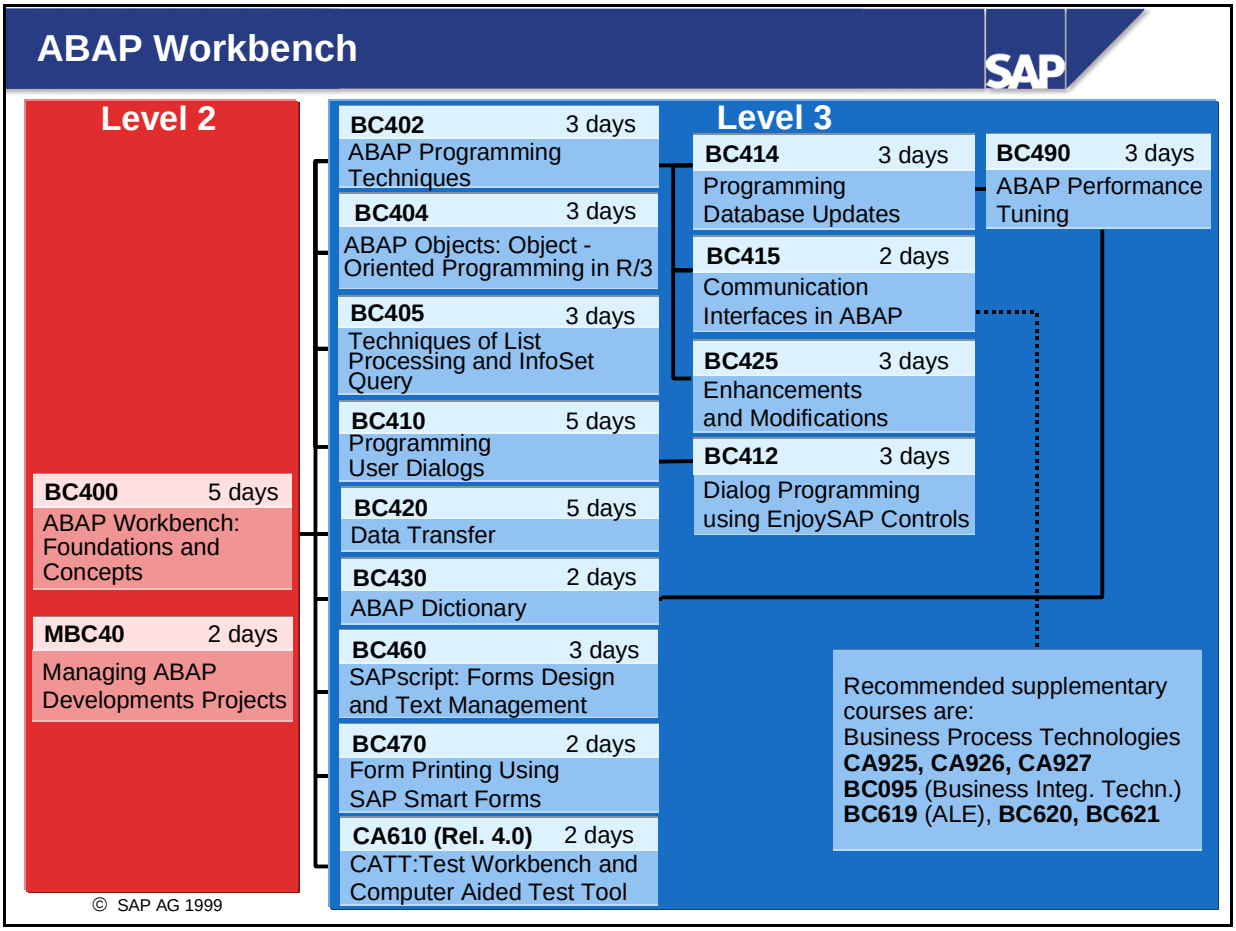
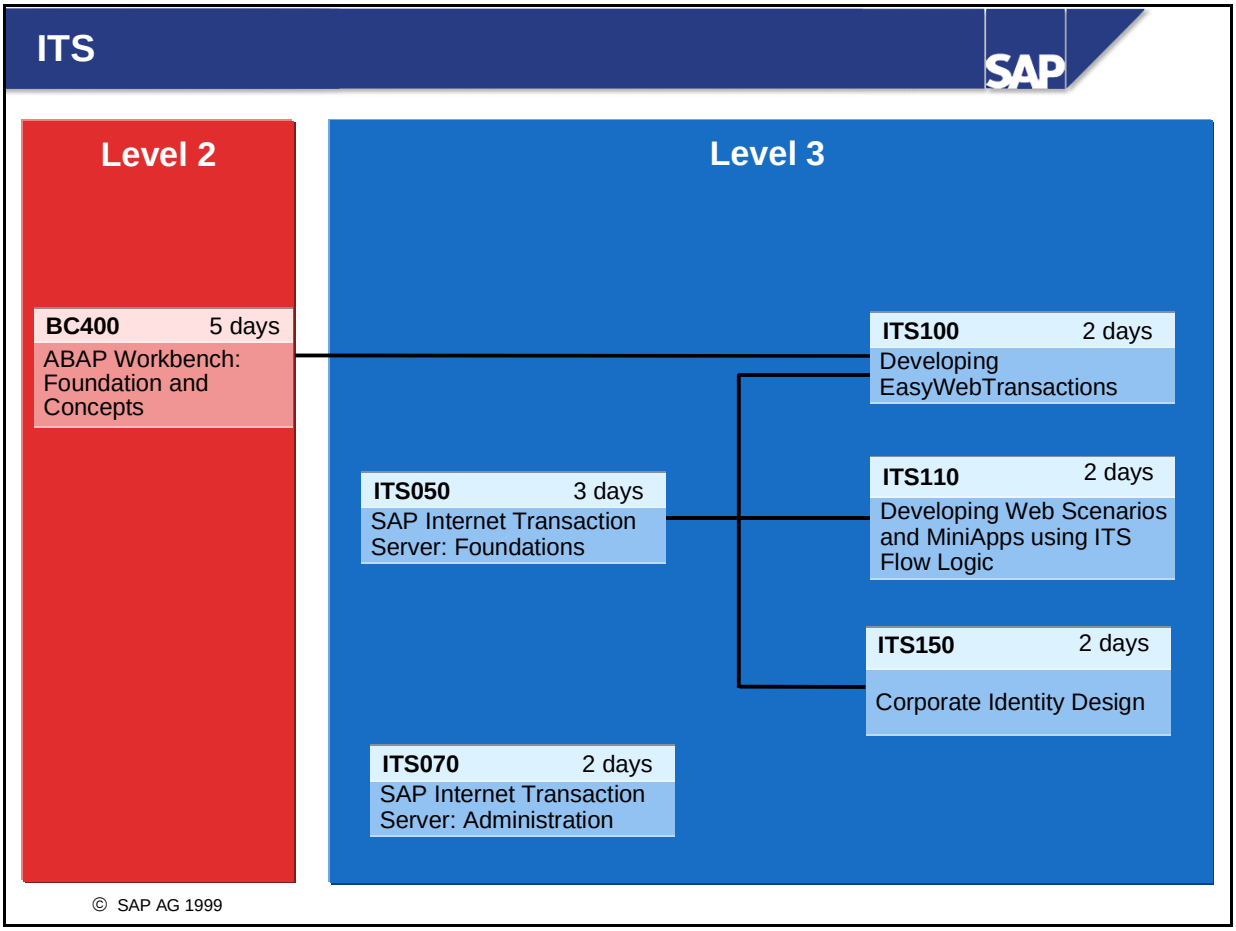
- R/3 System
- Release 4.6C
- January 2001
- 5004 2470

Copyright

- Design: SAP Communications Media


ABAP Workbench
SAP
Level 2
BC400 5 days
ABAP Workbench: Foundations and Concepts
MBC40 2 days
Managing ABAP Developments Projects
BC402 3 days
ABAP Programming Techniques
BC404 3 days
ABAP Objects: Object - Oriented Programming in R/3
BC405 3 days
Techniques of List Processing and InfoSet Query
BC410 5 days
Programming User Dialogs
BC420 5 days
Data Transfer
BC430 2 days
ABAP Dictionary
BC460 3 days
SAPscript: Forms Design and Text Management
BC470 2 days
Form Printing Using SAP Smart Forms
CA610 (Rel. 4.0) 2 days
CATT:Test Workbench and Computer Aided Test Tool
Level 3
BC414 3 days
Programming Database Updates
BC415 2 days
Communication Interfaces in ABAP
BC425 3 days
Enhancements and Modifications
BC412 3 days
Dialog Programming using EnjoySAP Controls
BC490 3 days
ABAP Performance Tuning
Recommended supplementary courses are:
Business Process Technologies
CA925, CA926, CA927
BC095 (Business Integ. Techn.)
BC619 (ALE), BC620, BC621
© SAP AG 1999


ITS
SAP
Level 2
Level 3
BC400 5 days
ABAP Workbench: Foundation and Concepts
ITS100 2 days
Developing EasyWebTransactions
ITS050 3 days
SAP Internet Transaction Server: Foundations
ITS110 2 days
Developing Web Scenarios and MiniApps using ITS Flow Logic
ITS150 2 days
Corporate Identity Design
ITS070 2 days
SAP Internet Transaction Server: Administration
© SAP AG 1999

Course Prerequisites

- **Required:**
  - **BC400 (ABAP Workbench: Concepts and Tools)**
  - **ABAP programming knowledge**
- **Recommended:**
  - **Standard Level 2 application training courses are an advantage**

- Hint: This course teaches you the basic principles of developing forms with SAP Smart Forms, but no application-specific knowledge.

Target Group

- **Participants:**
  - **Project team members, developers and consultants responsible for form printing**
- **Duration: 2 days**

Notes to the user

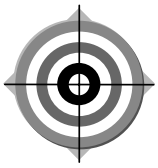
- The training materials are not teach-yourself programs. They complement the course instructor's explanations. On the sheets, there is space for you to write down additional information.

Course Overview

Contents:

- Course Goal
- Course Objectives
- Course Content
- Course Overview Diagram
- Main Business Scenario

Course Goal



- **Become familiar with SAP Smart Forms and learn how the graphical tools interact with each other**
- **Create and maintain forms**
- **Learn how forms are integrated into application programs**
- **Use Smart Styles**

Course Objectives

At the conclusion of this course, you will be able to:

- **Create and change SAP Smart Forms using the SAP Form Builder and its tools**
- **Create and change styles for forms using the Style Builder**
- **Explain how forms are implemented technically and how they are integrated into application programs**

Course Content

Preface

Unit 1	**Course Overview**	Unit 6	**Tables and Templates**
Unit 2	**SAP Smart Forms: Overview**	Unit 7	**Flow Control**
Unit 3	**First Steps with the SAP Form Builder**	Unit 8	**Integration into Application Programs**
Unit 4	**Texts, Addresses, and Graphics**	Unit 9	**Smart Styles**
Unit 5	**Data in Forms**	Unit 10	**Fonts and Bar Codes**

Appendix

Course Overview Diagram

BC470 Form Printing with SAP Smart Forms

1 **Course Overview**

2 **SAP Smart Forms: Overview**

3 **First Steps with the SAP Form Builder**

4 **Texts, Addresses, and Graphics**

&WA&

5 **Data in Forms**

6 **Tables and Templates**

7 **Flow Control**

8 **Integration into Application Programs**

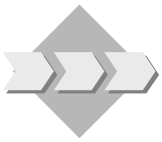
abcd

9 **Smart Styles**

10 **Fonts and Bar Codes**

Appendix

Main Business Scenario



- **You are an employee with the Fly & Smile travel agency. You are responsible for sending invoices about booked flights to customers.**
- **The invoice form is set up step by step.**
- **The business data is taken from the tables of the flight data model which is used in all BC-4XX training courses.**

Demos, Copy Templates, and Solutions

Development class BC470 with the following naming conventions:

- **Programs:**
  - **Demos: SAPBC470_####D...**
  - **Copy templates: SAPBC470_####T...**
  - **Solutions: SAPBC470_####S...**
- **Forms:**
  - **Demos: BC470_####D...**
  - **Copy templates: BC470_####T...**
  - **Solutions BC470_####S...**

where #### = unit identifier

- The exercises are built on each other. A sample solution is available for each step completed which you can use as a copy template for the next step.
- There may not be sufficient time to work through all the exercises during the course.
- Each unit has a four-character identifier:

–--	1.	Course Overview
–--	2.	SAP Smart Forms: Overview
STEP	3.	First Steps with the SAP Form Builder
TEXT	4.	Texts, Addresses, and Graphics
DATA	5.	Data in Forms
TABL	6.	Tables and Templates
FLOW	7.	Flow Control
PROG	8.	Integration into Application Programs
STYL	9.	Smart Styles
–--	10.	Fonts and Bar Codes

SAP Smart Forms: Overview

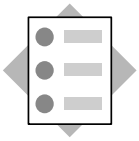
BC470 Form Printing with SAP Smart Forms

SAP Smart Forms: Contents

Contents:

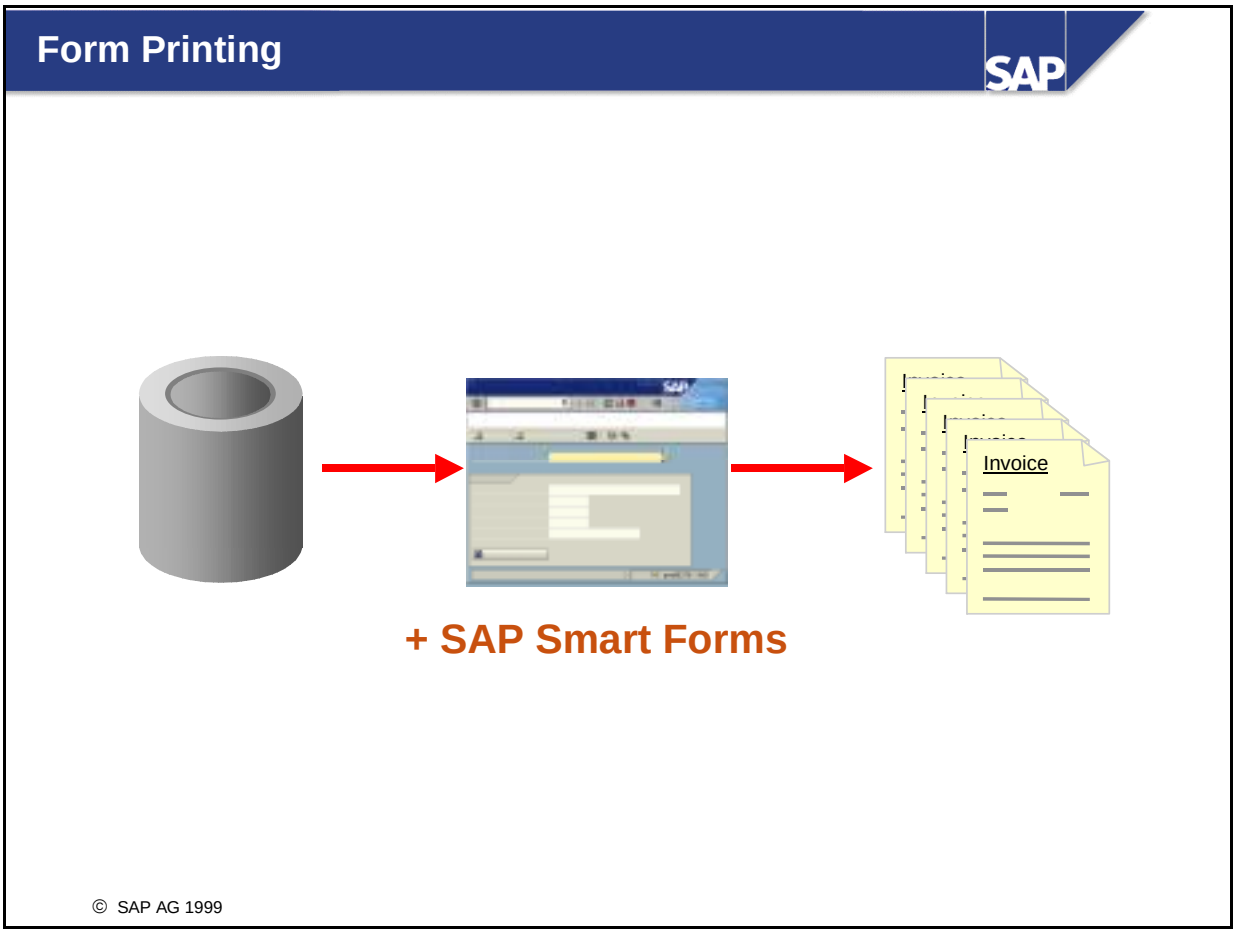
- **Overview of where SAP Smart Forms can be used and how they function**

SAP Smart Forms: Overview - Unit Objectives

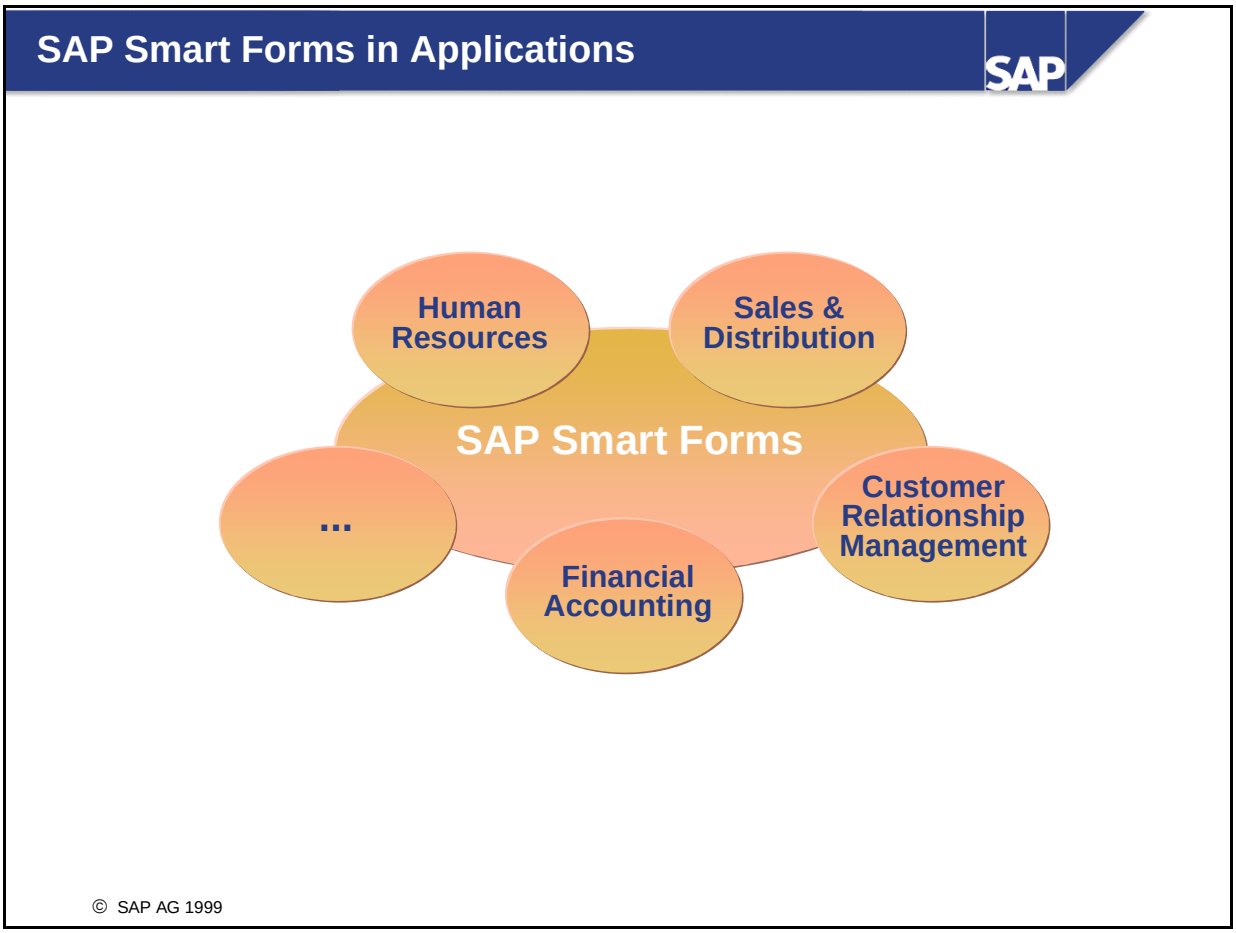


At the conclusion of this unit, you will be able to:

- **List the areas in which form printing with SAP Smart Forms can be used**
- **Explain how SAP Smart Forms function and how they are integrated into application programs**



- Every company regularly needs to print large numbers of documents with a consistent design, such as invoices or delivery notes. To do this, they must use their business application software. Documents can be output to a printer, a fax device or as e-mails.
- Starting with R/3 Release 4.6C, SAP provides a new tool for form processing - **SAP Smart Forms**. This tool includes utilities for designing forms and for defining the interface to the application programs that use forms for data output.



- In addition to the tool itself, the e-business platform mySAP.com comes with a number of forms for central business processes. These include forms for Customer Relationship Management (CRM) as well as for the R/3 application components Sales & Distribution (SD), Financial Accounting (FI), and Human Resources (HR). The R/3 System also contains several country-specific forms.
- Other SAP Smart Forms will be added over time in support packages and future releases.
- Some SAP Smart Forms functions require at least Support Package 1 in R/3 for Release 4.6C. However, we generally recommend that you always install the latest Support Package.
- Hint: This training course does **not** teach you any **application-specific knowledge**. It is the course's intent to teach you the basics of form development which will enable you to make changes to all forms and/or application programs once the course is completed.

Benefits of SAP Smart Forms

+ **Minimum time required for creating and maintaining forms**
+ **Adjusting forms without programming knowledge thanks to complete graphical user interface**

+ **Integration into SAP products**
+ **High performance when printing in large quantities**

+ **Link to 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. Besides, documents can be automatically created in the back-ground, which, for example, is useful in the 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, however, you need the graphical user interface SAP GUI for Windows.
- SAP Smart Forms provide multiple language support. You need to define the layout of a form only once and can then translate the texts using the translation tools. This ensures that your documents have a consistent design on an international level.

SAP Smart Forms and SAPscript

Minimum time required for adjusting forms

Consistent use of graphical tools

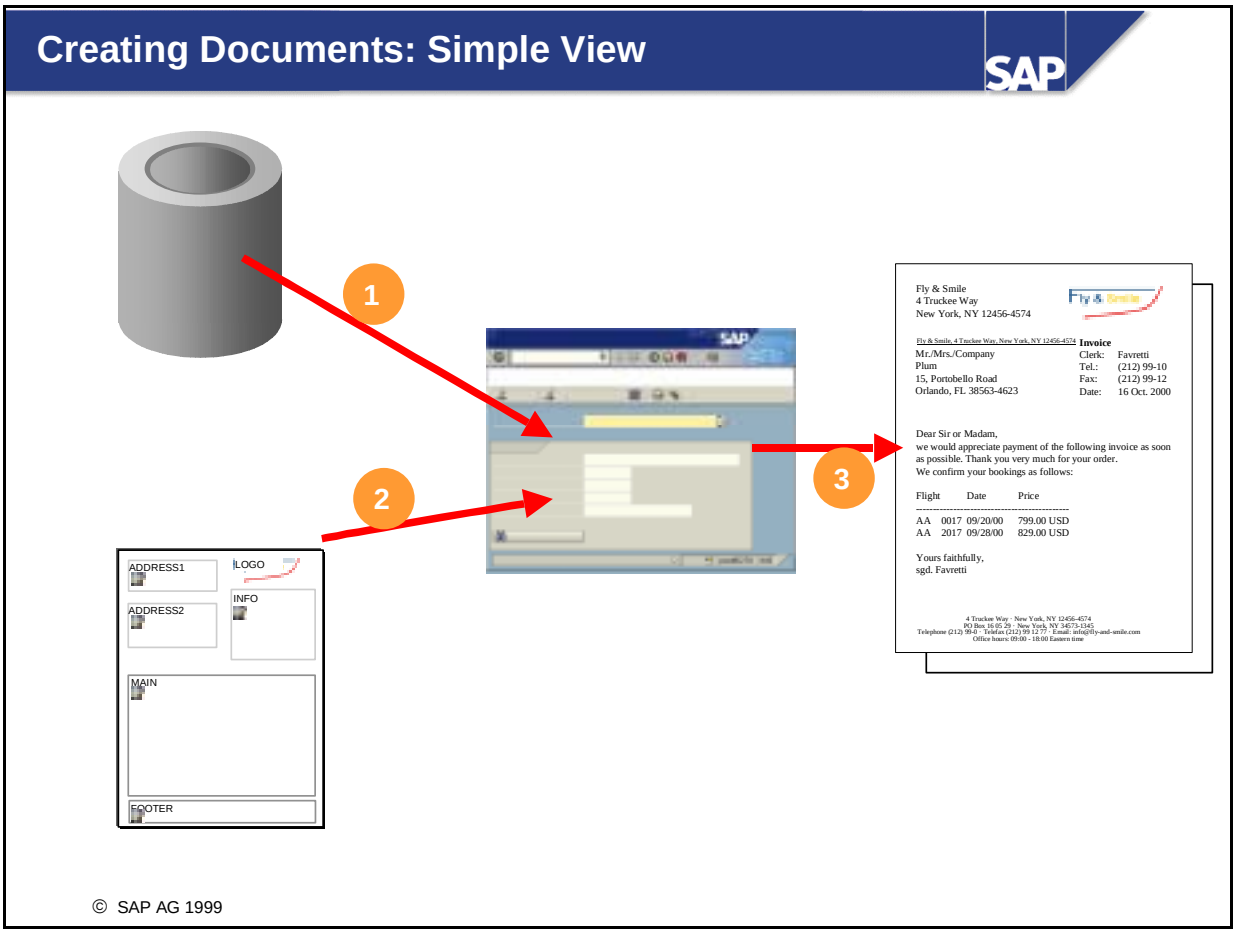
Separation of data retrieval and form logic

```
select
```

No special scripting language

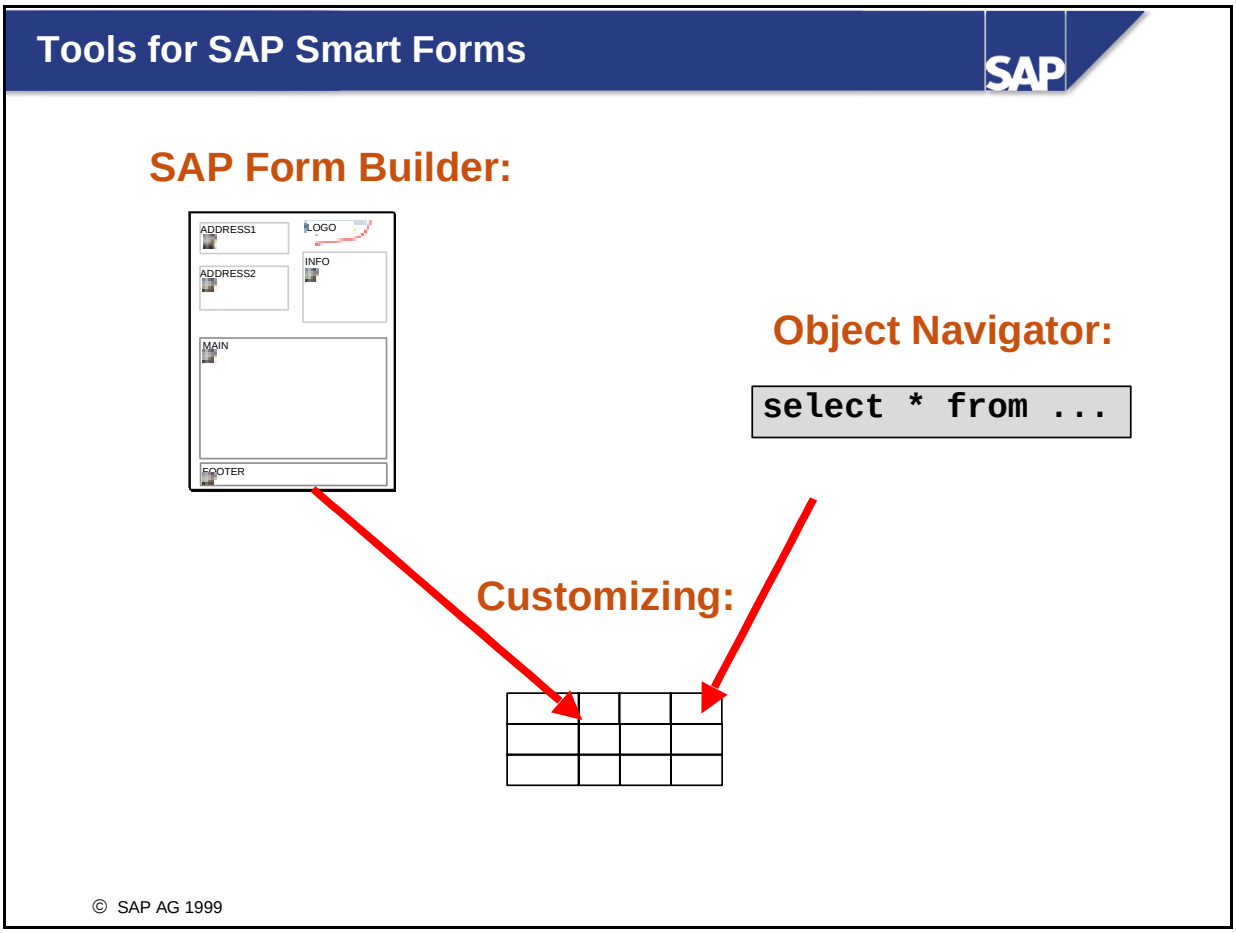
- **Migration of SAPscript forms and styles is supported**
- **SAPscript texts can be inserted into SAP Smart Forms**

- Compared with SAPscript, the existing form processing tool, SAP Smart Forms provide decisive advantages. Among them are:
  - Adjusting forms is now considerably easier, on the one hand because the tools offer more functions (for example, you can very easily create tables with the Table Painter of SAP Smart Forms), on the other hand because the interface between the forms and the application programs has been redesigned.
  - Particular tasks such as retrieving additional data within forms no longer require you to use special scripting language commands, as you had to in SAPscript. Besides, you can insert normal ABAP code.
- 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.
- In R/3 Release 4.6C, you can decide for some applications whether you want to work with SAPscript forms as usual or want to use the new SAP Smart Forms. New forms will use SAP Smart Forms.
- SAP will provide continued support and maintenance for SAPscript forms.



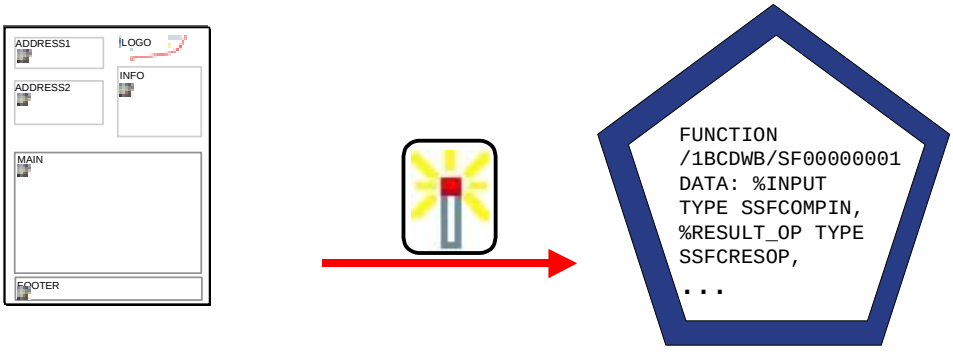
- To allow an application program to output a document, two basic steps are required which are described below in a highly simplified manner:

  1. Data retrieval. The data retrieval step is the central component of the program and can have any degree of complexity, involving user interaction, and so on.
  2. Starting form processing. During this step, the data read is written into the form.



- Maintaining SAP Smart Forms consists of the following tasks:
  1. Adjusting the form - your main task.
  2. Adjusting the application program. This is only necessary in special cases, for example, if you want to modify spool settings or read additional data once that is to be output in all documents of a printing operation.
  3. Saving the changes in Customizing.

Designing SAP Smart Forms



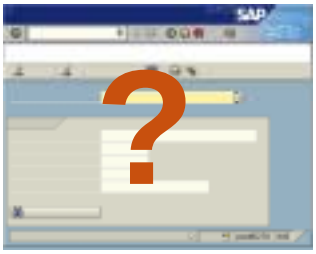
SAP Form Builder:

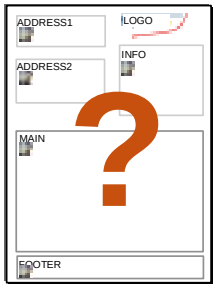
- **What?**
- **Where?**
- **What size?**
- **How often?**
- **Where from?**

Generated function module

- You use the SAP Form Builder and its associated tools to create and adjust forms (transaction: SMARTFORMS). There you define the layout (such as the position and 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.
- Once you have made any necessary adjustments, you must activate the form. During this process, the system first checks if the form contains errors and then saves the form automatically. The main step, however, 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.

Customizing

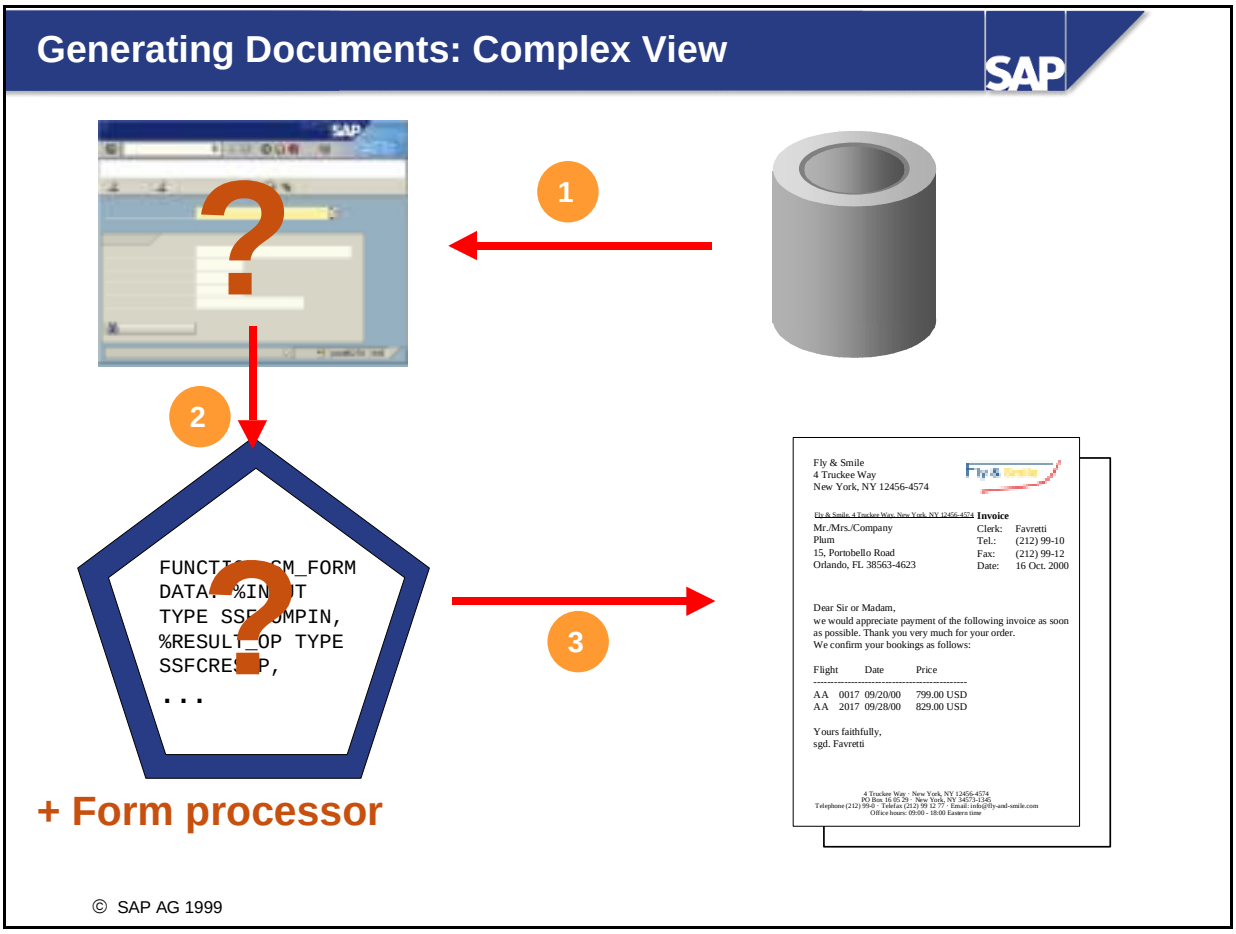




Customizing: Which form to use when?

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

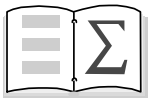
- For some applications, you can decide whether you want to use SAPscript or SAP Smart Forms. You make this decision in Customizing.
- If you decide to use SAP Smart Forms, you must also determine which form to use for the application concerned.
  The reasons are twofold: One, you should not directly modify the forms delivered by SAP but always copy them into your customer namespace and then change this copy. You must then tell the transaction the name of your own form - and you do this in Customizing.
  Two, some applications allow you to choose between various forms for different scenarios. For example, you can determine in Customizing that a separate form should be used for each dunning level (amount and duration of late payment) of the dunning procedure.
- The settings you have to make in Customizing depend on the application you use.



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

  1. The transaction looks up in Customizing which program to call. This program then reads the data.
  2. The transaction learns in Customizing which SAP Smart Form to use for the scenario chosen, calls the appropriate function module generated and thus triggers the form processing process. The interface is filled with the data read.
  3. When the form processing is started, the form processor (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.

SAP Smart Forms: Overview - Summary



You are now able to:

- **List the areas in which form printing with SAP Smart Forms can be used**
- **Explain how SAP Smart Forms function and how they are integrated into application programs**

First Steps with the SAP Form Builder

BC470 Form Printing with SAP Smart Forms

First Steps with the SAP Form Builder: Contents

Contents:

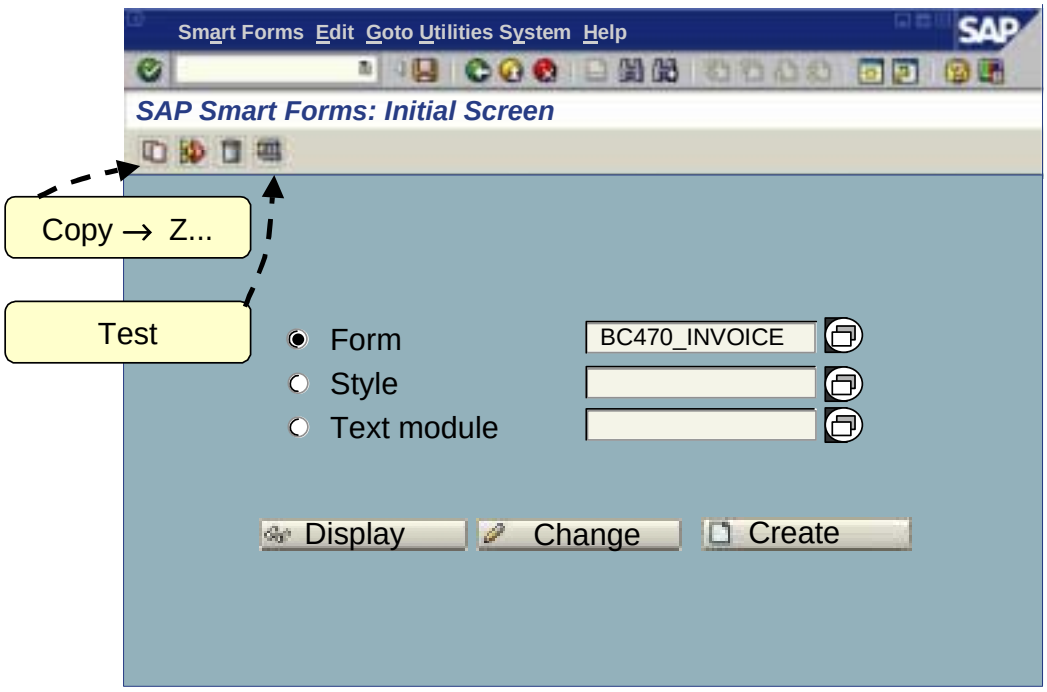
- Creating and maintaining SAP Smart Forms using the SAP Form Builder
- Overview of the most important form elements

First Steps: Unit Objectives

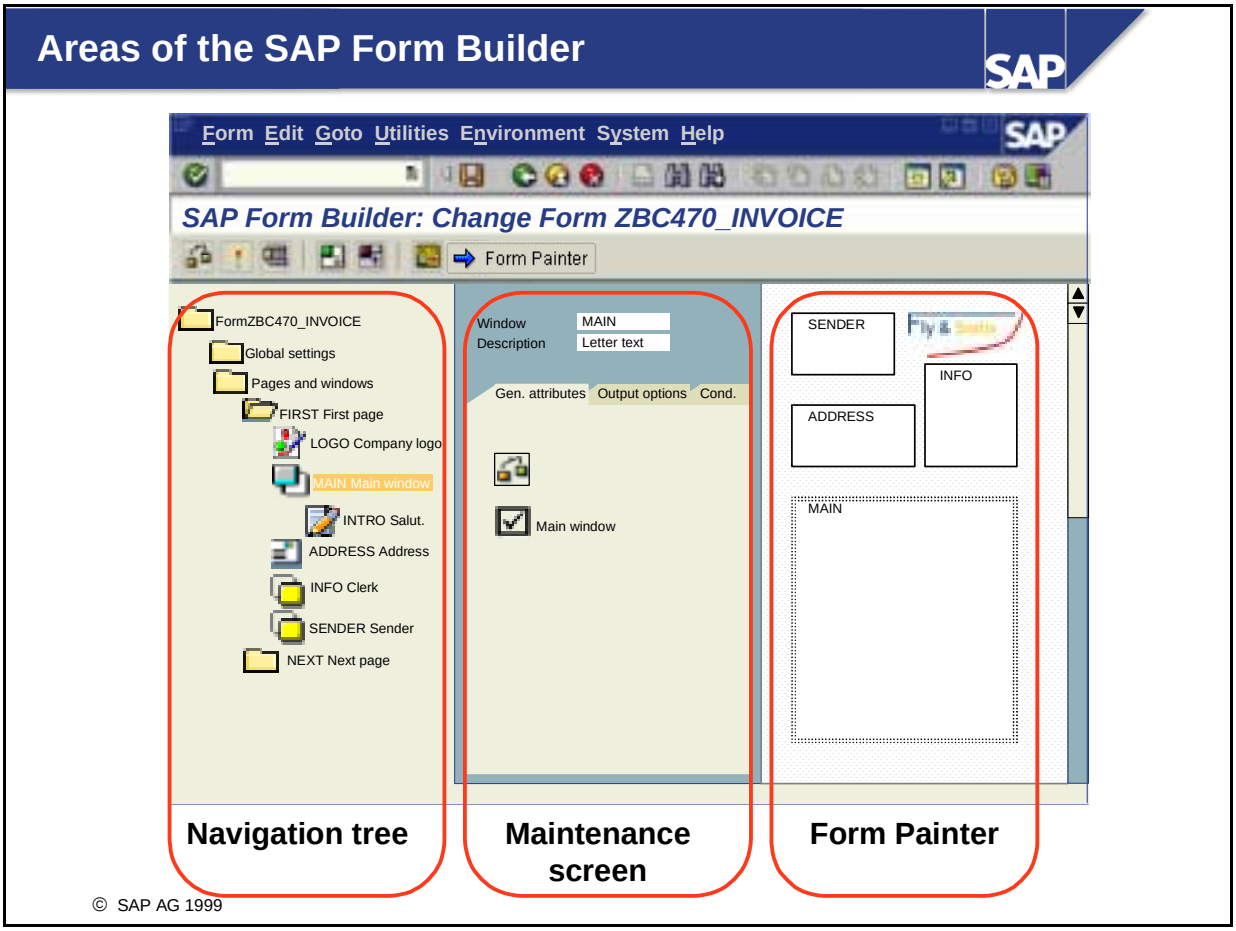
At the conclusion of this unit, you will be able to:

- **Work with the SAP Form Builder**
- **Create, copy and edit forms**
- **Create pages and windows**
- **Explain the different window types**
- **Use background pictures**
- **Set output options**
- **Test forms**

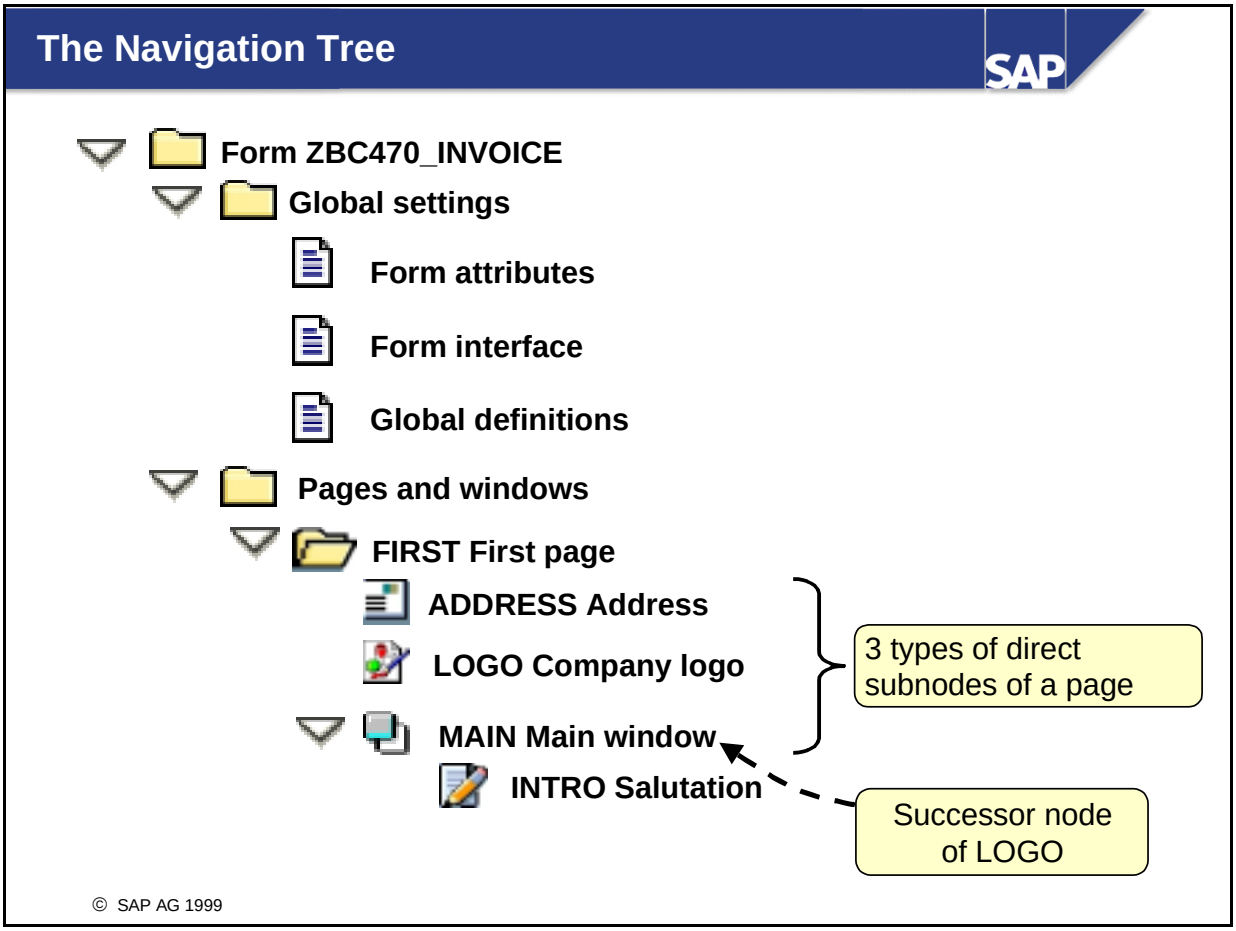
Initial Screen: Maintaining Forms



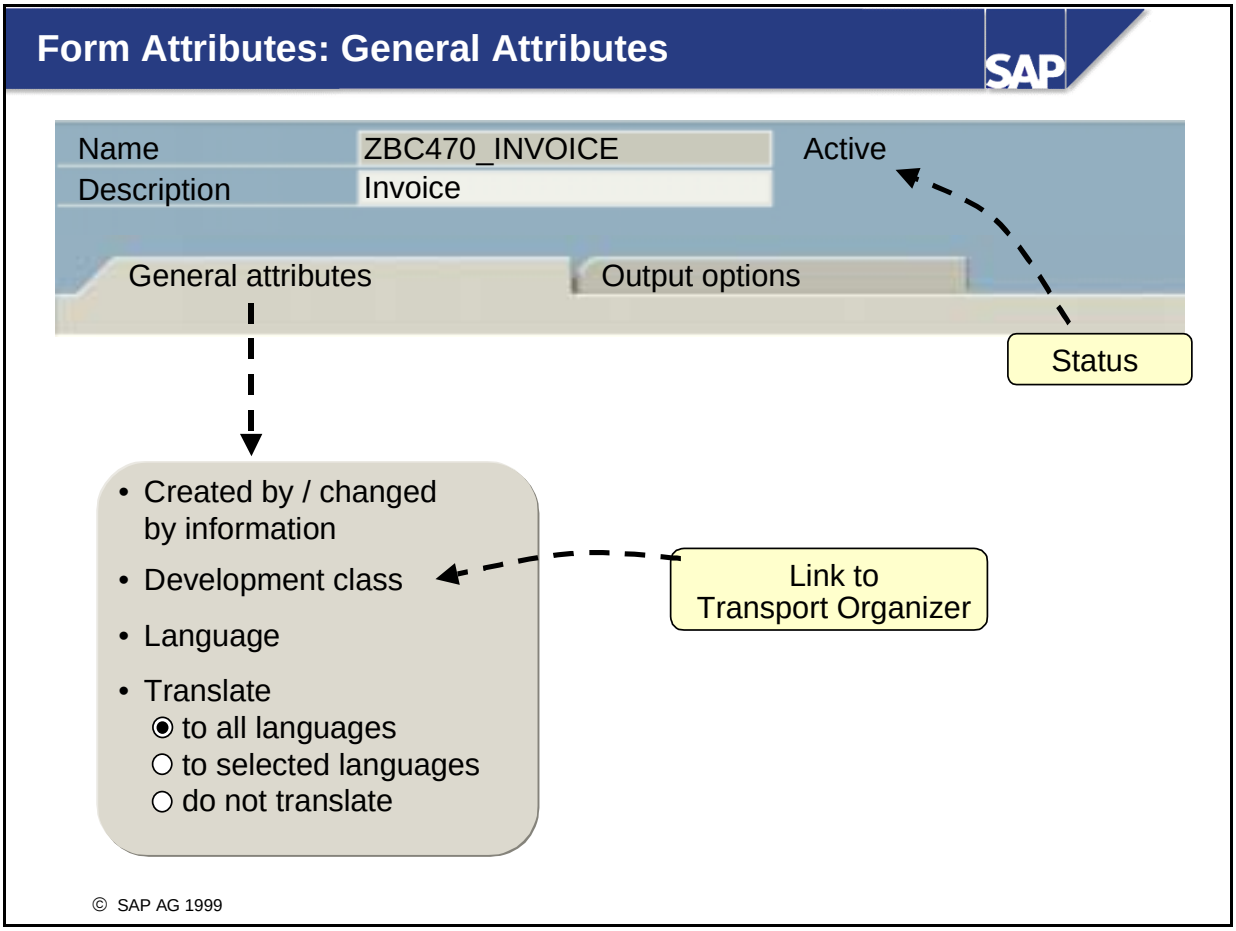
- To call the initial screen of the SAP Smart Forms maintenance transaction, enter *SMARTFORMS* in the OK code field, or choose *Tools → Form printout → Smart Forms* from the menu. Then choose one of the radio buttons depending on which type of SAP Smart Forms object you want to edit:
  - Forms
  - Styles (see Unit 9 - *Smart Styles)*
  - Text modules (see Unit 4 - *Texts, Addresses, and Graphics)*
- If you want 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 then 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 then modify the form copied as required.
- You can also rename, delete, or test forms. To do this, use the available pushbuttons or the options of the *Smart Forms* menu. Testing a form requires that it has been activated before (in the SAP Form Builder).
- Additionally, you can 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*.



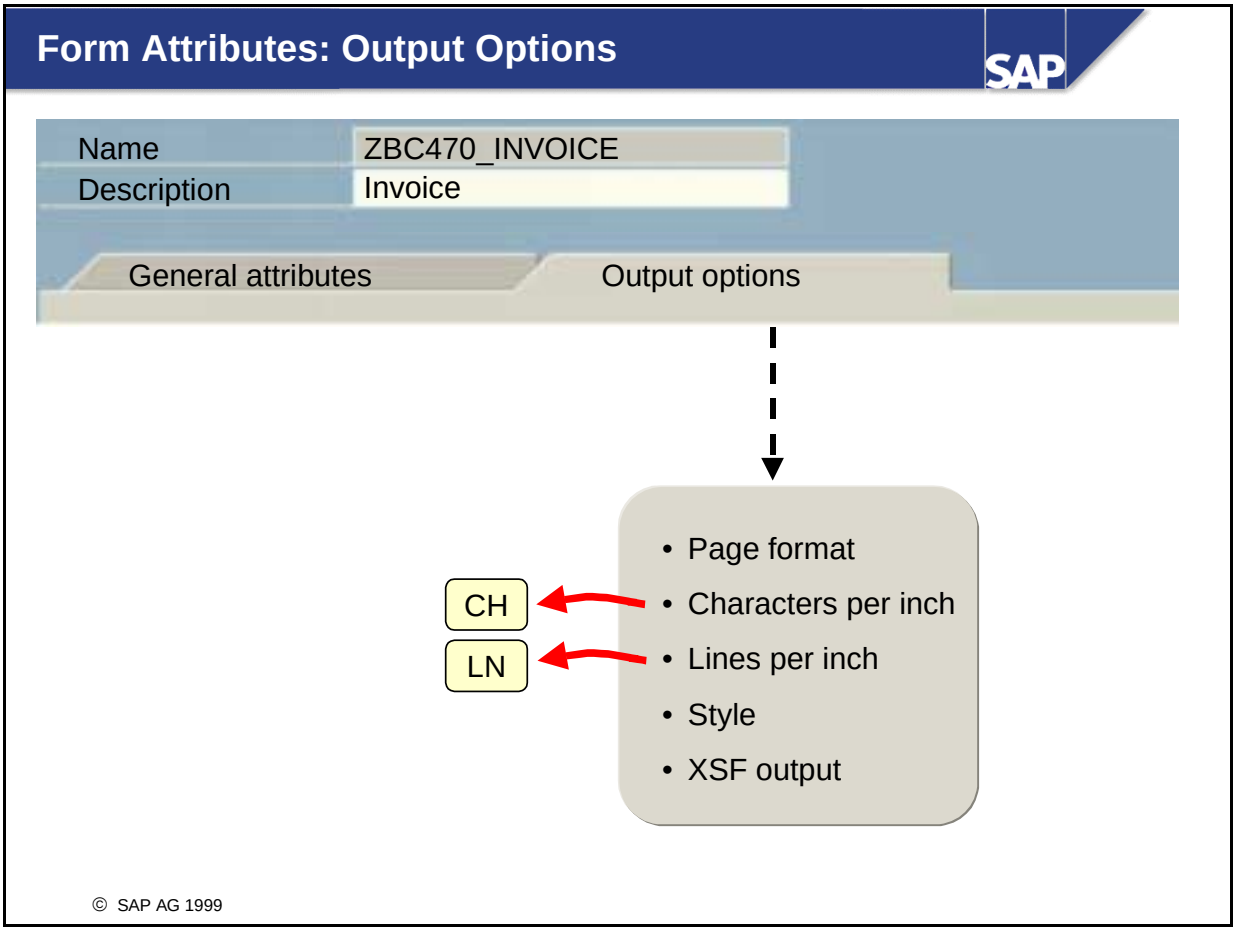
- You edit forms using the graphical SAP Form Builder.
- The SAP Form Builder is divided into three areas:
  - On the left-hand side: The navigation tree. This tree graphically displays the hierarchy of the SAP Smart Form. The individual form elements (such as pages or graphics) are represented by nodes. You can additionally display the field list with variables below the navigation tree. For more information, see Unit 5 - *Data in Forms.*
  - In the middle: 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, for example.
  - On the right-hand side: 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 if you wish: *Utilities → Form Painter on/off.*
- You can select nodes to edit them by double-clicking them in the navigation tree or the Form Painter.



