SAP Smart Forms
(BC-SRV-SCR)

HELP.BCSRVSCRSF

Release 4.6C

Copyright

Icons

Icon	Meaning
	Caution
	Example
	Note
	Recommendation
	Syntax
	Tip

Contents

SAP Smart Forms (BC-SRV-SCR)

SAP Smart Forms (BC-SRV-SCR)

Purpose

You use SAP Smart Forms to create and maintain forms for mass printing in SAP Systems. As output medium SAP Smart Forms support a printer, a fax, e-mail, or the Internet (by using the generated XML output).

In addition to the tool, SAP delivers a selection of forms for central business processes. This includes forms in Customer Relationship Management (CRM) as well as in the applications SD, FI, and HR of the R/3 Release.

SAP Smart Forms offer the following advantages:

- Creating and maintaining forms requiring half the time
- Adapting forms without any programming knowledge due to entirely graphical user interface
- Web Publishing using the generated XML output

The SAP Smart Forms replace the SAPscript forms. SAPscript forms will also be supported in the future; you can use them without making any changes for years to come. You can use SAPscript texts in the Smart Forms. Migration of SAPscript forms into Smart Forms is supported.

Features

SAP Smart Forms allow you to execute simple modifications to the form and in the form logic by using simple graphical tools; in 90% of all cases, this won't include any programming effort. Thus, a power user without any programming knowledge can configure forms with data from an SAP System for the relevant business processes.

To print a form, you need a program for data retrieval and a Smart Form that contains the entire from logic. As data retrieval and form logic are separated, you must only adapt the Smart Form if changes to the form logic are necessary. The application program passes the data via a function module interface to the Smart Form. When activating the Smart Form, the system automatically generates a function module. At runtime, the system processes this function module.

You design a form using the graphical Form Painter and the graphical Table Painter. The form logic is represented by a hierarchy structure (tree structure) that consists of individual nodes, such as nodes for global settings, nodes for texts, nodes for output tables, or nodes for graphics. To make changes, use Drag&Drop, Copy&Paste, and select different attributes. These actions do not include writing of coding lines or using a Script language.

For Web publishing, the system provides a generated XML output of the processed form.

You can insert static and dynamic tables. This includes line feeds in individual table cells, triggering events for table headings and subtotals, and sorting data before output.

You can check individual nodes as well as the entire form and find any existing errors in the tree structure. The data flow analysis checks whether all fields (variables) have a defined value at the moment they are displayed.

SAP Smart Forms allow you to include graphics, which you can display either as part of the form or as background graphics. You use background graphics to copy the layout of an existing

SAP Smart Forms (BC-SRV-SCR)

(scanned) form or to lend forms a company-specific look. During printout, you can suppress the background graphic, if desired.

SAP Smart Forms also support postage optimizing.

General Concepts on Form Printing

Form printing covers creating and maintaining a form for mass printing in SAP Systems. This includes design and layout of the form as well as form logic.

The layout determines the page structure, that is the number of differently structured pages and the positions of the output areas on these pages. Within the output areas, you use tables, paragraphs, paragraph formats and character formats to structure and format texts and data.

The form logic controls the dynamic formatting of the form. It allows you to display variable fields, to display texts only if a certain condition is true (for first dunning, use this text, for second dunning another), or to repeatedly process invoice items within a table.

Forms can be:

- Order confirmations
- Invoices
- Account statements
- Checks
- Salary statements
- Delivery notes
- Customs forms
- Industry-specific forms, such as quality forms in automobile manufacturing

The concepts on form printing presented below are independent of any tool. As of mySAP.com release 4.6C use the SAP Smart Forms [Extern] in SAP Systems.

This documentation does not cover document output and control on printers. Description of the process ends with transferring a processed file to the output management, which in the R/3 System is the spool.

Structure of a Form

Definition

A form consists of pages, output areas, addresses, graphics (such as company logo), and data or text contents. Within an output area, you can use static or dynamic tables to display data or texts in lines, columns, or cells. To further structure and format text and data, use paragraphs with the paragraph and character formats.

SAP calls output areas "windows". You can position windows freely on a page (see also Pages of a Form [Seite 10] and Main Windows and Subwindows [Seite 11]).

Example of a Form Structure

We use a simple invoice. For certain customers, we display their flight bookings with prices in a table in a window. The invoice also contains other windows for the company logo, the sender address, the customer address, company-specific data (clerk, customer number, reference, date, and so on), bank data, and pagination.

The first page displays the customer letter, followed by a table containing the flight bookings of that customer. The output length of the table depends on the number of booking items (dynamic table). The table header contains the column headings, the table footer contains the total. If the space on the first page is not sufficient to take all items, the table continues on the next page, thereby repeating the column headings. In addition, the first page contains:

- the company logo inserted as graphic in bitmap format.
- the customer address. This address appears only on the first page (for the window in the envelope) and is preceded by the sender address in small font.
- a window containing company-related data in several fonts and font sizes (invoice, clerk, phone/fax, reference, customer number, date).
- the footer containing the company and bank data.

The second page contains the current page number and the total number of pages in the document.

Pages of a Form

Definition

On the pages of a form you determine the page layout and the sequence of pages with different structures.

The page layout includes the page format (for example, DIN A4, Letter, DIN A5 landscape) and the position of the windows on a page.

Use

The individual pages of a form differ in their structure. The first page of an invoice, for example, contains the address, which you do not want to display on the next page; and you want to display the General Terms of Business on the last page. For each page, you must specify a next page to make the page sequence clear. You can also repeatedly call the same page if you want recursive output (for example, if the list of invoice items is very long and does not fit onto one document page).

The page break either is triggered automatically by the main window (see also Main Window and Subwindows [Seite 11]) or you code it manually into the form logic (see also Specifying the Page Sequence [Seite 83]).

Many forms contain only two pages of different structure: the first page, which calls

the next page, and the next page, which calls itself. If the text does not entirely fit onto the second page, another document page is automatically displayed.

Main Window and Subwindows

Definition

On a page, there are two different types of output areas for texts and data: the main window and the subwindow.

Use

You can position windows anywhere on a page, even overlapping. You can position the same window (determined by the technical name) on several pages of a form, so that the same contents are displayed on all these pages. You can choose a different size for the window on each page, except for the main window.

Main Window

In a main window you display text and data, which can cover several pages (flow text). As soon as a main window is completely filled with text and data, the system continues displaying the text in the main window of the next page. It automatically triggers the page break.

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

The main window must have the same width on each page, but can differ in height.

A page without main window must not call itself as next page, since this would trigger an endless loop. In such a case, the system automatically terminates after three

pages.

Subwindows

In a subwindow you display text and data in a predetermined output area. There is no flow text display with page break. If you position a subwindow with the same name on several pages, the system displays the contents of this subwindow on each page.

Text and data that do not fit into the subwindow are truncated and not displayed.

Texts and Data in a Form

Definition

You enter text in an editor (see also Positioning Text on a Form [Seite 41]).

Data can be character strings or numbers. The system reads this data from a database and uses fields (parameters) to dynamically display it (see also Using Parameters in a Form [Seite 58]).

Use

To display text and data in table format (for example, lists or invoice items), you use tables or templates.

To format texts and fields (data), you use paragraph and character formats. You assign paragraph formats to entire paragraphs and character formats to individual character strings, for example, to highlight a word.

The paragraph format determines:

- Font family and font size,
- Indents and spacing,
- Text alignment within the paragraph,
- Tabs,
- Outline options, such as numbering,
- Protection of paragraphs against page breaks.

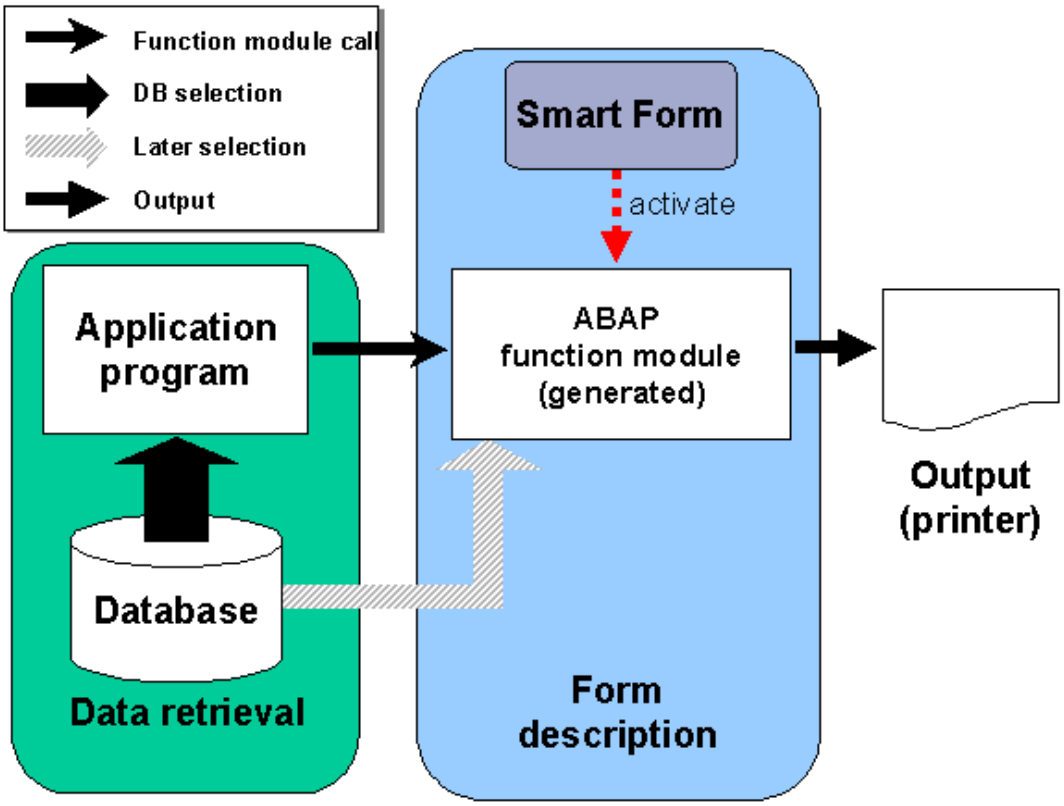
To determine the paragraph and character formats, use the Smart Styles (see also Smart Styles [Seite 87]).

Overview

This section describes the architecture [Seite 14]of SAP Smart Forms and how to create a form [Seite 16]. Essential for creating forms is to understand the form logic [Seite 19].

Architecture

If you create forms for an application, you want to include application data into the form. The architecture of SAP Smart Forms separates application data retrieval from the actual definition of the form:



In a *Smart Form* you describe:

- The *layout* of your form (element positions on a page)
- Individual elements you want to display (text, graphics, addresses, tables, and so on)
- the *form logic* you use, for example, to read application data from internal tables, to introduce conditions, and to control process flows (see also: Form Logic: Introduction [Seite 19])
- a *form interface* to transfer application data to the form definition

When you activate the Smart Form, the system generates a function module that encapsulates all attributes of the Smart Form. As soon as the application program calls the function module, the Smart Form uses the modules interface (which corresponds to the form interface) to transfer any table data previously selected and to print the form according to the form description.

The form description can also contain statements that select further data during form processing. However, you should not use this method of data retrieval. Especially if you print mass forms, performance will deteriorate considerably.

Creating Forms Using SAP Smart Forms

This section gives an overview of how to create a form in which to display application data from database tables.

For more detailed information on this procedure, see Using SAP Smart Forms [Seite 22].

When creating a form, you must:

1. Retrieve the application data [Seite 17].
2. Describe your form [Seite 18].
3. Pass the application data to the form [Seite 59].

Printing the Form

Call a function module generated by Smart Forms to print your form. Smart Forms support the forllowing output options:

- The form is printed on a printer connected to the SAP System. Before printing, use the print preview to check whether the form is correct.
- The function module generates an XSF datastream that transfers, for example, the form description including the retrieved data to programs of third-party companies.

Retrieving Application Data

Prerequisites

To achieve a good performance when printing a form, you must separate data selection from the use of data in the form. Thus you can bundle database operations before you start processing the form.

Before you retrieve data, you should know:

- Which application data you want to appear in the form
- Which database tables you must access to retrieve this data

Process Flow

Write an ABAP program to retrieve data or include a retrieval routine into your application. This code consists of statements that select data from the database according to certain selection criteria. Store the retrieved data in internal tables, structures, or variables and transfer it to the form in one step.

While you initialize the form, you can format the data, which you passed to the form, for output in the form. Formatting data in the form and not in the data retrieval program allows you to use one data retrieval program for several forms with different formatting routines.

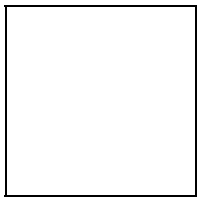
Result

You store the application data in internal tables of the data retrieval program. Since you know now which table types occur, you can define a form interface to transfer this data to the appropriate form.

Describing the Form

You describe a form using a *Smart Form*. To do this, use the Form Builder (see also: Graphical User Interface [Seite 23]):

1. Describe the interface of the form. It results from the application data previously selected.
2. Create one or more pages. On a page, you can position windows, addresses, and graphics. Within a window, you can create additional elements.
3. Create elements (text, graphics, tables, and so on) for each page, using other tools of the Form Builder:
   - Use the Form Painter [Seite 26] to position windows, graphics, and addresses on a page (the other elements are displayed in an assigned window).
   - Use the PC Editor [Seite 28] to write your texts.
   - Use the Table Painter [Seite 30] to format your tables.

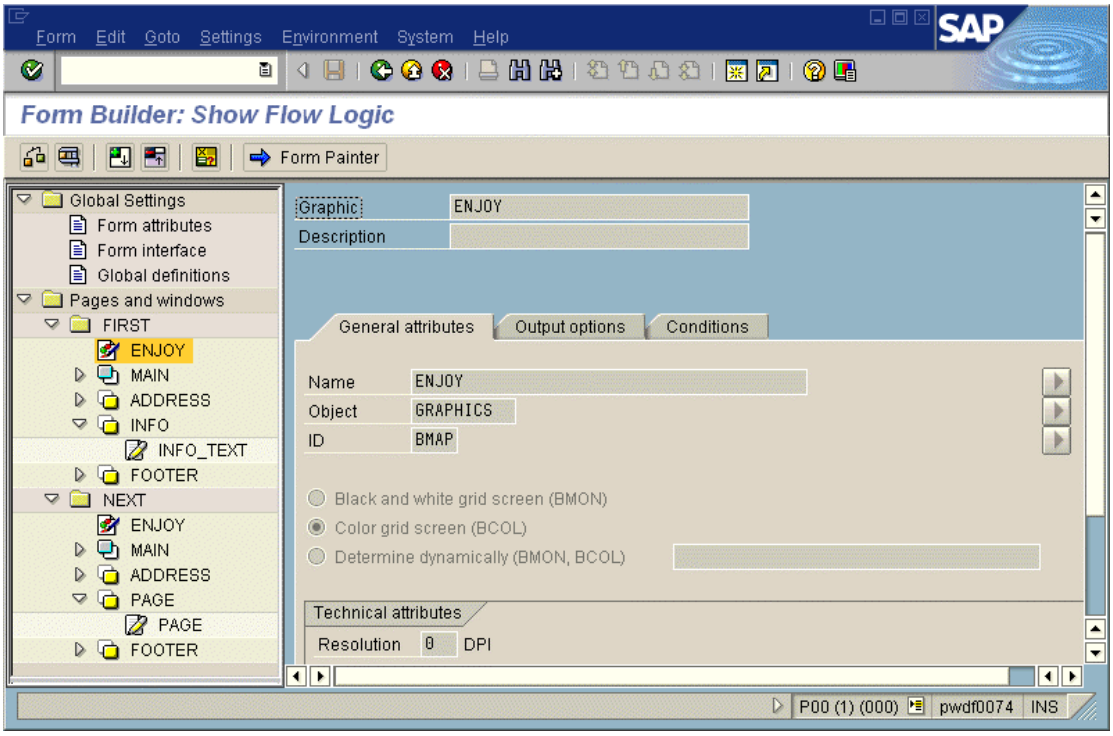
   For an overview of the elements available on page level, see Node Types: Overview [Seite 35].
4. Use the flow control [Seite 79] to control whether and when to print pages and elements.

