BC Style and Form Maintenance

HELP.BCSRVSCRFORM

Release 4.6C

Copyright

Icons

Icon	Meaning
⚠	Caution
💬	Example
💡	Note
🧭	Recommendation
SYN	Syntax

Inhalt

BC Style and Form Maintenance

Forms [Seite 58]

Styles [Seite 42]

SAPscript Control Commands [Seite 104]

SAPscript Symbols [Seite 142]

Forms: Components and Techniques

Concepts

Components

Techniques

Modifying SAP Forms

If you want to modify SAP forms, set up your development environment as follows:

1. Make sure that no SAP-standard forms are stored as client-specific copies in your development client.

   Such forms should be held only in client 000, the SAP development and installation client. If you access an SAP-standard form from another client, then the central copy in client 000 is used.

   If you need to remove SAP-standard objects from your development client, see Notes 10388 and 3355 in the SAP Online Service System (OSS). These notes explain the procedure for saving modified forms and then deleting all forms.

2. To modify SAP standard forms,

– Copy the forms you need from client 000 to your development client.

– Rename the forms using a name from the customer name reserve (names starting with Y or Z).

– Copy the forms to one of your own Y or Z development classes.

   Renaming the SAP standard object makes it possible to manage and transport your changes with the SAP workbench organizer. The organizer is not activated for SAP-standard objects that are modified in clients other than 000.

3. To put your modifications into effect, you must also modify the ABAP print program used to print documents that use this form. You should rename such print programs and store them in your own Y or Z development classes.

   You can use the SAP Customizing System to replace the SAP print program with your modified print program in the affected applications.

Forms: Concepts

Forms are used to control the page layout and also the text formatting in your documents. Before formatting a document for output to the screen or to a printer, you must assign a form to it. If you do not specify a form for a document, then the SYSTEM form is assigned to the document by default.

Application-specific forms are used in SAP applications to specify the page layout for such special documents as invoice or checks. These forms specify the structure of the information on the page(s) of such a document. They define, for example, the address header, item lines, the footer, and so on.

There are two ways to format texts with forms:

- In the standard SAPscript text processing (*Tools* → *Word processing* → *Standard text*), you can select a form for a document. You can then type text into the main window of the form and output the document in the format defined in the form.

  For example, you can select a form for a letter. You can then type the body text of the letter in the main window. When you print the letter, the default text elements in the other windows of the form (heading, footer, and so on) are printed with the body text.

- A document can be generated by a print program in one of the SAP applications. The print program uses a form to generate the document. Most correspondence and document generation in the SAP System are handled by way of print programs.

  A print program selects the text elements that are to be printed in the windows of a form. It may also collect information from the user or ask the user to input text directly, as in some correspondence functions. The print program may also provide data for variables defined in the form.

  Finally, the print program uses the form to format the document for display or printing.

Client and Language Versioning: Concepts

Forms and styles are client-specific. That is, a form or style other than the SAP standard in client 000 is available only in the client in which it was created.

Forms and styles are also language-specific. That is, the definitions and texts in a form or style are defined for a particular language. Forms and styles can be translated using the standard SAP translation tools.

Client 000 Defaulting: SAPscript accords forms and styles in client 000 a special status.

If a form or style that is used in a document is not available in the client in which the document is being printed, then SAPscript checks for the form or style in client 000. If it is found there, then the client 000 version is used to print the document.

SAP standard forms and styles are always held in client 000. You can take advantage of the client 000 defaulting as well by storing your Yxxx and Zxxx forms and styles there. That way, if a local version of a form or style is not present in a client, the client 000 version is used instead.

See Transporting, Copying, and Comparing Styles and Forms [Seite 96] for information on tools for managing forms and styles in multiple clients.

Language rules: SAPscript uses the following rules to manage versions of forms and styles in different languages:

- The language in which a form or style is created is its “original language.” You can translate a form or style into other languages using SAP’s translation tools.
- If a form or style is needed only in its original language and need not be translated, then you can indicate this in the language attributes in the header data. The form or style then does not appear in work lists in the translation tools.
- In versions other than the original language version, changes to a form or style are limited only to translation of texts. No changes to definitions and attributes are permitted.

Header Data

You can find header data in both form and style maintenance. In style maintenance, it is used primarily to present important information – information designed to make it easier for the end user to select a style. The header data in form maintenance, on the other hand, is used for information and control purposes. For this reason, the header data of a form will be described in more detail.

> Below, the header data is described as it appears in the alphanumeric Form Painter. For a description of header data in the graphical Form Painter – which is a bit different – see Form Components (Graphical Form Painter) [Seite 85].

Like the header data of a style, the header data of a form comprises two parts: the data set by the system and the data you are expected to enter. The latter is dealt with separately.

- `Device-independent entries.`
  - *Description*

    A short explanatory description of the form (also applies to the style), designed to make selection easier for the end user.
  - *Form class*

    You can assign a form to a class to help you organize and search for forms. The default set of classes is the set of program classes in your system.
  - *Start page*

    Tells the print program which page format in a form to use first for printing.
  - *Default paragraph*

    Paragraph set to `*` in standard text maintenance.
  - *Tab stop*

    A grid set at specified intervals in all windows defined in the form. However, you should note that the tab stops are only valid in paragraphs where you have not defined your own tabs.
  - *Language*, *Original language*, *Translation applic*

    Use these fields to record the master language and language of the current version of a form. Marking *Translation applic* makes the form accessible for translation from the SAP System’s translation tools (transaction SE63).
- `Device-dependent entries.`

  You can only enter values here that are supported in the R/3 printer definition. If you make other entries, this leads to errors in the check routine.
  - *Page format*

    Determined from the spool administration table with transaction SPAD. Make sure there is a printer assignment – there must be an additional spool format for the printer with the same page format.
  - *Orientation*

Depends on the page format selected. This can also be determined from the spool administration table. Please note that the formats landscape and portrait are not supported by all printers.

- *Lines per inch* (LPI)

  Basis for converting the unit of measurement LN in style and form maintenance. The value 6.00 is set by the system, as this value is supported by all printers.

- *Characters per inch* (CPI)

  Basis for converting the unit of measurement CH in style and form maintenance. The value 10.00 is set by the system, as this value is supported by all printers.

- Font attributes

  With these fields, you can set the default font for a form. The default font applies if other objects do not specify a font. SAPscript suggests a default font, which you can change.

Paragraph Formats and Attributes

In SAPscript, paragraphs are formatted using formats and their corresponding attributes. Text processing is simplified by the use of different paragraph attribute groups:

- Standard
- Font
- Tabs
- Outline

`There are naming conventions for paragraph tags:`

- The paragraph tag can have one or two characters.
- The first character in the paragraph tag must be a letter, the second a letter, number, or blank; special characters are not valid.
- The paragraph format must be identified in the *Description* field.

Standard Paragraph Attributes [Seite 17]

Font Attributes for Paragraphs [Seite 19]

Tabs in Paragraph Formats [Seite 20]

Paragraph and Heading Numbering [Seite 21]

Standard Paragraph Attributes

In the Standard attribute group, you find the general attributes that can be defined in paragraph formats:

- *Description*

  Precise explanation of your paragraph tag, so that the user can immediately identify it.

- *Left or right margin*

  Amount of space between the paragraph and the left or right border of the form window.

- *Indent first line*

  Indent of the first line of a paragraph. If the value is positive, it is indented to the right, if it is negative, it is indented to the left.

  If you specify a negative value, then you must place the minus sign after the number: `1-`.

- *Space before and space after*

  Space before and space after control the amount of space between paragraphs. The actual space between paragraphs results from the space after the preceding paragraph and the space before the following paragraph.

- *Alignment*

  Alignment of a paragraph.

Left-aligned	LEFT
Right-aligned	RIGHT
Centered	CENTER
Justified	BLOCK

- *Line spacing*

  Spacing between the lines. The default value is 1 line; the LPI value (lines per inch) in the header data is used to calculate the line spacing.

- *No blank lines*

  Suppression of blank lines. You can control whether the blank lines of a paragraph should be suppressed in the printout or not:

No entry	blank lines not suppressed
X	blank lines suppressed

- *Page protection*

  Cohesion of a paragraph. It is possible to determine whether or not a paragraph can be divided by a page break.

No entry	no page protection (default)
X	all lines of the paragraph are on one page

- *Next paragraph same page*

Cohesion of two adjacent paragraphs. Here you can define whether the subsequent paragraph should begin on the same page (that is, at least the first line of the subsequent paragraph must be on the same page).

No entry	subsequent paragraph is output on the same page or the next page, depending on the amount of space (default)
X	subsequent paragraph begins on the same page

Font Attributes for Paragraphs

You can specify font attributes for paragraph formats. They control the font used in the text. You can specify these attributes both for the default font in the header and for particular paragraph formats:

- *Font family*

  Enter a font supported in the SAPscript font maintenance.

- *Font size*

  Enter the size of a character font. It is measured in 1/10 point.

- *Bold/Italic*

  Specify whether to use bold-face printing or italics.

- *Underlined*

  Mark this attribute to underline entire blocks of text.

  When defining a paragraph format, use *More* to specify these underline attributes:

  – Spacing between the base line and the underline

  – Thickness

  – Intensity

    Intensity is expressed in percent: 0 % is a black underline; 100 % is no underline.

  If you defined default underlining in the header, then the fields for underline attributes are already displayed on the screen.

  The following selection criteria apply to the font attributes *bold*, *italics*, and *underlined*:

Off	attribute is not set
Retain	inherited
On	attribute is set

The combination of font family, font size, bold type attribute and italics attribute is referred to as a system font or SAP font. To use the SAPscript font maintenance, choose *Tools* → *Word processing* → *Font*.

Tabs in Paragraph Formats

You can define as many tab positions as you require for each paragraph format. The text can be aligned in different ways:

- Left-aligned with *LEFT*
- Right-aligned with *RIGHT*
- Centered with *CENTER*
- At the sign with *SIGN*
- At the comma or decimal point with *DECIMAL*

You can control the tab feed in a paragraph with tab positions. The tab stops you define in the paragraph format replace the tab spacing you defined in the header data of the form. However, this depends on the extent to which you have defined tab stops in the paragraph format. If there are fewer tabs in the paragraph formats than in the header data, the tab stops of the header data are used for the rest of the line. The tab stops are represented as, *,* in the text editor.

You can use different units of measurement to define a tab position:

• CH	Characters
• CM	Centimeters
• MM	Millimeters
• PT	Points
• TW	Twips (1/20 point)

The unit of measurement CH is converted to an absolute unit of measurement using the CPI value (characters per inch) from the header data of the form.

Paragraph and Heading Numbering

The paragraph numbering and marking attributes are used to structure texts into chapters, subchapters, and sections. Numbering is carried out automatically by SAPscript.

You can create an outline with the entry options available:

- *Outline*

  Enter the name of the highest-level paragraph in an outline hierarchy here. The outline hierarchy is created by assigning this paragraph to all outline paragraphs.

- *Outline level*

  Enter the level in the outline hierarchy. The outline levels of the paragraphs are numbered upwards from 1; the highest outline level therefore has the outline number 1.

- *Number margin*

  Specify the space between numbering and window border. Note that your numbering may extend into the text area of the paragraph if the difference between the left margin and the number margin is not great enough to hold the numbering characters.

- *Left/right delimiter*

  Specify the character that precedes or follows the numbering.

- *Number chaining*

  Specify whether you want the paragraph numbering of the paragraph to be preceded by the numbering of all higher paragraphs in the hierarchy.

...with number chaining	**...without number chaining**
3.	3.
3.1	1.
3.2	2.
3.2.1	1.

- *Character string*

  Specify the numbering format. The numbering can be assigned a different font or character format to the rest of the paragraph.

- *Numbering type*

ARABIC	Arabic numerals: 1, 2, 3.
CHAR	Fixed character: letter or numeral, entered in the field
LETTER	Letters: A-Z
ROMAN	Roman numerals: I, II, III, IV

Depending upon the numbering type that you select, the following attributes may also apply:

– *Fixed character*

  Define the fixed character to be used for numbering. You should only make an entry in the field *Fixed character* if you have specified CHAR as the numbering type. Fixed characters include **+ -** and **o**.

– *Output length*

  Enter the number of characters for Arabic numerals.

– *Upper case*

  Specify for letters or Roman numerals.

Character Formats and Attributes

Character formats, as opposed to paragraph attributes, allow you to format entire blocks of text within a paragraph.

Character attribute groups can be:

- Standard
- Font

When you define character formats, observe the following naming conventions:

- The character format can have one or two characters.
- The first character must be a letter, the second a letter, number, or blank; special characters are not valid.

Enter a simple explanation in the field *Description*. It is intended to help the user make a selection.

Standard Attributes for Character Formats [Seite 24]

Font Attributes for Character Formats [Seite 25]

Standard Attributes for Character Formats

- *Marker*

  Links a search key to the selected character string when the end user uses this character format. Examples include glossary, hypertext, and data element links. Here, selected character strings are assigned the appropriate key.

- *Bar code*

  Bar code that is required for certain variables and is known to the printer, for example EAN8. The character string is printed as a bar code if the character string concerned is selected.

  Bar code names, such as EAN8, refer to system bar codes. These are defined in the SAPscript font maintenance (*Tools* → *Word processing* → *Font)*.

- *Protected*

  The character string is not split by a line break, but printed together on the next line.

- *Hidden*

  The character string is not printed. The text is only visible in the text editor.

- *Superscript/subscript*

  The character string is printed half a line higher or lower.

The following options are available for defining these attribute types:

Off	Attribute is not set
Retain	Attribute is inherited
On	Attribute is set

Font Attributes for Character Formats

Font attributes can be specified for character formats as well as for paragraph formats. You have the same options as for defining font attributes for paragraph formats. For more information, see Font Attributes for Paragraphs [Seite 19].

Windows

If you are using the graphical Form Painter, read about the Administration Screen [Seite 86] as well.

Windows are defined in form maintenance. They represent areas that are positioned on pages – as page windows – and in which at a later time text is printed. You must define at least one window for each form. Otherwise, SAPscript cannot format the text.

You can assign window names and window types. However, note that you can define only one main window per form.

Use one of these window types:

- MAIN

  Main window in which continuous text is printed. This is the window used by dialog users of a print program and form. For example, the body text of a letter would be entered in MAIN.

  The text in the main window can extend over several pages. If the text fills one page, output continues in the window of the next and subsequent pages, as long as MAIN has been defined for these pages.

  For more information, see Main Window [Seite 32].

- VAR

  Window with variable contents. The text can vary on each page in which the window is positioned. Variable windows are formatted for each page.

  To every window you can assign text, which is printed in the corresponding window when the form is formatted. To assign text, use text elements, which are stored with the form.

  To create and maintain text elements with the SAPscript Editor, choose *Text elements*. For more information, see Text Elements [Seite 31].

  Should the text selected for the window exceed the window size, then the text is cut off.

- CONST

  Window with constant contents that is formatted only once.

  Currently, CONST windows are processed in the same way as VAR windows. You should only use windows of type VAR.

Default Paragraph

For a particular window, you can override the default paragraph format that is set in the form header. Enter the default format that should apply in the window in the *Default paragraph* field in the window definition screen.

Pages

You must define at least one page for every form. And you must designate a “first” page in the form header. Otherwise text formatting is not possible. In addition, you should inform the system which page is to be used after reaching the end of the first page. If you do not specify a next page, the output of your text ends at the end of the current page.

To define a page, give it a name and specify attributes for it:

- Name of the next page
- Page counter mode

INC	Increases the counter by 1
HOLD	Counter remains unchanged
START	Sets the counter to 1

You can display the contents of the page counter with the system symbol&PAGE&.

- Numbering type of the page counter

ARABIC	Arabic numerals
LETTER	Letters
ROMAN	Roman numerals

Although CHAR is displayed as an entry option for the numbering type of the page counter, internally CHAR is converted to ARABIC.

  - Output length for page numbering with numerals
  - Upper or lower case for numbering with Roman numerals or letters

- Resource name

With *Resource name*, you specify that the paper for this page should be taken from a particular paper tray at the printer.

In *Resource name*, enter the print control that has been defined for switching to the paper tray you want to use. In printer types pre-defined by SAP, these print controls are generally as follows:

TRY01	Select first paper tray
TRY02	Select second paper tray (if available at the printer)
TRY03	Select third paper tray (if available at the printer)
TRYEN	Print envelopes (if available at the printer)
TRYMN	Switch the printer to manual paper feed (if available at the printer). The printer pauses until you feed a sheet of paper into it.

TRYME	Switch the printer to manual envelope feed (if available at the printer). The printer pauses until you feed an envelope into it.

You can use all tray selection print controls except TRY03 with suitably equipped printers that are defined with the following SAP device types: HPLJSTND, HPLJ_II, HPLJIIID, HPLJ4, LX4039, and SNI20XX8.

You can use TRY01, TRY02, TRY03, and TRYMN on suitably equipped printers that are defined with these device types: KYOF1000, KYOF1200, KYOFS1500.

See the spool system (transaction SPAD) to check on how your printers are defined.

- Print mode

  With *Print mode*, you can specify single- or double-sided printing for a page. You can choose from the following values:

`' '` Currently active printing mode continues unchanged.

`S` The page is printed in **simplex** mode. That is, the printer should print on only one side of the paper. If another mode was previously active, then the printer is switched to simplex mode with the start of the page.

`D` The page is printed on the first side of a sheet in **duplex** mode. If another mode was previously active, then the printer is switched to duplex mode with the start of the pageand continues in this mode.

`T` The page is printed on the first side of a sheet in **tumble duplex** mode. That is, the printer prints on both sides. The page on the second side is always inverted, so that the footer of the second page is printed opposite the header of the first page.

If another mode was previously active, then the printer is switched to tumble duplex mode with the start of the page and continues printing in this mode.

Print modes are currently supported for printers that use the PCL-5 language. These are printers that are defined with the following SAP device types: HPLJ_II, HPLJIIID, HPLJ4, LX4039, SNI20XX8.

See the spool system (transaction SPAD) to check on how your printers are defined.

The print controls for these functions are SPMSI (begin simplex printing); SPMDU (begin duplex printing); SPMTU (begin tumble duplex printing); SPMFS (print on first side of sheet in duplex or tumble duplex mode); and SPMBS (print on second side of sheet in duplex or tumble duplex mode).

Defining Follow-On Pages in Duplex Print Modes

You switch to duplex or tumble duplex mode with a form page for which one of these modes is specified. To continue printing in the current mode, for follow-on pages you must define another page in which the *Print mode* field is empty. Otherwise, the following pages after the mode switch will continue to be printed only on the front sides of new sheets of paper.

The reason: When SAPscript sends a page with *Print mode* D or T to the printer, it not only sets the print mode accordingly. To ensure that the first page in the new mode is correctly output, SAPscript also instructs the printer to output the page on the front side of a sheet. If SAPscript sends a sequence of D or T pages to the printer, the output is printed only on the front side of each sheet.

> You define a first page named FIRST for a form to be printed in duplex mode. You therefore set the *Print mode* in FIRST to D.
>
> To make the duplex printing work correctly, you must define a second page FOLLOWER in which *Print mode* is left empty. In the form definition, you specify FOLLOWER as the follow-on page for FIRST and for itself as well.
>
> Your text is then printed in duplex mode. FIRST switches the printer to duplex mode and forces printing of the first page on the front side of a new sheet. FOLLOWER accepts the duplex mode and sends no further mode print controls to the printer. The printer therefore alternately prints FOLLOWER pages on the fronts and backs of sheets.

Testing Tray Selection and Print Mode Selection

SAP provides predefined SAPscript documents with which you can test whether tray selection and print mode selection are working properly on your printers.

For tray selection, print the SAPscript document SAPSCRIPT-TRAYTEST, *ID* ST, *Language* D or E.

For print mode selection, print the SAPscript document SAPSCRIPT-PRINTMODETEST, *ID* ST, *Language* D or E.

Page Windows

The page window component exists only for the alphanumeric Form Painter. The graphical Form Painter uses a separate design window [Seite 230] for the page layout.

When you define page windows, the window names are linked to page names. You must specify the position and size of the window on the assigned page.

Define the position of a window by specifying the left margin and upper margin and the size of a window by specifying its width and height.

Left margin	Space between the window and left margin of the page
Upper margin	Space between the window and upper margin of the page
Window width	Width of the window depending on the page format selected
Window height	Height of the window depending on the page format selected

Note that the width of the main window must be the same on all pages. All other window types can have different sizes and positions on different pages.

To print multiple columns, define several main windows on a page. As text entry is continuous, once the first main window has been filled, output continues in the second main window.

Text Elements [Seite 31]

Main Window [Seite 32]

Text Elements

You can define text elements (window texts) for each window. On the *Form: Request* screen, choose *Edit* → *Text elements*.

The print program accesses text elements by name, formats them and prints them in the respective window. That is, the program can decide which text elements should be printed in a particular window. It calls these text elements by name to print them. You must therefore change text element names in the print program if you make any changes to text element names in a form.

In the layout of a text element, you can use only the paragraph and character formats defined in the form.

Example of a text element in an order confirmation:

```
/E ITEM_LINE
IL &VBDPA-POSNR&,,&VBDPA-MATNR&,,&VBDPA-ARKTX&
/  &'Customerarticlenumber 'VBDPA-IDNKD' '&&'Position
/  'VBDPA-POSEX&
```

This example shows a section of a main window of a form, with an item line of an order confirmation. The `/E` in the tag column is used to identify the text as a text element, ITEM_LINE is the name of the text element.

Default Text Element

At the start of a window, you can define a text element without the **/E** command in the paragraph format column. This text is always printed at the start of the window. It is not necessary to insert this text explicitly via a print program.

Main Window

In the definition of page windows you can define `several` main windows per page. However, you must first specify an area for the main windows.

This function allows you to output text either in columns as in newspapers, or next to and below each other as in label printing.

You can specify the size and position of this area under the group heading *Area*:

Left margin	Amount of space from the left border of the page
Right margin	Amount of space from the right border of the page
Area width	Width of the area (required entry)
Area height	Height of the area (required entry)

To position several main windows in this area, you must assign values to the variables under the group headings *Horizontal* and *Vertical*.

Under the group heading *Horizontal*, enter

Spacing	Horizontal spacing of the main windows
Number	Number of main windows (horizontal)

Under the group heading *Vertical*, enter

Spacing	Vertical spacing of the main windows
Number	Number of main windows (vertical)

The units of measurement which can be used in the fields *Left margin, Upper margin, Area width, Area height*, and *Spacing* for both horizontal and vertical measurements are:

CH	Characters
CM	Centimeters
LN	Lines
MM	Millimeters
PT	Points
TW	Twips (1/20 point)

For the vertical area, CH is calculated using the CPI value in the header data of the form. LN is converted on the basis of the LPI value in the header data of the form.

The field *Start position* under the group heading *Positions* is a counter for the main windows defined in a page window. You can use this counter to number the columns. Always enter a value greater than or equal to 1 for the counter *Start position*.

If you change the counter of the form main window, the form main window can no longer be distinguished from the main windows of the page window.

Displaying Versions of Forms

To display the versions of a form, choose *Utilities → Versions*.

The version list includes the following information:

- Version and status:
  - New
    No active version, not available for use as yet.
  - Active
    The current form is the active version, in effect in the System.
  - Revised
    The current form is being changed. There are active and edit versions of the form.
  - Translated
    The current form must be translated.
- Current language key
- Original language of the form.
- Description of the form.

To display detailed information on a version of the form, put the cursor on the version and choose *Form info*.

Including Graphics

You can include graphics – such as diagrams or charts – in your SAPscript documents. Typically, such graphics are included in documents by way of forms.

Graphics are uploaded either in “Baseline TIFF 6.0” format (file extension.tif on PC files) or as printer macros. A printer macro in this case is the sequence of printer instructions needed to print out a diagram.

Graphics and printer macros are uploaded with program RSTXLDMC into individual standard text documents. At upload, the graphics or printer macros are converted to the format required by the target printer, either PostScript, PCL-5 for newer Hewlett-Packard and compatible printers, or PRESCRIBE for Kyocera printers. The resulting SAPscript document can be printed only on the target printer type. Online display is not possible.

In a form, you might include graphics such as a company logo in the header window. You can accommodate printing on different types of printers with separate includes for each format in which you have uploaded graphics. If you define the includes as text elements, then your print program can select the appropriate include depending upon the device type of the printer that the user selects. You can determine the device type by looking up the printer name in table TSP03 (field PADEST) and evaluating the device type (field PATYPE).

For more information, see the report documentation for RSTXLDMC.

Using graphics in forms can greatly increase the size of print requests and therefore seriously affect the performance of your printers. Graphics are not recommended for printing time-critical documents.

Using Boxes, Lines, and Shading

SAPscript provides these text commands for using boxes or frames, lines, and shading in documents:

- The BOX command for drawing a box or a horizontal or vertical line.
- The POSITION command for specifying the starting point (the upper left corner) of a box or line.
- The SIZE command for specifying the width and height of a box.

Supported Printers [Seite 36]

Boxes, Lines, Shading: BOX, POSITION, SIZE [Seite 135]

Pre-Setting BOX Position Arguments [Seite 37]

Using the Commands in Texts and Forms [Seite 39]

Tips and Guidelines [Seite 40]

Supported Printers

You can print boxes, lines, and shading on any page printer that uses one of these SAPscript printer drivers in its device-type definition:

- HPL2 Hewlett-Packard LaserJet family and compatibles
- POST PostScript-compatible printers
- PRES Kyocera printers (Prescribe printer language).

Pre-Setting BOX Position Arguments

You can use the POSITION and SIZE commands to preset some arguments in the BOX command. POSITION presets the start point (upper left corner) of a box or line. SIZE specifies the width and height of a box.

You can use POSITION and SIZE to preset arguments, but you can also set the start point and size arguments of a box or line directly in the BOX command.

By default, if no positioning is specified, the upper left corner of a box or halftone or the top of a line is aligned with current SAPscript window. That is, the upper left corner of the box, halftone, or line starts at the upper left corner of the current window in the active form. By default, the height and width of a box are set to the height and width of the current window.

Use POSITION and SIZE to preset the arguments in a BOX command in the following situations:

- The BOX command exceeds the 132-character (1 line in SAPscript) length limitation if you specify all arguments directly in the command. You may exceed this length limit if, for example, you use symbols in a command.

  By pre-setting arguments with POSITION and SIZE, you can work around the limitation on the length of a command. You do not need to specify the preset arguments in the BOX command.

- You want to use the enhanced capabilities of POSITION for adjusting the starting point of a box or line.