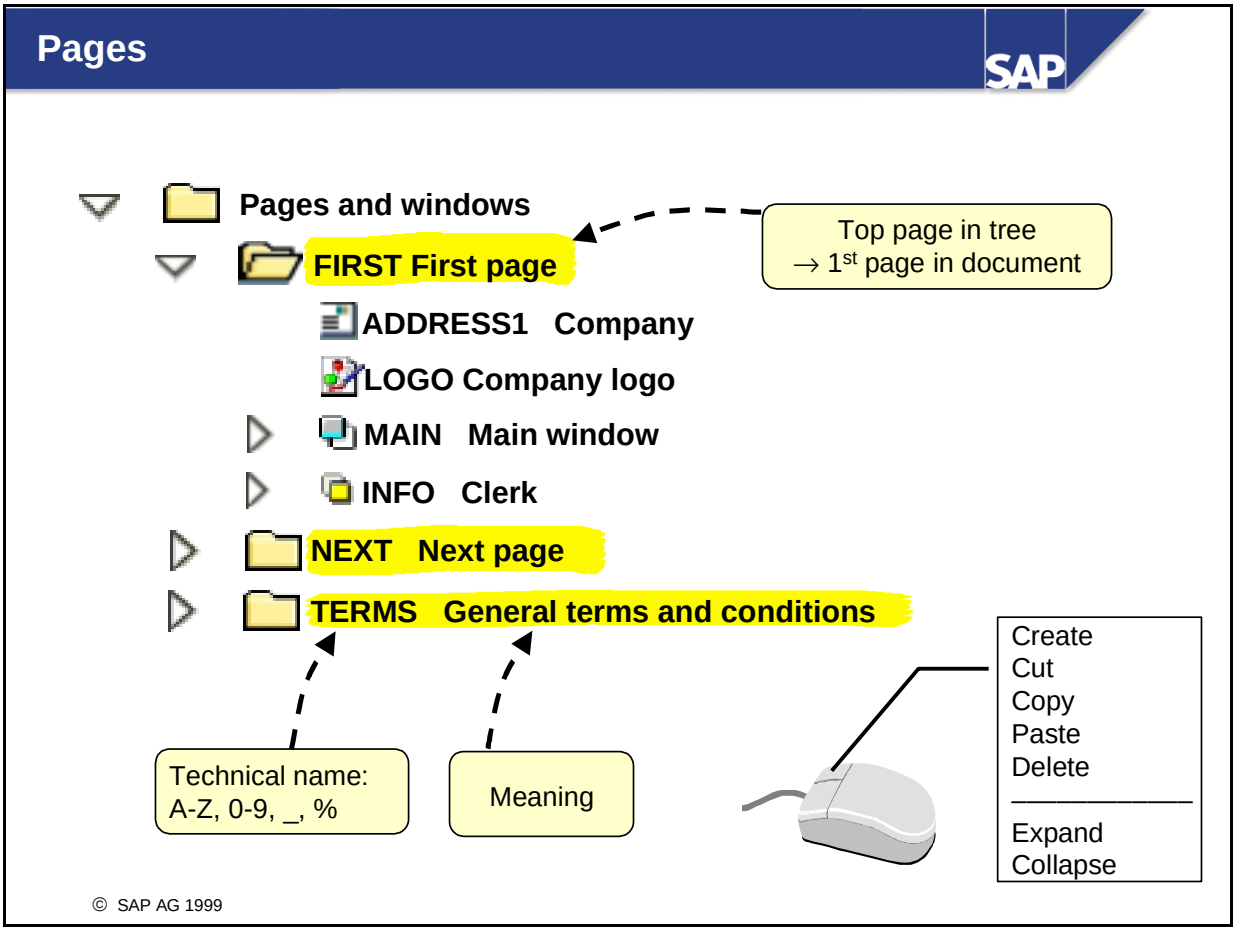
- All elements of a form are represented by a specific node in the navigation tree.
- **Subnodes** "inherit" the attributes of higher-level nodes, for example, the style. If a node is not processed, then also all its subnodes are not processed. A **successor node** of a node, however, is independent. 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 to edit it by double-clicking it. The system then 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 to be returned to the application program. (See Unit 5 - *Data in Forms.)*
    - Global definitions: Here you can define additional fields to be used in the form. (See Unit 5 - *Data in Forms*.)
  - Pages and windows (see later in this unit).



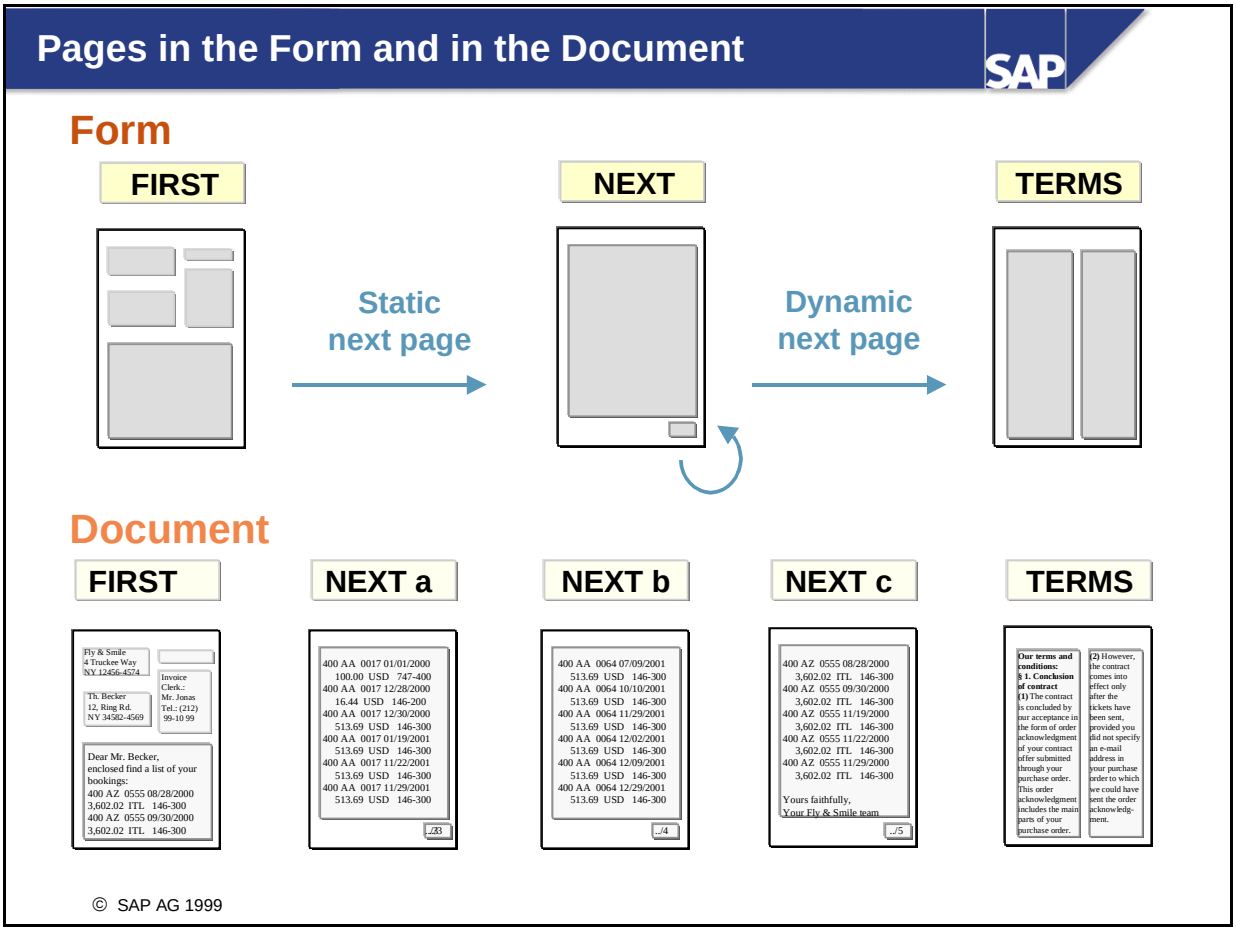
- The form attributes include not only the name and description of the form but also its current status: Active or inactive. A form can exist in any of these two versions. Application programs always use the active version. This means you can provisionally save your changes without directly affecting application processing. To activate a form, choose *Form → Activate*. Note that when you copy or rename a form, the status of the copy is always set to *inactive*.
- Since SAP Smart Forms are integrated with the R/3 transport system, you must assign them to a development class. You do this when you first save your form. You can subsequently change the development class assigned from within the SAP Smart Forms initial screen by choosing *Goto → Object directory entry*. Be sure to use the customer namespace (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 yes, into which ones.



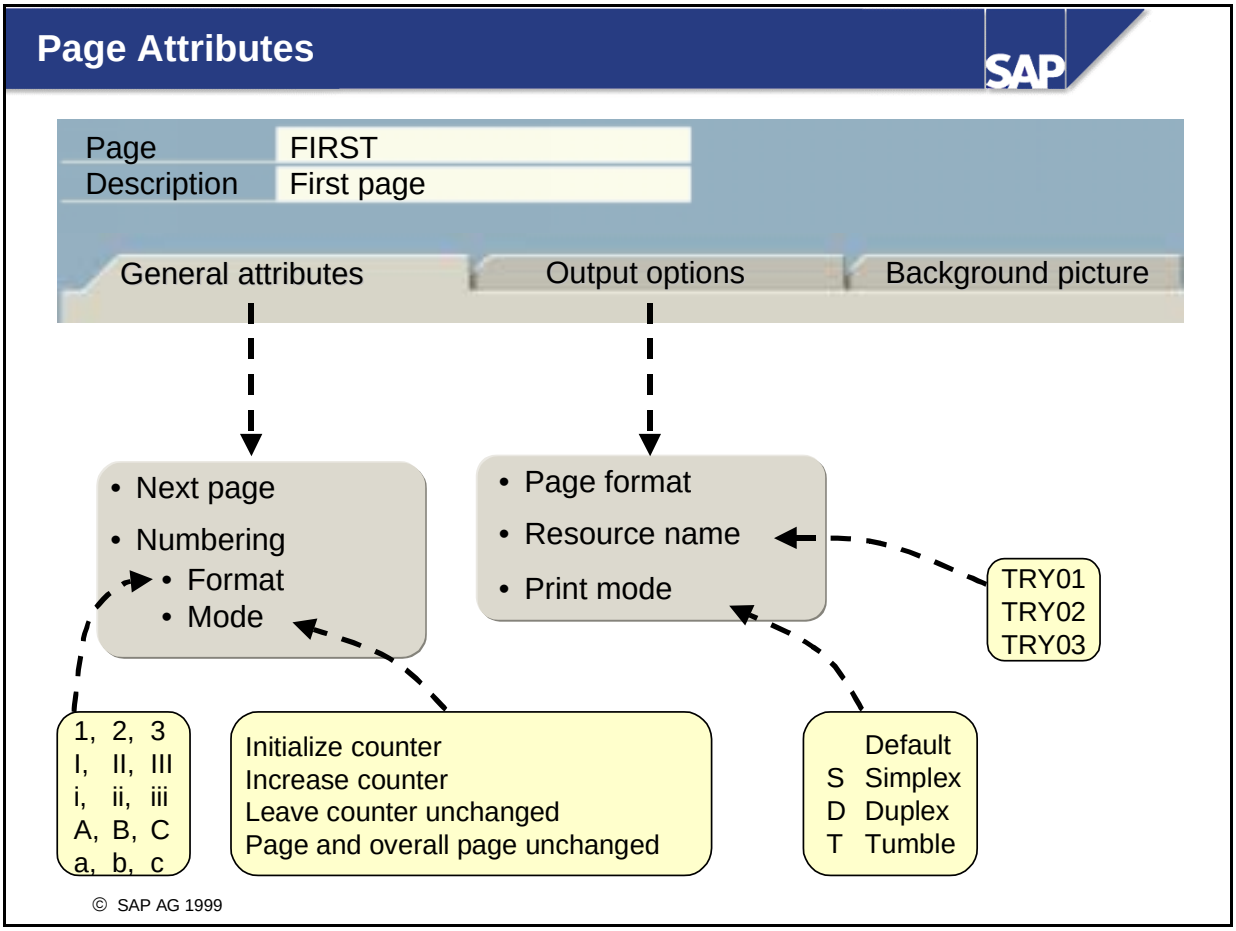
- The page formats available include the page formats provided in spool administration. The orientation (portrait or landscape) is set individually for each page.
- You can specify a default page format for new forms. From the initial screen of transaction SMARTFORMS, choose *Utilities* → *Settings* → *General* tab to do this.
- You must assign a style to each form. A style is a collection of different character and paragraph formats that 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. See Unit 9 - *Smart Styles*.
- *Characters per inch (CPI).* This field allows you to determine the *CH* unit of measure which 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 which you can use for vertical length specifications (such as window lengths) in the form.
- Instead of passing forms to spool management, you can also exchange data between systems by means of an XML data stream. A certified interface is available for this purpose, called the SAP Smart Forms XML Interface (XSF). To use this interface, select the *XSF output active* checkbox. (This setting can be overridden by the application program.)



- Each form has 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 relevant options available:
  - Create or delete (in change mode only). When you create a new node, the system proposes a unique technical name that 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
- You can also call these functions from the menu by choosing *Edit → Node* or *Edit → 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. Therefore you should not use it.

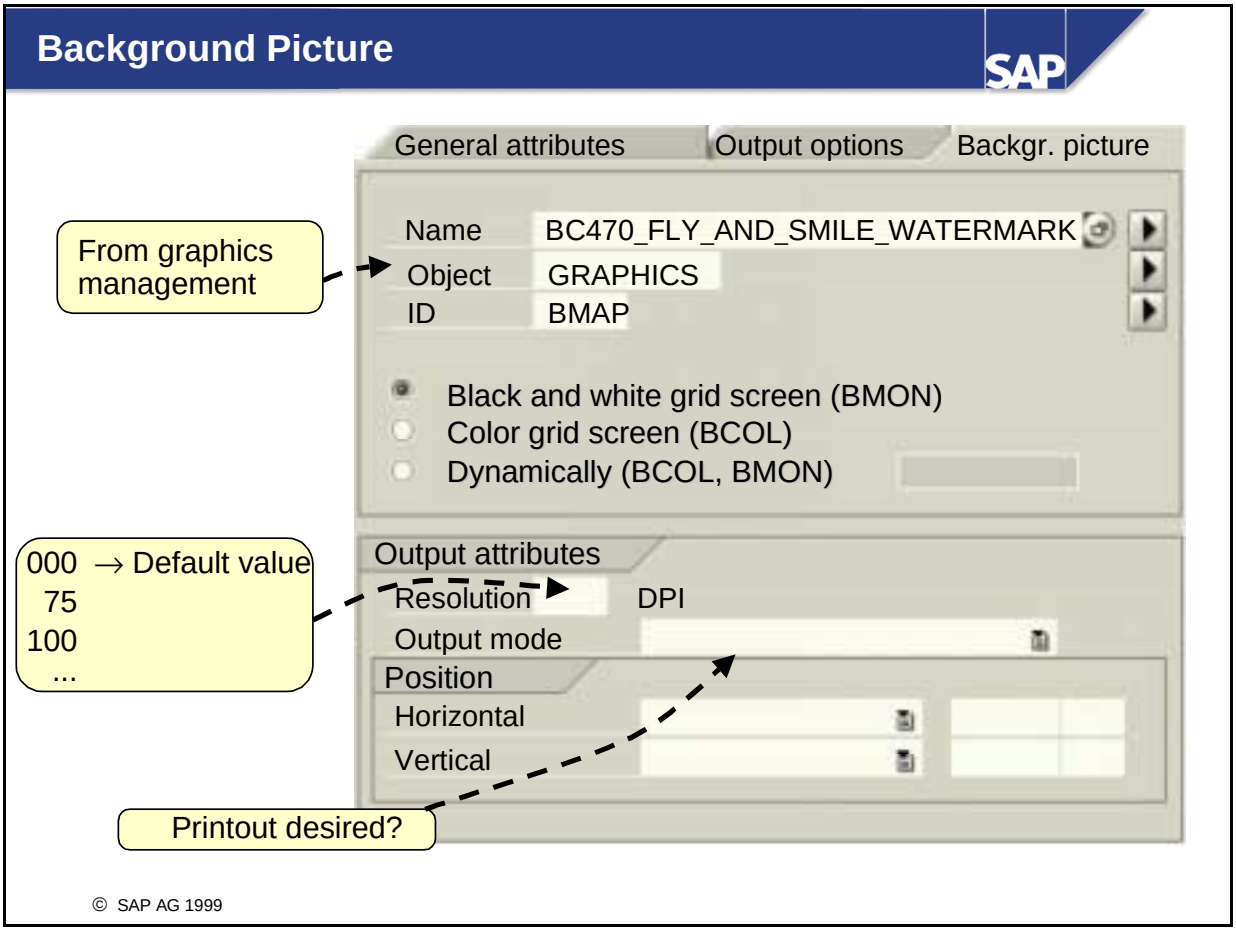


- You can define one or several pages with different layouts for a form.
- The top page of the navigation tree is processed first. You then control the processing sequence by specifying the next page (on the *General attributes* tab for the page) which is then processed automatically after the top page. (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 then force the system to change to the page containing the general terms and conditions of business. See also Unit 7 - *Flow Control*.)
- Depending on the amount of data to be processed, a form page can be used more than once in a document.

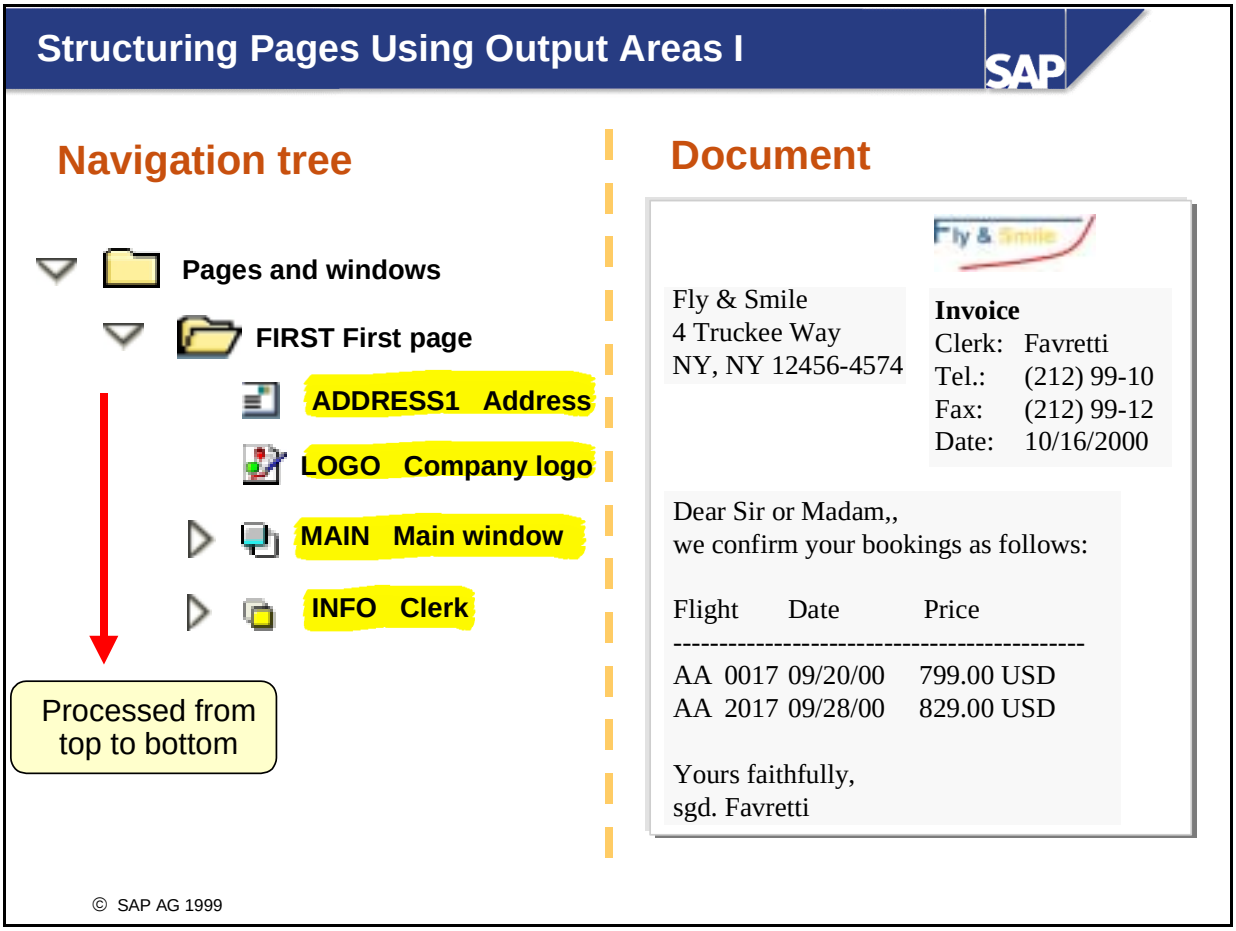


■ You can make settings on the following tabs:

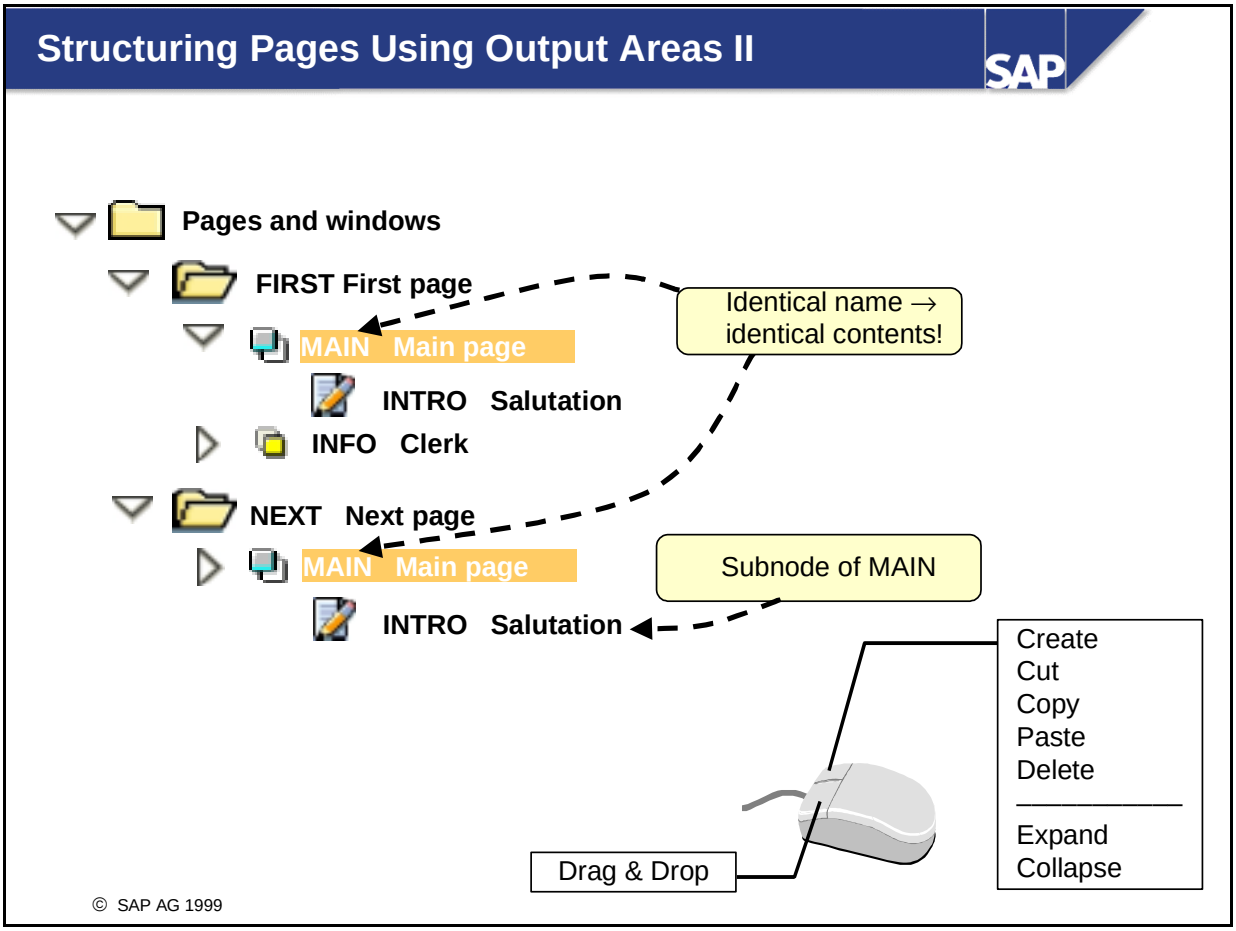
- *General attributes*
  - The next page. The default value is the page itself.
  - The type of automatic page numbering. You can choose between Roman and Arabic digits and upper case and lower case format on the other hand; on the other hand, you can determine the behavior of the page counter. Note: If you make page number settings, this does not mean that your pages are numbered. To have your pages numbered, you must output the variable SFSY-PAGE in a text window. See the *System Fields* slide in Unit 4 - *Texts, Addresses, and Graphics*.
- *Output options*
  - While the page format you specify applies to the entire form, you set the orientation (portrait or landscape) at page level only.
  - You can assign different paper trays to pages - provided your printer supports this feature. This makes sense, for example, 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 if required. Prerequisite: Your printer supports this feature.
- *Background picture*
  Prerequisite: The picture is already available in the system. See *Graphics Administration* in the Appendix. Text is printed over the picture.



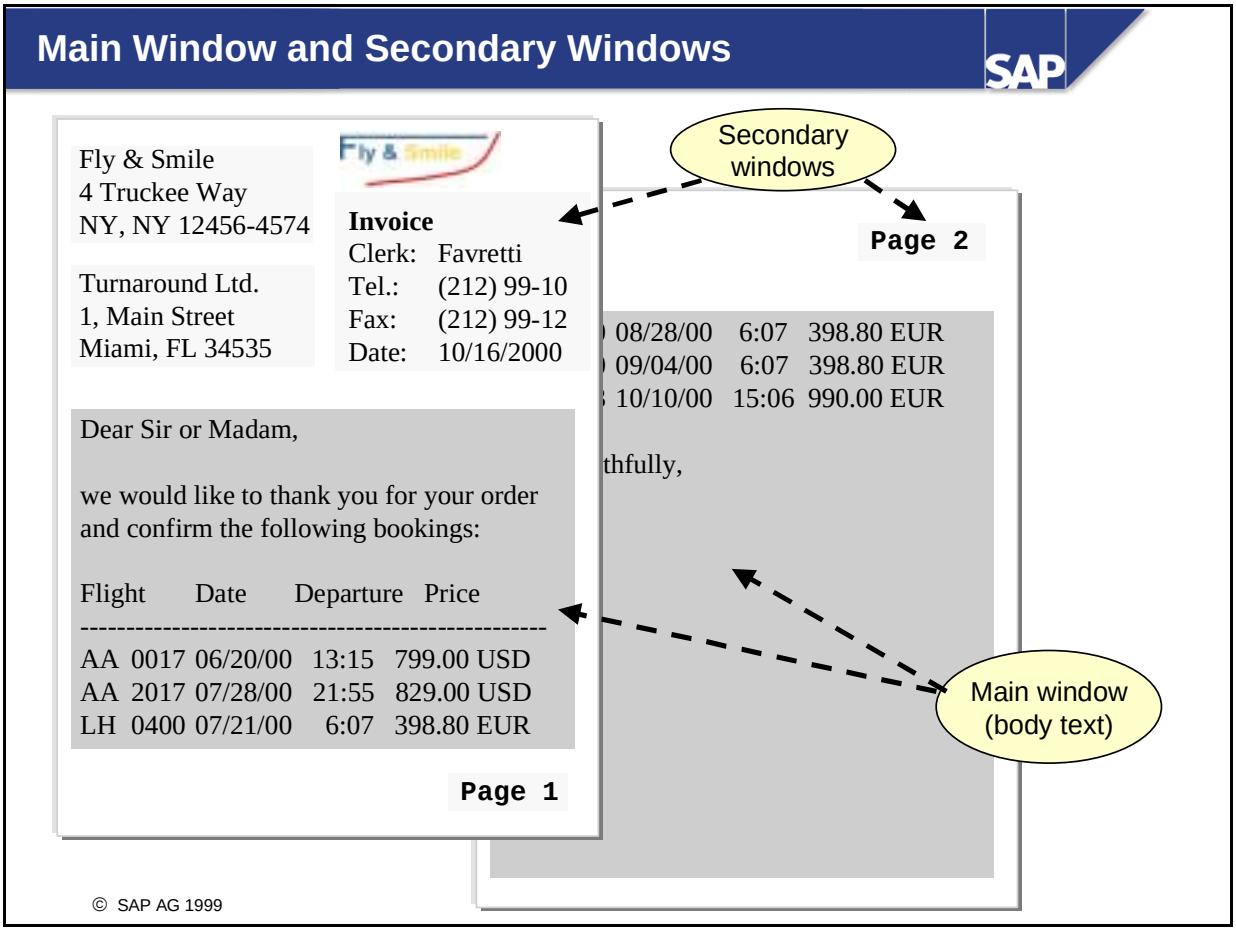
- Specify the name of the graphic as well as its description (object and ID). The default values are GRAPHICS and BMAP. The F4 help allows you to view which graphics are available in the system.
- Determine whether it is a black and white picture or a color picture.
- Hint: You can also select the background picture dynamically by entering fields instead of static names which must have a value assigned to them at runtime. Of course, this procedure is not suitable for redrawing templates since you need the preview in the Form Painter for such a task.
- 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 (which 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*. Nevertheless, you can still 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* on the maintenance screen.



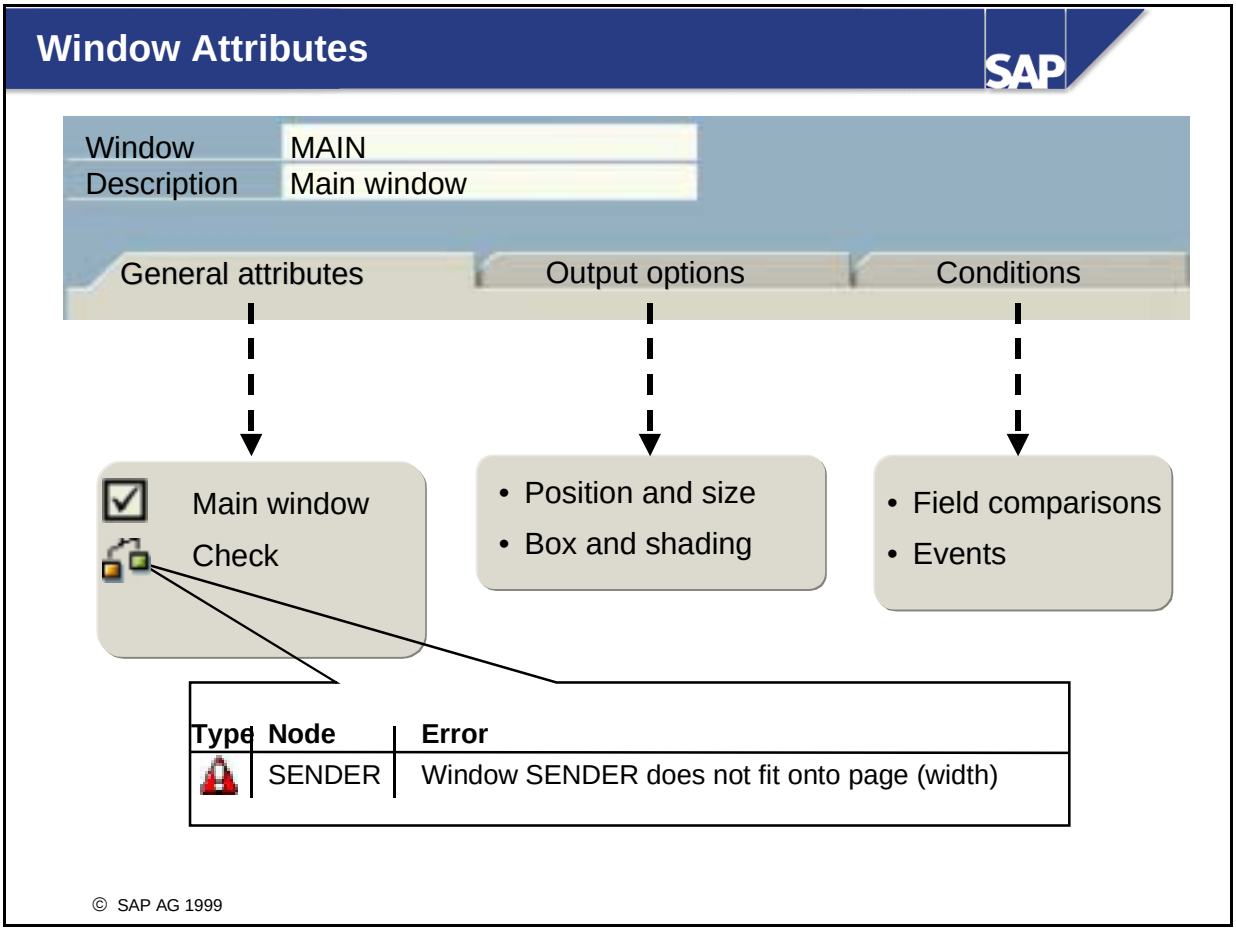
- To be able to output information in a form, you must create suitable output areas (windows) on the relevant page. The following output areas are available:
  - Windows: Subnodes of windows are used to output text and data.
  - Address windows: If the application program uses central address management (CAM), you can easily output formatted addresses in address nodes.
  - 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 (address, graphic, or window).
- The order of page subnodes in the navigation tree does not affect their position in the form, but 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 & Drop (left mouse button). The processing sequence is particularly important if you use fields (variables) that are only filled at runtime.



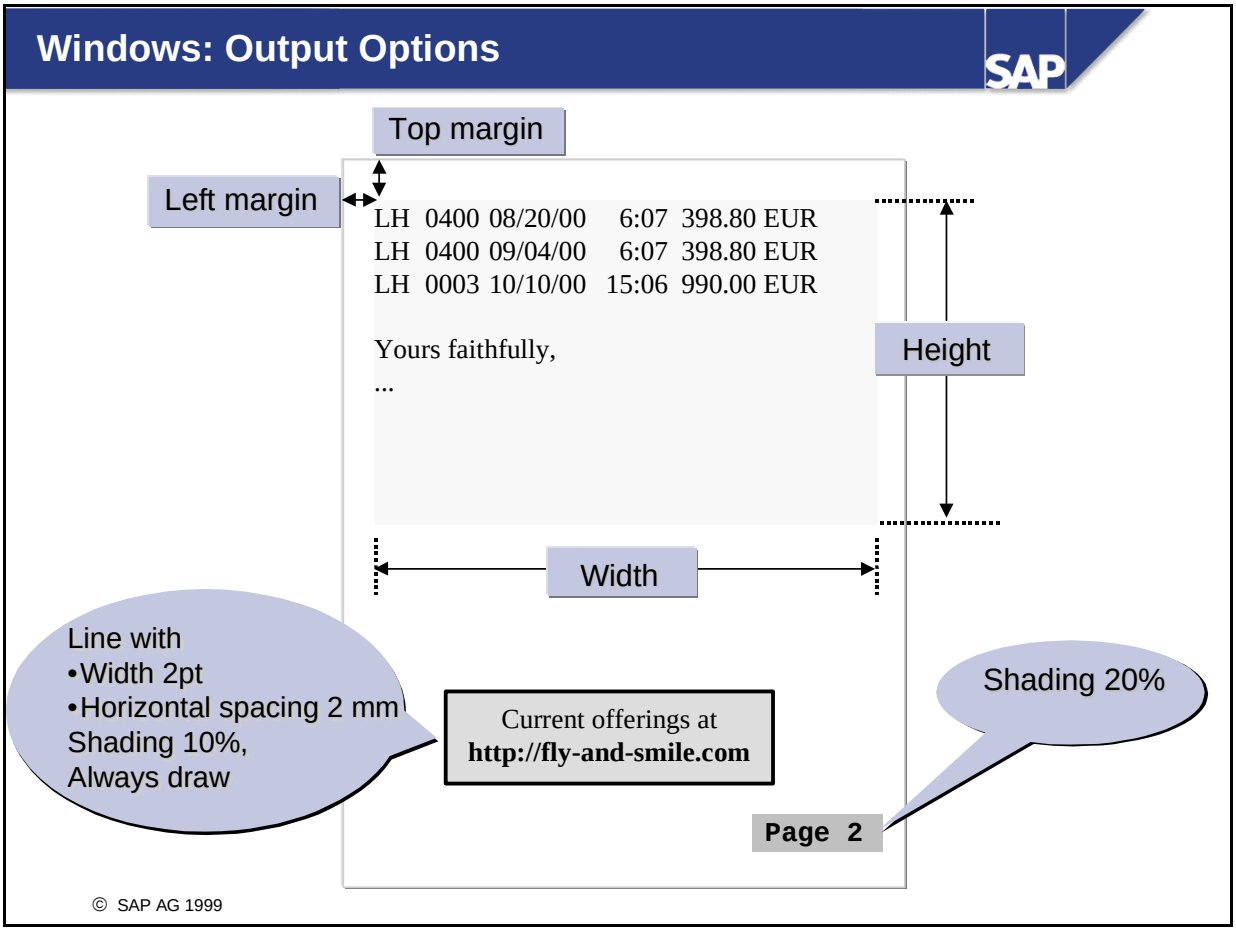
- You create output areas like any other node using the context menu (right mouse button). The system proposes a unique technical name that you can change if required.
- You can use Drag & Drop to move (left mouse button) or copy (Ctrl key and left mouse button) subtrees, including nodes with subnodes. Alternatively, you can use the clipboard (right mouse button: cut - copy - paste). For example, you can move or copy windows or text nodes between pages.
- If you drag a node A on a node B, node A is inserted after node B. In some cases, you can also insert node A as a subnode of node B. The system then displays a dialog box on which 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 node B.
- 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. However, although the output areas have the same contents, their position may 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 respective page.



- There are two types of windows: main windows and secondary windows.
- In the subnodes of the **main window**, you output text and data that may cover several pages (called the body text) such as the bookings of a customer. When the main window is completely filled with text and data, the text is displayed in the main window of the next page. The pages are broken automatically. (You can use a different next page than that set as the default. See Unit 7 - *Flow Control*.)
  - 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 as required.
  - A page without a main window may not refer to itself as the next page since this would cause an endless loop. In this case, the system terminates after processing three pages.
- 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 and data that do not fit into the secondary window are truncated and not output.
  - The height, width and position of a secondary window may be different for each page.
- Graphics are automatically set to the correct size. As far as addresses are concerned, only the most important information is shown if the output area is too small.

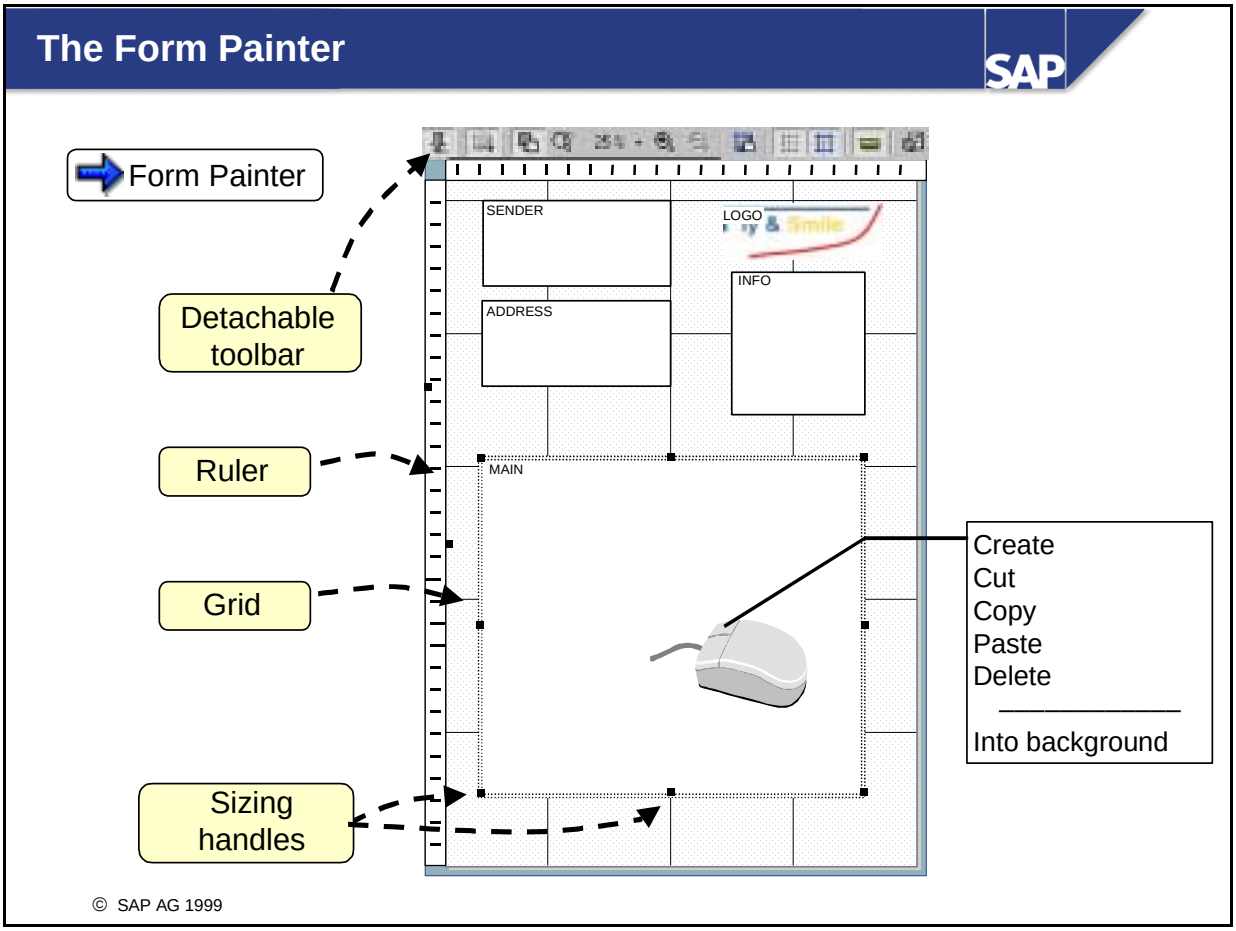


- You can make checks at several levels, from the bottom node to the entire form. To ensure that all windows are free of errors, they must fit onto the respective page, and the main window must have the same width on all pages. All error messages are displayed in the bottom part of the maintenance screen. 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 (see next slide).
- As with most node types, you can use conditions for windows to determine the time when they are processed. On the one hand, you can choose from a number of processing events (such as *not on first page* or *only on first page*); on the other hand, you can control processing using specific values. For example, you may want to print text A only for certain customers, and text B for all other customers. For more details see Unit 7 - *Flow Control*. If the conditions set for a window are not fulfilled, the window and all its subnodes are not 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, then each node has its own *Output options* and *Conditions* tabs.

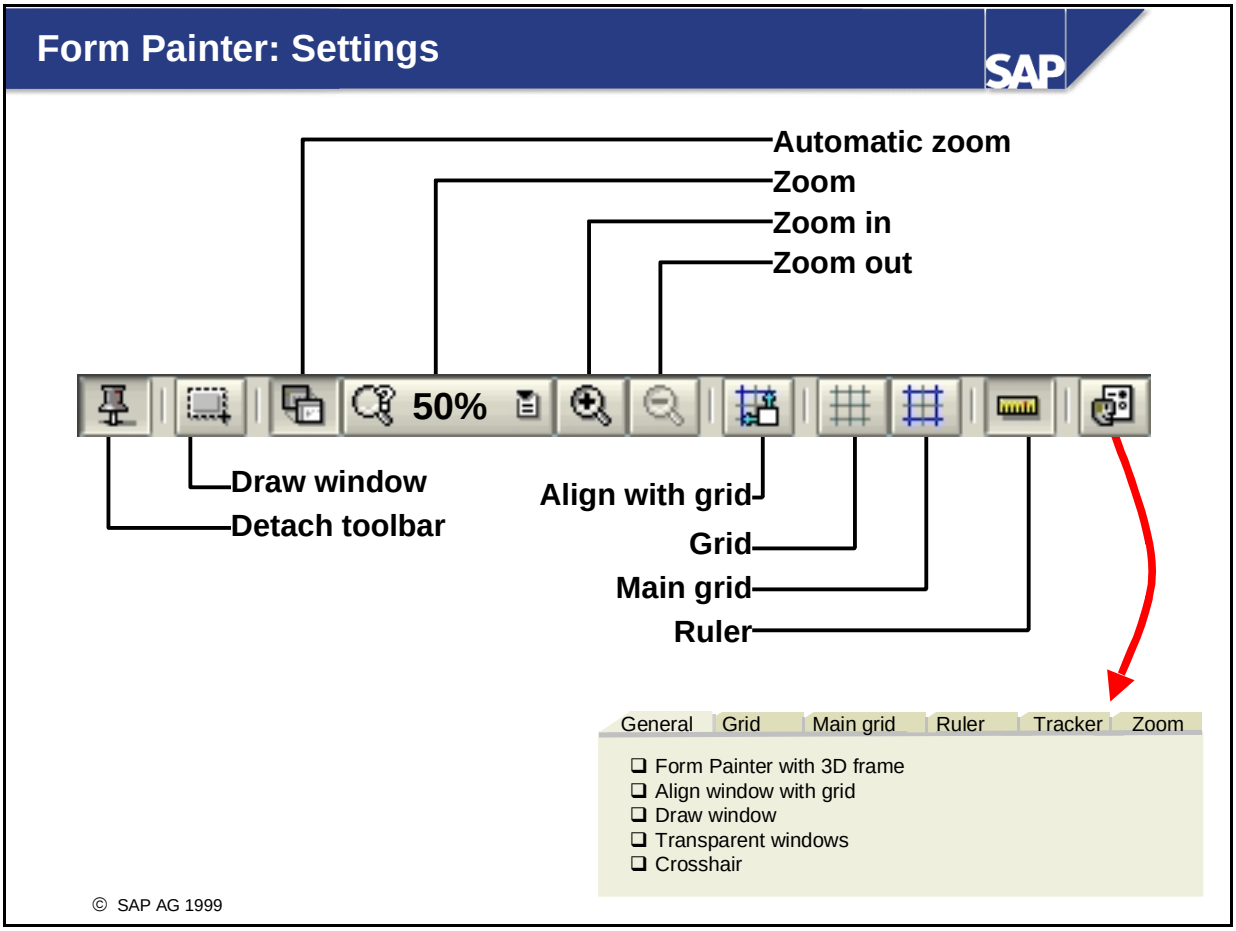


- You determine the position of a window by specifying the upper left margin, and its size by entering its height and width. If you draw a window in the Form Painter, the values you set are automatically copied to the maintenance screen, and vice versa.
- As with all other nodes that allow the output of text, you can define a box and a shading for windows.
- You use the fields *Vertical spacing* and *Horizontal spacing* to define the distance between the box and the text. You should set an appropriate value here, in particular, if you use a thick line for the box, since the box would otherwise be printed partially over the text.
- 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:

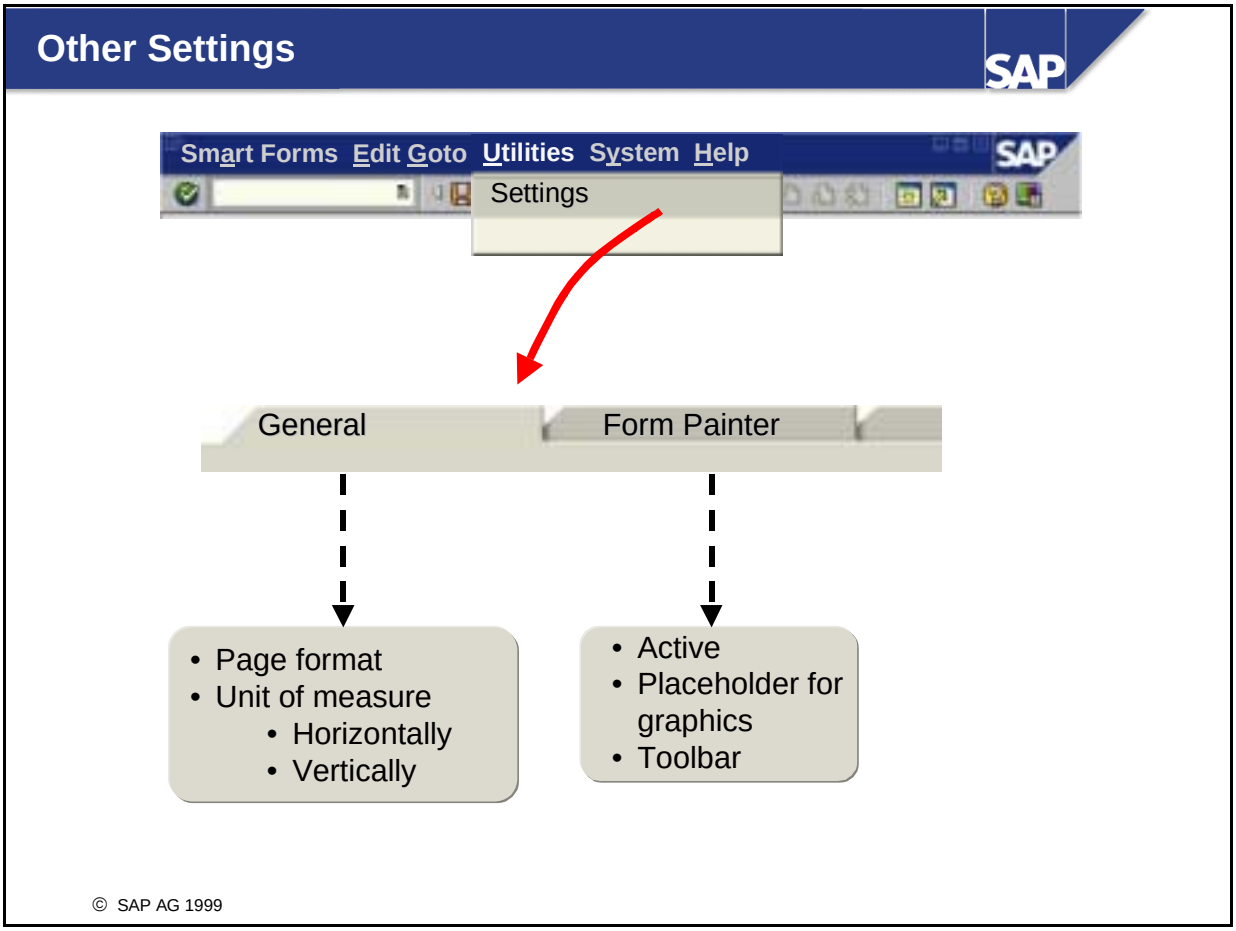
  CM, MM, IN (inch = approximately 2.54 cm), PT (point = 1/72 inches), TW (twip = 1/20 points). For vertical length specifications, you can also use LN, and for horizontal length specifications CH. You define these units in the form attributes.



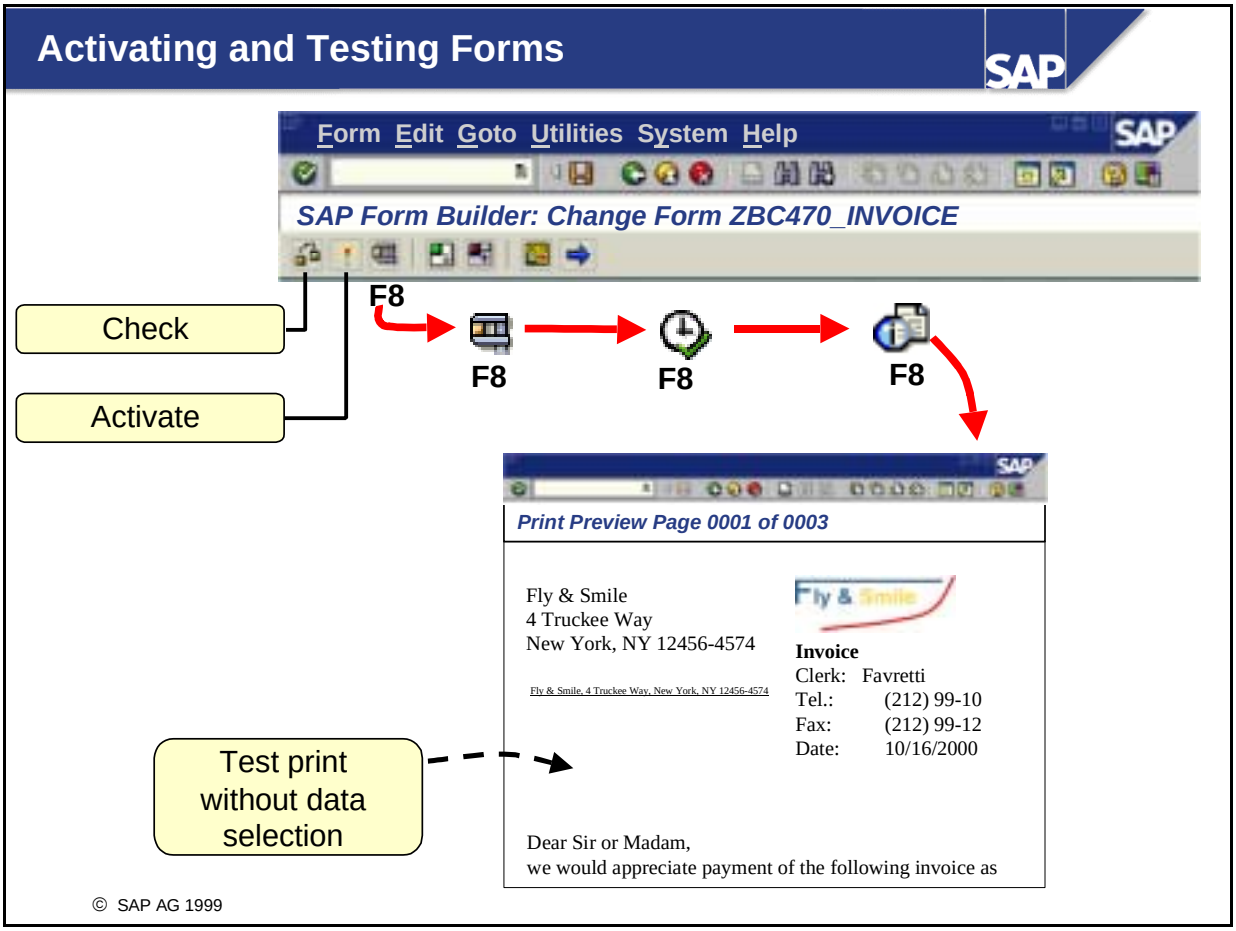
- You use the Form Painter to check/change the layout of a form. You can show or hide the Form Painter in the Form Builder by clicking the corresponding pushbutton or by choosing *Utilities → Form Painter on/off* from the menu. The Form Painter always displays the page selected in the navigation tree, 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 then 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 & Drop). All size- and position-related changes that you make are automatically copied to the maintenance screen.
- The context menu (right mouse button) is not only available in the navigation tree, but also in the Form Painter. You can use this menu to create or delete output areas and perform normal clipboard functions (cut, copy, 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. (The following principles are applied during printing: If windows, graphics or texts overlap, they are printed one by one over another. This means that form elements cannot hide each other.)



