the *Standard settings* tab of the maintenance screen, select Times as the font family and 12 as the font size.

Continued on next page

Task 3:

Change and create paragraph formats.

1. Specify for the paragraph format AS that the lines of a paragraph should not be separated by a page break.

   a) In the navigation tree, select the node *Paragraph formats* and then the node AS. On the Indents and spacing tab of the maintenance screen, select the Page protection checkbox.

2. Use the standard font family for the paragraph format AS.

   a) You must not enter a font family on the Font tab of the paragraph format AS to ensure that the standard font of the header data is used.

3. For the paragraph format TO, change the positions for the two tab stops to 13 and 14 cm.

   a) In the navigation tree, select the node *Paragraph formats* and then the node TO. On the maintenance screen, change the tab *stop positions* on the *Tabs* tab.

4. **Optional:** Create the paragraph formats G1 and G2. These formats should outline paragraphs as follows:

   I This paragraph has format G1

   II This paragraph has format G1

   II a) This paragraph has format G2

   II b) This paragraph has format G2

Continued on next page

III This paragraph has format G1

Note: The preview does not support WYSIWYG.

a) **Optional:** From the context menu of the node Paragraph formats, choose Create node. The system displays a dialog box. Enter G1 and choose *Enter*. Enter a description on the maintenance screen. For example, "Outline, Roman numerals". On the Indents and Spacing tab page, such as set a left margin of 1,2 cm.

On the Numbering and outline tab, enter G1 as the top outline paragraph. Choose Roman letters in uppercase as the numbering type. Leave the numerator position empty (0 cm).

Use the context menu of the paragraph format G1 to create the paragraph format G2. Enter a description.

On the Indents and Spacing tab page, such as set a left margin of 2.4cm.

On the Numbering and outline tab, enter G1 as the top outline paragraph. This inserts G2 as a subnode of G1 in the navigation tree. Choose lowercase letters as the numbering type and a closing bracket as the right delimiter. Select the Number chaining checkbox to display the Roman numbers of the higher numbering level also on the level G2. Enter 1.2 cm as the position for the numerator (reference point: left window margin).

Task 4:

Creating character formats.

1. Create at least one character format of your choice.

   a) You can do this using the context menu of an existing character format node. Assign a two-character ID and a description and make some settings of your choice on the Standard settings and the Font tabs.

Task 5:

Activating styles.

1. Activate your style.

   a) Activate your style by clicking the Activate pushbutton, which is the third pushbutton from the left.

Continued on next page

Task 6:

Assigning styles.

1. Assign your style to the text on the page TERMS of your form.

   Test your formats.

   Optional: Format several paragraphs using the outline paragraphs G1 and G2 you created. If necessary, add a few lines of text.

   a) Go to the SAP Form Builder for your form in a second session. On the General attributes tab of the page TERMS, enter your style ZBC470_##_STYLS. In the editor of the text node that you created on this page, you should now be able to choose the new formats for your general terms and conditions from the selection list.

Task 7:

Testing your form.

1. Test your form using the program SAPBC470_DEMO.

   a) Test your form using the program SAPBC470_DEMO.

Lesson Summary

You should now be able to:

- Use styles and Smart Styles in forms
- Use the Style Builder
- Create paragraph and character formats
- Use formats in text

Unit Summary

You should now be able to:

- Use styles and Smart Styles in forms
- Use the Style Builder
- Create paragraph and character formats
- Use formats in text

Test Your Knowledge

1. You can preview the barcode in the Style Builder.
   Determine whether this statement is true or false.
   - □ True
   - □ False

2. Line spacing is adjusted automatically if you use a larger font size.
   Determine whether this statement is true or false.
   - □ True
   - □ False

3. Styles can be activated only by choosing *Style* → *Activate*.
   Determine whether this statement is true or false.
   - □ True
   - □ False

4. How do you create Smart Styles?

Answers

1. You can preview the barcode in the Style Builder.

   Answer: False

   Barcodes are not displayed in the preview of the Style Builder.

2. Line spacing is adjusted automatically if you use a larger font size.

   Answer: False

   The Line spacing is not adjusted automatically if you use a larger font size. Therefore, you must change the line spacing to avoid overlapping.

3. Styles can be activated only by choosing *Style* → *Activate*.

   Answer: False

   Styles can be activated by choosing *Style* → *Activate* or clicking the corresponding pushbutton.

4. How do you create Smart Styles?

   Answer: Smart Styles can be created by either selecting the *Style* radio button on the initial screen of the SAP Smart Forms transaction or by starting the transaction SMARTSTYLES directly.

Course Summary

You should now be able to:

- Create and change SAP Smart Forms using SAP Form Builder tools
- Create and change form styles using Style Builder
- Explain the technical implementation of forms and their integration into application programs

Appendix 1

Orientation in the SAP Menu and in the Navigation Tree

- Menu paths and transaction codes
- Legend of icons in the navigation tree

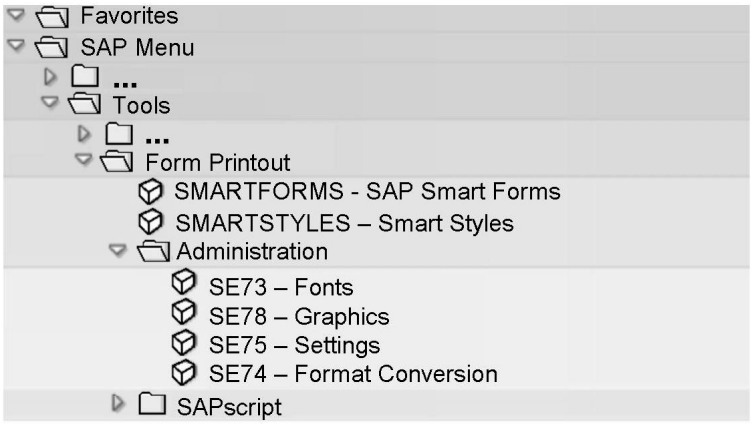


Figure 116: Menu Paths and Transaction Codes

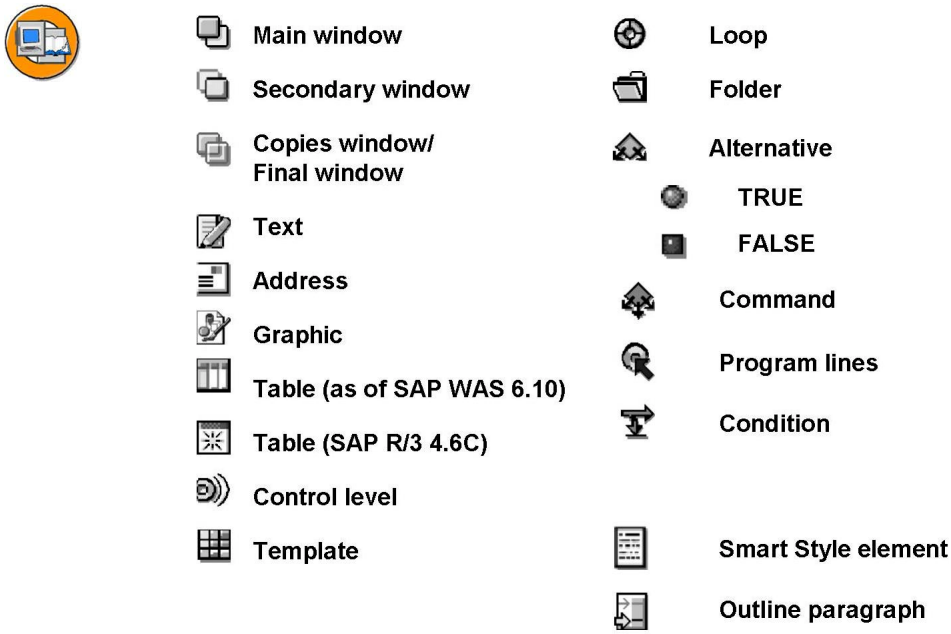


Figure 117: Legend for Icons in the Navigation Tree

Note: The appearance of the icons depends on your GUI.

Appendix 2

Changes: Customizing and Transport

- Customizing:
  - Dunning
  - Delivery note
  - Invoice
- Transport of SAP Smart Forms Objects

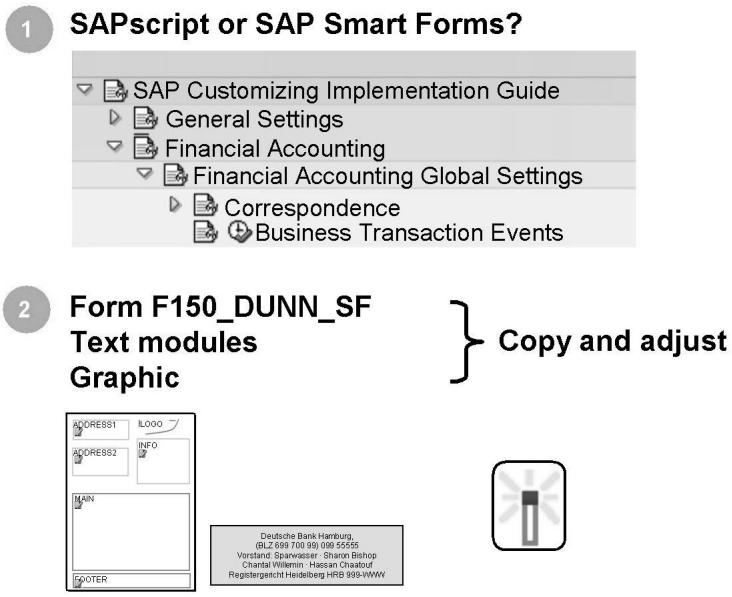


Figure 118: Change Procedure: Dunning Example I

We will use the dunning procedure as an example to illustrate all the steps that are required to ensure that a transaction executes your coding and uses your SAP Smart Forms.