Result

When you activate your Smart Form, the Form Builder generates a function module that encapsulates the form description.

Form Logic: Introduction

In the Form Builder you describe a Smart Form by a set of nodes. To do this, you build up a tree structure on the left side of the user interface:



This graphic already contains some nodes. The node *Global Settings* as well as its three successors *Form attributes*, *Form interface,* and *Global definitions* always exist for any newly created forms. To describe the *Form logic*, create a hierarchy under the node *Pages and windows*. This hierarchy determines the rules used to process the nodes of the tree. Depending on the node type, this could include:

- printing the node contents (text, addresses, graphics).
- executing the node statements.
- executing the successors of a node according to other rules (for example, in a *loop*).

You use the form logic to control the flow of the form output. The following rules apply throughout:

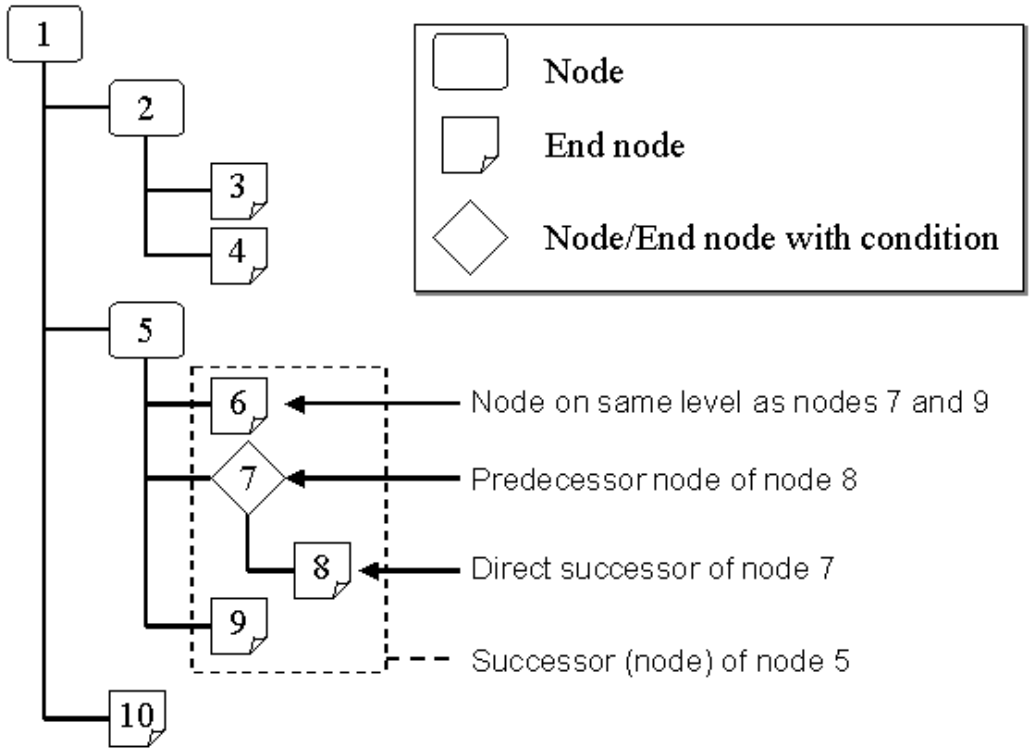
1. The nodes in the tree structure are processed from top to bottom. This is easier to understand, if you imagine all nodes to be expanded.
2. For each node there is a tab, which you can use to link the node to a condition [Seite 80]. If the condition is true, the system processes the node. If not, it skips the node **and all its successors**.
3. You must define a next page for each page. However, you can also go to other pages dynamically (see also: Flow Control [Seite 79]).

In addition, the page break depends on how much space there is left on the current page.

For the complete rules for processing the tree structure, see Processing a Form [Seite 101].

Example for Form Logic

The diagram below shows the tree structure of the root node *Pages and windows*:



The numbering of the nodes also shows the sequence in which they are processed (from top to bottom). The first node is the *root node*. The *successors* of a node are all nodes included directly under this node in the hierarchy. For node 5 these are the nodes 6 to 9 (direct successors are only the nodes of the next hierarchy level, that is 6, 7, and 9).

A node without a successor node is also called *end node*. Elementary nodes can never have successors (see also Node Types: Overview [Seite 35]). If the condition of node 7 is false, neither node 7 nor node 8 are processed. Processing resumes directly with node 9.

In this example, we assume that no pages are called dynamically and that there is enough space on the page for the output.

Using SAP Smart Forms

Graphical User Interface

Use

SAP Smart Forms provide a graphical user interface that helps you create and maintain the layout and the form logic of a form: the SAP Form Builder. You need neither have any programming knowledge nor use a Script language to adapt standard forms. Basic ABAP programming skills are required only in special cases (for example, to call a function module you created or for complex and extensive conditions).

Features

The SAP Form Builder of the SAP Smart Forms consists of:

- Form Painter for creating the layout of a form,
- PC Editor for entering texts and fields into output areas,
- Navigation tree for maintaining the form logic,
- Table Painter for creating templates and tables,
- Form check.

To define text formats, use style maintenance (transaction **SMARTSTYLES**;see also Smart Styles [Seite 87]).

Activities

The SAP Smart Forms initial screen is the starting point for maintaining forms, styles [Seite 87], and text modules [Seite 86]:

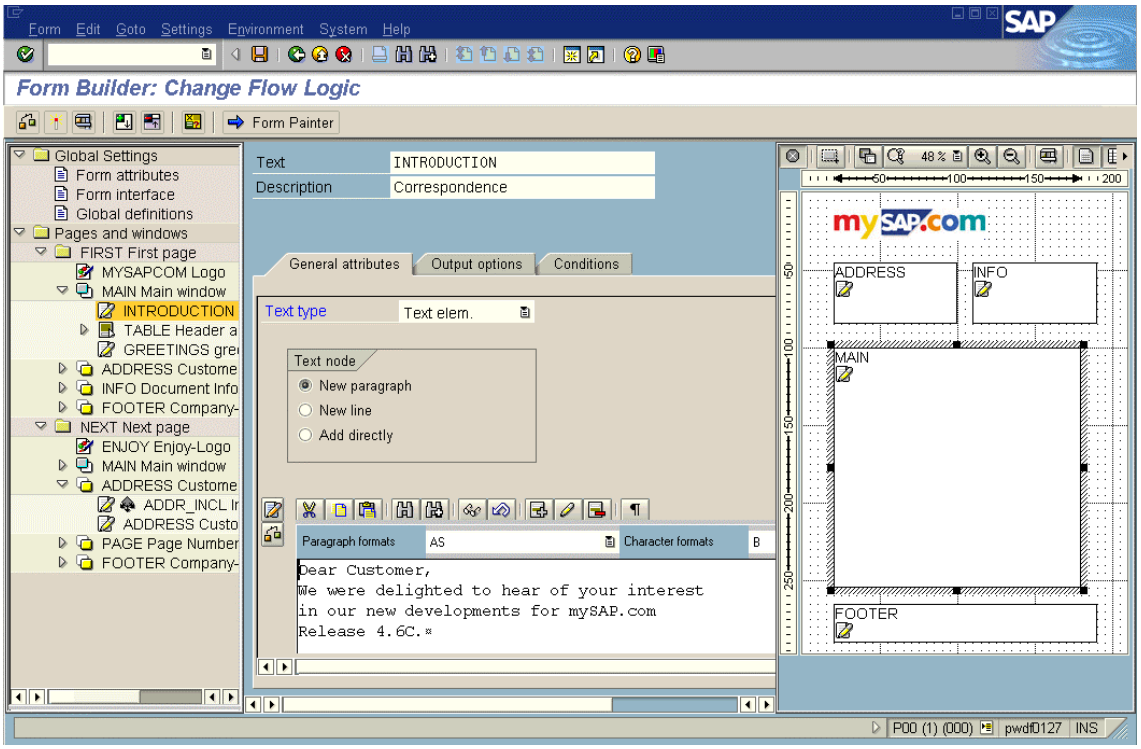
1. Choose transction **SMARTFORMS**.
   The dialog window *SAP Smart Forms: Request* appears.
2. Select *Form*, *Style*, or *Text module*, depending on which object you want to create, display, or change.
3. Enter the name of the object.
4. Choose *Create, Change*, or *Display*.

Depending on the radio button you selected (*Form, Style, Text module*), the system branches to the Form Builder, the style maintenance (transaction **SMARTSTYLES**), or to the text module maintenance.

Navigating in the SAP Form Builder

The screen capture below shows the three different areas of the SAP Form Builder: The navigation tree (hierarchy structure of a Smart Form) on the left, the maintenance screen in the middle, and the Form Painter on the right.

If you select a tree in the node, the system updates the maintenance screen and marks the relevant window in the Form Painter. You can also select a window in the Form Painter; the system them marks the relevant node in the tree.



In the tree you determine the hierarchy of the form logic (see also Form Logic: Introduction [Seite 19]); in the Form Painter you determine the layout. If you, for example, move a window in the tree, this does not effect the layout of the Smart Form.

Tree Navigation

Selecting a Node

To select a node in the tree, double-click it. The system updates the maintenance screen accordingly.

Drag&Drop

Use Drag&Drop to move (left mouse button) or copy (Ctrl + left mouse button) subtrees.

Drag&Drop is a "move" operation, consisting of a "cut" and a "paste" operation with its own clipboard. If the node you are moving can be appended either on the same level as the target node or as a successor of the target node, the context menu offers both possbilities.

Context Menu

Use the right mouse button to display a sensitive context menu in the tree (that is, depending on the current node type). The following functions exist:

- Creating or deleting (only in change mode)
  Depending on the selected node type only valid node types appear.
- Clipboard operations (see below), such as cutting, copying to clipboard, pasting from clipboard (only in change mode)
- Expanding and collapsing a subtree

Clipboard

There are three clipboard operations:

- Copy
  Copies the selected node with all its successor nodes into the clipboard.
- Cut
  Copies the selected node with all its successor nodes into the clipboard and deletes the copied nodes from the tree.
- Paste
  Before pasting, the system checks whether and where you are allowed to append the root node of the clipboard contents.

Maintenance Screen

On the maintenance screen you can select different tab pages depending on the node type. For more information see Node Types: Overview [Seite 35].

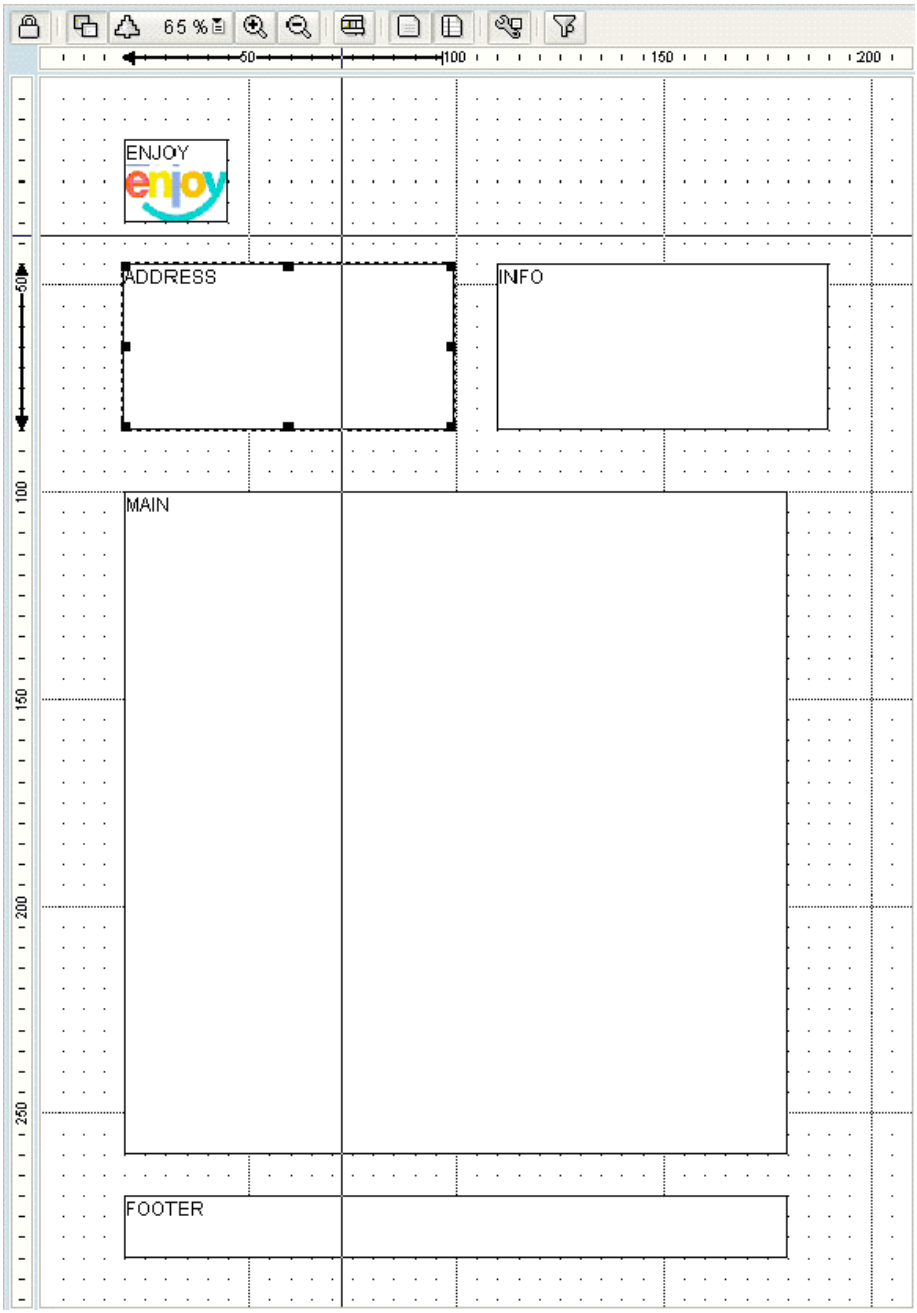
Form Painter

You can suppress or display the Form Painter (choose *Form Painter on/off*). For more information on how to work with the Form Painter see Graphical Form Painter [Seite 26].