- 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 directly draw a window by clicking on a free area in the Form Painter and dragging the mouse with its button pressed until the window has 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 show a detail grid and/or the main grid. You can also make a setting in the Form Painter which ensures that your 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 also helps you to align the nodes correctly. You activate the crosshair cursor on the *General* tab of the Form Painter settings.
- The *Tracker* tab of the Form Painter settings allows you to determine how the currently selected window should be highlighted.

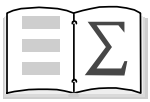


- If you choose *Utilities → Settings*, you can additionally make the following settings:
- *General* tab
  - Default value for the *page format* of new pages
  - Default values for horizontal and vertical *units of measure*
- *Form Painter* tab
  - Activate/deactivate the Form Painter (*Active* checkbox)
  - *Placeholder for graphics.* This setting is sometimes useful to increase performance, for example, if your workstation computer is too slow. If you select this checkbox, graphics are represented in the Form Painter by means of borders that are of the same size.
  - Activate/deactivate the *toolbar* of the Form Painter.



- When you make changes to your form, you should test it afterwards. As you have seen, you can make local checks at node level or check the entire form.
- Before you can test a form from within the SAP Form Builder or use it in application programs, you must first 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. To find out the name of the function module, choose *Environment → Function module name.*
- If you have changed and saved your form, but then 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 (active) form from within the Form Builder (by using the print preview or 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 that is the development environment for function modules. The name of the function module generated is already entered here. For testing purposes, you can also populate the interface of the function module with values. Choose *Test* again (menu: *Function module → Test → Single test)*. Choose *Execute*. On the print attributes dialog box, enter your printer in the *Output device* field and then choose *Print preview* or *Print*. Tip: The quickest way to go to the print preview is to press the function key F8 four times in the SAP Form Builder.

First Steps: Summary



You are now able to:

- **Work with the SAP Form Builder**
- **Create, copy and edit forms**
- **Create pages and windows**
- **Explain the different window types**
- **Use background pictures**
- **Set output options**
- **Test forms**

General Information About the Exercises

Explanation of the Symbols Used in the Exercises and Solutions

	**Exercises**
	**Solutions**
	**Objectives**
	**Overview**
	**Hints and Tips**

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

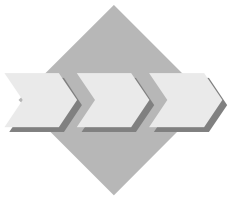
Please observe the naming conventions specified in the exercises. This makes it easier for you (and the instructor) to track down errors.

Exercises

Unit 3: First Steps with the SAP Form Builder

At the conclusion of these exercises, you will be able to:

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



Your task: Copy an existing form and make some changes to that form using the SAP Form Builder.

Copy template for the form: BC470_STEPT

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

1. Get an overview

   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.

2. Copy template

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

3. Set general attributes

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

4. Select page format

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

5. Set background picture

   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.

   The preview of the Form Painter is very rough. For a more WYSIWYG-oriented display, you need to activate the form and test the generated function module.

6. Create new window

   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.

7. Create additional page

   7-1 Create an additional page with the name NEXT.

   7-2 NEXT should be the next page of FIRST and of NEXT itself.

   7-3 On the NEXT page, you need 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.

   7-4 **Optional:** Page numbering should start on page NEXT, at II in Roman format.

8. Test form

   8-1 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?

   8-2 Test your form again using the program SAPBC470_DEMO.

Solutions

Unit 3: First Steps with the SAP Form Builder

1. Get an overview

   Choose *System -> Services -> Reporting*. Enter the name SAPBC470_DEMO, and execute the program (function key F8). Enter BC470_STEPT as the form name, and execute the program (function key F8).

2. Copy template

   Start transaction SMARTFORMS by entering the transaction code into the OK code field or by choosing *Tools -> Form printout -> Smart Forms* from 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. The name of the form copied is then automatically transferred into the *Form* field. Choose the *Change* pushbutton. The system then takes you to the SAP Form Builder.

3. Set general attributes

   You make these settings on the maintenance screen that 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. Then click on the icon to the right and select the languages on the dialog box that appears next.

4. Select page format

   The page format is defined centrally for the entire form (with the exception of the orientation that can be set at page level). Go to the *Output options* tab and choose the DINA4 page format.
   Check the form by clicking the left-most pushbutton 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 right-most pushbutton in the toolbar) and adjust the width of the window with 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. Test your form again. It should now be free of errors.

5. Set background picture

   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 grid screen (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 (right mouse button).

6. Create new window

   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 either 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 into 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.

7. Create additional page

   7-1 From the context menu of the page FIRST, choose *Create -> Page*. Change the name of the new page into NEXT on the maintenance screen and enter a description ("Next page").

   7-2 On the *General attributes* tab, enter NEXT as the next page. Repeat these steps for the page FIRST.

   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.

   7-4 **Optional:** 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.) Then go to the *General attributes* tab of the page NEXT and set Roman digits for the page numbers and the mode *Increase counter*.

8. Test form

   8-1 Check your form as before using the left-most pushbutton in the 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. When you activate a form, the system prompts you for a development class. Enter ZBC470_##. The name of the function module is only 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 three times more. Make sure that the output device entered is the printer your instructor told you.

8-2 See step 1.

Texts, Addresses, and Graphics

BC470 Form Printing with SAP Smart Forms

Texts, Addresses, and Graphics: Contents

Contents:

- Texts (Text Elements, Text Modules, Include Texts)
- Addresses
- Graphics

Texts, Addresses, and Graphics: Unit Objectives

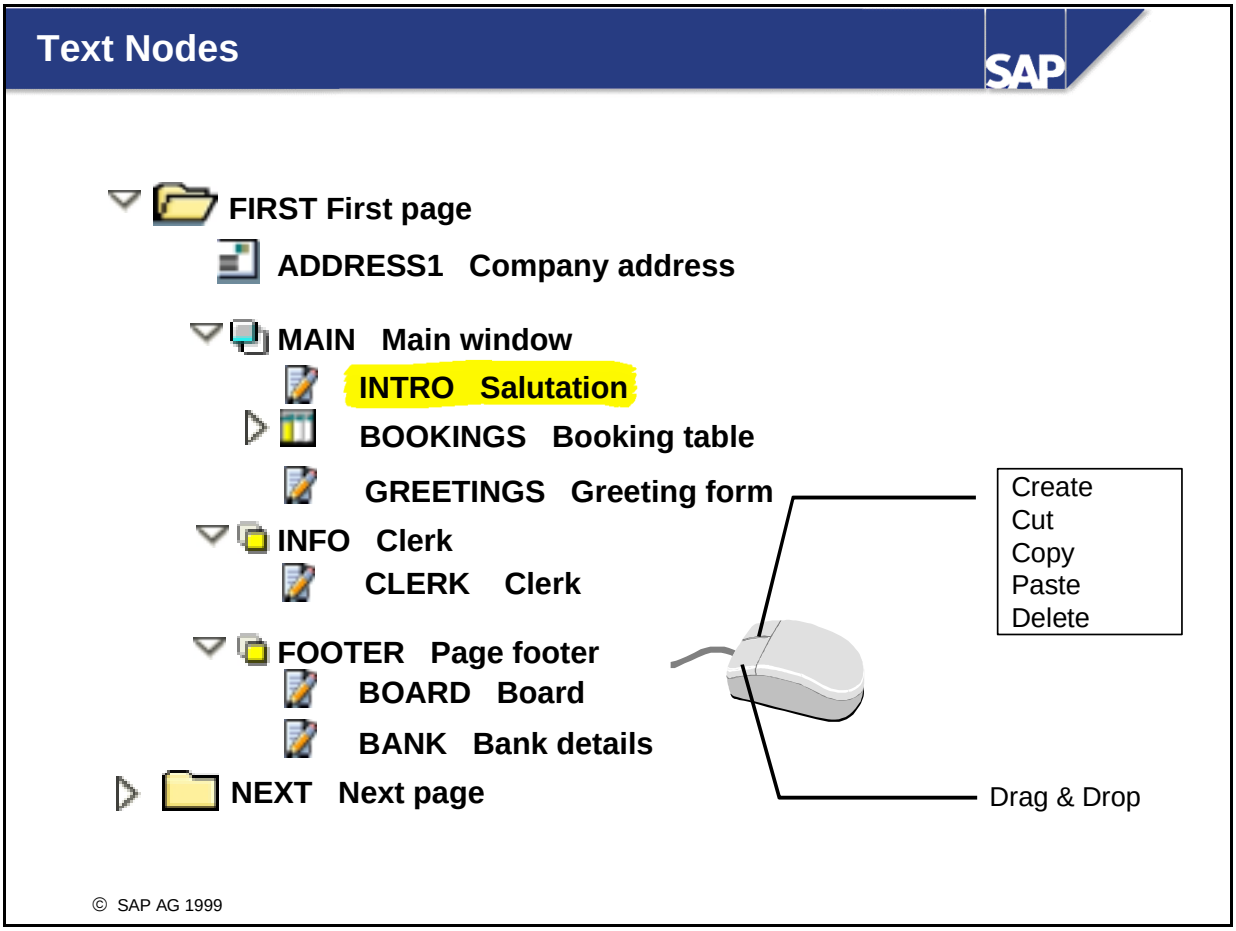
At the conclusion of this unit, you will be able to:

- **Create text nodes**
- **Create addresses**
- **Create graphics**

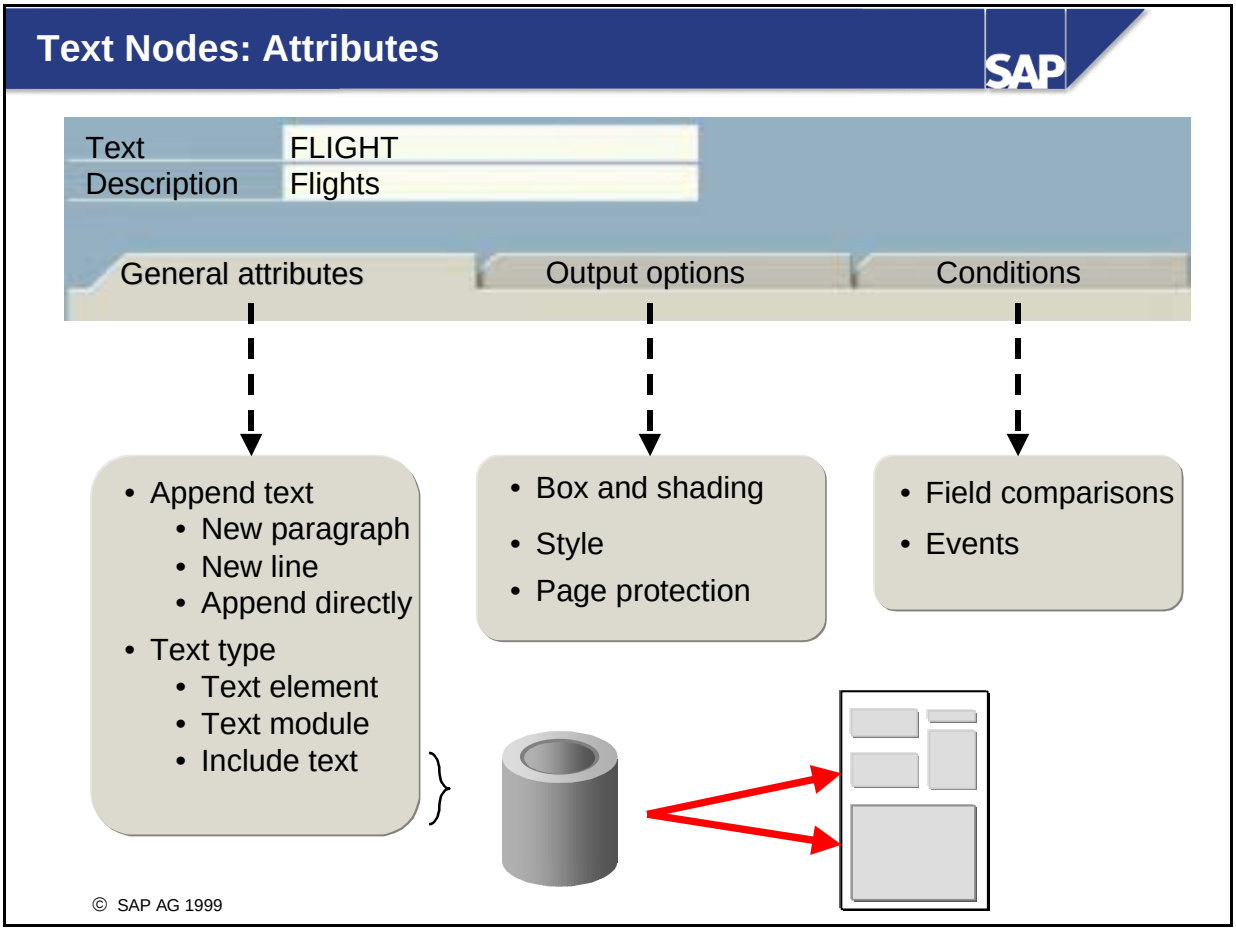
Texts: Topic Objectives

At the conclusion of this topic, you will be able to:

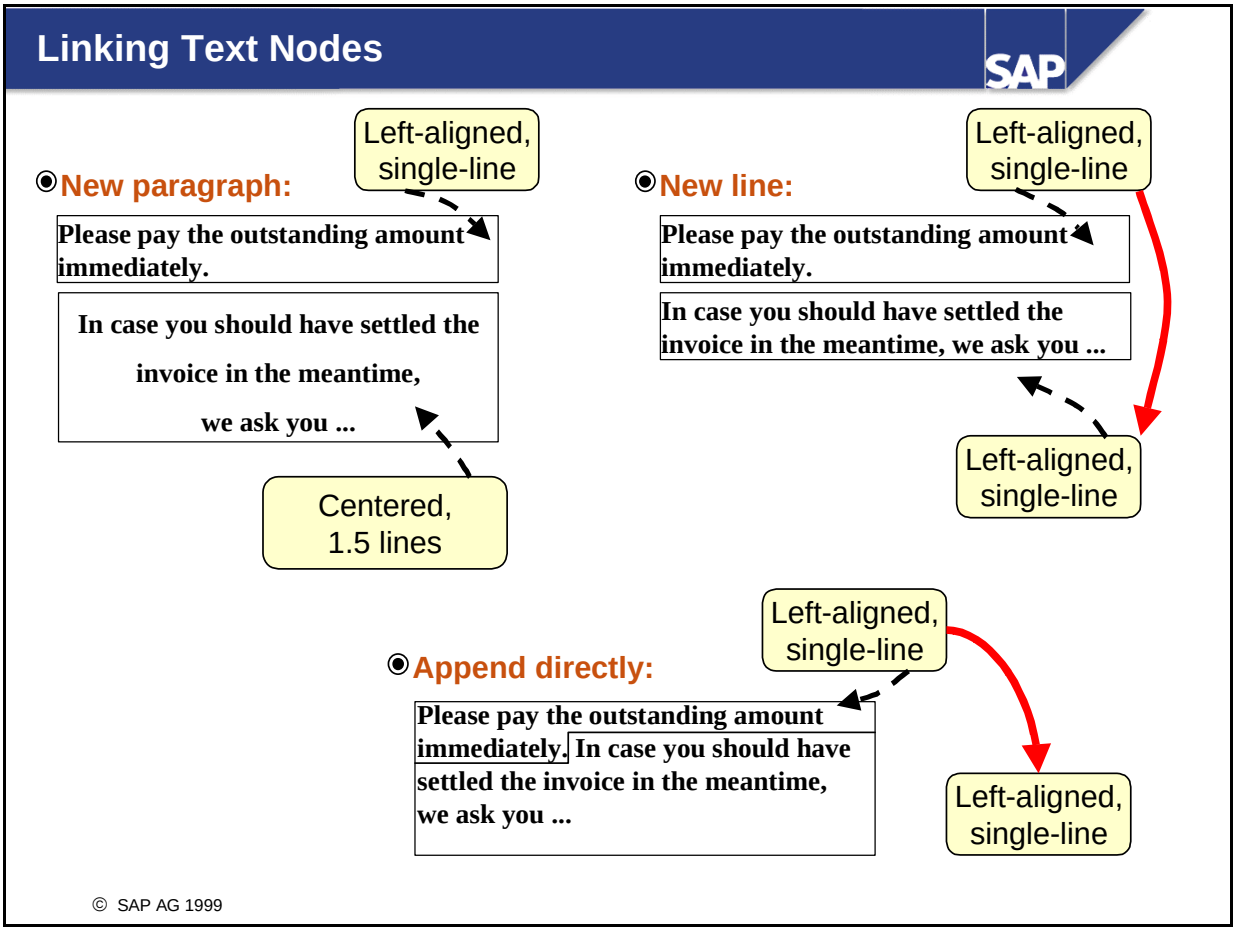
- **Create text nodes**
- **Decide which text node type (text element, text module, include text) you want to use**
- **Insert fields using the field list**
- **Format fields**



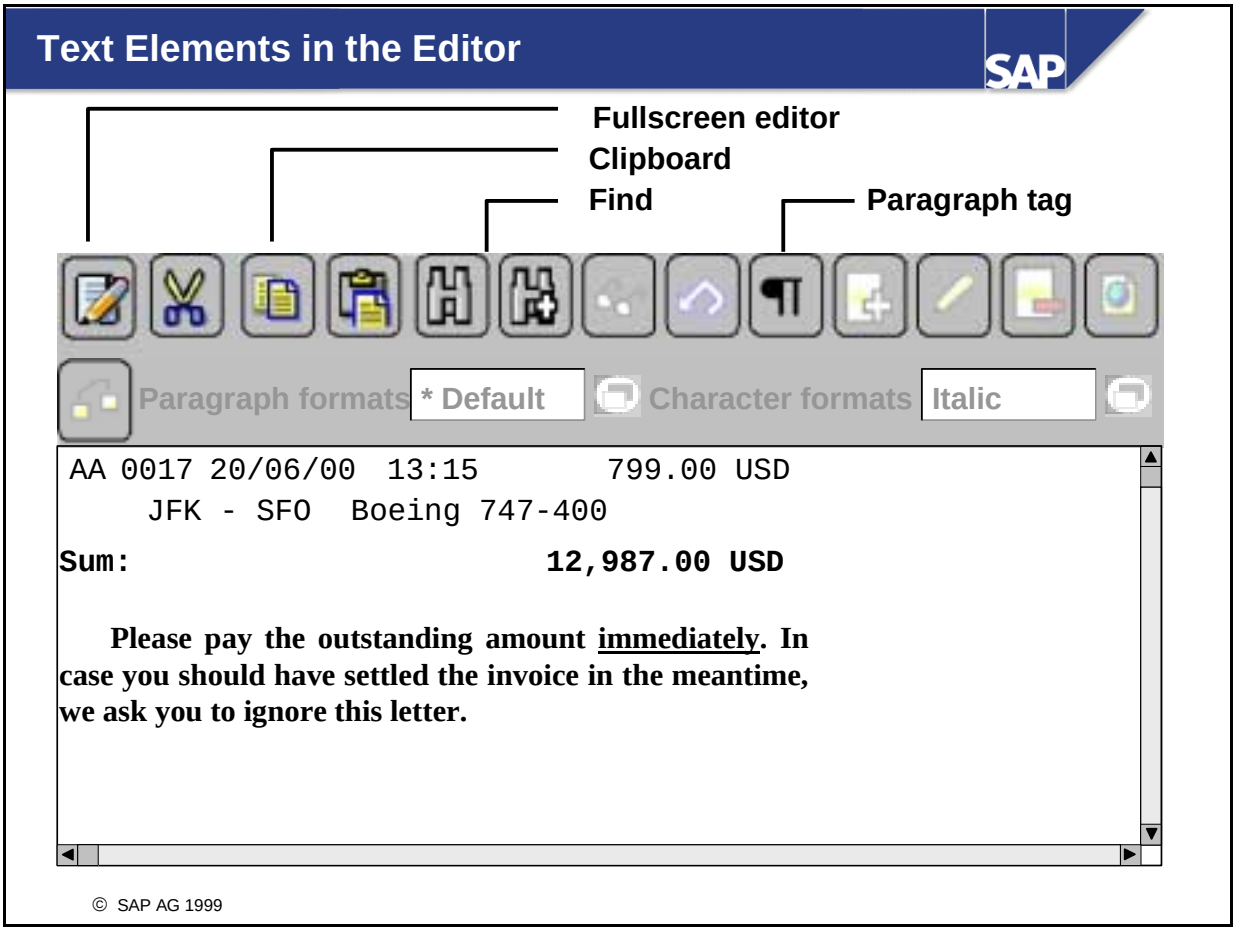
- You output all texts in the form using text nodes. (The only exception to this is address nodes.) Text nodes are either subnodes of windows or subnodes of subnodes, such as tables or templates.
- You can use the context menu (right mouse button) for existing text nodes. The context menu allows you, for example, to create another text node directly after a text node, that is at the same level. You can alternatively create text nodes using the context menu of the higher-level node. In this case, the new text node is inserted as the top subnode of that node.
- The same functions are available through the menu *Edit → Node*.
- Text nodes cannot have subnodes of their own.
- Note that text that does not fit into subwindows is truncated. It is only in the main window that the remainder of the text is processed on the next page.



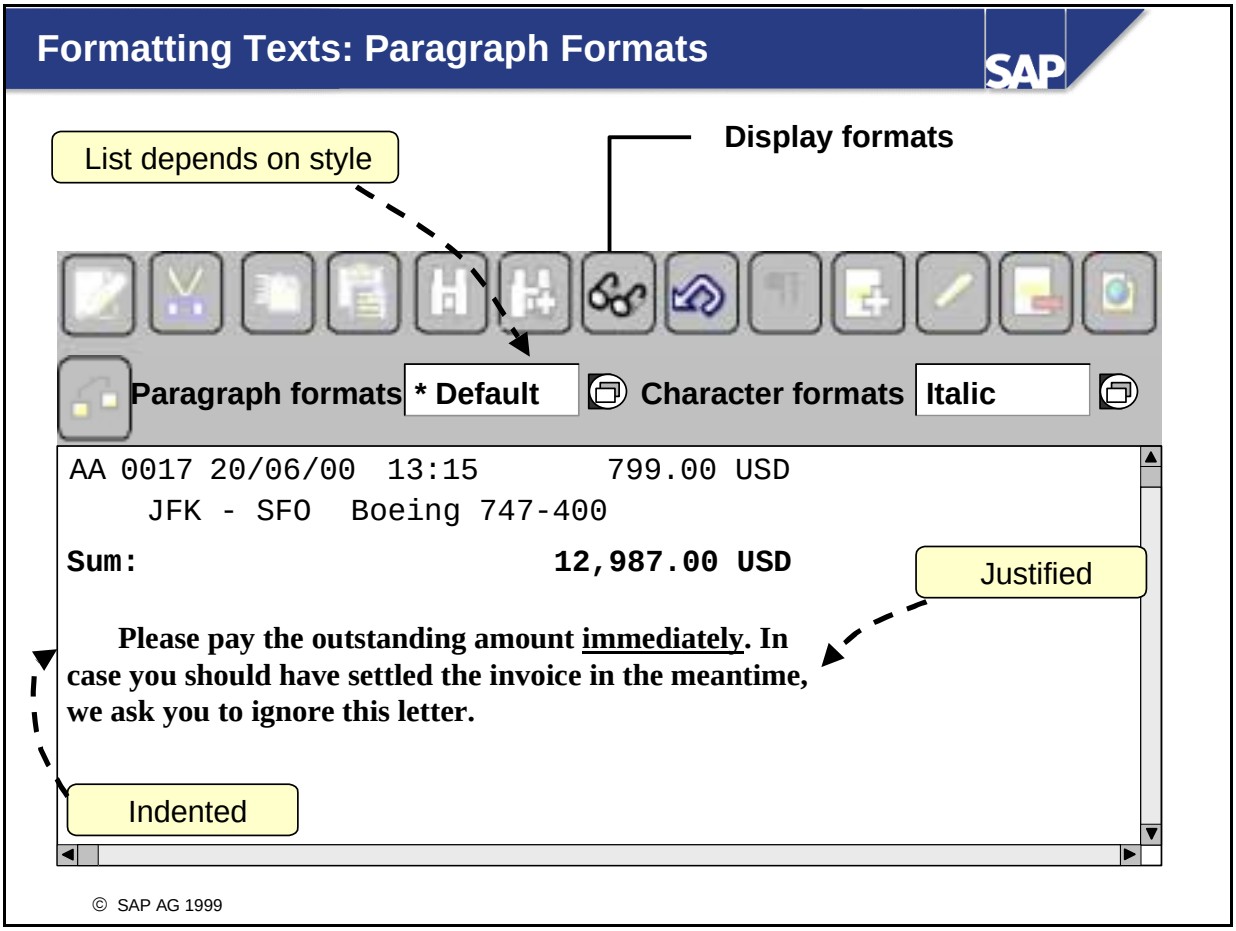
- If you select a text node in the navigation tree or in the Form Painter, its attributes are displayed on the maintenance screen.
- The three tabs are similar to those for windows.
  - *General attributes:*
    - The text type allows you to control whether the text should be saved and edited within 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 successive text elements should be combined. (See next slide).
    - The remaining fields you see on the tab vary depending on the text type.
  - *Output options:*
    - Everything you learned about the position and size as well as boxes and shadings for windows is also applicable to text nodes.
    - Besides, you can *assign a style* to the text node. A style is a collection of different character and paragraph formats. (See Unit 9 - *Smart Styles*.)
    - 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:* Refer to the conditions for windows and to Unit 7 - *Flow Control*.



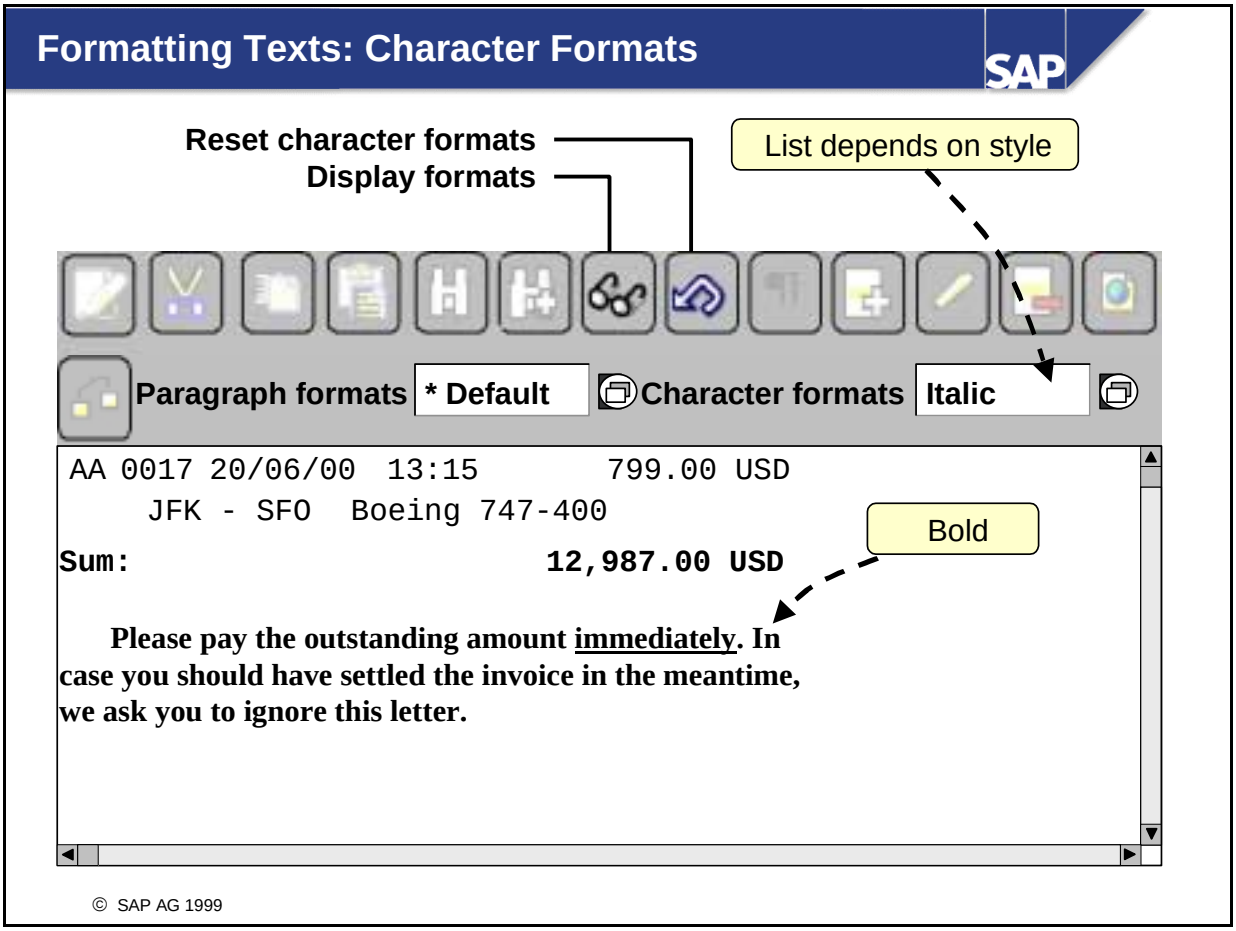
- On the *General attributes* tab you determine how two successive text nodes should be combined in the same window.
  - If you choose *New paragraph*, the text of the second node begins in a new line based on the paragraph format that you specified for this paragraph. This means that the two text nodes are completely independent of each other. (*Return* 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-Return* 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 what you select.



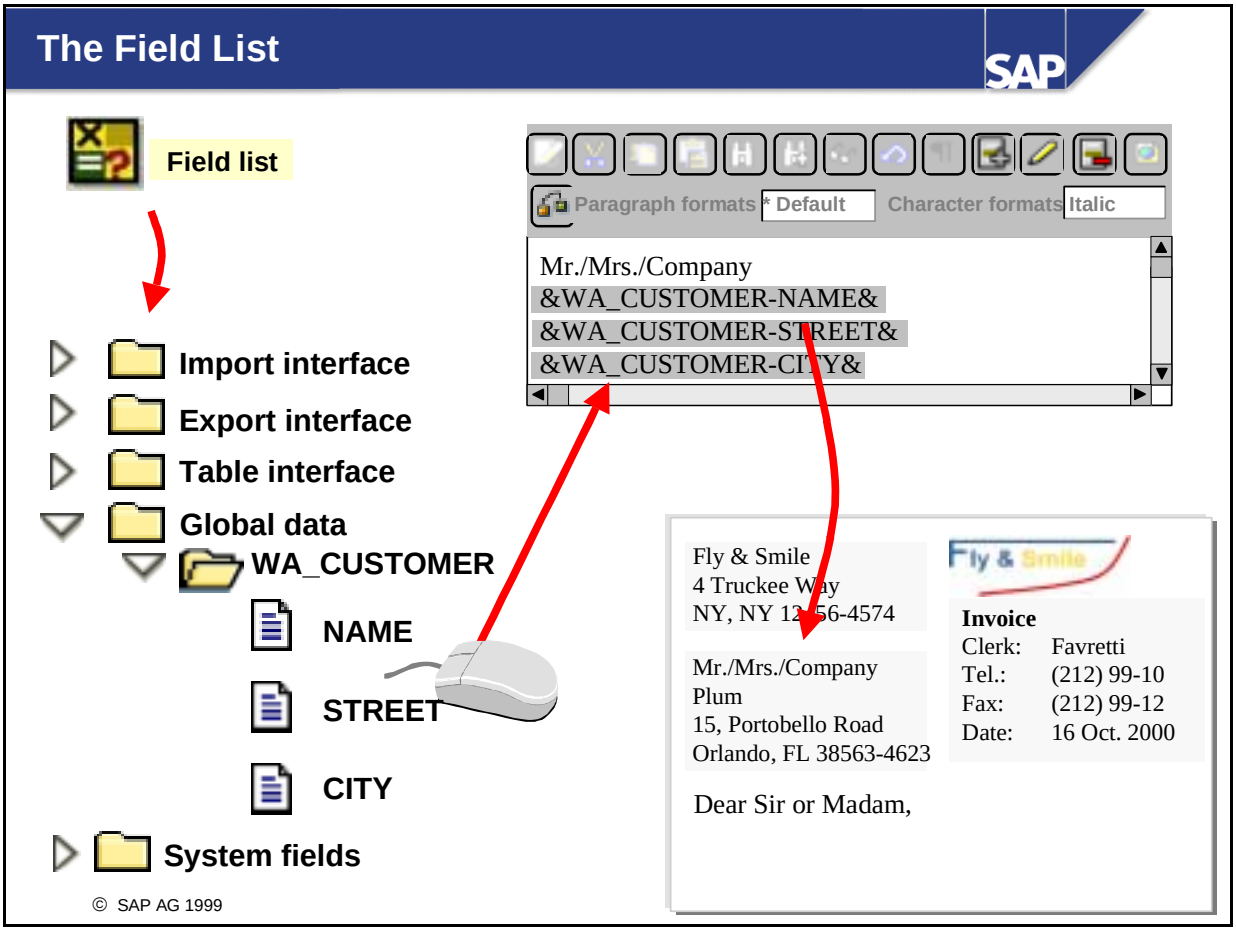
- If you choose *text element* as the text type, the inline editor is displayed on the *General attributes* tab. You can enter text in the usual way as you would in any common word-processing system. You can also switch to fullscreen mode by choosing the *Text editor* pushbutton.
- You can use the clipboard by selecting text blocks with the mouse and then clicking the *Cut*, *Copy*, or *Paste* pushbutton. This way you can copy text between different windows or forms.
- Lines in text nodes are broken automatically depending on the window width. However, you can use the *Return* key in the editor to create a new paragraph that may then have a different format than the preceding one. *Shift-Return* allows you to define a line break within a paragraph.
- The pushbutton *Paragraph tag on/off* allows you to determine whether you want to display all nonprinting characters (blanks, tabulators, paragraph marks, line breaks).



- You can format selected text. This text is then displayed as it looks like when you print it (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 of the editor. A paragraph format is a collection of format settings, such as tabs, type of justification, and so on (see Unit 9 - *Smart Styles*). The paragraph formats available for selection in the list depend on the style you have chosen. If you have entered a style for several nodes (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 default paragraph format of the style is used.
- Starting with R/3 Release 4.6C, support package SAPKB46C09, the system automatically displays the format set at the current cursor position in the paragraph and character format lists. You therefore need the *Display formats* function only if you want to obtain detailed information on the format or if text has been formatted using several character formats. See also OSS note 327636.



- You can also assign one or more character formats to 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 as for the paragraph formats. The pushbutton *Reset character formats* 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 used at a specific cursor position, click the *Display formats* pushbutton. The system displays a dialog box on which you see the formats. To get more information on a specific paragraph or character format, double-click the respective format.

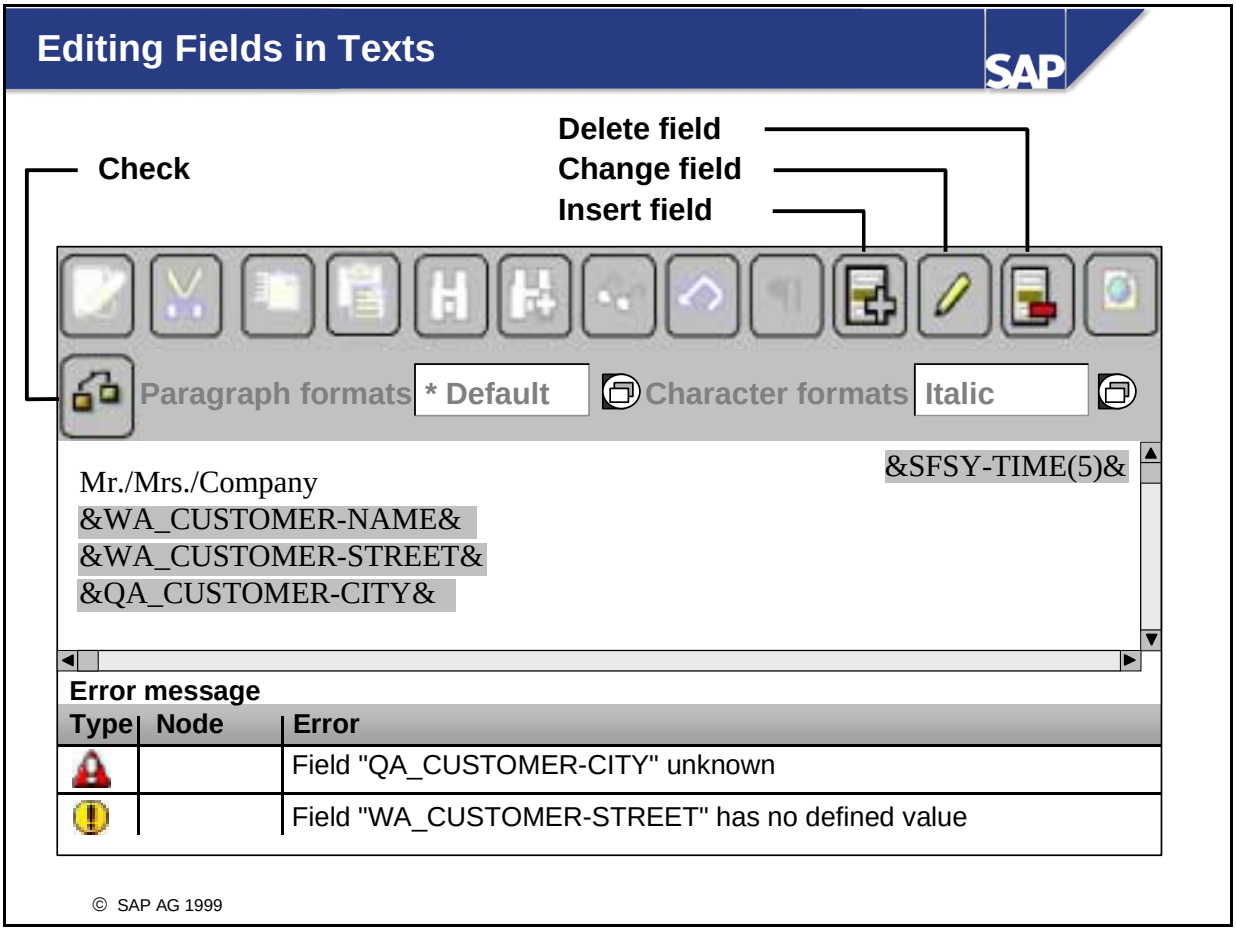


- Normally, your form contains not only static text but also variable data referred to as fields which are read from the database at application runtime or are entered by the user.
- The simplest way to insert fields is to use the field list. You can hide or show the field list by clicking the corresponding pushbutton or by choosing *Utilities → Field list on/off* from the menu.
- The following types of fields are available for selection:
  - All fields known as import, export or table parameters to the form through the form interface (which means they come from the application program)
  - All global data and field symbols you have created in the form in the global definitions
  - System fields which are filled automatically during program execution. See next slide.
- 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 you must click the triangle icon to the left of the corresponding folder to be able to access the individual subfields.

System Fields

&SFSY-DATE&	Date
&SFSY-TIME&	Time
&SFSY-PAGE&	Number of current print page
&SFSY-FORMPAGES&	Total number of pages in current document
&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

- 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 (for example, Arabic, or numeric) and the mode (increase, initialize, leave unchanged) on the *General attributes* tab of the page node.
  - FORMPAGES: Total page number for current document. You can print the page number in the form 'Page x of y', for example.
  - JOBPAGES: Total page number of all documents contained in the current print request.
- WINDOWNAME: Name of current window
- PAGENAME: Name of current page
- MAINEND: Is set if the processing of the main window on the current page ends.

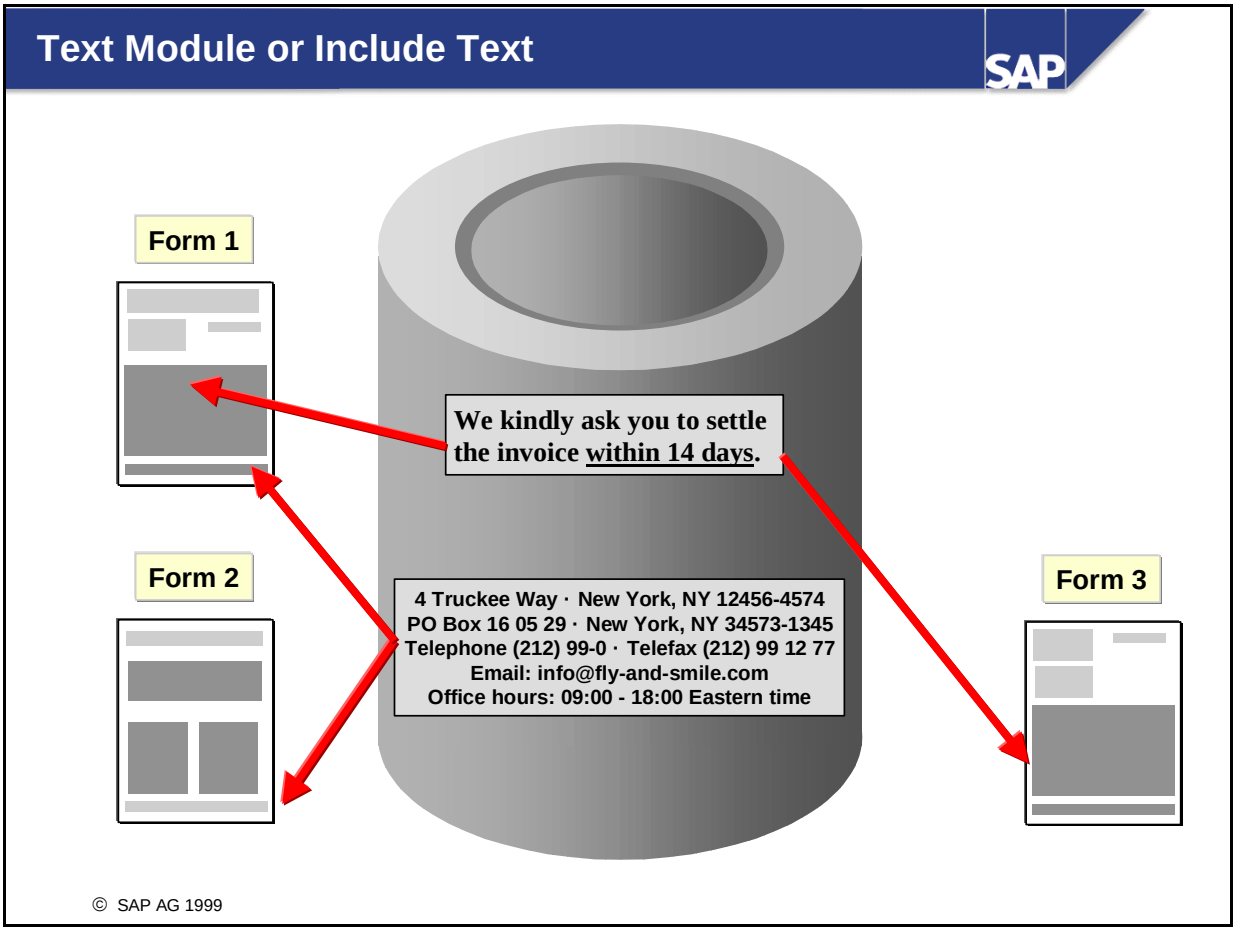


- You can also use the pushbutton *Insert field* to add fields to your text element. Enter the name of the field enclosed in ampersands. This also allows you to access the ABAP system fields of the SYST table, such as &sy-uname& (user name). Field names are not case-sensitive.
- So that fields can be distinguished from normal text, they are grayed out. Besides, 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 (see next slide).
- To ensure that your field entries are correct, run a check by clicking the corresponding pushbutton in the editor. The system notifies you of any errors that may exist (exclamation mark in a red triangle). You can then not 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 this is the case, the system issues a warning (exclamation mark in a yellow circle) but you can nevertheless activate and test the form. Fields which are initial at application program runtime will be ignored. For more information see the documentation on *Checking and Testing Smart Forms*.

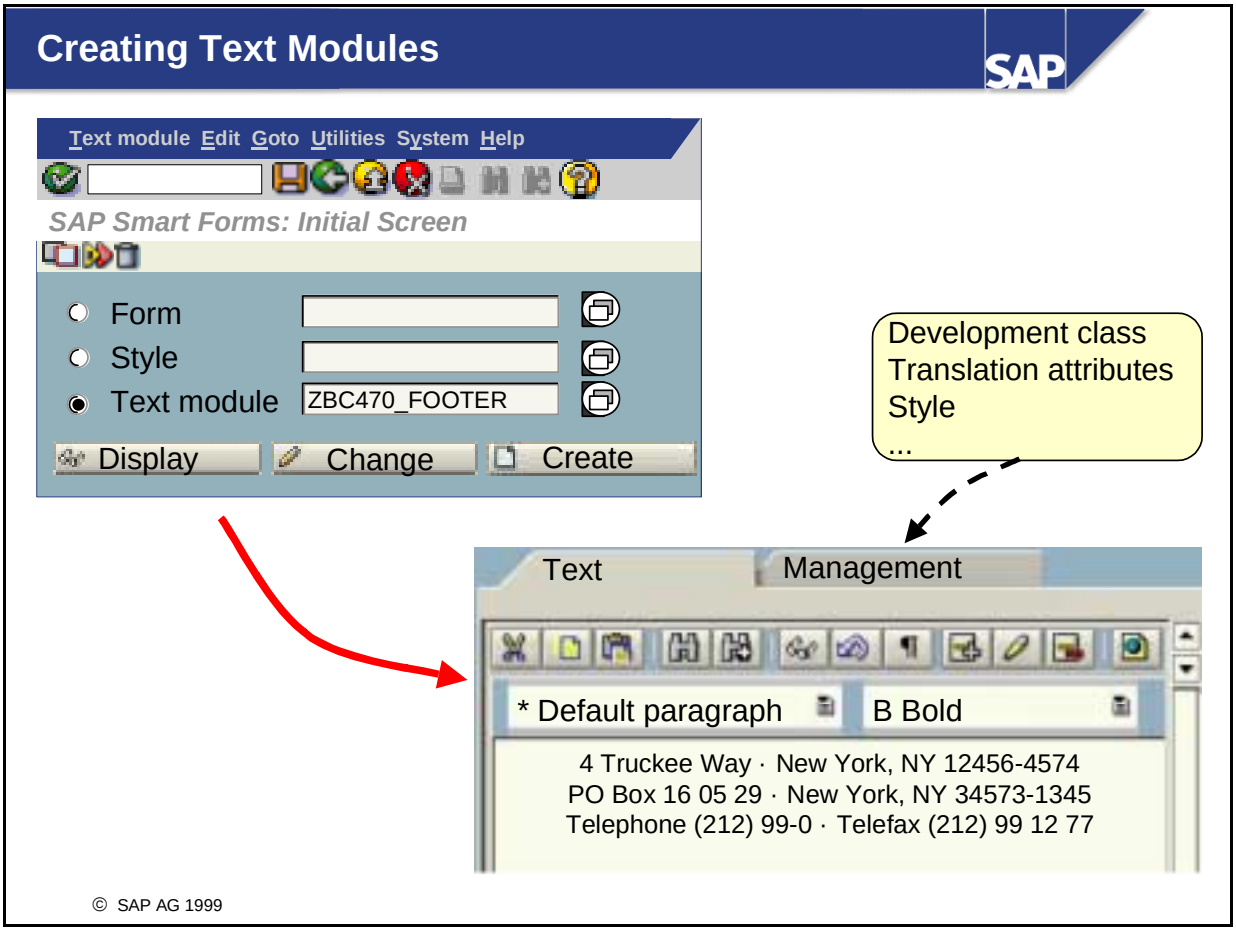
Formatting Options

&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 formatting (here: 8 digits, 2 after the comma)
&field(T)&	Suppress thousands separators
&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

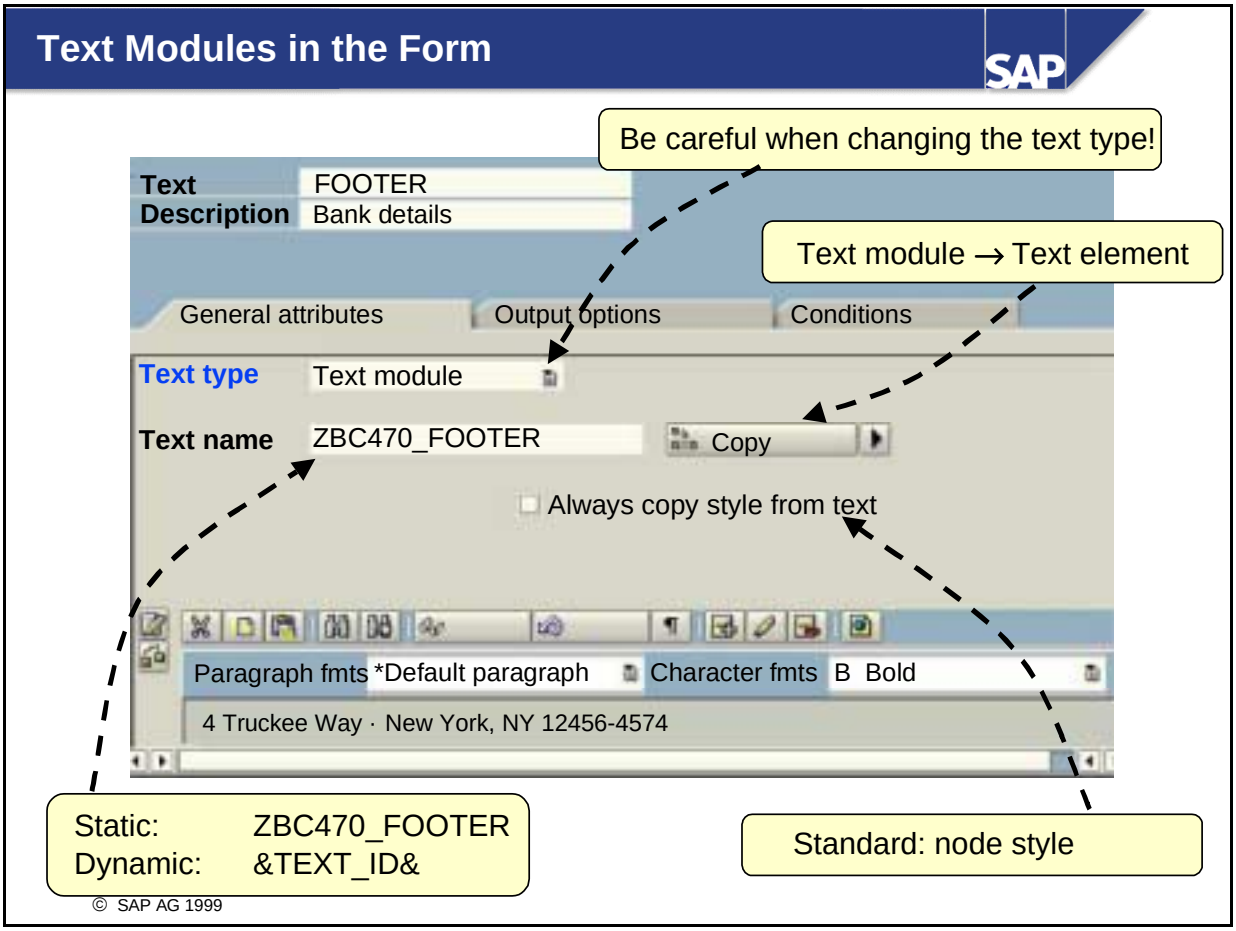
- Formatting options allow you to adjust the value of a field before it is output. You enter the option shortcut directly after the field name in **upper case**. Some of these options can be combined, for example +6(9) or (8.2).
- The formatting options are not appropriate for all data types of a field. (You cannot output numbers with an offset, for example). We distinguish between character and numeric fields:
- Numeric fields:
  - The closing blank is interpreted as a plus sign. To suppress it, use the formatting option S.
  - Evaluation sequence: (<Length>), +/- sign to the left of the number (<), Japanese date (L), suppress blanks (C), right-justified display (R), insert filler (F).
- 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).
- For information on other formatting options see the online documentation under *Output Options for Field Contents*.



- In some cases it makes sense to store texts not in the form itself but centrally in the database. The form then contains only a reference indicating which text should be used at application program runtime. This method provides the following advantages:
  - You need to create the texts only once and can then reuse them 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 and 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, since text already entered will 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.

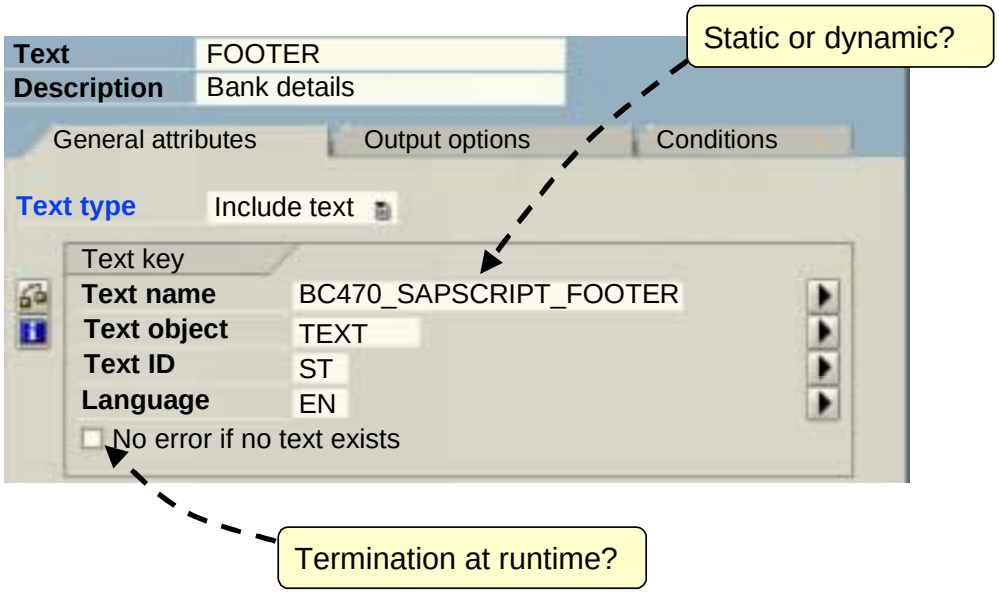


- You go to the maintenance transaction for text modules by choosing the *Text module* radio button on the SAP Smart Forms initial screen and then *Display*, *Change* or *Create* depending on what you want to do. From this screen, you can also copy, rename or delete existing text modules. To do this, 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 therefore begin with Y or Z.
- Since text modules, like SAP Smart Forms, are integrated with the R/3 transport system, you must assign them to a development class. You do this when you first save your form. You cannot change the development class assignment later on.
- 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. Please note, however, 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*.



- 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 known 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.
- The text of a text module is always grayed out in the inline editor of the Form Builder since you cannot change it directly from within form maintenance.
- If you determine the text module dynamically, its text cannot be displayed in the inline editor. Likewise, you cannot check from within form maintenance if the field is filled with a meaningful value at application program runtime. If the system cannot find the text module that you determined dynamically when the program executes, the current output request is terminated.
- If you choose the *Copy* pushbutton, the text of the text module is inserted as a text element into the form. This removes the reference to the text module. Any subsequent changes to the text module are not considered.
- By default, the system uses the style of the text node or of a higher-level node. You can also select *Always copy style from text* and override this setting for a text module.

