combit List \& Label

## Designer Help

> Lᄂ List \& Label q®

No responsibility is taken for the correctness of the information contained in this manual. The information is subject to alteration without previous notice. combit GmbH accepts no liabilities in connection with this document. The availability of many of the functions described in this manual (e.g. the procedure for accessing the functions) is dependent on your system's version and release, the installed service packs (e.g. operating system, text processing software, mail program etc.) and the configuration of the system. If in doubt, please ask the person responsible for IT.
This manual or excerpts from this manual may not be copied or replicated in any other form (e.g. digital) without the written approval of combit GmbH.
PDF creation utilizes wPDF3 (c) wpCubed GmbH - www.pdfcontrol.com
Copyright © combit GmbH 1992-2011; Rev. 16.002 www.combit.net
All rights reserved.

ゅ
combit List \& Label

## Introduction

With the List \& Label Designer, you create or edit different print templates for printing information that originates either from a database or another data source. In the Designer, you then have all the data at your disposal and can prepare it for printing in different ways.
The Designer always uses the data that is passed from the program, in other words, the application handles the process of selecting and compiling the data.
Thus, "the Designer" is not a self-contained application, but rather is incorporated in List \& Label as an inherent part of an application.
The Designer $\rangle$ the print, preview and export dialog $\rangle$ is normally started via a menu item (e.g. Configuration > Print or Output > Print).

See also:

- General
- Screen

ゅ
combit List \& Label

## General

The different print templates are referred to as "Projects".
Along with the actual print information, a project also contains, above all, layout specifications such as page size and orientation, fonts, colors, frames, circles, lines, graphics etc. The List \& Label Designer can handle three different types of projects: Lists, labels and cards.
The individual elements of such a project are called "objects". For example, a project can contain text objects, picture objects and a report container.
The required objects are normally selected in the object list, created on the workspace with the mouse and then given the respective contents and layout properties.
The List \& Label Designer provides different types of objects which you can position freely in the workspace and change the size as required.

| (A) | Text and contents of variables |
| :--- | :--- |
|  | Lines |
| $\square$ | Rectangles |
|  | Circles and ellipses |
| Picture object |  |
| Barcode |  |
| Report container or Table object (depending on |  |
| the application): The Report container can hold |  |
| tables, charts and crosstabs. |  |

OLE container. For linking OLE documents (e.g. Word, Excel).
Form templates are placed in the background of the workspace as a template so that other objects can be aligned to them. Form templates are not included when printing.

ゅ
combit List \& Label

## Screen

The Designer interface consists of a workspace and different tool windows.
§ You add objects by means of the 'Objects' toolbar.
§ The objects are placed in the workspace. You change the shape and size of the workspace via Project > Page Setup.
§ You can select the viewing mode by means of the tabs in the bottom margin of the workspace: Layout, Layout Preview, Preview.
§ The status bar with the current mouse position, the current operation and the position/size.
§ The variable list displays all variables which are available in the current project; for list projects, all available fields are also displayed.
§ The layer list shows the different display layers.
§ The hierarchical object list shows all objects in the order in which they will later be printed.
§ The "Report Structure" shows all elements belonging to a report container in the order in which they will later be printed.
§ The properties of the selected objects or project are displayed in the properties tool window.

ゅ
combit List \& Label

## Getting Started

This chapter describes how to create a simple list based on the combit List \& Label Sample Application.
If you use List \& Label as part of an application, you can usually start the Designer via a menu item or similar. The functionality can differ slightly from the description below.

See also:
-The Sample Application

- The Sample Database
- Produce a Simple Report
- Producing a Simple Invoice
- Creating a Simple Label

ゅ
combit List \& Label

## The Sample Application

The Designer is not available as a self-contained application, but rather is incorporated in List \& Label as an inherent part of an application. For this reason, the Sample Application is used to create the examples.
You will find the Sample Application in the program menu of the List \& Label installation. Alternatively, you can download it as part of the combit List \& Label program installation at www.combit.net/en/reporting/.
You can then start the Designer via the "Design" menu item. Start the print, preview and export dialog via the "Print" menu item. Under options, you will find various configuration settings such as the language of the program interface or the number of sample files.


Figure 2.1: Sample Application

ゅ
combit List \& Label

## The Sample Database

The List \& Label Sample Application contains a Microsoft Access Database for a fictitious company called "Northwind" with sample data for a food wholesaler.
It is a relational database meaning that the data is distributed across several tables. Each table contains data elements that are related to one another. For instance, one table contains the product data. Another contains the product categories. A table's columns are referred to as fields (product name), and the rows of a table as data records.


Figure 2.2: Structure of the Microsoft Northwind sample database

ゅ
combit List \& Label

## Produce a Simple Report

For a first simple report, you will produce a product list in alphabetical order with a report title and a company logo.