1. Ensure that SAP Smart Forms are used instead of SAPscript:
   - Implementation Guide: *Financial Accounting → Financial Accounting Global Settings → Business Transaction Events*
   - Menu *Settings → P/S function modules → of an SAP application*
   - Change the function module to FI_PRINT_DUNNING_NOTICE_SMARTF for the Business Transaction Event 1720 with the application indicator FI-FI.
   - Save
2. Adjust form and texts:
   - Copy the form F150_DUNN_SF in the SAP Form Builder to your customer namespace and adjust and activate it.
   - Adjust the inserted texts, such as the address, and the company logo.

3 Which Form?

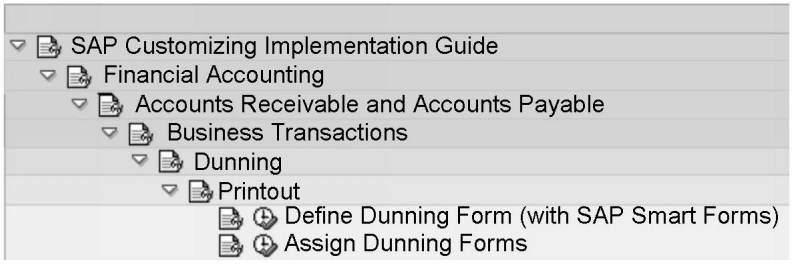


Figure 119: Change Procedure: Dunning Example II

Enter the form in Customizing:

- Implementation Guide: *Financial Accounting → Accounts Receivable and Accounts Payable → Business Transactions → Dunning → Printout → Assign Dunning Forms*
- Select the desired procedure, for example, *Four-level dunning, every two weeks*. Choose *Forms for normal or legal dunning procedure* in the tree and enter the company code.
- Enter your copy of the dunning form for the desired dunning level. Save your data and ignore the warning saying that the form does not exist or is inactive. This warning is displayed because the system only checks for SAPscript forms.

1 **Form LE_SHP_DELNOTE**
Text modules
Graphic
Program RLE_DELNOTE

} **Copy and adjust**

2 **Customizing entries**

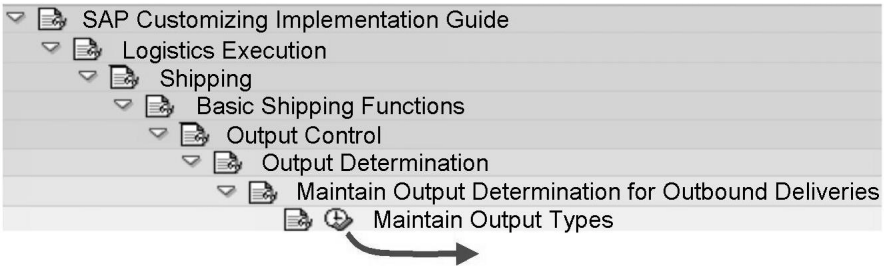


Figure 120: Change Procedure: Delivery Note Example I

The delivery note form provided by SAP is called LE_SHP_DELNOTE, the program name is RLE_DELNOTE. For company-specific adjustments, copy the form or the program to your customer namespace and then modify the copy as required.

For the delivery note, you make a setting in Customizing that specifies whether SAP Smart Forms should be used and which ones should be used.

Implementation Guide: *Logistics Execution → Shipping → Basic Shipping Functions → Output Control → Output Determination → Maintain Output Determination for Outbound Deliveries → Maintain Output Types.*

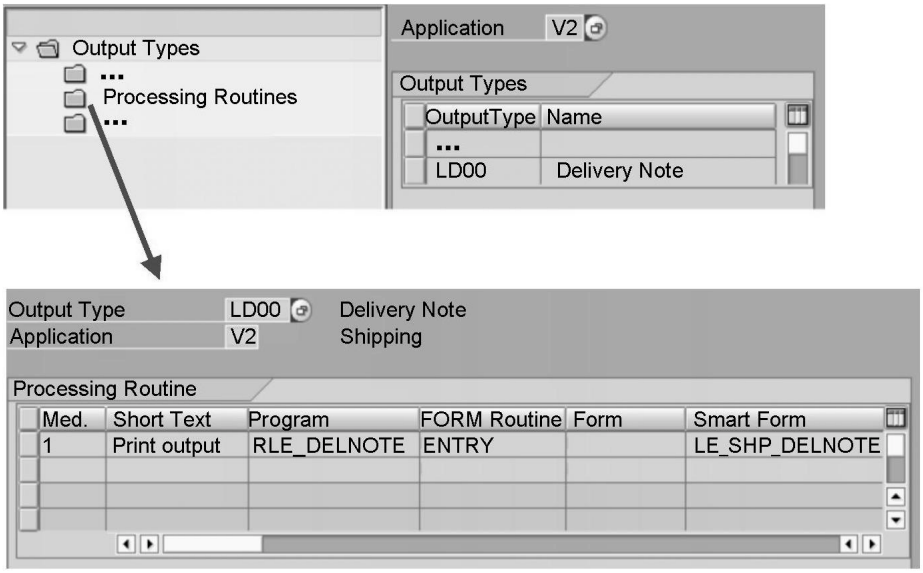


Figure 121: Change Procedure: Delivery Note Example II

Select the output type LD00 (delivery note) and double-click *Processing Routines*. On the next screen, make the following entries:

- Medium (fax or printer)
- Program: RLE_DELNOTE or the copy you made.
- FORM routine of the program that contains the call of the SAP Smart Form.
- Form: Do not enter anything here. This field is for SAPscript forms.
- Smart Form: LE_SHP_DELNOTE or the copy you made.

1 **Form LB_BIL_INVOICE**
Text modules
Graphic
Program RLB_INVOICE
} **Copy and adjust**

2 **Customizing entries:**
VOK2 –> *Output* –> *Processing Programs* -> *Billing Document*
Output Type: RD00; Application: V3

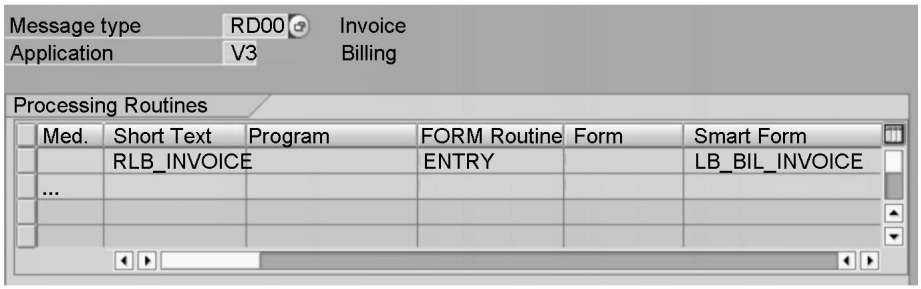


Figure 122: Change Procedure: Invoice Example

The name of the invoice form delivered by SAP is LB_BIL_INVOICE, the program is called RLB_INVOICE. For company-specific adjustments, copy the form or the program to your customer namespace and modify the copy as required.

You make a setting in Customizing for the invoice form which specifies that SAP Smart Forms should be used and which ones should be used.

To go to Customizing, start the transaction VOK2 and choose *Output → Processing Programs → Billing document*. On the dialog box that appears next, enter RD00 as the output type and V3 as the application.

On the next screen, you have to make the following entries:

- Medium
- Program: RLB_INVOICE or the copy you made.
- FORM routine of the program that contains the call of the SAP Smart Form.
- Form: Do not enter anything here. This field is for SAPscript forms.
- Smart Form: LB_BIL_INVOICE or the copy you made.

- **Forms**
- **Text modules**
- **Smart Styles**
- **Programs**

→ Transport Organizer

- **Include Texts**

→ Report RSTXRAN

- **Graphics**

→ SE78

Figure 123: Transport

Note: The following rules apply to the transport of SAP Smart Forms objects:

- Forms, text modules, Smart Styles, and programs must always be assigned to a development class. This ensures that they are automatically integrated with the Transport Organizer and are transported like all other Workbench objects.
  - When forms are transported, the function modules generated are not included in the transport. The function modules are automatically generated in the system when the form is called for the first time.
  - Language versions are automatically included in transports.
- Include texts, that is, SAPscript texts are not automatically linked to the Workbench Organizer. You must add them to a development request either manually, using the transaction SE09, or automatically using the report RSTXTRAN.
- Graphics must also be added to a development request. To do this, use the graphics administration transaction (SE78) and click the truck icon or choose *Graphic → Import*.

Appendix 3

Fonts and Bar Codes

- Adjusting Device Types
- Maintaining Font Families, System Fonts, and Printer Fonts
- Maintaining Bar Codes

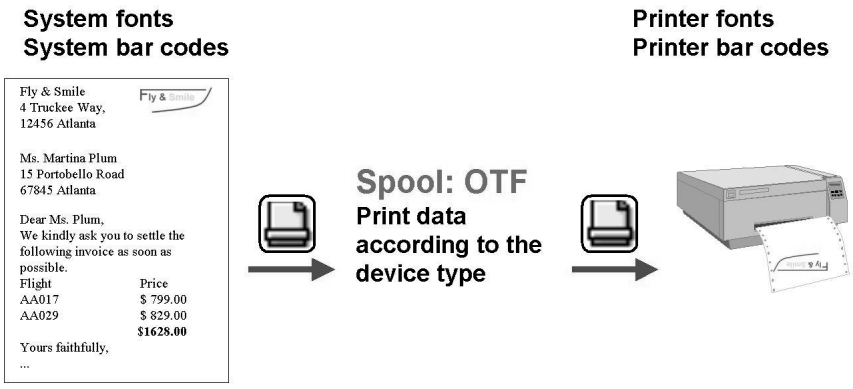


Figure 124: From Form to Device-Specific Output

Note: The following applies to both SAPscript and SAP Smart Forms.

In a form or a style, the fonts are selected independently of printers.

The users do not have to select a printer and a specific device type, until they print the form. A device type specifies which printer driver to use for formatting the output and which printer character sets are required. The temporary format for spool administration Output Text Format (OTF)

already takes the printer into account, such as page breaks. However, OTF still has the symbolic name (print control) for the control characters of the printer.