Include Texts

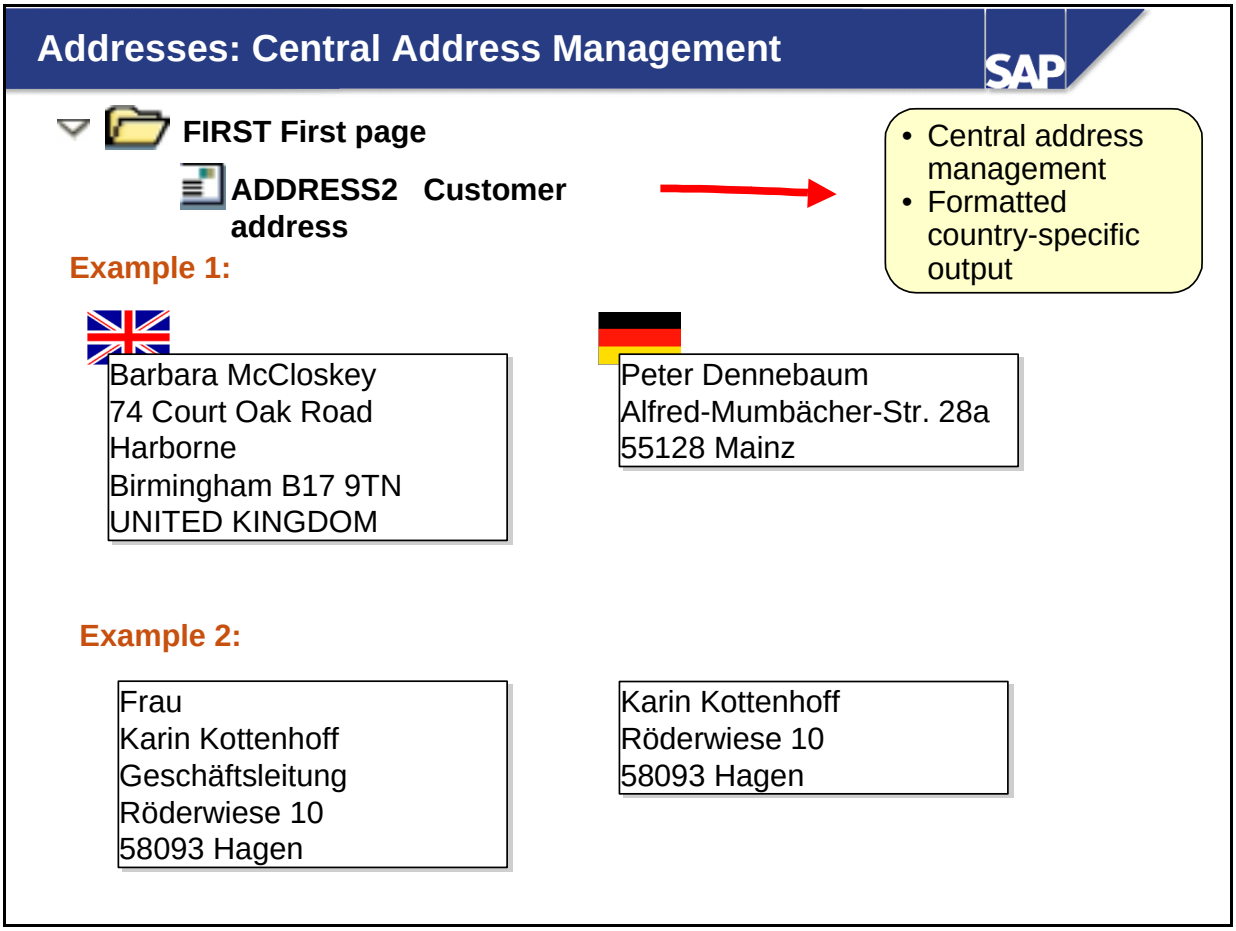


- You can also insert existing SAPscript texts. To do this, choose *Include text* as the text type of the text node and enter the necessary selection information in the *Text key* group box. You can also make these specifications dynamically (name of a field, enclosed in ampersands (&)).
- If you select *No error if no text exists*, the function module generated does not terminate if the text specified is not found at runtime.
- If you create new texts for use in SAP Smart Forms, you should always create SAP Smart Forms text modules rather than SAPscript texts. You do so for the following reasons:
  - You can neither maintain nor preview SAPscript texts from within 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.
- Please note that SAP Smart Forms ignore all SAPscript commands in SAPscript texts. If required, you must convert these commands into SAP Smart Forms logic.
- In Release 4.6C you cannot override the paragraph format of the SAPscript text in the *Paragraph formats* group box.

Addresses: Topic Objectives

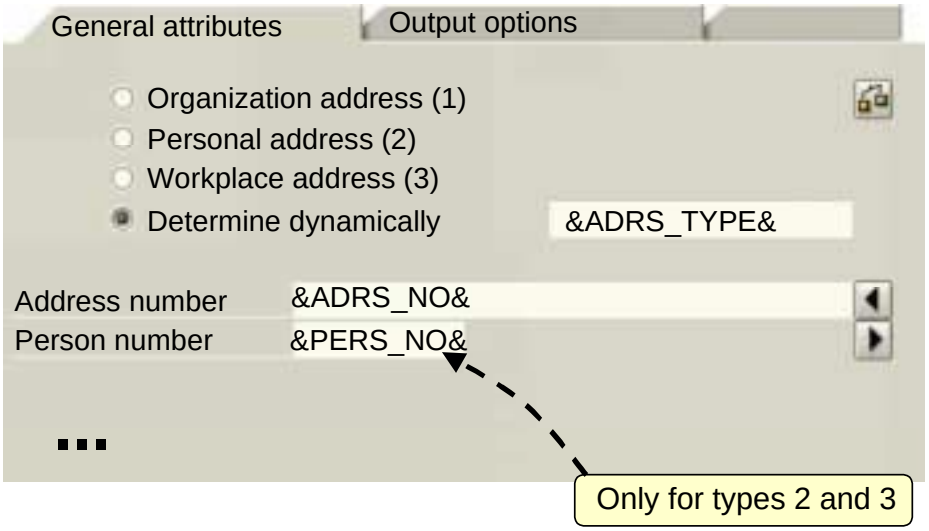
At the conclusion of this topic, you will be able to:

- **Create addresses of central address management as separate windows or as subnodes of a window**



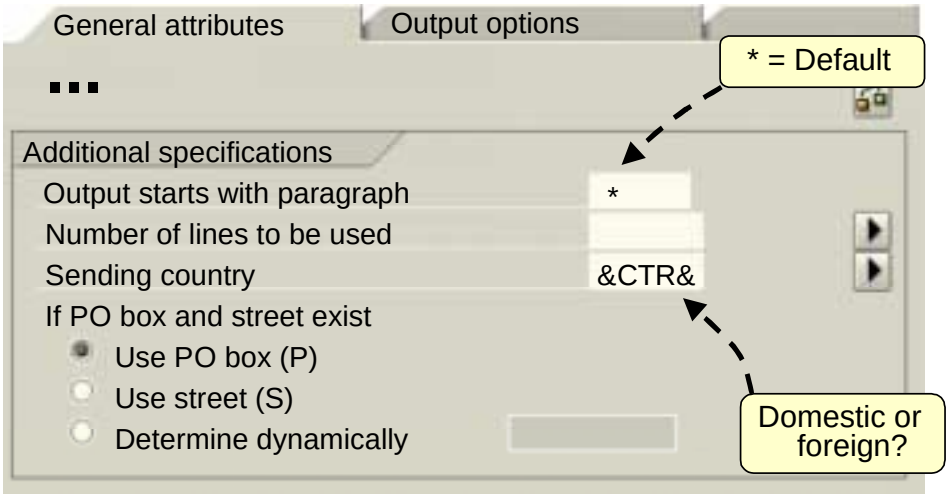
- Many applications no longer use their own tables for their address information but use central address management (CAM) instead. CAM allows addresses to be identified by means of numbers.
- SAP Smart Forms allow you to use CAM. There is no need to know the technical details of CAM or be concerned about the correct formatting of the addresses. 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 may not be output. For more detailed information, refer to 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 then 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, a table or a template.
- If you want to format addresses without CAM, you will have to create a program line node and call the function module ADDRESS_INTO_PRINTFORM.

Addresses: Attributes I



- You define the basic settings for the address node on the *General attributes* tab.
- Address type:
  - *Organization addresses:* Typical examples are ship-to addresses or company codes. These addresses are uniquely identified by their address number.
  - *Personal addresses:* Addresses of these types are assigned to one natural person and have other attributes such as a title. Since a person can have more than one address, you must enter both the address number and the person number for identification.
  - *Workplace addresses:* These are personal addresses in companies, that is, 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.
  - *Dynamically:* If you want to determine the address type at application program 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 CAM at runtime, the function module of the form terminates with an error message.

Addresses: Attributes II

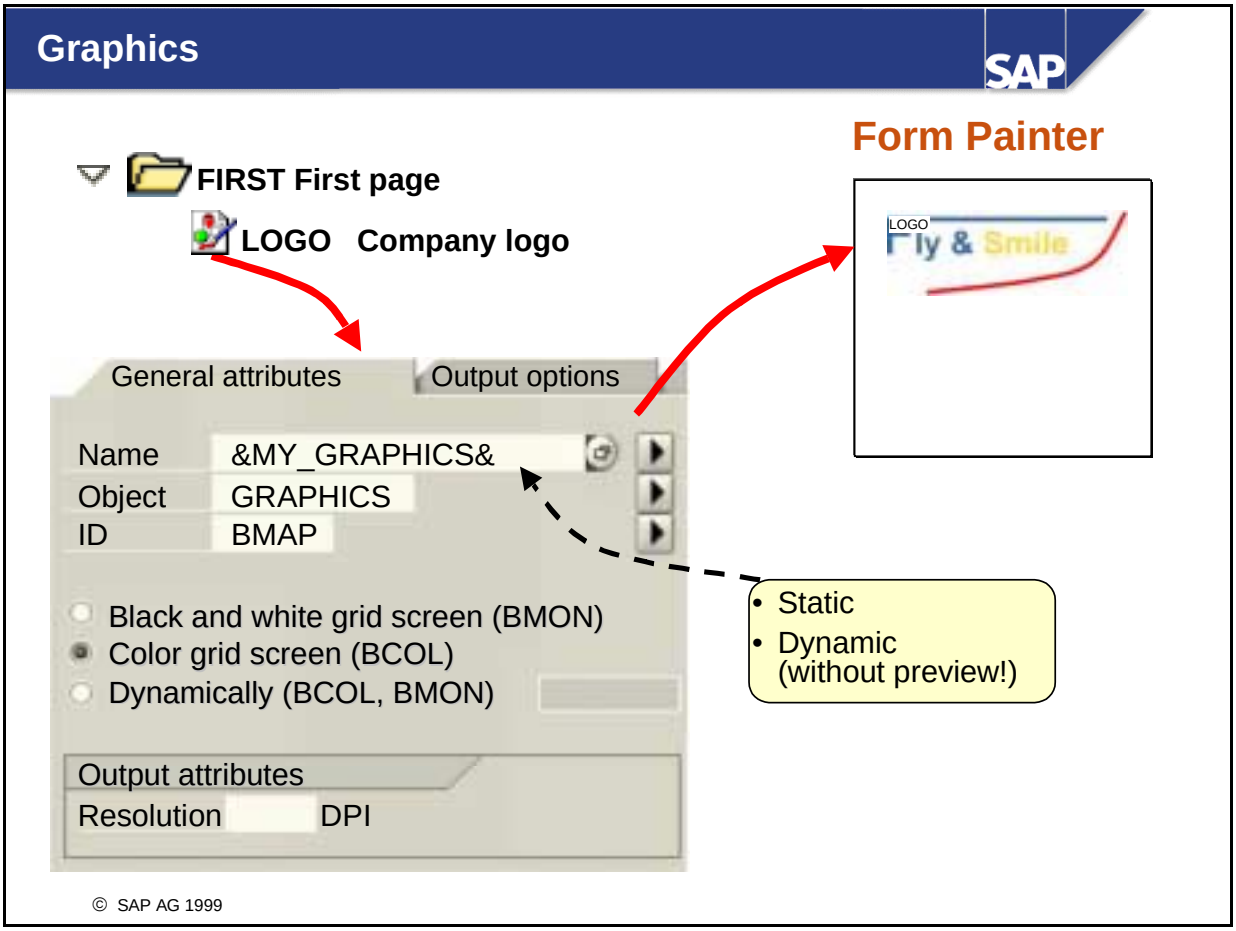


- In the *Output starts with paragraph* field, you set the paragraph format for the address node. This paragraph format must be defined in the style of the address node or, if you have not defined one in the output options, in the style of a higher-level node or the form. If you enter an asterisk, the system uses the default paragraph of the style.
- If in the field *Number of lines to be used* you set fewer lines than are required to output the full address, CAM suppresses address parts of minor importance, such as the salutation 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 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 national ones.
- For addresses that have both a PO number and a street address, you use the radio buttons in the group box to determine which one to use.
- Like other nodes, addresses have the *Output options* tab where you can determine the style, and the box and shading. Also on this tab (as in the Form Painter), you can set the values for the position and size of the output area - provided you have created the address node as a separate address window, that is, as a direct subnode of a page.

Graphics: Topic Objectives

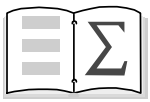
At the conclusion of this topic, you will be able to:

- **Create graphics as separate windows or as subnodes of a window**



- You can embed graphics not only as background pictures, but also as separate graphic windows (that is, as direct subnodes of a page) or as subnodes of a window. This requires that the graphic already exists in the system. See also the *Graphics Administration* slide in the appendix.
- You create graphic nodes like you create any other node by using the context menu (right mouse button) of the navigation tree 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 (provided you have not selected the *Placeholder for graphics* checkbox in the Form Painter settings). You can then easily position the graphic on the page using Drag & 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 define 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. Most importantly, 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 as a subnode, you must also determine its horizontal position on the *Output options* tab.

Texts, Addresses, and Graphics: Summary



You are now able to:

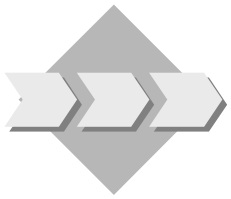
- **Create text nodes**
- **Create addresses**
- **Create graphics**

Exercises

Unit 4: Texts, Addresses, and Graphics

At the conclusion of these exercises, you will be able to:

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



Your task: Copy an existing invoice form that has windows but no text. Add this text.

Copy template for the form:	BC470_TEXTT
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

Perform the optional tasks only if you have time left at the end of the exercise.

1. Copy template

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

2. Create text nodes

   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.
   - GREETINGS – The closing lines

3. Use field list for customer address

The field WA_CUSTOMERS that is filled by the application program contains all-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, CITY, and, if required, REGION. (We shall not care about formatting the address in accordance with ISO standards at this point).

4. Define shading for window

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

5. Create text module

   5-1 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.

   5-2 Save the text in your development class.

   5-2 Embed this text module in the window FOOTER of the form.

6. Company logo

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

7. **Optional:** Insert include text

   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.

8. **Optional:** Determine page numbers

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

9. Test result

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

Solutions

Unit 4: Texts, Addresses, and Graphics

1. Copy template

   See exercise of the previous unit.

2. 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 & 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.

3. 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*.

   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 & Drop. Add some paragraphs.

4. Define shading for window

   Select the window INFO (by double-clicking it in the navigation tree or single-clicking it in the Form Painter). On the *Output options* tab, choose a gray value of 10 % in the *Shading* group box.

5. Create text module

   5-1 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 that is initially assigned to each new text module does not use small font sizes. You therefore must assign another style, that is, BC470. You do this on the *Management* tab. 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.

   5-2 Save your text module by clicking the disk icon. Enter your development class.

   5-3 Now go back to the other session and choose *Create -> Text* from the context menu of the window FOOTER. On the maintenance screen, change the name to read BANK and enter a description. On the *General attributes* tab of BANK, change the text type into *Text module*. Choose *Yes* in response to the confirmation prompt. The *General attributes* tab changes so that you can enter BANK in the *Text name* field. Press *Return* - this should display the text module in the editor. You cannot directly modify the text module form within the SAP Form Builder.

6. 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.

7. **Optional:** Insert include text

   From the context menu of the window ADDRESS2, 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 into *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.

8. **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 (see Exercise 3) choose *System fields -> SFSY*. Then drag the fields PAGE and FORMPAGES into the editor of the text node. Add the text "Page ... of".

9. Test result

   Activate your form by choosing *Form -> Activate* from the menu.

   To test your 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 have entered too few text lines 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.

Data in Forms

BC470 Form Printing with SAP Smart Forms

Data in Forms: Contents

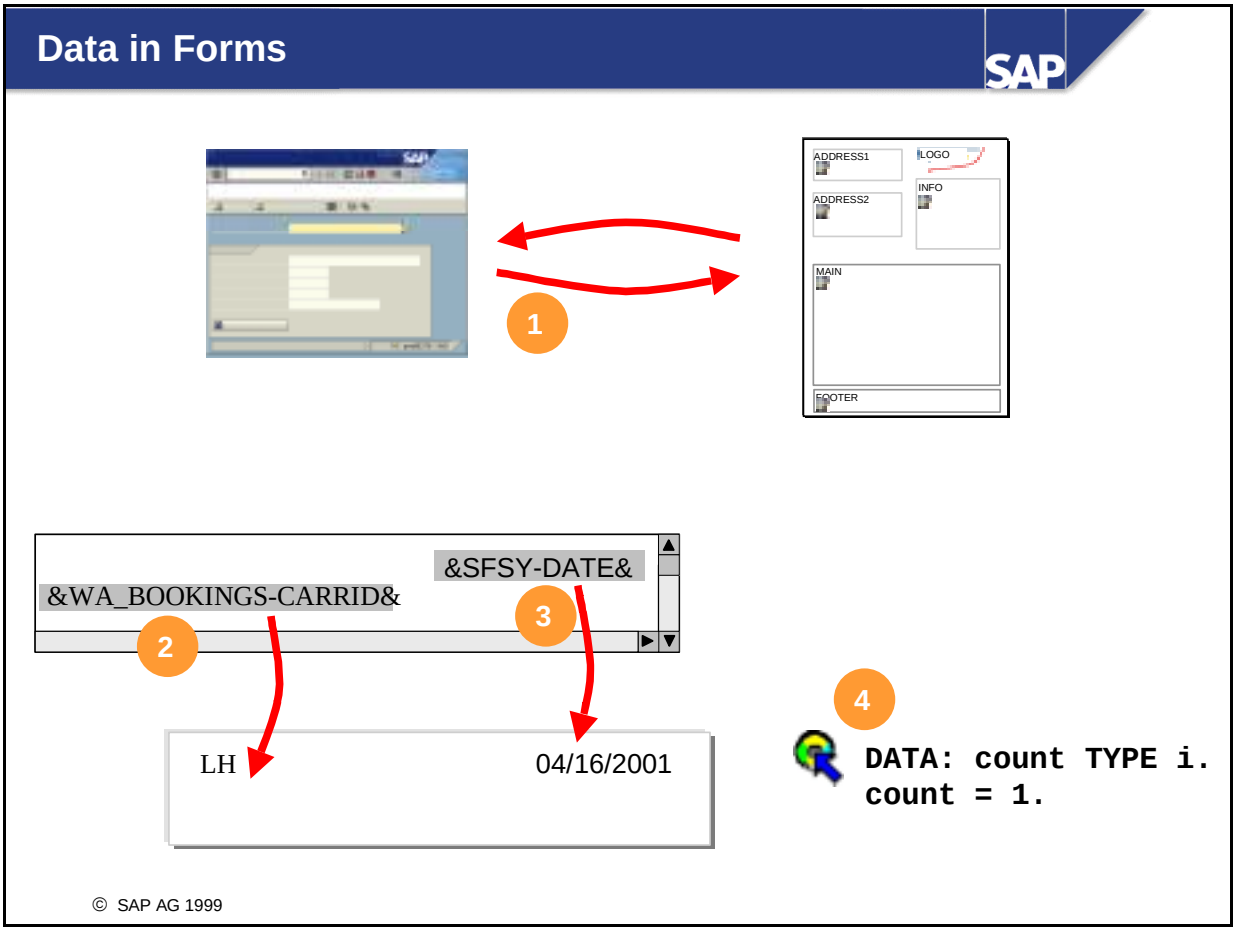
Contents:

- **Integrating data into forms which is only known at runtime**

Data in Forms: Unit Objectives

At the conclusion of this unit, you will be able to:

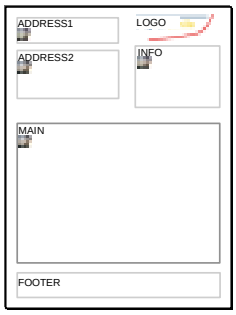
- **Define the interface of a form**
- **Create global data and types**
- **Explain the basic principle of global initialization and of field symbols**



- You have seen several times that you always need fields (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. (See slide *System Fields* in Unit 4 - *Texts, Addresses, and Graphics*)
  4. Local data that you create in program lines 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.)

Interface Parameters I

```
CALL FUNCTION <SMART_FORM_FM>
  EXPORTING ...
  IMPORTING ...
  TABLES ...
  EXCEPTIONS ...
```

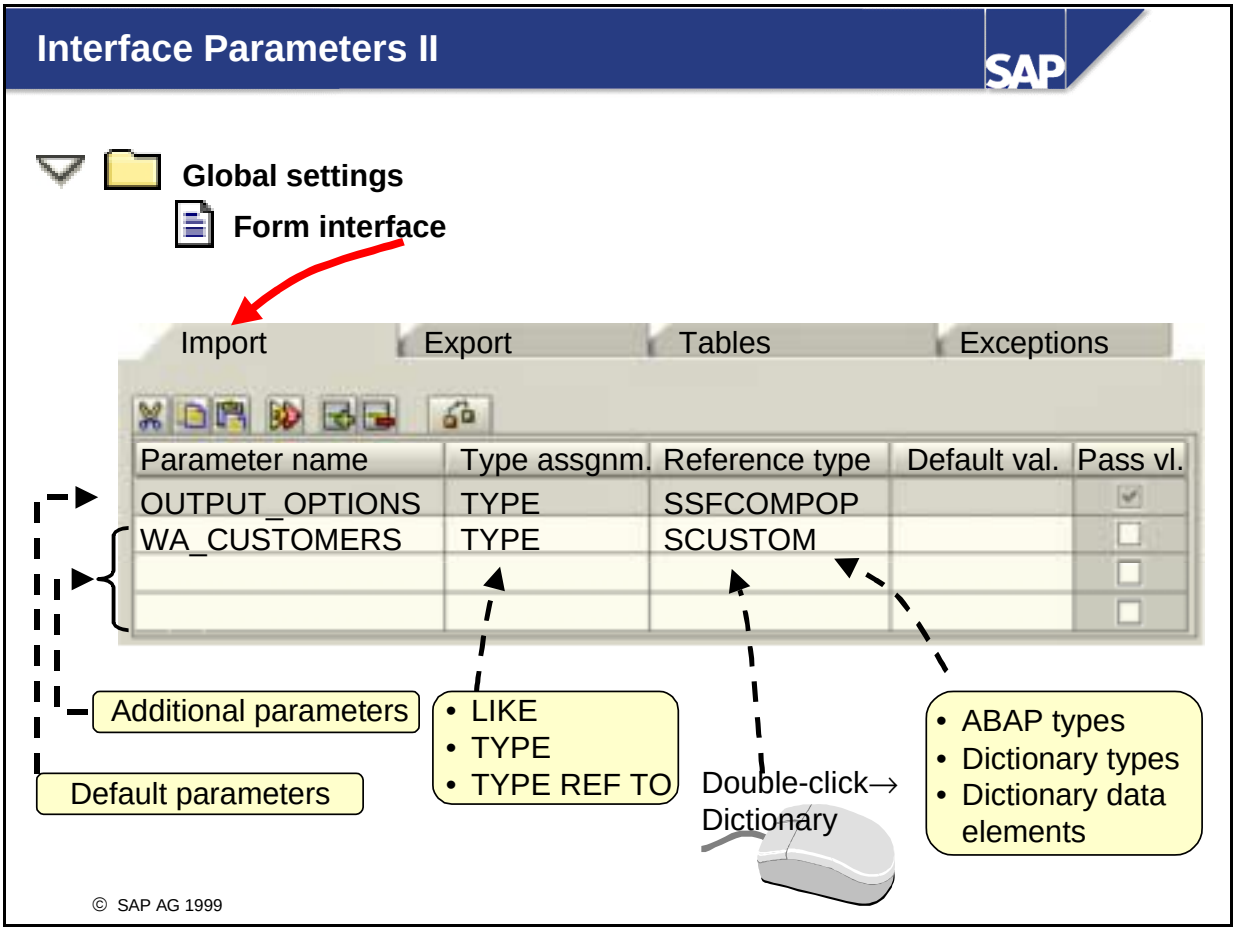


Global settings

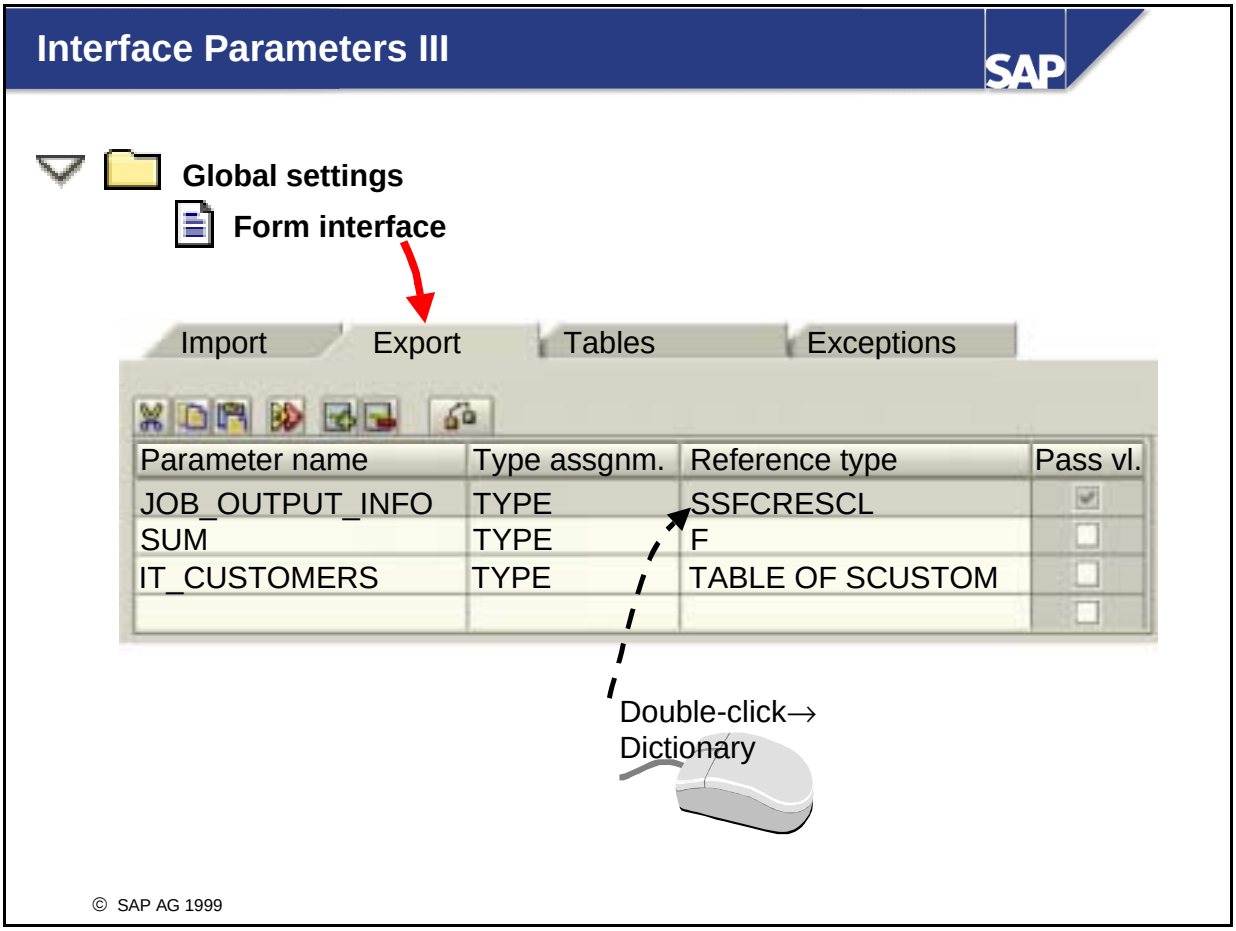
Form interface

Import | Export | Tables | Exceptions

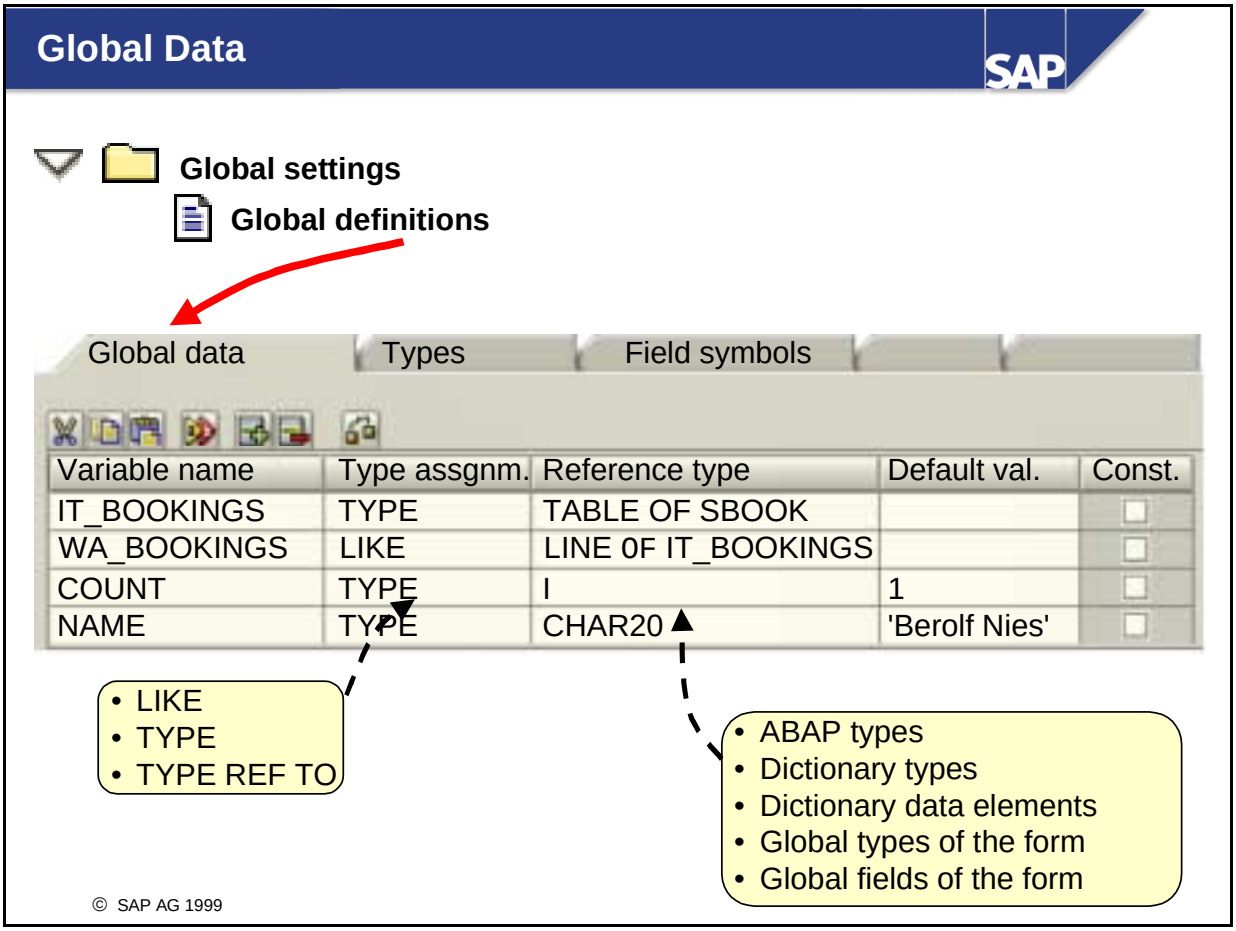
- If an application program calls an SAP Smart Form (or to be more precise: the generated function module of the form), there must be a way for the program to communicate with this function module. Data must be passed to the form and be returned by the form to the calling program. All data is exchanged through the interface of the form. 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



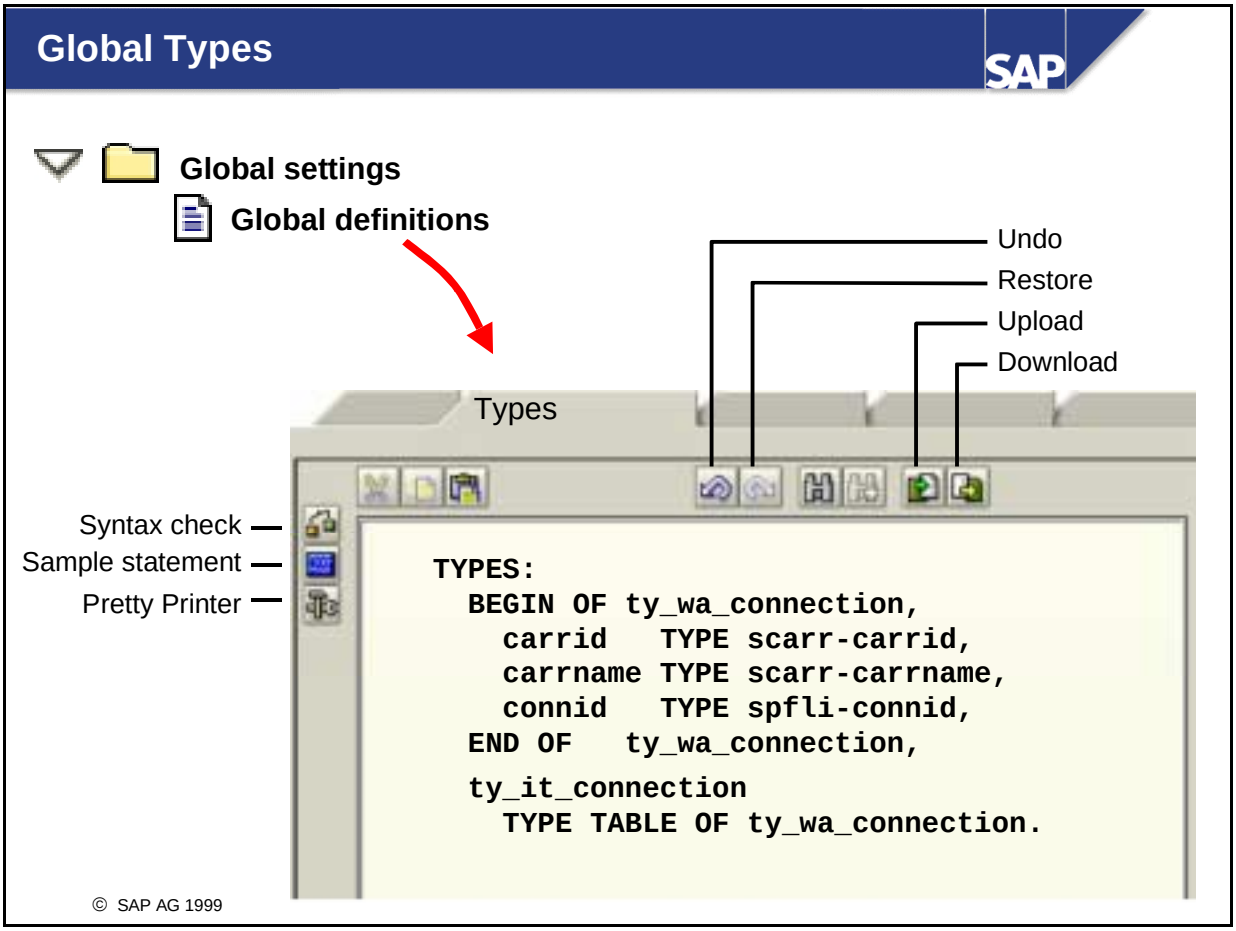
- *Import parameters* are read from the application program. There is a number of default parameters and an undefined number of others of any type. The default parameters - including the printout options - are covered in Unit 8 - *Integration into Application Programs*. Their fields are not ready for 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, blanks 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 *reference types*:
    - ABAP types (C, N, D, T, X, I, P, F, STRING, XSTRING)
    - Dictionary types
    - Dictionary data elements
  - For more information refer to the ABAP documentation on the keyword `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.
- If you check *Pass value*, the system passes a copy of the parameter to the form and not the value itself.



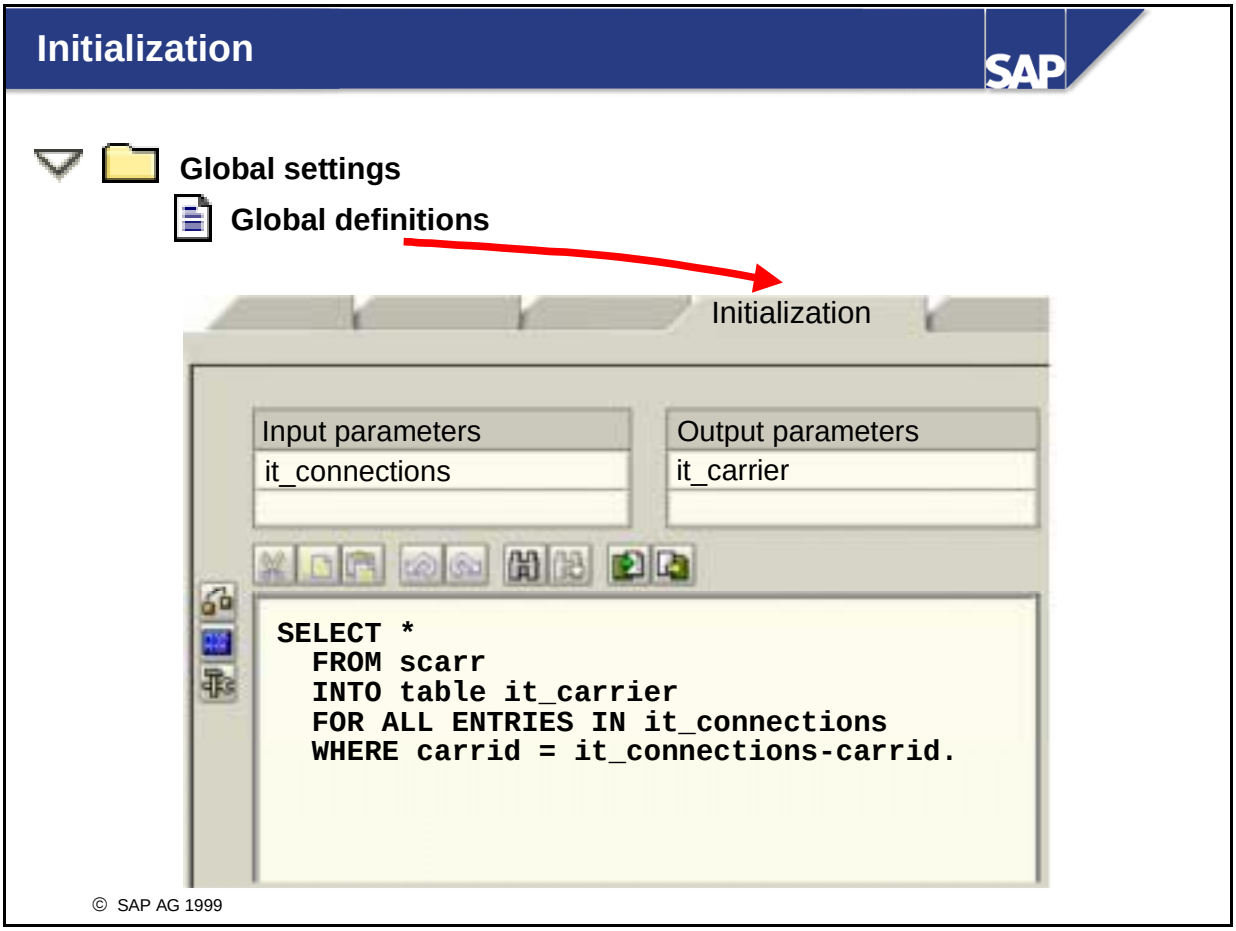
- *Export parameters* are returned to the application program. Again, there is a number of default parameters and an undefined number of others of any type. What we said about the *parameter names* and the *type assignment* of import parameters is also true for export parameters.
  - If you check *Pass 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 in the form, the original values of the application program are modified **directly**. If form processing terminates for whatever 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 tables to the form. You must type the table with reference to a non-nested table 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.



- 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 *reference types* for typing the global data provide the same options as the interface parameters and additionally:
  - Global types of the form (see next slide)
  - 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. In this case you must specify an initial value that you must not change in the form.
- If you have entered a Dictionary reference type or a data element, a double-click on the type or element takes you to the Dictionary where you can obtain information on the definition of the type.
- Check your entries with the *Check* pushbutton.

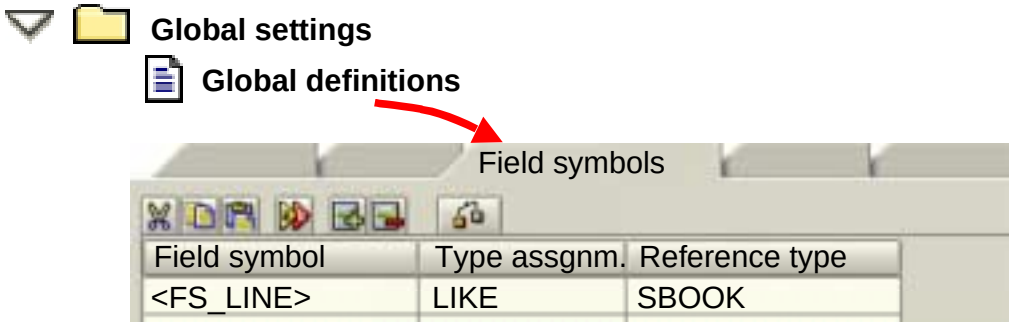


- The *Types* tab of the global definitions contains an ABAP editor that you can use to define types. You can then use these types in the entire form. In particular, you can type the global data with reference to these types.
- Types are defined with the ABAP statement TYPES.
- For more information refer to the ABAP documentation on the keyword TYPES.



- The *Initialization* tab of the global definitions allows you to enter ABAP code of your choice that you want to execute before the first page is processed. In particular, you can assign values to global data before the actual form formatting process begins.
- If you use data of the form interface or global data, you must explicitly specify this data as input or output parameters. This is necessary to allow the data flow analysis of the form check to determine if the fields have a defined value when they are to be output.

Field Symbols

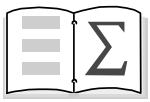


Processing an internal table:

With work area **With field symbol**

- In the global settings, you can create field symbols. The syntax corresponds to the ABAP command `FIELD-SYMOBLS`. This means you have to choose a name with an opening and a closing angle bracket, a typing assignment (`TYPE`, `LIKE`, `TYPE REF TO`) and a reference type.
- Field symbols are placeholders for data objects. To be more precise, they are dereferenced 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. Runtime errors or wrong data assignments may be the result. You should therefore use field symbols only if other ABAP statements do not provide an adequate solution for implementing the operation you want.
- You could use field symbols in loops over internal tables. The ABAP command `LOOP AT <itab> INTO <workarea>` copies each row 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 the program's runtime than work areas. SAP Smart Forms support field symbols for loops and tables.
- For more information refer to the ABAP documentation on `FIELD-SYMBOLS`.

Data in Forms: Summary



You are now able to:

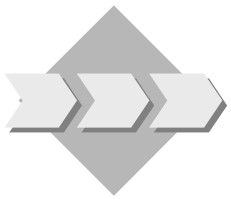
- **Define the interface of a form**
- **Create global data and types**
- **Explain the basic principle of global initialization and of field symbols**

Exercises

Unit 5: Data in Forms

At the conclusion of these exercises, you will be able to:

- Create interface parameters
- Create global variables and types (optionally)
-



Your task: Enhance your existing invoice form and make it more flexible by inserting fields. Perform the necessary preparatory steps for the output of the booking table.

Copy template for the form:	BC470_TEXTS (This form also contains all elements that you had to create/change in the optional parts of the previous exercise.)
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 template

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

2 Enhance interface

2-1 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.

2-1-1 Add an import parameter called COLOR of the Dictionary type CHAR4 to the interface of your form.

2-1-2 Use this parameter for the company logo and make the necessary change on the *General attributes* tab of the graphic.

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

2-2-1 Refer to the Dictionary type TY_BOOKINGS.

2-2-2 Make a local check.

2-2-3 TY_BOOKINGS is a table type. What is the name of the underlying structure? Determine the name of the field in this structure that contains the price of the booking (in foreign currency).

Double-clicking TY_BOOKINGS takes you to the Dictionary.

3 Create global variable

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 determined in 2-2-3.

4 Create and use global constant

4-1 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.

4-2 Use this constant on page FIRST, window INFO, text node INFO_TEXT instead of the clerk entered there.

4-3 Add the name of the clerk to the greeting form (text node GREETINGS of the main window) by inserting the constant.

5 **Optional:** Create global type

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

6 Test

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

If the program terminates, you may not have used the correct interface specifications. **Both the names and the 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.

Solutions

Unit 5: Data in Forms

1. Copy template

   See the exercise in Unit 3.

2. Enhance interface

   2-1-1 Choose *Global Settings* → *Form interface* and go to the *Import* tab. Add COLOR TYPE CHAR4 at the end of the list.

   2-1-2 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 which option you choose, the company logo is printed in color or in black and white.

   2-2-1 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.

   2-2-2 Click the *Check* pushbutton of the maintenance screen.

   2-2-3 The structure is called SBOOK. The name of the field which contains the price of the booking in foreign currency is FORCURAM.

3. Create global variable

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

4. Create and use global constant

   4-1 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.

   4-2 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&.)

   4-3 Repeat these steps for the text node GREETINGS. (This node is in the MAIN window.)

5. **Optional:** Create global type

   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.

6. Test: See previous exercise.

Tables and Templates

BC470 Form Printing with SAP Smart Forms

Tables and Templates: Contents

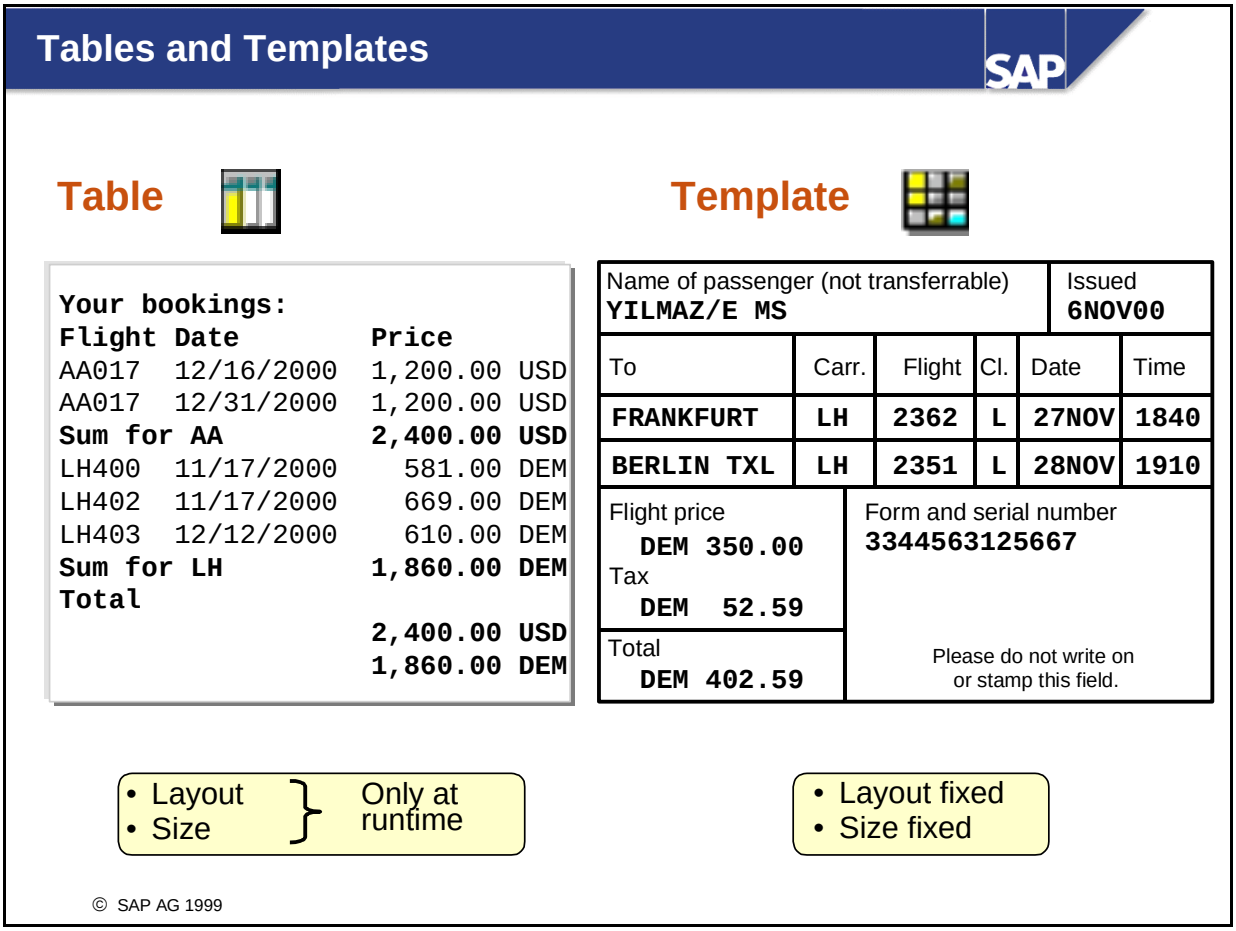
Contents:

- **Tables**
- **Templates**
- **Table Painter**

Tables and Templates: Unit Objectives

At the conclusion of this unit, you will be able to:

- **Explain the differences between tables and templates**
- **Use the Table Painter to create tables and templates**
- **Create control levels**
- **Create headers and footers**

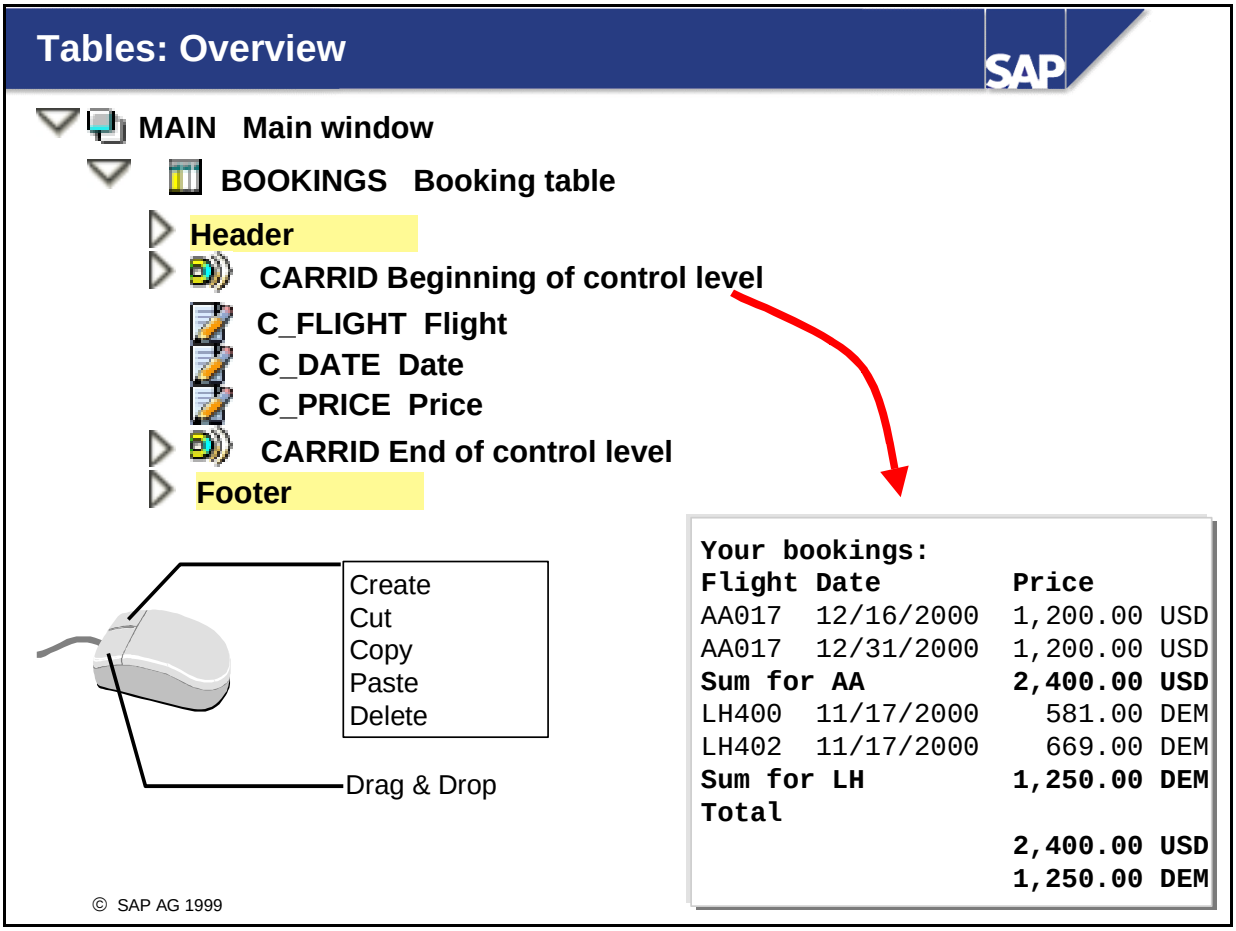


- This unit will introduce you to two other types of nodes: tables and templates. Tables and templates have several things in common. For example, they are both designed with the Table Painter, and they use different line types.
- The most important difference between them is how their layout is determined:
  - The precise layout and the length of tables can only be determined at runtime, depending on the type and the number of records read by the application program from the database.
  - Template layouts, however, are completely defined in the Form Painter. This means that the type and the number of their cells cannot be modified at application program runtime. You therefore use templates primarily for pre-printed forms, like checks or tax forms.