Form Painter

Use

You use the Form Painter to design the layout of the pages of a Smart Form. You can include windows and graphics on a page, determine their positions and choose the window sizes.



The Form Painter offers the following functions:

- Design area with ruler, cursor with help lines, and grid and main grid
- Display of the cursor position in the ruler, which you can suppress and set
- Two-level grid with engage function, which you can customize at will
- When creating, changing, and moving a window, the window size appears in the ruler.
- Detachable toolbar containing the most important functions
- Autoscrolling when moving windows; placing windows into the background
- Zoom factors you can set to any value and autozooming of the design area to the window size
- Sensitive context menu
- Placing a scanned graphic into the background of the design area (see also Printing Graphics [Seite 50])

Activities

To display or suppress the Form Painter choose *Form Painter on/off*.

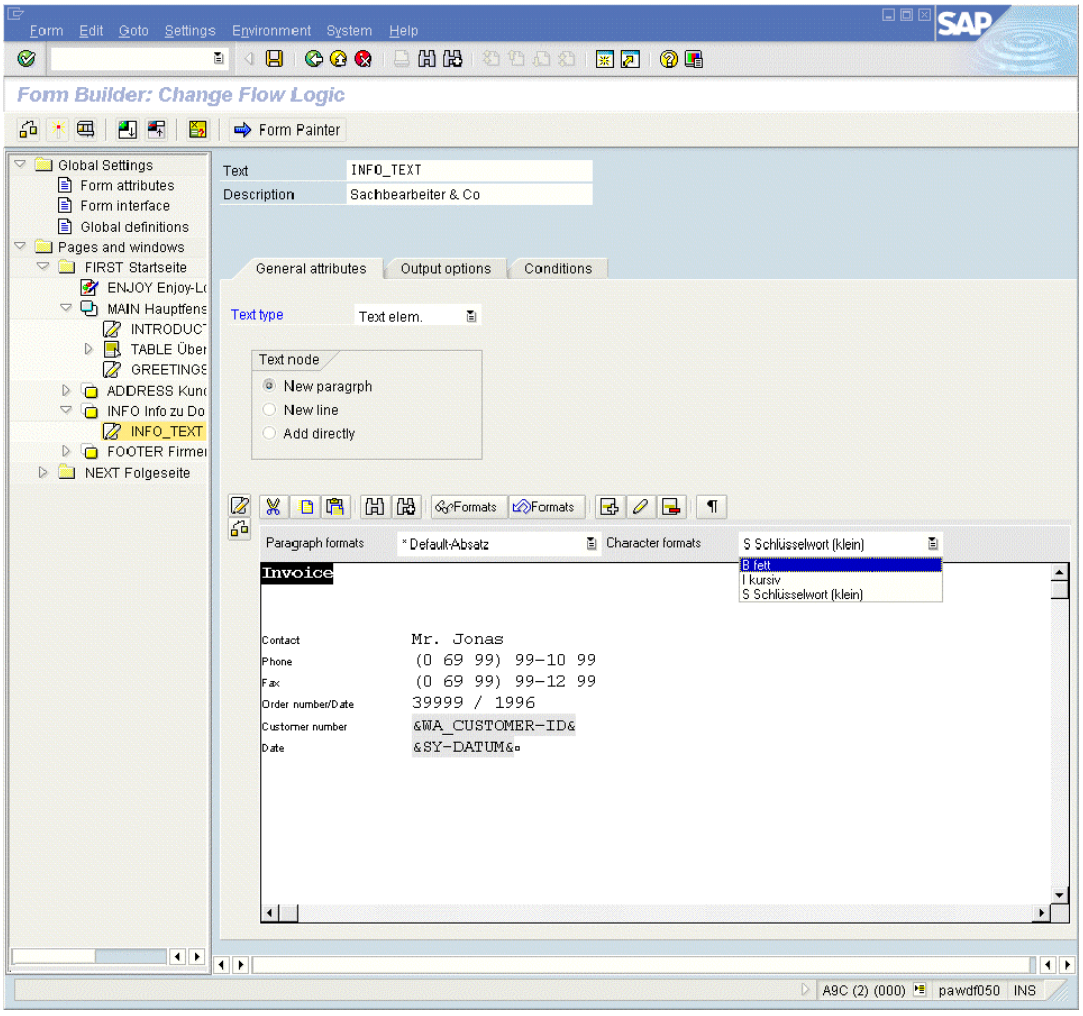
To make detailed settings in the Form Painter (for example, step size of the grid, zoom factors, and so on), choose *Settings* in its toolbar.

PC Editor

You use the PC Editor to enter and format texts and fields.

Among others, the PC Editor offers the following functions:

- Enter, delete, select, cut, and copy texts (see also Entering Text in the PC Editor [Seite 42]).
- Assign paragraph and character formats (for a list of the available formats use the list boxes *Paragraph formats* and *Character formats*)
- Include, change, and delete fields (see also Using Fields in the Form [Seite 65])



The PC Editor displays fields with a gray background.

Table Painter

Use

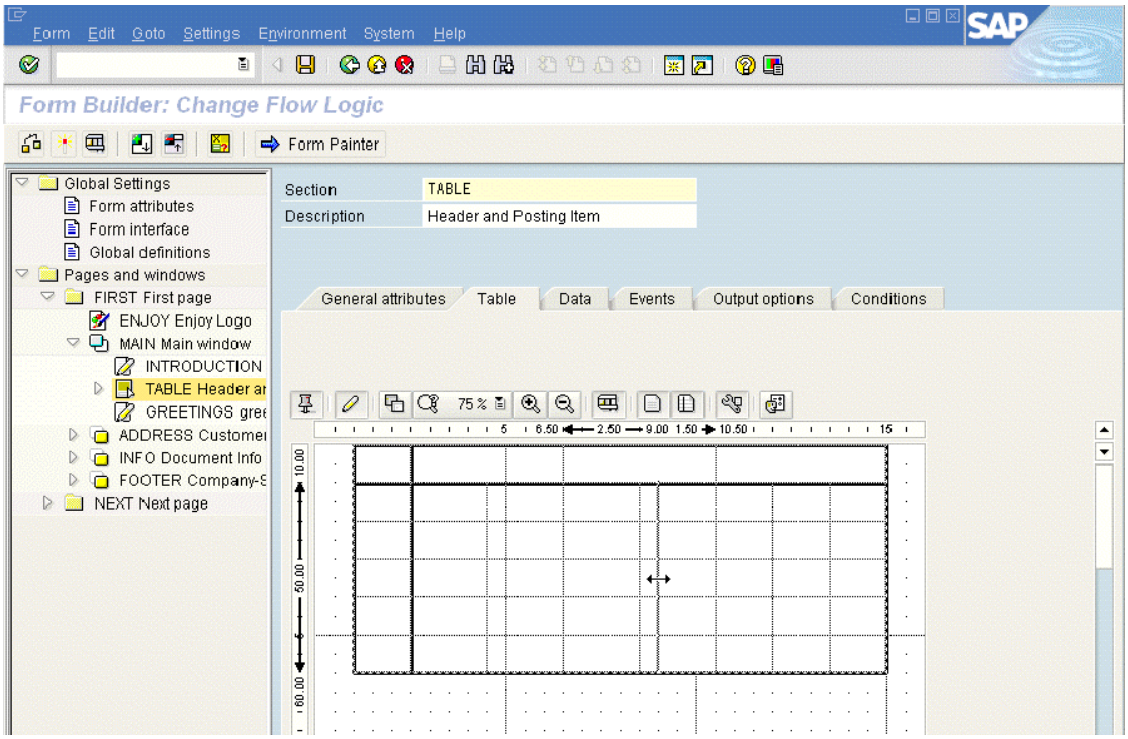
You use the Table Painter to design templates and tables in a window (see also Displaying a Table with Static Data [Seite 52]).

The Table Painter offers the following functions:

- Drawing lines and columns
- Inserting, cutting, copying, and deleting lines
- Deleting cells
- Changing cell size and moving cell separator lines
- Splitting cells
- Selecting table patterns

Activities

To display or suppress the Table Painter, choose *Table Painter on/off* on the *Table* tab on the maintenance screen of a table or template.



The design area does not mirror the size of your window into which you include the table. Therefore, remember to check the size in the ruler.

If you want to insert two tables one after the other, you must create two tables in the window node one after the other.

To execute a function, mark a cell and choose the function in the context menu (right mouse button).

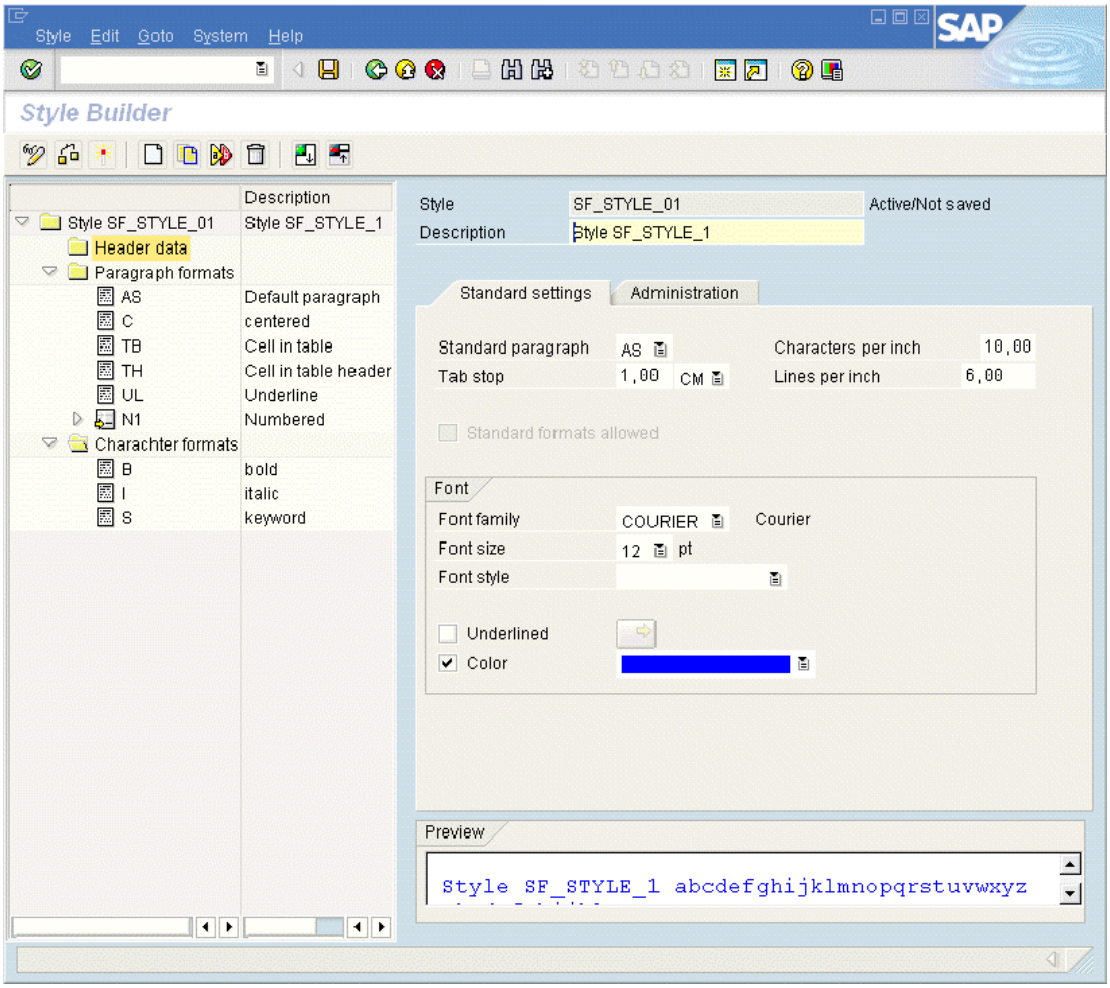
To select a table pattern, select the *Table* tab in the node and choose *Select pattern*.

To make detailed settings in the Table Painter (for example, step size of the grid, zoom factors, and so on), choose *Settings* in its toolbar.

Style Builder

The screen capture shows the Style Builder that you use to define Smart Styles. On the left, you see the style tree which consists of predetermined nodes (header data, folder for paragraph formats, folder for character formats). You can navigate between the nodes and create new nodes. On the right, you see the maintenance screen with its tab pages (here, for example, standard settings for the font in the selected color blue). At the bottom you see the preview of the selected font.

For detailed information on how to create and maintain a Smart Style see Smart Styles [Seite 87].



Field List and Error List

Field List

The field list displays the following data in a tree structure:

- all tables, structures, and fields passed via the form interface [Seite 59].
- system fields [Seite 68] and fields you defined in the global definitions [Seite 64]

This allows you to check that you enter the correct field name when you include a field, and that the Smart Form actually knows the field.

To display the field list, in the Form Builder choose *Field list on/off*.

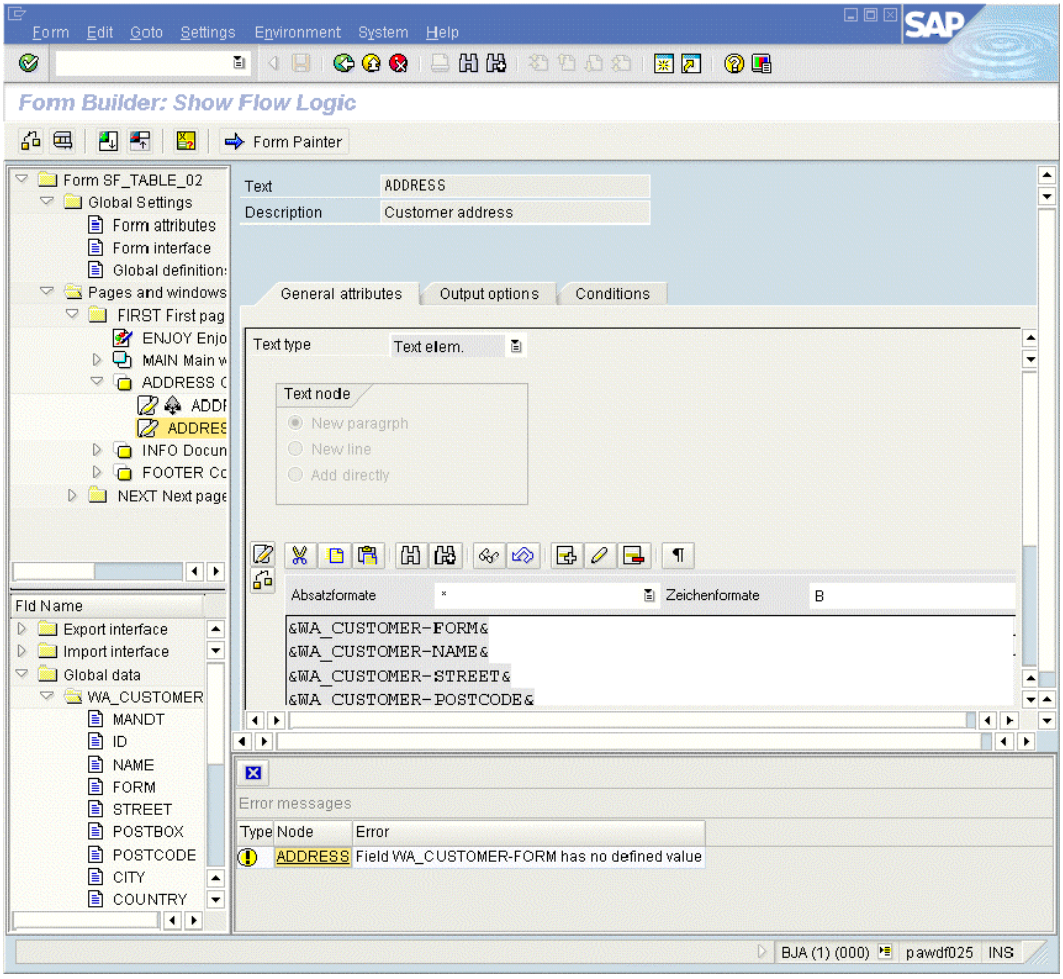
Error List

When you check the Smart Form and any errors occur, the system displays a list of errors and warnings at the bottom of the maintenance screen. To find the error or warning in the tree, select the node in the second column. The system then goes to the corresponding position in the tree and you can correct the error.