  With BOX, you can specify an offset for the starting point only as a whole number (non-negative integer). This command would print a box starting 1 CM to the right and 1 CM down from the left upper corner of a window:

  ```
  /:  BOX XPOS '1' CM YPOS '1' CM
  ```

  With POSITION; you can adjust the position of a line or box relative to a window much more precisely. In the POSITION command, you can specify positive and negative offsets and use non-integer numbers.

  Example: The commands shown below position a box slightly to the left and slightly above a window. This leaves a margin between the edge of the box and the text in the window.

  ```
  /:  POSITION XORIGIN '-.2' CM YORIGIN '-.2' CM
  /:  SIZE WIDTH '+.2' CM HEIGHT '+.2' CM
  /:  BOX FRAME 10 TW
  ```

  (Note that the box must be enlarged to accommodate the shift. If it is not enlarged, then it will not cover all of the window.)

  You can also use POSITION to set the starting point to the upper left corner of the active page format. Example: POSITION PAGE moves the starting point from the active window to the active page format.

- You want to use the relative sizing capabilities of SIZE to adjust the size of a box, line, or halftone.

  With BOX, you can make only absolute size specifications. BOX HEIGHT, for example, overrides the default height setting to the height of the current window.

With SIZE, you can adjust the size of a box or a line with respect to its previously-set dimensions. The following commands would, for example, draw a frame 1 CM in from the margins of the paper:

```
/:   POSITION PAGE
/:   POSITION XORIGIN 1 CM YORIGIN 1 CM
/:   SIZE PAGE
/:   SIZE HEIGHT '-2' CM WIDTH '-2' CM
```

Using the Commands in Texts and Forms

Since BOX, POSITION, and SIZE are text commands, you can insert them directly in a text. However, usually you use these commands in forms, where you have better control of how a box or line and the accompanying text fit together. SAPscript does not automatically fill text into a box or otherwise orient text with respect to these graphical elements.

Enter the following line as a command in text in a SAPscript document. The command draws a box of 17.5 CM length, 1 CM high, with 10% shading:

```
/:  BOX WIDTH '17.5' CM HEIGHT '1' CM INTENSITY 10
```

The left upper corner of the box is located at the left upper corner of the main window defined in the form of the document. The text that you type in is not automatically oriented in accordance with the box. Whether the text fits in the box or not depends on you. If you type in three lines of text, then the bottom line of text is likely to appear below the bottom of the box.

In a form, you can orient both text and graphical elements in the windows that you define. You therefore have much better control of how graphics and text fit together.

Tips and Guidelines

To ensure in forms that boxes, lines, and shading fit correctly with text, follow these guidelines:

- In your form design, match graphical elements and windows to each other. By default, a box defined in a window has the dimensions and starting point of the window.

  Defining a window for each graphical element that you want to include facilitates using boxes, lines, and shading, since the graphical element and the window have the same dimensions and positioning.

  Example: If a window is defined with the dimensions 6 CM high and 8 CM wide, then this statement in the text element of the window paints a 10 halftone with the same dimensions. The shading is oriented on the upper left corner of the window.

  ```
  /:  BOX INTENSITY 10
  ```

- Use the POSITION command to adjust the position of a box or line relative to a window.

  For example, these commands in a form window would allow more room above the first line of text in the window. The box would start 0.2 CM above the top of the window.

  ```
  /:  POSITION YORIGIN '-0.2' CM
  /:  SIZE HEIGHT '+0.2' CM
  /:  BOX INTENSITY 5
  ```

  Make sure to increase the size of the box to accommodate an offset. Otherwise, the box will not cover all of the window.

  In the example above, the SIZE command increases the height of the box by 0.2 CM to accommodate the positioning of the box above the window.

- Draw a horizontal line by setting the HEIGHT in a BOX command to 0. Draw a vertical line by setting WIDTH to 0.

  ```
  /:  BOX FRAME 10 TW WIDTH 0 TW HEIGHT '10' CM
          Vertical line 10 CM long
  ```

  ```
  /:  BOX FRAME 10 TW WIDTH '10' CM HEIGHT 0 TW
          Horizontal line 10 CM long
  ```

- Adjust the tabs in a window to match the position of lines and boxes.

  For example, you define a table with the commands shown below. The vertical lines in the table are drawn in at 10 CM and 13.5 CM from the left edge of the window:

  ```
  /:  BOX WIDTH '17.5' CM HEIGHT '13.5' CM FRAME 10 TW
  /:  BOX WIDTH '17.5' CM HEIGHT 1 CM FRAME 10 TW INTENSITY 15
  /:  BOX XPOS '10.0' CM WIDTH 0 TW HEIGHT '13.5' CM FRAME 10 TW
  /:  BOX XPOS '13.5' CM WIDTH 0 TW HEIGHT '13.5' CM FRAME 10 TW
  ```

  In the paragraph formats that you use to fill the table, you would define tabs at the positions shown below. With these tabs, your input would start right-justified in the first, second, and third columns of the table. You must ensure that your input is not too long to fit in the columns defined with the lines:

```
Format TB    Fill table
Tabs:      1     9.5 CM RIGHT
           2    13.0 CM RIGHT
           3    17.0 CM RIGHT
```

Styles

Styles: Components and Techniques [Seite 43]

Using Styles [Seite 47]

Releasing Styles [Extern]

Transporting, Copying, and Comparing Styles and Forms [Seite 96]

Styles: Components and Techniques

Concepts

Modifying SAP Styles [Seite 44]

Styles: Concepts [Seite 45]

Client and Language Versioning: Concepts [Seite 13]

Components

Header Data [Seite 14]

Paragraph Formats and Attributes [Seite 16]

Character Formats and Attributes [Seite 23]

Techniques

Displaying Versions of Styles [Seite 46]

Modifying SAP Styles

If you want to modify SAP styles, set up your development environment as follows:

1. Make sure that no SAP standard styles are stored as client-dependent copies in your development client.

   Such styles should be stored only in client 000, which is the SAP development and installation client. If you access SAP standard styles from within another client, the system uses the central copy from client 000.

   If you must delete SAP standard objects from your development client, read Notes 10388 and 3355 in the SAP Online Service System (OSS). These notes describe how to save edited styles and then delete all styles.

2. If you want to modify SAP standard styles, first copy them from client 000 to your development client. Then proceed as follows:

   – Assign a name from the name area reserved for customers to these objects (these names start with Y or Z).

   – Copy these objects into one of your own Y or Z development classes.

   Renaming SAP standard objects enables you to maintain and transport your changes using the Workbench Organizer. The Workbench Organizer is not available for SAP standard objects in clients other than 000.

Styles: Concepts

Use a **style** to define the paragraph and character formats for your documents, thus determining the text formatting. You can use a style, for example, to highlight certain character strings or entire paragraphs.

You can allocate a style to every text. However, usually styles are used for the main windows of forms, into which the user can directly enter text.

If you allocate a style to a window, the system deactivates any format specifications of the underlying form for this window. The system then formats and prints the text entirely according to the format specifications defined in the style.

Displaying Versions of Styles

To display the different versions of a style, choose *Utilities* → *Versions*.

The version list includes the following information:

- Version and status:
  - New
    No active version, not available for use as yet.
  - Active
    The current style is the active version, in effect in the System.
  - Revised
    The current style is being changed. There are active and edit versions of the style.
  - Translated
    The current style must be translated.
- Current language key
- Original language of the style.
- Description of the style.

You can display detailed information on a version of the style by putting the cursor on the version and choosing *Style info*.

Using Styles

In a style you can define character and paragraph formats that you want to use independently of forms. The SAP System provides styles, for example, for mail messages and for online documentation.

You can assign a style to any SAPscript text. If you do this, the text in the main window is formatted according to the paragraph and character definitions of the style, not the form. Any format definitions in the form are unavailable for use.

Observe these naming conventions for style names:

- The name must start with a letter.
- The name may contain only letters or numbers.

  The characters *****, **&** **/** as well as blanks are not valid.
- The name can be up to eight characters long.

Style Components [Seite 48]

Processing Options [Seite 49]

Style Components

- Header Data [Seite 14]

  In the header data, you can find a short description of the style and a default paragraph.

- Paragraph Formats and Attributes [Seite 16]

  Paragraph formats control the formatting of paragraphs in SAPscript.

- Character Formats and Attributes [Seite 23]

  Character formats are used for text formatting within paragraphs.

Processing Options

Style maintenance offers three processing options:

Use	For
Create/change	Creating and Changing a Style [Seite 50].
Display	Displaying a Style [Seite 55].
Catalog	Searching for Styles [Seite 56].

Creating and Changing a Style

To create a new style or change an existing style,

1. Choose *Tools* → *Word processing* → *Style*.

   The *Style: Request* screen appears.

2. Enter a style name in the field *Style*.

   Remember the input conventions when creating a new style.

3. Choose *Create/change*.

   The *Change Style Header* screen appears.

   The system automatically sets your logon language in the field *Language*.

4. Define the paragraph formats [Seite 51] of the style.
5. Define the character formats [Seite 53] of the style.
6. Complete the header data [Seite 54] of the style.

Defining Paragraph Formats

On the *Change Style Header* screen choose *Goto* □ *Paragraphs*. This takes you to the input screen for standard attributes, where you can define the paragraph formats required:

1. Choose *Edit* → *Create element*.

   A dialog box appears.

2. Enter the paragraph tag and description. Bear in mind the input conventions. Fill in both fields and confirm.

   The paragraph now exists. Line spacing, alignment, and units of measurement are set by the system.

3. Assign values to the input-enabled fields. You can overwrite the values set by the system. Your entries should correspond to the requirements for standard attributes.

4. Choose *Attributes* → *Font*. You can now specify font attributes for the paragraph tag just defined.

5. Enter values in the input-enabled fields according to the requirements for font attributes.

   The font attributes bold, italics, and underlined are set to * (inherited) by default. You can overwrite these values.

6. Choose *Attributes* → *Tabs*. You can now set tab positions.

7. Define the tab stops required in the input-enabled fields. Enter appropriate values for the tabs.

   When defining tab stops, you must not necessarily assign a value to the field *Alignment*. The default setting LEFT for left-aligned is used here, as for the alignment of the paragraph.

8. Choose *Attributes* → *Outline* to display the outline attributes and assign values, if necessary.

9. Enter values in the input-enabled fields according to the entry options for the outline.

   Numbering type, Outline, and *Number margin* are set by default. However, you can overwrite these values as required.

10. Repeat these steps for all paragraph tags you want to define.

To delete a paragraph format,

1. Mark the paragraph tag in the list.

2. Choose *Edit* → *Delete element*.

To save the paragraph formats, choose one of these two functions:

- *Style* → Save
- *Style* → Save *as*.

A dialog box appears. Enter a style name in accordance with the input conventions. The language key is taken from the logon language.

Defining Character Formats

Define character formats to determine text formatting within paragraphs. Choose *Goto* → *Character strings*. This takes you to the input screen for standard attributes, where you can define the character formats required:

1. Choose *Edit* → *Create element*.

   A dialog box appears.

2. Enter the character tag and a description.

   The character tag is created. As for paragraph tags, the system automatically sets the other values. By default, it assigns a * to each character attribute, which means that the attributes are taken from the next highest hierarchical level (form).

   You can overwrite these values.

3. Choose *Attributes* → *Font* to specify fornt attributes. You use font attributes to highlight blocks of text or character strings.

   The system assigns a * to the fields *Bold, Italics*, and *Underlined*. You can overtype these entries.

4. Enter the values required in accordance with the definition of font attributes.

5. Repeat these steps for all character tags you want to define.

6. Save the character formats.

Completing the Style Header Data

After defining all relevant formats and attributes, you must complete the header data:

1. Choose *Goto* → *Header* to go to the *Change Style Header* screen.
2. Fill in the fields *Description* and *Default paragraph*.

   The default paragraph is the paragraph the system uses if ***** is entered in the tag column of the SAPscript editor.

   If you start a new paragraph in your text by pressing ENTER, the system enters ***** in the tag column.
3. Save your style.

Displaying a Style

To display a style,

1. Choose *Tools* → *Word* processing → *Style*.

   The *Style: Request* screen appears.

2. Enter the style required in the field *Style*.

   The system sets your logon language in the field *Language*.

3. Choose *Display*.

   The header data of the selected style is displayed.

4. Choose *Goto* → *Paragraphs* to display the paragraph formats of the style.

   A list of all available paragraph tags is displayed. The standard attributes of the currently active paragraph format appear in the lower half of the screen.

- To display the standard attributes of another paragraph tag, position your cursor on the paragraph format required and choose *Choose*. Select the preceding paragraph tag with F7 (Previous operation) or the following paragraph tag with F8 (Next operation).
- To display the font attributes of a paragraph format, choose *Attributes* → *Font*. The corresponding font attributes are displayed in the lower half of the screen.
- To display the tabs of a paragraph format, choose *Attributes* → *Tabs*. All tab positions defined for the selected paragraph format are displayed.
- To display the outline of a paragraph format, choose *Attributes* → *Outline*. All outline attributes of the active paragraph are displayed.

5. Choose *Goto* → *Character strings* to display all character formats defined. The standard attributes of the currently active character format are displayed.
   - To display the standard attributes of the character format required, position the cursor on the character tag and choose *Choose*. Select the preceding character tag with F7 (Previous operation) or the following character tag with F8 (Next operation).
   - To display the font attributes, choose *Attributes* → *Font*. The information required is displayed in the overview in the lower half of the screen.

Searching for Styles

To display the available styles, choose *Tools* → *Word processing* → Style. The *Style: Request* screen appears.

1. Leave the field *Style* empty.

   The field *Language* is set by the system according to your logon language.

2. Choose *Catalog*.

   A hit list of all styles, whose selection criterion was the field *Language*, is displayed in alphabetical order. The list contains this information:

- Style name
- Description
- Current language key
- Original language of the style
- Current status

New	No active version
Active	Current object is active and operational version
Revised	Object is being revised; there is an active and an edited version
Translated	Current object is ready for translation

- Development class of the style

To display even more information about a style, choose *Style info* in the hit list.

This takes you to a list with comprehensive information about the selected style. Development-related information is displayed as well as definitions contained in the respective style. The list is subdivided into areas.

- Header data
- Characters
- Paragraphs

You can also search for character strings and download lists.

3. Select a style.

   Your initial screen reappears with the style you selected. You can now edit this style.

Stile freigeben

Stile müssen für den Endbenutzer freigegeben werden, da sie sonst dem Benutzer nicht angezeigt werden.

Zur Gewährleistung von fehlerfreien Stilen wird geprüft, ob sämtliche Mußeingaben vorgenommen wurden und ob alle Eingaben den Konventionen entsprechen.

Dazu wählen Sie in der Menüleiste *Stil* → *Prüfen*. In der Statusleiste wird der Status der Prüfung angezeigt, d.h., das System teilt Ihnen mit, ob die Prüfroutine Fehler gefunden hat oder nicht.

Wurde die Prüfroutine fehlerfrei beendet, müssen Sie Ihren Stil noch aktivieren. Wählen Sie dazu in der Menüleiste *Stil* → *Aktivieren*. Sie erhalten eine Meldung vom System, ob die Aktivierung erfolgreich war oder nicht.

Versuchen Sie trotz fehlerhafter Prüfung oder ohne Prüfung, Ihren Stil zu aktivieren, wird der Aktivierungslauf abgebrochen. Der Stil ist für den Endbenutzer nicht verfügbar.

Auch nach der Aktivierung können Sie sowohl Stile weiterbearbeiten und beispielsweise ändern, übersetzen oder löschen.

Wenn Sie in der Menüleiste *Stil* → *Löschen* wählen, wird ein Dialogfenster angezeigt. Hier werden Sie gefragt, ob Sie Ihren Stil wirklich in allen Sprachen löschen möchten. Positionieren Sie den Cursor auf der entsprechenden Antwort, und bestätigen Sie mit `ENTER` oder `F2`. Nach dem Löschen kehren Sie wieder zum Anforderungsbild zurück. In der Statusleiste wird der Status der Verarbeitung angezeigt.

Forms

Design Tools

SAPscript offers two methods for designing your forms: the alphanumeric Form Painter and the graphical Form Painter. The graphical version allows you to easily and intuitively design your forms, starting with Release 4.0. However, you can use the graphical Form Painter only if your frontend is equipped with the 32bit operating system Windows95 or WindowsNT.

Alphanumeric Form Painter [Seite 60]

Graphical Form Painter [Seite 81]

Alphanumeric Form Painter

The alphanumeric Form Painter is a tool for designing the form layout and maintaining the forms. You can always use this Form Painter version, regardless of the frontend operating system you are working with.

If your frontend is equipped with the 32bit operating system Windows95 or WindowsNT 4.0, you can choose between the alphanumeric and the Graphical Form Painter [Seite 81]. On the *Form:Request* screen, choose *Settings* → *Form Painter....*

Forms

Forms are used for the page layout of SAPscript documents. To be able to format a text for output to the screen or printer, you must assign a form first.

> If no form has been assigned to a text, the system automatically assigns the form SYSTEM, which contains minimal definitions for text formatting.

There are two ways of formatting texts using forms:

- Use the standard text maintenance to enter and print the text. You can assign any form. You can also enter text via the form, for example, a letter header.
- Use an ABAP program to format the text according to an assigned form. The program can either dynamically print individual predefined text modules – text elements – or transfer entire texts to be printed using the form.

Observe these naming conventions for form names:

- The name must start with a letter.
- The name may contain only letters.

  The characters `*`, `&` `/` as well as blanks are not valid.
- The name can be up to 16 characters long.

Overviews and Operations

Form Components (Alphanumeric Form Painter) [Seite 62]

Processing in Overview [Seite 76]

Finding a Form [Seite 77]

Test-Printing a Form [Seite 79]

Converting the Page Size of a Form [Seite 80]

Working With Form Components

Form Components (Alphanumeric Form Painter)

In the alphanumeric Form Painter, a form has the following components:

- Header data

  Data related to development (created by, development class, and so on) and form information (which elements are used) are both stored in the header data.

- Paragraph formats

  Paragraph formats are required in forms (as in styles) to format texts. However, they are also used for word processing in forms, for example, to format text elements.

- Character formats

  You can also use character formats to format texts or paragraphs. Unlike paragraph formats, however, they are used to format text within a paragraph.

- Windows

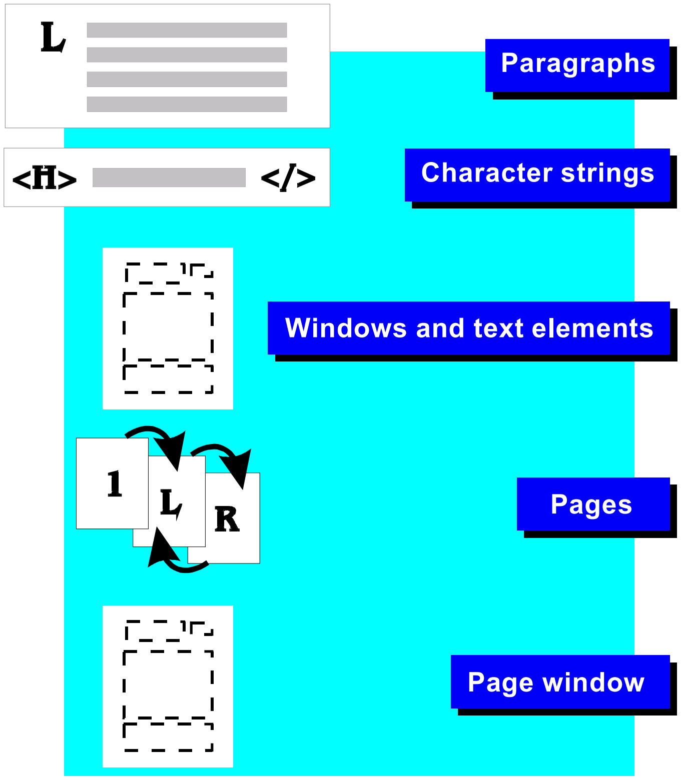
  Windows are output areas you position on the individual pages of the form. The system then prints the contents of the windows, the text elements, into these output areas.

- Pages

  Most forms comprise different pages, for example, one page with the customer address, the subsequent page containing the corresponding customer records. This implies that you must define different output areas (windows) on different pages.

- Page windows

A page window is the combination of a window and a page. You specify the dimensions of a window and its position on a page.


L
<H> </>
1
L
R
Paragraphs
Character strings
Windows and text elements
Pages
Page window

Defining Header Data

The header data of a form consist of *Administration data* and *Basic settings*.

The *Administration data* include *Administration information* and *Language attributes*.

- In the *Administration information* group, enter
  - Form name
  - Description
  - Development class
- In the *Lang. attributes (Language attributes)* group, you find
  - Original language
  - Current language (*language key*)
  - Information on the translation of the form.
  - Status information about the current form status (active, edited, and so on).

Before entering other header data,

1. Define the paragraph formats [Seite 66] of the form.
2. Define the character formats [Seite 67] of the form.
3. Return to the header and complete it.

Completing the Header Data After Defining the Attributes

1. Choose *Goto* → *Header*.

   The *Change Form Header* screen appears.

   The Basic data include the groups *Set up page* and *Default values for text formatting*.

- In the group *Set up page*, set defaults such as page format and page orientation.
- In the group *Default values for text formatting,* set a default paragraph, default font, and so on.

2. Enter values in accordance with the guidelines. The fields *Tab stop, Page format, Orientation, Lines/inch*, and *Characters/inch* are assigned default values by the system.
3. Enter the format of the pages in the header data of the form. This format can be taken from the spool administration table using the transaction SPAD.

The system always uses these values defined in the form header if no other values are specified for the paragraph and character format definition.

To activate the form, you must enter the *First page* and the *Default paragraph* as *Basic settings*.

Passing Data to an External Program

In the R/3 system, you can flag a form for **external printing**. If a form is not explicitly flagged for external printing, the system uses application-dependent settings from Customizing.

To flag a form for external printing

1. Choose *Tools* → *Word processing* → *Forms.*
2. In the header data, choose *Attributes* → *Miscellaneous.*

   In the lower part of the screen, you can now set a flag for passing data to an external program. The flag in the form then determines whether to print via the Raw Data Interface (RDI) or not. To overrule this setting, use the function module OPEN_FORM (see Raw Data Interface [Extern]).

Defining Paragraph Formats

Before entering all header data on the *Change Form Header* screen, you must create the form by defining paragraph and character formats. The system sends a warning if, in the header data, you specify a paragraph or a first page that has not yet been defined.

1. Choose *Goto* → *Paragraphs*.

   This takes you to the input screen for paragraph formats and their standard attributes.

2. Choose *Edit* → *Create element*.

   A dialog box appears.

3. Enter the paragraph tag and a short description.

   The paragraph tag is created. The system sets line spacing, alignment, and units of measurement.

4. Choose *Attributes* → *Font* to define the font attributes for the paragraph tags.

5. Enter values in the input-enabled fields in accordance with the guidelines.

   You can overwrite the default font attributes.

6. Choose *Goto* → *Tabs* to specify the tab position.

   An input screen appears, where you can define the required tab positions as well as the unit of measurement and the alignment. In the standard version, the alignment is set to LEFT (left-aligned).

7. Choose *Attributes* → *Outline* to define the outline.

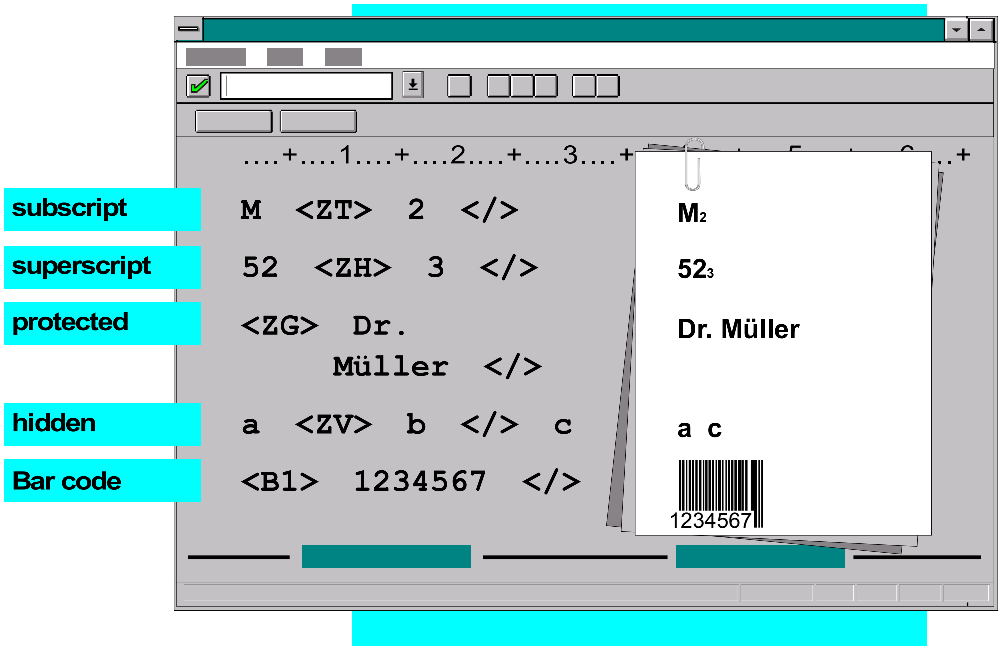
   Additional values are displayed that have been set by the system.

8. Enter values in the input-enabled fields. You can change the values set by the system.

9. Repeat the steps above until all paragraph formats are defined.

Defining Character Formats

Character formats in forms determine how text is formatted within paragraphs. They are a refined form of text layout.



```
....+....1....+....2....+....3....+
M  <ZT>  2  </>
52  <ZH>  3  </>
<ZG>  Dr.
     Müller  </>
a  <ZV>  b  </>  c
<B1>  1234567  </>
```

1. Choose *Goto* → *Character strings*.

   This takes you to the request screen for standard attributes of character strings.

2. Choose *Edit* → *Create element*.

   A dialog box appears.

3. Enter a character tag and description.

   The character tag is created. The system sets all standard attributes to *. You can overwrite this setting.

4. Choose *Attributes* → *Font* to define the font attributes for the character format. Font attributes provide a more detailed definition.

5. Enter values in the input-enabled fields.

   The fields *Bold, Italics*, and *Underlined* are automatically displayed with *. You can overwrite this with your entry.

6. Repeat the definition of character tags and their standard and font attributes until all required character formats have been defined.

7. Save the character formats.

Defining Windows

You must define at least one **window** for every form. Otherwise it is not possible to format texts. Window definition involves a list of window names and corresponding window types. Units of measurement are not taken into account here.

1. Choose *Goto* → *Windows*.

   This takes you to the input screen for windows. A main window (MAIN) is displayed automatically. You can define a default paragraph here.

2. Choose *Edit* → *Create element* to add new window names and types to the list.

   A dialog box appears.