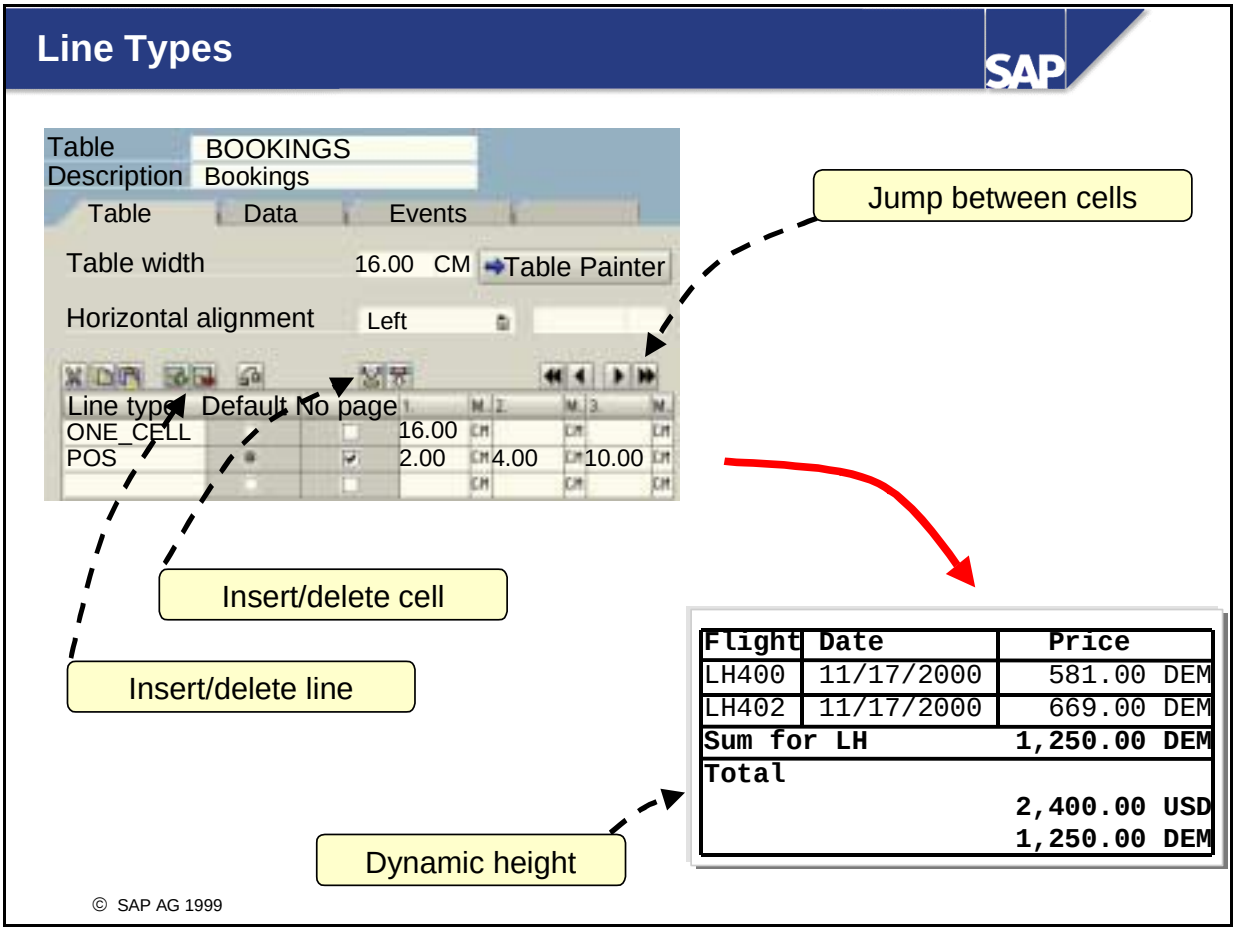
Tables: Topic Objectives

At the conclusion of this topic, you will be able to:

- **Create tables and their line types**
- **Explain how tables are processed**
- **Output data in tables**
- **Create control levels**
- **Create headers and footers**

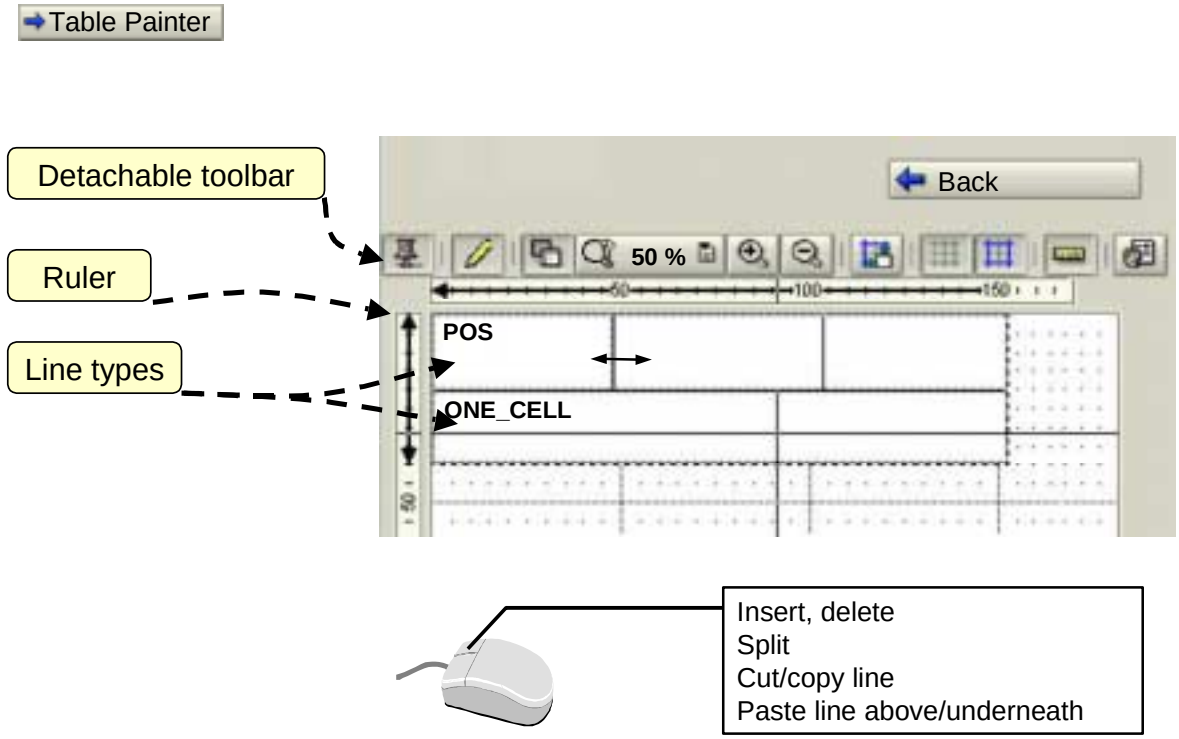


- Forms are frequently used to output data in tables. Tables in SAP Smart Forms are subnodes of windows and are created like all other subnodes using the context menu (right mouse button) of the navigation tree.
- Since the length of tables is dynamic, you should only use them in main windows since they 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, sorting levels, and subtotals.

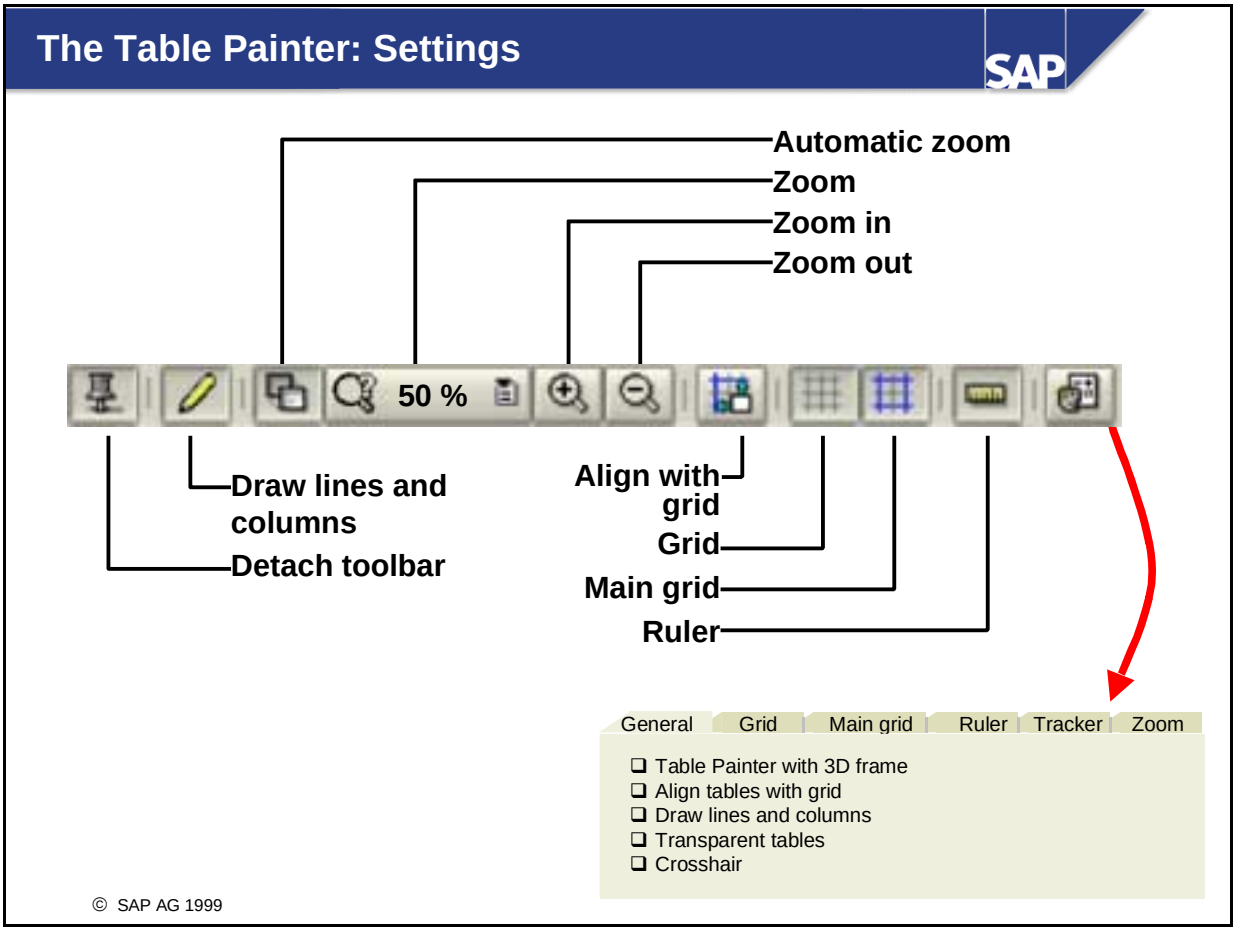


- Before you can fill tables with text, you must determine the table width and define line types on the *Table* tab of the maintenance screen. By doing this, you specify how many cells a table line should hold and what width these cells should have. (The height is determined automatically.) For simple applications, a single line type is sufficient. However, you can also create different types for hierarchical (multi-level) tables. You do this, for example, if you want to print the bookings for a flight in the next lines or if you want to use subtotals.
- In the output options of the table text nodes you specify which line types should be used when.
- The following information (in addition to the name and description) is required for line types:
  - Default type: You can only mark one type as the default type. If no line type is assigned to a subnode of the table, the system uses the default type.
  - Page 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 should 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.

The Table Painter

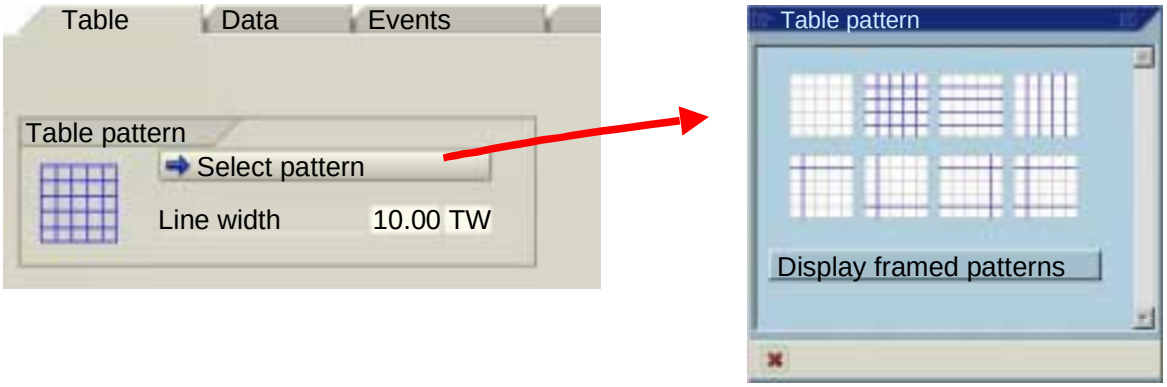


- Instead of entering the line types in the table control, you can also define them in the graphical Table Painter. The entries you make in the Table Painter are automatically copied to the alphanumeric representation and vice versa.
- You insert a new cell by vertically dragging the pencil-shaped mouse pointer at the desired position while keeping the left mouse button pressed. Alternatively, you can split 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 the boundary 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.



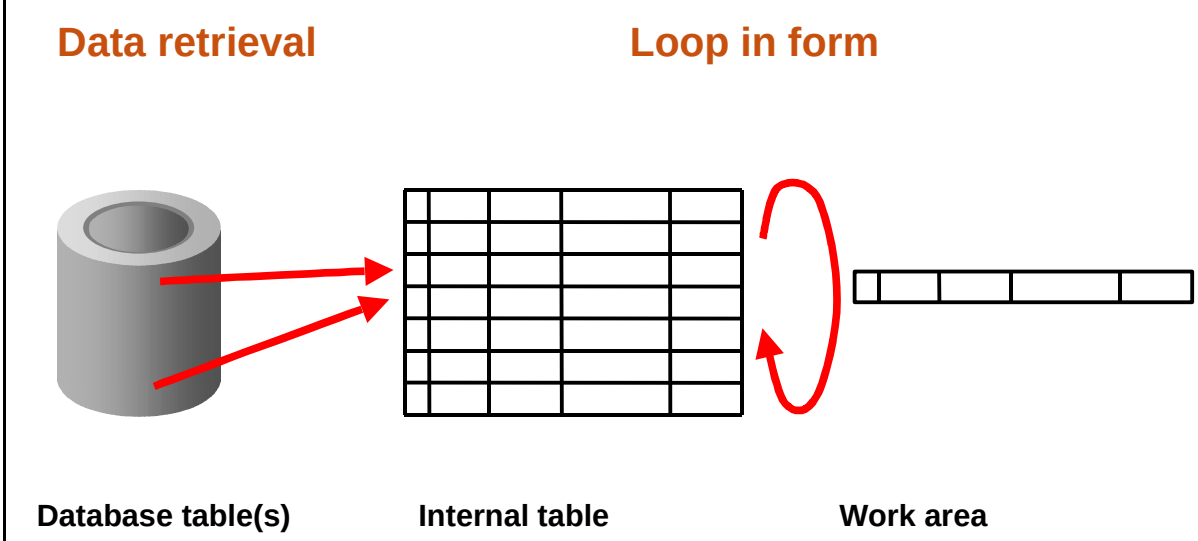
- The most important settings (which are basically identical to that of the Form Painter) are displayed in the detachable toolbar of the Table Painter. For further options, choose *Utilities* → *Settings* → *Table Painter* tab, or click the right pushbutton of the toolbar.
- If the *Draw lines and columns* checkbox is selected, you can draw the cells of the line types directly with the mouse (as described on the previous slide). If you do not select this checkbox, you can move the table within the window width to the left or to the right - provided you have not chosen *Centered* as the horizontal alignment of the table.
- Several zoom options are available to adjust the display. The most comfortable option is the *automatic zoom*.
- To make the drawing of cells easier, you can show a grid and/or the main grid. You can also make a setting in the Table Painter which 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 should be highlighted against the background.
- To hide the toolbar, choose *Utilities* → *Settings* → *Table Painter* tab, and deselect the *Toolbar* checkbox.

Table Patterns



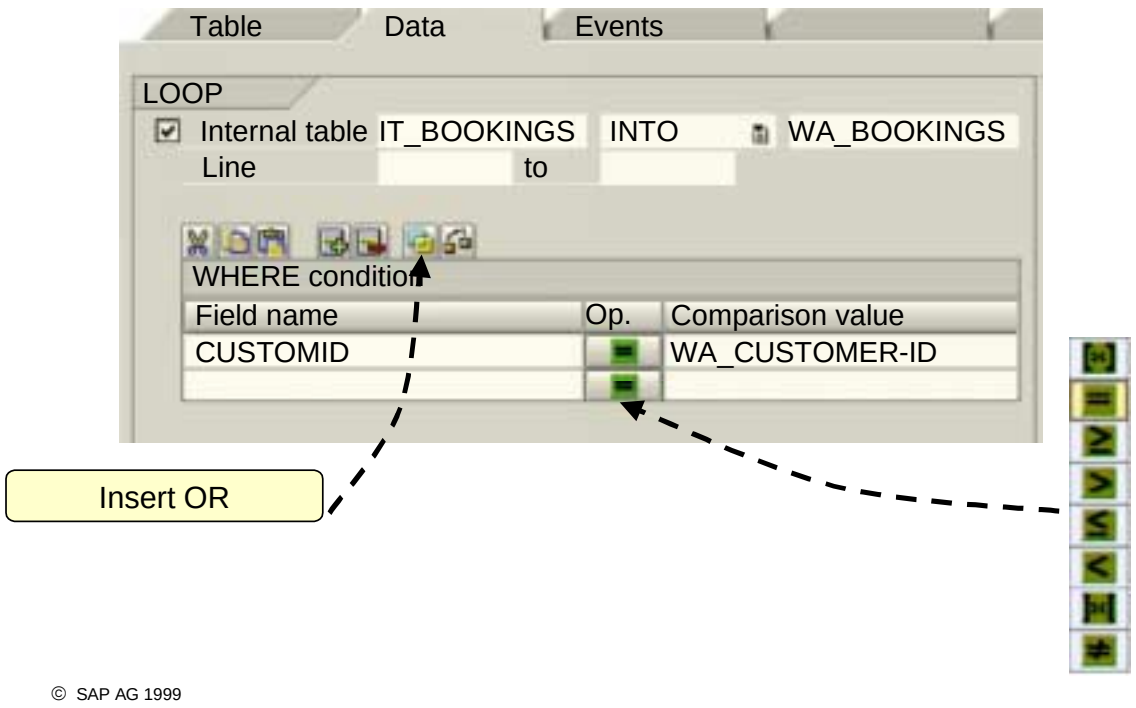
- You can define gridlines for the columns and lines of a table. To do this, you select from a number of table patterns. Choose *Select pattern* on the *Table* tab. You can also set the line width on the *Table* tab.
- Select the pattern you want to use by clicking it with the mouse. You can choose whether the first, the last or all lines should be separated by horizontal gridlines and/or whether the first, the last or all columns should be separated by vertical gridlines. All patterns are available with a border and without a border around them. 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.

Tables: Technical Background

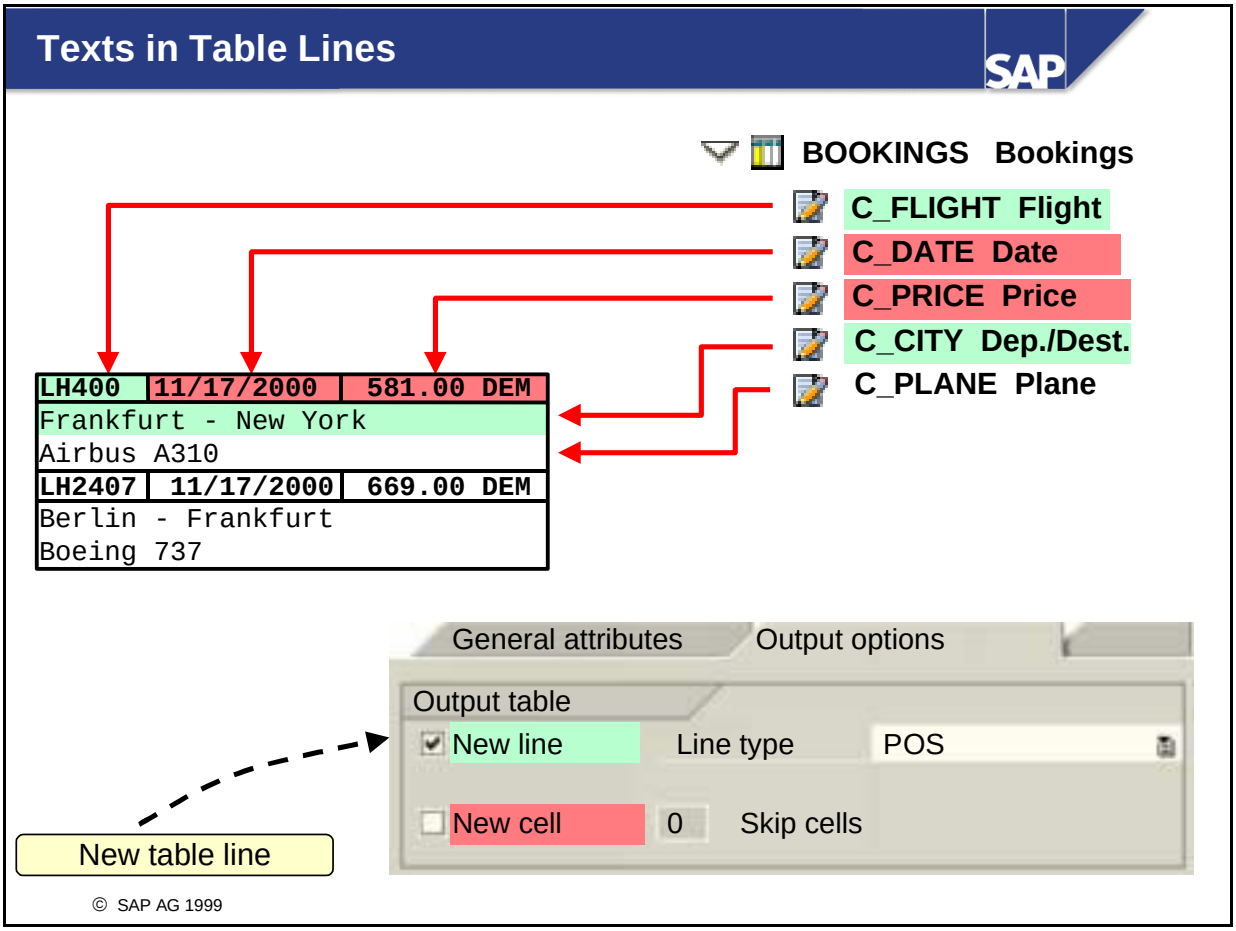


- From a technical point of view, a table in an SAP Smart Form is filled by processing a specific table called an internal table on a line-by-line basis. This is referred to as a *loop*. The respective lines can be copied into a work area that has the same structure as the table. The internal table must be filled in the application program (which is the regular case) or in the form. (For the latter case see Unit 7 - *Flow Control*.) The data is normally 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, we recommend that you process the table using field symbols instead of a work area since field symbols can directly access the individual lines so that the lines need not be copied.

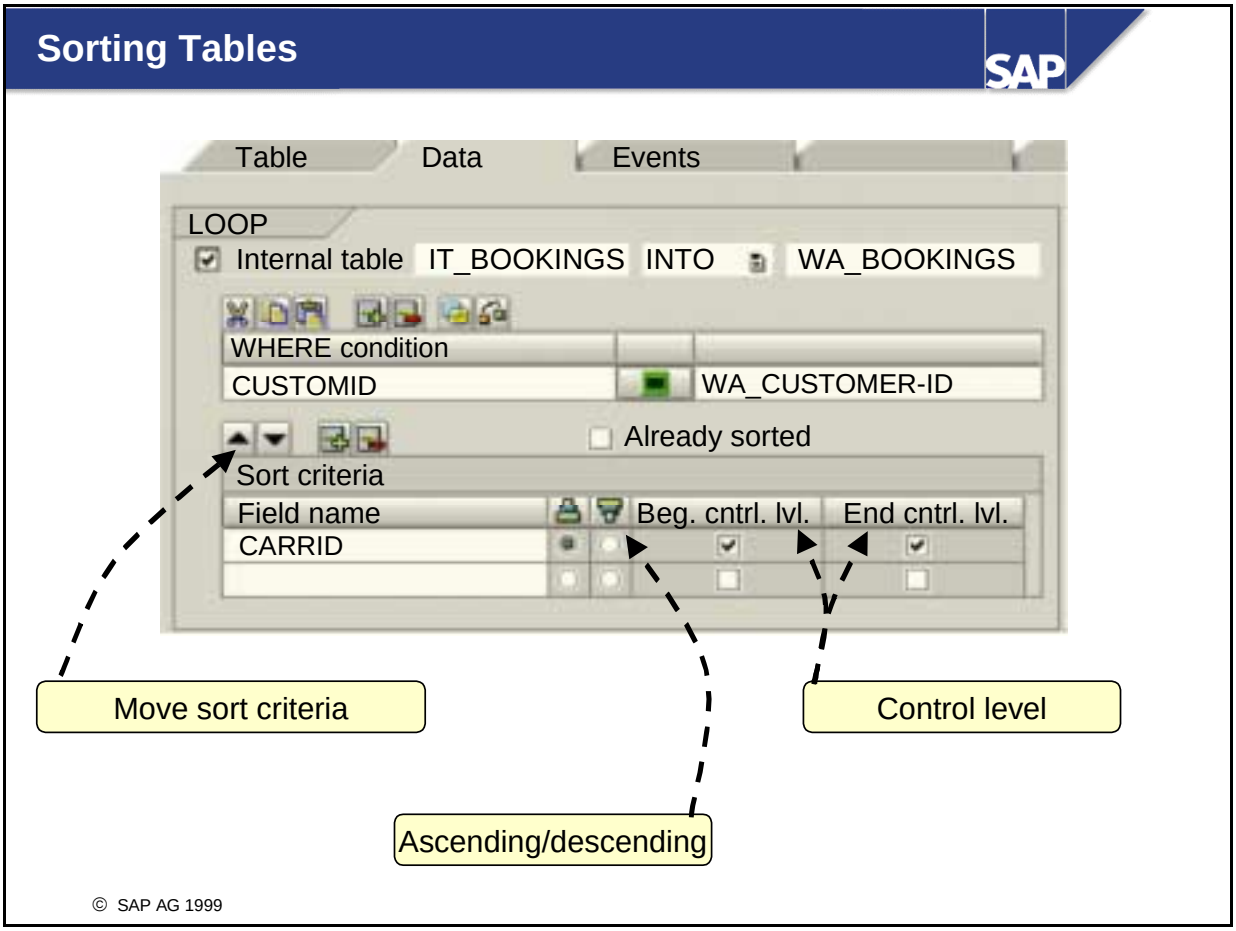
Filling Tables



- After defining the table design on the *Table* tab, you determine how the table should 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 field-symbol. Both the internal table and the work area must be known in the form. This means they must have been defined through the interface or as a global field. If you do not set the *Internal table* indicator, no loop processing takes place. This makes sense, 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 process 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 which lines of the internal table should 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. 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, Less.* If you do not enter an operator, *Equal to* is used automatically. You link several conditions with *and*; you can also use the OR pushbutton.



- In order to output text in tables, you must create at least one text node as a subnode of the table. Table text nodes are "normal" text nodes. This means you can choose the text type (text element, text module, or include text) and you can select *New paragraph* or *New line* on the *General attributes* tab to determine that the text should be written into a new text line. There is one difference, however, on the *Output options* tab. 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 marked 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 text is output in the next cell of the line type. If the line type has no more cells for the system 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 other nodes or 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*.



- You can sort the internal table within the form. To do this, enter the name of the fields as the *Sort criteria* to use. The order of the fields in this list determines the sort order. You can change the sort order later by placing your cursor on a field and moving it up or down one line by clicking one of the two black triangles displayed above the sort criteria. Two radio buttons to the right of each field allow you to determine whether the table should be sorted in ascending or descending order.
- For technical reasons, the system cannot recognize whether the internal table has already been sorted (for example, in the data retrieval program). Hence you must enter the sort criteria and select the *Already sorted* checkbox. (Otherwise, the table will be sorted again.)
- Sorting is required for subtotals and subheadings.

Control Levels I

Data records in the internal table:

AA	017	12/16/2000	1,200.00 USD
AA	017	12/31/2000	1,200.00 USD
AA	017	01/07/2001	1,200.00 USD
AA	026	02/12/2001	1,400.00 USD
AA	026	03/31/2001	1,400.00 USD
LH	400	11/17/2000	581.00 DEM
LH	400	12/19/2000	581.00 DEM

Table in the form:

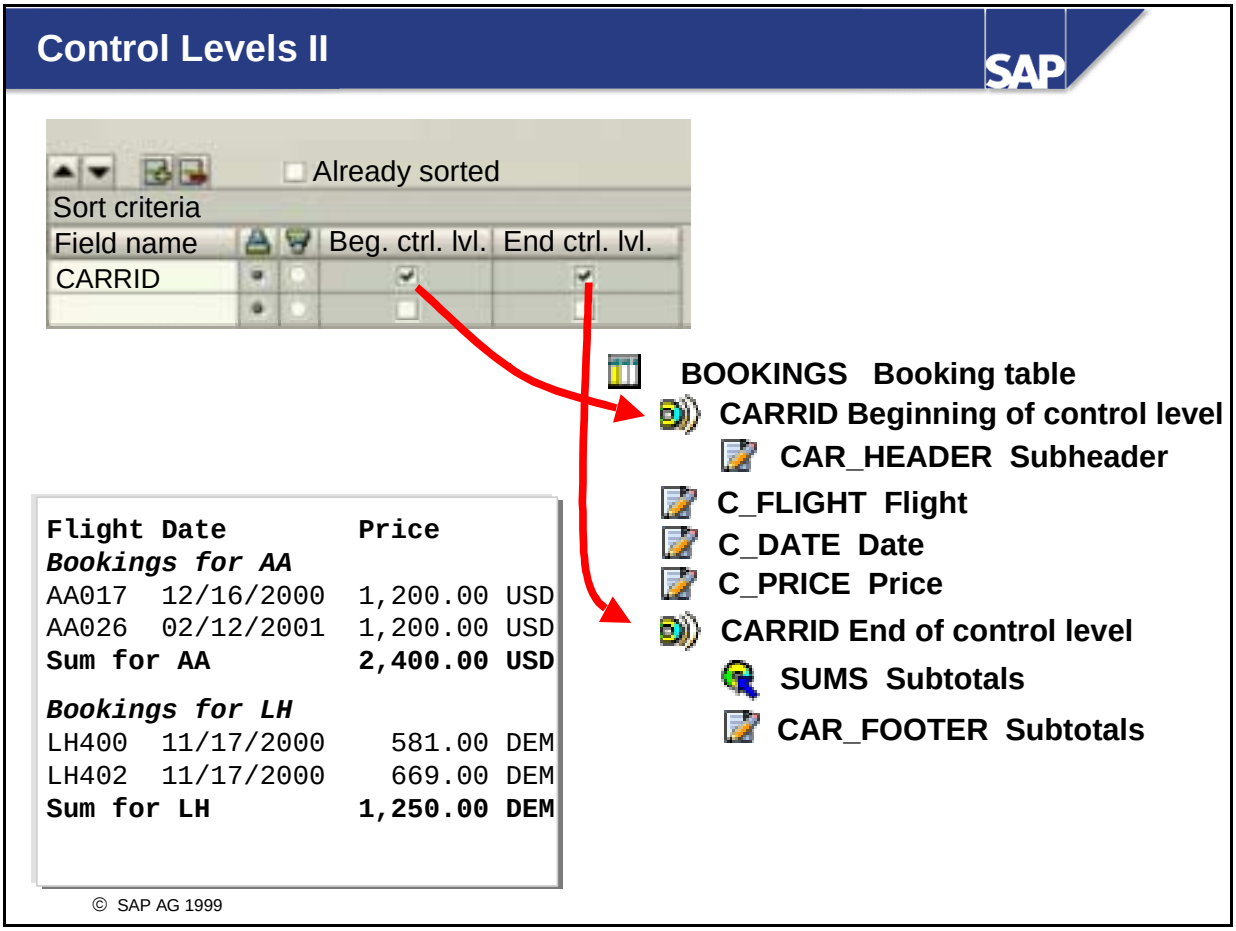
Bookings for AA

```
017  12/16/2000  1,200.00 USD
017  12/31/2000  1,200.00 USD
017  01/07/2001  1,200.00 USD
026  02/12/2001  1,400.00 USD
026  03/31/2001  1,400.00 USD
Sum for AA       6,400.00 USD
```

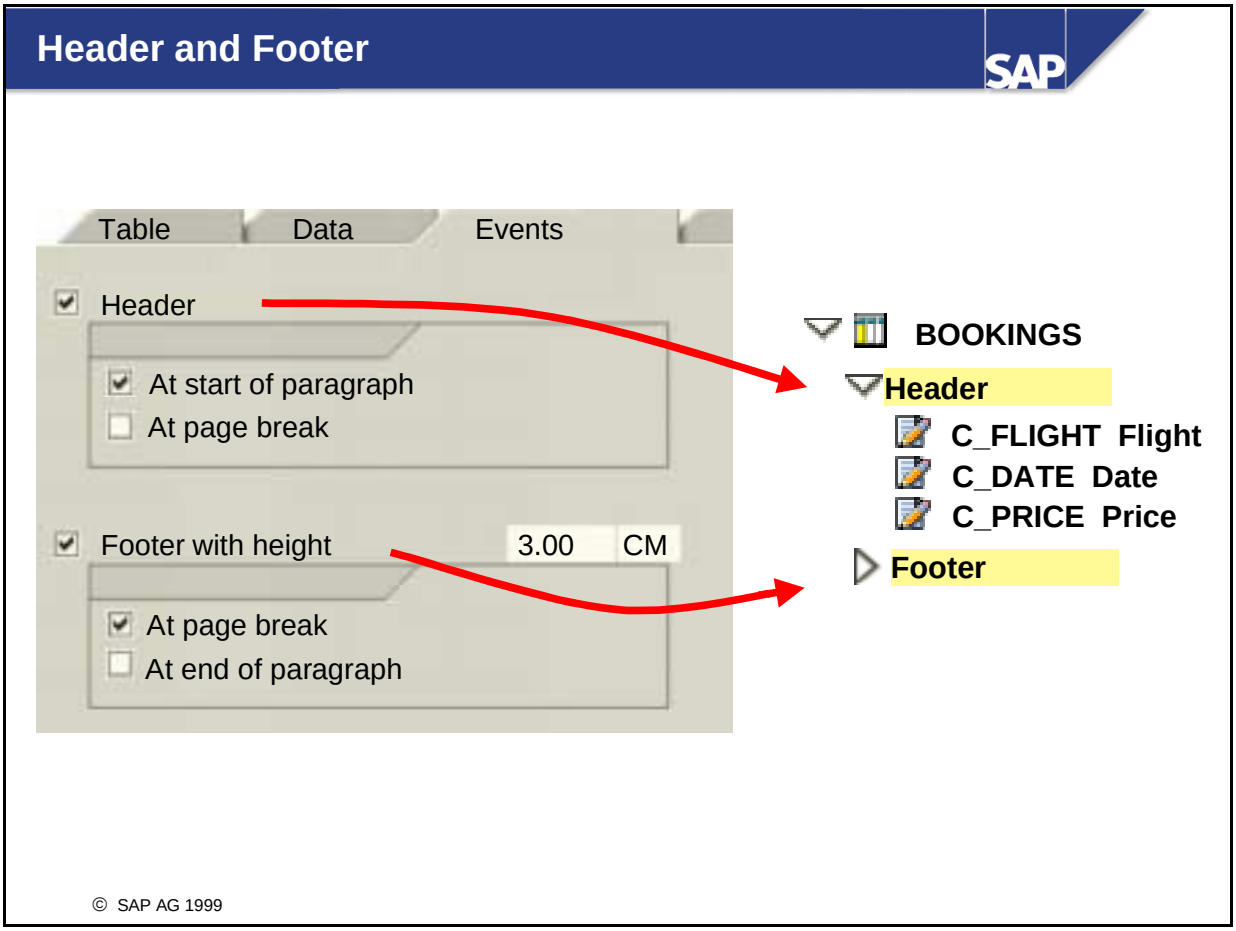
Bookings for LH

```
400  11/17/2000    581.00 DEM
400  12/19/2000    581.00 DEM
Sum for LH       1,162.00 DEM
```


- Frequently, tables are 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 subheadings or subtotals. Grouped data records that have certain identical values are called *control levels*. SAP Smart Forms allow you to create any number of control levels in a table. In the above example, there is one control level for airline carriers and another for the respective flight connections.



- If you select *Beginning of control level* and/or *End of control level* for a specific 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 can determine the data to be output at the individual control levels as required because you can insert all nodes that you could also insert as direct subnodes of a table, that is, in particular program lines for subtotals calculation, text nodes for the output of these subtotals, or command nodes for manual page breaks.
- The node of a control level, called an event node, has only one tab in the maintenance screen, the *Output options* tab. On this tab, you can make some of the settings that you can define for other subnodes of a table: You can set a style and determine the line type and the cell.
- You can define control levels for all sort fields. This means you can set up a hierarchical table that contains, for example, 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 *Beginning of control level* and/or *End of control level* checkbox.

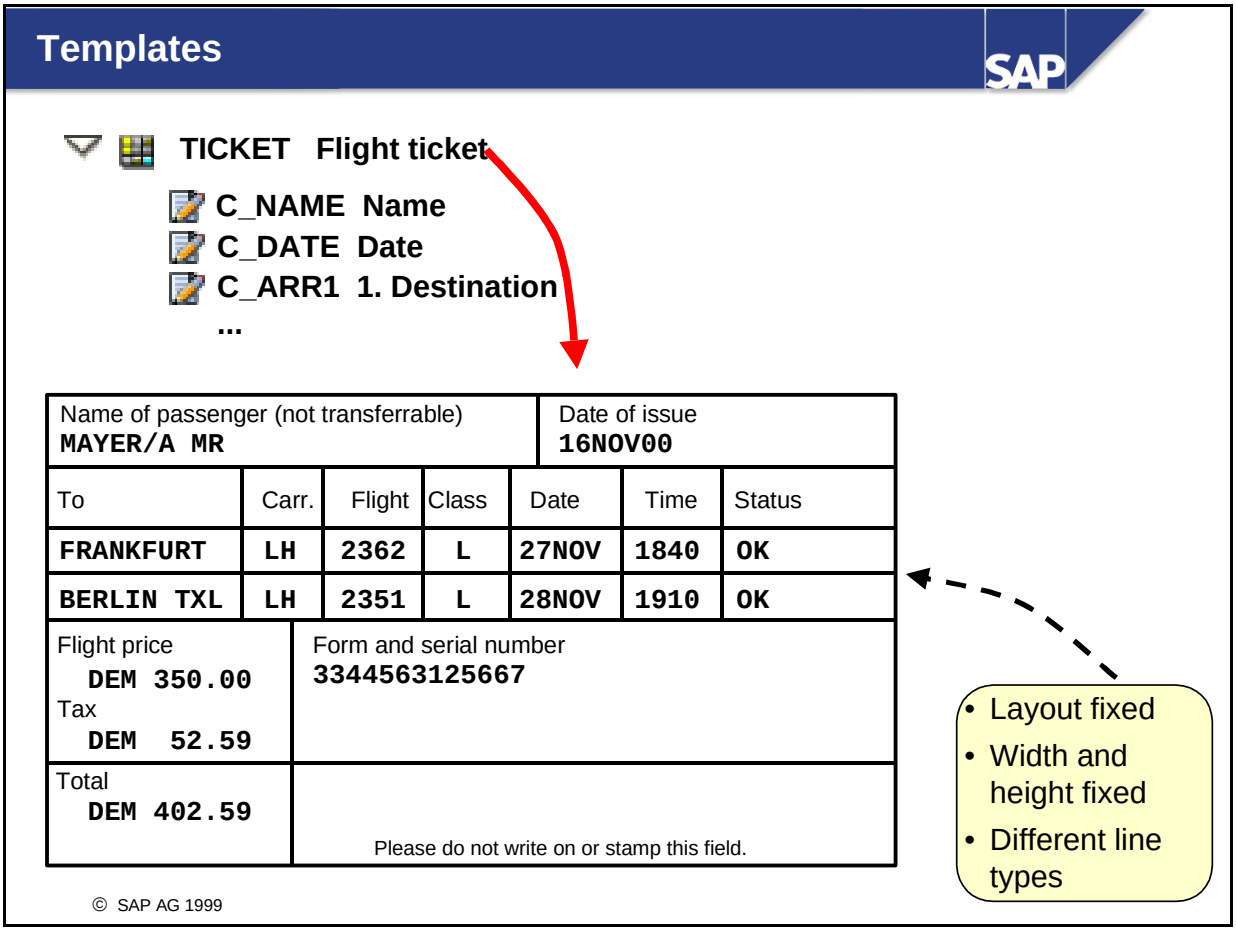


- You can use events to control the output of headers and footers in a table. To do this, you select a header and/or footer on the *Events* tab of the table node. The corresponding event node then appears in the navigation tree.
- You can output headers at the beginning of the table and/or after a page 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 to enable the form processor to reserve 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 not processed before the page break occurs. 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.

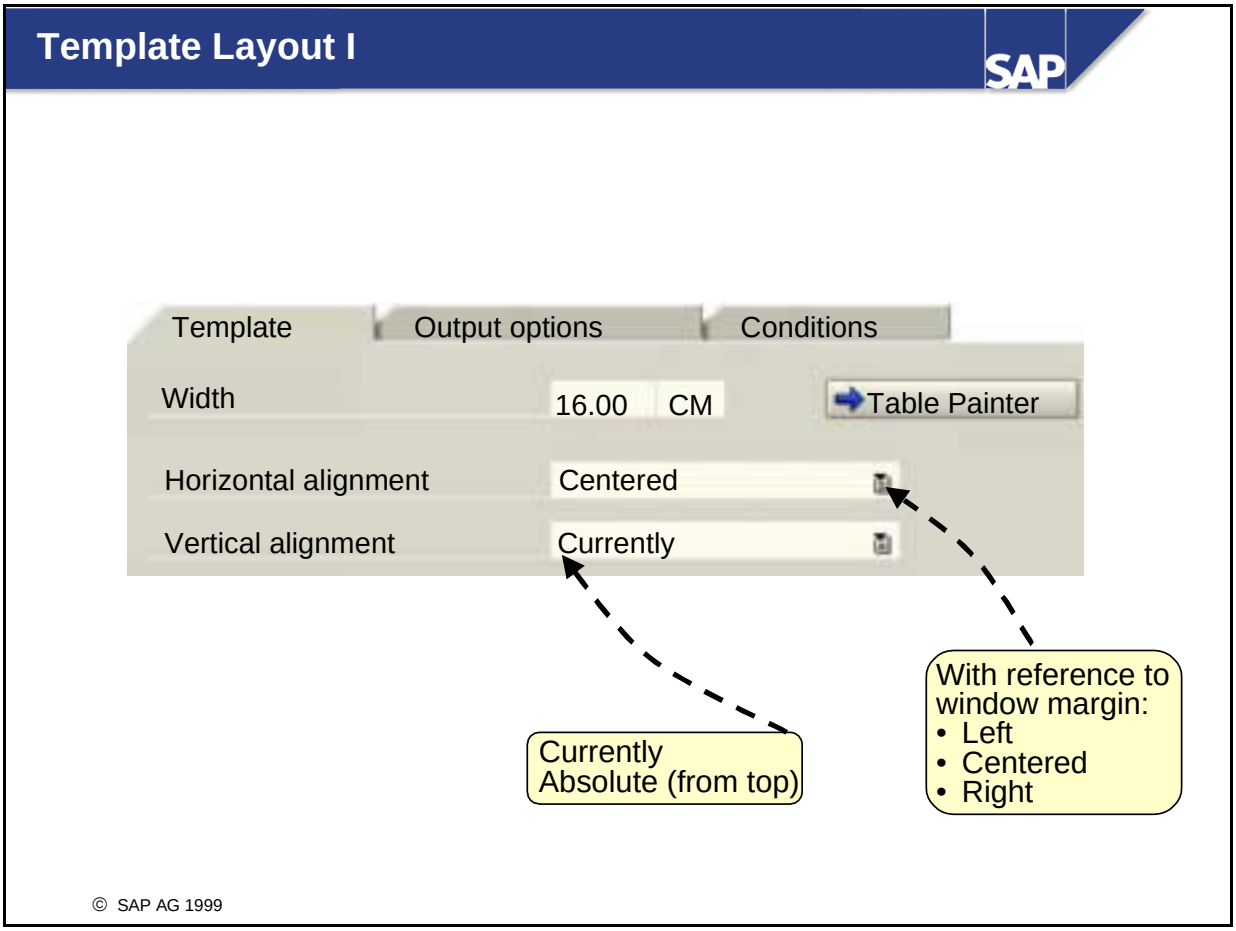
Templates: Topic Objectives

At the conclusion of this topic, you will be able to:

- **Create templates**
- **Output data in templates**
- **Create templates by redrawing**

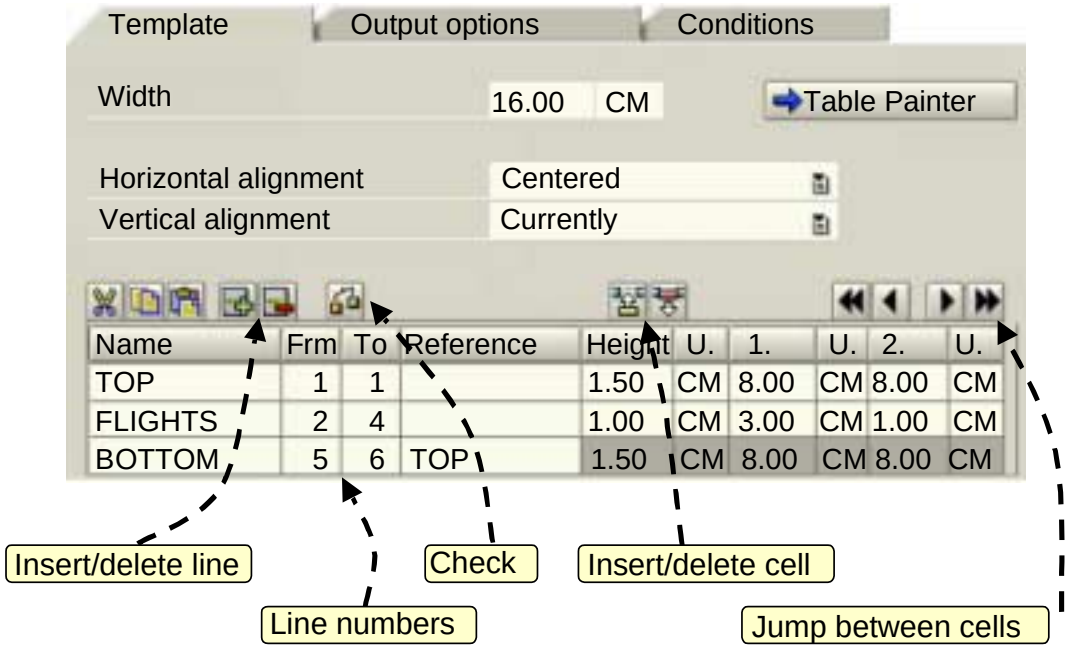


- You use the *Template* node type to output tables with a fixed layout and size. Templates are used, for example, for printing data on predefined forms such as flight tickets (see above) or tax forms.
- Like all other nodes, templates are created as subnodes of windows, that is, using the context menu (right mouse button) in the navigation tree.
- Templates cannot be nested.
- You can create different node types as subnodes of templates. Please note: Text that does not fit into the cell selected is not output since 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 them in the print preview.



- Cell- and line-related settings for templates are similar to those of tables. To define them, go to the *Template* tab.
- The width of the template must not exceed the width of the window into which the template is embedded.
- 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 into which you can enter the distance from the window margin. If you do not enter a value here, the template is placed directly on the margin.
- 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 way, you can place several templates side by side in the same window. (For example, this is useful if 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 then determined by the number of nodes processed before the template at the time of output.

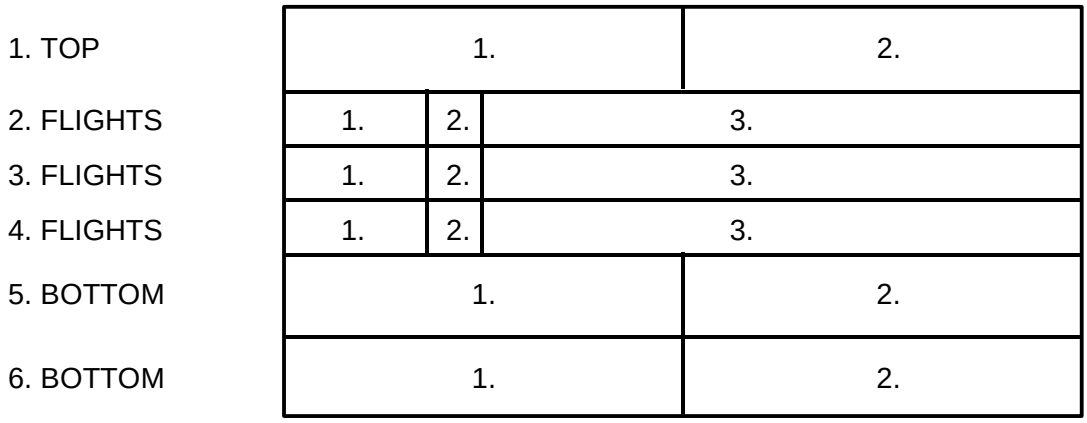
Template Layout II



- Since the layout of a template is fixed, you must describe each line - in a similar way as you describe the line types of a table.
- You first define a unique symbolic name and then the range of lines that use this line type. If several lines that are not successive use the same line type, you must define the line type only once and 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. Since the line type BOTTOM refers to the type TOP that has already been declared, 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 specify 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 identical to the width set for the template.
- You can set the values for the different lines directly in the fields or using the Table Painter. You use the Table Painter for templates in the same way as for tables (described earlier in this unit). The only differences are that you must determine the line height and that horizontal gridlines are automatically aligned with the grid - provided you have set this option in the Table Painter settings.
- You can also define a pattern for templates. What we said about table patterns is also true for template patterns. You cannot define separate patterns for individual lines or cells.
- See later in this unit for information on how to redraw existing external forms without having to determine the size of each cell.

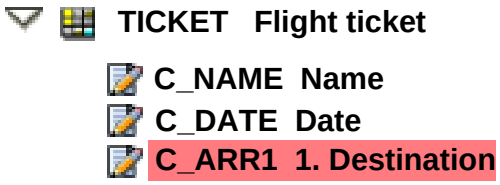
Template Layout: Example

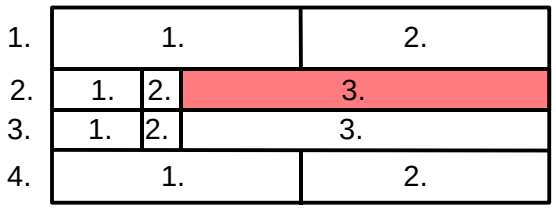
Name	Frm	To	Reference	Height	U.	1.	U.	2.	U.	3.	U.
TOP	1	1		1.50	CM	8.00	CM	8.00	CM		
FLIGHTS	2	4		1.00	CM	3.00	CM	1.00	CM	12.00	CM
BOTTOM	5	6	TOP	1.50	CM	8.00	CM	8.00	CM		

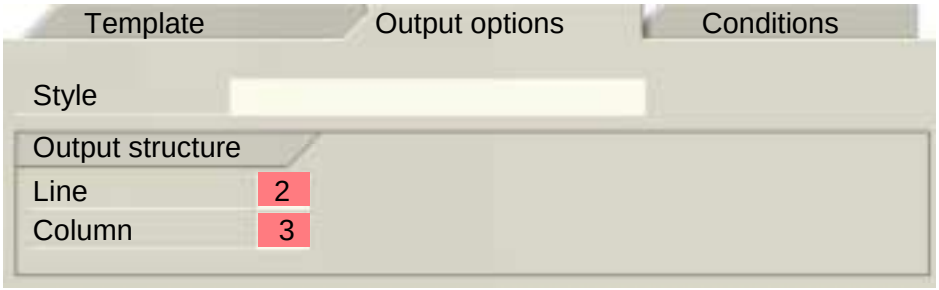


- Above you see a possible layout definition and the result of the print preview.
- You need the line and cell numbers to output contents in the cells.
- Tip: Since you cannot define cells with a different height in one line, you can put two or more tables on top of each other. To do this, enter absolute values for the vertical and the horizontal position.

Output Contents in Templates







- After defining the layout of the template, you can use the context menu (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 then 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 in which template line and cell the node is to be output. If you enter nothing here, the node is output in the current cell, that is, the cell in which data was last output. Note, however, that text that does not fit into a cell is not output.
- You can also assign several nodes to a cell. The output order in the cell is then determined by the order of the nodes in the navigation tree.

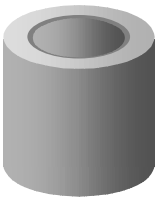
Creating Templates by Redrawing I

Template

Name of passenger				Date of issue		
To	Carr.	Flight	Class	Date	Time	Status
Flight price Tax	Form and serial number					
Total	Please do not write on or stamp this field.					

Background picture of a page

Name of passenger				Date of issue		
To	Carr.	Flight	Class	Date	Time	Status
Flight price Tax	Form and serial number					
Total	Please do not write on or stamp this field.					

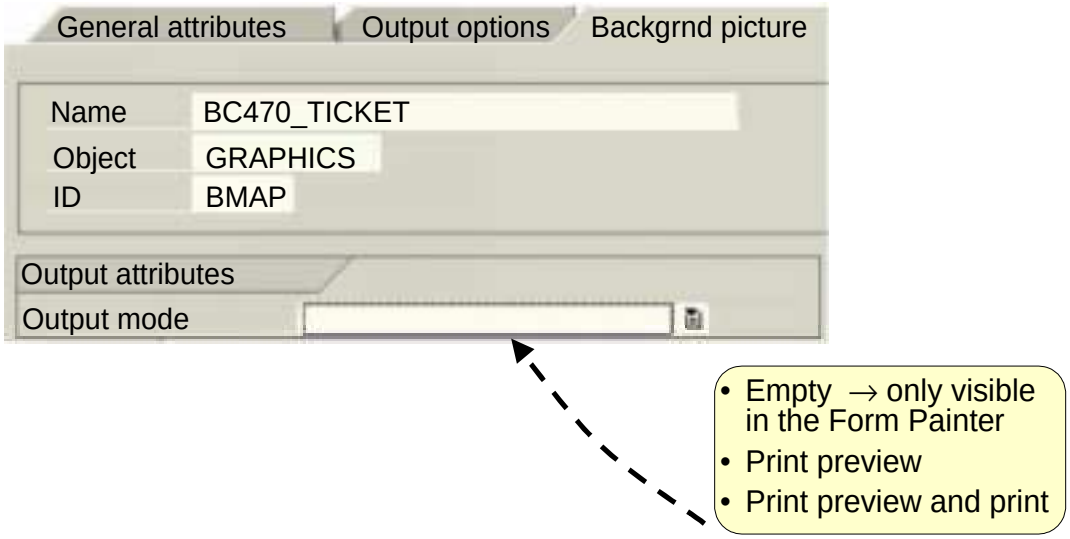


- You can create templates following the procedure we have just described. An alternative procedure - which is useful in the case of complex templates - is to scan an external form, import it using graphics administration (transaction SE78), set the graphic as the background picture for a page and then redraw the form in the Table Painter.
- To do this, follow the steps described on the next slides. (We assume that the form has already been imported as a graphic. See also *Graphics Administration* in the Appendix.)