To check the entire Smart Form choose *Check*. If an error occurs the error list appears.

For more information on testing and checking forms see Checking and Testing a Smart Form [Seite 92].

Field List and Error List



Node Types: Overview

When you create a Smart Form, the tree structure of the Form Painter already contains two root nodes:

- You use the successors of the *Global Settings* node to maintain *Form attributes*, the *Form interface,* and *Global definitions.*
- You use the successors of the *Pages and windows* node to create the pages of your form, position elements on these pages, and determine the sequence in which you want to process these elements.

Apart from the two root nodes, each node has attributes. You can maintain these attributes on tabs on the right half of the screen. The attributes you can maintain on the tabs *General Attributes*, *Output Options,* and *Conditions* are the same for most of the node types (see: Shared Attributes of the Node Types [Seite 37]).

Successors of Node *Pages and windows*

As direct successors to root node *Pages and windows* you can create only page nodes. The tables below describe these page nodes and the other successors:

Output areas

Node type	Description	Possible direct successors
Page [Seite 39]	A page of the form. Direct successors of this node can be placed directly on the page.	Window, graphic, address
Window [Seite 40]	Output area on a page. There are main windows and subwindows.	All except window and page nodes

Elementary nodes (without successors)

Node type	Description
Text [Seite 41]	You use this node to print any texts (and table contents), except addresses.
Graphic [Seite 50]	You use this node to position graphics in the form. To include background graphics, use node type *Page*.
Address [Seite 48]	You use this node to include an address. The system reads the address data directly from the database tables and formats them for print output.

Table output

Node type	Description	Possible direct successors
Template [Seite 52]	Output of a table containing static data	All except window, page, table, and template nodes
Table [Seite 73]	Output of a table containing application data	As with template node

Flow control

Node Types: Overview

Node type	Description	Possible direct successors
Command [Seite 83]	Executes special commands (next page, paragraph numbering, printer control)	(no successor)
Loop [Seite 82]	Processes successor node repeatedly.	All except window and page nodes
Alternative [Seite 81]	Branches depending on condition.	Direct successors are automatically the nodes TRUE and FALSE. The direct successors of these are all nodes except window and page nodes.

Other nodes

Node type	Description	Possible direct successors
Folder [Seite 57]	Combines successor nodes to logical groups	All except window and page nodes
Complex section [Seite 97]	Combines all attributes of the node types template, table, loop, folder	All except window and page nodes
Program lines [Seite 98]	Executes ABAP program code (for example, conversion routines).	(no successors)

Shared Attributes of the Node Types

The following tab pages are used in common by several node types:

Tabs for general node attributes

Name	Description
General Attributes	Shows contents or description of the node
Output Options	Contains attributes such as *position*, *style*, *box and shading*
Conditions [Seite 80]	Allows to display the node only if certain conditions are true (see also: Flow Control [Seite 79])

The node types *Table*, *Template, Loop, Folder*, and *Complex section* share a different tab in the place of *General Attributes*.

Output Options

Except for the page nodes in the subtree *Pages and windows*, you can always use this tab to determine a box and shading for the respective node. Nodes with text output also have the attribute *Style*. You use this attribute to overwrite for the current subtree the style that was set in the form.

Depending on where you insert a node, it can have additional output options (for example, if the predecessor is a template [Seite 52]).

Basic Elements of a Form

Creating Pages

Use

Each form consists of one or more pages. The first page in the tree structure is the start page (when you create a form, the start page already exists). With this page, processing of the form starts.

When you call pages repeatedly or process pages again because the main window is not filled yet, a page node creates several print pages.

See also: General Concepts on Form Printing [Seite 8].

Procedure

1. Open the context menu for an existing page node and choose *Create* → *Page*.
2. Enter a unique name for the node and a description (for example, business terms).
3. Determine the format and the mode of the page counter [Seite 85] on the *General Attributes* tab. In the default setting, the next page is the page itself.
4. Determine the print attributes of the page on the *Output Options* tab (see F1 help).
5. Determine a background graphic for the entire page on the *Background* tab. For information on how to select the graphic see Printing Graphics [Seite 50]. If you do not determine an *Output mode* in the *Output attributes* box (in print preview only, on the printer as well), the system ignores the background graphic.

If you select an output mode, the user can still choose *Background graphic* in the print preview to display or suppress it.

Result

You can now define the page contents. For each page, the Form Painter shows an individual layout. Whether the page is included into the printout depends on whether it is evaluated during form processing [Seite 99] (a page can be omitted even though it is part of the tree structure).

Creating Windows

Use

Windows are output areas for all output data. You can set the size and position of a window graphically in the Form Painter. There are main windows and subwindows [Seite 11]. The most important difference is that the output in a main window can cover several pages.

After you created a form, a main window already exists on the first page.

Prerequisites

A page [Seite 39] exists on which you want to create a window.

Procedure

6. Open the context menu for an existing page node and choose *Create* → *Window*.

   If you create the window in the Form Painter using the context menu, you can use the mouse to position it immediately.

7. Enter a unique name for the node and a description (for example, booked flights).
8. On the *General Attributes* tab indicate whether the window is a *Main window*. If you want to create a subwindow, leave the checkbox empty.

   For each page you can indicate only one main window.

9. The *Position* and *Size* values in the *Output options* box correspond to the position in the Form Painter. If you want to, maintain other attributes [Seite 37].

Result

You can see the window in the Form Painter and in the tree structure. You can always change the specification of the main window or subwindow.

Positioning Texts on the Form

Use

You display all texts in the form using text nodes. The only exception are addresses [Seite 48], which use their own node.

Integration

The predecessor node [Seite 21] of the text node determines its use:

Examples for using text nodes

Predecessor node	Used to
Subwindow	Exactly position text on one or more pages
Main window	Display text in relation to other nodes in the main window; it may cover several pages
Template	Display texts for table cells of a static table [Seite 52]
Table	Display table contents
Header and footer [Seite 77]	Display column headings and grand totals in tables
Event node [Seite 75]	Display subtotals in a table

Features

There are the following text types:

- *Text element*: to enter new text in the PC Editor [Seite 42]
- `Text module: to` include a text module [Seite 44]
- *Include text*: to include an existing SAPscript text [Seite 46]

Positioning the Text

The position of the text depends on the direct predecessor node (for example, you can assign text of a table cell) as well as on the processing sequence in the tree structure. In addition, you determine output of a text node in relation to the preceding node on the same level [Seite 21]. You can append the text directly to the output of that node or start it in a new paragraph or a new line (select via radiobutton).

Entering Texts in the PC Editor

Use

You use the PC Editor [Seite 28] to enter new text. The position of this text on the form is determined by the predecessor node.

To include data from the form interface (data from application tables) or system data (date, time) into the text, use system fields or user-defined fields, respectively, in the text (see Using Parameters in the Form [Seite 58]). When processing the form, Smart Forms replace these fields with the corresponding values.

Procedure

1. To create a text node, call the context menu for that node in the tree structure that should receive the text. Then choose *Create* → *Text*.
2. Enter a unique name for the node and a node description (for example, letter).
3. On the *General Attributes* tab choose *Text element* as text type.

   Since you edit a text element, an included text, or a text module exclusively using the text node, the system stores only information on the selected type. When you change the text type, the system therefore asks for your confirmation.
4. Enter your text in the PC Editor
   - either directly on the tab if you use the inplace version of the PC Editor
   - or choose *Text editor* to go to the fullscreen mode of the PC Editor. If, after entering the text, you use the green arrow (F3) to leave the fullscreen editor, the system transfers the text into the inplace version.
5. In the *Text node* box choose whether you want the text to start with a new paragraph or only in a new line. You can also choose to append the text directly to the end of the current paragraph.
6. If desired, choose the Output Options [Seite 37] tab to maintain the style or box and shading of the text.

Result

The system displays the node in the tree structure, including its name and description.

Including Text Modules

Use

You use the text type *Text module* of the text node [Seite 41] to refer to an existing text module [Seite 86]in the system. This allows you to easily use texts from text modules in several forms. In addition, it is not necessary to load the entire form description to maintain these texts.

You can use text modules in two ways:

- *Refer* to the text module. The text then appears read-only in the PC editor and is included when you print the form.
- *Copy* the text module. The system then copies the text of the module and automatically converts the text node into an editable text element.

Text modules can be used cross-client and are connected to the transport and translation systems.

Prerequisites

The text module you want to include must exist in the system. To create a text module use the SAP Smart Forms initial screen [Seite 23].

Procedure

5. To create a text node, call the context menu for that node in the tree structure that shall contain the text and choose *Create* → *Text*.
6. Enter a unique name for the node and a description (for example, business terms).
7. On the *General attributes* tab select the text type *Text module*.

Since you edit a text element, an included text, or a text module exclusively via the text node, the system stores only information on the selected type. When you change the text type, the system therefore asks for your confirmation.

8. Enter the name of the text module in the *Text name* field. Or click on the black arrow and enter the name of a field to be evaluated dynamically [Seite 58].
9. If you want to change the text of the text module for the current form, choose *Copy*. The Form Builder changes the text type to *Text element* and copies the text of the module into the PC Editor, where you can edit it. In this case, the original text module remains unchanged.
10. Use the checkbox *Always copy style from text* to determine that the style of the text module is of higher priority than that of the text node or any style inherited from a predecessor. This is important if you specified the name of a field in step 4 (see F1 help).
11. If required, use the Output options [Seite 37] tab to maintain attributes for style, box and shading of the text.

Result

The system displays the node including name and description in the tree structure. It includes the relevant text itself in the moment the form is processed. If you refer to a text module, it appears in the PC Editor in read-only mode, if you copy it you can edit it.

Including SAPscript Texts

Use

You use the text type *Include text* of the text node to refer to a SAPscript text that already exists in the system. To identify the text, you need the *text name*, the *text object*, the *text ID*, and the *language*. Thus, you can easily use the texts in several forms. In addition, you need not load the entire form description [Seite 18] to maintain these texts.

This text type corresponds to the SAPscript statement INCLUDE. However, Smart Forms do not allow any control statements in include texts. When processing the form, they are simply ignored.

To administer application-specific form texts, applications use transaction SE75 to define their own text objects with subordinate text IDs. Use text object *TEXT* with text ID *ST* to enter general standard texts. To maintain them, use the standard text editor (transaction SO10).

If there is no need for you to use old SAPscript texts, you better use text modules [Seite 44]. They can be used cross-client and are connected to the transport and translation systems.

Procedure

1. To create a text node, call the context menu for that node in the tree structure that shall receive the text and choose *Create* → *Text*.
2. Enter a unique name for the node and a node description (for example, letter).
3. On the *General Attributes* tab choose *Include text* as text type.

   Since you edit a text element, an included text, or a text module exclusively via the text node, the system stores only information on the selected type. When you change the text type, the system therefore asks for your confirmation.
4. In the *Text key* box, identify the include text.
   - To identify individual text objects, use the search help in this field.
   - The search help of the *Text name* field allows you include the attributes of SAPscript texts into the search.
7. If required, use the *Paragraph formats* box to format the include text:
   - The style assigned to the text node contains a format for the standard paragraph ('*****'). If you enter a paragraph format in the *Standard paragraph* field, this format overwrites the style format for all standard paragraphs in the include text that use this paragraph format.
   - Use the *First paragraph* field to set a paragraph format for the first paragraph of the include text. This format also overwrites the format set in the style. If you set the *First paragraph* field but leave the *Standard paragraph* field empty, the system uses the format set in the *First paragraph* field for any standard paragraphs in the include text.

6. In the *Text node* box choose whether you want the text to start in a new paragraph, in a new line, or directly at the end of the current paragraph.
7. If desired, choose the Output Options [Seite 37] tab to maintain the style or box and shading of the text.

Result

The system displays the node in the tree structure, including its name and description. The included text is included only at the moment the form is processed.

Inserting Addresses

Use

In many applications, addresses are administered using the Central Address Administration [Extern]. Depending on how the address is used, the application uses a particular address type [Extern]. The addresses are stored in the database tables of the central address administration and identified by the application via a number.

You use the address node to insert an address into the form. This guarantees that the address is formatted according to the postal rules of the sender country.

Prerequisites

Your application must administer addresses using the central address administration. Otherwise you must use the text node [Seite 41] to insert your addresses.

Procedure

1. To create an address node, call the context menu for that node in the tree structure that you want to contain the text and choose *Create* → *Text*.

   Create the address node as direct successor of the page node or use the context menu in the layout of the Form Painter. You can then position the address anywhere on the page.

2. Enter a unique name for the node and a description (for example, customer address).
3. Determine the address type on the *General attributes* tab.

   For a description of how to determine the address type dynamically and how to use fields instead of fixed values, see Using Parameters in the Form [Seite 58].

4. For organization addresses you must specify only an address number. For any other address types, you must specify a person number and an address number. To do this, use the search help.
5. In the box *Additional address specifications* you can maintain other attributes to specify how you want to display the address:
   - Use field *Output starts with paragraph* to set the paragraph format of the style that you want to use to display the address.
   - If the number of lines you specify in the field *Number of lines to be used* is smaller than the number of lines required to display the address, the central address administration suppresses lines of the address.
   - As default, the system uses as sender country the country that was specified when creating the address. You can use the parameter *Sender country* to overwrite this setting.
   - For addresses that have a P.O. box as well as a street address, use the other fields in the box to determine which of these addresses to display.

6. If desired, you can use the Output Options [Seite 37] tab to maintain attributes for style, box, and shading of the text. In addition, you can set values for position and size of the output area (instead of in the Form Painter).

Result

The system displays the node including name and description in the tree structure. If you insert the address as direct successor of a page or by using the Form Painter, the system displays an extra output area for the address. The address itself is inserted only at the moment the form is processed. You cannot view it in the Form Builder.

Printing Graphics

Use

You use the graphic node to display graphics, such as a company logo, on the form. For performance reasons, make sure that the graphics are held in the printer memory. SAP Smart Forms support this method, provided that:

- The printer can be controlled accordingly.
- There is enough memory space on the printer.
- You activated this property in transaction SE78 (see below).