3. Enter a window name and description.

   Input-enabled fields are displayed in the lower half of the screen. The field *Description* receives the value from the dialog box, the value for the field *Window type* is set by the system.

   By default, the system sets VAR as the window type for windows with variable contents. If you prefer another window type, simply overwrite this value.

4. Make your entries or changes.

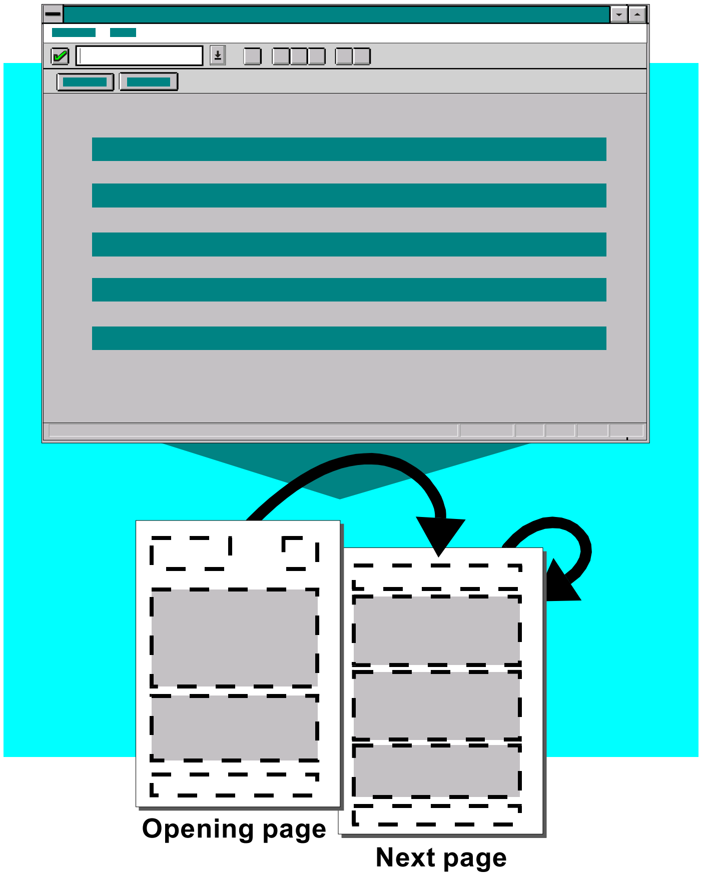
   You can define one standard paragraph per window in the field *Default paragraph*.

5. Repeat these steps for each of your windows.

6. Save your entries.

Defining Pages

You must define at least one logical page for every form. Otherwise it is not possible to format texts. To define a page, assign a name and specify attributes.



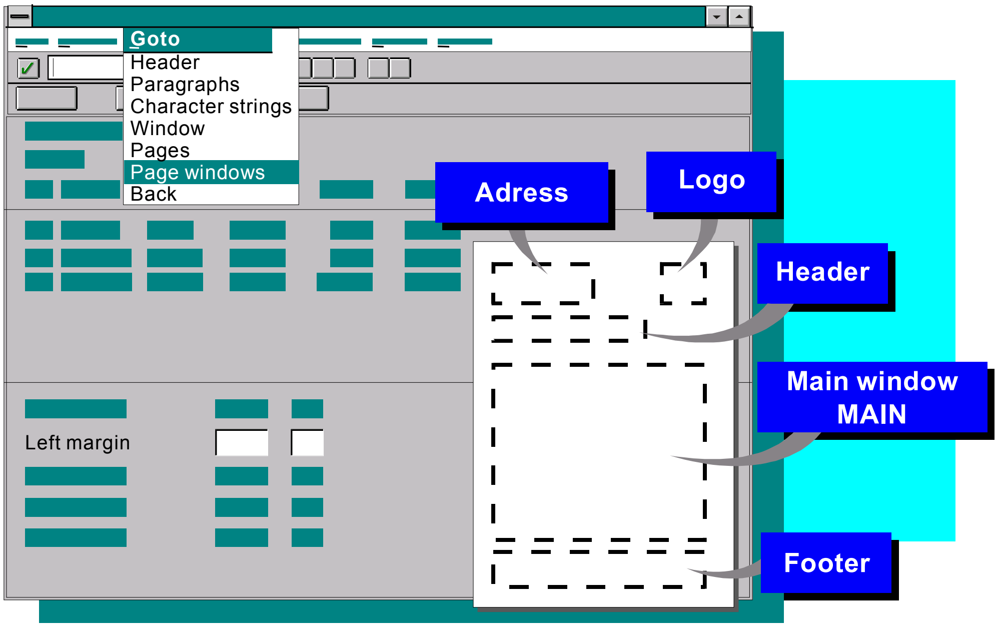
1. Choose *Goto* → *Pages.*
2. Choose *Edit* → *Create element*.
3. Enter a page name and a short description in the fields *Page* and *Description*.

   Input-enabled fields are displayed in the lower screen area.
4. Enter values in accordance with the guidelines for page definition.

   You can overwrite the default values set in the fields *Mode* and *Numbering type*.
5. Repeat these steps until all pages are defined.
6. Save your entries.

Defining Page Windows

To define a page window, allocate a logical window to a physical page and specify the position and size of the window (the window is logical, because you define it only once in the entire form by specifying name and type, but not its usage). However, the size is limited by the page format selected in the header data.



1. Choose *Goto* → *Page windows*.
2. Choose *Edit* → *Create element*.

   A list of defined logical windows appears.
3. Select the desired window from the list.
4. Enter valid values in the fields displayed in the lower half of the screen. The unit of measurement is set by the system.
5. Specify the position of the selected window on the page and the dimensions of the window.
6. To define page windows for the next page, you must first select the next page. Choose *Goto* → *Pages* and select the next page. Then choose *Goto* → *Page windows* and repeat the steps above.
7. Save your form.

To delete a selected element, choose *Edit* → *Delete*.

Filling Page Windows with Text

When defining page windows, you can request multiple columns. Define an area on a page and specify the position of the main windows in this area, as described in Defining Main Windows in Page Windows [Seite 73].

To fill a page window with text, you can also use text elements as described in Using Text Elements in Page Windows [Seite 72].

Using Text Elements in Page Windows

You can define and edit text elements in windows as well as in page windows. In both cases, the text elements are assigned to the active window. You can create several text elements in a module.

1. On the screen *Form: Change Page Windows* choose *Edit* → *Text elements*.

   This takes you to the text editor.

2. Enter the name of the text element in the text entry area by the tag column **/E**.

3. Define one line in the text element for each line in the main window. A section of a text element is shown below:

   ```
   /E   SPACE_LINE
   /
   /E   HEADER_TEXT
   /:   INCLUDE 'SD_RVADOR01' OBJECT TEXT ID
        SDVD PARAGRAPH HT
   /:   INCLUDE  OBJECT VBBK ID 0001 PARAGRAPH HT
   /E   ITEM_HEADER
   IL   <TI>Item,,Material,,Description</>
   IP   <TI>,,,,Quantity,,,,Price,,Price unit,,,,,,Value</>
   /    &ULINE(71)&
   /
   /E   ITEM_LINE
   IL   &VBDPA-POSNR&,,&VBDPA-MATNR&,,&VBDPA-ARKTX&
      ,,&'Customerarticlenumber 'VBDPA-IDNKD'
        '&&'Item 'VBDPA-POSEX&
   ```

   As you can see above, each line is defined individually.

4. Save the window texts (text elements) and return to the *Form:Change Page Windows* screen.

Defining Main Windows in Page Windows

Main windows in page windows allow you to format text in multiple columns. Define an area in the page window, in which to position the main windows.

1. Create a page window and assign it to a page.
2. Choose *Edit* → *Main windows*.

   A dialog box appears.
3. Enter values in the fields *Area width* and *Area height* in accordance with the input guidelines for main windows.
4. Enter values in the fields *Spacing* and *Number* in the *Horizontal* group if you want to use multiple columns. You can ignore the fields in the *Vertical* group.
5. Determine how many columns and line areas are required for label printing. Then enter the corresponding values in the fields in the *Horizontal* and *Vertical* groups.

   The value in the field *Left margin* varies from main window to main window if multiple columns are used. The following applies:

	Left margin of current column
+	Window width
+	Horizontal spacing
=	Left margin of next column

   In label printing, the field *Upper margin* also varies from main window to main window:

	Upper margin of current main window
+	Window height
+	Vertical spacing
=	Upper margin of next main window

6. Enter a value in the field *Start position.*

   This is a counter. Enter a starting value which is equal to or greater than 1.

   The main windows are added to the list.
7. Save your form.

Entering Documentation

When modifying forms, you must know the meaning of windows and text elements. Of special importance is the description of program symbols, their meanings and availabilites. As the creator of a form, you are responsible for maintaining these data in the form documentation. Make sure to describe also those program symbols that are not used in the standard version, but that contain values that may be important when modifying the form.

Format of the Form Documentation

To enter form documentation, on the *Form: Request* screen

1. Mark *Documentation.*
2. Choose *Create*.

   Depending on whether you are working in edit or display mode, you can either create or change the documentation or simply read it.

If you create documentation, the system automatically generates a template for the specified form as a model. This template contains read-only lines that represent certain sections. You can now enter documentation for each of these sections..

The read-only lines represent the different objects of a form (general information, pages, windows, and text elements). Each object begins with a key word followed by the corresponding name. The general section starts with FORM, followed by the form name. Then follow the different pages (key word PAGE), each one with its name. The system then lists the windows (ley word WINDOW) and finally the text elements (key word ELEMENT).

The names of text elements are unique only within one window, not within the entire form. Therefore, the system displays the text elements in the form documentation as a combination: window name-text element. As a proposal for the documentation of these objects, the system enters the short texts (description of the form, the pages, and the windows) into the input-enabled screen part. You can now add any relevant information.

Automatically Adjusting the Form Documentation

If a form documentation exists and you enhance the form (new pages, windows, or text elements), the system automatically adds these new objects to the form documentation when you save the form. You can then enter the appropriate documentation for these objects as described before.

If you delete pages, windows, or text elements of a form, the system does not delete the exising documentation, but flags it as no longer needed (‘*’ in front of the corresponding key word). If you create an object with the same name at a later time, the system deletes the flag, thus preserving the documentation. No longer needed objects always appear at the end of the objects list, but in the same sequence as before (general, pages, windows, text elements).

Deleting no Longer Needed Sections

You can delete documentation for objects you no longer need (see above). On the *Form: Request* screen, choose *Utilities* → *Documentation* → *Clean up*. The system then deletes all sections that are flagged as no longer needed ('*' in front of the key word). It also deletes all read-only lines for which no documentation (0 lines) exists.

To delete the entire form documentation, on the *Form: Request* screen choose *Utilities* → *Documentation* → *Delete*.

Printing the Form Documentation

To print the form documentation, choose *Utilities* → *Documentation* → *Print*. The system prints only those objects that are actually documented.

Storage and Transport

The system stores the form documentation in the form itself. This ensures that the documentation is language-dependent and takes part in the transport.

Translation

The form documentation is linked to the form translation. Choose *Window texts and documentation*. The list created contains the individual windows and, at the end, the *Documentation,* if it exists. The documentation can be translated there.

The subdivision into objects (general, pages, windows, text elements) contains both original and target language. This corresponds to the translation of text elements. There is no status administration for translated objects (new, edited, complete). Therefore, the system asks whether the documentation-if any-was translated completely when you release the form.

Processing in Overview

Form maintenance offers three processing options:

Create/change	Change a form that already exists or create a new one.
Display	Display a specific form.
Catalog	Display all available forms.

To create a form,

1. Enter the form name and fill in the header information [Seite 64].
2. Define any paragraph formats [Seite 66] and character formats [Seite 67] that you need for the form.
3. Define page formats [Seite 69] for the form.
4. Define the windows [Seite 68] you want to position on the pages.
5. Define text elements [Seite 72] in windows. These are the default texts and the texts selected by the program that can be printed in windows.
6. Specify how the windows are to be placed on the pages by defining page windows [Seite 70].

   For additional information relating to page windows, see Using Text Elements in Page Windows [Seite 72] and Defining Main Windows in Page Windows [Seite 73].

Finding a Form

To search through the available forms,

1. Choose *Tools* → *Word Processing* → *Form*.

   The field *Language* is set to your logon language.

2. Leave the field *Form* empty.

3. Choose *Find*.

   A hit list of all forms whose selection criterion was the field *Language* is displayed in alphabetical order. The following information is contained in the hit list:

   - Form name

     The form name specified by the creator

   - Description

     Short description of the form contents

   - Current language key

   - Original language

     Language for which the form was originally created

   - Current status

New	No active version
Active	Current object is active and operational version
Revised	Object is being revised; there is an active and an edited version
Translated	Current object is ready for translation

   - Development class

     Class in which the form was developed

   To display more information about a form, choose *Form info* in the hit list.

   This takes you to a list with comprehensive information about the selected form. Development-related information is displayed as well as definitions contained in the respective form. The list is subdivided into areas.

   - Header data
   - Characters
   - Windows
   - Pages
   - Text elements

   You can print the list. You can search through the list for character strings. You can also download the list to a local file.

4. Select a form.

   Your initial screen appears again, but with the selected form that you can now process further.

Test-Printing a Form

You can test a form without having to call the print program that uses it. Choose *Utilities* → *Test print*. SAPscript presents the print selection screen so that you can pick the printer type for which the test print should be formatted.

The *Test print* function

- Displays or prints all pages and text elements that are defined in the form.
- Replaces system, standard, and text symbols with their current values. Program symbols are filled with a test value, the character X.

If you test print from the *Form: Request* screen, SAPscript uses the active version of a form for the test print. If you are currently editing a form, then SAPscript uses this edit version of the form for the test print.

Under certain conditions, texts may not be printed in *Test print*: This is the case, for example, when the name of a text in an INCLUDE statement is defined with a program symbol.

Converting the Page Size of a Form

You can convert the page size used in a form. For example, to use SAP standard forms in the USA, you can convert a form from DINA4 to LETTER paper size.

To convert the page size of a form, on the *Form: Request* screen

1. Choose *Utilities → Change pages*.
2. Specify the name of the form and the target page size.

   Only active forms can be converted.

The conversion function can of course only convert between page formats that are of similar size, such as DINA4 and LETTER.

Make sure that you convert only forms you have copied into the customer naming space (form names beginning with Y or Z). To activate your converted form in the SAP System, you must enter the name of the new form as required in the print program or form specifications in the SAP Customizing System.

The conversion function

- Determines whether there are window pages that do not fit on the new page size. The conversion function checks for both vertical and horizontal overhang off the edge of the new paper.
- Reduces the size of the main window and moves the overhanging page window so that it will fit, if a page window does not fit on the new paper.
- Logs the results of the checks and any changes made in a list you can display.
- Allows you to choose a test mode, in which case only the log is generated; no changes are made to the form.

The conversion uses the physical dimensions of the paper to check that page windows will fit. Therefore, page windows may extend into the non-printable areas at the edges of a sheet of paper. After converting the page size, you should test print the form to check for this error and for any unwanted formatting changes.

Graphical Form Painter

Besides the ever available alphanumerical Form Painter, you can use the graphical Form Painter to design the layout of your forms – provided that you frontend meets certain prerequisites.

To switch between alphanumeric and graphical Form Painter, on the *Form: Request* screen choose *Settings* → *Form Painter....*

Before activating the graphical Form Painter, inform yourself about the Technical Prerequisites [Seite 82].

For details on the individual functions, see

Compatibility [Seite 83]

Functions of the Graphical Form Painter [Seite 84]

Technical Prerequisites

The operating system of your frontend must meet the following prerequisites to allow you to use the new graphical Form Painter:

Platform

On your frontend, you must have installed either WindowsNT 4.0 or Windows95. At present, other platforms are not supported.

Control files

You need several.DLL and OCX files.
The actual installation of these required components is carried out automatically via the SAP GUI (uploading the.DLL and OCX files). This requires no action by the user.

To switch between alphanumeric and graphical Form Painter, on the *Form: Request* screen choose *Settings* → *Form Painter....*

Compatibility

The graphical Form Painter (starting with Release 4.0) is entirely downward compatible. You can easily display and change any forms created with the alphanumeric Form Painter.

To start the graphical Form Painter, on the *Form: Request* screen choose *Settings* → *Form Painter....*

Functions of the Graphical Form Painter

You use forms to layout the pages of SAPscript documents. To process a document for printing, both on the screen or printer, you must assign a form first. Output is not possible without a form.

If a text has no form, the system automatically assigns the form SYSTEM, which contains the minimum of definitions required to process the text for output.

With the graphical layout design, creating and modifying forms becomes much easier. You can use the mouse to intuitively make changes, which you can immediately see on the screen (drag & drop, cut & paste).

Details on the Graphical Form Painter

Form Components (Graphical Form Painter) [Seite 85]

Administration Screen [Seite 86]

Design Window [Seite 90]

Operations

Finding a Form [Seite 77]

Test-Printing a Form [Seite 79]

Converting the Page Size of a Form [Seite 80]

Form Components (Graphical Form Painter)

A form of the graphical Form Painter consists of five components. The original components of the alphanumeric Form Painter (pages, windows, text elements) are united under the object *Page layout*.

On the initial screen of the graphical Form Painter, you can select one of the following components:

- **Header data**

  The header data store data related to the developer (for example, development class) as well as information on the form (which elements are used) (see also Defining Header Data [Seite 64]).

- **Page layout**

  The page layout allows you to define the individual pages and windows needed for the page layout. You can use the mouse to directly position the windows on an extra design window (see also Page Layout [Seite 230]).

- **Paragraph formats**

  You need paragraph formats in forms as well as in Styles [Seite 42] for processing the text. But they also serve for the internal text processing within the form, for example, for processing the text elements (see also Defining Paragraph Formats [Seite 66]).

- **Character formats**

  You can format texts or paragraphs using character formats. The only difference compared with paragraph formats is that you use character formats to layout the text within one paragraph (see also Defining Character Formats [Seite 67]).

- **Documentation**

  As the creator of a form, you are responsible for the form documentation. This documentation is designed to provide all information on windows and text elements required for a modification of the form (especially the description of the program symbols used) (see also Entering Documentation [Seite 74]).

Administration Screen

You use the administration screen in combination with the Design Window [Seite 90] to determine the output areas (windows), define the pages, and position the windows on the respective pages.

After opening a form (*Create/Change*), you use the *Administration Data* screen to determine the *Administration information* in the header data and set the *Basic settings* of the form. In addition to this **Administration screen** with its page and window data and several editing and display functions, you can call a **design window** to graphically design the page layout. To call this design window, choose *Layout*. The system always synchronizes these two windows whenever you execute an action on one of them.

On the administration screen, you edit the different pages and the respective windows in two structured areas. There, the system offers the most important menu functions on pushbuttons.

The Page Area [Seite 87]

The Window Area [Seite 88]

Pushbuttons

- **Display/Change**

  Corresponds to *Form* → *Display*↔*Change*.

- **Check definition**

  Corresponds to *Form* → *Check* → *Definition*. With this function, you can check whether all definitions required in the form exist (default paragraph, page format, and so on).

- **Check texts**

  Corresponds to *Form* → *Check* → *Texts*. Use this function to check the individual window texts (text elements). The system checks whether the control commands used are syntactically correct and correctly nested. It also checks whether the paragraph and character formats used are known in the form.

  This pushbutton is available only within the *Layout*.

- **Activate**

  Corresponds to *Form* → *Activate*. You must activate forms to make them accessible to the corresponding print programs.

- **Header data**

  Corresponds to *Goto* → *Header data*. Use this function to brach to the form header data.

- **Documentation**

  Corresponds to *Goto* → *Documentation*. Use this function to branch to the documentation of the windows and text elements used in the form.

The Page Area

Select pages

Use the combobox to display all pages defined and to directly select one. Choose *Start page, Next page, Overview* (for more information, see Pages [Seite 27]).

Create pages

To create individual pages, choose *Edit* → *Page* → *Create*. The system automatically assigns a default name and creates a main window. To rename the page, choose *Edit* → *Page* → *Rename*....

Copy pages

To create a page, you can also use the copy function (choose *Edit* → *Page* → *Copy*). The system copies all windows including the respective text elements and assigns a default name.

Delete pages

To delete individual pages, choose *Edit* → *Page* → *Delete.*

Page sequence

To display the current page sequence, choose *Overview*.

Page attributes

To change the next page and the meaning of the page, choose *Other attributes*.

Here, you can determine the numbering and the page counter mode. You can set print attributes such as ressource name and print mode for the respective page.

The Window Area

Select windows

Use the combobox to display a list of all windows on the current page. To switch to another window, select the window from the combobox or click on the window in the design window (for more information, see Windows [Seite 26]).

Create windows

To create new windows, choose *Create*. By default, the system creates a variable window (type VAR) and assigns a default name. To rename the window, choose *Edit* → *Window* → *Rename*. If you want to choose which type of window to create, you must use the function from the menu bar instead of the pushbutton to create the window.

To create a main window, you must choose *Edit* → *Windows* → *Create* → *Main window*.

Change window type

To change the type of a window (main, var), choose *Edit* → *Windows* → *Change type*.

Delete windows

To delete individual windows on the current page, choose *Edit* → *Window* → *Delete*. If the window does not appear on any other page, the system deletes it completely.

Cut windows

To remove individual windows from the current page, choose *Cut*. The system stores the object in the clipboard. To insert the object on any page of the form, choose *Paste*. The clipboard stores the text elements, the window size, and the position.

Copy windows

To copy individual windows into the clipboard, choose *Edit* → *Window* → *Copy.* To insert them on any page, choose *Edit* → *Windows* → *Paste*. The system copies all windows including their text elements, sizes, and positions.

Paste windows

When pasting a window, the system keeps the name of the window unless the window already exists on this page. If the window already exists on the current page, the system assigns a default name.

Align windows with the grid

To align all windows positioned on one page automatically with the closest grid tab, choose *Align to grid*.

Edit windows

To edit the text elements of the windows, choose *Text*. The system branches to the line editor or to the WYSIWYG PC editor, depending on the setting (SO10). Within the editor, you can switch between these two editors by choosing *Goto* → *Change editor*. To edit the text elements, choose *Edit* → *Text elements*.

Main window: A main window must always have the same width on all pages (unlike the variable window). If you change the width of the main window, the system executes this change on all pages.

Design Window

When you choose *Layout* within the Form Painter, the system displays a second window, the design window. Here, you can use the mouse to design the page layout (drag & drop, cut & paste). From within the administration screen, you can assign two different modes to the design window. Choose *Design/Text* to switch between

- Design mode
- Text mode

In design mode, you can

- Use the mouse to position the windows.
- Change the window size.
- Select a window for editing.

The design window has a grid surface to allow easy aligning of the windows.

To set the graduation of the grid in different measurement units, choose *Utilities* → *Options*. In addition, you can zoom the entire design window.

In text mode, the system displays the text elements of the individual windows. To see the entire text, scroll in the window. However, the text is read-only. To edit it, branch to the editor by choosing *Text* on the administration screen.

Inserting Graphics

Storing Graphics as Text (obsolete)

1. In the maintenance window, choose *Edit* → *Graphics* → *Create*.
   The dialog window for inserting graphics appears.
2. Enter the name and the language of the graphic.
3. In the group box *Raster screen attributes* select the desired attributes.
4. Choose *Continue*.

For more information, see Graphics [Seite 34].

To insert *.tiff graphics, choose *Import*. For more information, see Importing Graphics [Seite 92].

Storing Graphics on the Document Server

1. In the maintenance window, choose *Edit* → *Graphics* → *Create*.
   The dialog window for inserting graphics appears.
2. Enter the name, the object, and the ID of the graphic.
3. In the group box *Raster screen attributes* select *Raster screen black/white* or *Raster screen color*.
4. In the *Resolution* field enter the resolution of the graphic.
5. Choose *Continue*.

Importing Graphics

Storing Graphics as Text (obsolete)

1. In the maintenance window, choose *Edit* → *Graphics* → *Create*.
   The dialog window for inserting graphics appears.
2. Choose *Import*.
   The dialog box for importing graphics appears.
3. Enter the complete path of the graphic file.
4. In the group box *Save as* enter the name, language, and short description of the graphic.
5. In the group box *Raster screen attributes* select the attributes and the desired resolution of the graphic.
6. Choose *Continue*.

Storing Graphics on the Document Server

1. In the maintenance window, choose *Edit* → *Graphics* → *Create*.
   The dialog window for inserting graphics appears.
2. Choose *Import*.
   The dialog window for importing graphics appears.
3. Enter the complete path for the graphic file.
4. In the group box *Save as* enter the name, the description, and the type of the graphic.
5. In the group box *Print attributes* enter the print attributes.
6. Choose *Continue*.

As file formats for the graphic to be imported you can choose either *.BMP or *.TIF.

Creating Labels

Use

You can create labels, for example, for address stickers. For labels, several main windows are created on one page.

Procedure

1. In the maintenance window, choose *Edit* → *Windows* → *Create* → *Labels*.
   The dialog window for editing a main window appears.
2. In the group box *Area* enter the left and the upper margins as well as the area width and hight of the individual labels.
3. In the group box *Horizontal* enter the horizontal distance between the labels and the number of labels.
4. In the group box *Vertical* enter the vertical distance between the labels and the number of labels.
5. In the group box *Positions* enter the starting position.
6. Choose *Continue*.

Editing Other Page Attributes

Use

You can edit the description of the page, the next page, the page counter, and the print attributes.

Procedure

1. In the maintenance window, choose *Other attributes*.
   The dialog window for the page attributes appears.
2. You can edit these attributes:
   - In the field *Description* enter the page description. You can enter any text.
   - In the field *Next page* enter the next page. When printing with the form, the system prints the next page immediately after the current page.
   - In the group box *Mode* select the desired page counter. If you want to set the counter to 1, select *Initialize counter*. If you want to increase the counter by one, select *Increase counter*. If you want to keep the counter unchanged, select *Not changing counter*.
   - In the group box *Numbering* enter the numbering type and the output length. For the numbering type, you can choose between ARABIC numbers, LETTERs, and ROMAN numbers.
     If desired, select the uppercase atribute for letter numbering.
   - In the group box *Print attributes* enter the resource name and the print mode.
     For more information, see Pages [Seite 27].
3. Choose *Continue*.

Formulare freigeben

Formluare müssen für den Endbenutzer freigegeben werden, da sie sonst dem Benutzer nicht angezeigt werden.

Zur Gewährleistung von fehlerfreien Formularen wird geprüft, ob sämtliche Mußeingaben vorgenommen wurden und ob alle Eingaben den Konventionen entsprechen.