Creating Templates by Redrawing II

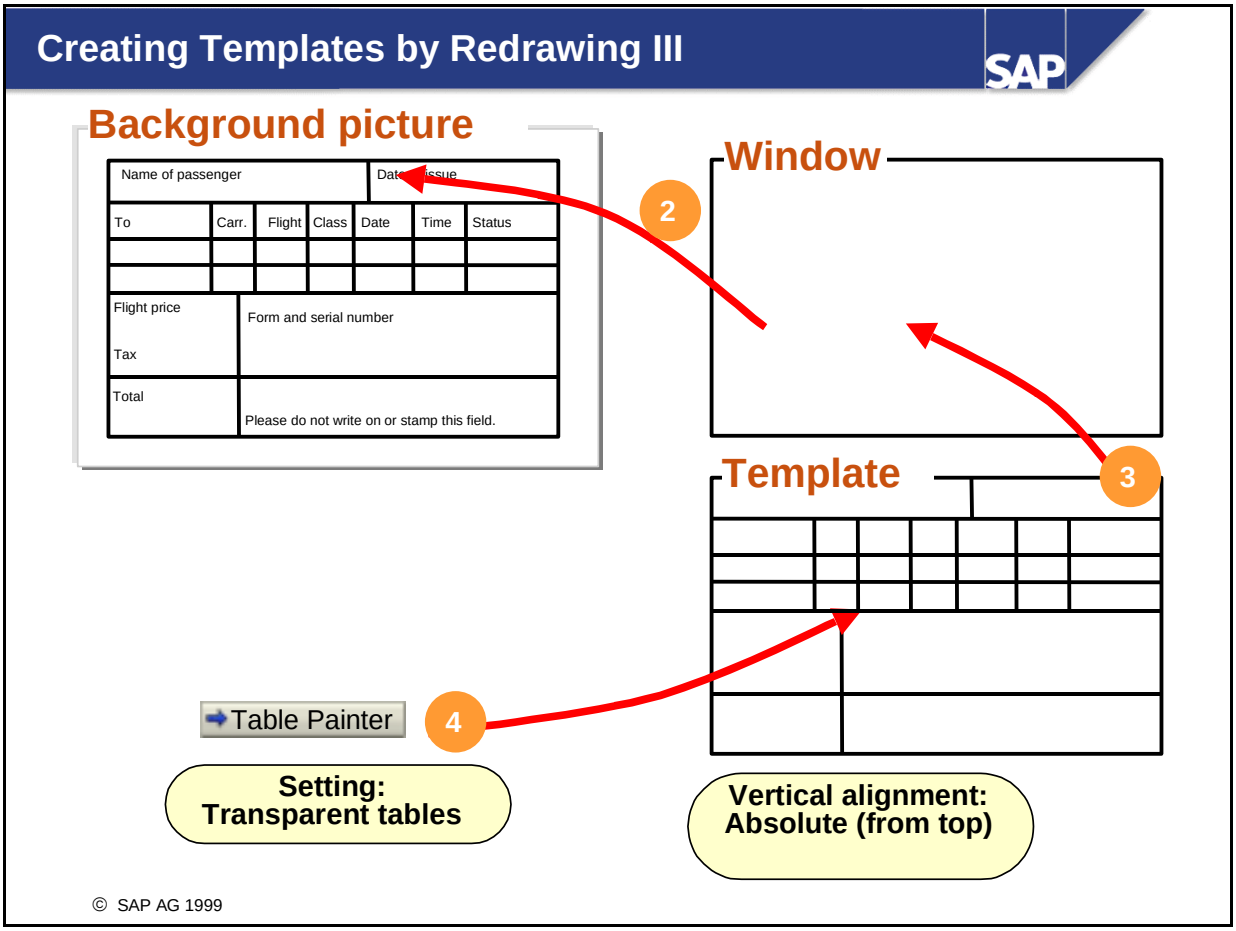
1

Graphic as background picture of a page



1. On the *Background picture* tab of the page on which you want to position the template, select the scanned form, that is, the graphic. See also the information on this subject in Unit 3. Determine in the *Output mode* combo box whether you want to print the scanned graphic or not. If you want to print your data to pre-printed paper, leave the field initial. If you want to print the background picture on blank paper, select *Print preview and print*.

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



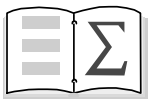
2. Use the Form Painter to place a window exactly over the graphic.

3. Create a template in that window. The template width should be identical to the window width. On the *Template* tab, select *Absolute (from top)* as the *Vertical alignment*. Start the Table Painter from this tab. 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.

4. 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 may need to create several templates above or next to each other.

Tables and Templates: Summary



You are now able to:

- **Explain the differences between tables and templates**
- **Use the Table Painter to create tables and templates**
- **Create sort levels**
- **Create headers and footers**

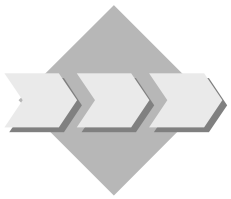
Exercises

Unit 6: Tables and Templates

Topic: Tables

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
-



Your task: Enhance your existing invoice form and create a "real" table to output actual customer bookings.

Copy template for the form:	BC470_DATAS (This form also contains all elements that you had to create/change in the optional parts of the previous exercise.)
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

1. Copy template

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

2. Create table

   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.

   2-1 Determine data for table

In the MAIN window, create a table node called BOOKINGS below 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.)

Consider the fact that IT_BOOKINGS contains the data for **all** customers selected on the selection screen. In the WHERE condition, determine 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 that is filled in the application program.)

Perform a local check.

2-2 Determine table layout

The table should have the following structure:

Flight	***Flight date***	***Price***
Bookings for AA		
AA0017	10/16/2000	1,200.00 DEM
AA0017	10/17/2000	1,200.00 DEM
AA0026	11/18/2000	700.00 USD
*********		

2-2-1 Set the table width to 15.3 cm.

2-2-2 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 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. (You should nevertheless try and use the Table Painter!)

2-2-3 Prevent page breaks from occurring within one line type.

2-2-4 Perform a local check.

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.

3-2 Select the paragraph format TB ("Cell in table body") for all text nodes.

3-2 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.

3-3 Output the flight date (WA_BOOKINGS-FLDATE) in C_DATE.

3-4 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 paragraph format TB. Define the following formatting options for WA_BOOKINGS-FORCURAM:

- Output length: 13
- Decimal places: 2

3-5 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.

4. Define a table pattern.

5. Define column headings

The table should have the column headings "Flight", "Flight date" and "Price".

5-1 Select the *Header* event for the table.

5-2 Create three text elements as subnodes of the header: H_FLIGHT, H_DATE and H_PRICE.

The easiest way to do this is to copy the existing elements C_FLIGHT, C_DATE, and C_PRICE to the header (Strg and mouse). This ensures that the descriptions are correct and that the headings are output in the correct cells. Select the paragraph format TH ("Cell in table header") for all three text nodes.

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.

6. **Optional:** Define control levels

The table should have control levels for the airline carriers. Mark 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.

6-2 Insert subheading

6-2-1 To output a subheading for each carrier, insert a text node called CARRIER_HEADING below the beginning of the control level.

6-2-2 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.

6-2-3 Make sure that the line type ONE_CELL is used for the subheading.

6-3 Insert asterisk string (dummy subtotals)

Output a string of asterisks after each airline carrier. (You will replace these asterisks with actual subtotals in one of the next exercises.)

6-3-1 Insert a text node called SUBTOTAL as a subnode of the control level end.

6-3-2 Enter a few asterisks.

6-3-3 Make sure that the line type ONE_CELL is used for the asterisks.

7. Delete the text node DUMMY_TABLE that you do no longer need.

8. If you have reduced the height of the main window on the page FIRST in one of the previous exercises and now use this form, increase the height again.
9. Activate your form and test it using the program SAPBC470_DEMO.

Solutions

Unit 6: Tables and Templates

Topic: Tables

1. Copy template

   See the exercise in Unit 3.

2. Insert table

   2-1 From the context menu of the text node DUMMY_TABLE, choose *Create -> Table*. Change the name to read 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.

   Click the *Check* pushbutton of the maintenance screen.

   2-2 Determine table layout

   You define the layout on the *Table* tab.

   2-2-1 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 1.8 cm, 3.0 cm, and 10.5 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.

   2-2-3 Select the *No page break* checkbox for the two line types POS and ONE_CELL.

   2-2-4 Click the *Check* pushbutton of the maintenance screen.

3. Fill table

   3-1 Use the context menu of the table (right mouse button) to create three text elements as subnodes (**not** as successor nodes) and change their names.

   3-2 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.

   3-3 Repeat these steps for the text node C_DATE and WA_BOOKINGS-FLDATE.

   3-4 Repeat these steps for the text node C_PRICE and WA_BOKINGS-FORCURAM and WA_BOKINGS-FORCURKEY.
   Place your cursor on WA_BOKINGS-FORCURAM. Click the Change field pushbutton (which is the third pushbutton from the right in the editor toolbar). On the dialog box that appears change &WA_BOOKINGS-FORCURAM& into &WA_BOOKINGS-FORCURAM(13.2)&.

   3-5 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*.
   Select *New cell* for the text node C_PRICE as well.

4. On the *Table* tab, click the *Select pattern* pushbutton. Select a pattern on the dialog box that appears.

5. Define column headings

   5-1 Select the Header checkbox on the Events tab of the table BOOKINGS.

   5-2 Use the context menu (right mouse button) of the header to create the three text nodes.
   You select the paragraph format for each node in the editor (General attributes tab).

   5-3 Enter the texts in the editor (*General attributes* tab).

6. **Optional:** Define control levels

   6-1 On the *Data* tab of the table BOOKINGS, enter CARRID as the field name for the sort criteria and select the *Beginning of control level* and *End of control level* 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 Beginning of control level" 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).

      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 Insert asterisk chain (dummy subtotals)

      6-3-1 You already created the event node "CARRID End of control level" 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.

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

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

Flow Control

BC470 Form Printing with SAP Smart Forms

1 Course Overview
2 SAP Smart Forms: Overview
3 First Steps with the SAP Form Builder
4 Texts, Addresses, and Graphics
5 Data in Forms
6 Tables and Templates
7 Flow Control
8 Integration into Application Programs
9 Smart Styles
10 Fonts and Bar Codes
Appendix

Flow Control: Contents

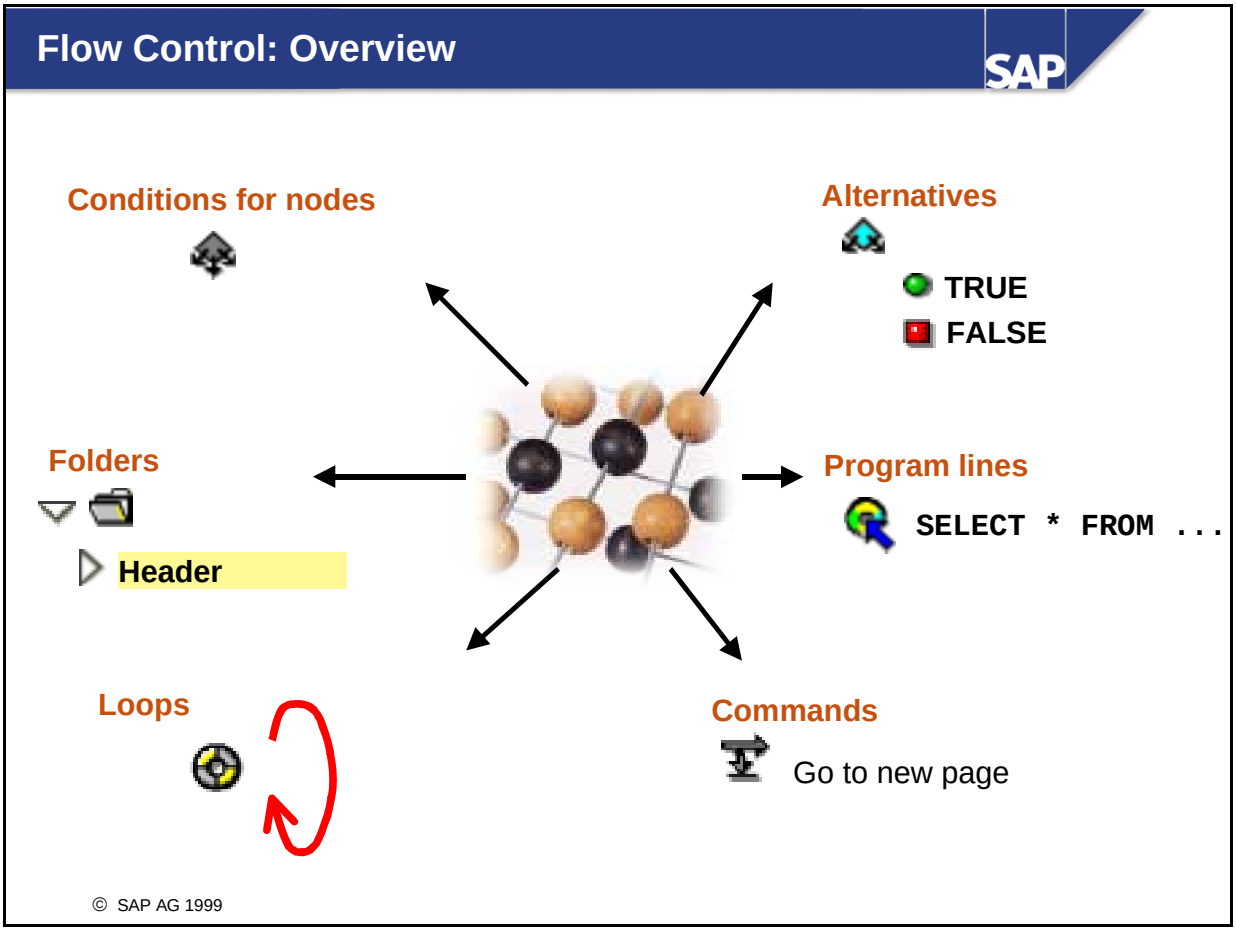
Contents:

- Node Conditions
- Alternatives
- Program Lines
- Global Form Routines
- Command Nodes
- Loops
- Folders

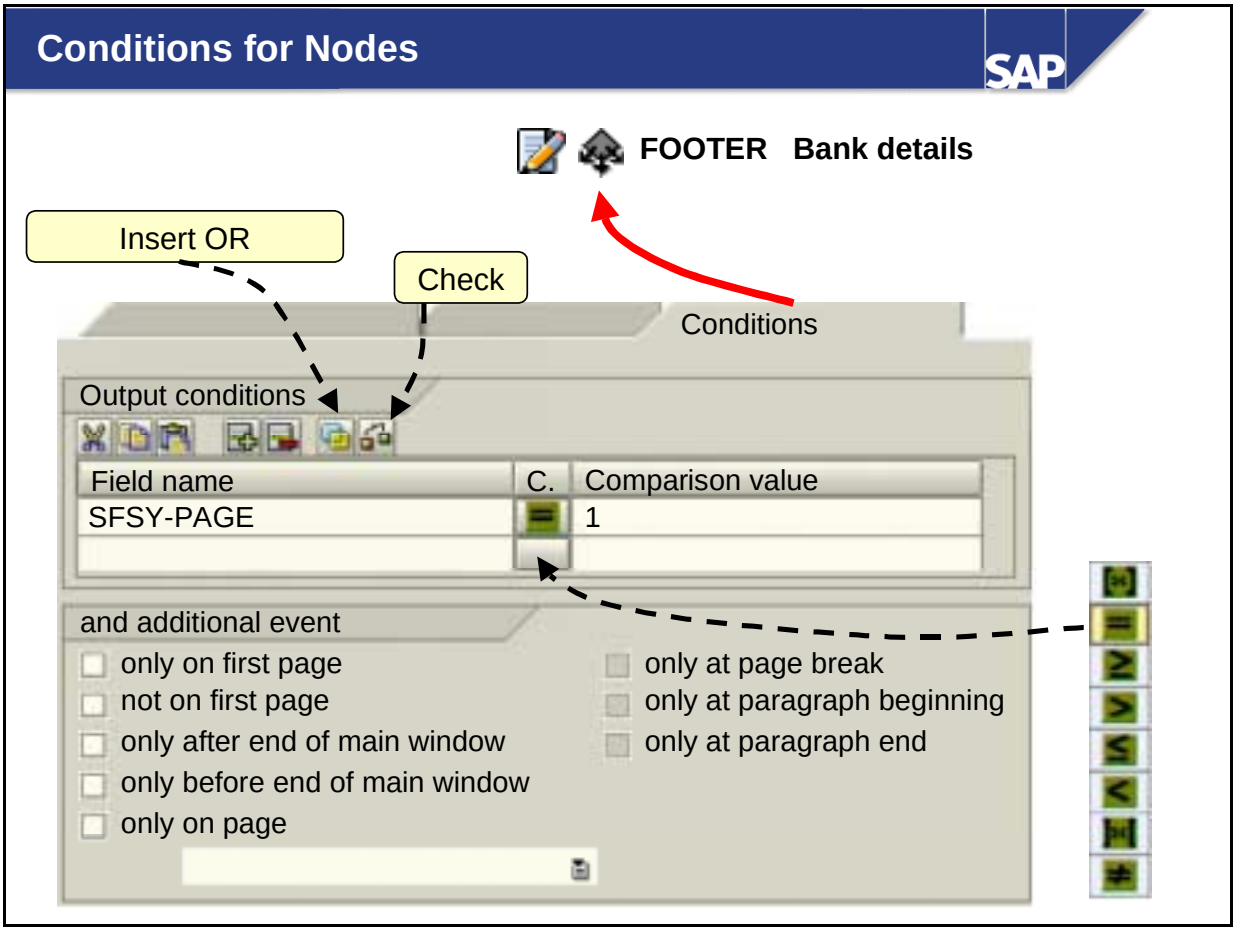
Flow Control: Unit Objectives

At the conclusion of this unit, you will be able to:

- **Explain how the various nodes can be used for flow control**
- **Create flow control nodes**

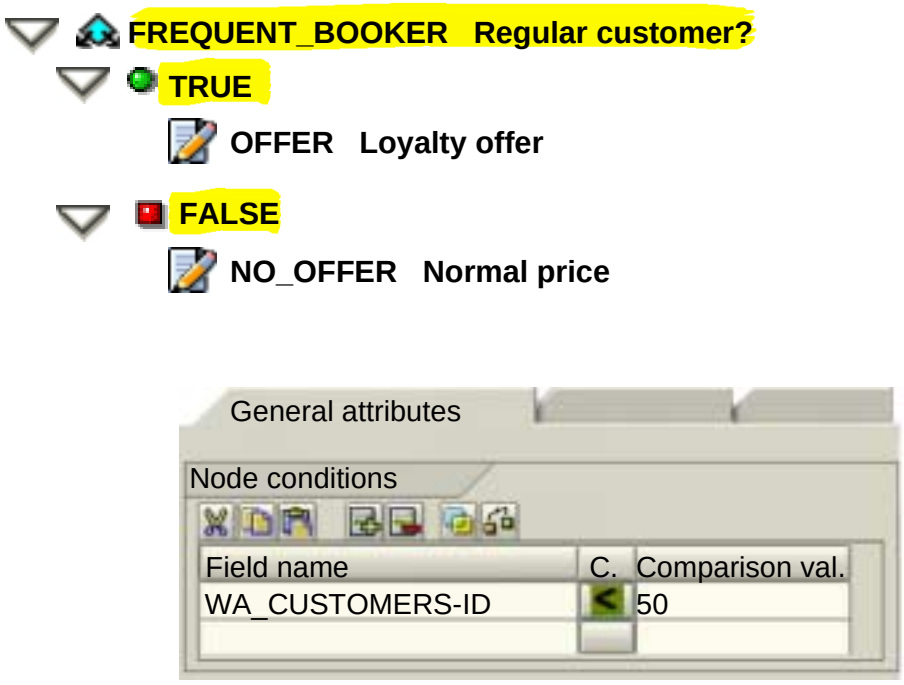


- The form elements presented to you up to this point are processed in a predefined order. Starting with the first page, the nodes of the tree structure are processed from top to bottom. It is helpful to imagine that all nodes are expanded.
- In some cases, however, the system can only determine at runtime which parts of a form should be processed. One 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** allow you to integrate ABAP statements into your form without having to adjust the application program.
  - **Command nodes** are used for dynamic page breaks, for example.
  - Subnodes of **loops** are executed several times.
  - **Folders** allow you to group nodes.

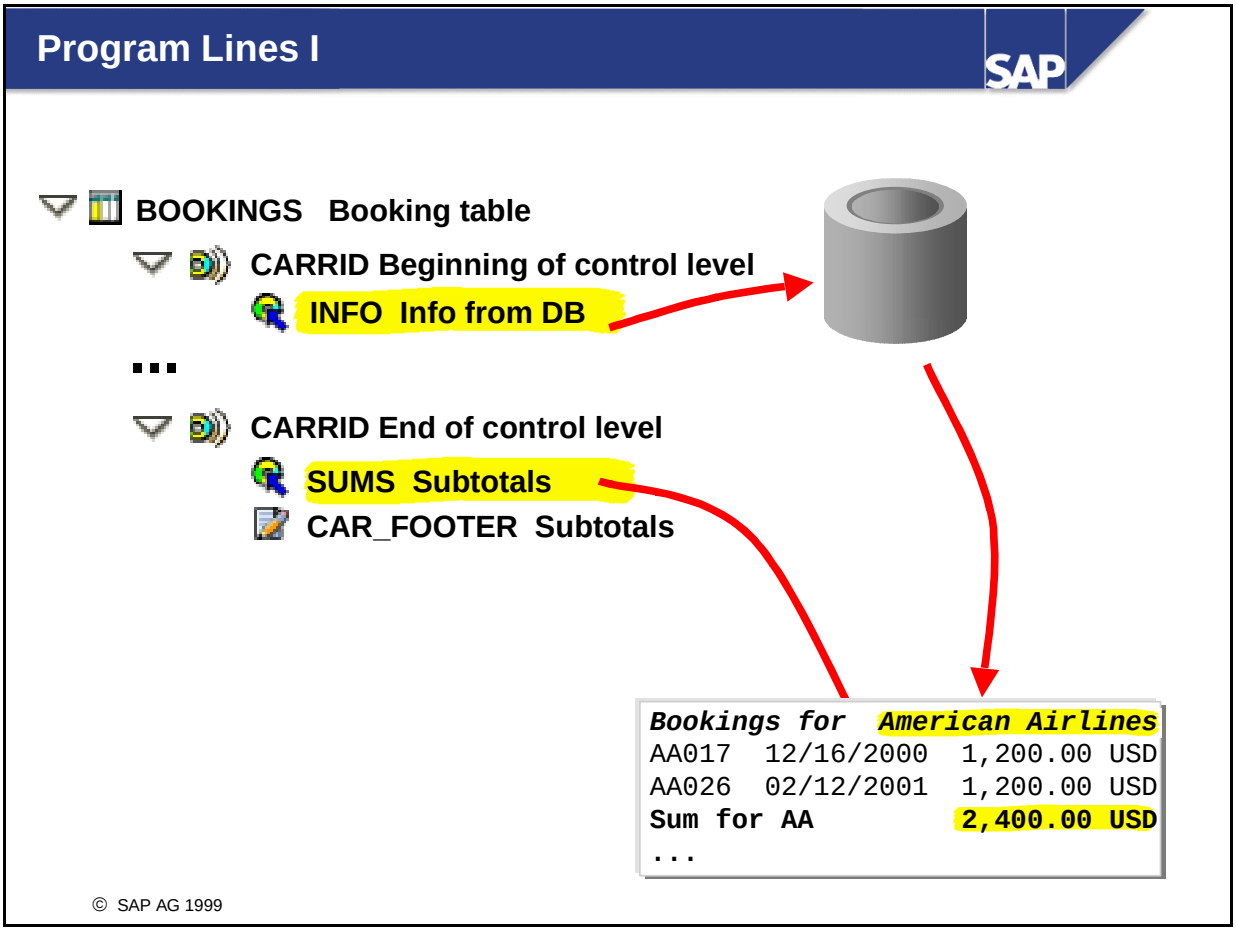


- Most nodes have the *Conditions* tab. You can define two types of conditions for processing the respective node and all of its subnodes:
  - Field comparisons:
    - Enter a field name without ampersands in each line. Then select a relational operator (the default operator is *Equal to*) by clicking 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-...). Do not use the system fields FORMPAGES and JOBPAGES in conditions because their value is only determined after the form is processed and is then inserted into the form in a second run.
    - If you insert several conditions, these are linked by a logical AND. Using the pushbutton shown above, you can also define an OR relationship.
  - Specific events: The options available depend on which node is selected. *Only at paragraph beginning* and *Only at paragraph end* are, for example, only available for headers or footers of tables and for complex sections.
- Check your entries with the *Check* pushbutton on the tab.
- In the navigation tree, the icon shown above is added to the respective node if you define a condition.
- If you use identical windows, graphic windows or address windows on different pages, then each has its own *Conditions* tab.

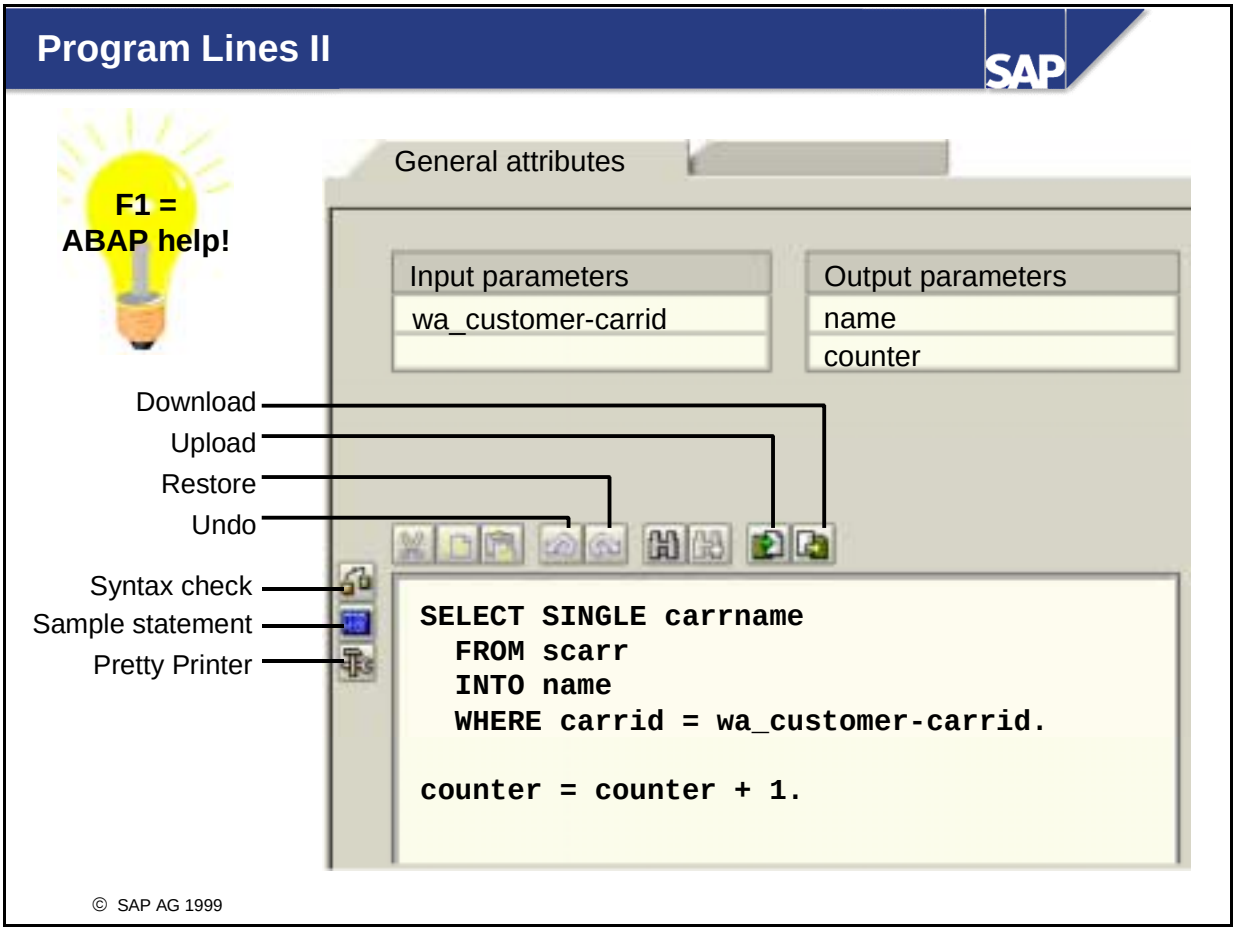
Alternatives



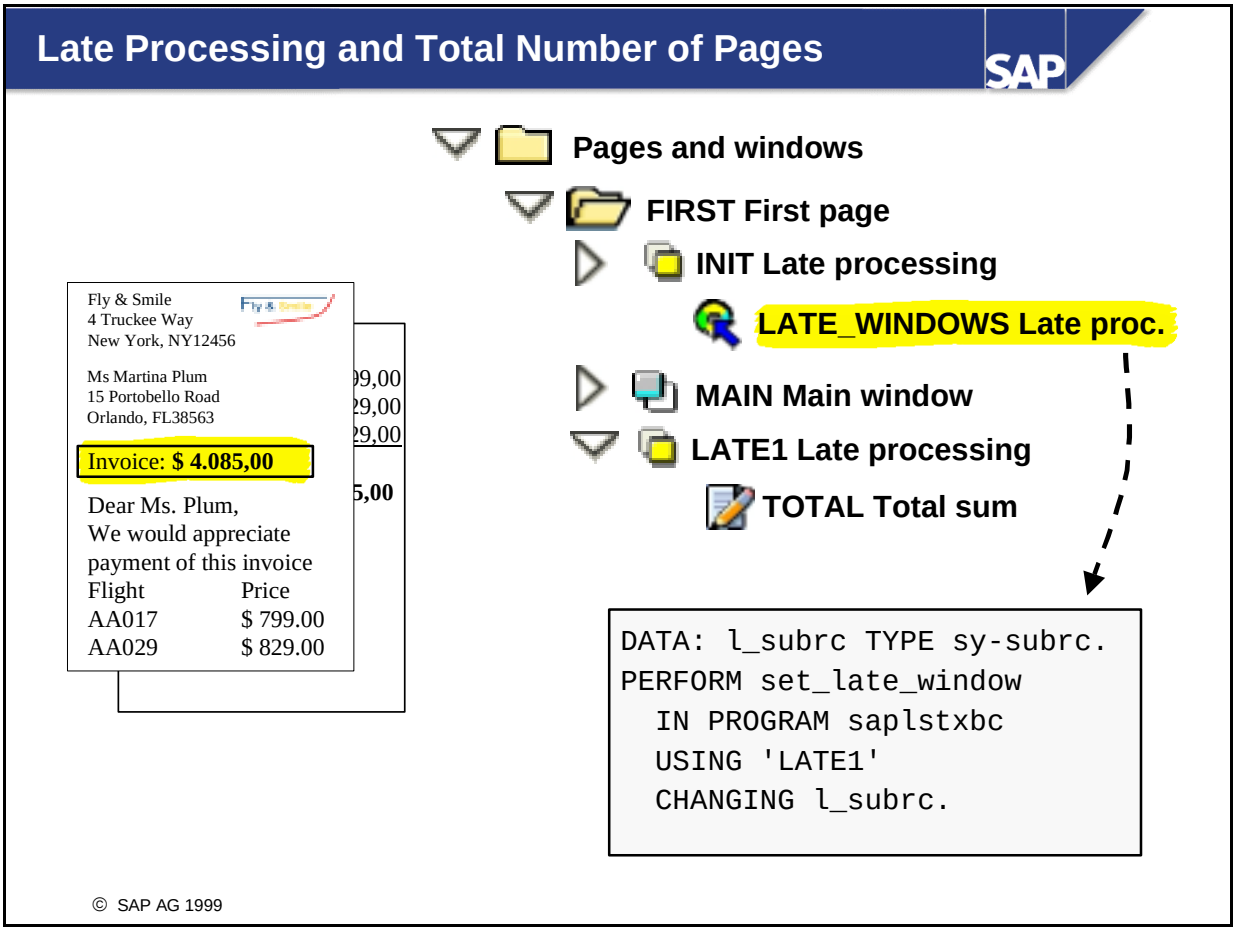
- An alternative is a node with two subnodes (which contain subnodes themselves). The condition(s) you enter on the *General attributes* tab of the alternative determine which of the two subnodes is processed. If the condition is fulfilled, the node TRUE is processed including all of its subnodes. If the condition is not fulfilled, the node FALSE is processed together with its subnodes. This is a query similar to that of the ABAP commands if and else.
- You create alternatives like you create any other node by using the context menu of the navigation tree or by choosing *Edit* → *Node* → *Create* from the menu.
- 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.



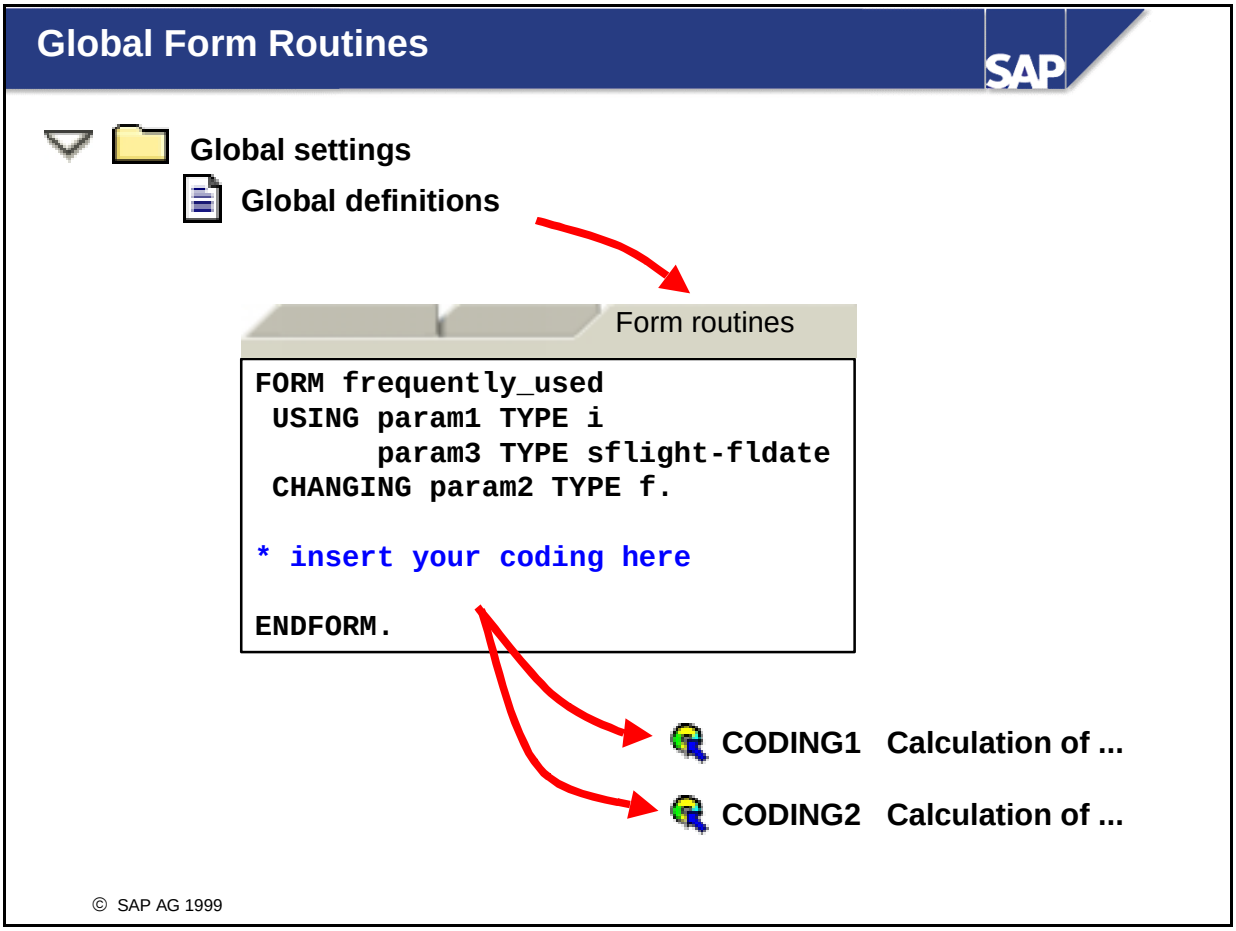
- Nodes of the type *Program lines* allow you to integrate ABAP code into your form. There are numerous situations where this is helpful, for example:
  - You need data that is not provided by the application program that is actually responsible for data retrieval. You could indeed modify the program but this could be a very complex task and, what is more, you would not be able to benefit from enhancements made to this program when performing an upgrade.
  - You want to reset counter variables.
  - You want to calculate subtotals and totals, for example, within a table.
- You create a program lines node using the context menu or by choosing *Edit* → *Node* → *Create* from the menu. Pay attention to 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 subsequently.
- Since program lines nodes cannot generate output and cannot have subnodes, they do not have an *Output options* tab.



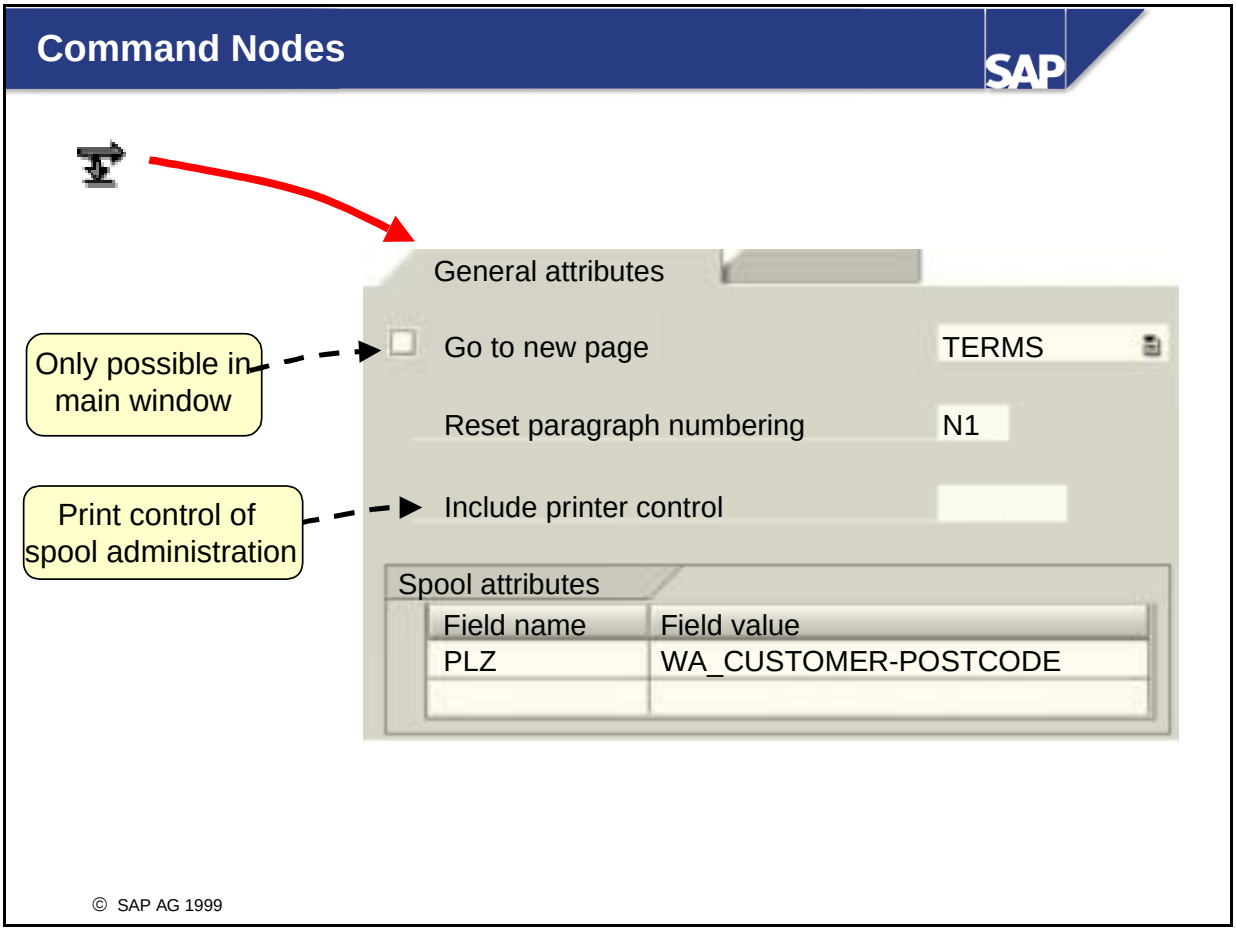
- Program lines in SAP Smart Forms are similar to subroutines in ABAP programs. This means that you must determine the interface but can also work with local variables that you create with the DATA statement. The input and output parameters must be globally known in the form, that is, they must be defined in the interface or in the global definitions.
- The differentiation between input parameters and output parameters is only made for structuring purposes. In particular, it has no effect on the changeability of the parameters since both input and output parameters are passed by reference to the program lines node. As a result, changes made to values of input parameters are also permanent and not only valid within the program lines.
- System fields of SAP Smart Forms (sfsy-...) or the ABAP system table SYST (sy-...) do not have to be declared in the interface but can be used directly in the coding. You should access system fields exclusively in read-only mode.
- You can use form routines in the program lines that you have created in the global definitions of the form. (See next slide.)
- If you prefer to work with the old line-oriented ABAP editor, you can set it by choosing *Utilities* → *Settings* → *Editor* tab → *Table Control Editor* radio button.



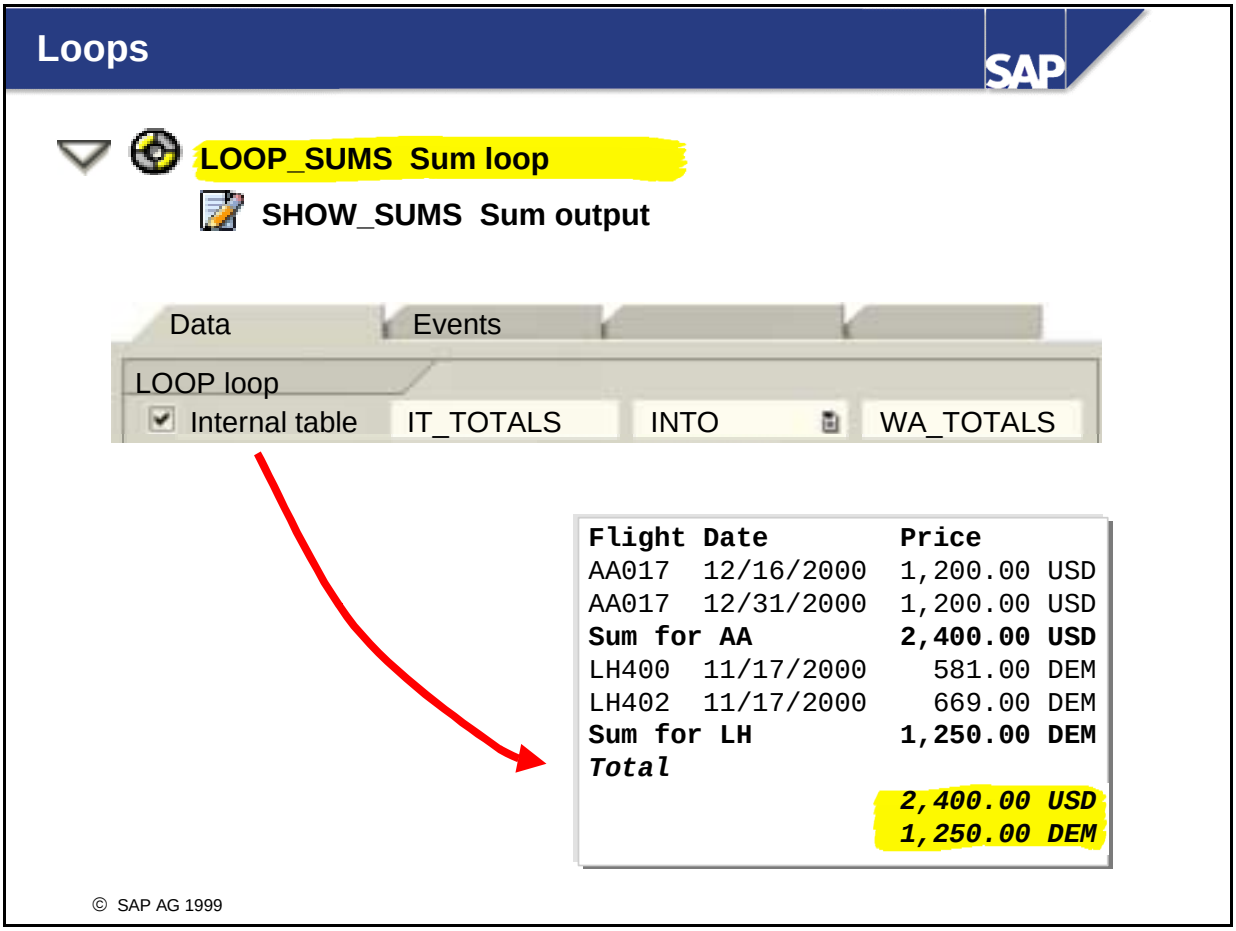
- Sometimes you want to evaluate the total number of pages of the currently processed form on the first page (e. g. with bar codes for envelope machines) or you need to print a total sum on the first page, although this sum is calculated in the course of the form processing. SAP Smart Forms provide you with the variable SFSY-FORMPAGES, but its value is known only at the end of the form processing and is then inserted into the form in a second processing round. Fields whose values are not known yet on the page that is currently processed cannot be evaluated in a normal window.
- As of R/3 Release 4.6C, Support Package 12, you can create windows that the composer will process only in a second round, i. e. at a time when even the values of those fields are known that are set on the last page.
- In order to achieve this late processing, here is what you should do:
  - Create those secondary windows that should be processed in the second round. In these windows you can use total sums or evaluate the total number of pages, e. g. in conditions.
  - On the first page that is processed, create a secondary window as the top node. In it, create a node of the type *Program lines*. In this node, call a subroutine for every window that needs late processing (see coding in the box above).
- For details, in particular on changes in the next release, please refer to OSS note 359009.



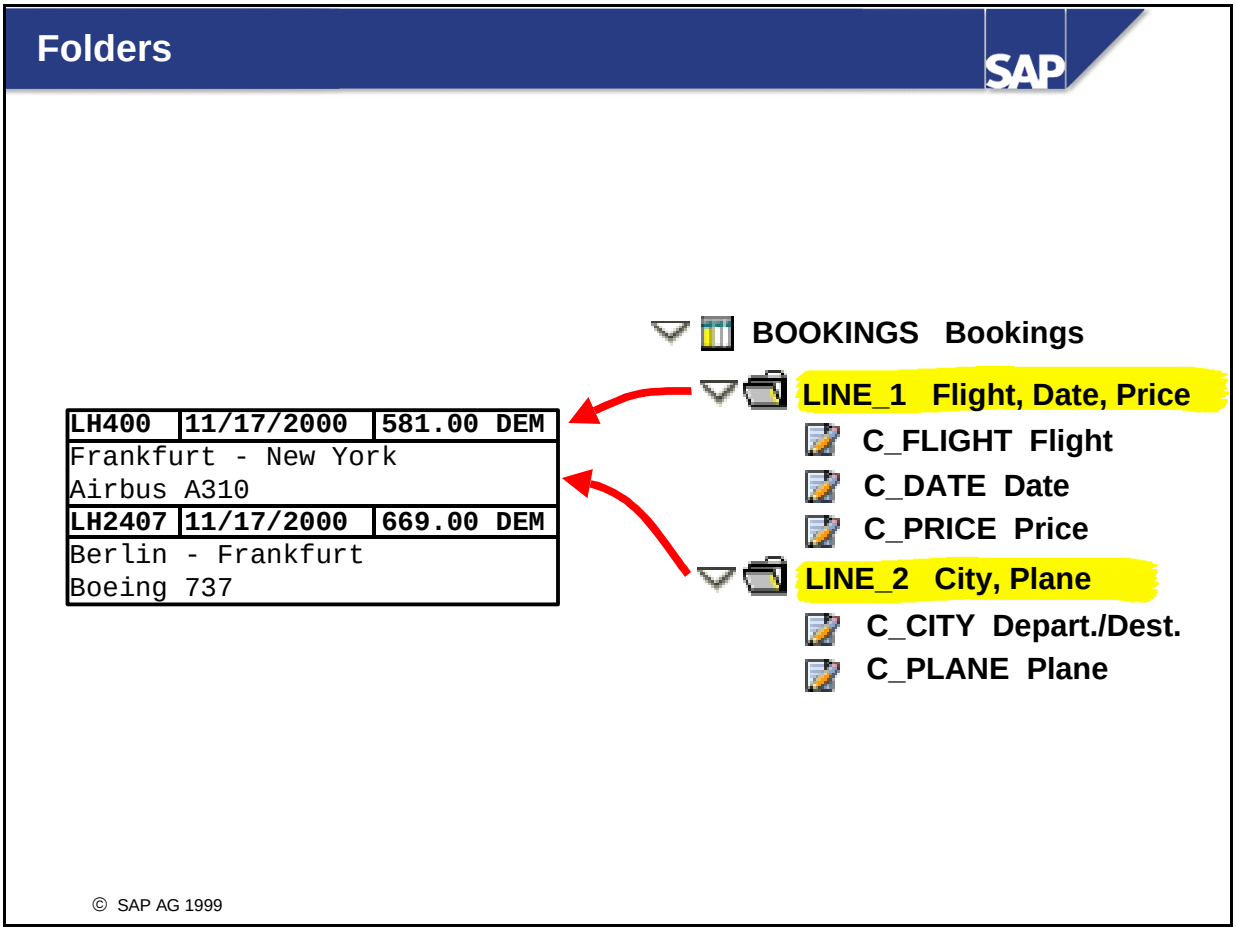
- If you have code to be used in different program lines nodes it makes sense to move this code into global subroutines (forms) and then call these as required. You create subroutines in the editor of the *Form routines* tab of the *Global definitions*.
- The syntax is normal ABAP syntax. You define the routines using FORM <form> [TABLES ...] [USING ...] [CHANGING ...]. You also use normal ABAP syntax to call a defined form using perform <form>. For more information refer to the ABAP documentation.
- Subroutines that you create in program lines are only known there and cannot be used in other program lines.



- 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 have entered on the *General attributes* tab of the page. Sometimes, 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 the page that 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.
  - *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 depths is reset to initial. Paragraph formats without outline attributes are ignored. See Unit 9 - *Smart Styles*.
  - *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. See Unit 10 - *Fonts and Bar Codes*.
  - You can also define free attributes for the spool request with values of your choice. These can be evaluated using table TSP02A. See OSS note 359379.

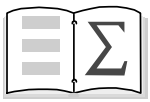


- A loop is very similar to a table. It has the same tabs - except for the *Table* tab. This means that for loops as well an internal table is read on a line-by-line basis. However, the output of the data is not predefined.
- *Data* tab:
  - Enter the name of the internal table over which the loop is to be executed as well as the 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. As with table nodes, the sorting of the internal table is a prerequisite for control levels.
- *Events* tab:
  - You can create a header and/or a footer for loops. Proceed in the same way as with tables.
- Loops can be nested. In particular, it is possible to use tables or templates in loops.
- Application areas of loops:
  - Output of an internal table containing amounts sorted by currency
  - Output of all bookings of all customers in one form



- The larger a form, the more complex its node hierarchy becomes. For clarity reasons, you can create folders and group nodes in these folders.
- Examples of the usage of folders:
  - Nodes of a table or a template which 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 should have 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 tables, folders have the *Events* tab. If you select the *Header* and/or *Footer* checkbox on this tab, one or two event nodes appear in the navigation tree. Then create your text nodes as subnodes of these event nodes. For more information, see the slide *Header and Footer* in Unit 6 - *Tables and Templates*.

Flow Control: Summary



You are now able to:

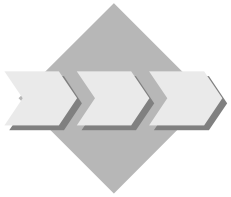
- **Explain how the various nodes can be used for flow control**
- **Create flow control nodes**

Exercises

Unit 7: Flow Control

At the conclusion of these exercises, you will be able to:

- Create program nodes
- Create node conditions
- Use alternative nodes
- Create loops
- Create command nodes
-



Your task: Make your invoice form more detailed and output the airport for each flight.

Grant a discount for American Airlines, assign the English-speaking customers to a special clerk and output the total of the invoice. Optionally, attach the general terms and conditions of business to your form and output the airport IDs.

Copy template for the form:	BC470_FLOWT (This form also contains all elements that you had to create/change in the optional parts of the previous exercise.)
Development class (for all exercises):	ZBC470_##
Name of the form to be created:	ZBC470_##_FLOWS
Model solution:	BC470_FLOWS
Application program for testing purposes:	SAPBC470_DEMO

1. Copy template

   Only use your existing form if you do not want to work through sections 5 and 6 (output of the totals and preventing page breaks). Otherwise, copy the exercise template BC470_FLOWT. This form already contains the totals calculations and outputs the subtotals.

2. Create discount for American Airlines bookings

   Output the following text after the American Airlines bookings: "Please note: A 2% discount applies to all American Airlines bookings."

   2-1 Create a text node called DISCOUNT at the appropriate position in the navigation tree. Specify paragraph format TB for this text node.

   2-2 Define the following condition for the text node: The text node should only be processed for American Airlines bookings, that is, only if WA_BOOKINGS-CARRID = 'AA'.

   2-3 Activate your form and test it using the program SAPBC470_DEMO.

3. Choose clerk

   Up to this point, only one clerk was responsible for creating the invoices. The travel agency Fly & Smile now hires a second clerk. Clerk A is responsible for the US and Canada, while clerk B is responsible for all other countries.

   3-1 Convert the global constant CLERK into a variable. (You do not have to make any changes in text nodes since they use CLERK already.)

   3-2 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.

   3-3 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.

4. **Optional:** Output totals by currency

   The copy template already contains the program lines node for calculating the totals. The totals are in the internal table IT_TOTALS. Each currency has its own line.

   4-1 Create a text node called TOTAL after the booking table. Output the text "Totals:" in this node.

   4-2 Create a loop node called TOTAL_LOOP for the totals. The data of the internal table IT_TOTALS should be copied line by line into the existing work area WA_TOTALS.

   4-3 Output the fields WA_TOTALS-FORCURAM (amount) and WA_TOTALS-FORCURKEY (currency) in a text node called CURRENCY_TOTAL. The amounts should be sorted by currency.
   Define the following formatting options for WA_TOTALS-FORCURAM:

   - Output length: 13
   - Decimal places: 2

   Select the paragraph format TO.

   4-4 **Optional:** Output the accumulated total (that is the total from the first page to the end of the respective page) on each page except the last page. You do not need to perform calculations since the totals, grouped by currency, are included in the table IT_TOTALS. (If you are interested, look at the program node ADD_TOTALS that is a subnode of the table BOOKINGS.)

   4-4-1 Create a footer for the booking table BOOKINGS.

   4-4-2 Repeat the steps you performed for the total for the text (PAGE_TOTAL) and the loop (LOOP_PAGE_TOTAL).

5. Prevent page break

   Ensure that the total and the greeting form are output on the same page. To do this, create a folder called TOTAL_FOLDER with the appropriate output option. (You output the total in exercise 4 that is an optional exercise. If you have not worked through this exercise, you must create a text node with a dummy total (without the actual total calculation) before the closing.

6. Create page for general terms and conditions of business

   Create an additional page called TERMS for the general terms and conditions of business. This page should be processed after the greeting form. Create a text node on this page and enter any text you like as the general terms and conditions of business.

7. **Optional:** Output airport IDs

   Output the IDs of the departure and the destination airport for each booking.

   7-1 Create a global field called AIRPFROM with reference to SPFLI-AIRPFROM for the departure airport and a global field called AIRPTO with reference to SPFLI-AIRPTO for the destination airport.

   7-2 Create a program lines node called SELECT_AIRPORTS in the table BOOKINGS in which you read the appropriate database fields AIRPFROM and AIRPTO from the table SPFLI into your global form fields AIRPFROM and AIRPTO. WHERE clause: CARRID = WA_BOOKINGS-CARRID.
   Check the program lines node.

   7-3 Change the line type POS in the Table Painter and add another cell for the airport IDs that are to be output after the flight date. The width of the four cells should be as follows: 1.80, 3.00, 3.80, and 6.70 cm.

   7-4 Create a text node called C_AIRPORTS for the airport IDs after the node C_DATE. We recommend that you copy the node C_DATE. This ensures that you have the correct formatting. Adjust the description. Output the two global fields AIRPFROM and AIRPTO in the text node C_AIRPORTS.

   7-5 Adjust the heading of the table BOOKINGS by adding a suitable text node.

   7-6 Create a folder called TABLE_LINE for the four nodes of the table lines to make the navigation tree less complex.

8. Activate your form and test it using the program SAPBC470_DEMO.

Solutions

Unit 7: Flow Control

1. Copy template

   See the exercise in Unit 3.

2. Create discount for American Airlines bookings

   2-1 Create a new text node called DISCOUNT as a successor node of C_PRICE (the text node containing the price for each booking). Enter text ("Please note ...") in the editor (*General attributes* tab of the text node) and assign the paragraph format TB to this text.

   2-2 Enter the following on the *Conditions* tab: WA_BOOKINGS-CARRID = 'AA'. (Note that you must write the carrier in upper case.)

3. Choose clerk

   3-1 In the global definitions of the form, deselect the *Constant* field for the CLERK variable on the *Global data* tab.