1. Start the combit List \& Label Sample Application
2. Choose Design > Extended Samples to start the Designer.
3. A file selection dialog will appear. To start a new project, choose the New button.

Note: This button is not available in Windows XP. Enter a name for the new print template in the "File name" field and then choose "Open".

To open or edit an existing project, choose Open.


Figure 2.3: Open dialog
4. The standard project for the respective project type is displayed. This standard project is normally an empty workspace with a certain paper size and alignment.


Figure 2.4: Empty workspace
With label projects, a certain label format (size and arrangement of the individual labels on the sheet) is already specified as a default value.
Note: A standard project is a standard template for creating projects. However, you can change it to suit your needs and save it again under the name "Default".
5. If you have chosen "New project wizard" under Project > Options > Workspace the project wizard will start. The project wizard simplifies the job of creating new projects by leading you through different page setup options. But you will not be using the wizard for this example.

See also:

- Inserting a Company Logo Into the Report
- Adding a Title to the Report
- Insert the Table for the Product List
- Formatting Table Fields
- Displaying a Preview of the Report
- Adding a Page Number
- Print or Export the Report


## Inserting a Company Logo Into the Report

For the company logo, please use the "sunshine.gif" file, which you will find in the directory for the Sample Application. We also need a "Picture" object. Proceed as follows:

1. In the "Objects" toolbar, choose the "Picture" object type.

Figure 2.5: "Objects" toolbar
Note: Objects are your project's building blocks. They are generated in the workspace where they are also given a border with which their size and position can be changed. This border defines the space that the object takes up and thus also the maximum size to which the contents of the respective object can be expanded. Objects may overlap fully or partly.
2. In the workspace, point the mouse to the position where the upper left corner of the object is to begin. The mouse cursor changes to a crosshair. Hold down the left mouse button and drag the crosshair to the lower right corner of the planned object. Release the mouse button when the object (the dashed border) is the right size.

Note: Objects can be added to the workspace in different ways: via the menu Objects > Insert, via the toolbar or via keyboard shortcuts, or with Drag \& Drop from the list of variables.
3. A file selection dialog will appear. Select the image file that you want.

Note: The following formats are available: WMF, EMF, BMP, DIB, PCX, SCR, TIFF, GIF, JPEG, PCD, PNG and ICO. As a general rule, you should use the RGB color space (not CYMK). List \& Label itself supports transparency in PNG files by using the corresponding Windows functions. In our experience the majority of printer drivers do not support transparency so that reports with e.g. partly transparent PNG files should thoroughly be tested on the actual hard-software combination. If that is not possible we recommend doing without the alpha channel. In addition, you can insert the picture into the project by enabling the ("Embed image in project file" checkbox option. This option copies the image to the project thus making it available even without the external file.
4. After you select the file, the logo will be inserted.


Figure 2.6: Report with picture object

## Adding a Title to the Report

Use a text object to add a title to the report. Text objects let you place text in the workspace. As well as fixed text, you can also insert the contents of fields (variables) from the database (e.g. company name), or you can use functions (page number, date etc.).

1. In the "Objects" toolbar, choose the "Text" object type.

Figure 2.7: "Objects" toolbar
2. In the workspace, hold down the left mouse button and pull the object to the required size.
3. The formula wizard will now appear which you can use to define the contents of the text object.
This dialog consists of a series of tabs each containing different elements to be edited. The following chapters explain the meanings of these elements in more detail. § Variables and Functions: the available variables, fields and functions.
§ Condition: for defining IF-THEN-ELSE conditions.
§ Text: for entering fixed text and tabs.
§ Date Format: different date formats.
§ Number Format: different number formats.
§ Operators: available conjunction operators.
You can also enter the expression that you want directly in the edit box or modify the text that is there (e.g. put something in brackets).
Therefore, enter our title "List of products" directly. Fixed text must be enclosed in quotation marks. Enter names of variables and functions without brackets.


Figure 2.8: Text in the formula wizard
Note: Please note that there are two ways of writing expressions, depending on their use. You will find more information about this in the chapter Variables, Formulae and Expressions.
4. The title will be displayed when you close the dialog with OK.
5. Enlarge the letters by changing the font size in the mini toolbar that appears automatically.


Figure 2.9: Formatting menu in mini toolbar
6. You have now added the title:


Figure 2.10: Report with text object

## Insert the Table for the Product List

Use the "Report Container" object to add a table to the report. As the name says, a report container can hold several objects: tables, charts and crosstabs can be added in any order.

Note: The report container is not available in all applications. In applications that don't have the report container, use the "Table" object.

1. In the "Objects" toolbar, choose the "Report Container" object type.

Figure 2.11: "Objects" toolbar
2. In the workspace, hold down the left mouse button and pull the object to the required size.
3. A selection dialog will appear for the chosen object type.

Choose the "Table" object type.