Dazu wählen Sie in der Menüleiste *Formular* → *Prüfen*. In der Statusleiste wird der Status der Prüfung angezeigt, d.h., das System teilt Ihnen mit, ob die Prüfroutine Fehler gefunden hat oder nicht.

Wurde die Prüfroutine fehlerfrei beendet, müssen Sie Ihr Formular nur noch aktivieren. Wählen Sie dazu in der Menüleiste *Formular* → *Aktivieren*. Sie erhalten eine Meldung vom System, ob die Aktivierung erfolgreich war oder nicht.

Versuchen Sie trotz fehlerhafter Prüfung oder ohne Prüfung, Ihr Formular zu aktivieren, wird der Aktivierungslauf abgebrochen. Das Formular ist für den Endbenutzer nicht verfügbar.

Auch nach der Aktivierung können Sie Formulare weiterbearbeiten und beispielsweise ändern, übersetzen oder löschen.

Wenn Sie in der Menüleiste *Formular -> Löschen* wählen, wird ein Dialogfenster angezeigt. Hier werden Sie gefragt, ob Sie Ihr Formular wirklich in allen Sprachen löschen möchten. Positionieren Sie den Cursor auf der entsprechenden Antwort, und bestätigen Sie mit `ENTER` oder `F2`. Nach dem Löschen kehren Sie wieder zum Anforderungsbild zurück. In der Statusleiste wird der Status der Verarbeitung angezeigt.

Transporting, Copying, and Comparing Styles and Forms

SAPscript offers comprehensive tools for working with styles and forms. The procedure used with these tools is the same for styles and forms.

The only distinction which needs to be made is where the individual tools can be used. There are tools that can be used directly in the style or form request screen. Others can only be used once the style or form has been selected.

Tools available in the request screen are:

- Compare clients
- Copy from client

Tools in the selected style or form are:

- Display style and form information
- Print style and form information
- Display versions

This section also describes the transport tools available for managing and moving documents, styles, and forms between SAP Systems.

Comparing Client Contents [Seite 97]

Copying Client Contents [Seite 99]

Displaying Style and Form Information [Seite 100]

Displaying Versions [Seite 101]

Comparing Client Contents

Choose *Compare clients*

- To check whether a particular standard text, style, or form already exists in another client.
- To identify differences in the object attributes and definitions.

You can mark single objects for a comparison, or generate comparisons of a whole set of objects.

A comparison covers only the attributes and, if applicable, the formal definition of an object. A comparison does not extend to the text level of an object. A comparison can highlight differences in the author of an object or the formats used in a style or form. But a comparison cannot detect differences in standard texts nor in the texts used in forms.

To compare clients,

1. Choose *Tools* → *Word processing* and then → *Standard text,* → *Styles*, or → *Forms*, depending upon the type of object you want to compare. Then choose *Utilities* → *Compare clients*.

   This takes you to the selection screen for determining which objects you wish to compare.

2. Make the following entries:

– *Name*

  Enter the name of the standard text, style, or form for which the check is to be carried out. You can enter a generic name such as ST* or *.

– *Language*

  Enter the appropriate language key.

– *Text ID*

  Enter the text ID (type) of the text(s) you have selected. (This selection appears only if you are comparing standard texts.)

– *Source client*

  Specify the client, on the basis of which you want the check to be carried out. By default, this is client 000.

– *Target client*

  Specify the client You want to check against the source client. By default, this is the client in which you are currently active.

3. Execute the client comparison.

   When the comparison is completed, the system displays a list with the source client in the left-hand column and the target client in the right-hand column. An empty line in either column indicates that the object does not exist in the corresponding client.

Click on an object in the list to display detailed comparison information. The list highlights attributes that differ. It also hightlights differences between the contents of the objects. For example, differences in the character and paragraph styles in two forms of the same name are highlighted.

Choose *Info* to display detailed information on one of the objects in the comparison list, without reference to the comparison object.

Copying Client Contents

To copy one or more standard texts, forms, or styles from one client to another client, choose *Copy from client*.

It is not possible to copy styles or forms to client 000.

To copy an object or set of objects from one client to another,

1. Choose *Tools* → *Word processing* and then → *Standard text,* → *Styles*, or → *Forms*, depending upon the type of object you want to copy. Then choose *Utilities* → *Copy from client*.

   This takes you to the selection screen for determining which objects you wish to copy.

2. Make the following entries:

– *Name*

Enter the name of the standard text, style, or form for which the check is to be carried out. You can enter a generic name such as ST* or *.

– *Language*

Enter the appropriate language key.

– *Text ID*

Enter the text ID (type) of the text(s) you have selected. (This selection appears only if you are copying standard texts.)

– *Source client*

Specify the client, on the basis of which you want the check to be carried out. By default, this is client 000.

– *Target client*

Specify the client you want to check against the source client. By default, this is the client in which you are currently active.

– *Action log*

Enter the following values:

X	to create an action log.
No entry	not to create an action log.

3. Execute the client copy.

   When copying is completed, the system displays a log if you have requested this.

Displaying Style and Form Information

In order to display detailed information on a style or form,

1. Call the style or form you are interested in in the SAPscript editor.
2. Choose *Utilities* → *Form info* or *Utilities* → *Style info*.

   The information is displayed in a list. To edit the list, choose *System* → *Lists.*

Displaying Versions

Another tool that helps you maintain styles and forms allows you to display the versions of a style or form. This function is only active when you edit the selected style or form.

To display the versions of a style or form,

1. Choose *Utilities* →*Versions*.

   This takes you to a list in which the following information is listed:

– Version

  Contains information about the status of the style or form.

– Language

  Contains information about the current language key.

– Original language

  Contains information about the language in which the style or form was originally created.

– Description

  Describes the style or form; it is a short description taken from the header data.

2. Choose *Style info*.

   This takes you to a list with comprehensive information about a specific style or form.

Transporting Documents, Styles, and Forms

SAPscript offers both a connection to the SAP System's Workbench Organizer (transport system) and also interactive tools for moving documents, styles, and forms.

As of Release 3.0A, the SAP transport system is fully activated for SAPscript styles and forms. That is, styles and forms that you create or modify are correctly transported from your development client. Imports into a target system are by default always into client 000. From there, SAPscript styles and forms are available in all clients of a target system.

This means that you no longer need to use the interactive reports to transport your styles and forms. Instead, you can rely on the Workbench Organizer for managing and moving your SAPscript objects. The interactive transport reports are still documented in this section, should you need them.

Transport and Languages

As of Release 3.0, both the Workbench Organizer and SAPscript's interactive transport tools use the SAP "language vector" to determine which languages may be imported or exported.

The language vector is the set of languages that are installed in an SAP System. The language vector is determined by the languages recorded in the SAP profile parameter zcsa/installed_languages.

Under the new mechanism, only documents, styles, and forms whose source language is contained in the language vector can be imported or exported. If the source language of a SAPscript object is not found in the language vector, then the Workbench Organizer issues an warning message (return code 4). The object is not exported from or imported into the affected SAP System.

You can override the language vector in the SAPscript online transport reports, RSTXR3TR and RSTXSCRP. For more information, see the report documentation on these two reports.

Transport and Standard Texts

The SAPscript editor is used for text processing in various SAP applications. For example, you can access SAPscript directly for editing texts by way of *Tools → Text processing → Standard texts*. SAPscript is also used in the development environment for creating documentation and can be used in the SAP mail system.

All texts that you create with SAPscript are transportable. However, SAPscript is not connected to the SAP Workbench Organizer in all of the applications that use SAPscript's services.

If you are working on texts that you wish to transport, then follow these guidelines:

- If the SAP System presents the Workbench Organizer's dialog box when you first save your document, then you can transport your document using the normal Workbench Organizer release and transport procedures. The application in which you are working has connected SAPscript to the organizer, and there are no special actions you need to take to transport your documents.
- If no organizer dialog box is presented when you first save your document, then your document has not been automatically recorded in the Workbench Organizer.

  If you wish to transport your document, then you can do so by entering the document by hand in a transport request in the Workbench Organizer. The entry that you need to make is:

PgmID	Obj	Object name
R3TR	TEXT	<NAME OF YOUR DOCUMENT>

You can then follow normal organizer procedures to transport your document.

SAPscript Control Commands

The functionality of the SAPscript editor is made available through a set of commands. These commands give you full editing control over your text. They are executed immediately when called.

There is, however, another kind of SAPscript command, namely the control commands. The purpose of these is to allow control of the output formatting. These commands are not interpreted by the SAPscript editor, but are passed through to the SAPscript Composer for processing. The Composer is the program that converts text from the form displayed in the editor to the form used for printing. This includes, for example, line and page formatting, the replacement of symbols with their current values and the formatting of text according to the paragraph and character formats specified.

The SAPscript control commands are described in the following sections.

Syntax of Control Commands

SAPscript control commands are entered and edited in the text editor in the same way as a normal line of text. They do, however, differ from normal lines of text:

- Enter the paragraph format **/:** in the format column to identify a control command.
- Enter the command itself in the text line. You will notice that all key words and other parts of the specification not given as literal values enclosed in inverted commas are automatically converted to capital letters.
- Make sure that a control command, together with any parameters it requires, does not occupy more than a single line.
- Enter only one control command in each line.
- Note that the editor formatting has no effect on lines containing control commands.

If a control command is unknown or it contains syntax errors, the line containing it is treated as a comment line. It is neither interpreted nor printed.

Explicit Page Break: NEW-PAGE

SAPscript automatically inserts a page break when the main window of a page (MAIN) is full. You can use the NEW-PAGE command to force a page break in the text at any point you want one. The text following this command then appears on a new page. The page break is always performed (it is an unconditional page break).

The NEW-PAGE command completes the current page. This means that all the windows that are still on the page are printed immediately. If you use the NEW-PAGE command without parameters, the page defined in the current form as the next page will be taken next. If, however, your form contains a number of different pages, then you can specify any one of these as the next page to be used.

Syntax:

```
/: NEW-PAGE [page_name]
```

```
/: NEW-PAGE
```

The current page will be completed and the text in the following lines will be written to the page specified in the form.

```
/: NEW-PAGE S1
```

As above, except that the page S1 will be taken as the next page.

- If, in a NEW-PAGE command, you specify a page not contained in the form, the specification is ignored.
- Take care that there are no blank lines immediately before a NEW-PAGE command. If an implicit page break occurs within the blank lines, an unexpected blank page may be printed.

Preventing Page Breaks: PROTECT

You can specify, either in the style or in the form, that a particular paragraph should not be split in two by a page break. If this page protect attribute is set, then the complete paragraph is always printed on one page. This property applies only to that particular paragraph.

This attribute is not intended to be used to protect all paragraphs against a page break. The point is that a page break is by its very nature a dynamic event and the exact point at which it occurs depends on the current state (length and contents) of the preceding text. It is also possible that you may want to protect only certain parts of a paragraph against a page break. One way to achieve this is to use the NEW-PAGE command immediately before the text concerned starts. Explicitly beginning a new page at this point should ensure that a further page break does not occur within the text. However, this technique is not change-friendly. For example, you format your text with the help of the NEW-PAGE command so that no page breaks occur where they should not. At a later time, you insert or delete some lines. These changes cause all the subsequent text to be moved relative to the printed page, and you must check each NEW-PAGE command you previously inserted to see if it is still in the correct place.

To allow you to define the areas to be protected against a page break on an individual basis, SAPscript provides the PROTECT.. ENDPROTECT command pair. If you enclose the text to be protected in these commands, then SAPscript will ensure that each line of this text is printed together on the same page. If the complete text fits in the space remaining on the current page, then it is printed on this page just as it would be if no PROTECT command had been used. If, however, the remaining space is not sufficient for the text, then the PROTECT command has the same effect as a NEW-PAGE command and text is printed on a new page.

Thus the PROTECT/ENDPROTECT commands may be regarded as a kind of conditional NEW-PAGE command, the condition being whether or not the lines enclosed between the two commands fit in the space remaining in the current main window.

Syntax:

```
/: PROTECT
      :
      :
/: ENDPROTECT
```

The text lines to be protected are enclosed between the two commands.

- An ENDPROTECT command without a preceding PROTECT command has no effect.
- If the terminating ENDPROTECT is missing, SAPscript assumes it at the end of the text.
- PROTECT.. ENDPROTECT command pairs cannot be nested. If a second PROTECT command occurs before the first one has been terminated by an ENDPROTECT, it is ignored.
- If the text enclosed by a PROTECT.. ENDPROTECT pair is itself too long for a single page, then a page break is generated immediately before the text and the text is printed in the normal way. It is then unavoidable that a page break will occur at some point within the text.

Next Main Window: NEW-WINDOW

Each page can consist of up to 99 main windows. Each main window is assigned a consecutive identifying number (0..98), and the windows are filled in this order. This feature enables SAPscript to print labels and to output multi-column text. When one main window fills up, the next main window on that page is taken, if there is a next one. A page break is inserted after the last main window.

You can use the NEW-WINDOW command to call the next main window explicitly, even if the current main window is not yet full. If you are in the last main window of the page, the command has the same effect as the NEW-PAGE command.

Syntax:

```
/: NEW-WINDOW
```

Assigning a Value to a Text Symbol: DEFINE

Text symbols acquire their values as a result of explicit assignment. To interactively assign text symbols, in the text editor choose *Include* → *Symbols* → *Text*. This method is available for all text symbols belonging to a text module as well as those of the associated form.

Values defined in this way are lost when the transaction is left. If you want to print the text module again, then you must enter the symbol values again. The purpose of the DEFINE command is to provide a means of making this value assignment a permanent part of the text, so that the values are available again when the text module is called again. This command can also be used to re-assign a new value to a text symbol half-way through the text.

Syntax:

```
/: DEFINE &symbol_name& = 'value'
```

```
/: DEFINE &subject& = 'Your letter of 7/3/95'
```

The value assigned can have a maximal length of 60 characters. It may itself contain other symbols. A symbol contained within the value assigned to another symbol is not replaced with its own value at the point at which the DEFINE command is executed. Rather, this replacement is made when the symbol defined in the DEFINE command is called in the text.

```
/: DEFINE &symbol1& = 'mail'
/: DEFINE &symbol2& = 'SAP&symbol1&'
/: DEFINE &symbol1& = 'script'
   &symbol2&     ->  SAPscript
```

If, however, the DEFINE command is written using the ':=' character rather than the '=' character, then any symbol contained within the value being assigned is replaced immediately with its current value. The assignment to the target symbol is made only after all symbols in the value string are replaced with their values. The total length of the value string may not exceed 80 characters. The target symbol must be a text symbol, as before.

```
/: DEFINE &symbol1& = 'mail'
/: DEFINE &symbol2& := 'SAP&symbol1&'
/: DEFINE &symbol1& = 'script'
   &symbol2&     ->  SAPmail
```

Formatting Date Fields: SET DATE MASK

To define the formatting of date fields, use the SET DATE MASK control command. Executing this command causes all subsequent date fields to be printed using the specified format.

Syntax:

```
/: SET DATE MASK = 'date_mask'
```

In the date mask, you can use the following codes:

- DD: day (two digits)
- DDD: day name - abbreviated
- DDDD: day name - written out in full
- MM: month (two digits)
- MMM: month name - abbreviated
- MMMM: month name - written out in full
- YY: year (two digits)
- YYYY: year (four digits)
- LD: day (formatted as for the L option)
- LM: month (formatted as for the L option)
- LY: year (formatted as for the L option)

All other characters found in a date mask are interpreted as simple text and are copied straight into the output.

Assuming the current system date is March 1st, 1997.

```
/: SET DATE MASK = 'Foster City, MM/DD/YY'
&DATE&     -> Foster City, 03/01/97
```

```
/: SET DATE MASK = 'MMMM DD, YYYY'
&DATE&     -> March 01, 1997
```

The date mask may be reset to the default setting by using an empty string:

```
/: SET DATE MASK = ' '
```

The abbreviated and full forms of the names of the days and months are stored in the language dependent TTDTG table under the following keys:

- %%SAPSCRIPT_DDD_dd: abbreviated day name
- %%SAPSCRIPT_DDDD_dd: full form of day name
- %%SAPSCRIPT_MMM_mm: abbreviated month name
- %%SAPSCRIPT_MMMM_mm: full form of month name

dd: day number 01 = Monday,..., 07 = Sunday mm: month number 01 = January,..., 12 = December

Formatting Time Fields: SET TIME MASK

To format time fields to your needs, use the SET TIME MASK control command. Executing this command causes all subsequent time fields to be printed using the specified format.

Syntax:

```
/: SET TIME MASK = 'time_mask'
```

In the time mask, you can use the following codes:

- HH hours (two digits)
- MM minutes (two digits)
- SS seconds (two digits)

All other characters found in a time mask are interpreted as simple text and are copied straight into the output.

Assuming the current time is 10:08:12,

```
/: SET TIME MASK = 'HH:MM'
   &TIME&      -> 10:08
```

```
/: SET TIME MASK = 'HH hours MM minutes'
   &TIME&      -> 10 hours 08 minutes
```

The time mask may be reset to the default setting by using an empty string:

```
/: SET TIME MASK = ' '
```

Country-Dependent Formatting: SET COUNTRY

The formatting for certain field types depends on the country settings. These field types include, for example, date fields and number fields that include either a decimal point or the 'thousands' separator character. The formatting options defined in the user master record are usually the ones used here. To choose a formatting option other than the one specified in the user master record, use the SET COUNTRY control command. The country-dependent formatting options are stored in the T005X table.

Syntax:

```
/: SET COUNTRY country_key
```

You can enter the country key either as a literal value enclosed in quotes or as a symbol.

```
/: SET COUNTRY 'CAN'
/: SET COUNTRY &country_key&
```

Use a blank country name to revert to the setting found in the user master record:

```
/: SET COUNTRY ' '
```

This SAPscript command actually calls the corresponding ABAP command internally. This guarantees the effect of the SAPscript command to be identical with that of the ABAP command.

If the formatting turns out not to be as required, then you should check the settings in table T005X.

Position of the Leading Sign: SET SIGN

The usual convention in business applications is to show the leading sign to the right of the figure to which it applies. However, it is sometimes necessary to show the leading sign to the left of the figure. To set the sign explicitly, use the SET SIGN control command. Executing this command affects the formatting of all subsequent program symbols that possess a leading sign.

Syntax:

```
/: SET SIGN LEFT
```

The leading sign appears to the left of the number.

```
/: SET SIGN RIGHT
```

The leading sign appears to the right of the number.

Initializing Numbered Paragraphs: RESET

To reset the numbering of an outline paragraph to its initial value, use the RESET control command. If you do not use the RESET command, then the numbering of all outline paragraphs in a text is continuous. If you specify the name of an outline paragraph in the RESET command, then its paragraph numbering and that of subordinate paragraphs is reinitialized.

Syntax:

```
/: RESET paragraph_format
```

The paragraph format specifies the outline paragraph to be reset.

Assume that the paragraph N1 is defined in the style you are using. This kind of paragraph is intended for enumerated lists and causes a sequential number to be printed.

```
*  Proceed as follows if you want to work with the SAP R/3
   system:
N1 Ensure that you have a PC
N1 Switch the PC on
N1 Click on the SAP icon using the mouse.
*  You will then enter the SAP logon screen. In order to log
   on here, you must carry out the following actions:
/: RESET N1
N1 Enter your user name
N1 Enter your password
N1 Select the application you want to use
```

This text specification would be output as follows:

Proceed as follows if you want to work with the SAP R/3
system:

1. Ensure that you have a PC
2. Switch the PC on
3. Click on the SAP icon using the mouse.

You will then enter the SAP logon screen. In order to log on here, you must carry out the following actions:

1. Enter your user name
2. Enter your password
3. Select the application you want to use

If there is no RESET command between the two sections, then the two lists would be numbered in a single sequence:

Proceed as follows if you want to work with the SAP R/3
system:

1. Ensure that you have a PC
2. Switch the PC on

3. Click on the SAP icon using the mouse.

You will then enter the SAP logon screen. In order to log on here, you must carry out the following actions:

4. Enter your user name
5. Enter your password
6. Select the application you want to use

Including Other Texts: INCLUDE

To include the contents of another text into the current text, use the INCLUDE control command. SAPscript still treats the text to be included as a separate text. The text is copied over only at the point at which the output is formatted.

Thus the use of the INCLUDE command always ensures that the most current version of a text is included into the output, since the text is not read and inserted until the output is formatted.

Syntax:

```
/: INCLUDE name [OBJECT o] [ID i] [LANGUAGE l] [PARAGRAPH p]
         [NEW-PARAGRAPH np]
```

You must specify the name of the text to be inserted. It can be up to 70 characters long. If the name of the text contains spaces, then you must enclose it in quotes as a literal value. You can, alternatively, specify the name via a symbol. All remaining parameters in the INCLUDE command are optional. If an optional parameter is not specified, then SAPscript uses default values as applicable for the calling environment.

```
/: INCLUDE  MYTEXT
```

The text MYTEXT is included in the language of the calling text.

```
/: INCLUDE MYTEXT LANGUAGE 'E' PARAGRAPH 'A1'
```

The text with the name MYTEXT and the language E is included, regardless of the language of the calling text. The paragraph format A1 will be used as the standard paragraph type for this call.

Optional parameters:

- LANGUAGE

  If this parameter is not specified, then the language of the calling text or the form language are used for the text to be included. If the language is specified, then the text will be fetched in this language, regardless of the language of the calling text.

- PARAGRAPH

  The text to be included is formatted using the style allocated. The PARAGRAPH parameter can be used to redefine the standard paragraph for this style for the current call. All *-paragraphs in the included text will then be formatted using the paragraph specified here.

- NEW-PARAGRAPH

  The first line of the text to be included will be given this format indicator, as long as it is not a comment or command line. If the optional PARAGRAPH parameter (see above) is not specified, then all *-paragraphs of the included text will also be formatted using the paragraph specified in the NEW-PARAGRAPH command.

- OBJECT

  In order to completely specify a text, information about the text object is also required. There are a number of restrictions and other rules that depend on the object type of the calling text:

  - Any kind of text can be included in a form. If no object is specified, then TEXT will be used (standard texts).
  - In the case of a document text (DOKU object), you can include only document texts. This object type is also assumed if no object is specified in this environment.
  - Only hypertexts and document texts can be included in a hypertext (DSYS object). If the OBJECT parameter is missing, then DSYS is used as the default value.
  - In the other kinds of text you can include only standard texts (TEXT object), document texts or hypertexts. If there is no specification, then the default object is TEXT.

- ID

  The text ID is a part of the text key, which permits further text objects within a given object. If no ID is specified, then the default include ID is used from the TTXID table for the calling text. If there is no entry in this table, then the text ID of the calling text is used.

  The following consistency check is applied both to the ID and the object:

- All text IDs are allowed in a form.
- In document texts, only document texts may be included that have text IDs TX (general texts) or UO (authorization objects) and also other document texts with the same text ID as the calling document text.
- In DSYS texts, all DSYS texts can be included, whatever ID they have. Document texts to be included must have one of the IDs TX or UO.
- Into the other texts, standard texts with any allowable text ID, DSYS texts with all IDs, and document texts with the IDs TX and UO can be included.

The INCLUDE command returns a status code in the SAPSCRIPT-SUBRC symbol:

- 0: the text include was successful.
- 1: the command could not be executed because it contained syntax errors.
- 2: the rules governing the text to be included were not followed (see above).

  This value cannot occur if the command is used in a SAPscript form.
- 4: the specified text could not be found.

Changing the Style: STYLE

The STYLE control command allows you to change the style within a text. The new style is in force until another STYLE command is issued. If you specify * as the name of the style, then the system reverts to the default paragraph of the original style or form.

Syntax:

```
/: STYLE style [DOMINANT]
```

```
/: STYLE *
```

A style set with this command has no effect in a text included with INCLUDE. The system takes the paragraph and character formats from the calling text. To use the style set with STYLE in the INCLUDE text as well, you must add DOMINANT to the command.

If the INCLUDE text has a style assigned to it, in both cases, the system always takes the paragraph and character formats from the directly assigned style.

Formatting Addresses: ADDRESS

The ADDRESS - ENDADDRESS control command formats an address according to the postal convention of the recipient country defined in the COUNTRY parameter. The reference fields are described in the structures ADRS1, ADRS2, or ADRS3, depending on the type of address. Either direct values or symbols may be assigned to the parameters.

Syntax:

```
/: ADDRESS [DELIVERY] [TYPE t] [PARAGRAPH a] [PRIORITY p] [LINES l]
/: TITLE title
/: NAME name1[,name2[,name3[,name4]]]
/: PERSON name of natural person [TITLE form of address]
/: PERSONNUMBER number of the personen
/: DEPARTMENT department
/: STREET street name HOUSE house number
/: LOCATION additional location information
/: POBOX po box [CODE post code / zip code] [CITY city]
/: POSTCODE post code / zip_code
/: CITY city1[,city2]
/: NO_UPPERCASE_FOR_CITY
/: REGION county / state
/: COUNTRY recipient country [LANGUAGE language code]
/: COUNTRY_IN_REC_LANG
/: LANG_FOR_COUNTRY language key
/: FROMCOUNTRY sender country
/: ADDRESSNUMBER address number
/: ENDADDRESS
```

The parameter values contain both formatting and address information. The address data are formatted for output according to the data in the following parameters:

- TYPE
- FROMCOUNTRY
- COUNTRY
- LANGUAGE
- PRIORITY
- DELIVERY
- LINES

For more information, see the documentation for the SAP function module ADDRESS_INTO_PRINTFORM.

If DELIVERY is not specified and if a POBOX is specified, then the POBOX is used in an address instead of a STREET.

Parameters

- DELIVERY

  Means that the address should be formatted as a complete delivery address, using the street name and number rather than the P.O. Box.

- TYPE

  Specifies the type of address. The following types are possible:

  1. Normal address (ADRS1). This is the address of a company or organization. It corresponds to the address structure that is expected in most SAP applications.
  2. Private or personal address (ADRS2). This is the address of a natural person, a private or home address.
  3. Company addressDienstadresse (ADRS3) with contact person. This is the address of a colleague or contact within a company or organization. The company name should be specified in the TITLE and NAME fields; the ATTN: contact person should be named in PERSON and TITLE.

  Should you enter another address type or leave the field blank, then type 1 is used for formatting.

- PARAGRAPH

  Specifies the paragraph format to be used for outputting the address. If this parameter is not given, the address will be output using the default paragraph format.

- PRIORITY

  Specifies which of the address lines may be omitted should this be necessary. Any combination of the following codes may be specified. The order in which you list the codes determines the order in which address lines are left out.