   3-2 Your task is to create an alternative with program lines and determine the clerk in these program lines. 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 node INTRODUCTION.
   You enter the two conditions on the *General attributes* tab of the alternative. WA_CUSTOMERS-COUNTRY = 'US' and WA_CUSTOMERS-COUNTRY = 'CA'. 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 of the maintenance screen.)

   3-3 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. Set a value for the variable CLERK (in the ABAP editor of the *General attributes* tab), for example: CLERK = 'Mr. Miller'. Declare the variable CLERK 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.

4. **Optional:** Output totals by currency

   4-1 Create the text node TOTAL using the context menu of the text node GREETINGS. (If you created this node using the context menu of the table BOOKINGS, you would create a subnode instead of a successor node.) Use Drag & Drop (left mouse button pressed) to move the node up so that it becomes a **successor** node of the table BOOKINGS. Enter the text ("Totals:") in the editor of the text node (on the *General attributes* tab).

4-2 Create the loop node TOTAL_LOOP using the context menu of the text node you have just created. Select the *Internal table* checkbox on the *Data* tab of the loop node and enter the following data into the fields on the right side: IT_TOTALS INTO WA_TOTALS.

4-3 Use the context menu of the loop to create a text node called CURRENCY_TOTAL and enter the two fields WA_TOTALS-FORCURAM and WA_TOTALS_FORCURKEY by means of the field list.
To determine the output length, place your cursor on the field WA_TOTALS-FORCURAM in the editor. Click the *Change field* pushbutton (which is the third pushbutton from the right in the editor toolbar). On the dialog box that appears change &WA_TOTALS-FORCURAM& into &WA_TOTALS-FORCURAM(13.2)&.
Choose the appropriate paragraph format from the selection list of paragraph formats in the editor.
To ensure that the totals are output by currency, you must sort the internal table IT_TOTALS. Enter FORCURKEY as the sort criterion on the *Data* tab.

4-4 **Optional:** Output accumulated totals

4-4-1 Create a footer for the table BOOKINGS by selecting the *Footer* field on the *Events* tab. You must determine the height. Enter 3 cm. Since you want to output the footer only at the end of the page and not again at the end of the table (where the totals are displayed), you must only select *at page break* as the *Output event*.

4-4-2 Since the totals are automatically included in the internal table IT_TOTALS, you must not make any special calculations. Create a text node called PAGE_TOTAL as the subnode of the footer and enter text ("Total up to and including this page:"). Then create a loop called PAGE_TOTAL_LOOP and make the same entries as for the loop TOTAL_LOOP. Handle the text node CURRENCY_PAGE_TOTAL in the same way as the text node for the total (CURRENCY_TOTAL).

5. Prevent page break

Use the context menu of the text node GREETINGS to create a folder called TOTAL_FOLDER. Drag the text nodes GREETINGS and TOTAL as well as the loop TOTAL_LOOP into this folder. (If you have not worked through the optional exercise 4, you must create a text node with a dummy total and drag this node together with the GREETINGS text node into the folder.) Select the *Page protection* checkbox on the *Output options* tab of the folder.

6. Create page for general terms and conditions of business

Use the context menu of the page NEXT to create the page TERMS. To prevent an endless loop, you must ensure that no next page is entered on the *General attributes* tab.
There are several ways how you can proceed. You could do the following, for example:
Copy the main window from the page FIRST to the page NEXT. Create a command node after the folder TOTAL_FOLDER, select the *Go to new page* checkbox and enter TERMS as the page.
Create a text node after the command node to enter your terms and conditions of business.

7. **Optional:** Output airport IDs

   7-1 Create the two fields on the *Global data* tab of the global definitions.

   7-2 The code for the program lines is as follows:

```
SELECT SINGLE airpfrom airpto
  FROM spfli
  INTO (airpfrom, airpto)
  WHERE carrid = wa_bookings-carrid AND
        connid = wa_bookings-connid.
```

   You must enter WA_BOOKINGS as the input parameter and AIRPFROM and AIRPTO as the output parameters.

   7-3 Proceed as described in exercise 2 of unit 6.

   7-4 Copy the node C_DATE. Enter the two global fields AIRPFROM and AIRPTO in the editor using the field list.

   7-5 Add a new text node called H_AIRPORTS below the node H_DATE. Enter the text "Departure/Destination" in the editor.

   7-6 Create a folder after the program lines node ADD_TOTALS and move the text nodes C_FLIGHT, C_DATE, C_AIRPORTS and C_PRICE into that folder.

Integration into Application Programs

BC470 Form Printing with SAP Smart Forms

1 Course Overview
2 SAP Smart Forms: Overview
3 First Steps with the SAP Form Builder
4 Texts, Addresses, and Graphics
5 Data in Forms
6 Tables and Templates
7 Flow Control
8 **Integration into Application Programs**
9 Smart Styles
10 Fonts and Bar Codes
... Appendix

Integration into Application Programs: Contents

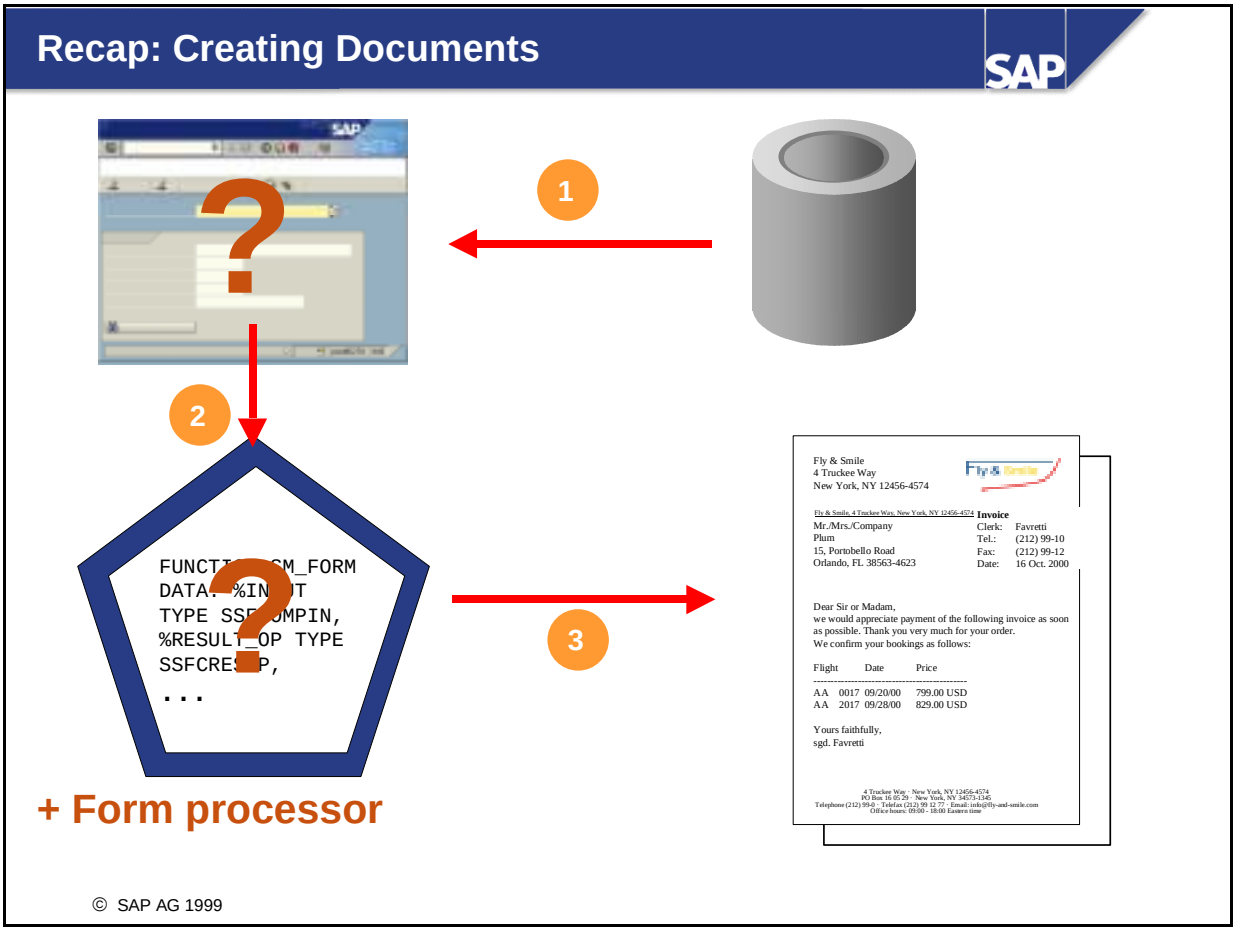
Contents:

- Generated function module
- Customizing the application program

Integration into Application Programs: Unit Objectives

At the conclusion of this unit, you will be able to:

- **Explain the interaction between application programs and SAP Smart Forms**
- **Fill the interface of the generated function module with parameters**



- Here is a recap of the document creation process:

  1. The transaction looks up in Customizing which program to call. This program then reads the data.
  2. The transaction learns in Customizing which SAP Smart Form to use for the scenario chosen, calls the appropriate function module generated and thus triggers the form processing process. The interface is filled with the data read.
     When the form processing process is started, the form processor (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.

Components of the Application Program

a) Data retrieval

b) Name of generated function module?

c) Call of function module

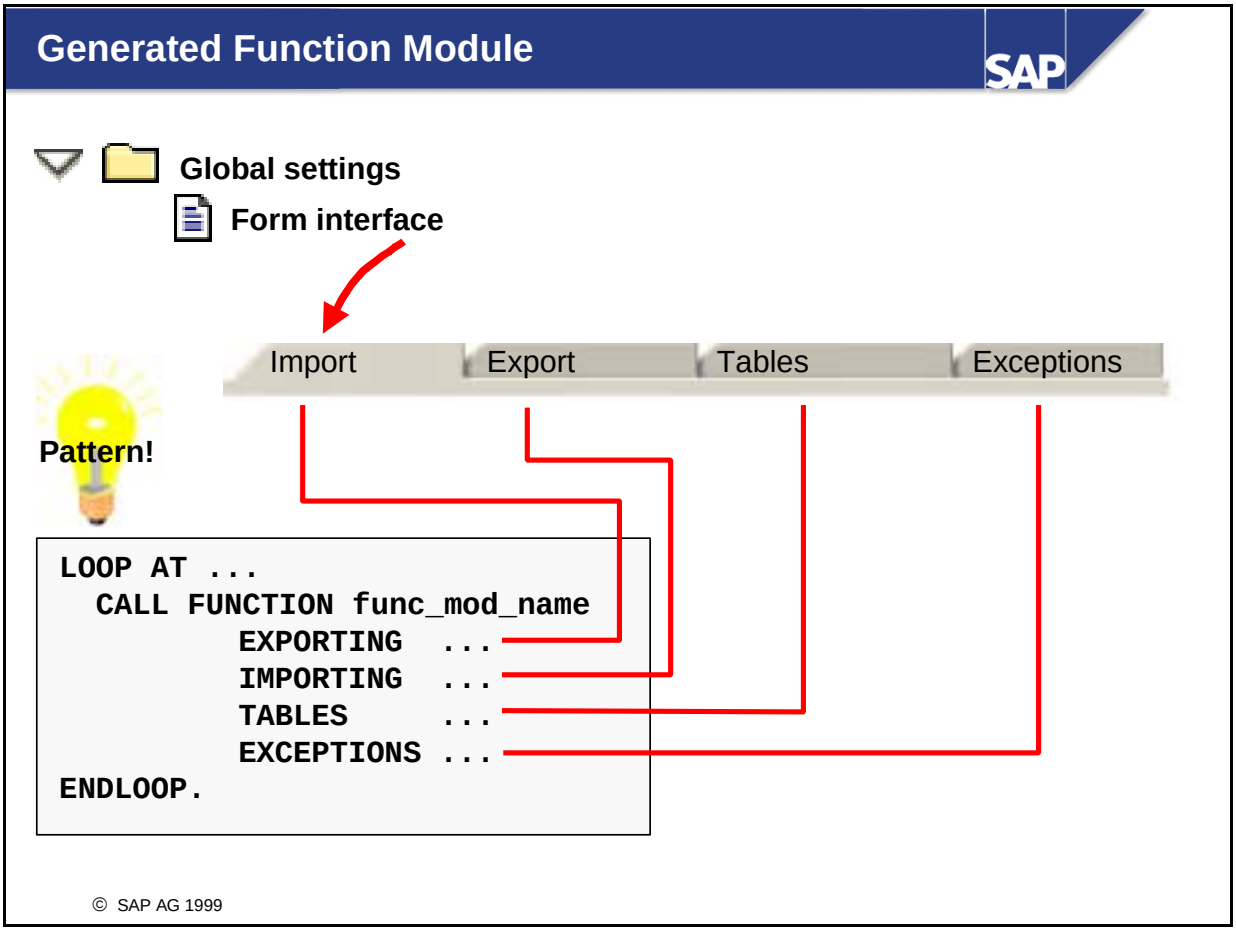
```
PROGRAM ...
DATA:
   ssf_name      TYPE tdsfname,
   func_mod_name TYPE rs38L_fnam.

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.
```


- From the perspective of the SAP Smart Forms, the application program consists of three parts:
  a) The data is selected from the database - which is by far the most comprehensive part.
  b) 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`.
  c) Actual form processing starts. The generated function module is called once for each document to be created, for example, once for each customer for whom you want to create an invoice.



- 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 pattern button 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 pass the name of the function module in the pattern 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 that is filled by calling `SSF_FUNCTION_MODULE_NAME` and contains the current name of the generated function module at runtime.

Interface Parameters

```
CALL FUNCTION func_mod_name
  EXPORTING
*   control_parameters    =
*   output_options        =
*   user_settings         = 'X'
    wa_customers          =
* IMPORTING
*   job_output_info       =
*   job_output_options    =
  TABLES
    it_bookings           =
* EXCEPTIONS
*   formatting_error      = 1
*   others                = 2.

IF sy-subrc <> 0.
* ... Error handling
ENDIF.
```

Optional default parameters

Required additional parameters

- The generated function module has both required and optional parameters:
  - Required parameters: You have entered these in the SAP Form Builder. These are application data to be output or used for calculations in the form.
  - Optional parameters. These exist for each form. They are not ready for input on the form interface tabs of the SAP Form Builder. They include:
    - `control_parameters`: General output control (see next slide)
    - `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.
    - `archive_index, archive_index_tab, archive_parameters`: Parameters for archiving
    - `mail_appl_obj, mail_recipient, mail_sender`: Parameters of the Business Comunication 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 (for example, with XML output)

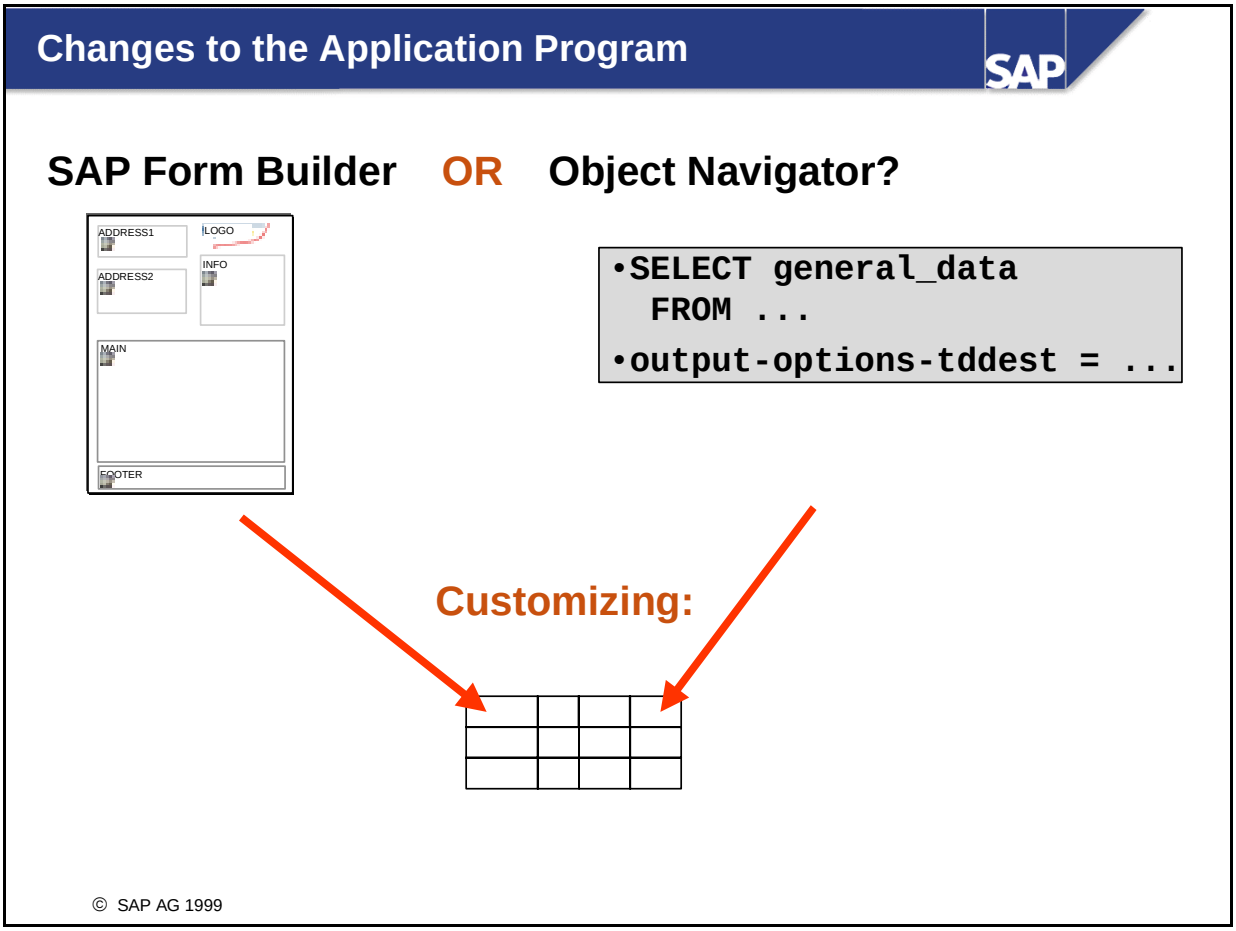
- You handle errors as with other function modules by querying the return code (sy-subrc) directly after the call of the function module.

Control Structure CONTROL_PARAMETERS

CONTROL_PARAMETERS
(Export parameters of the function module generated)
Type: **ssfctrlop**

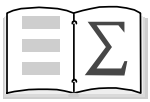
no_open	No new spool request
no_close	Do not close spool request
device	Output device **('PRINTER', 'TELEFAX', 'MAIL')**
no_dialog	No dialog box for output
preview	Print preview
langu	Language
startpage	Start page ≠ default

- 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.`



- 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, for example, disable the print preview.
- The transaction determines which part of the application program you actually need to change (for example, a function module or a subroutine). As a rule, you should change original SAP programs only in exceptional cases.
- If you want to use your own modified copies instead of the SAP originals (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.

Integration into Application Programs: Summary



You are now able to:

- **Explain the interaction between application programs and SAP Smart Forms**
- **Fill the interface of the generated function module with parameters**

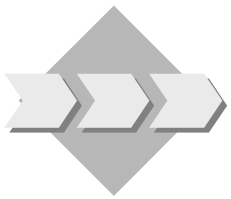
Exercises

Unit 8: Integration into Application Programs

Optional

At the conclusion of these exercises, you will be able to:

- Insert the generated function module for the invoice form into a program
- Set output options for the function module (optional)
-



Your task: Add appropriate lines of code to the body of an application program so that the invoice form BC470_FLOWS is processed.

Copy template for the program:	SAPBC470_PROGT
Name of the program to be created:	ZBC470_##_PROGS
Development class (for all exercises):	ZBC470_##
Name of the form to be used:	BC470_##_FLOWS
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.

2. Determine the function module name

   2-1 The program must know the name of the function module generated for the form BC470_FLOWS. Call the function module SSF_FUNCTION_MODULE_NAME at the end of the program code. To do this, use the pattern button. This ensures that the interface is correct.

2-2 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.

3. Call the generated function module in a loop

   3-1 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.

   3-2 Call the generated function module in this loop. To do this, follow these steps:

      3-2-1 In the SAP Form Builder, determine the name of the generated function module for the form BC470_FLOWS (by choosing *Environment -> Function module name* from the menu) and use CTRL-Y and CTRL-C to copy the name to the clipboard.

      3-2-2 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.

      3-2-3 Fill the required interface parameters WA_CUSTOMERS, COLOR, and IT_BOOKINGS with the identically named parameters of the program (these exist already and are filled with values).

   3-3 Test your program.

4. **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 the 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*.

5. **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 (see the corresponding slide in this unit).

Solutions

Unit 8: Integration into Application Programs

The optional parts are printed in bold.

```
*&---------------------------------------------------------------------*
*& Report  SAPBC470_PROGS
*&---------------------------------------------------------------------*
*& Solution for the exercise of unit 8 of BC470
*&---------------------------------------------------------------------*

REPORT  sapbc470_progs.
TABLES: spfli, scustom.

* selection-screen
SELECTION-SCREEN COMMENT 1(45) text-se1.

SELECTION-SCREEN SKIP 1.
SELECT-OPTIONS:
  so_cust FOR scustom-id   DEFAULT 1    TO 3,
  so_carr FOR spfli-carrid DEFAULT 'AA' TO 'LH'.

* printing options
SELECTION-SCREEN SKIP 1.
PARAMETERS:
  pa_prnt(4) DEFAULT 'P280'.           " printer

* 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'.
```

```
DATA:
  it_bookings         TYPE ty_bookings,                  "#EC NEEDED
  it_customers        TYPE ty_customers,                 "#EC NEEDED
  wa_customers        TYPE scustom,                      "#EC NEEDED
  color(4).                                              "#EC NEEDED


DATA:
    func_mod_name TYPE rs38l_fnam.

DATA:
  output_options TYPE ssfcompop,       " optional part of exercise
  control_parameters TYPE ssfctrlop.   " optional part of exercise

*********************************************************************
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
    ORDER BY PRIMARY KEY.

  SELECT * FROM  sbook
    INTO TABLE it_bookings
    FOR ALL ENTRIES IN it_customers
    WHERE customid = it_customers-id 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_FLOWS'
       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'.
      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.
* make sure the spool request is re-used
     control_parameters-no_open = 'X'.
  ENDLOOP.
```

Smart Styles

BC470 Form Printing with SAP Smart Forms

1 Course Overview
2 SAP Smart Forms: Overview
3 First Steps with the SAP Form Builder
4 Texts, Addresses, and Graphics
5 Data in Forms
6 Tables and Templates
7 Flow Control
8 Integration into Application Programs
9 Smart Styles
10 Fonts and Bar Codes
Appendix

Smart Styles: Contents

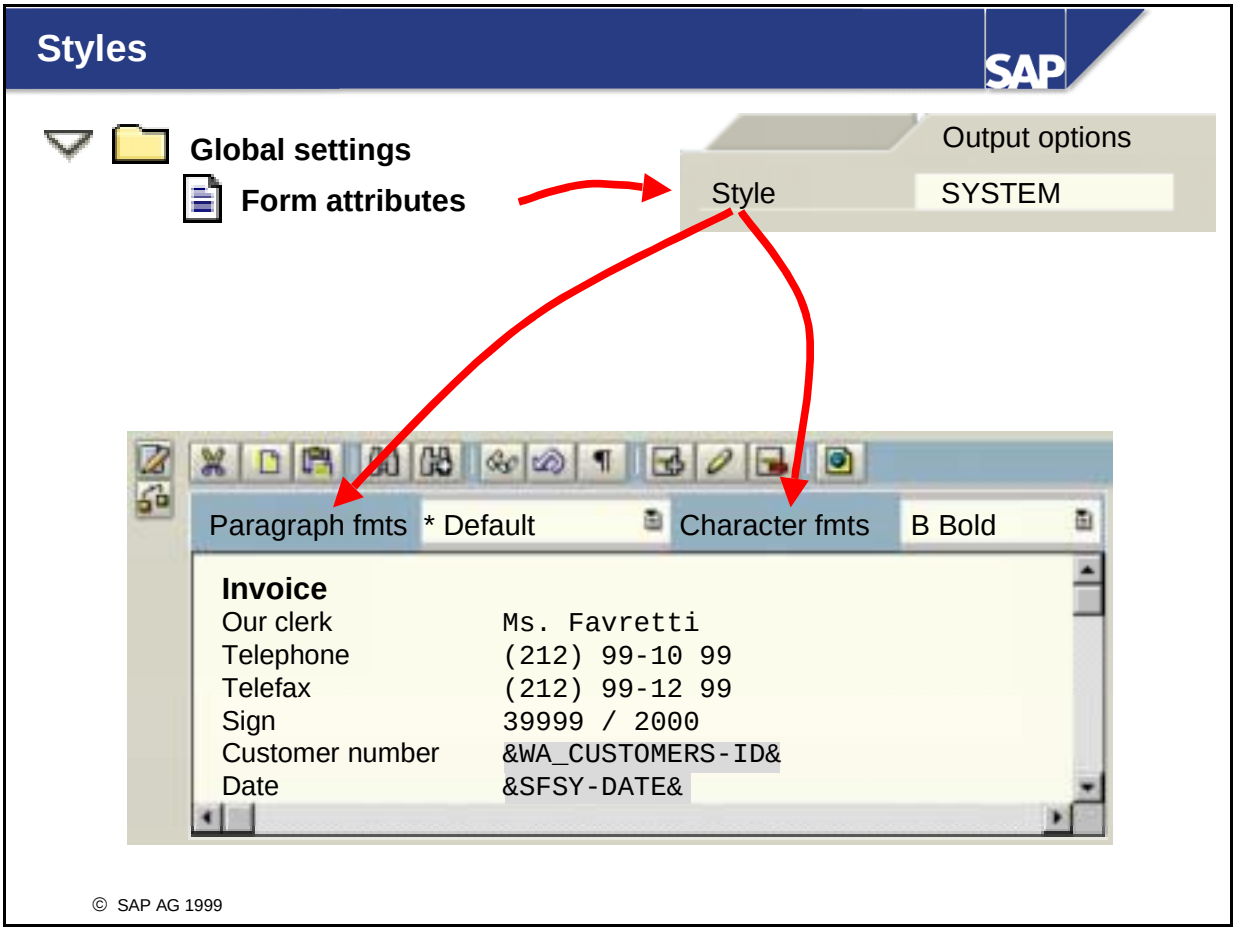
Contents:

- **Smart Styles**
- **Style Builder**

Smart Styles: Unit Objectives

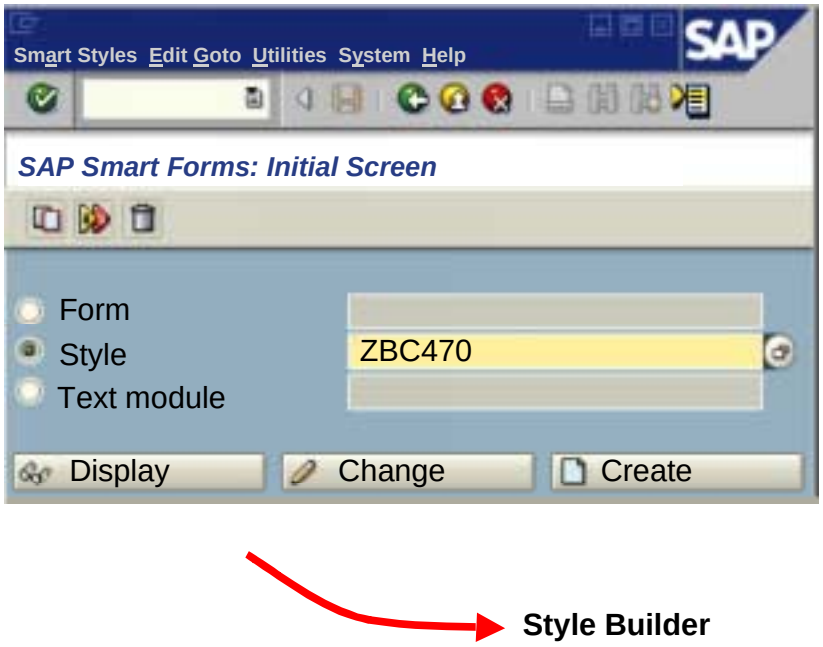
At the conclusion of this unit, you will be able to:

- **Use the Style Builder to create/change Smart Styles**
- **Create paragraph and character formats**
- **Use formats in texts**

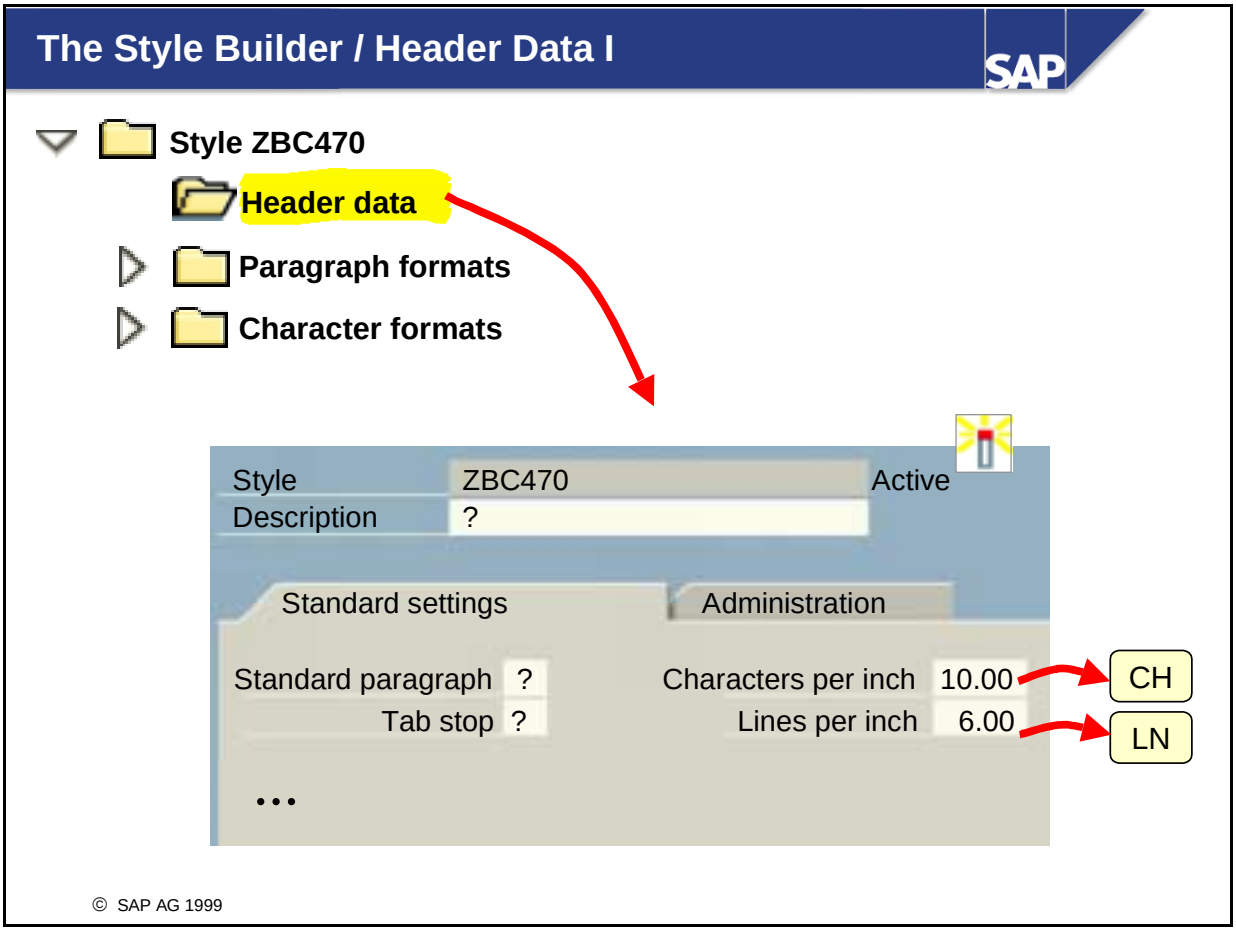


- If you want to format texts in a form, select a character and/or paragraph format from the format lists in the editor. The formats offered in this list depend on the style (Smart Style) chosen. The following rules apply:
  - Each form must be assigned a style (on the *Output options* tab of the form attributes). The default for new forms is the style *System*.
  - You can assign a different style to most nodes (on the *Output options* tab). This style then applies to all lower-level nodes.
- Styles are collections of character and paragraph formats and 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.
- Please note that SAPscript styles of include texts are not recognized irrespective of whether they are set dynamically or statically. If formats are used in an include text not defined in the Smart Style which applies to that include text, then these formats are ignored.

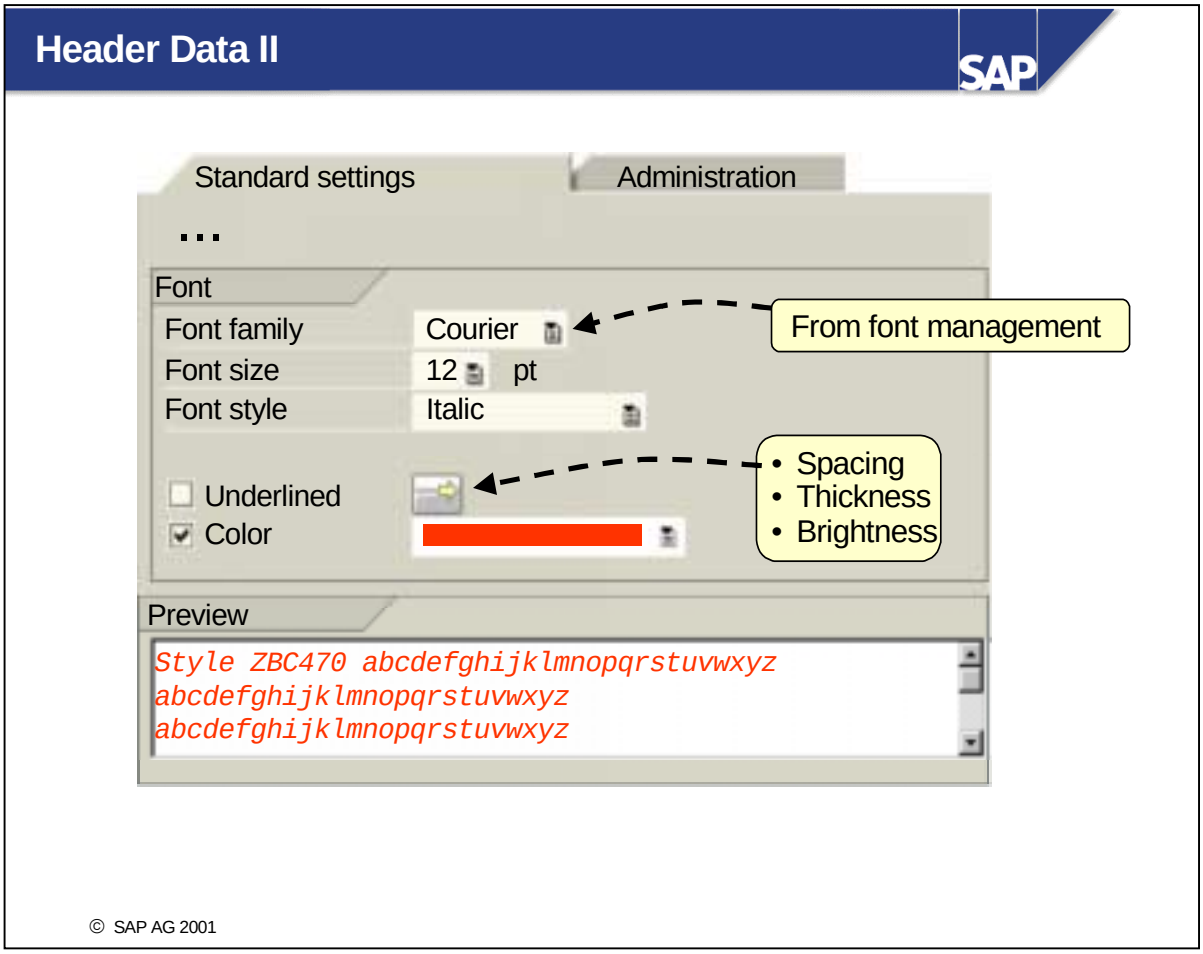
Smart Styles: Initial Screen



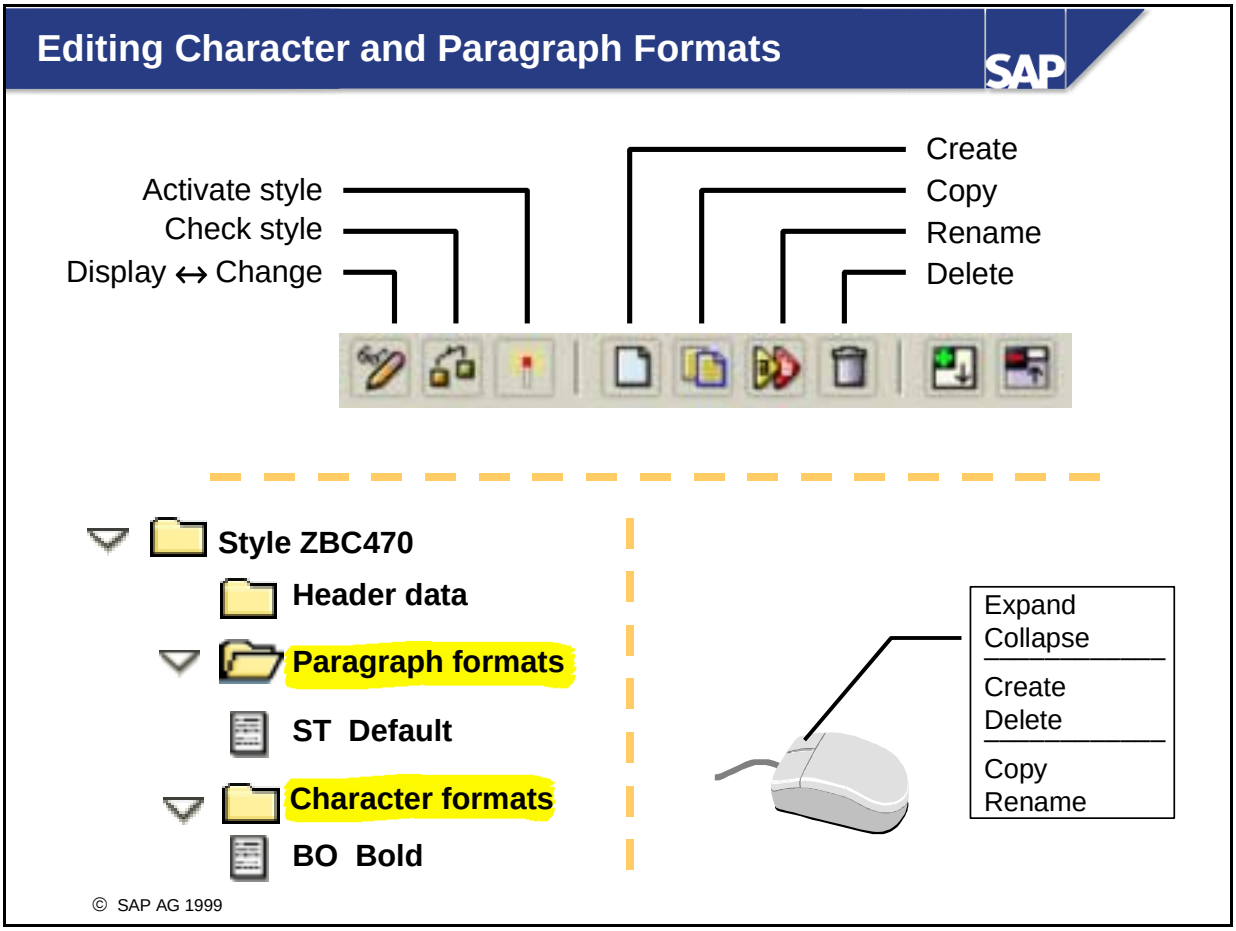
- To create a Smart Style, you can either select the *Style* radio button on the initial screen of the SAP Smart Forms transaction, or start transaction SMARTSTYLES directly. Enter the name of a style. We recommend that you only change styles in your customer namespace, that is, styles beginning with 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 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. The style variant is of no relevance in Release 4.6C.



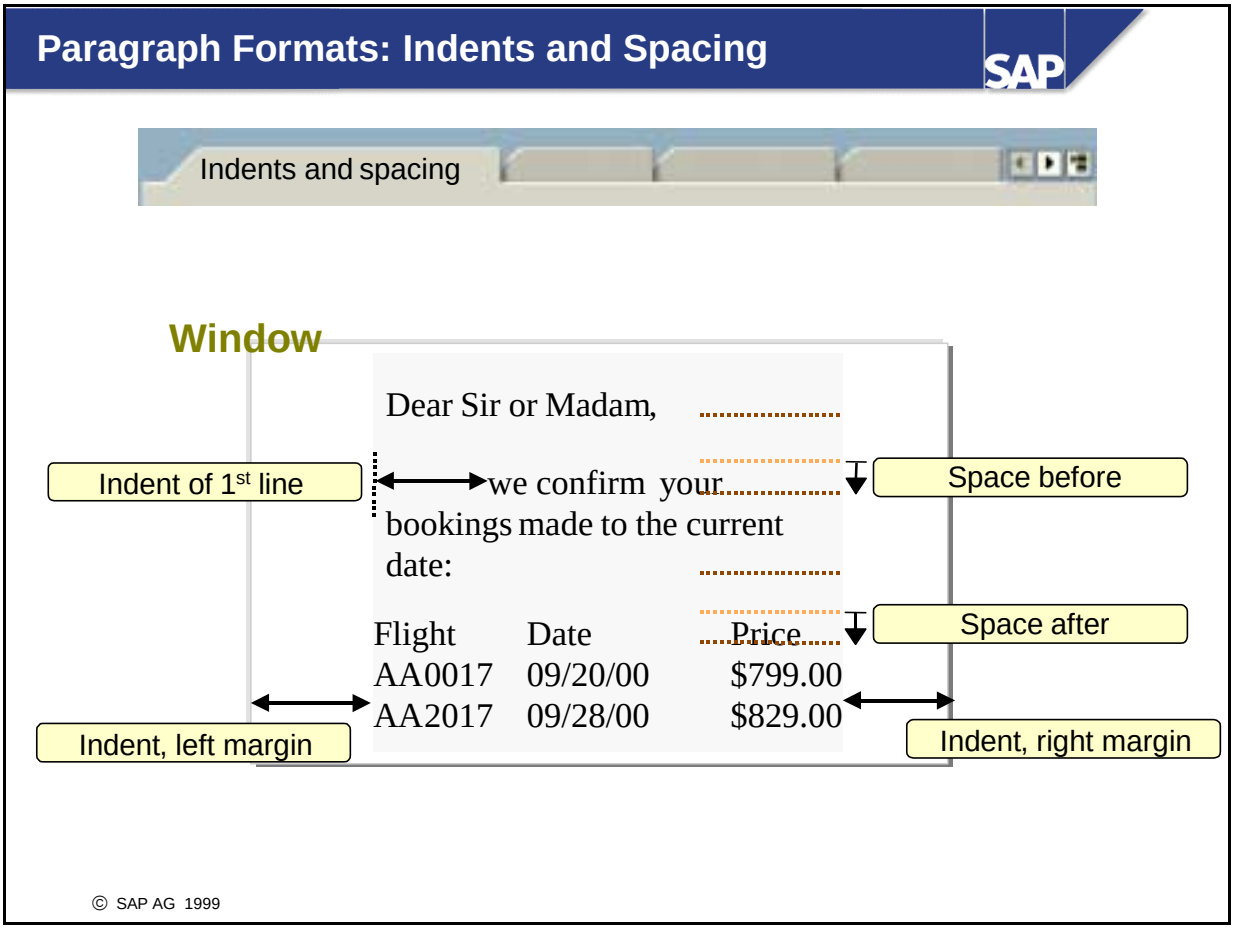
- The maintenance tool for Smart Styles, called the Style Builder, consists of two areas:
  - On the left-hand side is the navigation tree.
  - On the right-hand side you see detailed information on the element you select in the tree. You may also see a preview of the element there.
- Like forms, styles can exist in an active and an inactive version. You activate a style by clicking the corresponding pushbutton or by choosing *Style* → *Activate*.
- The header data has two tabs: *Standard settings* and *Administration*. On the above slide you can see some of the *Standard settings*:
  - *Standard paragraph*: Here you specify the paragraph format with which 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 the description of the style) if you want to activate the style. Before you can make a selection here, you must have defined at least one paragraph format.
  - *Tab stop*: Here you set the space between the standard tab stops. (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.



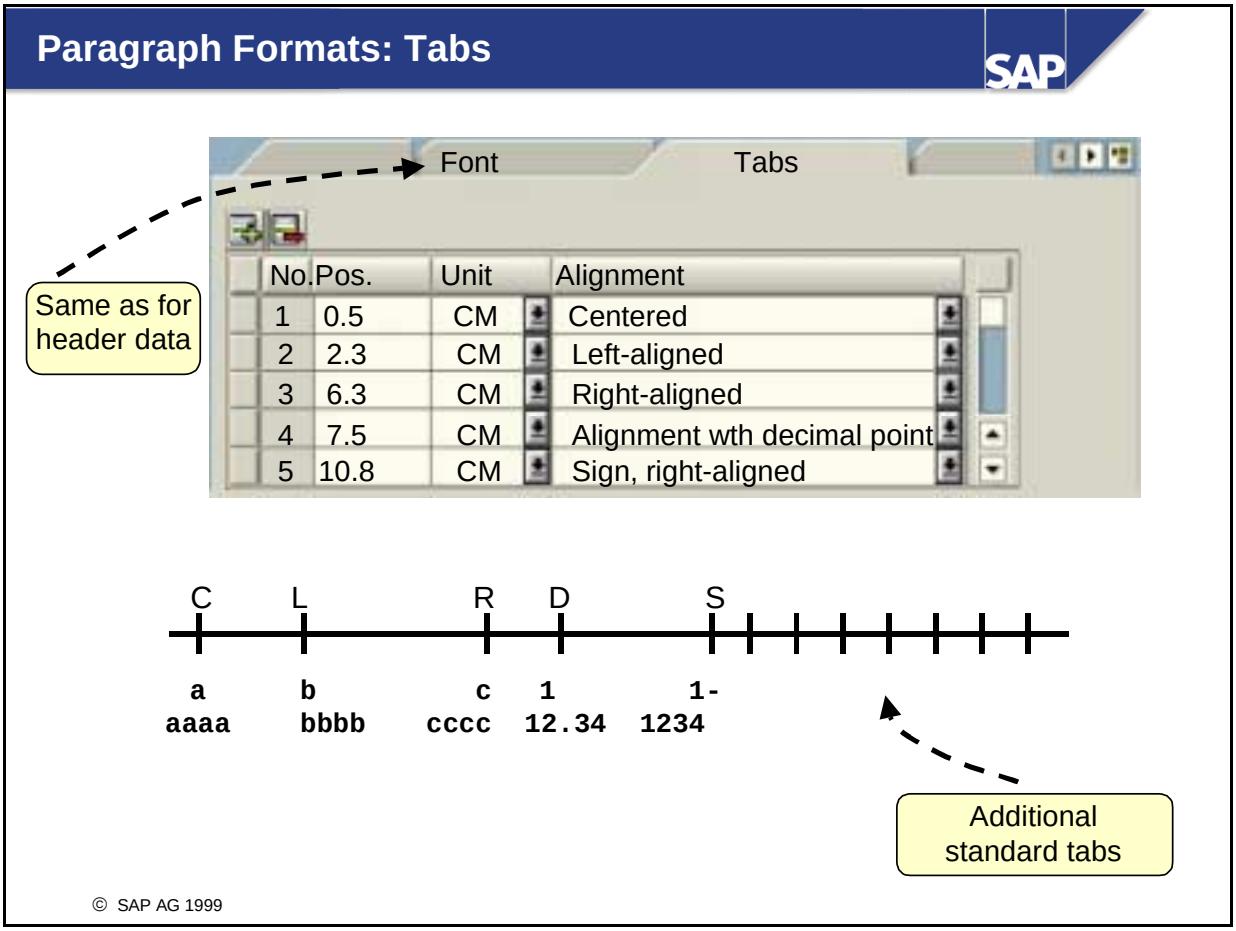
- The slide above shows the remaining options of the *Standard settings* tab. The selections you make in the *Font* group box apply to all paragraph and character formats of the style - unless you explicitly specify individual font attributes for these formats.
- The *font families* available depend on the settings in font maintenance. See Unit 10 - Fonts and Bar Codes.
- The *font size* determines the height of the font in points (PT). The default 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, however, 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 printed 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% corresponds to the full hue.
  - 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 *Standard formats allowed* checkbox has no function in Release 4.6C.



- 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.

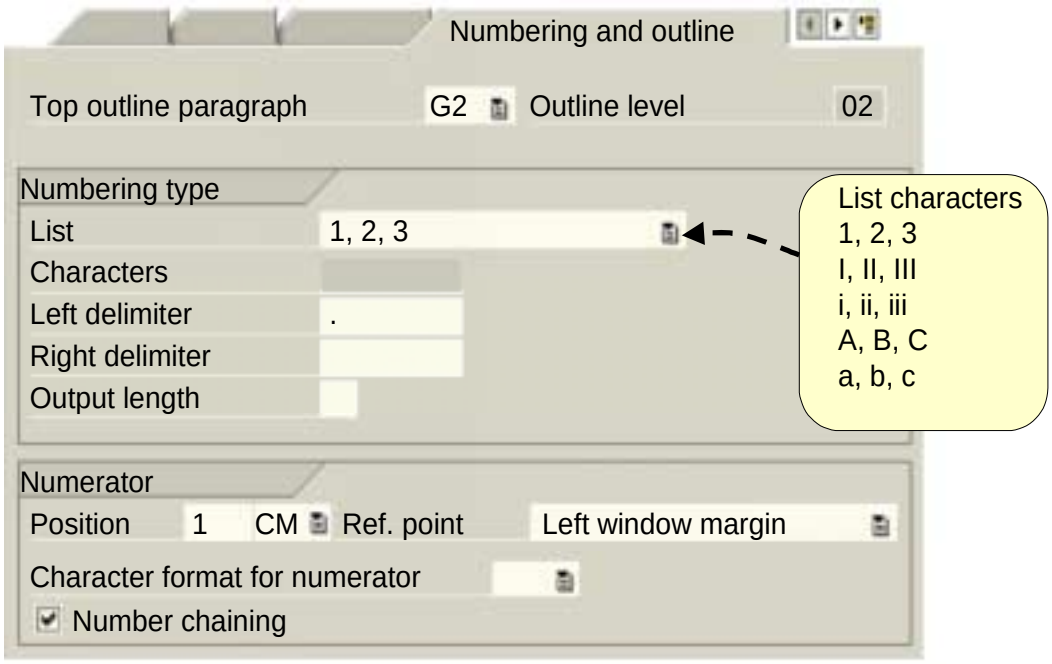


- On the *Indents and spacing* tab of a paragraph format, you can make the following settings:
  - *Indent:*
    - *Left / right margin of indent*: Amount of space between the window and the text.
    - As far as the left margin is concerned, you can determine a different indent for the first line of a paragraph.
  - *Spacing:*
    - *Space before*: This is the space by which the current paragraph is moved down. Similarly, the *space after* defines the space by which the next paragraph is moved down.
    - *Line spacing*: Please note that the line spacing is not adjusted automatically if you use a larger font size. To avoid overlappings, you must change the line spacing if required.
  - *Text flow:*
    - You can determine that the text of a paragraph should always be printed on the same page (*Page protection* checkbox). If a paragraph does not fit entirely onto one 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.



- The *Font* tab has the same fields as the *Standard settings* tab of the header data. If you do not make any settings here, the system uses the settings of the header data: *Font family*, *Font size*, *Font style*, *Underlined*, *Color*. You can override the header data by specifying a different font. Please 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 you define individual tab stops for the paragraph format.
  - The *Sign* alignment type lets you define numbers right aligned at the tab stop position, taking into account 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 their decimal points being printed at the tab stop position.
  - After the last tap stop, the standard tab stops of the standard settings (header data) are used.

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 (we recommend 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. For an example see the next slide.
- 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 numbers (lower case or upper case) or letters (lower case or upper case). You can also specify a *left* and a *right delimiter* such as a round 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. (The reference point is always *Left window margin* in Release 4.6C). Make sure that the paragraph margin is not affected by the numerator margin. To avoid overlappings, define a paragraph margin that is large enough.

- If you select *Number chaining*, the system uses the previous numerator for multi-level numberings. Example: If level G1 = 1, then level G2 (with number chaining) = **1.**2.

Paragraph Formats: Numbering and Outline II

Example: Paragraph definition

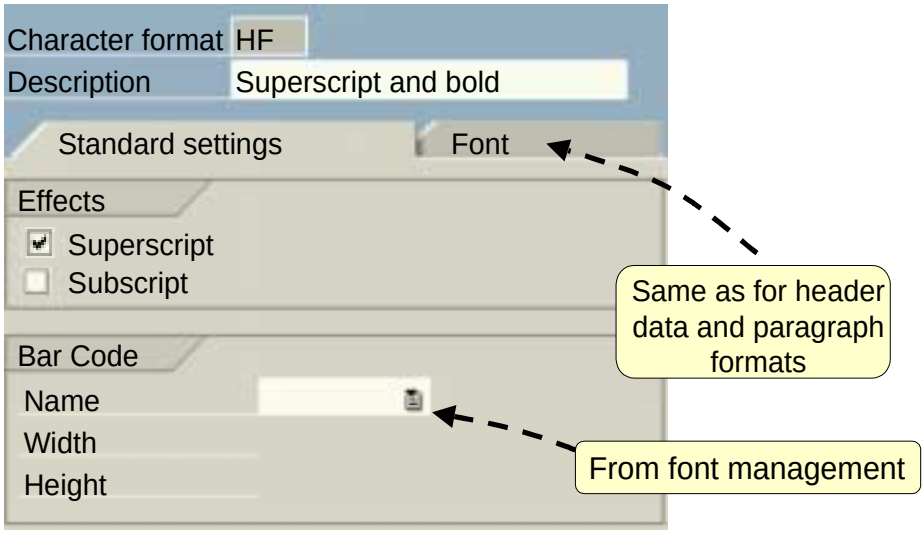
	List	Left delimiter	Right delimiter	Number chaining	Position
G1	1, 2, 3				
G2	1, 2, 3	.		☑	0.5 cm
G3	a, b, c	.	)	☑	1 cm

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

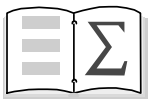
- Here you see an example of a three-level outline. G1 is the top level.
- Outlines and numberings are **not** shown in exact WYSIWYG mode in the preview of the editor. For a real WYSIWYG display, you need to test the function module of the form.
- If you want to reset the numbering of a paragraph to its initial value, create a command node in your form and use the command *Reset paragraph numbering*. See Unit 7 - *Flow Control*.

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 header data 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, Color*. Please 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 *bar code*.
  - The height and the width of a bar code are automatically adopted from font maintenance.
  - You cannot combine the *Superscript* or *Subscript* attribute with a bar code.
  - Bar codes 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.

Smart Styles: Unit Summary



You are now able to:

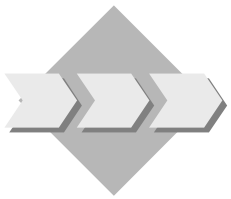
- **Use the Style Builder to create/change Smart Styles**
- **Create paragraph and character formats**
- **Use formats in texts**

Exercises

Unit 9: Smart Styles

At the conclusion of these exercises, you will be able to:

- Maintain Smart Styles



Your task: Copy and enhance your existing style and use it in the invoice form.

Copy template for the style:	BC470
Name of the style to be created:	ZBC470_##_STYLS
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 task in unit 7 where you had to create the TERMS page. Otherwise, copy the form BC470_FLOWS to ZBC470_##_STYLS.