The print controls are converted to printer-specific commands by an output request.

Because a device type specifies attributes that apply for all devices of a particular model, the type can be used by several device definitions. For example, all devices with printers compatible to the type Hewlett Packard LaserJet 4 use the device type HPLJ4 in the spool system.

You can change any part of a device type according to requirements, such as to install a printer font that is not a standard SAP font.

Do not directly modify a device type delivered by SAP, make the changes in a copy instead. If you do not do this, your changes may be overwritten when an upgrade is carried out.

Figure 125: Example: Non-Printing Fonts

Problem 1: Errors have occurred in printing an invoice. Some fonts have not been correctly printed.

Cause 1: You are using a printer that is not yet supported by SAP.

Problem 2: You cannot use all the fonts that the printer can offer.

Cause 2: You have installed a new font in your printer for which the SAP standard device type, such as HPLJ4, has not been set up or you are using a printer that is not yet supported by SAP.

Solution: Generate a device type with the additional fonts/font sizes.

To set up additional fonts/font sizes:

1. In spool management (transaction SPAD), create a new device type in your namespace by copying an existing device type.
2. Create new print controls for the new device type (transaction SPAD or SE73).
3. If necessary, create a new font family.
4. Define system fonts for the font family in the required sizes and bold/italic variants.
5. For the device type, define printer fonts to correspond to the new system fonts, and assign the corresponding print controls to them.
6. Assign the new device type to your printer (the output device). For example, for the printer P280, replace the original device type HPLJ4 with the new type, ZHPLJ4.

1 **Transaction SPAD:**

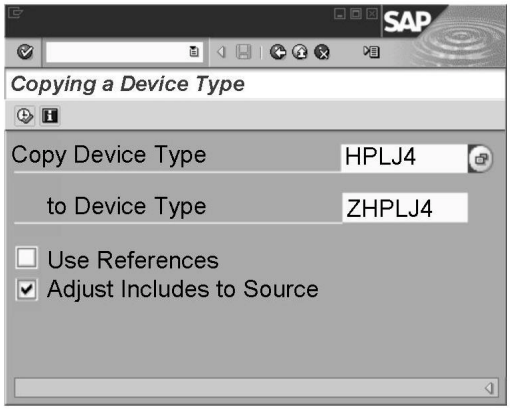


Figure 126: Copying a Device Type

To create a new device type, copy an existing one into your customer namespace. For example, copy HPLJ4 to ZHPLJ4.

In transaction SPAD (SAP Easy Access Menu: *Tools → CCMS → Spool → Spool Administration),* choose *Utilities → For Device Types → Copy Device Type.*

Use References: If you select this option, the formatting steps and font metrics contained in the source device type are not copied. In such cases, the target device type contains references to the source device type. The advantage of this is that the new device type always has the most current status when the SAP device type is updated. You should activate this

parameter only when you are sure that the source device type exists in every system where the target device type is used. If your source device type is already a customer copy, (YXXX or ZXXX), do not select this option.

When you copy a device type, the following are copied:

- The device type definition
- The print controls (you can add to the print controls at a later time)
- Format types
- Font metrics
- SAP Smart Forms and SAPscript printer fonts and printer bar codes

2

- **Process print controls for new device type**
  Add new entries for required font sizes

PrintCtrl (Example)	Font size ($^1/_{10}$ Point)	Control character sequence
SF910	180	1B28304E1B28733170313876307333623431303154
SF915	240	1B28304E1B28733170323476307333623431303154
SF920	360	1B28304E1B28733170333676307333623431303154

Figure 127: Adding Print Controls to a Device Type

Process print controls for new device type. Add new entries for required font sizes.

You need to set up a new print control SFXXX for the device type ZHLPJ4. The device type contains the printer control characters to implement the required font according to the SAP Smart Forms font. See the printer handbook to find out the control characters. Also, check which print controls are already defined for the device type. The XXX number of the SFXXX print controls is arbitrary.

To add new print controls to your copy of a device type, start the transaction SPAD and click on *Full Administration*. Choose the *Device Types* tab page. Enter the name of the device type and confirm your entry. Choose *Print Control* and switch to the change mode. Choose *Edit → Insert Line*, and enter a new SFXXX print control.

Enter the attributes and the control character sequence that you found out in the printer handbook for the print control. You can enter the command in hexadecimal format or as text.

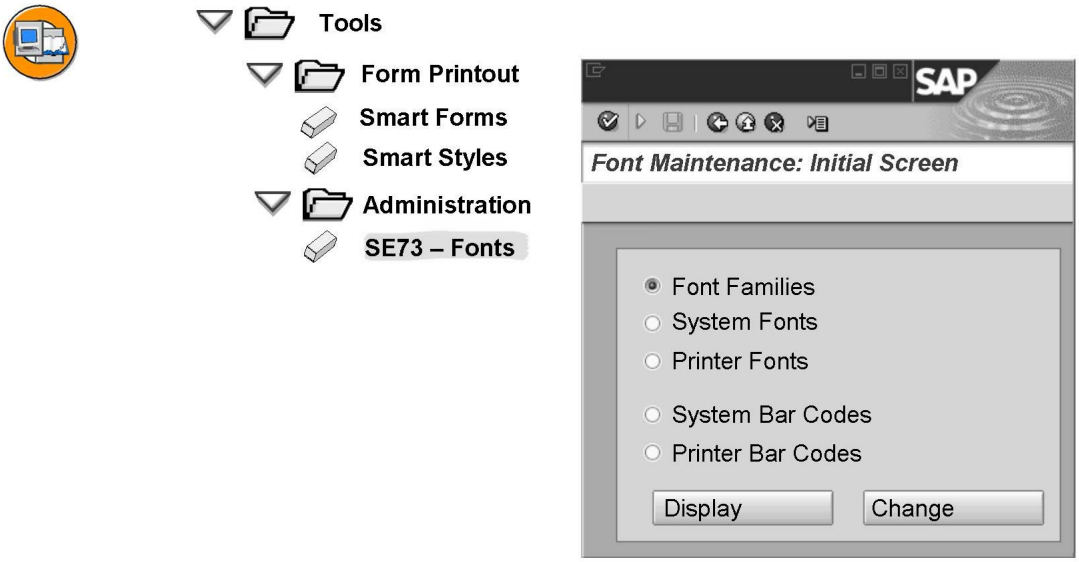


Figure 128: Font Maintenance: Initial Screen

All fonts and bar codes for SAP Smart Forms are administrated using Font Maintenance, that is the transaction SE73. In the SAP Menu, choose *Tools → Form Printout → Administration → Font*.

In **font families**, the usable font names are entered along with the attribute for proportional or not proportional.

A **system font** or SAP font is the combination of font family, font size, and bold/italic attributes. The font selections used when entering text with SAP Smart Forms always refer to system fonts.

A **printer font** is the combination of printer type, font family, font size, and bold/italic attributes. Printer fonts are fonts of the output device that are available for use with SAP Smart Forms. For printer fonts, certain metric data, such as information on the width of the letters, must be maintained along with control data for setting the fonts on the output devices.

System bar codes, such as system fonts, are independent of the device type. To use these in SAP Smart Forms, choose a character format that has the bar code attribute activated.

Printer bar codes contain the device-type-specific control character sequences that activate bar codes for suitable printers.

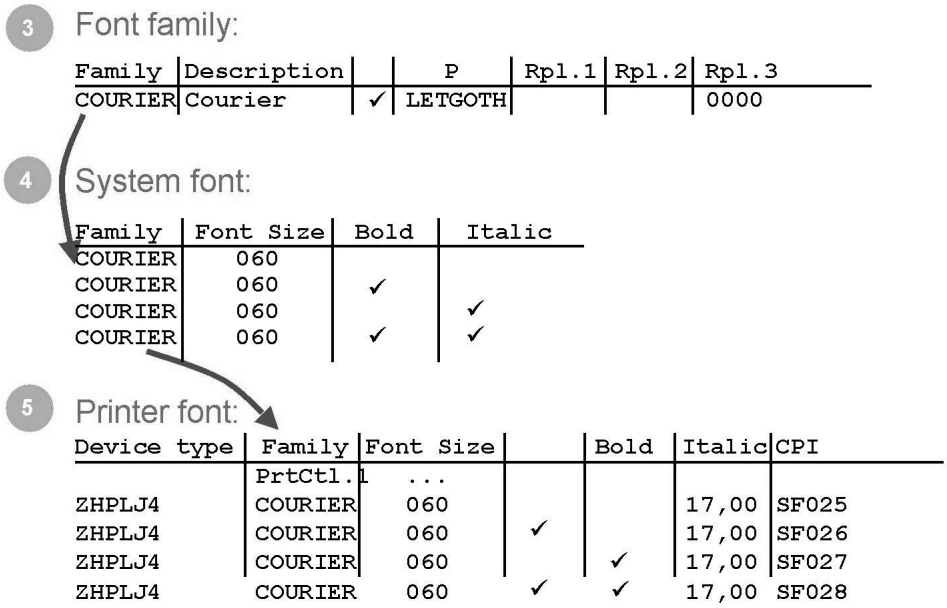


Figure 129: Creating Font Families, System, and Printer Fonts

To create a new font family, choose *Font Family* and *Change* on the initial screen of transaction SE73. Then, choose *Font Family* → *Create.* Enter a *Replacement Family* if one exists in the system. The replacement family is used in an output request if the font family is not installed on the device type you are using.

If required, enter the name of the SAP character set (code page).

A new printer font requires the following information:

- Device type
- Font family, that is, font name in SAP R/3 Enterprise such as Courier or Times
- Size in 1/10 point, such as 240. For printer drivers that support scalable fonts, 000 is entered.
- The font attributes *bold* and/or *italic.*
- Characters per inch
- Print control to use.

If you are using a proportional font, such as Times, you have to maintain an AFM file that contains width information for the individual characters. To maintain an AFM file, place the cursor on a printer font and choose *Edit Metric*. You can also copy the font metrics of an existing font: Choose *Edit* → *Copy Metric*.