  The codes are as follows:

A	Title
P	Mandatory empty line
4	Name4
3	Name3
R	Region
T	Neighborhood, administrative section of a city (CITY2)l
D	Department
L	Country name
C	Post code or zip code
2	Name2
B	P.O. Box (Japan only)
S	Street name and number or P.O. Box, depending upon DELIVERY parameter
N	Name and form of address of natural person (PERSON and TITLE)
I	Location information in LOCATION
O	City

- LINES

  Specifies how many lines may be used for formatting the address. If there are too few lines available to allow all the address data to be formatted, then the data specified in the

PRIORITY parameter are omitted. If there is no LINES parameter and if this command is in a form window of a type other than MAIN, then the number of lines available for formatting the address are automatically calculated based on the current output position and the size of the window.

- TITLE

  Title or form of address. Used only with addresses of types 1 and 3.

- NAME

  Up to four names may be given, separated by commas. Used only with addresses of types 1 and 3.

- PERSON

  Name of the addressee. Used only for addresses of type 2 (private or personal address) or type 3 (company contact address). In type 3 addresses, use PERSON for the name of your contact person: 'Attn: Mr. Jeffries'. The name fields should be used for the company address.

- PERSONNUMBER

  Personal number. Can be used only for address types 2 or 3 (private or personal address).

- TITLE (with PERSON)

  Title of the addressee. Can be used only for address types 2 or type 3 (private or personal address).

- DEPARTMENT

  Department of the addressee. Can be used only for address type 3 (company address).

- STREET

  Street name.

- HOUSE

  House number for the corresponding street.

- LOCATION

  Additional location information, such as the building, "Upstairs Apartment" and so on. Appears on its own line in the address.

- POBOX

  P. O. Box

- CODE

  The post code / zip code of the P. O. Box if this differs from the post code / zip code of the recipient.

- CITY

  The city in which the destination P.O. Box is located if this differs from the city of the recipient.

- POSTCODE

Post code / zip code of the recipient.

- CITY

  Addressee’s city. city1 is expected to be the city; city2 is the neighborhood or administrative section, if required.

- NO_UPPERCASE_FOR_CITY

  Default = **NO_UPPERCASE_FOR_CITY ‘ ‘**
  Usually, the system prints the city and country names of foreign addresses in uppercase (**NO_UPPERCASE_FOR_CITY ‘ ‘**).
  You can use this parameter to force the system to output city and country name unchanged (uppercase/lowercase).
  (**NO_UPPERCASE_FOR_CITY ‘X’**)

- REGION

  This allows an administrative region, county, province, or state etc. to be specified.

- COUNTRY

  Specifies the recipient country, i.e. the country according to whose postal conventions the address is to be formatted.

- COUNTRY_IN_REC_LANG

  This flag tells the system to use the recipient language for the country name.
  (**COUNTRY_IN_REC_LANG ‘X‘**)

  (Default: Recipient language is not used: **COUNTRY_IN_REC_LANG ‘ ‘**)

- LANG_FOR_COUNTRY

  Default = Space

  Use this parameter to explicitly set the language in which to print the country name of a foreign address. By default, the system uses the language of the sending country.

- LANGUAGE

  Language code of the language of the recipient country, if it differs from that of the recipient COUNTRY. Example: addresses in Switzerland. Standard SAP language codes are used; you can display these in the initial SAPscript text processing screen or in table T002.

- FROMCOUNTRY

  Specifies the language to be used for formatting the name of the recipient country. For most European countries, the recipient country is specified by placing the international car registration letters in front of the post code and separating them from the post code with a hyphen. You must always specify the sender country.

- ADDRESSNUMBER

  The number is used to index a central address file, from which the desired address is read instead of using the set of the above fields. You can find more information on this facility in the documentation for the function module ADDRESS_INTO_PRINTFORM.

  ⚠

  You use this one parameter instead of the set of parameters described before.

Formatting Addresses: ADDRESS

```
/: ADDRESS
/: TITLE 'Firma'
/: NAME 'Schneider & Co', 'Finanzberatung'
/: STREET 'Kapitalgasse 33'
/: POBOX '12345' CODE '68499'
/: POSTCODE '68309'
/: CITY 'Mannheim'
/: COUNTRY 'DE'
/: FROMCOUNTRY 'DE'
/: ENDADDRESS
```

This produces the following output address:

Firma
Schneider & Co
Finanzberatung
Postfach 12345

68499 Mannheim

If the DELIVERY parameter is specified on the ADDRESS command, then the street name and number will appear in the address in place of the P. O. Box number.

Firma
Schneider & Co
Finanzberatung
Kapitalgasse 33

68309 Mannheim

SAPscript makes an internal call to the ADDRESS_INTO_PRINTFORM function module for formatting the address. If the result is not as expected, you should check the settings for this function module (see the function module documentation).

Setting a Header Text in the Main Window: TOP

You can use the TOP.. ENDTOP control command to specify lines of text that you want to print always at the top of the main window. These text lines are also known as header texts. For example, you would use header texts in the case of a very long table covering several pages of output to ensure that the table heading information were repeated at the start of each new page of output.

Syntax:

```
/: TOP
      :
      :
/: ENDTOP
```

The lines of text enclosed between the two control commands will be output from now on at the start of the main window.

An existing header text can be disabled by using the TOP.. ENDTOP command pair without enclosing any text lines between the two command lines:

```
/: TOP
/: ENDTOP
```

Subsequent pages will contain no header text.

- If the main window already contains some output then a newly specified header text takes effect on the next page only.
- The same applies to the deletion of a header text. If a header text has already been output on the current page then it cannot be retracted.
- Header texts should not be employed in texts that are printed from applications programs, such as reminder texts, order texts. These applications programs can also work with header texts via the form interface, which may lead to unexpected results.

Setting a Footer Text in the Main Window: BOTTOM

You can specify footer texts for the main window in a similar way to header texts. Footer texts are always printed at the bottom of the window.

Syntax:

```
/: BOTTOM
:
:
/: ENDBOTTOM
```

The lines of text enclosed between the two control commands will be output from now on at the bottom of the main window.

An existing footer text can be disabled by using the BOTTOM.. ENDBOTTOM command pair without enclosing any text lines between the two command lines:

```
/: BOTTOM
/: ENDBOTTOM
```

This and subsequent pages will contain no footer text.

- Assuming there is still sufficient space in the main window, a newly specified footer text will also be printed on the current page.
- Footer texts should not be employed in texts that are printed from applications programs, such as reminder texts, order texts. These applications programs can also work with footer texts via the form interface, which may lead to unexpected results.

Conditional Text: IF

You can use the IF control command to specify that text lines should be printed only when certain conditions are met. If the logical expression contained within the IF command is true, then the text lines enclosed by the IF... ENDIF command pair are printed. Otherwise they are ignored.

Syntax:

```
/: IF condition
       :
       :
/: ENDIF
```

The logical expression can use the following comparison operators:

=	EQ	equal to
<	LT	less than
>	GT	greater than
<=	LE	less than or equal to
>=	GE	greater than or equal to
<>	NE	not equal to

The following logical operators can be used to combine conditions:

- NOT
- AND
- OR

Evaluation of both the individual logical expressions and of the combinations of expressions is performed strictly from left to right. There are no precedence rules. Bracketed expressions are not supported.

The comparison is always performed on literal values, that is, the symbols are formatted as character strings before they are compared. This is particularly significant in the case of program symbols, because the formatting of these may depend on various parameters. For example, the formatted form of a currency field employs a variable number of decimal places and a variable ‘decimal point’ symbol (a period or a comma) depending on the applicable currency key.

You can extend the IF command with the ELSE command to allow text lines to be specified that you want to print in case the condition is false. If the condition is true, the text lines enclosed by the IF and ELSE commands are formatted; otherwise the text lines enclosed by the ELSE and ENDIF commands are formatted.

Syntax:

```
/: IF condition
       :
/: ELSE
       :
/: ENDIF
```

The ELSEIF command allows you to specify multiple cases.

Syntax:

Conditional Text: IF

```
/: IF condition
      :
/: ELSEIF condition
      :
/: ELSE
      :
/: ENDIF
```

You can use any number of ELSEIF commands within one compound IF.. ENDIF control command. The use of an ELSE command is then optional.

- You must not extend a condition over more than one line. Both the IF or ELSEIF command and the attached condition must be completely contained within a single line.
- You can nest IF commands.
- You must terminate an IF command with an ENDIF command. If you forget this, there will be no more output following the IF command if the condition is false.
- If a syntax error occurs in the interpretation of this command, then the command is not executed. This may have an unexpected effect on the subsequent text output. For example, if the IF statement is incorrect, then all following ELSEIF and ELSE commands will be ignored, since the opening IF command is ‘missing’. This will cause all the text lines attached to the ELSEIF and ELSE commands to be printed.

Finding a Match: CASE

The CASE command covers a special case of the multiple case IF command. Rather than allowing an arbitrary condition to be tested in each of the individual cases (as in the general multiple case IF command), the CASE command allows a given symbol to be tested against specific values until a matching value is found.

Syntax:

```
/: CASE symbol
/: WHEN value1
       :
/: WHEN value2
       :
/: WHEN valuen
       :
/: WHEN OTHERS.
       :
/: ENDCASE
```

The symbol in the CASE line is formatted. If its value is found in one of the WHEN lines, then the text lines following this WHEN line are printed. If no matching value is found then the text lines enclosed by the WHEN OTHERS line and the ENDCASE command are printed. The WHEN OTHERS section is optional.

As with the IF command, the comparison is always performed on literal values.

- A CASE command must be terminated by an ENDCASE command.
- The WHEN OTHERS section is optional.

Calling ABAP Subroutines: PERFORM

You can use the PERFORM command to call an ABAP subroutine (form) from any program, subject to the normal ABAP runtime authorization checking. You can use such calls to subroutines for carrying out calculations, for obtaining data from the database that is needed at display or print time, for formatting data, and so on.

PERFORM commands, like all control commands, are executed when a document is formatted for display or printing. Communication between a subroutine that you call and the document is by way of symbols whose values are set in the subroutine.

> The system does not execute the PERFORM command within SAPscript replace modules, such as TEXT_SYMBOL_REPLACE or TEXT_INCLUDE_REPLACE. The replace modules can only replace symbol values or resolve include texts, but not interpret SAPscript control commands.

Syntax in a form window:

```
/: PERFORM <form> IN PROGRAM <prog>
/: USING &INVAR1&
/: USING &INVAR2&
......
/: CHANGING &OUTVAR1&
/: CHANGING &OUTVAR2&
......
/: ENDPERFORM
```

INVAR1 and **INVAR2** are variable symbols and may be of any of the four SAPscript symbol types.

OUTVAR1 and **OUTVAR2** are local text symbols and must therefore be character strings.

The ABAP subroutine called via the command line stated above must be defined in the ABAP report **prog** as follows:

```
FORM <form> TABLES IN_TAB STRUCTURE ITCSY
                   OUT_TAB STRUCTURE ITCSY.

...
ENDFORM.
```

The values of the SAPscript symbols passed with **/: USING...** are now stored in the internal table **IN_TAB**. Note that the system passes the values as character string to the subroutine, since the field Feld VALUE in structure ITCSY has the domain TDSYMVALUE (CHAR 80). See the example below on how to access the variables.

The internal table **OUT_TAB** contains names and values of the CHANGING parameters in the PERFORM statement. These parameters are local text symbols, that is, character fields. See the example below on how to return the variables within the subroutine.

> From within a SAPscript form, a subroutine GET_BARCODE in the ABAP program QCJPERFO is called. Then the simple barcode contained there (‘First page’, ‘Next page’, ‘Last page’) is printed as local variable symbol.
>
> **Definition in the SAPscript form:**

```
/: PERFORM GET_BARCODE IN PROGRAM QCJPERFO
/: USING &PAGE&
/: USING &NEXTPAGE&
/: CHANGING &BARCODE&
/: ENDPERFORM
/

/  &BARCODE&
```

Coding of the calling ABAP program:

```
REPORT QCJPERFO.


FORM GET_BARCODE TABLES IN_PAR     STUCTURE ITCSY
                                   OUT_PAR STRUCTURE ITCSY.

DATA: PAGNUM     LIKE SY-TABIX,                 “page number
      NEXTPAGE LIKE SY-TABIX.                   “number of next page

   READ TABLE IN_PAR WITH KEY ‘PAGE’.
   CHECK SY-SUBRC = 0.
   PAGNUM = IN_PAR-VALUE.

   READ TABLE IN_PAR WITH KEY ‘NEXTPAGE’.
   CHECK SY-SUBRC = 0.
   NEXTPAGE = IN_PAR-VALUE.

   READ TABLE OUT_PAR WITH KEY ‘BARCODE’.
   CHECK SY-SUBRC = 0.
   IF PAGNUM = 1.
         OUT_PAR-VALUE = ‘|’.                 “First page
   ELSE.
         OUT_PAR-VALUE = ‘||’.                “Next page
   ENDIF.

   IF NEXTPAGE = 0.
         OUT_PAR-VALUE+2 = ‘L’.               “Flag: last page

   ENDIF.

   MODIFY OUT_PAR INDEX SY-TABIX.


ENDFORM.
```

Inserting Print Controls: PRINT-CONTROL

You can use this command to call certain printer functions from a SAPscript text. Although you cannot enter control characters for the printer directly in your text, you can define a print control via the spool maintenance transaction SPAD that contains the printer commands you want. You can then call a print control using the PRINT-CONTROL SAPscript command.

Syntax:

```
/: PRINT-CONTROL name
```

Specify the name of the print control either with or without inverted commas.

The print control printed via PRINT-CONTROL should always be a print control created in the customer name area (Zxxx) for the device type used. Use no Sxxx print controls, since their contents may be changed by SAP at any time with a release or driver modification. To modify SAP device types, refer to Note 3166.

The printer commands in the print control must never effect the current printer settings, since the SAPscript printer driver has no information on any changes triggered by a print control, which would have to be undone by commands of the driver itself. The printer driver assumes that the print control has no effect on any texts or graphics printed afterwards.

Never use PRINT-CONTROL to control print attributes covered by the SAPscript driver. To change such print attributes, use the usual method of defining form and style and the device type accordingly.

- The contents of the print control called are transparent to SAPscript. SAPscript cannot check whether the printer commands contained in the control are correct. Therefore, if you experience problems when printing a text containing these commands, you should first try to print the text without the print controls. Then, reintroduce the PRINT-CONTROL commands one by one until the command causing the problem is identified.
- You should ensure that the printer control sequences you define leave the printer afterwards in a well-defined state. SAPscript assumes that after completing each text output certain settings (for example, font, current page) are still valid for subsequent printing. If your printer commands change these settings without resetting them again afterwards, the results may be unpredictable.

After executing a PRINT-CONTROL command, SAPscript inserts a space character at the start of the next text line. If you do not want this, you should give this text line the '=' paragraph format.

Boxes, Lines, Shading: BOX, POSITION, SIZE

Use the BOX, POSITION, and SIZE commands for drawing boxes, lines, and shading to print particular windows within a form or passages of text within a window in a frame or with shading.

The SAP printer drivers that are based on page-oriented printers (the HP LaserJet driver HPL2, the Postscript driver POST, the Kyocera Prescribe driver PRES) employ these commands when printing. Line printers and page-oriented printers not supported in the standard ignore these commands. You can view the resulting printer output in the SAPscript print preview.

Syntax:

1. `/: BOX [XPOS] [YPOS] [WIDTH] [HEIGHT] [FRAME] [INTENSITY]`
2. `/: POSITION [XORIGIN] [YORIGIN] [WINDOW] [PAGE]`
3. `/: SIZE [WIDTH] [HEIGHT] [WINDOW] [PAGE]`

BOX Command

Syntax

```
/: BOX [XPOS] [YPOS] [WIDTH] [HEIGHT] [FRAME] [INTENSITY]
```

Effect: draws a box of the specified size at the specified position.

Parameters: For each of XPOS, YPOS, WIDTH, HEIGHT, and FRAME, you must specify both a measurement and a unit of measurement. Specify the INTENSITY parameter as a percentage between 0 and 100.

XPOS, YPOS

Upper left corner of the box, relative to the values of the POSITION command.

Default: Values specified in the POSITION command.

The following calculation is performed internally to determine the absolute output position of a box on the page:
X(abs) = XORIGIN + XPOS
Y(abs) = YORIGIN + YPOS

WIDTH

Width of the box. Default: WIDTH value of the SIZE command.

HEIGHT

Height of the box. Default: HEIGHT value of the SIZE command.

FRAME

Thickness of frame.

Default: 0 (no frame).

INTENSITY

Grayscale of box contents as %.

Default: 100 (full black)

Measurements: You must specify decimal numbers as literal values (like ABAP numeric constants) by enclosing them in inverted commas. Use the period as the decimal point character. See also the examples listed below.

Units of measurement: The following units of measurement may be used:

- TW (twip)
- PT (point)
- IN (inch)
- MM (millimeter)
- CM (centimeter)
- LN (line)
- CH (character).

The following conversion factors apply:

- 1 TW = 1/20 PT
- 1 PT = 1/72 IN
- 1 IN = 2.54 CM
- 1 CM = 10 MM
- 1 CH = height of a character relative to the CPI specification in the form header
- 1 LN = height of a line relative to the LPI specification in the form header

```
/: BOX FRAME 10 TW
```

Draws a frame around the current window with a frame thickness of 10 TW (= 0.5 PT).

```
/: BOX INTENSITY 10
```

Fills the window background with shading having a gray scale of 10 %.

```
/: BOX HEIGHT 0 TW FRAME 10 TW
```

Draws a horizontal line across the complete top edge of the window.

```
/: BOX WIDTH 0 TW FRAME 10 TW
```

Draws a vertical line along the complete height of the left hand edge of the window.

```
/: BOX WIDTH '17.5' CM HEIGHT 1 CM FRAME 10 TW INTENSITY 15
/: BOX WIDTH '17.5' CM HEIGHT '13.5' CM FRAME 10 TW
/: BOX XPOS '10.0' CM WIDTH 0 TW HEIGHT '13.5' CM FRAME 10 TW
/: BOX XPOS '13.5' CM WIDTH 0 TW HEIGHT '13.5' CM FRAME 10 TW
```

Draws two rectangles and two lines to construct a table of three columns with a highlighted heading section.

POSITION Command

Syntax

```
/: POSITION [XORIGIN] [YORIGIN] [WINDOW] [PAGE]
```

Effect: Sets the origin for the coordinate system used by the XPOS and YPOS parameters of the BOX command. When a window is first started, the POSITION value is set to refer to the upper left corner of the window (default setting).

Parameters: If a parameter value does not have a leading sign, then its value is interpreted as an absolute value, in other words, as a value that specifies an offset from the upper left corner of the output page. If a parameter value is specified with a leading sign, then the new value of the parameter is calculated relative to the old value. If one of the parameter specifications is missing, then no change is made to this parameter.

XORIGIN, YORIGIN

Origin of the coordinate system.

WINDOW

Sets the values for the left and upper edges to match those of the current window (default setting).

PAGE

Sets the values for the left and upper edges to match those of the current output page (XORIGIN = 0 cm, YORIGIN = 0 cm).

```
/: POSITION WINDOW
```

Sets the origin for the coordinate system to the upper left corner of the window.

```
/: POSITION XORIGIN 2 CM YORIGIN '2.5 CM'
```

Sets the origin for the coordinate system to a point 2 cm from the left edge and 2.5 cm from the upper edge of the output page.

```
/: POSITION XORIGIN '-1.5' CM YORIGIN -1 CM
```

Shifts the origin for the coordinates 1.5 cm to the left and 1 cm up.

SIZE Command

Syntax

```
/: SIZE [WIDTH] [HEIGHT] [WINDOW] [PAGE]
```

Effect: Sets the values of the WIDTH and HEIGHT parameters used in the BOX command. When a window is first started, the SIZE value is set to the same values as the window itself (default setting).

Parameters: If one of the parameter specifications is missing, then no change is made to the current value of this parameter. If a parameter value does not have a leading sign, then its value is interpreted as an absolute value. If a parameter value is specified with a leading sign, then the new value of the parameter is calculated relative to the old value.

WIDTH, HEIGHT

Dimensions of the rectangle or line.

WINDOW

Sets the values for the width and height to the values of the current window (default setting).

PAGE

Sets the values for the width and height to the values of the current output page.

```
/: SIZE WINDOW
```

Sets WIDTH and HEIGHT to the current window dimensions.

```
/: SIZE WIDTH '3.5' CM HEIGHT '7.6' CM
```

Sets WIDTH to 3.5 cm and HEIGHT to 7.6 cm.

```
/: POSITION WINDOW
/: POSITION XORIGIN -20 TW YORIGIN -20 TW
/: SIZE WIDTH +40 TW HEIGHT +40 TW
/: BOX FRAME 10 TW
```

A frame is added to the current window. The edges of the frame extend beyond the edges of the window itself, so as to avoid obscuring the leading and trailing text characters.

Hexadecimal Data: HEX, ENDHEX

Use this command to send printer commands in a printer language directly to a printer that supports that language. SAPscript does not interpret the data enclosed by the HEX and ENDHEX command pair, but inserts it unchanged into the output stream. This technique allows objects with a pixel-oriented format (for example, graphics) to be printed as part of a SAPscript text. The HEX and ENDHEX command pair enclose the printer commands and data as hexadecimal text, so that the printer formatting routines interpret each successive pair of characters as a single hexadecimal value in the range of 0 to 255. The characters 0 to 9 and A to F for representing the values 10 to 15 are valid hexadecimal characters. The text may also include comment lines (these begin with /* in the format column), which will not be interpreted as hexadecimal data but are simply passed over by the formatting routines.

Syntax:

```
/: HEX [TYPE printer_language]
```

The HEX command denotes the start of the hexadecimal data. Subsequent text lines are interpreted as described above. If the TYPE parameter is present, the data will be sent to the printer only if the printer understands the specified printer language. The following printer languages are currently supported: POST (Postscript), PRES (Kyocera Prescribe) and PCL (HP Printer Control Language).

```
/: HEX [TYPE printer_language] [XPOS x_position] [YPOS y_position]
```

The output cursor is set to the absolute position indicated by the specified X and Y position parameters before the hexadecimal data is printed. If either the X or the Y position is not specified, then 0 will be assumed for this parameter.

```
/: HEX [TYPE printer_language] [HEIGHT height] [LEFT left_indentation]
```

The HEIGHT parameter determines the amount of space to be reserved on the page for the output of the hexadecimal data. Any text following the ENDHEX command will be printed below this point. If the LEFT parameter is also specified, then the output of the hexadecimal data will is indented from the left margin by the specified amount.

```
/: HEX TYPE PCL HEIGHT '7.5' CM LEFT '2.25' CM
/* Creator: report ZQVNTE30 date 19940705 time 125129 user
SAPSCRIPT
/=
1B2A7230461B2A743735521B2A7231411B2A62304D1B2A62343057FFFFFFFF
FFFF
/=
FF1B2A62343057FFFFFFFFFFFFC0007D00DFC0F7D000000000000000000000
0017
/: ENDHEX
```

This data is printed only by an HP PCL printer (fro example, HP LaserJet). 7.5 cm of space is allocated on the page for the output of the data and the output cursor is indented 2.25 cm to the right of the form window edge.

⚠

You can use the RSTXLDMC program to upload correctly formatted pixel data to the R/3 system and to prepare it as a HEX-ENDHEX control command. This can then be saved as normal SAPscript text.

Summing a Program Symbol: SUMMING

The SUMMING command is used for accumulating a total value for a program symbol. The command should be specified just once. Then, each time the specified program symbol is formatted, its current value is added into the total symbol. Several program symbols may all be added into a single total symbol.

Syntax:

```
/: SUMMING program_symbol INTO total_symbol
```

SAPscript cannot create the field for the total dynamically. The summing symbol used for totalling must be declared with TABLES in the ABAP program. Otherwise, only zero is added. Declaring the symbol with the DATA statement is not sufficient (global data).

For details on summing and carrying forward, see Summing and Carrying Forward is Incorrect [Seite 227].

SAPscript Symbols

Text in the SAP system does not usually exist independently of other objects, but often contains a reference to some other stored object. For example, for a letter this could be the address data in the vendor master record or information in the material master record that is to be included in a purchase order text. You solve this problem by using placeholders for the data rather than entering the actual values into the text. Thus, you can create flexible text modules by using these placeholders at all points where the text needs to be variable. Since much of the data to be inserted in the text reflects the contents of fields in SAP tables, this technique ensures that the text modules always contain the current values of these fields when printed.

In SAPscript, these placeholders are known as symbols. They represent data that will not be added to the text until a later point. This is normally the point at which the output is formatted. All symbols occurring in the text are then replaced with their current values. This replacement is performed only in the output text. The original version of the text module is unaffected.

SAPscript recognizes four different kinds of symbols:

- System symbols
- Standard symbols
- Program symbols
- Text symbols.

The main difference between these is the source of their values. SAPscript provides values for the system symbols. Standard symbols and their values are defined in the TTDTG table. Program symbols represent data supplied by the program that is currently executing. The values for text symbols come either from control commands in the text itself or are supplied dynamically by the *Include* function in the text editor.

SAPscript automatically recognizes the type of a symbol. First of all, it checks whether the symbol is a system symbol. If not, then it checks whether the symbol name is defined in the data area of the calling program. In this case, it is a program symbol. Otherwise, SAPscript reads table TTDTG. If the symbol is found in this table, then it is a standard symbol. If a symbol is neither a system symbol nor a program symbol nor a standard symbol, then it is a text symbol.

Syntax of Symbols [Seite 143]

System Symbols [Seite 144]

Program Symbols [Seite 162]

Standard Symbols [Seite 169]

Text Symbols [Seite 170]

Formatting Options [Seite 171]

Formatting Conventions [Seite 194]

Syntax of Symbols

Each symbol has a name that is used when the symbol is called. A call to a symbol is normally made in a line of text that also includes other kinds of text. Therefore it is necessary that symbols can be distinguished from normal text, and that the call is structured in such a way that it is possible to identify it as a call to a symbol.

- Use the delimiter & both immediately before and after the symbol.
- Do not use blank characters in the name of a symbol. Moreover, since the characters '+() are used for defining formatting options, you must not use any of these a symbol name either.
- Make sure that no SAPscript editor line break occurs between the symbol delimiters. If necessary, use a long line to avoid this (paragraph format = or /=).
- Enclose additional formatting options in round brackets and insert them immediately after the symbol name. The code letters identifying these options must be given in capitals.

A string that does not satisfy all the above conditions is not interpreted as a symbol but is copied directly into the output text.