In this case, the system sends the graphics to the printer only once during one print job.

To include background graphics, use the *Background graphic* tab of a page node.

Prerequisites

You use the *SAPscript Graphic Administration* (transaction SE78) to import graphics into the SAP-System:

1. Double-click on a graphic format in the tree structure in the forder *Document server* → *GRAPHICS General graphics*.
2. Choose *Graphic* → *import*.

The transaction imports the graphic and stores it on the Business Document Server (BDS). Now you can display it on a form.

See also: Graphic Administration [Seite 104]

Procedure

1. To create a graphic node, call the context menu for that node in the tree structure that you want to contain the graphic and choose *Create* → *Graphic*.

   Create the graphic node as direct successor of the page node or use the context menu in the layout of the Form Painter. You can then position the graphic anywhere on the page.

   In the Form Painter the system displays a small box for the newly created node.
2. Enter a unique name for the node and a description (for example, company logo).
3. On the *General Attributes* tab determine whether you want to include a colored graphic or a graphic in black and white.
4. Use the fields *Object*, *ID,* and *Name* to identify the graphic. Use F4 help of the *Name* field to copy the values of these fields. If you copy them with `ENTER`, the system displays the graphic in the Form Painter.

The box for the graphic in the Form Painter also contains the name of the graphic node, which in part overlays the graphic.

5. In the *Technical Attributes* box determine the graphic resolution according to what the printer supprts. Small resolutions increase the size of the graphic on the form, higher resolutions reduce the size.
6. For graphic nodes that are displayed in a window, there is a box *Horizontal position* in addition to the general output options [Seite 37]. The parameters *Reference point* and *Alignment* determine the horizontal position in the window, while the vertical position is determined by the previous output in the window.

   At present, you cannot overlay graphics with text. However, if you use a template

   node [Seite 52], you can display graphics and text side by side.

Result

The system displays the node with name and description in the tree structure. If you insert a graphic as direct successor of a page or if you used the Form Painter, the system displays the graphic. Graphics in windows are inserted only when the form is processed. You can therefore not see them in the Form Builder. In addition, the system automatically dispalys the graphic on the next page if there is not enough space left on the current page.

Displaying a Static Table

Use

Use node type *Template* to display a table whose layout and size (number of lines and columns) is determined before the runtime of the print program.

To create a template, define a table layout to determine the cell structure for each line. The cells are used to display the contents of the successor nodes of the template node. This allows you to position text and a graphic side by side (see Displaying Graphics in Templates [Seite 56]).

The template node is also suited for label printing.

Integration

You can use system fields or your own fields — for example, to display the current date — in table cells (see also Using Parameters in the Form [Seite 58]).

To display application data, use the table node. For this node type the number of

table lines to be displayed is not known before the form is actually printed (the number depends on the volume of application data selected in the data retrieval program [Seite 59]).

Prerequisites

You can create a template node only as a successor to a window node. If you want to display the template in a particular area of the page, create a subwindow for it. Otherwise, use the main window.

Features

The node type displays a table layout [Seite 53]. You can either enter the table layout directly or use the Table Painter to define it.

To set other attributes, use the attributes the node types share [Seite 37].

Activities

To use a template, you must:

1. Define the table layout [Seite 53].
2. Display contents in the table cells [Seite 55].

Defining the Table Layout

Use

You use the table layout to determine:

- The number of lines and cells
- The height of each line
- The width of each cell
- The alignment of the table in the window
- Whether and where to display separator lines or frames

Procedure

Create a template node and maintain the attributes *Width*, *Horizontal alignment*, and *Vertical alignment* (See F1 help).

Use the table control on the *Template* tab to define the layout of the lines. Each line of the template must have a definition. You can use one line in the table control to define either the layout of one template line or a common layout for several template lines:

- Use the columns *From* and *To* to specify the lines of your template, for which the definition applies. The numbering starts with 1. The intervals in the line definition may neither overlap nor leave gaps.
- The *Height* you specify applies for all cells of this line. Specify a *Width* for each cell you want to display. The sum of the width values must amount to the specified *Width* of the template.
- You can reuse the *Name* of the template line for other lines of the same template. Enter it in the *Reference* column of the desired lines. The system then copies the values you specified for the line height and the individual cell widths.

Use the *Pattern* box to select the desired table pattern.

Example

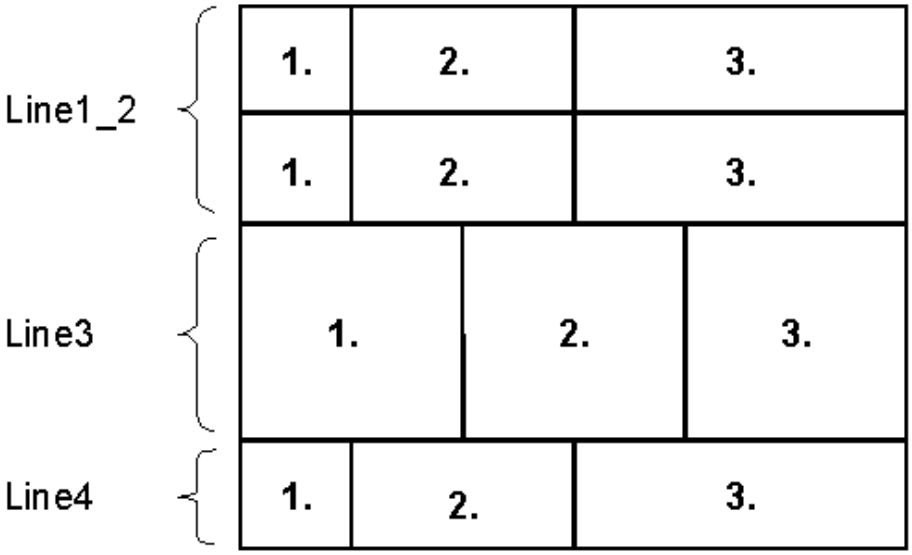
You define the following lines in the table control (in the example without measurement units):

Name	From	To	Reference	Height	1.	2.	3
Line1_2	1	2		1	1	2	3
Line3	3	3		2	2	2	2
Line4	4	4	Line1_2	1	1	2	3

You use the *Pattern* box to specify that all cells are separated by visible lines and that the template is framed.

Result

On the form, the template may look like this:


Line1_2
1.
2.
3.
1.
2.
3.
Line3
1.
2.
3.
Line4
1.
2.
3.

Displaying Contents in Cells

Use

The template node defines the table layout [Seite 53]. The successor nodes of the template node determine the data to be displayed in the table cells.

Prerequisites

You already created a template node as successor of a window node and defined the table layout.

Procedure

1. Use the context menu of the template node to create successor nodes.

   For clarity reasons create one folder node [Seite 57] for each line. The folder node is an outline node that you can use to combine related nodes. For folder nodes, you need not fill in the input fields in the *Output structure* box (see next step).

2. The *Output options* tabs of the inserted nodes now contain an additional box *Output structure* with the fields *Line* and *Column*. Assign each node to the cell in which you want the output to appear.

   You can assign several nodes to one cell. The output sequence within the cell is determined by the sequence of the nodes in the tree.

Result

The system displays the contents in the print preview or prints them, respectively.

Displaying Graphics in Templates

Use

If you insert graphic nodes as successors of a template node, the *Output options* tab contains the *Horizontal position* box. The horizontal position usually refers to the output window.

This procedure describes how you position a graphic within a template.

Prerequisites

You already created a template node as successor of a window node and inserted a graphic node for a cell (see Displaying Contents in Cells [Seite 55]).

Procedure

Horizontal Position

You can determine the horizontal position of a graphic within a cell only in relation to the left cell margin. On the Output options [Seite 37] tab of the graphic node you find the *Horizontal position* box. Set the *Reference point* to *Window* and the *Alignment* to *Left*. In the printout, the graphic appears at the left cell margin.

You can determine the horizontal position via the distance from the left margin (input field next to the alignment.

Vertical Position

By default the graphic is displayed at the upper cell margin. To influence the vertical position of a graphic within a cell, insert other nodes before the graphic node, for example, a text node whose output consists of blanks. You must assign this node to the same cell as the graphic node. Now the graphic is shifted downwards.

Result

The system positions the graphic in the cell according to the settings you made.

Combining Nodes

Use

The more extensive a form becomes, the less clear becomes the node hierarchy in the tree structure. To avoid this, you can combine related nodes by appending them under a folder node.

Nodes assigned to cells of a template [Seite 52] can better be identified in the tree structure if there is a folder for each line or column.

Depending on how long the last date for payment has been exceded, each dunning letter should start with a different introduction. To to this, you create a text node for each dunning level that contains the relevant dunning letter and specify conditions [Seite 80] to print only one of the nodes. In this case, it is reasonable to combine the different text nodes in a folder.

Procedure

1. To create a folder node call the context menu for that node in the tree structure that shall contain the text, and choose *Create* → *Folder*.
2. Enter a unique name for the node and a description (for example, cells of the third line).
3. To assign other nodes to the folder, use Drag&Drop to pull them to the folder node or insert new nodes as successors.

On the *Events* tab you can activate a header and a footer area. For more information see Displaying Table Data [Seite 73].

Using Parameters in a Form

Use

You want to merge data into a form that has not been determined when you create the form. This data could be:

- Data selected from database tables of your application. You pass them in the form interface.
- Data provided by the SAP Standard or by Smart Forms. This could be, for example, the date or the current page number.
- Data you calculated or introduced yourself in the form, for example to display totals in a table.

All this data is determined only at runtime of the form and then included into it. You use fields as placeholders in the form to determine how and where to display this data. To include these fields, you use text nodes [Seite 41]. At runtime the system replaces them with the corresponding values.

Features

Smart Forms provide the following possibilities to merge data into the form:

- A form interface in which you define parameters for application data (see Passing Data to the Form [Seite 59]). You can use these parameters as fields for displaying table data [Seite 73].
- Global definitions [Seite 64] that you use to define your own fields [Seite 65] (for example, work areas to display internal tables)
- System fields [Seite 68] whose contents are predetermined (date, page number)

Passing Data to the Form

Purpose

You can pass any data retrieved by the data retrieval program to the form using the form interface. Since the system generates a function module from the Smart Form as soon as you activate it, the form interface equals that of the generated function module. For this reason, you can pass the same data in the form interface as you could pass to a function module. This includes data from database tables which the application wants to display on the form, but also variables that control output to the form. In addition, you can define exceptions to which the data retrieval program must react. To determine other settings for form printing, Smart Forms offers standard parameters whose definitions you cannot change.

See also: Architecture [Seite 14] and Creating Forms Using SAP Smart Forms [Seite 16].

Process Flow

1. In the data retrieval program, select all data you want to display in the form.
2. In the Form Builder define the form interface [Seite 60]. You know the parameter types from the data retrieval program.
3. Activate your form. Smart Forms generates a function module. The application must call this function module to print the form.
4. Call the Smart Form [Seite 61] in your application. You can copy the interface from the function module.

Result

The generated function module triggers spool processing. The user can now preview the form and print it.

Defining the Form Interface

Use

You define the form interface to pass data to the form.

In the interface you can define import and export parameters, tables, and exceptions.

Prerequisites

The data retrieval program provides the data you want to include into the form. Therefore, you know the types of the corresponding variables.

Procedure

1. Click the node *Global Settings* → *Form interface*.
2. Enter the parameter names for the data you want to pass. These names are valid throughout the form; they need not correspond to the variables names in the data retrieval program. Standard parameters of Smart Forms are not input-enabled.

   As export and import parameters any data types are allowed. On the *Tables* tab, however, you can pass only tables with a flat structure.
3. Enter the appropriate reference type. You can assign a type using the keywords **`LIKE`** or **`TYPE`**. Double-click on the reference type to display its definition.

   You can also use F1 help and F4 help for the input fields.
4. If desired, define *Exceptions* for the form.
5. Use the local check on each tab page to check your entries for the form interface.

Result

You defined the form interface. You can use any parameters specified there as fields [Seite 65] in the form.

Integrating the Smart Form into the Application

Use

You trigger form printing by calling only two function modules. The first module uses the name of the form to determine the name of the generated function module. Then you call this module.

The name of the generated function module is unique only within one system. Therefore, you must always call the function module first that uses the form name to determine the current name of the generated module.

Prerequisites

You defined the form interface in your form and activated the form.

Procedure

1. In the Form Builder call the function *Environment* → *Name* of the function module and use **STRG-Y** and **STRG-C** to copy its name.
2. In the data retrieval program define a variable of type **rs281_fnam** for the name of the generated function module:

```
data  fm_name type rs38l_fnam.
```

You can call the Smart Form in other parts of the application program as well. However, in this case you must make sure that the system can access the data to be passed from that place. We recommend to encapsulate the data retrieval in a function module as well.

3. If desired, you can call the function module **SSF_FIELD_LIST**. It returns a list of the form parameters actually used in the form. You can use this information to limit data selection, if necessary.
4. Call function module **SSF_FUNCTION_MODULE_NAME**. It returns the name of the generated function module:

```
CALL FUNCTION 'SSF_FUNCTION_MODULE_NAME'
  EXPORTING
    FORMNAME              = '<form name>'
  IMPORTING
    FM_NAME               = fm_name
  EXCEPTIONS
    NO_FORM               = 1
    NO_FUNCTION_MODULE    = 2
    OTHERS                = 3.

IF SY-SUBRC <> 0.
   <error handling>
ENDIF.
```

5. Call the generated function module. To do this, use the *Insert statement* function for **CALL FUNCTION** in the ABAP Editor and use the name you copied in step 1. (to avoid having to copy all interface parameters manually). Then replace the function module name with the variable **fm_name** defined in step 2.

```
CALL FUNCTION  fm_name
  EXPORTING
*   ARCHIVE_INDEX              =
*   ARCHIVE_PARAMETERS         =
*   CONTROL_PARAMETERS         =
*   MAIL_APPL_OBJ              =
*   MAIL_RECIPIENT             =
*   MAIL_SENDER                =
*   OUTPUT_OPTIONS             =
*   USER_SETTINGS              = 'X'
    G_CARRID                   = <variable>
    G_CONNID                   = <variable>
    G_FLDATE                   = <variable>
* IMPORTING
*   DOCUMENT_OUTPUT_INFO       =
*   JOB_OUTPUT_INFO            =
*   JOB_OUTPUT_OPTIONS         =
  TABLES
    GT_SBOOK                   = <internal table>
  EXCEPTIONS
    FORMATTING_ERROR           = 1
    INTERNAL_ERROR             = 2
    SEND_ERROR                 = 3
    USER_CANCELED              = 4
    OTHERS                     = 5.

IF SY-SUBRC <> 0.
   <error handling>
ENDIF.
```

In this example, three variables and an internal table are passed. The parameters **G_CARRID**, **G_CONNID**, **G_FLDATA**, and **GT_SBOOK** have been defined before in the form interface.

6. In the interface pass all data you want to transfer to the form.

Result

The generated function module processes the form logic defined in the Smart Form. Its output is sent to the printer spool for processing.

As long as you do not change the form interface, you can make any changes to the form. When you activate it again, the system generates the current version of the form as soon as you call the function module. Only if you change the form interface, you must adapt the interface in the data retrieval program.