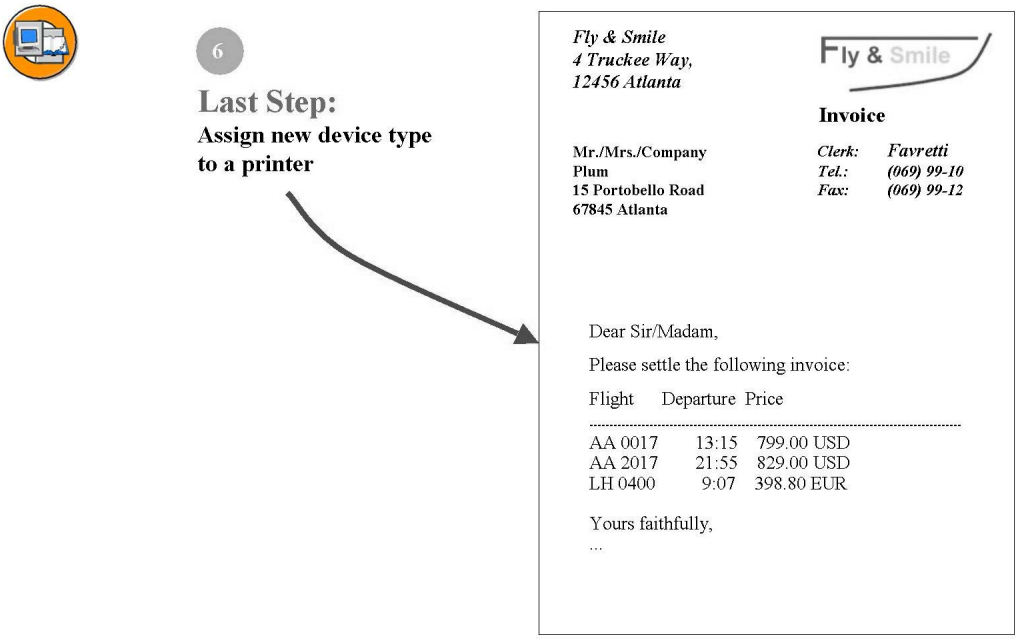


Figure 130: Result: Correct Printout of the Invoice

Finally, use spool administration, that is, the transaction SPAD to convert the device type of the output device (the printer) from the original SAP setting to your new type. For example, change the device type of the output device P280 from HPLJ4 to ZHPLJ4.

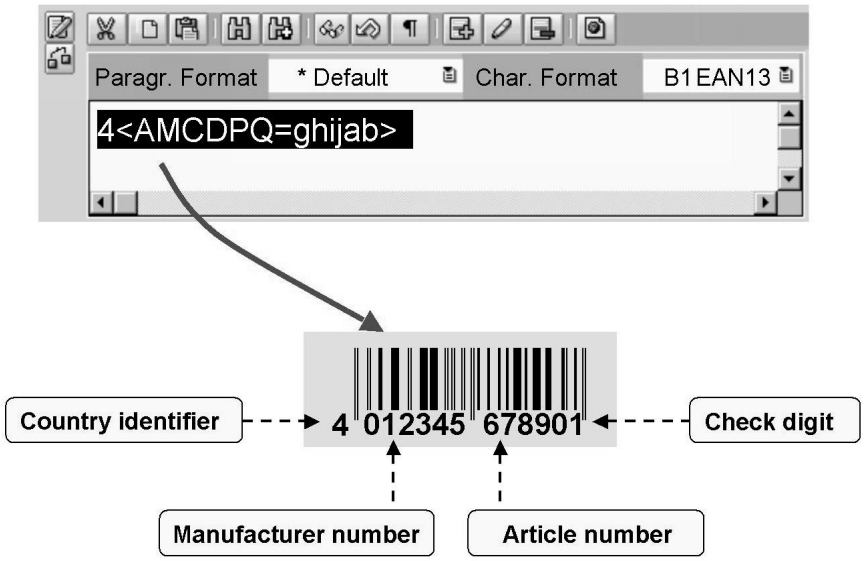


Figure 131: How a Bar Code Works?

Bar codes are used for automatic identification. Bar codes enable information to be read quickly and precisely.

A bar code consists of a series of parallel stripes and empty spaces. Predefined width patterns are used to encode data as a bar code. Scanners are used to read bar codes and a decoder interprets the width of the stripes and empty spaces.

In SAP Smart Forms, bar code data is handled in the same way as text data, (for example, the character string 4<AMCDPQ=ghijab>). This text data must be formatted using a bar code character format, which is a character format used by a system bar code. In this way, for example, the character format B1 could be defined as a bar code format used by the bar code EAN13.

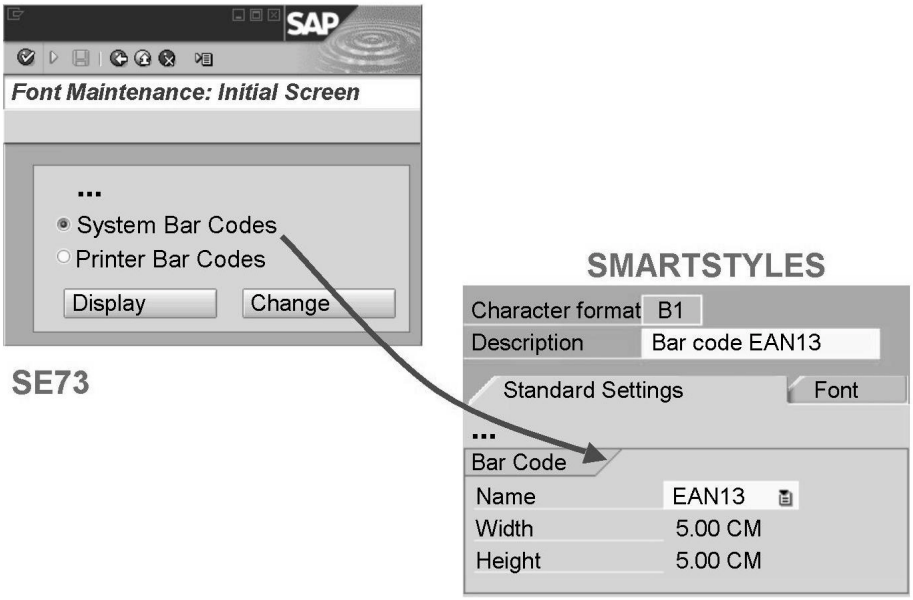


Figure 132: Maintaining and Using Bar Codes

When you print a bar code, a print control called the bar code prefix is first sent to the printer, followed by the actual bar code data, such as an eight-figure number. The printer processes a print control called a bar code suffix. The naming convention is SBPXX for prefixes and SBSXX for suffixes.

Device-type-specific control character sequences, that is, print controls are maintained with the printer bar codes.

For maintaining bar code print controls, system bar codes, and printer bar codes, the respective notes for maintaining system fonts and printer fonts apply.

- **0008928** **List of supported printers/device types**
- **0005196** **Printing bar codes with SAPscript**
- **0017054** **How to copy or change a device type**
- **0012462** **How can I define a new printer font?**
- **0317851** **Printing PDF files in 4.6C/4.6B/4.5B/4.0B**
- **0201307** **TrueType fonts for Smart Forms/SAPscript**

Figure 133: SAP Notes for Fonts and Bar Codes

0085469: SAPLPD parameters in WIN.INI & SAPLPD.INI

0119604: Bar codes Code 128, EAN-128, UCC-128 (only in English)

0045643: Bar code control sequences for JetCAPS BarSIMM

0133660: OCR and MICR SIMMs for IBM network printers

0197177: Printing 2-D bar codes using SAPscript

124987: Classes of SAP code pages (character sets)

0008928: List of supported printers/device types

0005196: Printing bar codes with SAPscript

0017054: How to copy or change a device type

0012462: How can I define a new printer font?

0317851: Printing PDF files in 4.6C/4.6B/4.5B/4.0B

0201307: TrueType fonts for Smart Forms/SAPscript

Appendix 4

Forms in Multiple Languages

Text elements:

In the application program, you set the language in which you want to print the form. To do this, use the field langu in the control structure control_parameters. This is a parameter of the generated form function module. You can specify up to three alternative languages in the fields replangu1, replangu2, and replangu3 of the control structure.

If the form or individual text elements do not exist in any of the specified languages or if you have not set the language, text elements are printed in the logon language. If the form does not exist in this language either, the original language is used.

As of SAP Web Application Server 6.20, you can select Restricted Languages in the General Attributes of a form. This stops texts in the logon language and texts in the original language being used if these texts do not exist in the language in which the form is processed.

Text modules:

Text modules are output in the language in which the form is output. As of SAP Web AS 6.10, you can set an alternative language for each text module.

Include texts:

With include texts, you can set a language explicitly in the corresponding text node of the form. If you do not specify a language, the language in which the form is output is used. If the include text does not exist in this language, the system searches for this include text in the logon language and then in the original language of the form.

Addresses:

To ensure addresses are formatted correctly for specific countries, use address nodes (with Business Address Services) or call the function module ADDRESS_INTO_PRINTFORM in program line nodes.

Format for decimal numbers and dates:

You set the format for decimal numbers and dates using the ABAP command set country <Country> in the application program. If you do not make an entry, the formats that were entered in the user master during system logon are used.

Form maintenance:

The form can only be maintained in the original language. You either log on to the system in this language or you are asked if you want to set the original language to a different language before you change the form.

Create translation:

You translate forms in the transaction SE63. You can only translate a form in the languages that you have set in the form attributes.

- Short texts of forms and text modules: Logical object SSFO
- Short texts of styles: Logical object SSST
- Long texts of forms and text modules: In SAP Web AS 6.10:*Menu Translation → Long Texts → Smart Forms.* In SAP R/3 4.6C:*Menu Translation → Long Texts → SAPscript → Smart Forms.*

Copying/transporting forms:

All languages are always copied/transported.