Examples of valid symbols:

```
&symbol&
&MY_symbol&
&KNA1-NAME1&
&DATE&
&KNA1-UMSAT(I)&
```

Examples of invalid symbols:

`&mysymbol`	closing delimiter missing
`&my   symbol&`	name contains blanks
`&mysymbol)&`	name contains an invalid character
`&symbol(Z&`	closing bracket of formatting option missing
`&KNA1-UMSAT(i)&`	formatting option not in capitals

The symbol names themselves are not case-sensitive, that is, SAP script does not distinguish between capital and lower case letters in symbol names. These symbol names are identical:

&mysymbol&
&Mysymbol&
&MYSYMBOL&

A symbol name can contain a maximum of 130 characters. However, only the first 32 characters are used for unique identification.

System Symbols

You can use system symbols in all kinds of text. SAPscript supplies the values for system symbols. The names of the system symbols are fixed.

Current Date

&DATE&	The current date is displayed. It is formatted according to the specifications found in the user master data. You can adapt this format to your own requirements by specifying a date mask (SET DATE MASK) or by using a country-specific formatting option (SET COUNTRY). The current value for this symbol is taken from the SY-DATUM field. This value is not copied every time that the date is called, but only at the following times: – When printing starts (OPEN_FORM, PRINT_TEXT) – When symbols are replaced in the text editor – When a text is exported in the ASCII or RTF format – When the TEXT_SYMBOL_REPLACE function module is called (optional)

Current Day Number

&DAY&	The current day number is printed. The display includes leading zeros.

Current Month Number

&MONTH&	The current month number is printed. The display includes leading zeros.

Current Year Number

&YEAR&	This symbol is used to print the current year as a four digit number.

Local Date (Currently Only for Japan)

&LDATE&	The current date in converted to a local date and printed. The formatting is performed using the JPDAT edit mask. Since this representation employs the Japanese character set, it is of interest only in the Japanese version of the R/3 system.

Current Day Name (Long Form)

&NAME_OF_DAY&	The name of the current day is written out in full. The language used for the output is determined by the appropriate text language or form language. The names of the days are stored in the TTDTG table under the key %%SAPSCRIPT_DDDD_dd, where dd is the day number (01= Monday,.., 07 = Sunday).

Current Month Name (Long Form)

`&NAME_OF_MONTH&`	The name of the current month is written out in full. The language used for the output is determined by the appropriate text language or form language. The names of the months are stored in the TTDTG table under the key %%SAPSCRIPT_MMMM_mm, where mm is the month number (01,.., 12).

Current Time

&TIME&	The current time is printed in the form hours:minutes:seconds. Each of the components for hours, minutes, and seconds consists of two digits, using a leading zero if necessary. You can adapt this format to your own requirements by specifying a time mask (SET TIME MASK). The value for the time field is taken from the SY-UZEIT field. This value can be copied over only at particular times (c.f. DATE).

Hours Component of Current Time

&HOURS&	The component of the current time referring to hours is printed. The display includes leading zeros.

Minutes Component of Current Time

&MINUTES&	The component of the current time referring to minutes is printed. The display includes leading zeros.

Seconds Component of Current Time

&SECONDS&	The component of the current time referring to seconds is printed. The display includes leading zeros.

Current Page Number

&PAGE&	You can use this symbol to insert into the text the page number that the current page will have when printed. You can specify the formatting option for the page number in the form for each page type.

Page Number of the Next Page

&NEXTPAGE&	This symbol is used to print the number of the following page. The output format is the same as with &PAGE&. Note that on the last page of the output, in each window that is not of type MAIN, &NEXTPAGE& has the value 0.

Selected Device Type

&DEVICE&	The &DEVICE& symbol is used to print the type of the output device. This type is passed in the DEVICE parameter when the SAPscript output (OPEN_FORM, PRINT_TEXT) is started, and it specifies the device for which the output should be formatted. Possible values are: PRINTER SCREEN TELEX TELEFAX ABAP (ABAP list display)

Spaces

&SPACE&	You can use this symbol to generate a string of space characters. You must pass the number of space characters required with the symbol. If you leave out the number, then no spaces are printed.

Underline

&ULINE&	You can use this symbol to insert a string of underline characters into the output text. You must pass the number of underline characters required with the symbol. If you leave out the number, then just one underline character is printed.

Vertical Line

&VLINE&	You can use this symbol to insert a string of vertical line characters into the output text. You must pass the number of vertical line characters required with the symbol. If you leave out the number, then just one vertical line character is printed.

Program Symbols

The integration of SAPscript allows to link data that is stored in various applications of the SAP system into text modules; for example a form letter to be sent to several customers. The address information of these customers is in the SAP database and must be incorporated into the letter. SAPscript cannot read this data out of the SAP database itself, but has to call another program to do this. The data is then copied into work areas declared with TABLES.

Starting with Release 3.1G, you are no longer restricted to the TABLES statement. You can address any global variable using a program symbol. The system can evaluate the ABAP Dictionary information (output length, number of decimal places, and so on) not only for TABLES fields, but also for INFOTYPES fields and variables with a LIKE reference.

Example:

```
DATA: MYCOUNTRY LIKE USR03-LAND1.
```

The system considers all output characteristics that can be retrieved using the ABAP statement DESCRIBE.

If SAPscript is now called from this program to format a text, it can copy the data out of these work areas.

Symbols that obtain their values from this kind of data area are called program symbols. The value of a program symbol is limited up to a maximum of 255 characters. The name of a program symbol, when using TABLES statements, consists of the table name and the field name, separated by a hyphen. Examples of customer address fields are: &KNA1-NAME1&, &KNA1-ORT01&, &KNA1-PFACH&. Otherwise, the symbol is used in the way it is defined in the print program (for example, &MYCOUNTRY). When SAPscript encounters a symbol, it first checks whether the symbol is a system symbol. If it finds no corresponding entry, it tries to find a table or structure in the calling ABAP program (declared with TABLES). If there is, the symbol is a program symbol and SAPscript next checks in the Dictionary to see whether this table contains the specified field.

If there is no table or structure in the calling ABAP program, SAPscript checks whether the symbol is a standard symbol. If no entry is found in table TTDTG, the system checks the calling program for global definitions (DATA, CONSTANTS, INFOTYPE, PARAMETER). If it finds any definitions, SAPscript processes the symbol as program symbol.

Only if no global definitions are found either, does SAPscript process the symbol as text symbol.

Basically, a defined text symbol remains a text symbol even if in the print program, for example, a DATA statement with the same name is used.

For replacing the variables, the sequence of the variables in the corresponding text is decisive.

Form/text:

```
....
/:  DEFINE &mysymbol& = ‘abc’
*   &mysymbol&
....
```

Print program:

```
....
Data: mysymbol(5) value ‘xyz’.
....
```

In this example, in the form/text instead of &mysymbol& the value of the text symbol defined with DEFINE is printed: **abc**

Form/text:

```
....
*   &mysymbol&
/:  DEFINE &mysymbol& = ‘abc’
....
```

Print program:

```
....
Data: mysymbol(5) value ‘xyz’.
....
```

In this example, in the form/text instead of &mysymbol& the value of the program symbol defined in the print program is printed: **xyz**

Usually, SAPscript looks for the table work areas and the global variables in the main part of the ABAP program that was started. For example, if you call function modules that you want to use the table work areas, enter the name of the corresponding program in the field TDPROGRAM of the structure you can enter in the OPTIONS parameter at OPEN_FORM. This definition is then valid for all program symbols up to the next CLOSE_FORM. However, you can also specify the name of the program in the PROGRAM parameter of the START_FORM function module. If you do this, it will be valid up to the next END_FORM. If this parameter is not set in START_FORM, then the setting in OPEN_FORM applies.

For formatting a program symbol, SAPscript uses the specifications found in the Dictionary (output length, formatting routine, and so on). Certain data types require additional information out of other fields to format a value. For example, the contents of a currency field can be formatted correctly only when the information from the currency field key is also available. SAPscript itself is responsible for obtaining the necessary information from the Dictionary.

For printing the program symbols, SAPscript uses the WRITE statement of the ABAP programming language. The effect of this statement is controlled externally via parameters (such as specifications in the user master record, table T005X), depending on the data type. If a program symbol is not printed in the way you expected it, first check whether the control parameters stated above are set correctly.

To fields of the table work areas shown below, you can refer in all SAPscript text modules:

- SYST: System Fields in the ABAP Programming Environment [Seite 165]

Program Symbols

- USR03: User Address Data [Seite 166]
- SAPSCRIPT: General SAPscript Fields [Seite 167]

SYST: System Fields in the ABAP Programming Environment

You can refer to all the fields in this table. You should, however, note that some of the fields are specific to a certain environment. They contain values that do not come from your application but have been set by the SAPscript programming environment (for example, SYST-REPID).

USR03: User Address Data

This structure contains information from the user master record for a given user:

- Business address
- Telecommunication (telephone, telefax)
- Other data such as user language, department, cost center.

You can maintain the contents of these fields in the ‘Address’ section of the user maintenance.

If this table exists in the calling program, then SAPscript copies the data from the work area of this program. Otherwise, SAPscript uses the values for the currently active user.

SAPSCRIPT: General SAPscript Fields

You can print the following fields of structure SAPSCRIPT as program symbols in SAPscript forms:

- &SAPSCRIPT-SUBRC&:

  Receives a value after executing an INCLUDE statement. The value shows whether the INCLUDE was found (that is, the INCLUDE text exists) or not. You can query this value in an IF statement.
  INCLUDE was found: &SAPSCRIPT-SUBRC& = 0
  INCLUDE was not found: &SAPSCRIPT-SUBRC& = 4.

- &SAPSCRIPT-DRIVER&:

  SAPscript formats a text for a specific output device. The initial formatting is independent of the specific language of this device. SAPscript then calls a driver to convert the device-independent format to device-specific control commands. This field contains the name of the driver.

POST	Postscript driver
HPL2	HP Laserjet driver for the PCL4/PCL5 languages
PRES	Driver for output devices using the PRESCRIBE language

  The available drivers are stored in table TSP09.

- &SAPSCRIPT-FORMPAGES&:

  This field contains a number representing the total number of pages of the currently formatted form (any output between START_FORM and END_FORM). The page counter mode (START, INC, HOLD) of the individual pages is ignored. You can use this symbol to formulate information like

  'Page x of y'   for your output.

- &SAPSCRIPT-JOBPAGES&:

  This field contains a number representing the total number of pages of all forms contained in the currently formatted print request, in other words, of all forms created using the OPEN_FORM, START_FORM.. ENDFORM, START_FORM.. END_FORM,..., CLOSE_FORM function modules.

  When using the SAPSCRIPT-FORMPAGES or SAPSCRIPT-JOBPAGES symbols, SAPscript leads all output pages of the current form or current print request into main memory to replace the symbol by the appropriate value. For large output jobs, this can mean a very large amount of memory.

- &SAPSCRIPT-COUNTER_x& (x = 0.. 9):

  These fields represent ten counter variables that you can use in your text and forms for any counting purposes. You can use the '+' and '-' formatting options to increment or decrement a counter before its value is printed. You can use the DEFINE control command to assign any specific value to a counter.

SAPSCRIPT: General SAPscript Fields

- &SAPSCRIPT-TELELAND&:

  This field contains the country key of the fax target address when using fax output via SAPscript (field ITCPO-TDTELELAND of parameter OPTIONS of function module OPEN_FORM).

- &SAPSCRIPT-TELENUM&:

  This field contains the local fax number of the fax target address when using fax output via SAPscript (field ITCPO-TDTELENUM of parameter OPTIONS of function module OPEN_FORM).

- &SAPSCRIPT-TELENUME&:

  This field contains the complete fax number of the fax target address when using fax output via SAPscript (field ITCPO-TDTELENUME of parameter OPTIONS of the function module OPEN_FORM).

Standard Symbols

Standard symbols are defined in table TTDTG. This table contains both the name of each symbol and its value. The value, which is language-dependent, can contain up to 60 characters. SAP supplies this table filled with standard entries. You can extend it with customer-specific symbols.

You can use standard symbols in all kinds of text.

Text Symbols

All symbols that do not correspond to one of the three types of symbols described above are text symbols. You define the value of a text symbol yourself in the text module.

There are two ways of doing this:

- In the text editor, choose *Include* → *Symbols* → *Text*.

  All the text symbols contained either in the current text or in a form assigned to the text are displayed. You can assign a value of up to 80 characters to a text symbol. Enter this value in the same form as it is to appear in the output.

  The effect of assigning a value is temporary, since the values assigned are not stored with the text but are lost as soon as you leave the editor. You use this kind of value assignment if you had a ready-made text containing symbols that you want to print with specific values in place of the symbols, and you want to do this only once without storing the 'changed' text.

- In the text, use the control command DEFINE.

  Since control commands are stored with the text module, any value you assign in this way is preserved when you save the text. You can change the value assigned to a symbol in the text at any time simply by issuing another DEFINE command.

  Remember always to use the ' (inverted comma) character to delimit a value. The maximal length for these values is also 80 characters.

  A text in the editor contains the following DEFINE commands:

  ```
  /:  DEFINE &mysymbol& = 'xxx xxx xxxxx xxxx'
      &mysymbol&
  /:  DEFINE &mysymbol& = 'yyyyy yyy yyyy'
  /   &mysymbol&
  ```

  The printed text appears like this:

  ```
      xxx xxx xxxxx xxxx
      yyyyy yyy yyyy
  ```

Formatting Options

The value of a symbol is normally printed using its full length, although trailing spaces are removed. An exception are program symbols of these data types: CURR, DEC, QUAN, INT1 INT2, INT4, PREC, and FLTP. These are formatted right-justified with an output length as specified in the Dictionary.

You can adapt the standard formatting to your own requirements by using one of the additional formatting options available. You supply the parameters for these options together with the symbol itself. Many of these options are abbreviated to a single letter, which has to be given as a capital letter. You can combine two or more options on a single symbol, as long as the result still makes sense.

Offset

Specifying an offset has the effect that a certain number of bytes of the symbol value, starting with the first byte on the left, will not be displayed. If the offset specified is greater than the length of the value, nothing is printed.

Syntax

`&symbol+offset&`

If <`symbol`> has the value 123456789, the following will be displayed:

```
&symbol&       ->    123456789
&symbol+3&           ->    456789
&symbol+7&           ->    89
&symbol+12&          ->
&symbol+0&           ->    123456789
```

Output Length

If you need only a part of the symbol value, or the output has to fit into a box or a field on the screen without overlapping the edges of this area, then you can use an output length specification to define how many bytes of the value should be copied.

Syntax

```
&symbol(length)&
```

If <symbol> has the value 123456789.

```
&symbol(3)&          ->    123
&symbol(7)&          ->    1234567
```

You can combine an output length specification with an offset specification. The specified length is then counted from the specified offset position.

```
&symbol+4(3)&  ->    567
```

If a length specified is greater than the current length of the value, then spaces are appended to the symbol value.

You can use the character * to specify the length of a program symbol. The value of the symbol is then printed using the output length defined in the ABAP Dictionary.

Syntax

```
&symbol(*)&
```

The SYST-UNAME field contains the logon name of a user called Einstein. The Dictionary entry for this field contains an output length of 12.

```
&SYST-UNAME&...            ->    Einstein...
&SYST-UNAME(9)&...   ->    Einstein...
&SYST-UNAME(*)&...   ->    Einstein   ...
```

Omitting the Leading Sign

Program symbols with numeric values can have a leading sign. This sign usually appears to the right of the numeric value, either as a space for positive numbers, or as a minus sign for negative numbers. You can use the S option to ensure that the value is formatted without the sign.

Syntax

```
&symbol(S)&
```

The ITCDP-TDULPOS field contains the value -100.00. The ABAP Dictionary definition for this field includes a leading sign.

```
&ITCDP-TDULPOS&            ->     100.00-
&ITCDP-TDULPOS(S)&   ->      100.00
```

Leading Sign to the Left

The leading sign is normally displayed to the right of a numeric value, except in the case of a floating point number. This option enables you to specify that the leading sign should be placed to the left of the number.

Syntax

```
&symbol(<)&
```

```
&ITCDP-TDULPOS&            ->     100.00-
&ITCDP-TDULPOS(<)&   ->     -100.00
```

The SET SIGN LEFT control command specifies that all subsequent symbols with a numeric value should have a left-justified leading sign. If you use this control command, you must no longer repeat the < option for each individual symbol.

Leading Sign to the Right

The default setting is to print the leading sign to the right of a numeric value. If you used the SET SIGN LEFT control command to specify that the leading sign should be printed in front of the value, you can override this specification for individual symbols. The symbols specified with the > option are then printed with the leading sign to the right.

Syntax:

```
&symbol(>)&
```

You can use the SET SIGN RIGHT control command to switch back to the default setting for the output of the leading sign.

Omitting Leading Zeros

Certain symbol values are printed with leading zeros. If you want to suppress these, use the Z option.

Syntax

```
&symbol(Z)&
```

Assuming the current date is 1.1.1994,

```
&DAY&                ->    01
&DAY(Z)&       ->    1
```

Space Compression

The symbol value is viewed as a sequence of ‘words’, each separated from the next by either one or a string of space characters. The C option has the effect of replacing each string of space characters with a single space and shifting the ‘words’ to the left as necessary to close up the gaps. Leading spaces are completely removed. The results are the same as those of the ABAP command CONDENSE.

Syntax:

```
&symbol(C)&
```

Assuming `'    Albert    Einstein  '` is the symbol value,

```
&symbol&       ->        Albert    Einstein
&symbol(C)&    ->    Albert Einstein
```

Number of Decimal Places

A program symbol of one of the data types DEC, QUAN, and FLTP can contain decimal place data. Use the option below to override the Dictionary definition for the number of decimal places for the formatting of this symbol value.

Syntax

```
&symbol(.N)&
```

The EKPO-MENGE field contains the value 1234.56. The Dictionary definition specifies 3 decimal places and an output length of 17.

```
&EKPO-MENGE&         ->                1,234.560
&EKPO-MENGE(.1)            ->                1,234.6
&EKPO-MENGE&(.4)     ->                1,234.5600
&EKPO-MENGE&(.0)     ->                1,235
```

Omitting the Separator for 'Thousands'

Symbols of the DEC, CURR, INT, and QUAN data types are normally formatted with the a 'thousands' separator character. The T option allows you to omit this separator character.

Syntax:

```
&symbol(T)&
```

The EKPO-MENGE field contains the value 1234.56. The Dictionary definition specifies 3 decimal places and an output length of 17.

```
&EKPO-MENGE&         ->            1,234.560
&EKPO-MENGE(T)&            ->             1234.560
```

Specifying an Exponent for Floating Point Numbers

The way a floating point number is formatted depends on whether an exponent is specified. The mantissa is adjusted by shifting the decimal point and, if necessary, introducing leading zeros, according to the exponent chosen. Using an exponent value of 0 means that the exponent representation will not be used for displaying the symbol.

Syntax

`&symbol`**`(EN)`**`&`

If you specify an exponent of 0, then the number is displayed without using the exponent representation. This has the same effect as completely omitting the specification of an exponent: `&symbol(E0)&` has the same effect as `&symbol(E)&`

In this example, the PLMK-SOLLWERT field is assumed to have the value 123456.78 and to be of data type FLTP.

```
&PLMK-SOLLWERT&            ->    +1.23456780000000E+05
&PLMK-SOLLWERT(E3)&  ->    +123.456780000000E+03
&PLMK-SOLLWERT(E6)&  ->    +0.12345678000000E+06
&PLMK-SOLLWERT(E0)&  ->    +123456.780000000
&PLMK-SOLLWERT(E)&   ->    +123456.780000000
```

Right-Justified Output

Symbol values other than numeric values are normally formatted left-justified. To specify right-justified formatting, use the R option. You must use this option in conjunction with an output length specification.

Syntax

```
&symbol(R)&
```

If `symbol` has the value 1234.

```
&symbol&       ->    1234
&symbol(8R)          ->        1234
```

For program symbols, the length specification contained in the Dictionary definition may be used instead of an explicit length.

Fill Characters

You can replace leading spaces in a value with a fill character. Use the F option with the character immediately following the F in the specification as the fill character.

Syntax

&symbol**(Ff)**&

f = fill character

The figure for customer sales in the KNA1-UMSAT field is $700. The Dictionary description of the field specifies an output length 8.

```
&KNA1-UMSAT&                ->      700.00
&KNA1-UMSAT(F*)&            ->    **700.00
&KNA1-UMSAT(F0)&            ->    00700.00
```

Suppressing Output of Initial Values

Use the I option to suppress the output of symbols that still contain their initial values.

Syntax

```
&symbol(I)&
```

Assuming KNA1-UMSAT contains the value 0 and the currency is DEM.

```
&KNA1-UMSAT&          ->          0,00
&KNA1-UMSAT(I)&             ->
```

If the field contains an amount other than 0, this value is printed in the normal way.

```
&KNA1-UMSAT&          ->        700,00
&KNA1-UMSAT(I)&             ->        700,00
```

Ignoring Conversion Routines

SAPscript conversion routines specified in the Dictionary are automatically recognized and used when program symbols are formatted. To suppress conversion, use the K option.

Syntax

```
&symbol(K)&
```

Local Dates (Currently Only for Japan)

Use the L option to convert a date field to a local date and to print it in the appropriate format.
The edit mask JPDAT is used for the formatting.
This representation uses the Japanese character set, and therefore should be used only in a Japanese version of the R/3 system.

Syntax

```
&symbol(L)&
```

Changing the Value of a Counter

You can increase or decrease the value of a SAPSCRIPT-COUNTER_x (x=0.. 9) counter variable by 1, before the current counter value is printed.

Syntax:

```
&SAPSCRIPT-COUNTER_x(+)&          Increases by 1 the contents
                                        of the counter variable x
                                        (x=0.. 9)

&SAPSCRIPT-COUNTER_x(-)&          Decreases by 1 the contents
                                        of the counter variable x
                                        (x=0.. 9)
```

If you want to change the value of a counter variable without actually printing the new value, use this formatting option together with an additional option to set the output length to 0 (see above). If you want to set a counter variable to some specific value, use the DEFINE control command.

Assume that &SAPSCRIPT-COUNTER_1& initially has the value 2.

```
&SAPSCRIPT-COUNTER_1&            ->    2
&SAPSCRIPT-COUNTER_1(+)&         ->    3

&SAPSCRIPT-COUNTER_1(-)&         ->    2

&SAPSCRIPT-COUNTER_1(-)&         ->    1

&SAPSCRIPT-COUNTER_1(+0)&        ->

&SAPSCRIPT-COUNTER_1(+)&         ->    3
```

Preceding and Subsequent Texts (Pre-Text / Post-Text)

In addition to using initial values for symbols, you can specify additional texts that are printed only when the value of the symbol is no longer the initial value. You can specify a text to be printed immediately before the symbol value (the pre-text), and a text to be printed immediately after the symbol value (the post-text). If the symbol contains its initial value, these texts are suppressed.

Syntax:

```
&'pre-text'symbol'post-text'&
```

Make sure that the symbol, the pre-text, and the post-text all appear on a single line of the editor. You may have to use a long line (paragraph attribute = or /=) in the editor.

The inverted comma ' is used as a delimiter for these texts. If this character is also part of one of these texts, enter it twice at this point, so that it is not interpreted as a delimiter character. A pre-text or post-text may itself contain symbols in addition to normal text. However, these symbols may not have a pre-text or a post-text.

The KNA1-PFACH field contains a customer P.O. Box number. Since the text “P.O. Box” is not stored in the field along with the value, you would normally write the following for the P.O. Box line of an address:

```
P.O. Box &KNA1-PFACH&
```

However, if no P.O. Box has been specified, the text “P.O. Box” would still appear on its own in the address. To prevent this, use pre-text or post-text (in this case, pre-text).

```
P.O. Box &KNA1-PFACH&      ->    P.O. Box
&'P.O. Box 'KNA1-PFACH&    ->
```

If a P.O. Box has been specified, then this will be displayed together with the appropriate text in the normal way.

```
&'P.O. Box 'KNA1-PFACH&    ->    P.O. Box 123456
```

Country-Dependent Formatting

Certain fields are formatted specific to a particular country. These include fields for displaying a date and numeric fields containing either a decimal point or a ‘thousands’ separator character. The formatting applied is usually determined by the definitions contained in the user master record. You can use the SET COUNTRY control command to choose a different formatting operation. The various country-dependent formatting options are stored in table T005X.

Syntax

```
/: SET COUNTRY country_key
```

You can specify this country key either by quoting it directly enclosed in inverted commas or by using a symbol.

```
/: SET COUNTRY 'CAN'
/: SET COUNTRY &KNA1-LAND1&
```

You can revert to the settings of the user master record by using the SET COUNTRY control command again with an empty country name.

```
/: SET COUNTRY ' '
```

When SAPscript encounters this command it calls the corresponding ABAP command internally. The effect of the SAPscript command is thus identical with that of the ABAP command.

If the formatting turns out other than expected, check the settings in table T005X.

Date Mask

To format date fields, use the SAPscript SET DATE MASK command. Executing this command causes all subsequent date fields to be printed with the specified formatting.

Syntax

```
/: SET DATE MASK = 'date_mask'
```

The following templates may be used in the date mask:

DD	day (two digits)
DDD	name of day (abbreviated)
DDDD	name of day (written out in full)
MM	month (two digits)
MMM	name of month (abbreviated)
MMMM	name of month (written out in full)
YY	year (two digits)
YYYY	year (four digits)
LD	day (formatted as for the L option)
LM	month (formatted as for the L option)
LY	year (formatted as for the L option)

Any other characters occurring in the mask are interpreted as simple text and are copied directly to the output.

Assuming a current system date of March 1st, 1997.

```
/: SET DATE MASK = 'Foster City, MM.DD.YY'
   &DATE&           ->    Foster City, 03.01.97
   &DATE(Z)&   ->    Foster City, 3.1.97

/: SET DATE MASK = 'MMMM DD, YYYY'
   &DATE&           ->    March 01, 1997
```

You can revert to the standard setting by using the SET DATE MASK command again with an empty string in place of the date mask:

/: SET DATE MASK = ' '

Time Mask

You can use the SAPscript SET TIME MASK command to format time fields in a way that differs from the standard setting. Executing this command causes all subsequent time fields to be printed with the specified formatting.

Syntax:

```
/: SET TIME MASK = 'time_mask'
```

The following templates may be used in the time mask:

HH	hours (two digits)
MM	minutes (two digits)
SS	seconds (two digits)

Any other characters occurring in the mask are interpreted as simple text and are copied directly to the output.

Assuming the current time is 10:08:12.

```
    &TIME&            ->    10:08:12

/:  SET TIME MASK = 'HH:MM'
    &TIME&            ->    10:08

/:  SET TIME MASK = 'HH hours MM minutes'
    &TIME&            ->    10 hours 08 minutes
    &TIME(Z)&   ->    10 hours 8 minutes
```

You can revert to the standard setting by using the SET TIME MASK command again with an empty string in place of the time mask:

```
/: SET TIME MASK = ' '
```

Using TrueType Fonts

Use

TrueType fonts are fonts that are not built into the printer, but that are loaded at request from the computer into the printer memory before the actual print process starts.