Global Definitions

Use

Global definitions apply for the entire form. You can use any objects defined there in all nodes of the tree.

Features

The following global definitions are possible:

- Variables or constants whose types are stored in the ABAP Dictionary
- Types as free ABAP coding, provided there is no type in the ABAP Dictionary
- Field symbols [Extern]

You can initialize the global data before you start processing the start page. Usage examples are: conversion of selected application data, optimizing form logic by previously reorganizing internal tables, and so on.

On the *Form routines* tab you enter routines that you want to use in the form via the *program lines* node. Within these form routines you cannot access any global data unless you explicitly pass them to the form routine interface.

Fore more information see Advanced Form Development [Seite 95].

Activities

To maintain global definitions use the node *Global Settings* → *Global definitions*.

Using Fields in the Form

In the SAP System, texts usually exist not without context but refer to an object stored there. For a letter, this can, for example, be address data from the vendor master record or information from the material master record to be included into an order text. In these texts you need not enter the current values, but you use placeholders for this data. This allows you to make text modules flexible, since you can define any variables by using these placeholders.

In SAP Smart Forms these placeholders are called **fields**. Fields represent data that is included into the text at the moment it is formatted for output. When generating the output, the system replaces all fields in the text with their currently valid values.

There are two types of fields:

- System fields [Seite 68] provided by Smart Forms
- Your own fields, which you define in the form interface [Seite 60] or declare as global definitions [Seite 64]

In the Form Builder choose *Utilities* → *Field list on/off* to display a field string [Seite 33] beneath the tree structure. In this field string all fields are sorted according to their definitions. This allows you to keep track when including the fields [Seite 66].

Where can you Include Fields

In Text Nodes

You include fields in text nodes if you want to display values at runtime. After including these fields, they appear with a gray background and are enclosed in ampersands ('**&**') to set them apart from normal text. This is the right method to print tables in the form [Seite 73].

In Input Fields of a Node

Usually, these are your own fields which you defined in the form interface or in the global definitions (for example, internal tables or control parameters). In this case you enter the field name without ampersand ('**&**') into the input field. If you want to process a node depending on a condition [Seite 80], use a field whose value you want to check at runtime.

Including Fields in the PC Editor

Use

You include fields (your own fields and system fields) to display variable data in the form. The field serves as placeholder. At the moment the system displays the form, it replaces the field with the value that was assigned to it during processing.

This procedure describes how to include fields into texts to print values that are valid at runtime. You create these texts using a text node [Seite 41].

If the text node refers to a text module [Seite 44] or includes a SAPscript text [Seite 46], the system copies all fields of the included text.

Prerequisites

If your own fields are concerned, you must have defined them either as global data or as parameters in the form interface.

The text node into which you want to include a field must exist.

Procedure

In order for Smart Forms to distinguish fields in the text from normal text, you must use a special PC Editor function to include fields.

You can:

- Include fields in the field list [Seite 33] using Drag&Drop. The system inserts the field at the position of the text cursor.
- Include fields using the PC Editor functions.

To do this, proceed as follows:

1. In the inplace version of the PC Editor there are three functions for fields:
   - *Include field* () to include a defined field into the text
   -
   - *Change field* ()to change an included field
   -
   - *Delete field* ()to delete an included field from the text

   Choose *Include field*.
3. A dialog window appears. Enter the field name in a line. Make sure to enclose the field name in ampersands ('**&**').

   Observe the syntax rules [Seite 70] for field names. In addition to the field name you can specify formatting option [Seite 71] for the output of the field contents.
4. Confirm your input to return to the editor.

5. Use the local check to check whether the system can find the definition of the entered field name.

Result

The system displays the field with gray background to set it apart from the other text.

System Fields

Within a form you can use the field string **SFSY** with its system fields. During form processing the system replaces these fields with the corresponding values. The field values come from the SAP System or are results of the processing.

System fields of Smart Forms

Field name	Description
&SFSY-DATE&	Displays the date. You determine the display format in the user master record.
&SFSY-TIME&	Displays the time of day in the form HH:MM:SS.
&SFSY-PAGE&	Inserts the number of the current print page into the text. You determine the format of the page number (for example, Arabic, numeric) in the page node. See also: Creating Pages [Seite 39].
&SFSY-FORMPAGES&	Displays the total number of pages for the currently processed form. This allows you to include texts such as 'Page x of y' into your output. See also: Creating Pages [Seite 39].
&SFSY-JOBPAGES&	Contains the total page number of all forms in the currently processed print request. See also: Creating Pages [Seite 39].
&SFSY-WINDOWNAME&	Contains the name of the current window (string in the *Window* field)
&SFSY-PAGENAME&	Contains the name of the current page (string in the *Page* field)
&SFSY-PAGEBREAK&	Is set to '**X**' after a page break (either automatic [Seite 11] or command-controlled [Seite 83])
&SFSY-MAINEND&	Is set as soon as processing of the main window on the current page ends
&SFSY-EXCEPTION&	Contains the name of the raised exception. You must trigger your own exceptions, which you defined in the form interface, using the **user_exception** macro (syntax: **user_exception** <exception name >). See also: Checking and Testing a Smart Form [Seite 92].

When using the fields **&SFSY-FORMPAGES&** or **&SFSY-JOBPAGES&** you must keep all output pages in the main memory til the end of the form or the print job, to allow these fields to be replaced with their respective values. For large forms or print jobs, this may require a huge amount of memory space.

Field Syntax

You call a field by its name. This field name must correspond to the following rules:

- The name must be enclosed in ampersand characters ('**&**').
- The name must not contain any blanks. It must not contain the characters **'+()** since they are used to define formatting options [Seite 71].
- The name can be up to 130 characters long. However, for a distinction of the existing fields, the system uses only the first 32 characters.
- There is no difference between uppercase and lowercase letters in field names. The field names **&myfield&**, **&MYfield&** and **&MYFIELD&** all identify the same field.
- You cannot use the names of system fields for defining your own fields.

Fields that are placeholders for a structure refer to the contained fields by the minus character ('-').

These are valid field names: **&field&**, **&MY_field&**, **&KNA1-NAME1&**, **&SFSY-DATE&**, **&KNA1-UMSAT(I)&.**

Output Options for Field Contents

Use the *Formatting options* to adapt the value of a field before printing it. You can enter the relevant parameters directly behind the field name. Make sure to write the short forms of the different options in uppercase letters. Some of the options can be combined.

General Information

The formatting options are not suited for all data types of fields (for example, for character fields you need no exponential representation). You must distinguish between numeric fields and character fields.

Numeric Fields

- If specified, the system first evaluates the length (<length>).
- If no length is specified, the system displays the value in its overall length.
- The trailing blank indicates a positive sign. To suppress it, use formatting option **S**.
- Any offset <offset> specified is ignored.

Sequence of evaluation: (<length>), sign to the left(**<**),Japanese date (L), suppress blanks (C), right-justified display (R), insert fillers (F).

Character Fields

By default, the system displays the value of a field in its overall length, but truncates trailing blanks.

Sequence of evaluation: suppress blanks (C), <offset> and (<length>), right-justified display (R), insert fillers (F).

Overview

Formatting Options for Fields

Syntax	Description
&field+**<offset>**&	Skips <offset> places of the field value (character fields only). If the offset is greater than the length of the value, nothing is displayed.
&field(**<length>**)&	Sets the output length to <length>.
&field(*)&	If the field is defined by a Data Dictionary type, Smart Forms set the output length to the value specified there.
&field(S)&	Suppresses the sign
&field(<)&	Displays the sign to the left of the number
&field(.**<nat.number>**)&	Limits output of decimal places to <nat.number>
&field(E**<nat.number>**)&	Displays the field value with the fixed exponent <nat.number>. The mantissa is adapted to this exponent by shifting the decimal character and inserting zeros.

Output Options for Field Contents

&field(T)&	Suppresses thousand indicators when displaying fields of types **DEC**, **CURR**, **INT**, and **QUAN**.
&field(Z)&	Suppresses leading zeros of numbers
&field(I)&	Suppresses display of initial values
&field(K)&	Deactivates a conversion routine specified in the Data Dictionary.
&field(R)&	Right-justified display. Use this option only when specifying an output length as well.
&field(F**<filler>**)&	Replaces left-justified blanks in the value by the fill character <filler>.
&field(L)&	Converts and a date field to a local date and displays it. The date is then formatted using edit mask **JPDAT**. Since this representation uses Japanese characters, use it in the Japanese version of the SAP System only.
&field(C)&	The system takes the field value as a sequence of words separated by blanks. Option C shifts these words to the left and leaves only one blank inbetween as separator. Any leading blanks are suppressed. This effect corresponds to that of the ABAP statement **CONDENSE.**

Displaying Table Data

Use

You use the form interface to pass internal tables that you filled before with the data retrieval program. To display table contents, you read each line of the table separately in a loop, then process and display it. While processing this loop, you don't know how many table lines are still to come (unlike with templates [Seite 52], where you specify the number of columns and lines explicitly). This is why you display tables in the main window [Seite 40].

Features

The node types [Seite 35] below offer a *Data* tab, which you can use to read table lines in a loop:

Node type for table output	Purpose
Table	Displays a table. Offers tabs to determine attributes for table layout, data selection, header and footer.
Loop [Seite 82]	Accesses table data in a loop. The tabs offered match those of the table node, but you cannot define a table layout.
Complex section [Seite 97]	Combines the properties of the node types table, loop, template, and folder. Designed for advanced form developers.

You cannot nest tables, which means that you cannot create a table node as a successor of a table node. A loop, however, can have several tables as successor nodes.

Activities

To display a table, proceed as follows:

1. Use the context menu for a window to create a table node. As an alternative, use a complex section.
2. Use the Table tab [Seite 74] to define a table layout. In steps 4. and 5. determine the line types for the output.
3. Use the Data tab [Seite 75] to select your data.
4. Use the Events tab [Seite 77] to determine the table header and footer output (for example, column headings and grand totals).
5. Determine the contents of the table [Seite 78].

Table Tab

Use

You use this tab page to determine the table layout for a table node (or a complex paragraph).

In contrast to the template node, where you fix all lines and columns of the table, you determine mere table types on the *Table* tab. You select these table types when displaying the successor nodes.

Specifying the Table Layout by Line Types

You use a line of the table control of the *Table* tab to define a *Line type*. The line type specifies:

- How many cells the line contains and how wide these cells are. The sum of all cell widths must correspond to the table width. In addition, the table width must never excede the width of the output window.

  The height of the table line is determined dynamically and depends on the cell contents as well as on the style used. The highest cell determines the height of the line.

- Whether the table cells within the line are protected against page breaks (mark *No page break* field)
- Which line type to use if no line type is specified for the output of the node

  Determine a line type for the table heading, a line type for the items in the table, and a line type for the output of a total. To do this, fill in three lines in the table control.

Defining the Line Types

To describe the line types, use either the table control or the Table Painter (to call it, use a pushbutton on the tab page).

General Settings for the Table Layout

The attributes in the *Table pattern* box and for the table width as well as the *Horizontal alignment* apply for the entire table. These attributes are identical with the attributes of the template (see: Defining the Table Layout [Seite 53]).

Data Tab

Use

You use the *Data* tab to select data from internal tables for a table node or a loop node or a complex paragraph.

If you mark the radiobutton in the left upper edge of the tab page, you deactivate the entire tab. In this case, you do not select any data and use the node similar to a template node [Seite 52] (with the one difference that you create the table using line types).

Selecting Data

1. If you want to use a loop to read and process the lines of the internal table one after the other, mark the checkbox in front of *internal table*.
2. Enter at least the name of one internal table.
3. If the table has a header line, the system automaticall uses it. If not, enter the assignment type (`INTO` or `ASSIGNING`) and a work area (structure of the table line type or field symbol).
4. If desired, use the input fields *Line* and *to* to determine the part of the internal table you want to read.
5. Use the *WHERE condition* box to select only part of the data from the internal table. Describe the conditions as you did on the Conditions tab [Seite 80].

The system selects the data at runtime according to your settings. After processing the loop, the output area contains the last entry selected (or the field symbol **points** to this entry).

Sorting and Control Levels

You can use the second table control of the *Data* tab to sort the table output by fields of the internal table. The sequence in which you enter the field names into the table control determines the sorting sequence. To change it later, you can use the black arrows above the control. Use the rediobuttons to specify whether to sort the field in ascending or in descending order.

For technical reasons Smart Forms cannot see whether the internal table was sorted

before (for example, in the data retrieval program). If yes, you must still enter the sort sequence into the table control and mark *already sorted*.

A control break results from the sort process whenever the value of a sorted field changes from one line to the other. The blocks in which the value of the field remains constant are control levels.

If you want to display output before or after a control break, mark *Event on sort begin* or *Event on sort end*. Then in the tree structure an event node appears for which you can create successor nodes to display output at these events (for example, subtotals).

Data Tab

Events Tab

Use

You use the *Events* tab to define events at the beginning and end of a node (folder, complex paragraph, table, loop), at which you display additional contents.

You can display the header at the beginning of the paragraph and/or after a page break. You can display the footer before a page break and/or at the end of the paragraph. For the footer, you must specify a height to tell the composer how much space to reserve for the footer.

This disadvantage of having to specify the footer height is compensated by the fact that you can use the footer (unlike the footer of SAPscript forms) to display subtotals, since the footer is processed only at the moment the page break occurs.

The *Events* tab is especially useful for displaying table headings and totals lines.

Result

If header and footer are active, in the tree two event nodes appear within the paragraph. Under each node you can include, for example, texts for the heading or the footer line (they can even have their own line type specified in the output options of the texts).

Constraints

You cannot nest events. After you defined a header or footer for a paragraph, no other subordinate paragraph is allowed to define its own header or footer. In addition, if you use both header and footer, you must always define these areas in the same paragraph. The Form Builder checks these conditions.

Determining Table Contents

Use

You want to determine the contents of the table lines of a table.

Prerequisites

You defined line types for the table on the Table tab [Seite 74]. In addition, you created a header for the column headings on the Events tab [Seite 77]. To be able to access the contents of an internal table, you must have filled a work area using the Data tab [Seite 75].

Procedure

1. Use the context menu to create successor nodes for the *Header* event.

   For more clarity combine the nodes for the line under a folder node [Seite 57]. This is an outline node that allows you to combine related nodes.

2. The *Output options* tabs of the included nodes now have an additional *Output table* box:
   - For the first node select *New line* and use the *Line type* list box to select a line type. In addition, mark *New cell* and do not skip any cells.
   - For the next node of the header it is enough if you mark the *New cell* field.

   For formatting within a cell you must know that a cell always starts with a paragraph.

3. If you want to display the heading in two or more lines, you must mark *New line* (as you did for the first node) for the node in which you want the next line to start. You can again specify a new line type.
4. To display field contents [Seite 65] in the cells of a line include text nodes [Seite 41] as successors of your table node. To position the node contents in the cells, see steps 2. and 3.
5. To display the contents of a field, use the PC Editor to include the field name (see also: Including Fields [Seite 66]).

   You can also display the contents of several nodes in one cell.

Result

The system displays the contents in the print preview or prints them.