Appendix 5

Further Enhancements for 6.10/6.20

As of the SAP Web Application Server 6.10, the Form Builder keeps a record of the changes you have made since the form was last saved. This includes changes to the navigation tree as well as the Table Painter, Form Painter, PC editor, and the input fields of the maintenance screen. You can undo these changes in steps. It is also possible to subsequently redo the changes you have undone.

For this function, the Form Builder saves the intermediate statuses of the form. The status is only saved when you press RETURN or call an application function.

After you have called a function, Smart Forms collapses all the nodes in the navigation tree of the Form Builder. Otherwise, the Form Builder would have to remember the exact status of the navigation tree after every activity. This would lead to performance problems when working with the Form Builder.

For very large form descriptions, saving the intermediate statuses in the Form Builder runtime can have a noticeably negative effect on performance. To deactivate the function, choose: Menu *Utilities* → *Settings*, tab page *General*, radio button *Undo/Redo Form Changes*.

As of the SAP Web Application Server 6.10, there is a new window type: The copies window. A copies window is similar to a secondary window, in that you can define multiple copies windows for one form with any size and positioning on the page, and a copies window cannot contain continuous text. A special feature of the copies window is the additional feature that you can decide to make processing depend on whether the form is printed as an original or as a copy (if you are printing more than 1 copy). It is therefore possible to identify copies individually.

You have three main options for determining when the window is processed:

- On the original and the copies.
- Only on copies
- Only on the original

For the first two options, you can then further specify between *Copies Differ* and *Copies Identical*.

- For identical copies, the window content is only entered in the spool once (with references to the pages on which the window is to be printed). This can notably improve performance, particularly for graphics.
- For different copies, you can query the system variables SFSY-COPYCOUNT0 and SFSY-COPYCOUNT, for example, in conditions. SFSY-COPYCOUNT0 returns the value 0 for the original, followed by 1 for the first copy, 2 for the second copy, and so on. SFSY-COPYCOUNT is greater by 1, which means the original has the value 1, the first copy 2, the second copy 3, and so on.

As of the SAP Web Application Server 6.10, it is possible to download a form or parts of a form in XML format to a PC, and subsequently upload it into the same form, or into a different form.

To download, choose *Utilities → Download Form* or *Download Subtree*.

To upload, choose *Utilities → Upload*. If you upload a form from your PC into an existing form, the existing form is overwritten (a warning is displayed first).

If you upload the subtree of a form from your PC into a existing form, the subtree is temporarily stored on a clipboard. You can then select the appropriate position in the navigation tree, and choose *Paste* from the context menu (right mouse button).

Note: When uploading subtrees of a form, you must ensure that the styles and fields in the downloaded form also exist in the target form.

Normally, a form and its flow logic are processed by the composer. The following output formats are possible:

- OTF (Output Text Format). OTF is the standard format used in the spool for normal output. It already contains the symbolic names (print controls) for the control characters of the installed printer, and line and page breaks. OTF can be converted into PDF (Portable Document Format). See SAP Note 317851.
- XSF (XML for SAP Smart Forms) is an XML format that only contains information on the content of the processed form, but not the layout. XSF is used so that external tools can access and further process the data (using the certified XSF interface).
- As of the SAP Web AS 6.10, a CSS style sheet (Cascading Style Sheet) can be generated in addition to the XSF output. XSF and CSS are then automatically converted on the server side to HTML format, which can be displayed in every Web browser.

As of SAP Web AS 6.10, you can also call a generated function module so that the form is not processed, but the content of the interface is read. This results in the XDF format (eXtensible Data Format). This format contains the field name of every field, along with the name, type, and time stamp of the dictionary type.

In the form, you can specify which output format to use. You specify this using the form attributes on the *Output Options* tab page. In the *Output Mode* field, determine whether the output can be forwarded to the spool, or to an internal table.

You can also set the output format dynamically. To do this, make the settings in the fields of the import parameter OUTPUT_OPTIONS as specified above.

After form processing, you can query the OTF output in the internal table OTFDATA of the export parameter JOB_OUTPUT_INFO. A prerequisite for this is that the GETOTF field of the import parameter CONTROL_PARAMETERS is selected.

The XSF, CSS, and HTML data is contained in the fields of the internal table XMLOUTPUT, which is a component of the export parameter JOB_OUTPUT_INFO.

You can also have fields ready for input in text elements or text modules of your form, which are then filled by the user when displaying the HTML output in a Web browser and then evalauted by a program.

The following fields are available:

- Checkboxes
- Radio button groups: Only one radio button can be selected for each group of radio buttons.
- Text fields (maximum length 255 characters)
- Submit pushbuttons. If the user presses a key, the page is called that was either determined statically in the output options of the form or dynamically using the field XSFACTION of import parameter OUTPUT_OPTIONS. The value of the field used gives the labelling of the pushbutton.
- Reset pushbuttons. When press you this pushbutton, all the entries in the form are reset.
- Hidden fields: Invisible fields that contain data that is necessary for further processing of the form.
- List boxes (dropdown lists)
- Multiline text (text areas): For the input of longer and also multiline text (from SAP Web AS 6.10, Support Package 5).

You can also mark text as a URL by clicking on the pushbutton on the far right in the Editor.

1. If you want to output input fields or hidden fields in a text node, you must first create these as global fields. (That is, you use fields from the interface.)
2. You then integrate the fields in the text node as normal, for example, using the field list.
3. Finally you have to determine how the fields are to be output on the *Web Attributes* tab page of the text node. These settings only apply to the HTML output. To suppress your output for other formats, you can query field SFSY-XSF (value X for XSF- and HTML output).

The parameters *Field length* and *Maximum length* are only of significance for the input type *Text*.

To generate multiline text (text areas), you have to create a separate text node and choose the entry *Text area* in the field *Text area*. You can then determine the width and height of the input area.

See the online documentation for information on which name/value pairs of the HTML form are transferred when the submit pushbutton is used.

The procedure when grouping the input types (*Listbox* and *Radiobutton)* is similar - here you must, if necessary, also first define the fields and then integrate them in a text node. However, a difference is that two or more

fields within a group are merged in both of the grouping input types. The user can then select exactly one entry for each group in the Web form. You enter the name (any) of a group in the column *Group name*.

All fields that belong to a group must have the same input type (either listbox or radio button).

You have the option of marking a value as the standard selection for each group by entering a tick in the column *Std Val.*

For radio buttons, the value of the field used as a text is displayed next to the selection button. However, you can also output normal text before it or after it.

The possible values of a listbox correspond to the values of the fields used.

The SAP Web Application Server is a further development of the traditional client server architecture. By using it, you can directly process requests from the Internet/intranet that were created using a browser with a HTTP log, for example. It is also possible to send a response to the browser.

Web applications that use this technology are called Business Server Pages (BSP). A BSP application is completely integrated in the SAP Web AS, that is, data can be read from the database or SAP Smart Forms can be called.

Here is an overview of the steps for displaying a Web form with the help of a BSP:

1. The user calls the BSP application using a URL in the browser.
2. The BSP calls the event OnInitialization. It dynamically defines the layout of the HTML page.
3. The generated function module of the SAP Smart Form is called, whereby the HTML output must be activated (statically or dynamically).
4. The function module returns the HTML data in the parameter JOB_OUTPUT_INFO.
5. This must then (after a conversion) be transferred to the response object with the method SET_DATA, so that a HTTP response is generated.
6. The Web AS sends this HTTP response to the browser so that the Web form can be displayed there.

You find the most important coding for the BSP event OnInitialization here. Here, for the sake of simplicity, any business data was left out.

So that input fields can be evaluated, your SAP Smart Form requires a submit pushbutton. By mouse-clicking on it, the page that you added to parameter OUTPUT_OPTIONS-XSFACTION is called. This can be a

further BSP or for example another HTML page. Note: For each SAP Smart Form there is only one HTML form, that is, you can only specify one subsequent page.

If you want to query the user entries on a subsequent BSP, call the method GET_FORM_FIELDS of the request object in the event OnInitialization there.

For details see the online documentation for the Business Server Pages or the BSPs SF_WEBFORM_01, SF_WEBFORM_02, and SF_WEBFORM_03. For general questions on Web forms, see SAP Note 445605.

Appendix 6

Special Features for SAP R/3 4.6C

The table function has existed since SAP R/3 4.6C, but was converted for SAP Web Application Server 6.10. You can still use and edit forms with old table types, but not create them. Tables of the old type have their own icon (see Appendix).

This unit only describes the procedure for SAP R/3 4.6C:

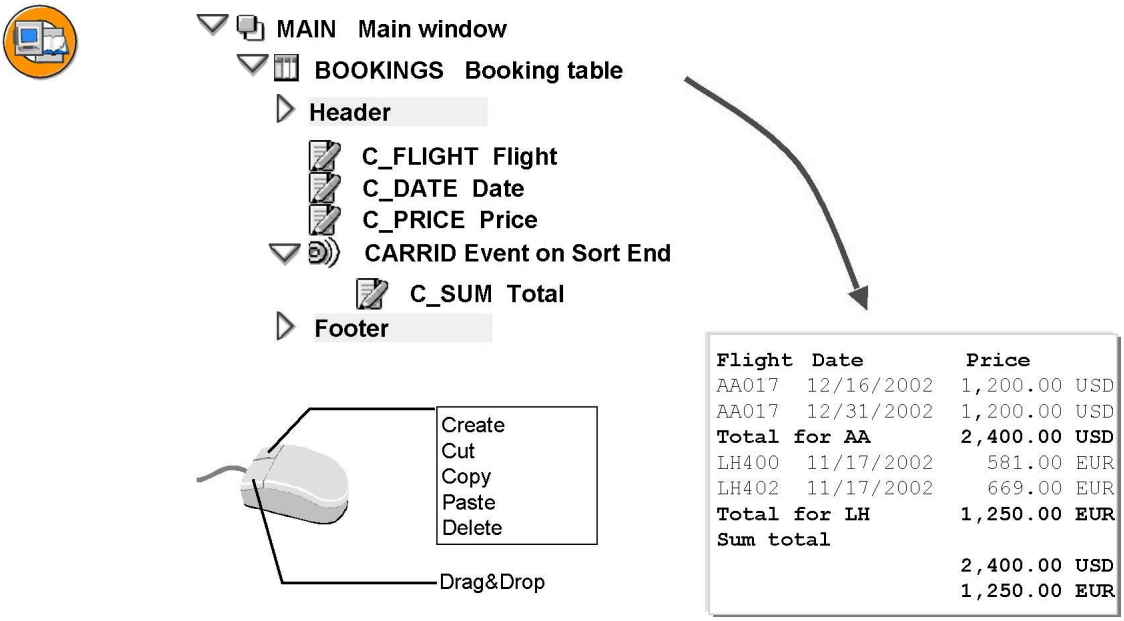


Figure 134: Tabelle: Course Overview

Forms are frequently used to output data in tables. Tables in SAP Smart Forms are subnodes of windows and are created like all other subnodes using the context menu (right mouse button) of the navigation tree.