As of SAP Release 4.6C, you can import TrueType font files and then use these fonts for printing SAP forms. This allows you to, for example, use your company's own fonts.

Prerequisites

You need a TrueType file for the desired fonts (for example, **myfont.ttf**), which you can load into the SAP System. Note that you cannot load double-byte and multi-byte fonts (for example, Japanese, Chinese), since these font files are too large.

Performance

Each time you create a new spool request, you include the font definition into the request. If there are many requests to be printed, the performance of the SAP system may be reduced.

Features

TrueType fonts can by default be used by all device types that use the drivers POSTSCRIPT, PCL, or SAPWIN. Exceptions are the device types:

POSTSCPT	PostScript-Printer Latin 1
HPLJ1100	HP Laserjet 1100

Activities

To use TrueType fonts for printing SAP forms, proceed as follows:

1. Start SAPscript font maintenance (transaction **SE73**). Choose *Install TrueType font*.
2. Enter a font name that starts with Z (for example, ZMYFONT). The system uses this name for the TrueType font.
3. Specify the attributes of the font (normal, bold, italic, or bold and italic). To do this, mark the respective fields:

   normal: no field

   bold: field *Font attribute BOLD*

   italic: field *Font attribute ITALIC*

   bold and italic: fields *BOLD and ITALIC*

   To use a font with all attributes, you therefore need four font files (z.B. **myfont.ttf**, **myfonti.ttf**, **myfontb.ttf**, **myfontbi.ttf**).
4. Enter the path and file name of the font file and choose *Execute*.
5. Specify the control sequence that identifies the font in Rich Text Format.

   The SAPscript text editor and the graphic print preview use this information to display the characters correctly.

The font info has the following structure:

\f<name of the font family>**\fcharset**<WinCharSet number>**\f**<font name>

For the SAPscript font TIMES this information is:

\froman\fcharset &&& Times New Roman

For more information see the RTF Specification by Microsoft (http://www.microsoft.com).

6. Create a style (transaction **se73**) or a form (transaction **so10**) and define formats using the TrueType fonts.

Result

You can print and archive forms using TrueType fonts. To print a form, use an output device whose device type uses one ot the drivers PCL, Postscript, Prescribe, or SAPWIN.

Formatting Conventions

Many allowed combinations of option and symbol type either make no sense or have no effect at all. The formatting process is divided into two stages: primary formatting and end formatting. The purpose of primary formatting is to convert the internal form of a symbol value to a character-based form. The end formatting is then applied to this intermediate form. The following sections describe the formatting process for each kind of symbol and in which of the two stages of conversion the various options are applied.

Primary Formatting of System Symbols [Seite 195]

Primary Formatting of Standard Symbols [Seite 196]

Primary Formatting of Program Symbols [Seite 197]

Primary Formatting of Text Symbols [Seite 200]

End Formatting [Seite 201]

Primary Formatting of System Symbols

Certain system symbols must first be converted from their internal representation into a character representation before the end formatting can be applied to them. The following system symbols require a primary conversion:

- &DAY&, &MONTH&, &YEAR&, &HOURS&, &MINUTES&, &SECONDS&

  The relevant components are extracted from the system date or system time. Leading zeros are eliminated if the Z formatting option is specified.

- &DATE&

  The default behavior is that the system date is printed according to the setting in the user master record or according to the country-dependent formatting specified in a SET COUNTRY command.

  If a date mask has been defined (using SET DATE MASK), then the formatting specified there is used for the output.

- &TIME&

  The default behavior is that the current system time is printed in the form

  hours:minutes:seconds

  If an alternative time mask is specified (using SET TIME MASK) then this mask overrides the default setting.

- &PAGE&, &NEXTPAGE&

  This symbols are initially converted using the options specified in the form of the page definition.

- &LDATE&

  This date representation is always formatted using the JPDAT conversion routine.

All remaining system symbols are treated by the primary conversion as a character string and thus are not converted.

The end conversion is started when the primary conversion is complete.

Primary Formatting of Standard Symbols

Standard symbols are treated by the primary formatting as a character string and thus are not converted. The end formatting is started immediately.

Primary Formatting of Program Symbols

Program symbols are initially converted from their internal representation into a character representation. This is done using the ABAP WRITE command together with additional parameters as appropriate for the specified options. The options to which the primary conversion applies depend on the data type.

The formatting is carried out in an internal work field by the ABAP WRITE command. This field is chosen according to the data type and length. The various options are handled by appropriate extensions to the WRITE commands:

S	NO-SIGN
Z	NO-ZERO
.	DECIMALS
E	EXPONENT
I	A check is made for an initial value by using an IF statement to do an IS INITIAL comparison.

	I	**S**	**Z**	*****	**K**	**<**	**>**	**E**	**.**	**Length**
CHAR	x		x	x						
CURR	x	x		2	x	x	x		x	2
DEC	x	x		2	x	x	x		x	2
NUMC	x		x	x	x					
VAR	x			x	x					

VARC	x			x	x					
LANG	x			x	x					
UNIT	x			x	x					
DATE	x				x					
DATS	x				x					
TIME	x				x					
TIMS	x				x					
QUAN	x	x		2	x	x	x		x	2
INT1	x	x	x	2	x	x	x			2
INT2	x	x	x	2	x	x	x			2
INT4	x	x	x	2	x	x	x			2

PREC	x	x	x	2	x	x	x			2
CUKY	x			x	x					
ACCP	x			x	x					
CLNT	x			x	x					
FLTP	x	1		2	x			x	x	2

1. ABAP WRITE currently ignores this option for floating point numbers.
2. The value is formatted using the ABAP WRITE command in a work field with the specified length

If a conversion routine is defined in the ABAP Dictionary, then the primary formatting is performed using this routine only. In this case, none of the options listed in this table are applicable unless you deactivate the conversion routine with the K option.

After the primary conversion of a program symbol is completed, the formatted value is then subjected to the end formatting.

Primary Formatting of Text Symbols

Text symbols are treated by the primary formatting as a character string and thus are not converted. The end formatting is started immediately.

End Formatting

In end formatting, the following formatting options are applied to the symbol value in the given order. These are the same for all symbol types.

1. If the C option is used, a space compression is performed.
2. If the L option is used, local date formatting is performed.
3. If an offset is specified, the offset is interpreted.
4. If an output length is specified then the length is interpreted.
5. If a fill character is specified (the F option), these are added to the formatted value.
6. If pre-texts and post-texts are soecified, they are processed and added to the formatted value.

Printing Labels

Use

You want to print labels (for example, shipping labels or barcode labels) on a special printer. Release 4.6A allows you to create labels using an external design program and to print them from within a SAPscript form.

Prerequisites

Nearly all special label printers on the market use individual control languages, for which no printer drivers exist in the SAP Standard. To be able to address these label printers from within the R/3 system nevertheless, you use an external program to define the entire layout.

Activities

1. **Create the label using a design program**: Define the entire layout of the label, including the fields you later want to fill from within the R/3 system.
2. **Download the print file**: Export the data using the printer commands of the design program. The printer commands must be in ASCII format, that is, you can use only printable characters plus carriage return, line feed, and, maybe, form feed. In addition, the file must not comprise more than 80 characters per line; otherwise unwanted line feeds may occur during upload. There must be no binary control characters (for example, escape).
3. **Upload the print file into the SAPscript form**: Upload the print file into a SAPscript standard text. However, you use this standard text as "intermediate storage" only. Then copy the text into a SAPscript form. In the form, enter variables of the print program (program symbols) in those places where you want variable data to appear in the label. At runtime, the system fills these variables with the current field values from the application program.
4. **Adapt the form**: For most label printers you must adapt the form; for example, you must enlarge the MAIN window to maximum page size and delete all other windows. In addition, the MAIN window should contain only one single text element: the imported print file.
5. **Create an output device**: Define an output device for label printing. As device type use either ASCIIPRI or one of the special device types for label printers.

The subsequent topics tell you how to create labels for the individual printer types and how to print them from within SAPscript.

- Printing Labels With Avery TTX 450 [Seite 204]
- Printing Labels With CAB Apollo 2 [Seite 206]
- Printing Labels With Intermec Easycoder 501 XP [Seite 208]
- Printing Labels With Intermec Easycoder 4420 [Seite 210]
- Printing Labels With Printronix T 3204 [Seite 213]
- Printing Labels With SATO CL408 [Seite 215]
- Printing Labels With Zebra Z4000 [Seite 217]

Printing Labels With Avery TTX 450

Use

This topic tells you how to use the method described under Printing Labels [Seite 202] to create and print labels with the TTX 450 of Avery-Dennison.

Procedure

1. **Create the label**: Use the design program *Jetmark 32* of Avery Dennison to create the label layout. This program runs under Windows 95/98 and under Windows NT. As printer select the Avery printer *TTX 450 12 dpmm* in *Jetmark 32*.
2. **Define the R/3 form fields**: In *Jetmark 32*, define all those fields as "normal" input fields with constant text (not as database variables) that will later be filled with variable data from the R/3 system. Into each field enter the field name that will later be used in R/3 (for example, `VBAK-KUNNR`) as plain text; this allows you to easily find the field in the print file created by the download. To "simulate" the size the field will have later, you can extend the field name to the desired size by adding any characters (for example, `VBAK-KUNNRXXXXX`).

   In some barcode fields, you cannot enter field names but only characters corresponding to the barcode type (for example, only digits). In this case, use a text constant that corresponds to the input data (for example, 01234567 for an 8-digit numeric barcode field).
3. **Select fonts for text fields**: For text fields with variable contents you can use only printer-internal fonts. For text fields with constant contents you can use either printer-internal fonts or TrueType fonts. During download, constant texts with TrueType fonts are transformed into bit patterns and stored in the print file, which enlarges them.
4. **Insert bitmap graphics**: You can include bitmap graphics (for example, company logos) into the label definition; during download, they are inserted into the print file. However, transform any colored bitmaps into black-and-white bitmaps before importing them into *Jetmark 32*.
5. **Download the label definition into the print file**: In *Jetmark 32* choose *File* → *Print* to transfer the printer commands in a print file. Select the field *Print in file* and enter path and file names for the file to be created. You can use *Settings* to set options such as "cutting the label at the end of the job" or "Blank label and cutting at the end of the job".
6. **Upload the print file into SAPscript and insert the variables**: Start the SAPscript standard text editor (*Utilities* → *SAPscript* → *Standard text*). Create a new standard text. Upload the label file using function *Text* → *Upload*. As format choose `ASCII`. Save the file.

   In the text, search for the variable fields you defined under 2.). Replace the constant text you entered there (for example, `VBAK-KUNNR`) with the name of the actual variable as it is used by the print program of the application (for example, `&VBAK-KUNNR&)`. In the editor choose *Edit* → *Command* → *Insert command* to insert SAPscript variables (program symbols).
7. **Adapt the SAPscript form**: To print the label, adapt the SAPscript form you want to use. Choose *Utilities* → *SAPscript* → *Form*:
   - Copy the standard text that contains the print file into a text element of the MAIN window. The name of this text element depends on the R/3 application program you use.

- The first page of the form should point to itself as next page, since the label file in the MAIN window may be very large, especially if graphics are included.

  Since Avery printers ignore any unknown commands in the print data stream, it is not necessary to execute the actions described below. However, they make the form clearer and easier to use:

- Delete all windows except MAIN. If you keep any windows, they must not contain any data you want to print (you may mark texts as comments).
- The MAIN window should contain only one text element, which contains the label file you just created.
- If there are any other text elements defined in MAIN that are called from within the print program, mark their contents as comments.

8. **Create an output device**: Define the Avery printer as Output Device in the SAP System [Extern].

   The easiest way is to connect the printer to a Windows PC, install any Windows printer driver there, and start the output program SAPlpd. Then define the printer in the spool administration (transaction SPAD) using coupling type "S" or "U".

   As device type select **`LB_AVE`**.

Printing Labels With CAB Apollo 2

This topic tells you how to use the method described under Printing Labels [Seite 202] to create and print labels with the printer Apollo 2 of the manufacturer CAB GmbH. According to CAB, you can use the same procedure for the printer models Apollo 1, Apollo 3, Apollo 4, and for the fully automatic labelling system "CAB Hermes 4" as well.

Procedure

1. **Create the label**: To design the label layout, use the design program *Easylabel* of the manufacturer Tharo Systems, Inc. This program runs under Windows 3.1, Windows 95, and Windows NT. As printer in *Easylabel* select CAB Apollo 2.

   For technical reasons, you can use the CAB printer from within SAPscript only if you previously stored the label definition in the printer (on memory card) and then transfer from within SAPscript only the field contents of the variable label fields. To be able to use this method, each printer must be equipped with a memory card.

2. **Define the R/3 form fields**: In *Easylabel* define all those fields as type *FIX* (not as database fields) that will later be filled with variable data from the R/3 system. As field contents select sample data that allows you to estimate the space needed for the field contents. To load the label definition onto the memory card of the printer at a later time, each variable field must have a field name in *Easylabel*. The field names in *Easylabel* can be up to 10 characters long and alphanumeric. Therefore, you cannot use R/3 field names directly. Instead, use neutral field names such as **T1** for the first, **T2** for the second text field, **B1** for the first barcode field, and so on.

3. **Select fonts for the text fields**: For text fields, you can use either printer-internal or TrueType fonts. Since the label definition is stored together with the fonts on a memory card in the printer, the font selected does not influence the size of the print file created by R/3.

4. **Insert bitmap graphics**: Bitmap graphics are also stored on the memory card.

5. **Transfer the label definition to the memory card**: To transfer the label definition to the memory card installed in the printer, choose *Print a batch of formats*, load the stored label, and execute the function *Download format to memory card*. Enter the name of the label, which may be up to eight characters long. This name is used later to identify the label definition.

   Set the parameter *Quantity* to the value *Printer prompts for quantity*, otherwise the printer will always print one label with the current data and, additionally, one original label.

   Besides the label definition downloaded to the printer, *Easylabel* creates a file of type "[filename].rpl" on the PC. Usually, you find this file in the installation directory of *Easylabel* on the PC. This RPL file is the basis for the print file to be sent from R/3, which passes values for the variable fields of the label and finally prints the label.

6. **Upload the RPL file into SAPscript and insert the variables**: Start the SAPscript standard text editor (*Utilities* → *SAPscript* → *Standard text*). Create a new standard text. Upload the RPL file using function *Text* → *Upload*. As format select **ASCII**. Save the file.

   For each variable field to which you assigned a field name in *Easylabel* under 2.), the file contains a line R XX, where XX is the field name (for example, **R T1;**). After the semicolon enter the program variable from the print program of the application, whose value you want to print in this field (for example, `R T1;&VBAK-KUNNR&)`. In the editor choose *Edit* → *Command* → *Insert command* to insert SAPscript variables (program symbols).

7. **Adapt the SAPscript form**: To print the label, adapt the SAPscript form you want to use for printing. Choose *Utilities* → *SAPscript* → *Form*:

   - The MAIN window must cover the entire page format (for example, DINA4), which means that there may not be an upper or left margin between MAIN and the page margin.
   - The MAIN window should contain only one text element which contains the newly created RPL file. The name of the text element depends on the R/3 application program you use. Insert the RPL file into this text element by copying the entire text from SO10 into the form window.
   - If there are any other text elements defined in MAIN that are called from within the print program, mark their contents as comments.
   - Delete all windows except MAIN. If you keep any windows, they must not contain any data you want to print (you may mark texts as comments).
   - The first page of the form should point to itself as next page, since the label file in the MAIN window may be very large.

8. **Create an output device**: Define the CAB printer as Output Device in the SAP System [Extern].

   The easiest way is to connect the printer to a Windows PC, install any Windows printer driver there, and start the output program SAPlpd. Then define the printer in the spool administration (transaction SPAD) using coupling type "S" or "U".

   As device type select **`LB_CAB`**.

Printing Lables With Intermec Easycoder 501 XP

This topic tells you how to use the method described under Printing Labels [Seite 202] to create and print labels with the printer EasyCoder 501 XP of Intermec.

Procedure

1. **Create the label**: To create the label layout, use the design program *Label 98* of the manufacturer Intermec GmbH. This program runs under Windows 95/98 and under Windows NT. As printer in *Label 98* select *Intermec Easycoder 501XP*.

2. **Define the R/3 form fields**: Define in *Label 98* all fields that will be filled later in R/3 with variable data as database fields. As field names use the R/3 field names. To do this, select the *Database* tab of the field parameters: Into the field *takes data from* enter the R/3 field name as it will later be used in the SAP*script* form, but **without** the the "&" characters used in SAP*script* at beginning an end and **without** formatting options. In the field *Format* enter the usual SAP*script* outptu options for symbols, for example fixed output length. In the field *Output,* the variable then appears in the way it will be used in the SAP*script* form, that is, including output options and "&" characters.

   You want to print the field `VBAK-KUNNR` from the print program of the R/3 application in a *Label 98* text field with a fixed length of 8. To do this, fill in the fields of the *Database* tab of *Label 98* as follows:

Database	
Field name	VBAK-KUNNR
Takes data from	VBAK-KUNNR
SAP R/3	
Format	8
Output (the contents of field "Output" is generated by *Label 98* itself)	&VBAK-KUNNR(8)&

   To display the field contents of such database fields in the graphical display of *Label 98* enter sample data or the field name into the field "*Variable text*" of the *Contents* tab.

3. **Select fonts for the text fields**: For text fields, automatically only printer-internal fonts of the Intermec printer are displayed (*Fonts* tab).

4. **Insert bitmap graphics**: You can include bitmap graphics (for example, company logos) into the label definition. During download, they are included into the print file.

5. **Download the label definition to a file**: In *Label 98* choose *Labels* → *Layout* and enter the path and file names for the file you want to create. In the specified path *Label 98* then creates two files with identical names but different extentions:

   - `[Dateiname].cod`
   - `[Dateiname].itf`: You can upload this file in the SAP*script* ITF format into the R/3 system.

6. **Upload the label file into SAP*script***: Start the SAP*script* standard text editor (*Utilities* → *SAPscript* → *Standard text*). Create a new standard text with any name. Upload the ITF file using function *Text* → *Upload*. As format select **ITF**. Save the file.

7. **Adapt the SAP*script* form**: To print the label, adapt the SAP*script* form you want to use for printing. Choose *Utilities* → *SAPscript* → *Form*:

    - The MAIN window must cover the entire page format (for example, DINA4), which means taht there may not be an upper or left margin between MAIN and the page margin.
    - Delete all windows except MAIN. If you keep any windows, they must not contain any data you want to print (you may mark texts as comments).
    - The MAIN window should contain only one text element which contains the newly created RPL file. The name of the text element depends on the R/3 application program you use. Insert the ITF file into this text element by copying the entire text from the standard text editor into the form window.
    - If there are any other text elements defined in MAIN that are called from within the print program, mark their contents as comments.
    - The first page of the form should point to itself as next page, since the label file in the MAIN window may be very large, especially if it contains graphics.

8. **Create an output device**: Define the Intermec printer as Output Device in the SAP System [Extern].

    The easiest way is to connect the printer to a Windows PC, install any Windows printer driver there, and start the output program SAPlpd. Then define the printer in the spool administration (transaction SPAD) using coupling type "S" or "U".

    As device type select **LB_UBI**.

Printing Labels With Intermec Easycoder 4420

Use

This topic tells you how to use the method described under Printing Labels [Seite 202] to create and print labels with Intermec Easycoder 4420.

According to the manufacturer you can use the same procedure also for the printer models Easycoder 3240, 3400, 3440, 3600, 4400, 4420, 4440, 4630, 4830 and 7421 by Intermec.

Procedure

1. **Create the label**: To design the label layout, use the design program *LabelShop PRO*, version 4.21 of the manufacturer Intermec. This program runs under Windows 95/98 and Windows NT.

    In addition, you need the product "ERPLabel(TM) - for SAP R/3" by Intermec. (http://www.intermec.com/products/printers.htm)

    As printer, select the relevant Intermec printer in *LabelShop PRO*, in our example *Easycoder 4420*.

2. **Define the R/3 form fields**: In *LabelShop PRO* define all those fields as variable fields of type *Import* that will later be filled with variable data from the R/3 system. As variable name use the R/3 field name (for example, **VBAK_KUNNR**).

    Do not enclose the variable in the ampersands ("&") generally used in SAPscript, because *LabelShop PRO* automatically adds them.

    LabelShop PRO does not allow certain characters in field names. The character "-", which is used in R/3 to separate table name and field name, is not allowed. Use the underline ("_") instead. If you use an "illegal" character in the variable name, *LabelShop PRO* ignores you newly created variable and names it with a default name, such as "Import1". If you detect during label printout from SAPscript that field contents are missing, check the variables (program symbols) in the SAPscript form.

    You want to print the field **VBAK_KUNNR** from the print program of the R/3 application in a *LabelShop PRO* text field, with a fixed length of 8: In *LabelShop PRO* create a variable text field. On the next screen, choose *New* to create a field of type *Import* and name **VBAK_KUNNR**. Then choose *Add*.

    Use the context menu (right mouse button) of the new field to specify the attributes of the variable. Determine the field position and the output length of the variable. To enable the system to display field contents of the variable fields in the graphical display, enter sample data ("") in field *Value of variable - In the label*. In the icon bar you can toggle between field name display and field contents display.

3. **Select fonts for the text fields**: For text fields with variable contents you can use only printer-internal fonts. For text fields with constant contents you can use either printer-internal or TrueType fonts.

4. **Insert bitmap graphics**: You can include bitmap graphics (for example, company logo as *.BMP file) into the label definition. The system passes them to the print file during the

download. You can use these graphic formats: *.bmp, *.dib, *.rle, *.dxf, *.eps, *.fmf, *.img, *.jpg, *.pcd,*.pcx, *.dcx, *.png, *.tga, *.wmf, *.wpg.

5. **Barcodes**: When defining barcode fields, use only printer-internal barcodes ("printer barcodes"). If you use others as well ("graphic barcodes"), the file created for upload into R/3 contains non-printable characters that interfere with the import into R/3.

6. **Download the label definition into the print file**: To convert the printer commands into an ITF file suited for import into SAPscript on the PC, choose *File* → *Make an object file for printer*. You can now choose between two variants: template SAPHOST.POC and template SAPSTORE.POC.

   – SAPHOST.POC: With variant SAPHOST.POC, the system stores the entire label definition together with the label data in a file suited for import into R/3. When printing the label from R/3, the label definition together with the variable data is passed to the printer each time. This produced rather large files, especially if you use graphics and TrueType fonts. Specify the path and file names of the file to be created. When executing the function, *LabelShop PRO* creates a file xxxxx.ITF in the specified path, where xxxxx is the label name used in *LabelShop PRO*.

   – SAPSTORE.POC: With variant SAPSTORE.POC, the label definition is stored in the printer. From R/3, you must retrieve only the variable data of the label. This produces extremely small print files in R/3, which makes SAPSTORE.POC much better for the performance than SAPHOST.POC.

     Choose *File* → *print* to pass the label definition into the resident memory of the Intermec printer. Make sure that for the printer driver settings (*File* → *Printer* → *Setup*) the option *Store layout when printing - with format number XX* is activated on the *Forms* tab. Enter the path and file names of the file to be created. When executing the function, *LabelShop PRO* creates a file xxxxx.ITF in the specified path, where xxxxx is the label name used in *LabelShop PRO*.

7. **Upload the print file to SAPscript and insert the variables**: Start the SAPscript standard text editor (*transaction SO10*). Create a new standard text. Use *Text* → *Upload* to load the label file; choose `ITF` as format. Save the file.

8. **Adapt the SAPscript form**: To print the label, adapt the SAPscript form you want to use for printing. Choose *Utilities* → *SAPscript* → *Form*:

   – The MAIN window must cover the entire page format (for example, DINA4), which means that there may not be an upper or left margin between MAIN and the page margin.

   – Delete all windows except MAIN. If you keep any windows, they must not contain any data you want to print (you may mark texts as comments).

   – The MAIN window should contain only one text element which contains the newly created label file. The name of the text element depends on the R/3 application program you use. Insert the label file into this text element by copying the entire text from SO10 into the form window.

   – If there are any other text elements defined in MAIN that are called from within the print program, mark their contents as comments.

   – The first page of the form should point to itself as next page, since the label file in the MAIN window may be very large.

9. **Create an output device**: Define the Intermec printer as Output Device in the SAP System [Extern].

Printing Labels With Intermec Easycoder 4420

The easiest way is to connect the printer to a Windows PC, install any Windows printer driver there, and start the output program SAPlpd. Then define the printer in the spool administration (transaction SPAD) using coupling type "S" or "U".

As device type select **`LB_INT`**.

Printing Labels With Printronix T 3204

This section explains the procedure described in Printing Labels [Seite 202] to create and print labels using the printer Printronix T3204.

Procedure

1. **Creating the label**: Use the design program *Codesoft Pro 4.20* by Techniques Avancées to create the label layout. You can run this program under Windows 3.1, Windows 95, and Windows NT. As printer choose within *Codesoft: Printronix T 3204 (PGL)*.

   Define the printer within *Codesoft* with output port *FILE* to be able to load the print data into a file at a later time.

   When defining the output format, deactivate the option *Automatically adapt size*. To do this, choose *File* → *Format* → *Page*.

2. **Defining the R/3 form fields**: Define all fields, which R/3 later fills with variable data, as fields with constant values, not as database fields. As field contents, use the R/3 field name. If this is not possible – for example, for certain barcodes – enter a combination of numbers which you can easily find in the print file.

   If the field contents exceeds the printable area of the defined output format, the printer T 3204 shows an error message and does not print the label. Therefore, make sure that the contents of a completely filled output field still fits into the defined label. If necessary, enhance the R/3 field names (for example, **VBAK-KUNNR**) with additional characters to ensure output of a variable in its maximum length.

3. **Selecting the fonts for text fields**: For text fields, use only printer-internal fonts of the Printronix printer, no TrueType fonts.

4. **Inserting bitmap graphics**: You cannot include bitmap graphics into the label definition. During download they are converted into binary data which cannot be interpreted by SAPscript.

5. **Downloading the label definition into the file**: To transfer the printer commands to a file, choose *File* → *Print*. Select the printer with output port *FILE* and choose *Print*. Enter a name for the print file.

6. **Uploading the label file into SAPscript and inserting the variables**: Start the SAPscript standard text editor (*Utilities* → *SAPscript* → *Standard text*). Create a new standard text. Choose *Text* → *Upload* to upload the print file. As format select **ASCII**. Save the file.