Flow Control

Within the tree structure, there are several options that allow you to control form processing:

- Use output conditions [Seite 80] to make the processing of nodes and their successors depend on conditions.
- Use a condition to set an alternative [Seite 81] for processing two successor nodes.
- Use a loop to process output repeatedly [Seite 82].
- Use a command to determine the page sequence dynamically [Seite 83].

The section Rules for Processing a Form [Seite 101] describes in detail how Smart Forms process the form logic.

Determining Output Conditions

Use

You use output conditions to suppress processing of individual nodes or entire subtrees in the tree structure.

Prerequisites

A node exists for which you want to determine a condition. The node must offer the *Conditions* tab. This applies to most node types [Seite 35].

Procedure

1. Select the *Conditions* tab of the node.
2. In the *Output conditions* box you can enter for each line a condition with two operands. An operand can be a field [Seite 65](in the *Comparison value* column as well) or a value (in the *Field name* column as well).

   Enter fields without enclosing ampersands ('&').
3. To select the comparison operator, choose the pushbutton between the two columns.
4. By default, the system evaluates the conditions in the lines using a logical **AND**. To use a logical **OR**, position the cursor on the desired line and choose *Insert OR* ().
5. Use the local check to check your output condition.
6. If desired, link the output conditions to output events of pages and windows. To do this, use the *and additional event* box.

Result

If the condition is true, the system processes the node. Whether it also processes the successor nodes may depend on other conditions in these nodes. If the condition is false, the system ignores this node and all successor nodes [Seite 21].

Branching Within the Form

Use

You want to process one of two nodes, alternatively (including successors).

Procedure

7. Use the context menu in the tree structure to create an alternative node. To do this, choose *Create* → *Alternative*.
8. On the *General Attributes* tab determine an unstructured condition in the *Node conditions* box. The system offers the same features as in the *Output conditions* box on the conditions [Seite 80] tab.
9. The alternative node has two direct successors: **TRUE** and **FALSE**. Insert your successor nodes there.
10. Use the local check function to check your output conditions.

Result

If the condition is true, the system processes the direct successor [Seite 21] **TRUE**, otherwise **FALSE**.

Processing Output Repeatedly

Use

You use the loop node to read data from an internal table line by line. As far as the successor nodes are concerned, it is still open how you will process this data further.

You pass an internal table containing customer data and an internal table containing customer orders to the form interface [Seite 60]. You use a loop to read the customer data and display it. Within the loop, you create a table and use the customer number to display the orders of this customer. This allows you to display the orders of all customers on one form.

Procedure

1. In the context menu choose *Create* → *Loop* to create the node in the tree structure.
2. Enter a unique name for the node and a description (for example, loop on customer table).
3. Read the data from an internal table, as described under Data Tab [Seite 75].
4. Create successor nodes in which you display the fields [Seite 65] of the read table lines.

Page Sequence and Numbering

The page sequence of a form is determined by:

- the attributes of the page node [Seite 39],
- the volume of data to be displayed and the space available in the main window (see Processing a Form [Seite 99]),
- dynamic page breaks [Seite 84].

During form processing the system maintains internal counters, which you can use to display the current page number [Seite 85].

Determining the Page Sequence

Use

You want to branch to a new page.

Prerequisites

The page to which you want to branch must exist. You can branch to a new page only as long as you are still displaying the contents of a main window.

Procedure

You use a command to stop the output of a page in the main window:

1. Choose *Create* → *Command* in the context menu to create a command node in the main window of the page.
2. On the *General Attributes* tab mark *Go to new page*. Determine the new page using the list box next to the checkbox.

Result

The output of the main window continues on the new page.

Page Numbering

Smart Forms use page counters to determine the page number. Query these system fields [Seite 68] for the page counters:

- **&SFSY-PAGE&** for the current page number
- **&SFSY-FORMPAGES&** for the total number of pages in the form
- **&SFSY-JOBPAGE&** for the total number fo pages in all forms in the print job

Use the *General Attributes* tab of the page node to determine the properties of these page counters:

- Use the *Format* box to determine whether to display the page number in Arabic or Roman numbers or in letters.
- Use the *Mode* box to determine which values you want the different page counters to accept (see below).

Values of the Page Counters

The settings in the *Mode* box have the following effects on the values of the page counters:

- *Initialize*, *Increase* and *Leave counter unchanged* effect only **&SFSY-PAGE&** accordingly. **&SFSY-FORMPAGES&** and **&SFSY-JOBPAGES&** are increased by 1 independent of this setting.
- *Page and overall page unchanged* keeps **&SFSY-PAGE&** as well as **&SFSY-FORMPAGES&** unchanged.

Text Modules

Use

You use text modules to centrally store texts in the system that you frequently use in forms. This detaches the text maintenance from the form maintenance so that you need not call the Form Builder [Seite 23] to edit individual texts.

Integration

You include text modules into forms using text nodes. The text node can either refer to the text module or copy its text. In the latter case you can change or enhance the text according to the requirements of the form.

Features

SAP Smart Forms provide a text module maintenance function that allows you to create and edit text modules. Similar to text nodes, text modules have the following characteristics:

- They can be used cross-client
- They are connected to the transport system
- They are connected to the translation tools in the system

Due to the latter characteristic the logon language uniquely assigns a text module to a language.

SAP Smart Forms include only text modules of language **A** into forms of language **A**. The language of a form is equally determined by the logon language.

Activities

1. To maintain a text module, call the SAP Smart Forms initial screen [Seite 23].
2. Check the data on the *Administration* tab. View *Language attributes* on this tab to see in which language the text module is created. On this tab, you also set the style to be used and other parameters for translation.
3. Use the PC Editor on the *Text* tab to enter your text.
4. After you have saved your text, you can include the text module into a form [Seite 44].

Smart Styles

Use

In a Smart Style you define the paragraph and character formats, which you can then assign to texts and fields in the Smart Form. You maintain a Smart Style in the Style Builder (see also Style Builder [Seite 32]).

You must assign a Smart Style to each Smart Form. You do this globally for the entire Smart Form in the form attributes. In addition, you can assign a Smart Style locally to a node, for example, a text node (see also Shared Attributes of the Node Types [Seite 37]). This assignment then applies for the entire subtree and overrules the global settings.

Features

A Smart Style contains:

- Header data containing the default values of a Smart Style
- Paragraph formats including indents and spacing, font attributes, tabs, and outline and numbering
- Character formats including effects (superscript, subscript), barcode and font attributes
- Colors and underlines for a paragraph or character format
- Preview

You can convert an existing SAPscript style into a Smart Style (see also Migrating SAPscript Forms [Seite 109]).

Activities

1. Choose transaction **SMARTSTYLES**.
   The initial Smart Styles screen appears.
2. Enter the style name.
3. To create a Smart Style choose *Create*.
4. To activate an exisiting Smart Style choose *Activate*.

Before you can use a Smart Style in a Smart Form and transport it, you must activate it. During the activation the system checks the Smart Style for errors and, if it detects any, displays an error list.

Header Data of a Smart Style

Use

In the header data you maintain the default values of a Smart Style. If you do not assign other values to the paragraph and character formats in the Smart Form, the system uses these default values.

Prerequisites

You must define these values in the header data:

- Standard paragraph
- Default tab stops
- Characters per inch/Lines per inch
- Font family and font size

Features

Standard Paragraph

You must flag one of the existing paragraphs as default paragraph. To assign this standard paragraph to a text, you use paragraph format '*'.

Default Tab Stops

The system uses these values if you do not specify any other defined tabs.

Default Font Attributes

These include the font family, the font size, bold/italic, underline, and color. You can overrule these settings in the individual paragraph or character formats.

Characters per Inch/Lines per Inch

You need these specifications if you want to specify measurements (for example, for margins or indents) in the style in the measurement units CH (character) or LN (line).

The system proposes optimized values. You should change them only if the need arises.

Creating Paragraph Formats

Use

A paragraph format contains information on indents, spacing, font settings, text color, tabs, numbering and outline. Each paragraph format must have a unique name. In the PC Editor, you assign these formats to a text or a field (see also Entering Text in the PC Editor [Seite 42]).

Indents and Spacing

You can set the following attributes:

- Alignment (default: left)
- Indent
- Spacing (default value for line spacing of the first line)
- Text flow
  - Page protection
    Use this attribute to determine whether or not to display a paragraph completely on one page. Mark it if you want to avoid that a paragraph is split up by a page break. If on the current page (only in the main window) there is not enough space left for the paragraph, the entire paragraph appears on the next page.
  - Next paragraph
    Use this attribute to link the subsequent paragraph to the current paragraph. Mark it if you want to avoid that two successive paragraphs are separated by a page break.

Font Settings

If a paragraph format does not contain any font settings, the system uses the default values from the header data.

You can assign a color from a color palette to the paragraph format. Provided that the printer driver and the printer support color printing, the relevant text is printed in color.

Tabs

For each paragraph format, you can define any number of tabs.

Numbering and Outline

You can create paragraph structures with several levels to allow outlines and numbered paragraphs.

If you want to use a paragraph format as a numbered paragraph format, enter the name of the paragraph in the field *Top outline paragraph*. You can set the following attributes:

- *List*
  Choose whether to number the list in Arabic or Roman numbers or in letters (field *Character*).

Creating Paragraph Formats

- *Left/right delimiter*
  Enter the character string that you want to be displayed before the actual number in the outline paragraph (for example, parentheses).
- *Output length*
  If you use Arabic numbers for the *List*, you can enter the length of a paragraph number.
- *Position of the numerand*
  Distance between the paragraph number and the left margin of the form window. This distance is set independently from the current setting of the left paragraph margin or the indent of the first line.
- *Character format of numerand*
  Enter a format for the paragraph number if you want it to differ from the current paragraph format (for example, if you want to display the number bold or italic or in a different size).
- *Chaining outline numbers*
  Activate or deactivate outline number chaining. If you activate number chaining, the number of the current paragraph is preceded by the numbers of the predecessor paragraphs in the hierarchy. If you deactivate number chaining, each paragraph has only its own number.

Procedure

1. In change mode of the Smart Style, select the node *Paragraph format* and choose *Create*.
2. In the *Paragraph format* field enter a two-character paragraph key.
3. Select the desired attributes on the individual tabs.
4. Choose *Activate*.

Creating Character Formats

Use

You use character formats to assign special output attributes to sections of texts or character strings within a paragraph. You must choose a different name for each character format.

For a character format, you can choose among the following attributes:

- Superscripting/subscripting the character string
- Barcode
- Font attributes (font family, font size, bold/italic, underlined, color)

You maintain barcodes in transaction *SE73*. If you want to display a field or a number chain as barcode, choose the *Standard settings* tab. In the *Name* field select the desired barcode key. After activating the Smart Style, assign the character format to the field or character string in the PC Editor.

Procedure

5. In change mode of the Smart Style select the *Character formats* node and choose *Create*.
6. In the *Character format* field enter a two-character character key.
7. Select the desired attributes on the individual tab pages.
8. Choose *Activate*.

Checking and Testing a Smart Form

Checks

To generate a Smart Form, the individual nodes as well as the interactions between the nodes must be correct. To achieve this, the system offers a large variety of checks.

Before you can use a Smart Form, you must activate it. During activation, the system checks the entire form (all individual checks executed together). If this check does not reveal any errors, the system generates the form, which means that it creates a new function module. Then you can start testing.

Each node and each tab has its own check function (*Check* pushbutton on tab). This check is always a local check for the current node or tab. The type of check depends on the type of node. The system checks the conditions which were already mentioned during the discussion of the individual nodes in the preceding sections.

In addition, the application toolbar contains a *Check* function for an overall check of the form. This check executes any existing individual checks plus some special checks that are reasonable only in the overall context. The result of this overall check is displayed in an error list (see also Field List and Error List [Seite 33]).

In the overall check, the system displays only the first error in a message, provided

the error occurs in the interface or global data. In this case it is impossible to execute any valid individual checks (resultant errors).

Data Flow Analysis

The system checks whether a particular field used in a text has a defined value at the moment the text is displayed in a window of the form. This check is called data flow analysis. The data flow analysis is part of the overall check of the form (see above).

After a modification of the output control, you can check whether any printed data still has a defined value or whether the modification created a state in which some data is still initial at the moment it is printed.

The system assumes that a data object of a node has a defined value if one of the following conditions is true:

- The data object is part of the form interface.
- The data object is defined globally and appears as output parameter in a coding node that was processed before.
- The data object is defined globally and is part of the header of a data table, for which in one paragraph repeated processing is determined.

For globally defined data objects there are three statements concerning the defined value:

- The data object has a defined value.
- The data object has no defined value.
- The data object can have a defined value.
  This happens if an alternative was processed before and the data object receives a value only in one branch of the alternative.

The system checks for all nodes whether the symbols in use have defined values. If not, the system displays a list of warnings (see also Field List and Error List [Seite 33]). These warnings do not terminate the activation or generation.

Checking Text

You can check whether all fields that appear in the text are declared in the form interface or in the global data of the form.

To do this, go to node type *Text* and choose *Check*.

Testing the Smart Form

To test a Smart Form use the function module test. After activating the Smart Form choose *Test*. The initial screen of the Function Builder (transaction SE37) appears. Choose *Single test* to go to the test mode for the generated function module. Now you can use all the methods for testing a single function module that the Function Builder provides.

Since the application program for data retrieval is not called in this case, the Smart Form is not filled with data. If you want data to appear, you must execute the relevant application program.

Runtime Errors

Each Smart Form can trigger a number of exceptions during its execution. To limit this number of exceptions, you can bundle the occurring errors in error classes and store them in an error log in the composer. For each error, you store an internal error number and (if possible) a message from table T100 (work area, message number, message text and parameters) in addition to the error class. To each error class, you assign an exception, which means that if the recognized error teminates the form processing, the system triggers exactly this exception. When an exception occurs, the caller of the Smart Form can use predefined function modules to read the error log and then decide which reaction is appropriate in the respective error context.

Possible exceptions for the generated function module are:

- FORMATTING_ERROR
- INTERNAL_ERROR
- SEND_ERROR
- USER_CANCELED

In addition, you can define your own exceptions.

If an error (or a warning) occurs, the system internally fills an error table. The application can now read this table as soon as the function module for the form is processed to the end. To read the table that contains all errors and warnings that occurred, call function module SSF_READ_ERRORS. The table has structure SSFERROR. The fields in the table are the number of the document within the job, the form name, an error number, a message class, a message type, a message number and four additional message variables. If processing is terminated, the application can dynamically display a message. The error numbers are defined in the include SSF_ERROR.

If you want to trigger your own exception in the free coding of the Smart Form, you can use one of the following methods:

Checking and Testing a Smart Form

- Use the macro **`user_exception`** <exception>, which triggers the exception <exception> and fills the above tables. After the termination additional system variables for the error (sy-msgno, sy-msgtyp, and so on) do not contain the values required for your own exception.
- If you want to access the system variables sy-msgno, sy-msgtyp, and so on for your exception directly, proceed as follows: Call function module **`SSF_MESSAGE`** and trigger your exception yourself using the **`RAISE`** statement.

The system field [Seite 68]**`&SFSY-EXCEPTION&`** contains the name of the triggered exception.