Because the length of tables is dynamic, you should only use them in main windows since they may be truncated in secondary windows.

You can format the individual line types in the graphical Table Painter.

Tables provide functions to output headers and footers, control levels, and subtotals.

For each line type:

- Definition of number and width of columns
- Height dynamic

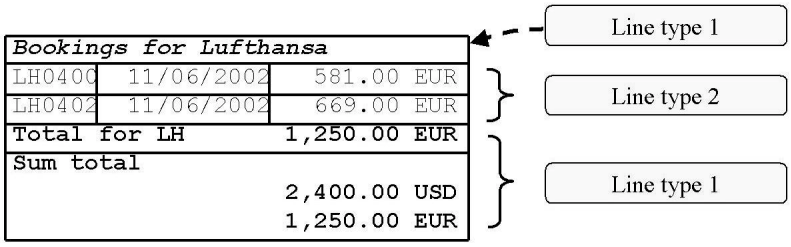


Figure 135: Line Types

Before you can fill tables with content, you must determine the table width and define line types on the Table tab of the maintenance screen. You also specify how many cells are to be in one table line and what width these cells are to have. (The height is determined automatically at runtime.) For simple applications, a single line type is sufficient. However, you can also create different types for hierarchical (multi-level) tables, for example. You do this, for example, if you want to print the different bookings for a flight in the next lines or if you want to use subtotals.

In the output options of the table contents, you specify which line types are to be used when.

You can maintain line types graphically or numerically.

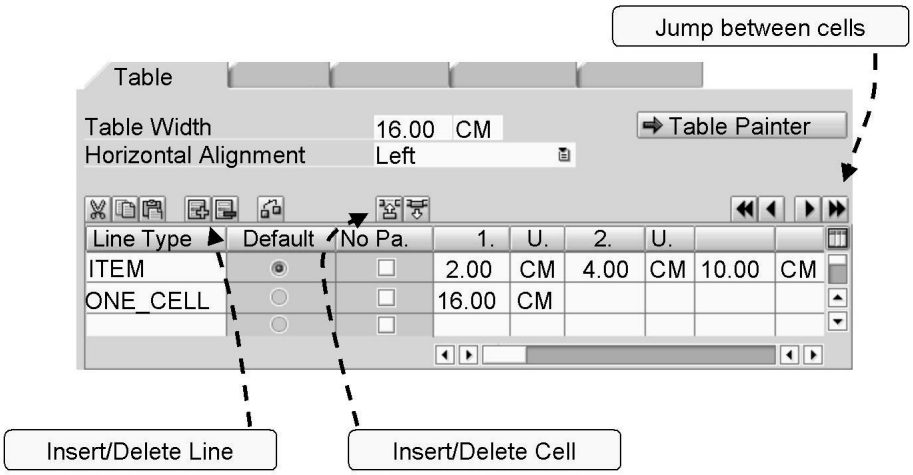


Figure 136: Defining Line Types Numerically

The following information is required for line types:

- Name
- Default type: You can only mark one type as the standard type. If no line type is assigned to a subnode of the table, the system uses the default type.
- Protection against page breaks
- Number and width of the cells

The total width of the table must be identical to the total width of all cells for each line type.

You use the *horizontal alignment* to determine how the table is to be aligned with reference to the window margin. You can choose between *Left*, *Right*, and *Centered*. If you choose *Left* or *Right*, you can optionally specify a distance from the respective window margin.

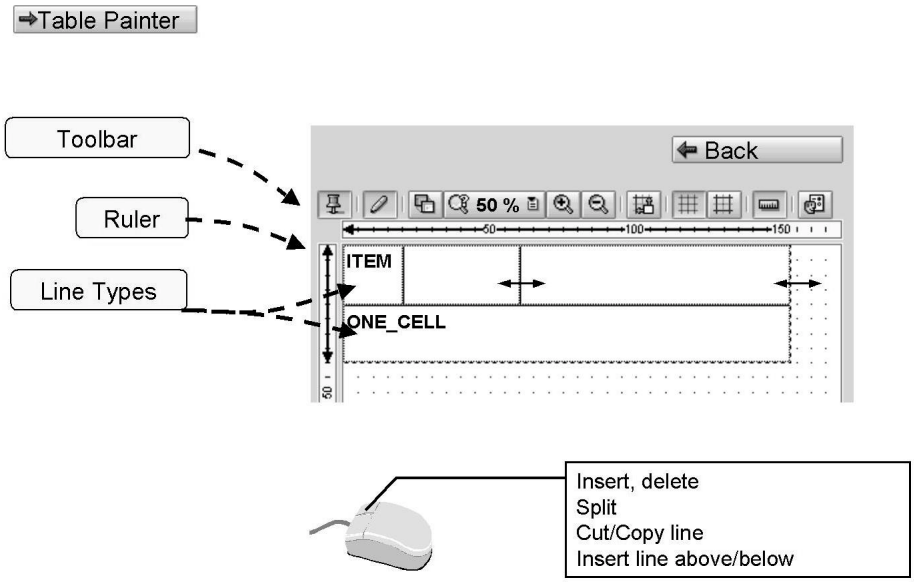


Figure 137: Line Types in the Table Painter

Instead of entering the line types in the table control, you can also define them in the graphical Table Painter. The entries you make in the Table Painter are automatically copied to the alphanumeric overview and vice versa.

You insert a new cell by vertically dragging the mouse pointer which has the shape of a pencil at the desired position while keeping the left mouse button pressed. Alternatively, you can divide the cell in which the mouse pointer is positioned using the context menu (*right mouse button → Split → Cell*).

You insert a new line by horizontally dragging the mouse pointer at the desired position. Alternatively, you can use the context menu (*right mouse button → Insert → Line*).

You change the width of a cell by placing the mouse on a cell boundary and dragging it to the desired position while keeping the left mouse button pressed. (The mouse pointer assumes the shape of a double arrow.)

You cannot set the height of individual line types because the height is determined dynamically at application program runtime, depending on the data that is output.

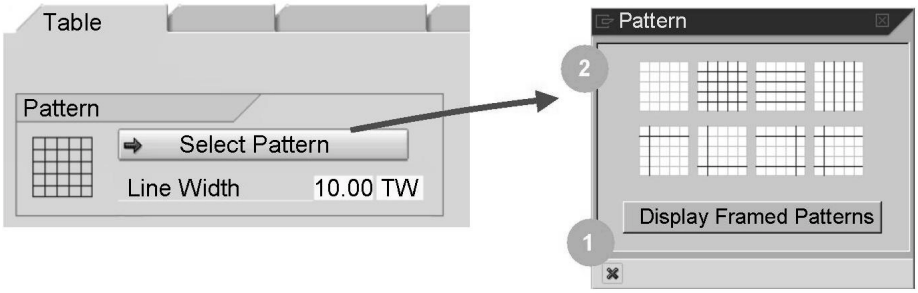


Figure 138: Table Frames

You can define gridlines for the columns and lines of a table. To do this, you select from a number of table patterns. Choose *Select Pattern* on the *Table* tab. You can also set the line width on the *Table* tab.

Decide whether you want the whole table to be framed or not (*Display Framed Pattern* pushbutton) Select the pattern you want to use by clicking it with the mouse. You can choose whether the first, the last, or all columns are to be separated by vertical gridlines and/or whether the first, the last, or all line types are to be separated by horizontal gridlines.

The selected pattern then appears on the *Table* tab.

You cannot set separate patterns for different line types because the pattern is always applied to the entire table.

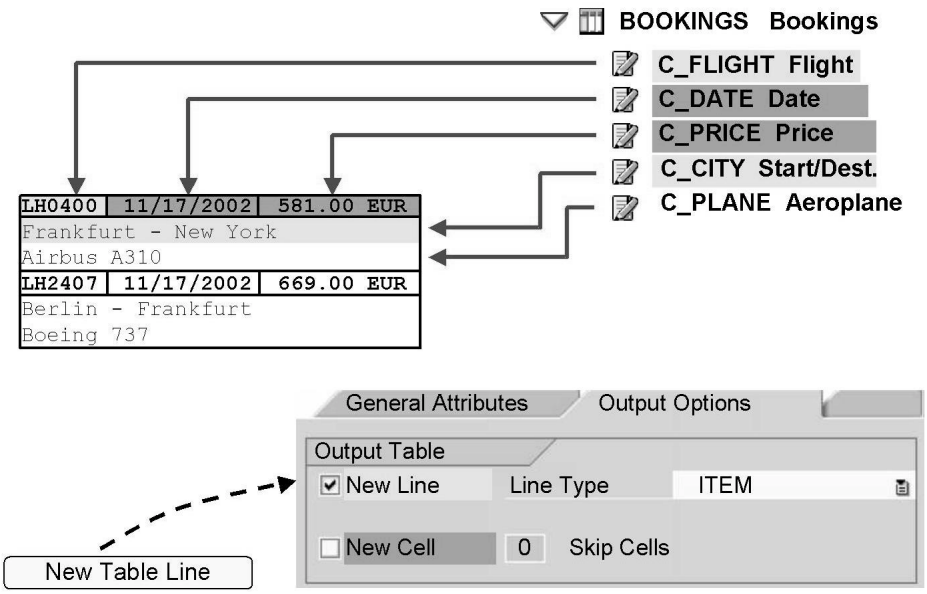


Figure 139: Contents in Table Lines

To output content in tables, you need to create text nodes, addresses, or graphics as subnodes of the table. You will notice that these table subnodes have specific input fields on the *Output Options* tab page: On this tab, you determine the text output in table lines:

- Option *New line*:
  - This option allows you to select one of the line types for this table line that you defined on the *Table* tab. If you do not select a line type, the system automatically uses the line type selected as the default type.
  - The new table line is displayed with gridlines - provided you have selected a table pattern for the table.
- If you select *New cell*, the content is output in the next cell of the line type. If the line type has no more cells to go to, an error message is issued during program execution. You can also skip several cells. If you select *New line*, the text is automatically output in the first cell of the line type selected unless you want to skip cells.