   Search for the variable fields and replace the constant text there (for example, **VBAK-KUNNR**) with the variable actually used in the print program (for example, **&VBAK-KUNNR&**). To do this, choose *Edit* → *Command* → *Insert command* and specify the variable in the field *Symbols*.

7. **Adapting the SAPscript form**: To print the label, adapt the desired SAPscript form. Choose *Utilities* → *SAPscript* → *Form*:

   - Define the first page of the form as its own next page, since the label file in the MAIN window may be very large.

- The MAIN window should contain only one text element which contains the label file. The name of this text element depends on the R/3 application program used. To insert the label file into this text element, copy the entire text from the standard text editor into the form window.

The modifications described below are not mandatory, since the printer language Printronix PGL ignores any unknown commands and leading blanks or blank lines are no problem. However, for better readability, you should make the following changes to the form:

- Delete all windows except MAIN, or make sure that these windows don not contain any data to be printed (make them comments).
- If the print program calls any other text elements in MAIN, change their contents to comments.

8. **Creating an output device**: Define the Printronix printer as Output Device in the SAP System [Extern].

   The easiest way is to connect the printer to a Windows PC, install any Windows printer driver there and start the output program SAPlpd. The define the printer in the spool administration (transaction SPAD) using coupling type "S" or "U".

   As device type choose `LB_PRI`.

Printing Labels With SATO CL408

Use

This topic tells you how to use the method described under Printing Labels [Seite 202] to create and print labels with SATO CL408.

According to the manufacturer, you can use this procedure also for the printer models CL412, CL608, CL 612, M8400RV, M5900 RV, M8485S, M8460S, M8490S, M8459S, XL400, XL410 by SATO.

Procedure

1. **Create the label**: To design the label layout, use the design program *Dynamic Publisher, version 2.5* of the manufacturer SATO. This program is part of the program package *Dynamic Aviator* and runs under Windows95/98 and Windows NT.

   As printer, select the desired SATO printer in *Dynamic Publisher*; in this example, it is *CL408*.

2. **Define the R/3 form fields**: In *Dynamic Publisher* define all those fields as variable fields that will later be filled with variable data from the R/3 system. *Dynamic Publisher* allows you to assign variable names for this purpose. You can choose variable names that ressemble the "real" R/3 field namens, for example MATNR.

   However, the field names in *Dynamic Publisher* cannot be very long. Therefore, you won't be able to use the complete R/3 field names directly in most cases.

3. **Select fonts for the text fields**: You can use printer-internal as well as TrueType fonts. Since you cannot download any TrueType font definitions from R/3 to the printer, you must use the program *Dynamic Memory Card Compiler* (included in the software package *Dynamic Aviator*) beforehand to load any TrueType fonts you may want to use onto a memory card that must be built into the printer.

4. **Insert bitmap graphics**: You must also load any required bitmap graphics beforehand onto the printer memory card, using the *Dynamic Memory Card Compiler*.

5. **Download the label definition into the print file**: Choose *File* → *Export* → *Export to SAP*. Enter the variable name used in R/3 (that is, the genuine R/3 field name, for example, `VBAK-KUNNR`) for each variable field used in the label. Do not include the ampersands ("&") used by SAPscript to identify variables; *Dynamic Publisher* automatically includes them during the export.

   Dynamic Publisher creates a file in the Spscript ITF format in the subdirectory *Labels* of the *Dynamic Aviator*. The file name is the same as the name of your label in *Dynamic Publisher*, however, the file extension is "ITF".

6. **Upload the print file to SAPscript and insert the variables**: Start the SAPscript standard text editor (transaction SO10). Create a new standard text. Use *Text* → *Upload* to load the label file; choose `ITF` as format. Save the file.

7. **Adapt the SAPscript form**: To print the label, adapt the SAPscript form you want to use for printing. Choose *Utilities* → *SAPscript* → *Form*:

   – The MAIN window must cover the entire page format (for example, DINA4), which means that there may not be an upper or left margin between MAIN and the page margin.

- Delete all windows except MAIN. If you keep any windows, they must not contain any data you want to print (you may mark texts as comments).
- The MAIN window should contain only one text element which contains the newly created label file. The name of the text element depends on the R/3 application program you use. Insert the label file into this text element by copying the entire text from SO10 into the form window.
- If there are any other text elements defined in MAIN that are called from within the print program, mark their contents as comments.
- The first page of the form should point to itself as next page, since the label file in the MAIN window may be very large, especially if graphics are included.

8. **Create an output device**: Define the SATO printer as Output Device in the SAP System [Extern].

   The easiest way is to connect the printer to a Windows PC, install any Windows printer driver there, and start the output program SAPlpd. Then define the printer in the spool administration (transaction SPAD) using coupling type "S" or "U".

   As device type select **LB_SAT**.

Printing Labels With Zebra Z4000

Use

This topic tells you how to use the method described under Printing Labels [Seite 202] to create and print labels with Zebra Z4000.

Procedure

1. **Create the label**: To design the label layout, use the design program *BAR-ONE Tool for SAP R/3 label printing* of the manufacturer Zebra. For more information see http://www.zebra.com.

   As printer, select the printer Zebra Z4000 in *BAR-ONE.*

2. **Define the R/3 form fields**: In *BAR-ONE* define all those fields as variable fields of type *SAP variable field* that will later be filled with variable data from the R/3 system.

   In the input field *Identifier* of the field attributes enter the R/3 field name to be used in SAPscript, enclosed in ampersands ("&"), for example **`&VBAK-KUNNR&`**.

3. **Select fonts for the text fields**: For variable and constant text fields you can use printer-internal fonts as well as TrueType fonts.

   For Zebra label printing, SAP has defined two different device types (see 8.). The difference lies in the character set coding for special characters (for example, German umlauts):

   – The device type LB_ZEB supports the character set IBM 850 of the printer-internal font "CG Triumvirate Bold Condensed". Use this device type if you want to print special characters in variable text fields in this printer-internal font.

   – The device type LB_ZEB2 supports the character set "Latin-1" of the TrueType fonts. Use this device type if you want to print special characters in variable text fields using TrueType fonts.

4. **Insert bitmap graphics**: You can include bitmap graphics (for example, company logo as *.BMP file) into the label definition. The system passes them to the print file during the download.

5. **Download label definition into print file**: Choose *File* → *Create Format for SAP R/3* to convert the printer commands into an ITF file that is suited for import into SAPscript. The *Download Stored Format Wizard* appears and allows you to choose between two procedures:

   – Download the label definition into the Zebra printer (RAM or nonvolatile memory) and create a merge file for SAPscript.

   – Create a file for SAPscript that contains label definition plus variable data.

   For performance reasons you should prefer the first method, since that print files created in R/3 to be sent to the printer have only minimal size. With the second method, even though you avoid keeping the label definition in the printer, you must pass the entire label definition to the printer for each label you print from R/3 (including graphics, fonts, and so on), which is considerably slower.

   BAR-ONE in both cases creates a file with the extension ".ITF".

6. **Upload the print file to SAPscript**: Start the SAPscript standard text editor (transaction SO10). Create a new standard text. Use *Text* → *Upload* to load the label file; choose `ITF` as format. Save the file.

7. **Adapt the SAPscript form**: To print the label, adapt the SAPscript form you want to use for printing. Choose *Utilities* → *SAPscript* → *Form*:

   - The MAIN window must be as wide as possible to avoid printer commands being split up by line feeds. If the width of the main window is less than 15 cm (6 inches), enlarge it to at least this value.

   - You can keep all other windows unless they overlap with the MAIN window. The Zebra printer ignores any texts you may output in these windows.

   - Delete all windows except MAIN. If you keep any windows, they must not contain any data you want to print (you may mark texts as comments).

   - The MAIN window should contain only one text element which contains the newly created label file. The name of the text element depends on the R/3 application program you use. Insert the label file into this text element by copying the entire text from SO10 into the form window.

   - If there are any other text elements defined in MAIN that are called from within the print program, you can leave them unchanged since the printer ignores unknown commands. However, for better readability you should mark their contents as comments.

   - The first page of the form should point to itself as next page, since the label file in the MAIN window may be very large, especially if it includes graphics.

8. **Create an output device**: Define the Zebra printer as Output Device in the SAP System [Extern].

   The easiest way is to connect the printer to a Windows PC, install any Windows printer driver there, and start the output program SAPlpd. Then define the printer in the spool administration (transaction SPAD) using coupling type "S" or "U".

   As device type select `LB_ZEB` or `LB_ZEB2`.

Analyzing Problems when Printing with Forms

To find and resolve the problem when a SAPscript form prints out incorrectly,

- Make sure that you have collected the necessary basic information.
- Choose the topic that most closely matches the problem you have.

This section assumes that the printer in question is correctly installed and has been successfully tested for printing from the SAP System. If this is not the case, then also refer to the error analysis procedure in the SAP *Printing Guide*.

Collecting Basic Information [Seite 220]

Data is Missing, or Field Contents or Includes are not Printed [Seite 221]

Formatting is Incorrect [Seite 223]

Text is Shifted or Wrongly Positioned [Seite 225]

Characters are Printed Incorrectly [Seite 226]

Summing and Carrying Forward is Incorrect [Seite 227]

Using the SAPscript Form Debugger [Seite 229]

Collecting Basic Information

As the first step in analyzing a form printing problem, be sure that you have collected all of the following information:

- The form and ABAP print program used

  Are you using the SAP standard versions of these objects or have the form or the print program been modified?

- The printer used

  What is the device type of the printer in the SAP spool system? Is it an SAP standard device type or have you defined or modified the device type?

- The formatting option (that is, the page format in the form) you use to process the spool print request

Data is Missing, or Field Contents or Includes are not Printed

In your printout, one of the following errors occurs:

- Data is missing.
- Field contents (data from program fields) is not printed.
- Documents that you have inserted with the INCLUDE command are not printed.
- Symbols are not resolved and printed.

Do the following to localize the error:

- Can you display the data that is missing in the SAP spool system (transaction SP01, display spool request)?

  If you can display such data, then the problem is in the printer setup, either in the SAP spool system or in your host PC or workstation. Otherwise, the problem is in the print program or form.

- Do the fields in the print program get filled with the right values at the right time?

  You can check this by running the print program in the ABAP debugger. Note that SAPscript can read fields only if they are declared with the ABAP TABLES statement.

- Is the name of the print program passed to SAPscript correctly?

  The name of the print program must be in the ITCPO-TDPROGRAM parameter (OPTIONS parameter) of the call to the OPEN_FORM function module in the print program. Otherwise SAPscript cannot "find" the data fields. The default is to search in the active main program.

- If an INCLUDE document is missing, then have the parameters of the INCLUDE command been set correctly?

  ```
  /: INCLUDE <name> OBJECT <object> ID <id> LANGUAGE <l>
  ```

  Both the name of the text and the language are usually set with variables whose values are not known until runtime. Does the INCLUDE text actually exist in the current client in the language that is requested?

- Is a text is not being printed **and** does the corresponding text element in the form consist only of an INCLUDE command or symbols?

  Then you can test the text element by inserting a few literal characters as ‘hard text’ into the text element.

  Activate the form and start the print run again. If the new characters are printed, then the print program calls the text element. The cause of the problem therefore lies in the INCLUDE parameters.

  If the new characters are not printed, then either of the following errors is likely to have occurred:

  – The text element is not called from the print program. You can check for an error in the print program with the form debugger.

- The form is used in a language variant other than that specified for the printout. This language to use is specified by the print program.

Formatting is Incorrect

If paragraph or character formats are incorrect when you print out a form, do the following to localize the error:

- Is SAPscript actually being used for printing?

  Some applications (such as the Reportwriter) that use the SAPscript editor for maintaining texts do their printing using the "normal" ABAP list printing. In this case, SAPscript formatting options will have no effect at all.

  You can check whether SAPscript is being used for printing by checking attributes of the spool request in transaction SP01. If the spool request uses ABAP printing, then a format name beginning with X_ will appear in the *Format* field. If the spool request uses SAPscript formatting, then a name such as DINA4, INCH12 that does not begin with X will appear in *Format*. :

  In the print programs in some SAP applications (for example RFDUZI00), you can request on-screen formatting of the form that is to be printed. This is realized in SAPscript by setting DEVICE=SCREEN. This switch causes SAPscript not to format the output as for a printer, but instead to format the output as a normal list for screen display. The screen display then includes a *Print* button, which, however, merely causes the screen list to be printed using the normal ABAP list printing functionality. A print preview for DEVICE=PRINTER (which can be printed by SAPscript) can be identified by the fact that “print preview” appears in the title bar of the screen

- What are the typeface attributes in the form? The typeface used for printing a piece of text is determined by the following settings:

  – Default type setting for the form (in *Header data* → *Font attributes*)

  – Font attributes of the paragraph used or of the default paragraph (in *Paragraphs* → *Font attributes*)

  – Font attributes of any character format used (in *Character formats* → *Font attributes*)

  – Fonts and type faces supported by the device type in use

  SAPscript must choose a printer font (font available on the target printer) that meets the font attributes specified for the form. If the requested typeface does not exist on the target printer, then SAPscript chooses a substitute. If the substitute is not an exact match to the SAPscript typeface, then this substitution can produce output errors.

- Are both the paragraph and character formats defined in the form?

- If the incorrectly formatted text is inserted into the main window (MAIN) using INCLUDE, then check that no style is assigned to the text that is included. If a style (or form that includes styles) is assigned to the document, then these formats are used to format the included text.

- If the incorrectly formatted text is inserted with the INCLUDE... PARAGRAPH <format name> command, then check to be sure that there are no INCLUDE commands in the

Formatting is Incorrect

included document. Specifying a paragraph format for an INCLUDE does not work for “nested” INCLUDEs.

- Use the following procedure to test whether a text element in a form is correctly formatted. This procedure lets you test without having to start the print program in the SAP application, which may require considerable knowledge of the application.

  To test a text element, do the following:

  a) Copy the form to a name in the customer name space, Z<xxxx>, ZTEST, for example. Use transaction SE71 to do this.

  b) Edit the copy of the form. Specify the text element that you want to test as the default text element in the window in which it is defined. That is, put the text element right at the start of the window text, and do not use the **`/e element_name`** identifier with it.

  c) Activate the test form.

  d) Create or change a standard text. Assign your test form to the document by choosing *Format → Form*. You can do this in the standard SAPscript text processing function (transaction SO10).

  e) Print the test document on the printer at which the output errors occurred.

  If the text element is incorrectly formatted, you can correct it in your standard text document. You can then transfer the corrections to the original text element in its form.

Text is Shifted or Wrongly Positioned

If the entire output or individual texts are shifted horizontally or vertically on paper, check for the following possible sources of the error:

- Are tabs used in the document where the shifting occurs? Then check the lengths of the character strings in that line. If a character string is so long that it extends past a tab stop, then the following tab is forced over to the next tab stop. The text may then be shifted too far to the right or may even be forced onto the next line.

  Assume that the following paragraph format and text line are in your text:

  ```
  T4,,STRING1,,&TABLE-FIELD1&,,STRING2,,STRING3
  ```

  The doubled commas are interpreted by SAPscript as tabs.

  If the contents of **TABLE-FIELD1** are longer (dependent on the typeface and the printer being used) than the space between the 2nd and 3rd tab stops, then STRING2 will be shifted across to the 4th tab position and STRING3 may even cause a line break and end up on the next line. Also check the alignment of the texts: LEFT; CENTER; RIGHT; DECIMAL; SIGN.

- Printing on a device type that is controlled by the standard driver (STN2) for line printers can result in horizontal or vertical shifting of text. This shifting occurs because the standard driver cannot overprint text (print text on top of other text) and can work only within a fixed-width scheme for character and line spacing.

  Such shifting occurs when any of the following is true:

  - The device type SAPWIN is used with the “page printer” attribute set wrongly. This problem has been fixed as of System R/3 Release 2.2E/2.1K (see also *Note 19807*).
  - A non-integer line spacing (for example, 0.6 LINES) has been set for the paragraph. The STN2 driver has to round this up to the next integer number of lines.
  - Characters or strings have been defined to be on top of others (for example, in the case of overlapping windows). When using a page-oriented printer (HP LaserJet, PostScript or Kyocera) this usually is no problem (page-oriented printers can print characters on top of others). However, the STN2 driver has to shift strings that overlap others either to the left or the right.

Characters are Printed Incorrectly

If characters in your document are printed out incorrectly, for example with the # character substituted for special characters, check these possible sources of error:

- Is an SAP standard device type being used? Is the required character set actually supported by this device type?
- Have changes been made to the codepage definitions in the spool administration function? If they have, you should delete the SAPscript load in all clients by using the RSTXDELL report (parameter: client='* ').
- Check the character with the Display function in the spool output controller (transaction SP01). If the character is displayed correctly in the output controller, then look for a problem in the spool system or the printer definition. For example, check that the character set buffer is up to date, that the character set in the SAP printer definition is correctly maintained, and that the printer itself offers a compatible character set.

  You can reset the character set buffer used by the spool system by entering transaction SP12 (Temporary Sequential Objects Administration). Then choose *Character sets → CS conversion buffer CCC → Invalidate CCC*.

  You can check the SAP character set definition in transaction SPAD (Spool Administration). Check that the character set specified in the printer device type matches the printer character set as documented in the printer handbook. (The printer character set is usually specified in the *Printer initialization* action in *Device initialization*, transaction SPAD. Use the DINA4 or INCH12 *Device initializations* for the printer type in question as your guide).

See also SAP *Note 4367* for additional information.

Summing and Carrying Forward is Incorrect

Assume that for a multiple-page invoice, you want to print the current total as carry forward amount or subtotal on the current page and on the subsequent page. However, the carry forward amount is incorrect or missing.

The following causes are possible:

- You do not use program symbols with Dictionary reference for totalling.
- You place the carry forward amount into the BOTTOM area. SAPscript processes the BOTTOM area at the beginning of a page; therefore it is not suited for carry forward amounts.
- If you place the carry forwards amount into the TOP area of the main window on the subsequent page, the carry forward amount may be higher than it should be: This happens if the last part of text of the previous page does not fit onto the page, but the text is held together, for example, by a PROTECT command. In this case, a local text symbol must receive the carry forward amount.

You must place the carry forward amount on the current page into a window of type VAR. On the subsequent page, use a local text symbol [Seite 170] to print the amount in the TOP area of the main window:

1. At the beginning of the form main text (before printing the first text element), define the amount variable and the total variable (both must be program symbols or Dictionary amount fields).

   In the example below, we use the SUMMING [Seite 141] command to determine that for each output of &SUMTAB-AMOUNT& the system automatically sums up the amount in the total variable &SUMTAB-TOTAL&. At the end of the page, &SUMTAB-TOTAL& contains the carry forward amount of the current page or the grand total, respectively. In this example, we also define a local symbol &LASTPAGE& to print the grand total on the last page.

   ```
   /:  SUMMING &SUMTAB-AMOUNT& INTO &SUMTAB-TOTAL&
   /:  DEFINE &LASTPAGE& = ‘ ’
   ```

2. At the end of the form main text (when printing the last text element of the main window), set the local textsymbol &LASTPAGE& to a different value, such as ‘X’:

   ```
   /:  DEFINE &LASTPAGE& = ‘X’
   ```

3. To print the carry forward amount in the TOP area of the subsequent page including the pre-text 'Carry forward' and a tab, we use the local text symbol &CARRY. The corresponding text element is also defined in the main window:

   ```
   /E  CARRY
    *  &’Carry forward:,,’CARRY&

         (CALL FUNCTION WRITE_FORM EXPORTING ELEMENT = ‘CARRY’
                                             TYPE = ‘TOP’)
   ```

Summing and Carrying Forward is Incorrect

4. Define the carry forward window on the current page as type VAR and position it at the end of the main window of the page. Print the carry forward amount there and define the local text symbol &CARRY& to print the amount again in the TOP area of the subsequent page. Use the local text symbol &LASTPAGE& to print the grand total on the last page. The carry forward window then contains the following text (define it as paragraph T1 with tab):

```
/:    IF &LASTPAGE& = ‘X’
T1    <H>Grand total:,,&SUMTAB-TOTAL&</>
/:    ELSE
T1    <H>Carry forward:,,&SUMTAB-TOTAL&</>
/:    DEFINE &CARRY& = &SUMTAB-TOTAL&
/:    ENDIF
```

Using the SAPscript Form Debugger

Errors in the print program or the form that occur when printing in dialog mode (not when printing in background mode or in the update task) can often be identified quite easily by using the SAPscript form debugger.

1. Activate the debugger by choosing *Utilities* → *Activate debugger* in the *Form: Request* screen. You can then generate printer output in dialog.

   A dialog box containing the default breakpoints appears.

2. Track each call of a SAPscript function module, each INCLUDE call, and each warning in the debugger.

To switch off the debugger, choose *Debugger* -> *Exit* in the debugger window.

For more information on the debugger, see Note 19104.

Page Layout

For the page layout, the system offers two synchronized windows:

- The Administration Screen [Seite 86]

  On the administration screen, you can directly enter values (pages, windows, positions, and so on).

- The Design Window [Seite 90]

  On the design window, you graphically determine the layout of the different output areas.

Form Tree

Use

You use the form tree to store forms in structure nodes and thus classify them uniquely. You can then search for them more easily.

Prerequisites

You must classify the form either when you create it or at a later time. Any unclassified forms are stored in the node *Unclassified forms*.

Features

The classified form is stored in the form tree. By default, there are trees for all applications. If you need new trees and new nodes in a tree, you can create them in any system. For more information, see Creating Nodes [Seite 234].

You can display and suppress the technical names of the forms. You can use symbols for each form to display form information, to print the form for testing purposes, or to display information on the existing language versions or on the clients in which it exists.

You can add the following types of nodes:

- Form nodes for storing forms
- Structure nodes for creating new tree parts
- Reference nodes for adding existing trees for reference purposes
- Nodes for unclassified forms

Finding Forms

Finding Forms via the Form Tree

1. Choose *Tools* → *SAPscript* → *Form* and then *Form* → *Find* or directy use the possible-entries help of the *Form* field.

   The dialog window *SAPscript Form tree* appears.

2. Expand the nodes of the appropriate application and double-click on the desired form.

   If you want to use *Search via Characteristics* in the form tree, choose *Find* → *Technical search function*.
   To display the technical name of the form, choose *Additional information* → *Technical names on/off*.

   All forms that are not classified are stored under the node *Unclassified Forms.* They are sorted in alphabetical order by the form name.

Search via Characteristics (Generic Search for Forms)

1. Choose *Tools* → *SAPscript* → *Form* and then *Form* → *Find*.
2. Select *Search via Characteristics* and then choose *Continue*.

   The dialog window for finding forms appears.

3. Enter the desired values into the fields and choose *Execute*.
4. Double-click on the desired form to copy it from the hit list.

Classifying a Form

1. In the initial Form Painter screen select the desired form and the sub-object *Header*.
2. Choose *Change*.

   The dialog window *Change form header* appears.
3. Choose *Form* → *Classify*.

   The dialog window *SAPscript Form tree* appears.
4. Select the desired form tree and node.
5. Select *Insert at same level*.

   The dialog window *Valid node types* appears.
6. As node type select *Form node* and choose *Continue.*

   If you want to add several nodes under a node, select *Multiple nodes*. Enter the forms.
7. Choose *Continue*.

You cannot classify locally stored objects.

Creating, Renaming, and Deleting Nodes

Use

You can add the following types of nodes:

- Form nodes for storing forms
- Structure nodes for creating new tree parts
- Reference nodes for adding existing trees for reference purposes
- Nodes for unclassified forms

Nodes for unclassified forms can be created only once for each tree.

You can either add reference nodes locally in the current system or refer to them via a Remote Function Call (RFC) from other systems.

Procedure

Creating Nodes

1. In the initial Form Painter screen choose *Utilities* → *Classify multiple forms*.

   The dialog window *SAPscript Form tree (change mode)* appears.

2. Select the position in the tree where you want to add a new node.
3. If you want to add the node at the top of the tree, select the tree name and choose *Edit* → *Insert node* → *Insert one level down*.

   The dialog window *Valid node types* appears.

4. If you want to add a new node under an existing node, select the node and choose *Edit* → *Insert node* → *Insert at same level.*

   The dialog window *Valid node types* appears.

5. Select the desired node type and choose *Continue*.

   The dialog window *Ender node name* appears. If you want to create several nodes, select *Multiple nodes*.

6. Enter the new node name(s) and choose *Continue*.

Renaming Nodes

1. Select the node you want to rename and choose *Edit* → *Rename node*.

   The dialog window *Change node* appears.

2. In the *Title* field change the node name and choose *Copy*.

Deleting Nodes

Select the node you want to delete and choose *Edit* → *Delete node.*

Creating Reference Nodes

1. In the initial Form Painter screen choose *Utilities* → *Classify multiple forms*.

   The dialog window *SAPscript Form tree (change mode)* appears.

2. Select the position in the tree where you want to add a new node.

3. If you want to add the node at the top of the tree, select the tree name and choose *Edit* → *Insert node* → *Insert one level down*.

   The dialog window *Valid node types* appears.

4. If you want to add a new node under an existing node, select the node and choose *Edit* → *Insert node* → *Insert at same level.*

   The dialog window *Valid node types* appears.

5. Select *Reference nodes* and choose *Continue*.

   The dialog window *Reference to a structure* appears.

6. If you want to add a node that exists locally in the system, choose *Find structure*.

   The dialog window *Find structure* appears.

7. Enter the relevant search criteria and choose *Execute*.

8. In the hit list, select the desired structure and choose *Copy*.

9. If you want to add a node that was created in another system, choose *Find structure remote*.

   The dialog window *Enter R/3 connection* appears.

10. Select the desired system ID and choose *Execute*. Log on to the system and proceed as described in steps 6 to 8.

Administering Graphics

Use

Use the SAPscript *Graphics Administration* to import, administer, transport, and preview graphics before printing.

Features

In the navigation tree, you can go to any imported graphic. Up to Release 4.6, the graphics were stored as standard texts. This is obsolete now. As of Release 4.6, graphics are stored on a document server. In addition to *.TIF files, you can now import *.BMP graphics files as well.

Activities

Choose *Utilities* → *SAPscript* → *Administration* → *Graphic*.
The dialog window for administering graphics appears.

To import graphics, choose *Graphic* → *Import*.

Before you can continue with other functions, select an appropriate node in the navigation tree and specify the name and the attributes of the corresponding graphic. To find graphics, use F4 help.

To transport graphics, choose *Graphic* → *Transport.*

To display screen information, choose *Graphic* → *Screen information*.

To display the print preview, choose *Graphic* → *Print preview*.