Model solution for the style:	BC470_STYLS
Application program for testing purposes:	SAPBC470_DEMO

1. Copy template

   Copy the style BC470 to ZBC470_##_STYLS.

2. Change standard settings

   Set Times, 12 pt as the standard font type.

3. Change and create paragraph formats

3-1 Specify for the paragraph format AS that the lines of a paragraph should not be separated by a page break.

3-2 Use the standard font family for the paragraph format AS.

3-3 For the paragraph format TO, change the positions for the two tab stops to 13 and 14 cm.

3-4 **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

IIa) This paragraph has format G2

IIb) This paragraph has format G2

III This paragraph has format G1

Note: The preview does not support WYSIWYG.

4. Create at least one character format of your choice.
5. Activate your style.
6. 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.)
7. Test your form using the program SAPBC470_DEMO.

Solutions

Unit 9: Smart Styles

1. Copy template
   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 clicking the *Copy* pushbutton (which is the leftmost button in the toolbar). Enter the new name (ZBC470_##_STYLS) as the target style, and assign your development class on the dialog box that is displayed next.

2. Change standard settings
   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.

3. Change and create paragraph formats

   3-1 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.

   3-2 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.

   3-3 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.

   3-4 **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 numbering"). On the *Indents and spacing* tab, determine a left margin of your choice, for example, 1 cm. On the *Numbering and outline* tab, enter G1 as the top outline paragraph. Choose Roman letters in upper case as the numbering type.

   Use the context menu of the paragraph format G1 to create the paragraph format G2. Enter a description.
   On the *Indents and spacing* tab, determine a left margin of your choice, for example, 1.2 cm.
   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 lower case 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 0.5 cm as the position for the numerator (reference point: left window margin).

4. Create at least one character format of your choice.
   You can do this, for example, 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.

5. Activate your style by clicking the *Activate* pushbutton (which is the third pushbutton from the left).
6. Go to the SAP Form Builder (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.
7. Test your form using the program SAPBC470_DEMO.

Fonts and Barcodes

BC470 Form Printing with SAP Smart Forms

1 Course Overview
2 SAP Smart Forms: Overview
3 First Steps with the SAP Form Builder
4 Texts, Addresses, and Graphics
5 Data in Forms
6 Tables and Templates
7 Flow Control
8 Integration into Application Programs
9 Smart Styles
10 Fonts and Bar Codes
Appendix

Fonts and Bar Codes: Content

Contents:

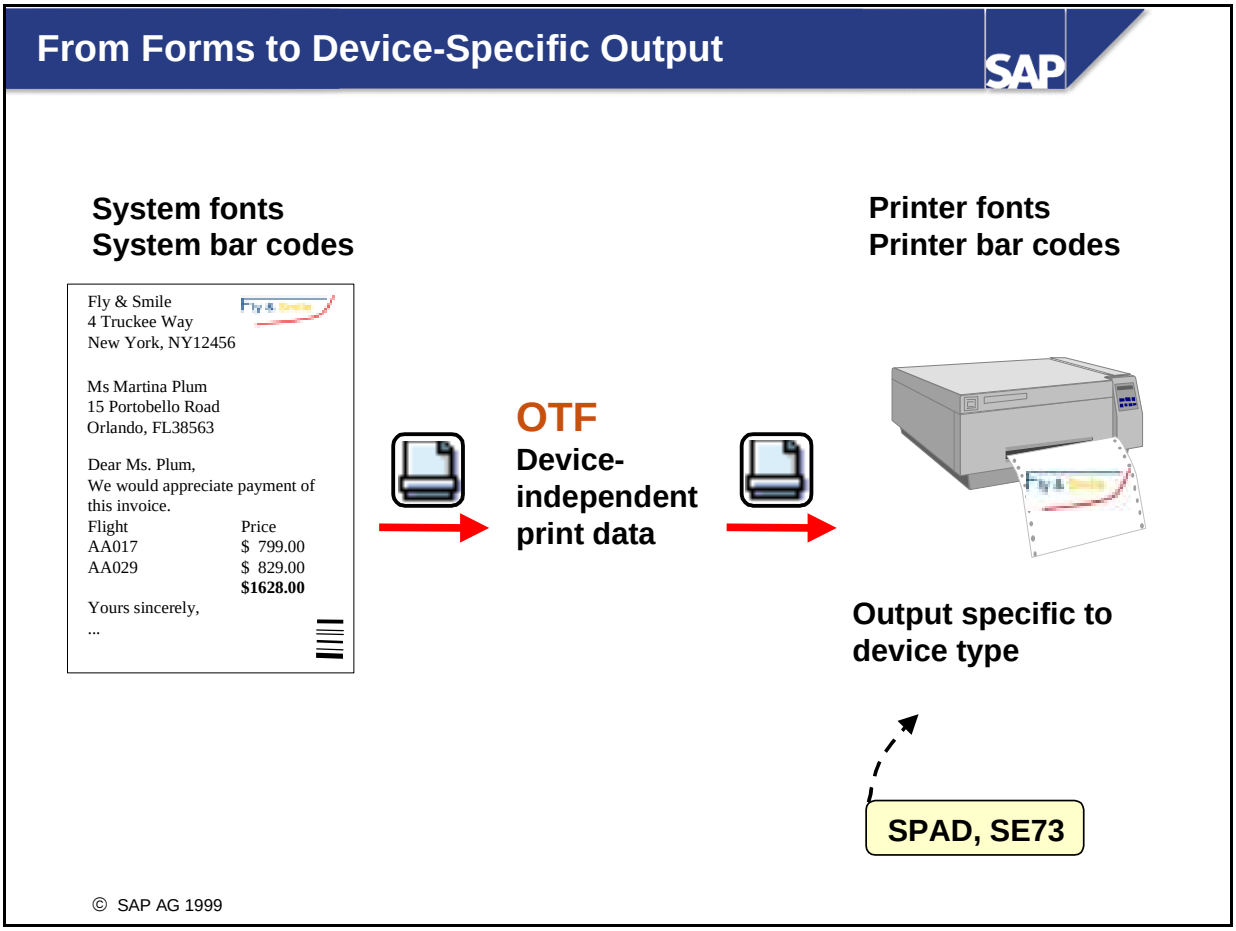
- **Adjusting Device Types**
- **Adding Print Controls to a Device Definition**
- **Creating New Font Families**
- **Defining New Printer Font Sizes**
- **Maintaining Bar Codes**

Font and Bar Code Maintenance: Unit Objectives

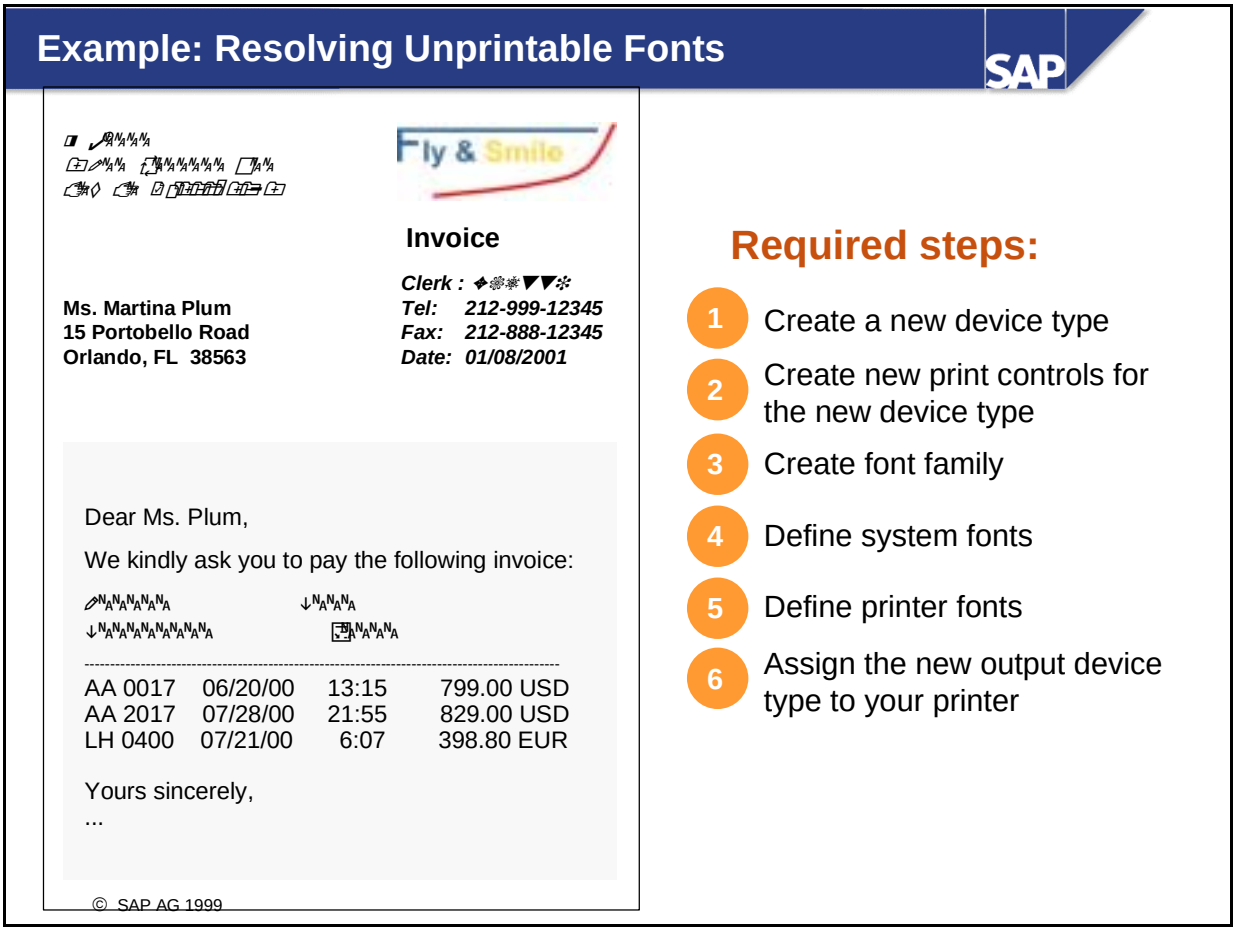
At the conclusion of this unit you will be able to:

- **Set up support for additional font sizes**
- **Maintain printer bar codes**

- **Problem**: The standard SAP device type for your printer is not set up for the requirements of a document and must be adjusted.
- **Solution:** Set up a device type with the additional fonts required.
- **Additional problem:** Need to create printer bar codes.
- Note: Font and bar code maintenance is identical in SAP Smart Forms and SAPscript.



- In the R/3 spool system, an output request is needed to convert device-independent print data into the appropriate printer language understood by the output device.
- A device type specifies which SAP Smart Forms printer driver the system should use for output formatting for devices of this type, as well as which printer character sets are required.
- The device type information is used to convert a document from the internal output format (OTF or Output Text Format) into a device-specific, print-ready data stream. Because the device type specifies attributes that apply to all devices of a particular model, it can be used by multiple device definitions. For example, all Hewlett-Packard LaserJet 4-compatible devices use the device type HPLJ4 in the R/3 spool system.
- You can modify any part of a device type as required. For example, you may want to add support for a printer font that is not part of the R/3 standard font set. This change would require adding new print controls to the device type to enable it to switch to the font.
- Never change an R/3 standard device type. Instead make your own copy of the device type. Otherwise your changes may be overwritten when you upgrade your R/3 System.

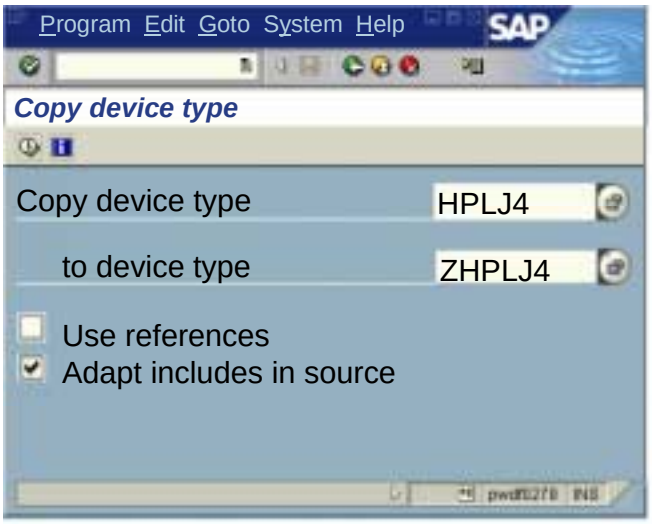


- Problem 1: Printout of invoice has errors. Some fonts are not properly printed.
- Cause 1: You have bought a new printer that is not yet supported by SAP.
- Problem 2: You cannot select all the fonts that your printer offers.
- Cause 2: You have installed a new font cartridge into your printer which the standard SAP device type for this printer (e.g. HPLJ4) is not set up for, or you have bought a new printer which is not yet supported by SAP.
- Solution: Set up a device type with the additional fonts/font sizes required.
- To set up the additional required fonts/font sizes the following steps are required:

  1) Create a new device type in your namespace by copying the existing device type, using the Spool Management (transaction SPAD).
  2) Create new print controls for the new device type, using transaction SPAD or SE73.
  3) Create a new font family if necessary.
  4) Define system fonts for the particular family in the desired sizes and bold/italics settings.
  5) Define printer fonts for newly created system fonts for the new device type and assign the corresponding print controls.
  6) Assign the new output device type to your printer (i. e. your output device). For example, change the entry for your HP printer: replace the old device type (HPLJ4) with the new one you created (ZHPLJ4).

Copying a Device Type

1 Transaction SPAD:



- To create a new device type you should copy an existing device type to the customer namespace, for example HPLJ4 is copied to the target device type ZHPLJ4.
- Use transaction SPAD (menu path: *Tools→ CCMS → Spool → Spool administration) → Utilities → For device type → Copy device type*).
- *Use references*: Select this option to have device format actions and font metrics not copied to the new device type but included by reference from the original device type. Advantage: the device format actions remain up-to-date with changes made in the R/3 standard device type. You should only set this parameter if you are sure that the source device type exists in all systems in which the target device type is used. If your source device type is a customer-defined copy (that is, YXXX or ZXXX), you should leave this flag disabled.
- The R/3 System copies all of the following when you copy a device type:
  - The device type definition
  - The print controls. (You can add to the list of standard print controls without restriction.)
  - Format types
  - Font metrics
  - SAP Smart Forms and SAPscript printer fonts and printer bar codes

Adding Print Controls to a Device Type

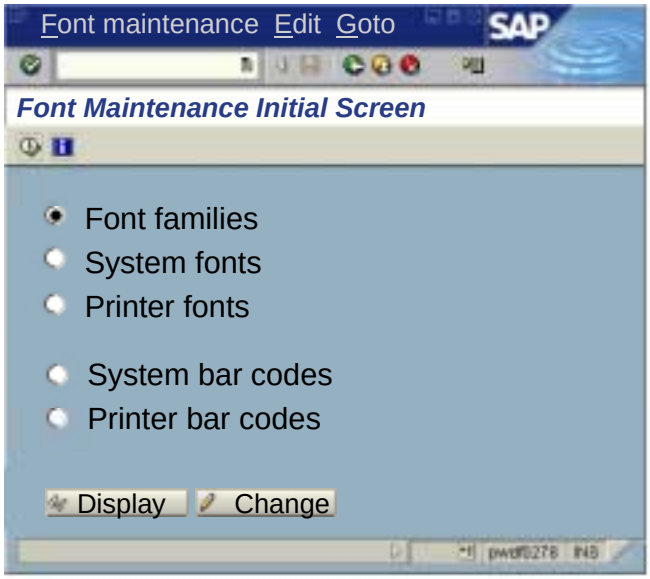
2

- **Edit the print controls in the new device type**
  Insert the entries for the desired font sizes for your font

PrintCtrl (Example)	FontSize ($^{1}/_{10}$ points)	Control character sequence
SF910	180	1B28304E1B28733170313876307333623431303154
SF915	240	1B28304E1B28733170323476307333623431303154
SF920	360	1B28304E1B28733170333676307333623431303154

- A new print control SFXXX must be maintained for device type ZHPLJ4. It will contain the printer control commands for setting the desired font corresponding to the SAP Smart Forms font. To find out what this printer command looks like it is necessary to refer to the printer manual and to the print controls already contained in device type ZHPLJ4. A certain amount of knowledge of the printer language is a prerequisite. The XXX numbering of the SFXXX print controls is arbitrary.
- To add new print controls to a copy of a device type, proceed as follows: From the SAP menu, choose *Tools → CCMS → Spool → Spool administration,* and click on *Full administration.* Choose the tab *Device Types.* Enter the name of the device type in the *Device types* field and confirm your input. Choose *Print controls* and enter the change mode. Choose *Edit → Insert row* and enter a new SFXXX print control you want to add to the device type definition.
- Enter the attributes and control character sequence (you found in the printer manual) of the new print control. You can enter the command either as a hexadecimal string or plain text.

Accessing Font Maintenance



- All fonts and bar codes used in SAP Smart Forms are administered in the font maintenance transaction (SE73). To access this transaction, choose *Tools→ Form Printout →Administration →Font* from the SAP menu.
- The set of **font families** is maintained in font maintenance where you enter the names of the fonts that can be used. Each font family is also assigned the attribute *proportional* or *non-proportional*.
- A **system font** (or SAP font) is a combination of font family, font size, and the attribute for bold and italic. The font selections used in text layout with SAP Smart Forms are always system fonts.
- A **printer font** is the combination of printer type, font family, font size, and the attribute for bold and italic. Printer fonts are the character fonts of the output devices that are available for SAP Smart Forms. Printer fonts require the maintenance of certain metrics data (information about the width of the characters) and control data to set the fonts on the output devices
- **System bar codes** maintain the various bar code types independently of any device. The character string attribute *Barcode* allows you to output bar codes using SAP Smart Forms.
- The **printer bar code** establishes a connection with the device-specific control sequence known as a print control for printer types that support bar code printing.

Creating Font Families, System, and Printer Fonts

3 Font Family:

Family	Description	P	Rpl.1	Rpl.2	Rpl.3	Character set
COURIER	Courier	✓	LETGOTH			0000

4 System Font:

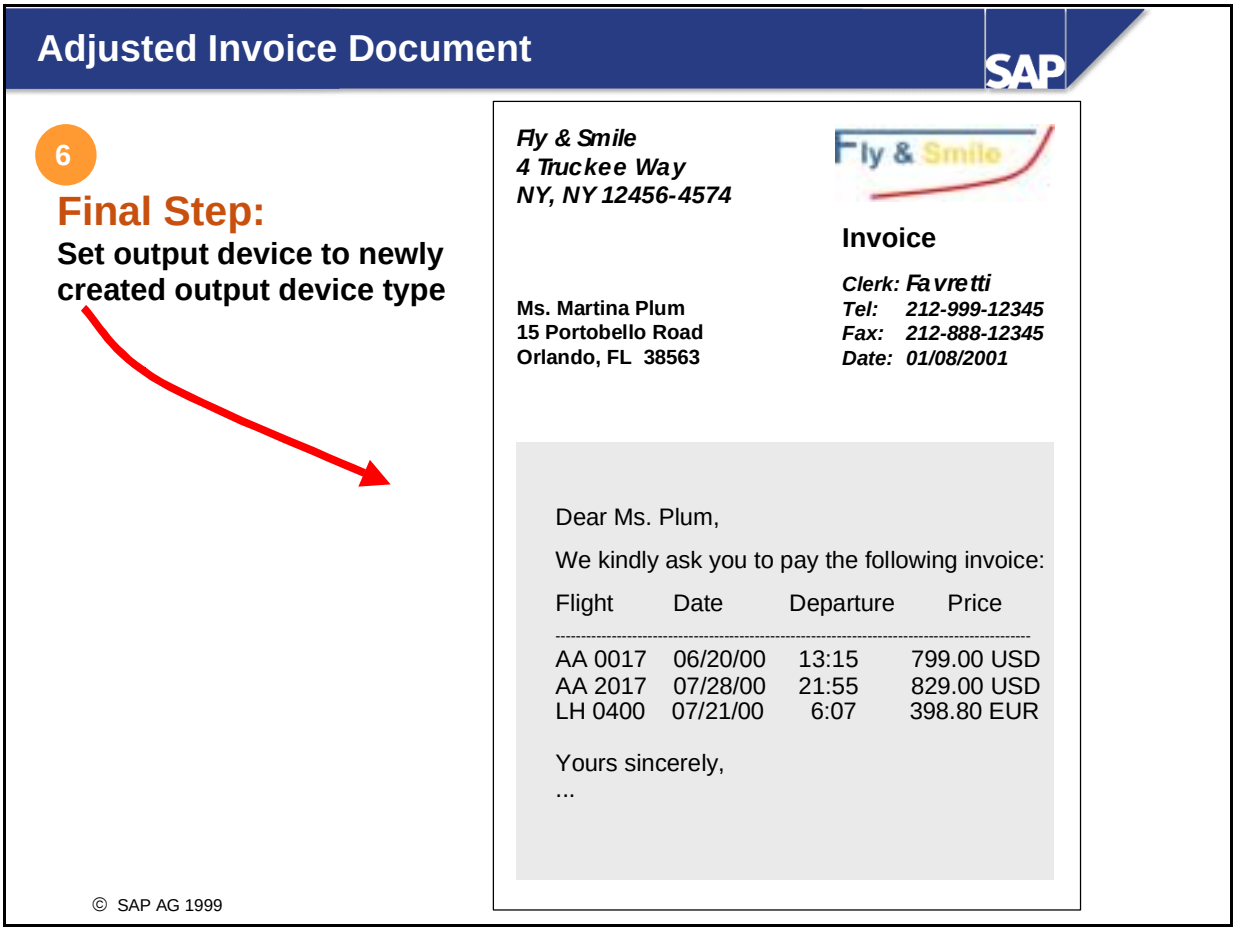
Family	Font size	Bold	Italic
COURIER	060		
COURIER	060	✓	
COURIER	060		✓
COURIER	060	✓	✓

5 Printer Font:

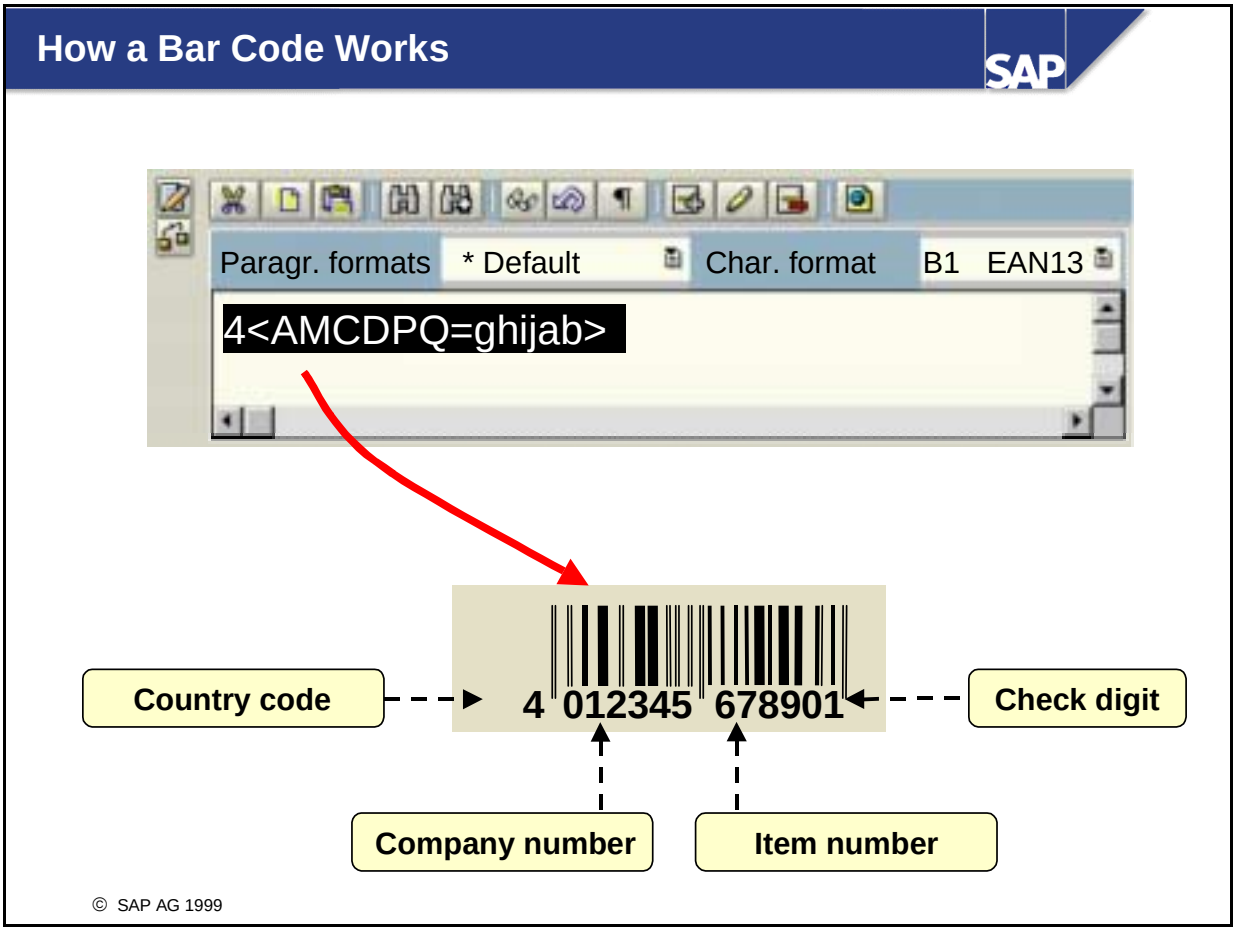
Device type	Family	Font size	Bold	Ital.	CPI	PrtCtl.1 ...
ZHPLJ4	COURIER	060			17,00	SF025
ZHPLJ4	COURIER	060	✓		17,00	SF026
ZHPLJ4	COURIER	060		✓	17,00	SF027
ZHPLJ4	COURIER	060	✓	✓	17,00	SF028

- To create a new font family, choose *Font families* and *Change* from the initial screen of SE73. Choose *Create* to add a new font family.
- Specify a *Substitute family* if this family exists in the system. The substitute family is used in the event that this font is not defined in the target device type of an output request.
- For *Character set*, enter the number of the R/3 character set (codepage) to use for printing this font.
- To create a new printer font, the following information must be given:
  - Device type (printer type to which the font belongs)
  - Font family (font name used in R/3, for example Courier or Times)
  - Size (font size in 1/10 points, for example 240. For printer drivers that support scalable fonts, 000 is entered here.)
  - The font attributes bold and/or italic.
  - CPI (number of characters per inch)
  - Print controls
- If the font is a proportional font (like Times), an AFM file that contains the width values for the individual characters in the font must be entered. Direct maintenance of the AFM data is done from the list of printer fonts by choosing the pushbutton *Edit metrics* while the cursor is on the font. You could

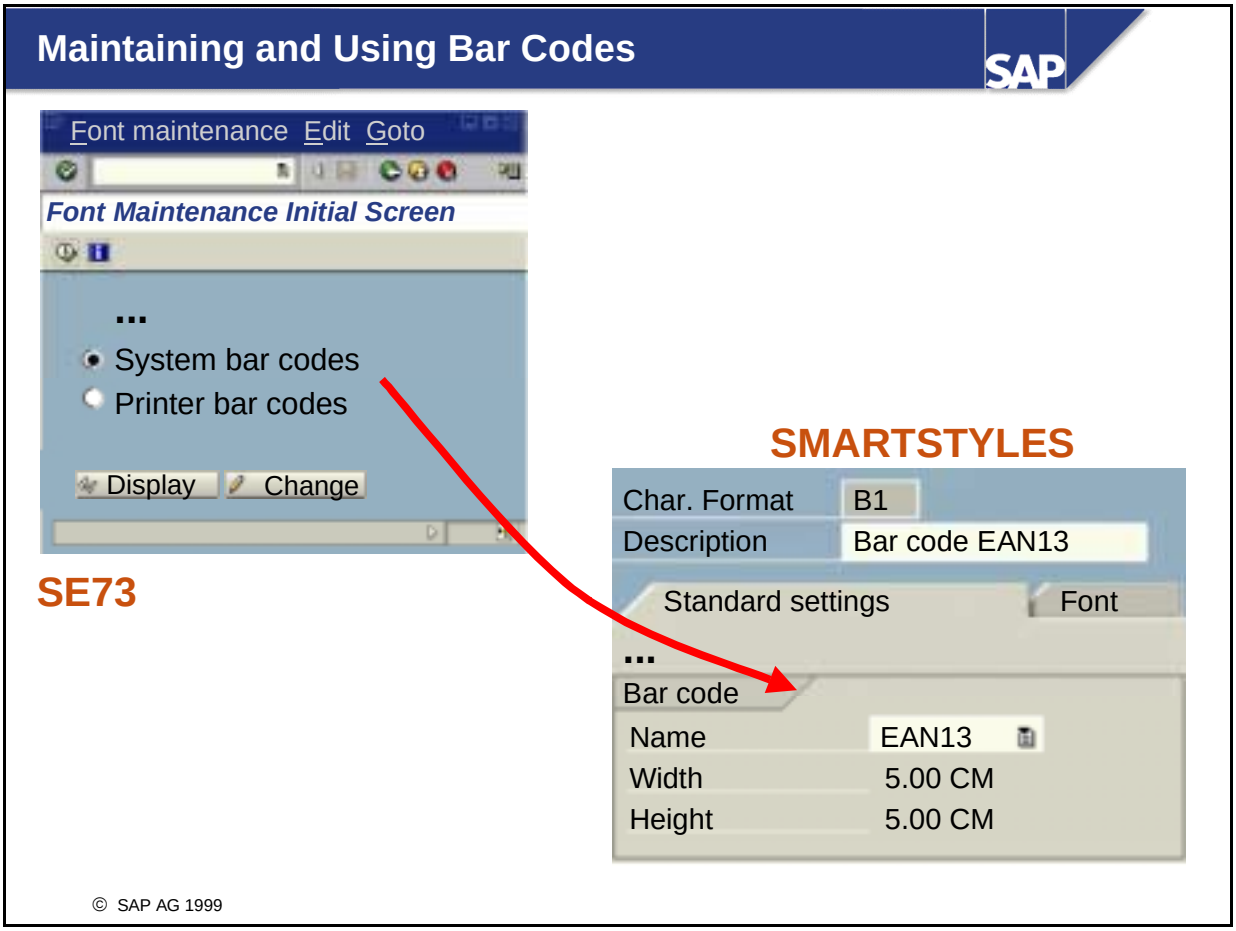
also copy the font metrics of an existing system font by choosing *Edit → Copy font metrics* from the menu.



- As a final step, the device type of the output device (printer) used must be changed from the SAP original to the new printer definition in Spool Administration (transaction SPAD). Example: printer P280 previously had device type HPLJ4 and is now being given device type ZHPLJ4.
- The transaction SPAD is also accessible from Font Maintenance: *Environment → Administration → Spool Administration: Output Devices* pushbutton.



- A bar code is an automatic identification technology. It allows data to be collected accurately and rapidly.
- The bar code symbol consists of a series of parallel, adjacent bars and spaces. Predetermined width patterns are used to code actual data into the symbol. To read information contained in a bar code symbol, a scanning device, such as a light pen (or wand), is used. As a scanning device is moved across the symbol, the bar code width pattern of bars and spaces is analyzed by the bar code decoder, and the original data is recovered.
- In SAP Smart Forms, bar codes are treated as text data (for example, the character string 4<AMCDPQ=ghijab>). This text data must be formatted with a character format that uses a system bar code. For example, in a Smart Style the character format B1 might be defined as a bar code format using bar code EAN13.



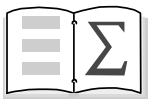
- When a bar code is printed, a print control called the bar code prefix is first sent to the printer. This is followed by the bar code data (for example, an 8-digit number). Finally, another print control, called the bar code suffix, is sent. The naming convention is: SBPXX for the prefix and SBSXX for the suffix.
- Device-specific control sequences known as print controls are maintained with the printer bar codes.
- For maintaining bar code print controls, system bar codes, and printer bar codes, follow the instructions given for maintaining system fonts and printer fonts.

OSS Notes for Fonts and Bar Codes

- **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**

- 0085469 SAPLPD parameter in WIN.INI & SAPLPD.INI
- 0025344 Interface SAPLPD barcode DLL (details)
- 0119604 Bar codes Code 128, EAN-128, UCC-128
- 0045643 Bar code control sequences for JetCAPS BarSIMM
- 0133660 OCR and MICR SIMMs for IBM network printer
- 0197177 Printing 2-D bar codes with SAPscript

Font and Bar Code Maintenance: Summary



You can now:

- **Set up support for additional font sizes**
- **Maintain bar codes**

Appendix

BC470 Form Printing with SAP Smart Forms

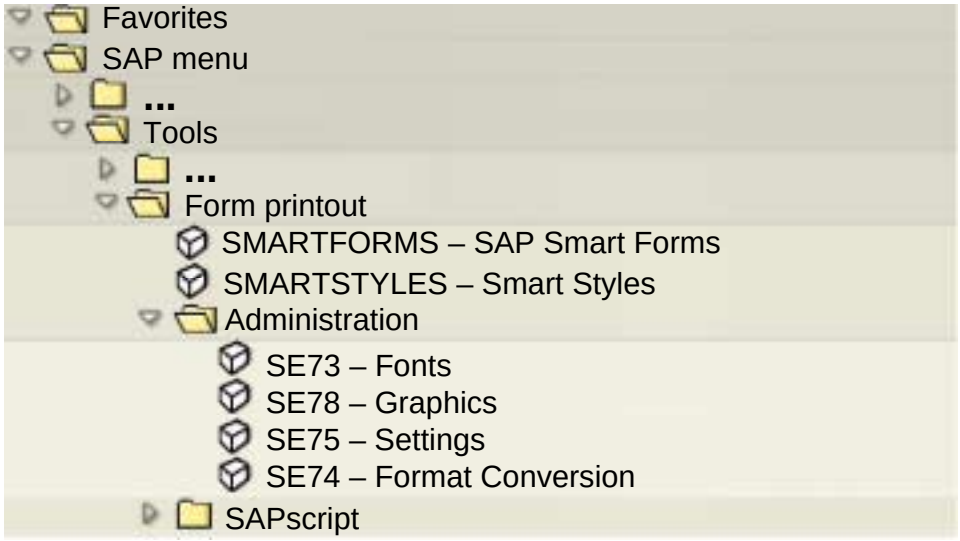
1 Course Overview
2 SAP Smart Forms: Overview
3 First Steps with the SAP Form Builder
4 Texts, Addresses, and Graphics
5 Data in Forms
6 Tables and Templates
7 Flow Control
8 Integration into Application Programs
9 Smart Styles
10 Fonts and Bar Codes
... Appendix

Appendix: Contents

Contents:

- Menu Paths and Transaction Codes
- Legend of Icons in the Navigation Tree
- Change Procedure: Dunning, Delivery Note, Invoice
- Graphics Administration
- Transport
- Using SAPscript Objects
- Forms in Multiple Languages

Menu Paths and Transaction Codes



Legend of Icons in the Navigation Tree


Main window


Secondary window


Text


Address


Graphic


Table


Control level


Template


Loop


Folder


Alternative


TRUE


FALSE


Command


Program lines

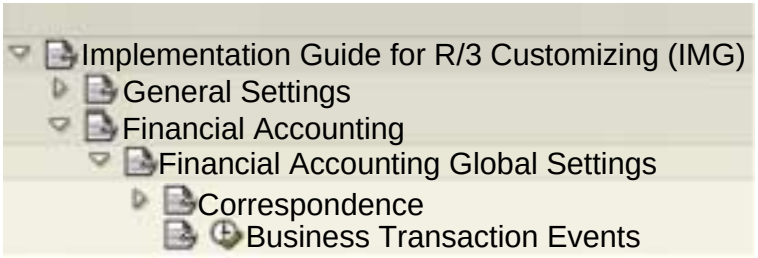

Condition


Smart Style element


Outline paragraph

Change Procedure: Dunning Example I

1 SAPscript or SAP Smart Forms?



2 Form F150_DUNN_SF
Text modules
Graphic

} **Copy and adjust**

4 Truckee Way · New York, NY 12456-4574
PO Box 16 05 29 · New York, NY 34573-1345
Telephone (212) 99-0 · Telefax (212) 99 12 77
Email: info@fly-and-smile.com
Office hours: 09:00 - 18:00 Eastern time

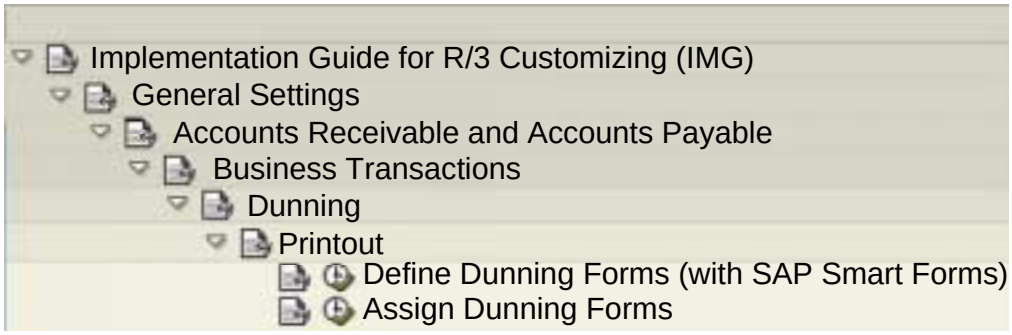
- We will use the dunning procedure as an example to illustrate all steps that are required to ensure that a transaction executes your coding and uses your SAP Smart Forms.

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 then adjust and activate it.
   - Adjust the inserted texts (such as the address) and the company logo.

Change Procedure: Dunning Example II

3 Which form?



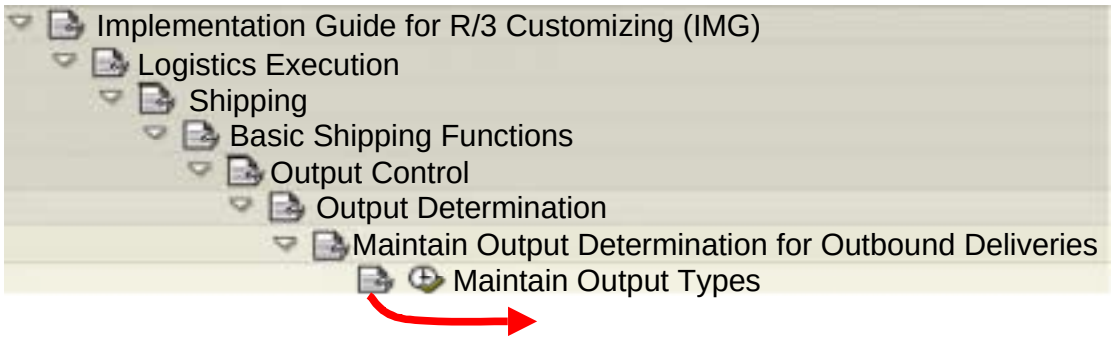
- 3. Enter the form in Customizing:
  - Implementation Guide: *Financial Accounting* → *Accounts Receivable and Accounts Payable* → *Business Transactions* → *Dunning* → *Printout* → *Assign Dunning Forms*
  - Select the desired procedure (for example, *Four-level dunning, every two weeks*), then choose *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).

Change Procedure: Delivery Note Example I

1 **Form LE_SHP_DELNOTE**
Text modules
Graphic
Program RLE_DELNOTE

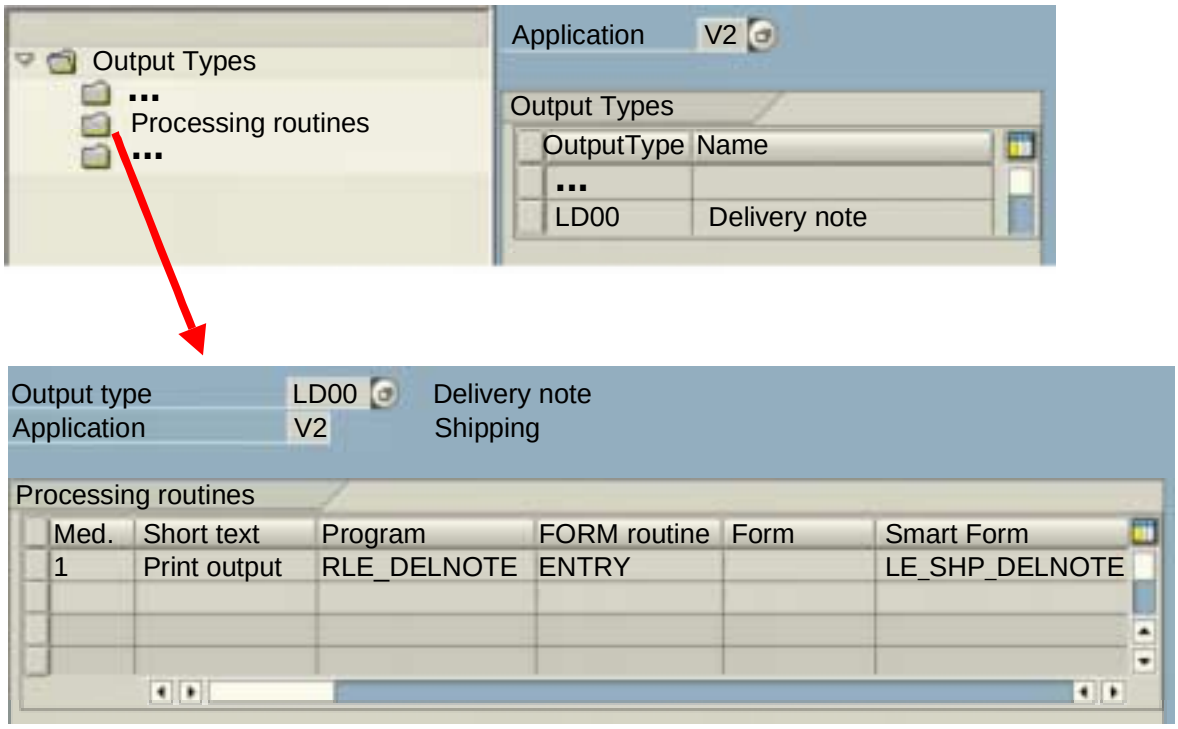
} **Copy and adjust**

2 **Entries in Customizing**



- 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.
- You make a setting in Customizing for the delivery note that specifies that 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.*

Change Procedure: Delivery Note Example II

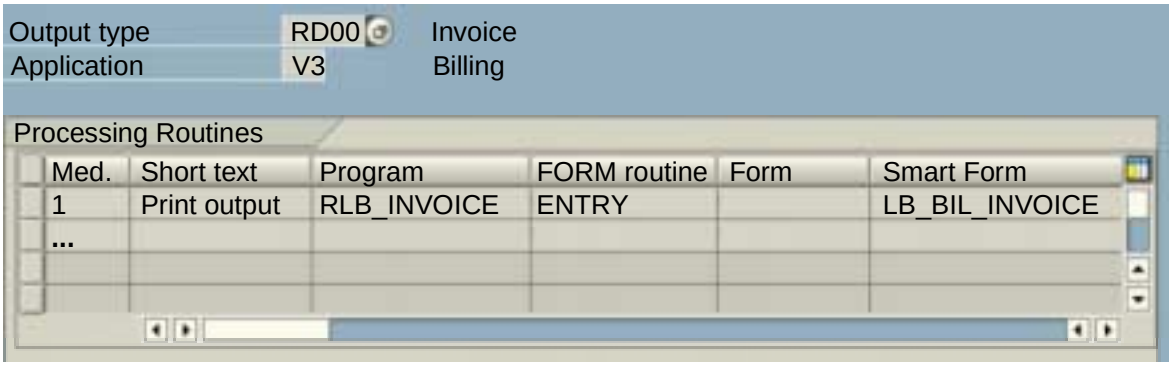


- Select the output type LD00 (delivery note) and double-click *Processing routines*. On the next screen, you have to make the following entries:
  - Medium (fax or printer)
  - Program: RLE_DELNOTE or the copy you made.
  - FORM routine of the program which contains the call of the SAP Smart Form
  - Form: Do not enter anything here. (This field is for SAPscript forms.)
  - Smart Form: LE_SHP_DELNOTE or the copy you made.

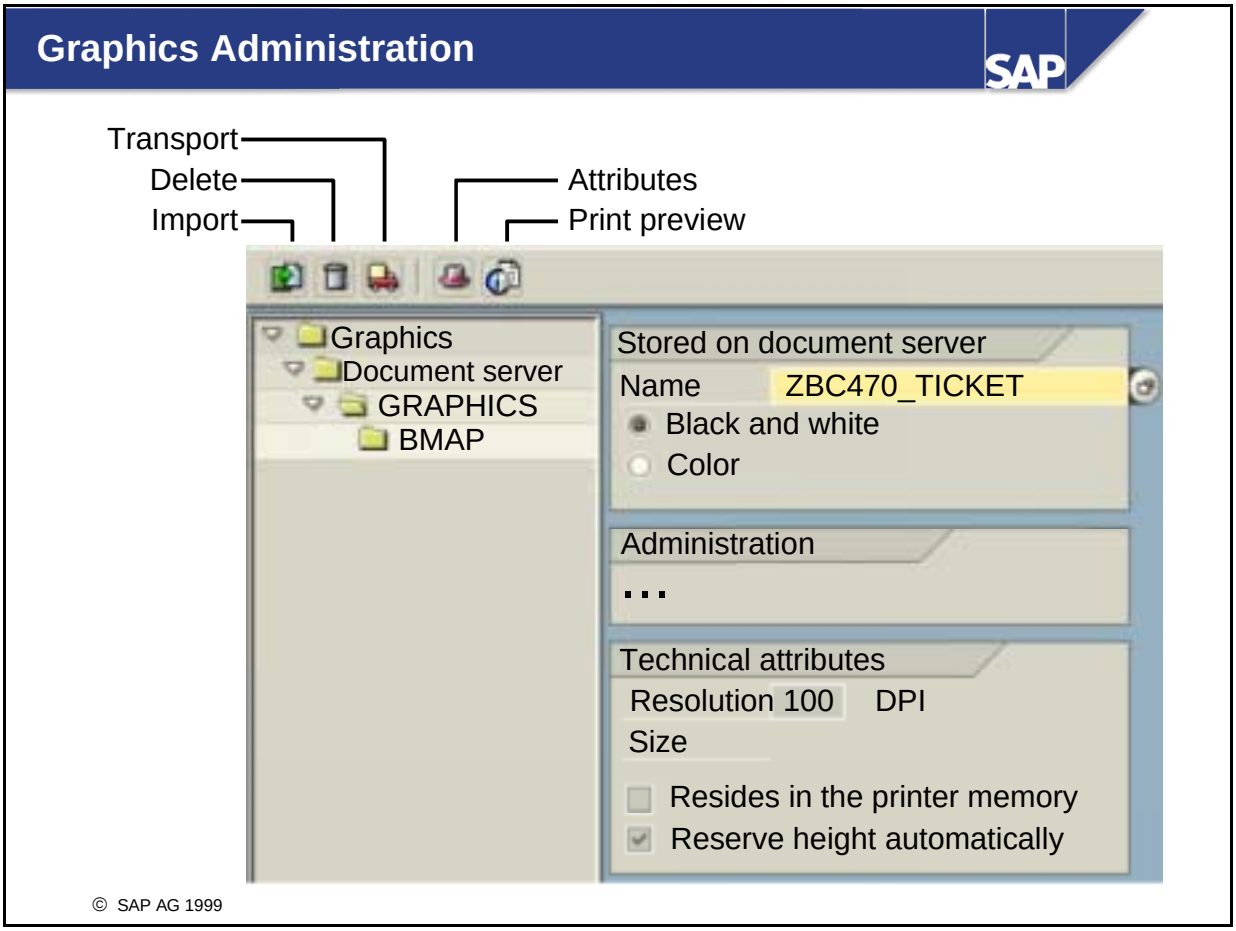
Change Procedure: Invoice Example

1. **Form LB_BIL_INVOICE**
   Text modules
   Graphic
   Program RLB_INVOICE
   } **Copy and adjust**

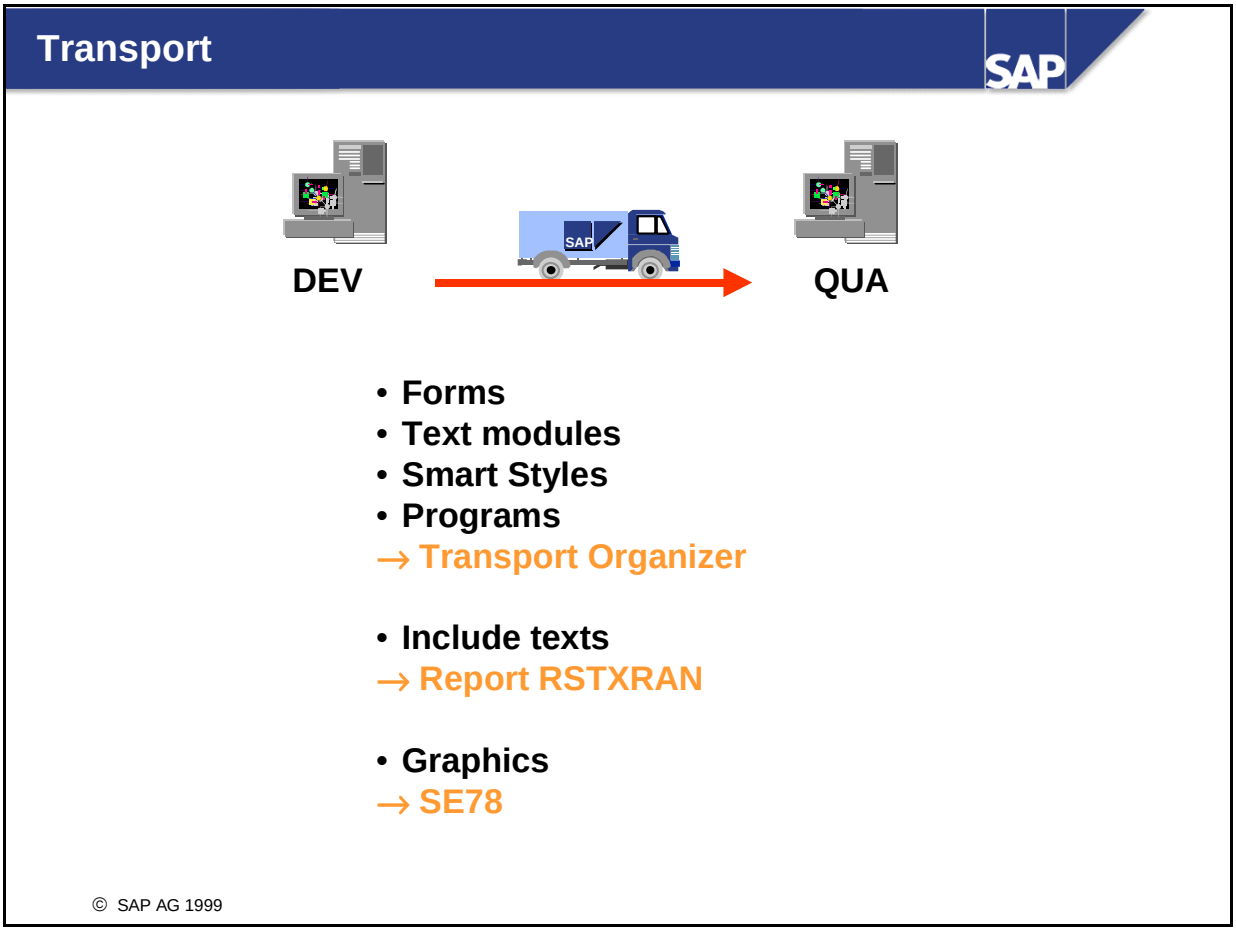
2. **Settings in Customizing:**
   VOK2 → *Output* → *Processing programs* → *Billing document*
   Output type: RD00; Application: V3



- 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 then modify the copy as required.
- You make a setting in Customizing for the invoice form that 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 which 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.



- Inserting a graphic, either as the background picture of a page or as a graphic node, requires that it has been imported from your front-end computer into the system using graphics administration. You call graphics administration by choosing *Tools → Form printout → Administration → Graphic* from the SAP menu. The corresponding transaction code is SE78.
- On the left side of the screen you see the storage paths in the system. For SAP Smart Forms you can only use the *Stored on document server* option. You can set the subfolders displayed in transaction SE75 (menu path: *Tools → Form printout → Administration → Settings*). To manage graphics, select a subfolder.
- On the right side of the screen you see information on the graphic and possibly a preview.
- To transfer a graphic to the document server, enter the name in the field on the right and determine whether you want to have a black and white graphic or a color graphic. Then click the leftmost pushbutton in the toolbar. On 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 if it is needed again in the same print request. This may increase performance considerably.
- You start the import by clicking the *Enter* button.
- You can import graphics in TIFF or Bitmap format.
- Once you have imported a graphic, you can delete it using the corresponding pushbutton or you can add it to a transport request. Besides, you can display the graphic attributes or the graphic itself in the print preview on the right side of the screen.



- 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 associated 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 (SAPscript texts) are not automatically linked to the Workbench Organizer. You must add them to a development request either manually (with transaction SE09) or automatically using report RSTXTRAN. For more information refer to the SAPscript documentation.
  - 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*.

Using SAPscript Objects I

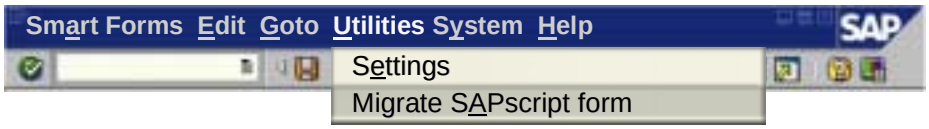
SAPscript

Forms

Problem: Different concepts

- **Interface between print program and form**
- **Formats**
- **SAPscript commands**
- **Loop logic**

→ Migrate with tool and adjust



- 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 then choose *Utilities → Migrate SAPscript form.* 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.
- You can also migrate several active forms simultaneously. To do this, run the report SF_MIGRATE. The forms are automatically saved as local objects.
- When you migrate a form, the general form attributes as well as the layout information including the pages and 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 but centrally in SAP Smart Forms styles). 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.

Using SAPscript Objects II

SAPscript

Texts

Use as include texts

Caution:

- **Commands**
- **Styles**
- **Fields**
- **Client-specific**

Print programs

Adjust manually

Styles

Migrate with tool

- 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 texts in SAP Smart Forms 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 by the generated function module with the interface. The processing of the text elements with the function module `WRITE_LINE`, including the setting/deleting of headings, must be completely transferred 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.*

Forms in Multiple Languages

Text elements:

The language in which you want the form to be printed should be passed on to the application program. Use the field `langu` in the parameter `control_parameters` of the generated function module. Use the fields `replangu1`, `replangu2` and `replangu3` to specify up to three replacement languages.

If no translation of the form or certain text elements exists in any of the specified language or if you have not specified any language at all, the text elements will be printed in the logon language. In case the form does not exist in this language either, the original language of the form will be taken.

Text modules:

The output of text modules is always in the language of the form (Release 4.6C).

Include texts:

You can specify the language for include texts in the respective text node of the form. If you omit it, the logon language or the original language will be taken.

Addresses:

In order to achieve a country specific formatting, use address nodes (when working with the Central Address Management) or call the function module `ADDRESS_INTO_PRINTFORM` in a program lines node.

Decimal notation and date format:

You determine the decimal notation and the date format by using the ABAP command `set country <language>` in the application program. Without this command, the settings are taken from the user's master record as valid at logon.

Maintaining a form:

Maintaining a form is possible only in the original language. Either you log on to the system in this language, or you will be asked whether you want to set the original language of the form to a different one before you can maintain the form.

Creating a translation:

Use transaction SE63. You can translate only into those languages that you have specified in the form attributes.

Copying/transporting forms:

All languages are copied/transported automatically with the original language.