It is also possible to create more than one node in a cell.

A table line may extend over two pages at the most.

Create a folder for all nodes in a line in the navigation tree to mark them as such. See Unit 7 - *Flow Control*.

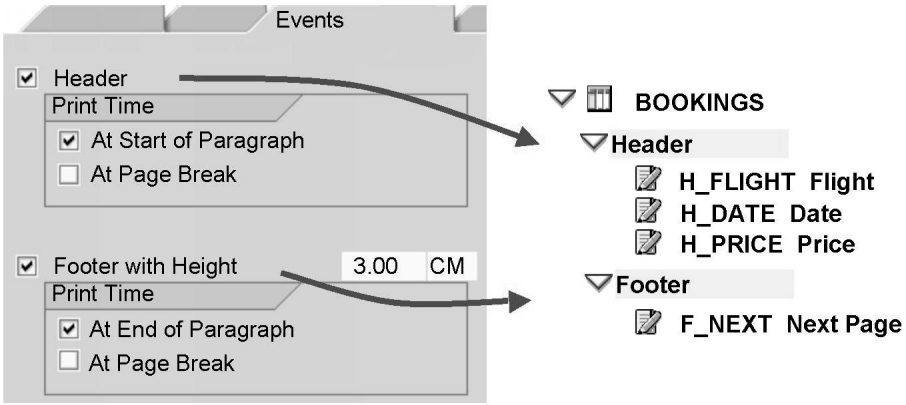


Figure 140: Headers and Footers

You can use events to control the output of headers and footers in a table. To do this, you select a header and/or footer on the *Events* tab of the table node. The corresponding event node then appears in the navigation tree.

You can output headers at the beginning of the table and/or after a page break. Similarly, footers can be output at the end of the table and/or before a page break. You must specify a height for the footer so that the form processor reserves sufficient space.

You use headers for column headings, for example. To do this, create a text node and - if required - select an appropriate line type on its *Output options* tab.

Footers are typically used to output subtotals since footers are always processed when the page break takes place. You calculate subtotals using nodes of the *Program lines* type. (See Unit 7 - *Flow Control*.)

You cannot create footers and headers directly as nodes in the navigation tree. You must always follow the procedure described for the *Events* tab.

Note that all lower-level nodes are lost if you deactivate a header or footer.

Exercise for Appendix: Tables (SAP R/3 4.6C)

Special Features for SAP R/3 4.6C

At the conclusion of these exercises, you will be able to:

- Create tables
- Create line types
- Output text in tables
- Define control levels for tables

You are an employee of the Fly & Smile travel agency. You need to enhance an existing invoice form and integrate a “real” table to output actual customer bookings.

Note: This exercise demonstrates how in a system with SAP Web Application Server 6.10 or higher, you can work with tables that are “inherited” from an SAP R/3 4.6C system.

Note:

- Copy template for the form: BC470_TABLT_OLD_TABLE
- Development class (for all exercises): ZBC470_##
- Name of the form to be created: ZBC470_##_TABLS_OLD
- Model solution: BC470_TABLS_OLD_TABLE
- Application program for testing purposes: SAPBC470_DEMO

1 **Copy template**

Copy the copy template BC470_TABLT_OLD_TABLE to ZBC470_##_TABLS_OLD.

Note: This template contains a node of the old table type (originating from a 4.6C system). You will have to add to this table. It is not possible to create a table of the old table type if you use a system with Web Application Server 6.10 or higher.

See the exercise in the unit on SAP Form Builder.

2 **Adapt table**

Note: You do not need Form Painter for tables. You can hide Form Painter to allow more space for the maintenance screen.

In the main window, you will find a table node called BOOKINGS for the bookings of each customer. The application program passes the data as an internal table (IT_BOOKINGS) to the form.

2.1 **Determine data for table**

The system should fill this form table by reading each line of the internal table IT_BOOKINGS into the work area WA_BOOKINGS. You defined WA_BOOKINGS as a global variable in the previous exercise. Note that IT_BOOKINGS contains the data for all customers selected on the selection screen. In the WHERE condition, define that the field CUSTOMID of the internal table is identical to WA_CUSTOMERS-ID. Remember that WA_CUSTOMERS contains all important information on the customer for whom the invoice is to be created. It is an interface parameter, which is filled in the application program. Perform a local check.

On the *Data* tab of the table node BOOKINGS, select the Operand (Internal table) checkbox and enter the following data into the fields on the right side: IT_BOOKINGS INTO WA_BOOKINGS. Enter the following for the WHERE condition: `CUSTOMID = WA_CUSTOMERS-ID`.

Choose the Check pushbutton on the maintenance screen.

2.2 **Determine table layout**

The table should have the following structure:

Flight	**Flight Date**	**Price**
Bookings for AA		
AA0017	16.10.2002	1,200.00 EUR
AA0017	17.10.2002	1,200.00 EUR
AA0026	18.11.2002	700.00 USD
********		

Set the table width to 15.3 cm.

You need two different line types:

- Line type POS with three cells for the heading and the items. These cells should have the following widths: 2 cm, 4.0 cm, and 9.3 cm. POS should be the default line type.
- Line type ONE_CELL with one cell for the subheadings.

Create both line types using the Table Painter. Alternatively, you can also enter the values on the maintenance screen.

Prevent page breaks from occurring within one line type.

Perform a local check.

You define the layout on the Table tab.

Click on the pushbutton Details to get to the alphanumeric table maintenance. In SAP R/3 4.6C, the Details view is the default setting. Enter the width of 15.3 cm into the topmost field of the tab.

Select the No page break checkbox for the two line types POS and ONE_CELL.

Choose the Check pushbutton on the maintenance screen.

Add a line for POS and a line for ONE_CELL to the table you see on the tab. Select the Default radio button for POS and create the three cells with a width of 2 cm, 4.0 cm, and 9.3 cm. Create a single cell for ONE_CELL that has the same width as the table (15.3 cm). To go to the Table Painter, click the *Table Painter* pushbutton to the right of the Table width input field. The Table Painter pushbutton may be hidden by Form Painter. Therefore, you may need to close Form Painter first.

3 **Fill table**

Output the ID of the airline carrier, the connection number, the flight date, and the price and currency for each booking.

3.1 Create three text elements as subnodes of the table: C_FLIGHT, C_DATE, and C_PRICE. Assign a meaningful description.

Using the context menu of the table (right mouse button), create three text elements as subnodes (not subsequent nodes), and change their names.

3.2 Select the paragraph format TB (“Cell in table body”) for all text nodes.

For the new text elements, go to the text editor. For practical reasons, do not enter the paragraph format until you enter text.

3.3 Include the carrier ID and the connection number in C_FLIGHT. Use the field list and drag the fields CARRID and CONNID of the global field WA_BOOKINGS into the text node.

Select the text node C_FLIGHT and go to the General Attributes tab. Drag the fields WA_BOOKINGS-CARRID and WA_BOOKINGS-CONNID with the mouse from the field list into the editor of the text node.

3.4 Output the flight date (WA_BOOKINGS-FLDATE) in C_DATE.

Repeat these steps for the text node C_DATE and WA_BOOKINGS-FLDATE.

3.5 Output a tabulator as well as the price and currency (WA_BOOKINGS-FORCURAM and WA_BOOKINGS-FORCURKEY) in C_PRICE. The decimal tabulator is already contained in the paragraph format TB. Define the following formatting options for WA_BOOKINGS-FORCURAM:

- Output length: 13
- Decimal places: 2

Repeat these steps for the text node C_PRICE and WA_BOOKINGS-FORCURAM and WA_BOOKINGS-FORCURKEY. Place your cursor on WA_BOOKINGS-FORCURAM. Click the Change field pushbutton, which is the third pushbutton from the right in the editor toolbar. On the subsequent dialog box, change `&WA_BOOKINGS-FORCURAM& into &WA_BOOKINGS-FORCURAM(13.2)&`.

3.6 Ensure that the texts for each new booking line are output in a separate table line. Make the correct settings for the fields New line and New cell on the Output options tab of the three text elements.

Go to the *Output Options* tab of the text node C_FLIGHT. Select the New line checkbox and select the line type POS. Go to the *Output Options* tab of the text node C_DATE. Select New Cell. Also, select New Cell for the text node C_PRICE.

4 **Define a table pattern**

On the Table tab, click the **Select pattern** pushbutton. In the dialog box that appears, select whether you want an external box. Then, choose a pattern.

5 **Define column headings**

The table is to have the column headings “Flight”, “Flight date”, and “Price”.

5.1 Select the Header event for the table.

Select the Header checkbox on the Events tab of the table BOOKINGS.

5.2 Create three text elements as subnodes of the header: H_FLIGHT, H_DATE, and H_PRICE.

Hint: The easiest way to do this is to copy the existing elements C_FLIGHT, C_DATE, and C_PRICE to the header (Ctrl and mouse). This ensures that the descriptions are correct and that the headings are output in the correct cells.