Advanced Form Development

The functions and features covered in this section either are of interest only for a particular circle of Smart Form users or their description would have gone beyond the scope of the first part of the documentation.

The features concerned are:

Complex Sections and Program Lines

This section describes node types you do not need for simple forms.

Complex Section

Use

The node type *Complex section* is absolutely sophisticated. It allows you to combine the tab pages of the node types Folder [Seite 57], Template [Seite 52], Table [Seite 73]and Loop [Seite 82] in any way.

> If you can create the desired output by using one of these node types alone, then do it. In this case, the icon in the tree structure immediately indicates the type of output
>
> 
>
> created with the node.

Procedure

5. To create the node choose *Create* → *Complex section* in the context menu in the tree structure.
6. Enter a unique name for the node and a description (for example, template with data).
7. Determine the general purpose of the complex section. To do this use the *General Attributes* tab:
   - Select *unstructured* if you need only the Events tab [Seite 77] to display header and footer areas. In this case, you can use the *Data* tab for repeated processing.
   - Select *Table* if you need the Table tab [Seite 74]. In this case, you can use the *Data* tab for repeated processing.
   - Select *Template* if you need the Template tab [Seite 52]. In this case, you **cannot** use the *Data* tab for repeated processing.
8. To read data from an internal table using the Data tab [Seite 75] mark the *Repeat processing* field in the *Data* box.

> 
>
> If, due to changing the selection, tab pages would disappear, the Smart Builder offers a warning beforehand.

Result

You can now use the selected tab pages.

Program Lines

Use

You use program lines nodes to enter free ABAP coding. If you insert program lines at the appropriate position in the tree structure, you can use it, for example, to calculate subtotals and totals in tables.

Integration

On the *General Attributes* tab use the tables *Input parameters* and *Output parameters* to pass global fields. Within the ABAP coding you can access only those global fields that you enter in these tables.

The program lines node corresponds to the *Initialization* tab of the node Global definitions [Seite 64].

Prerequisites

You must be familiar with ABAP programming.

See also: ABAP Programming (BC-ABA) [Extern]

Features

Apart from the editor functions, the following functions are integrated:

- Syntax check
- Statement pattern
- Pretty Printer

See also: Checking and Testing a Smart Form [Seite 92].

Processing a Form

Each form consists of one or more pages. The first page in the tree is the start page. With the start page processing of the form starts. Processing of a page (that is, of the respective nodes) is executed in the sequence determined in the tree.

If a page is processed (filled) and processing of the form is not finished yet (if not all texts or data are displayed), the system processes another page. The page used as next page can be determined statically during the form definition (*Next page*) or dynamically at runtime (manual page break).

A particular page is repeated until form processing is finished.

Main Window and Subwindows

Each page contains one or more windows. The system takes the formatting rules for the texts and data in a window from a Smart Style (see also Smart Styles [Seite 87]). The texts are included depending on the data in the way that is determined by the output control. Each window points to an output control.

All windows of one page must have different technical names. However, each window can occur on several pages. If a window occurs on several pages, it always points to the same output control.

You can flag one window as main window. It must be flagged as main window on all the pages on which it appears. All other windows are called subwindows.

Processing a window means processing the assigned output control. Processing of a window on one page ends as soon as the window is filled. For the main window, processing of the output control is interrupted at this point and resumed right there as soon as the main window is processed again on a subsequent page. For subwindows, processing of the output control starts again at the beginning if the subwindow is processed again on a subsequent page.

The main difference between a main window and a subwindow thus is that the contents of the main window, which is determined by the output control, can be distributed to several pages (flow text, automatical page break for texts and data). This is not possible for subwindows.

Processing of a window can be connected to conditions (only on the first page, not on the first page, and so on). If such a condition is false, processing of the window is suppressed entirely on the current page.

Output Control

The output control of a window describes how the contents of the window are composed and formatted in accordance with the given data.

The output control is defined by the nodes in the hierarchy structure (tree), whose root node is the window. The nodes of the tree have different types and different attributes, respectively (type of processing, number of successors, and so on). You can assign output options and conditions to each node.

Processing an output control means processing the successor nodes of the root node in the sequence in which they appear in the tree. Processing of a node consists of these steps:

Processing a Form

1. If a condition is assigned to the node, the system first evaluates this condition. It it is false, the system does not process the node including all its direct successors.
2. The system executes the action assigned to the node (for example, displaying a text in the window, processing program lines, and so on).
3. The system processes the direct successors (children) of the node in the tree. The sequence in which they are processed (sequentially from top to bottom, at certain events, alternatively, repeatedly) depends on the type of node.
4. If output options are assigned to a node, these output options apply for any output in the window which is described by the node and its successors.

You can use these conditions when processing a node:

- Simple IF condition (using AND and OR)
- Complex IF condition (using program coding)
- Event queries (for example, first page, page break)
- Any combinations

You can determine the following output options:

- Assigning a Smart Style
- Page protection (any output triggered by the node and its successors is displayed on one page)
- Boxes and shading (any output triggered by the node and its successors is framed in a box or shaded according to the specification)
- Accessing lines and cells of a template or table

Rules for Processing a Form

The overview below lists the exact rules for processing a form:

1. Processing starts with the start page (the first page in the tree).
2. Processing of a page is executed in the sequence of the assigned windows in the tree.
3. Processing a window means processing the output control of the window (see also Output Control [Seite 79]).
4. Processing of the main window on a page is finished:
   i. If the main window is filled completely on the page and the output control has not yet been processed to the end *or*
   ii. If the output control has been processed to the end *or*
   iii. If a manual page break (see also Page Sequence and Numbering [Seite 83]) is reached.
5. Processing of a subwindow is finished:
   i. If the subwindow is completely filled on the page *or*
   ii. If the output control has been processed to the end.
6. After all windows of a page are processed, processing of the page is finished.
7. A page break to a new page occurs as soon as the processing of a page is finished and
   i. The output control of the main window has not yet been completely processed *or*
   ii. A manual page break was executed in the output control of the main window *or*
   iii. The page does not contain a main window.
8. The new page is the static next page. If a new page is called by a manual page break with page specification within the main window (dynamic next page), the system processes this page.
9. Processing of the form ends if the processing of a page is finished and
   i. On this page the output control of the main window has been finished and no manual page break occurred at the end *or*
   ii. On this page the output control of the main window has not been finished but there is no next page (neither static nor dynamic) *or*
   iii. This page does not contain a main window and no static next page exists.

The manual page break is executed only in the main window, which means that you can specify it only in the output control of the main window.

For pages without a main window the system goes to the static next page after processing all windows. If there is no next page, processing stops.

Rules for Processing a Form

For particular applications it is possible to repeat the processing described above (one main window and several subwindows), so that actually a form with several main windows is created.

Delivery and Translation

Transport Object

SAP Smart Forms and Smart Styles are stand-alone development objects. This means that for every Smart Form and for every Smart Style a TADIR entry is created. The corresponding transport objects are called

- SSFO for Smart Forms (R3TR SSFO name)
- SSST for Smart Styles (R3TR SSST name)

For creating and maintaining TADIR entries and transport objects, use the Transport Organizer as you would for any other development object. When creating a Smart Form or a Smart Style, you must assign it to a development class. If the development class is not temporary, the Transport Organizer delivers the Smart Form or Smart Style to other systems in the usual way.

Client-Independency

SAP Smart Forms and Smart Styles are client-independent objects.

Translating Texts

SAP Smart Forms and Smart Styles take part in the usual translation process for development objects. The transport objects SSFO and SSST are known to the translation tools and appear in the relevant worklists. For a Smart Form, you can translate the following texts:

- All long texts of nodes
- The contents of all text nodes.

To translate these texts, use the general procedures for translating long texts.

Master Language and Language Vector

When you create a Smart Form, the master language is your logon language. You create the form and any language-specific texts (descriptions, texts) in the master language.

The language vector indicates the languages into which the form must be translated.

Graphic Administration

Use

You use the graphic administration to import, administer, and transport graphics, and to display them (for example, company logos, material images) in a print preview.

You can either incorporate graphics statically into a form (for example, the company logo) or include them dynamically using an appropriate field (for example, material graphic showing the displayed material). For more information see Printing Graphics [Seite 50].

To copy any form in the Form Painter, you can scan the desired form as graphic and include it into the Form Painter as background graphic (see also Printing Graphics [Seite 50]).

Printing graphics redurces the printer performance, since it requires large amounts of data to be sent over the net. For this reason, always make sure that a graphic is really necessary, especially for performance-critical documents (see also Importing Graphics [Seite 105]).

Features

You can find any imported graphics in the navigation tree. Before Release 4.6 graphics were stored as standard texts. This is obsolete. As of Release 4.6 graphics are stored on a document server. In addition to *.TIF graphics you can now also import *.BMP files and use them in forms.

To display graphics in their original size, use the preview. To transport graphics into other systems, use the graphic administration as well.

Activities

To start the graphic administration, choose transaction **SE78**.

Before you can start to execute any functions, select the desired node in the navigation tree (for example: Stored on document server, Graphics, BMAP grid screen) and specify the name and the attributes of the corresponding graphic. To search for graphics, use F4 help on the *Name* field.

Importing Graphics

Use

You can import graphics from your PC.

Printer Attributes

Resident in printer memory

When a graphic appears for the first time, it is defined as macro during the print process and stored in the printer memory. If the graphic appears again during the same print request, it can be taken from the printer memory. This considerably reduces the size of the print files created by the system.

However, the printer memory capacity is limited. Only a small number of graphics (recommended: one to three) should be buffered in this way in the printer memory during each print request. If the printer memory is full, it can no longer buffer any data so that subsequent graphics are then missing in the printout.

Use this attribute only for frequently repeated graphics within one print request (for example, for a company logo).

Automatically reserve height

Graphics do not influence the page break, which means that the space consumed by the graphic on the print page is not considered during formatting. If you set this attribute for the import, the system automatically reserves the actual height of the graphic for the page break. This means that any subsequent text starts below the graphic. Graphics for which this attribute is set, can therefore trigger a page break.

Prerequisites

Supported graphic formats are *.bmp and *.tif files. As TIFF format you must use a Baseline 6.0 (for example, without compression). If you do not have the required format, you can use standard graphic programs to convert the graphics into this format.

Note that many printer drivers or printers support only the following resolutions: 75 dpi, 100 dpi, 150 dpi, 200 dpi, 300 dpi, 600 dpi.

Procedure

1. Choose transaction **SE78**.
   The dialog window *Graphic Administration* appears.
2. Choose *Graphic* → *Import*.
3. Enter the file path on your PC (you can use F4 help).
4. Enter the graphic name for the system and select the graphic type (*grid screen black/white* or *grid screen color*).

5. If required, set one of the printer attributes *Resident in printer memory* or *Automatically reserve height*.
6. Choose *Continue*.

Result

The graphic is loaded into the system and stored on the Business Document Server (BDS). You can now include it in the Form Painter (see also Printing Graphics [Seite 50]).

Graphic Information and Preview

Use

You can display information on graphics and change some attributes, and you can test graphics in theis original size in a preview.

Procedure

Graphic information

1. Choose transaction **SE78**.
   The dialog window *Graphic administration* appears.
2. In the tree choose *Store on document server, Graphics, BMAP grid screen*.
3. Enter the name of the graphic.
4. Choose *Screen information*.
   The system displays the technical attributes and the administrative data of the graphic.
5. If you want to change the attributes *Resident in printer memory* and *Automatically reserve height*, choose *Change attributes*. Then save your changes.

Preview

1. Choose transaction **SE78**.
   The dialog window *Graphic administration* appears.
2. In the tree choose *Store on document server, Graphics, BMAP grid screen*.
3. Enter the name of the graphic.
4. Choose *Preview*.
   The system displays the graphic in its original size.

Transporting Graphics

Use

You can use the graphic administration to directly transport an imported graphic into another system.

Procedure

1. Choose transaction **SE78**.
   The dialog window *Graphic Administration* appears.
2. Select the desired graphic in the tree.
3. Choose *Graphic* → *Transport*.
   The dialog window asking for the Workbench request appears.
4. Select an existing request or create a new one (*Create request*).
5. Choose *Continue*.

Result

The graphic is included into a Workbench request. After it is released in the Transport Organizer, it will be transported into the requested system.

Migrating SAPscript Forms

Use

SAP delivers Smart Forms for important business processes. If for your needs no such standard forms exist or if you have your own SAPscript forms, you can use two tools to migrate them.

Make use of this migration only if you want to make extensive changes to existing SAPscript forms. Especially the necessary modification of the data retrieval program (ABAP program) can take several days for complex applications, since all the techniques you use (for example, EXTRACT and LOOP) must be adapted.

Individual Migration

You can migrate a SAPscript form into a Smart Form and convert a SAPscript style into a Smart Style.

When converting a SAPscript style into a Smart Style, the system converts all paragraph and character formats with all their properties and attributes without any changes. Thus you can use the converted Smart Style without making any adaptations.

When migrating a SAPscript form into a Smart Form, the system executes the following steps:

- It copies the language attributes and the output options.
- It migrates the layout information including pages, windows, and their attributes and positions on the page.
- It copies the texts in the form.
- It displays the fields (SAPscript notation: program symbols) in the texts.
- It converts the SAPscript commands (such as NEW-PAGE or IF…ENDIF) to comment lines and displays them in the texts.

After the migration, you have a template that you can enhance and modify according to your needs. Adapt the form logic by specifying the commands which the system converted to comment lines.

The system does not migrate the print program (data retrieval) or the form logic contained in the print program. You must make the required changes to the respective print program yourself.

Mass Migration

You can migrate any number of SAPscript forms in one go.

Procedure

Migrating a SAPscript form

1. Go to the SAP Smart Forms initial screen (transaction **SMARTFORMS**).

2. In the *Form* field enter the name of the Smart Form you want to create.
3. Choose *Utilities* → *Migrate SAPscript form*.
   The dialog window *Migrate SAPscript Form* appears.
4. Enter the name and the language of the source form (SAPscript).
5. Choose *Enter*.
   This takes you to the change mode of the SAP Form Builder.

   If the selected SAPscript form does not exist in the selected language, a dialog window appears on which you can select one of the existing languages.
6. Now change the design of the form and of the form logic. To activate the Smart Form choose *Activate*.

Converting a SAPscript style

1. Go to the Smart Styles inital screen (transaction **SMARTSTYLES**).
2. In the *Style name* field enter the name of the Smart Style you want to create.
3. Choose *Smart Styles* → *Convert SAPscript style*.
4. Enter the name of the SAPscript style you want to convert.
5. Choose *Enter*.
   A list of the converted styles appears.
6. Choose *Back*. You can now change the Smart Style (*Change*).To activate the Smart Style choose *Activate*.

Mass Migration of SAPscript Forms

1. In Reporting select the program SF_MIGRATE and execute it.
2. Select the names and the language of the SAPscript forms and choose *Execute*. The system creates the Smart Forms under the names of the SAPscript forms plus the extension _SF. It displays a list of the migrated forms.
3. To change and adapt a form, go to transaction **SMARTFORMS**. Then activate the changed Smart Form.

Result

You created one or more Smart Forms based on the respective SAPscript form(s) and a Smart Style based on the respective SAPscript style.