Copy the existing text elements C_FLIGHT, C_DATE; and C_PRICE into the header area of the table (left mouse button while holding down the CTRL button). Change the names. If you are creating new nodes instead of using this procedure, do not forget to choose the line type POS in the Output Options.

5.3 Enter “Flight” in the text node H_FLIGHT, “Flight date” in the text node H_DATE, and “Price” in the text node H_PRICE. Choose the paragraph format TH (“Cell in table header”) for all three text nodes.

Enter the texts in the editor (General Attributes tab page) and choose the paragraph format TH.

6 **Define control levels**

The table is to have control levels for the airline carriers. Select the beginning and the end of the bookings for each carrier.

6.1 Make the necessary entries on the Data tab to ensure that the table is sorted by the carrier (CARRID) and event nodes are created for the beginning and the end of the control levels in the navigation tree.

On the *Data* tab page of the table BOOKINGS, enter CARRID as the field name for the sort criteria and select the Event on Sort Begin and Event on Sort End checkboxes. This automatically inserts the nodes for the control levels in the navigation tree.

6.2 **Insert subheading**

To output a subheading for each carrier, insert a text node called CARRIER_HEADING below the beginning of the control level.

Enter the text “Bookings for <carrier ID>”. Use the field list to insert the appropriate ID (WA_BOOKINGS-CARRID). Choose the Italic character format and the TB paragraph format for the subheading.

Make sure that the line type ONE_CELL is used for the subheading.

Use the context menu (right mouse button) of the event node “CARRID Event on Sort Begin” to create the text node CARRIER_HEADING.

Enter your text in the editor of the text node CARRIER_HEADING (General attributes tab page).

Go to the Output Options tab of the text node CARRIER_HEADING. Select New Line and choose the line type ONE_CELL.

6.3 **Optional: Insert asterisk chain (dummy subtotals)**

Output a string of asterisks after each airline carrier.

Insert the text node called SUBTOTAL as a subnode of the control level end.

Enter a few asterisks.

Make sure that the line type ONE_CELL is used for the asterisks.

You have already created the event node "CARRID Event on Sort End" in exercise above. Use the context menu (right mouse button) of that event node to create the text node called SUBTOTAL.

Enter the asterisks in the editor (General Attributes tab) of the text node SUBTOTAL.

Go to the *Output Options* tab of the text node SUBTOTAL. Select New Line and choose the line type ONE_CELL.

7. Activate your form and test it using the program SAPBC470_DEMO.

Solution for Appendix: Tables (SAP R/3 4.6C)

1 Copy template

See the exercise in Unit 3.

2 Adapt table

2.1 On the Data tab of the table node BOOKINGS, select the Operand (Internal table) checkbox and enter the following data into the fields on the right side: IT_BOOKINGS INTO WA_BOOKINGS. Enter the following for the WHERE condition: CUSTOMID = WA_CUSTOMERS-ID.

Choose the *Check* pushbutton on the maintenance screen.

2.2 Determine table layout

You define the layout on the *Table* tab.

2.2.1 Click on the pushbutton Details in order to get to the alphanumeric table maintenance. (In SAP R/3 4.6C, the Details view is the default setting).

Enter the width of 15.3 cm into the topmost field of the tab.

2.2.2 Add a line for POS and a line for ONE_CELL to the table you see on the tab.

Select the Default radio button for POS and create the three cells with a width of 2 cm, 4.0 cm, and 9.3 cm.

Create a single cell for ONE_CELL that has the same width as the table (15.3 cm).

To go to the Table Painter, click the *Table Painter* pushbutton to the right of the *Table width* input field. Since the *Table Painter* pushbutton may be hidden by the Form Painter you may need to close the Form Painter first.

2.2.4 Choose the *Check* pushbutton on the maintenance screen.

3 Fill table

3.1 Using the context menu of the table (right mouse button), create three text elements as subnodes (not subsequent nodes), and change their names.

3.2 For the new text elements, go to the text editor. For practical reasons, do not enter the paragraph format until you enter text.

3.3 Select the text node C_FLIGHT and go to the *General Attributes* tab. Drag the fields WA_BOOKINGS-CARRID and WA_BOOKINGS-CONNID with the mouse from the field list into the editor of the text node.

3.4 Repeat these steps for the text node C_DATE and WA_BOOKINGS-FLDATE.

3.5 Repeat these steps for the text node C_PRICE and WA_BOOKINGS-FORCURAM and WA_BOOKINGS-FORCURKEY.

Place your cursor on WA_BOOKINGS-FORCURAM. Click the *Change field* pushbutton (the third pushbutton from the right in the editor toolbar). On the subsequent dialog box, change &WA_BOOKINGS-FORCURAM;& into &WA_BOOKINGS-FORCURAM;(13.2)&.

3.6 Go to the *Output Options* tab of the text node C_FLIGHT. Select the *New line* checkbox and select the line type POS.

Go to the *Output Options* tab of the text node C_DATE. Select *New Cell*. Also select *New Cell* for the text node C_PRICE.

4 Define a table pattern

On the *Table* tab, click the *Select pattern* pushbutton. In the dialog box that appears, select whether you want an external box. Then choose a pattern.

5 Define column headings

5.1 Select the *Header* checkbox on the *Events* tab of the table BOOKINGS.

5.2 Copy the existing text elements C_FLIGHT, C_DATE; and C_PRICE into the header area of the table (left mouse button while holding down the CTRL button). Change the names. (If you are creating new nodes instead of using this procedure, do not forget to choose the line type POS in the *Output Options*).

5.3 Enter the texts in the editor (*General Attributes* tab page), and choose the paragraph format TH.

6 Define control levels

6.1 On the *Data* tab page of the table BOOKINGS, enter CARRID as the field name for the sort criteria and select the *Event on Sort Begin* and *Event on Sort End* checkboxes. This automatically inserts the nodes for the control levels in the navigation tree.

6.2 Insert subheading

6.2.1 Use the context menu (right mouse button) of the event node "CARRID Event on Sort Begin" to create the text node CARRIER_HEADING.

6.2.2 Enter your text in the editor of the text node CARRIER_HEADING (*General attributes* tab page).

6.2.3 Go to the *Output Options* tab of the text node CARRIER_HEADING.

Select *New Line* and choose the line type ONE_CELL.

6.3 Optional: Insert asterisk chain (dummy subtotals)

6.3.1 You have already created the event node "CARRID Event on Sort End" in exercise 6-1. Use the context menu (right mouse button) of that event node to create a text node called SUBTOTAL.

6.3.2 Enter the asterisks in the editor (*General Attributes* tab) of the text node SUBTOTAL.

6.3.3 Go to the *Output Options* tab of the text node SUBTOTAL. Select *New Line* and choose the line type ONE_CELL.

You may sometimes need to use a condition on the first page to evaluate the total number of pages of the form being processed (for example, for bar codes in machines for filling envelopes) or want to display a total on a page although this total is not calculated until the form is processed. The variable SFSY-FORMPAGES does indeed already exist, however, its value is not determined until the end of form processing, and it is subsequently inserted by the form processor in a second processing run. Fields for which a value has not yet been determined cannot be evaluated in a normal window.

As of SAP Web AS 6.10, there is a new window type, *Final Window*. Windows of this type are only processed in a second repetition, that is, at a time when the values of all the fields are available, that are not filled until the last page.

Before SAP R/3 4.6C, there was no solution for the problem before Support Package 12. However, as of Support Package 12, the following procedure can be used to create windows that are first processed by the form processor in a second run:

- Create the secondary window that is to be processed in the second run.
- On the first form page that is processed, create a secondary window as the top subnode that can only contain a node of the type *Program lines*. In this node, a form routine is called for every post processsing window and the name of the window is passed to it (see coding in the graphic above).

For more information, see SAP Note 359009. Note: Forms that use the described technology must be changed for SAP Web AS 6.10 (remove coding and mark secondary window as final window).

Appendix 7

Using SAPscript Objects

SAP delivers SAP Smart Forms for important business processes. You can also convert your existing (active) SAPscript forms into SAP Smart Forms. There are two tools that support this conversion process. Note, however, that SAPscript forms are integrated into programs in an entirely different way than SAP Smart Forms and that you might therefore need to manually adjust the form and the print program which may be very time-consuming if the form you want to convert is very complex.

You can convert individual forms on the initial screen of the SAP Smart Forms transaction. Enter the name of the SAP Smart Form to be created in the *Form* field and choose *Utilities → Migrate SAPscript form*. Then choose the SAPscript form and the language you want to migrate. The program first tries to find the SAPscript form in the current client; if the form does not exist there, the system looks in client 000. If you choose *Enter*, the form is migrated. The system then takes you automatically to the Form Builder where you can make any necessary adjustments and save and activate the new form.

When you migrate a form, general form settings and layout information including pages, windows, and their attributes and positions on the pages are copied unchanged. However, the definition of the paragraph and character formats is lost (since these are not saved in the form itself but centrally in styles in SAP Smart Forms). As far as text elements are concerned, their character formats are preserved but not their paragraph formats or SAPscript commands. Note that in SAP Smart Forms, in contrast to SAPscript, all fields must be defined explicitly.

SAPscript texts can be used directly in SAP Smart Forms. Note, however, that SAPscript commands are not executed within these texts and that SAPscript styles are generally ignored. Besides, all fields of the texts in the SAP Smart Form must be defined - otherwise, the generated function

module terminates with an error message. Note that SAPscript texts are client-specific, while SAP Smart Forms are not. This is why you have to make sure that the texts are available in all production clients.

Print programs must be adjusted manually. In particular, you must replace the function modules OPEN_FORM, CLOSE_FORM, START_FORM and END_FORM that were needed previously with the generated function module with the interface. The processing of the text elements with the function module WRITE_FORM, including the setting/deleting of headings, must be completely moved from the print program to the SAP Smart Form.

SAPscript styles can be easily converted into Smart Styles. To do this, go to the initial screen of the Smart Styles maintenance transaction (SMARTSTYLES) and choose *Smart Styles → Convert SAPscript style*.

Index

Feedback

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