Figure 2.12: Choosing the object type
4. Now supply the data source in the following dialog. All available tables are shown hierarchically; in other words, under the tables, you will find the relational tables in each
case.


Figure 2.13: Choosing the data source
Select the "Products" table because it contains the fields that we want for our product list.
5. A selection dialog will appear with all the fields in the "Products" table. In addition, underneath the "Products" table, you will also find the fields in the tables "Categories" and "Suppliers" which have a 1:1.relationship with
"Products".


Figure 2.14: Field selection dialog
In this dialog, now choose the columns for the table. Double-click a field, e.g. "ProductName". The field will be added to the "Columns" area.
6. Repeat this step for all fields that are to be shown in the table; i.e. also the "CategoryName" field from the linked table "Categories" and the "UnitPrice" field for the unit price. Confirm your selection with OK.
7. The table will now be displayed in the workspace.
§ The selected fields are displayed in the data line, in other words, the data line contains the data.
§ In addition, a header line is automatically produced. Header lines are used mostly as column titles, i.e. the selected field names are now shown here as text.
§ If you pull the report container widthwise to make it wider or narrower while holding down the CTRL key, the columns will be adjusted proportionally to fit.
§ List \& Label adjusts the width of the columns automatically. You can adjust the width of a column manually by moving the separating line to the right or the left with the mouse.

Note: This changes all table columns, whose separators are within $+/-2 \mathrm{~mm}$ from the mouse position. If you hold down the Ctrl key, the action will only be carried out for the line on which the mouse is positioned. If the "Column width modification affects next column" option is enabled (Project > Options > Workspace), you can alter the column width while making the next column smaller.


Figure 2.15: Report with table in the report container object

## Formatting Table Fields

Chapter Creating Professional Tables describes how you can format and modify a table. Only a few basic formatting options are explained here.

1. In the product category column title, the field name "CategoryName" is shown. Click the respective field in order to change this text to "Category". The formula wizard will now appear, which you can use to change the contents of the field. Please consider that text must be enclosed in quotation marks.


Figure 2.16: Formula wizard with text input
2. In addition, we want to format the unit price as currency and align the entire column to the right. You will find both formatting options in the mini toolbar that appears automatically. Click in the upper left corner to select the field containing the unit price.


Figure 2.17: Selecting a single field
3. Click the button for the formatting dialog


Figure 2.18: Formatting a field directly
4. A selection dialog will now appear in which you can select the formatting that you want for the numeric field. To do this, choose the type on the left hand side i.e. "Currency". In the right pane, you can specify the currency formatting in detail. The settings for the relevant application will be used in each case as standard. Alternatively, you can choose the system setting or a user defined setting.

| [1] Formst |  |  | $\underline{\square}$ |
| :---: | :---: | :---: | :---: |
| Thoformat | Dscimal places: | Appleation setting | - |
| Cuntency | Decimal pgint | Application setting | - |
| Date <br> Time <br> Date and Time <br> Percentage <br> Angle <br> Date-Tme Difference | Thourands reparator: | Applacation setting | - |
|  | Currency yymbol | Application setting | * |
|  | Format for positive values: | Applcation setting | - |
|  | Format for pegative values: | Application setting | - |
|  | Exponential format:No output if yalue is $\theta$Always a before decimal point | No | - |
|  |  |  |  |
| Preview |  |  |  |
|  | (\$12.35.67) |  |  |
| (2) |  | OK | Cancel |

Figure 2.19: Formatting dialog
5. Finally, select both fields (the header and the data line) to align them to the right. To do this, hold down the Ctrl key and select both fields by clicking in the upper left corner in each case. Then click the button for right alignment.


Figure 2.20: Multiple field selection

## Displaying a Preview of the Report

Until now, you have only seen the report in layout view as a structure with a record. To get an impression of the result, you can display a preview of the report. Use the tabs in the bottom margin of the workspace to change to the preview.
Note: Real data preview mode is not supported by all applications.

## Clayout Layout Preview

Figure 2.21: "Preview Options" toolbar
The report will now be displayed with the data from the "Products" table. You can also change the number of "products" or data records in the Sample Application.

1. To do this, save the report with File > Save.
2. End the Designer with File $>$ Close.
3. In the Sample Application, choose Options > Settings and increase the maximum number of root records to 50 .
4. Open the print template that you created again via Design > Extended Samples, and switch to preview mode. Your report will look roughly like this:


Figure 2.22: Real data preview

## Adding a Page Number

It's a good idea to add a page number in the lower area of the page. To do this, add a new text object. You will be using functions (such as the page number function) as well as fixed text with this object.

1. In the "Objects" toolbar, choose the "Text" object type.

Figure 2.23: "Objects" toolbar
2. In the workspace, hold down the left mouse button and pull the object to the required size.
3. The formula wizard will now appear which you can use to define the contents of the text object. The available functions are shown in the right pane. You can use an auto filter with this list. Type "page" in the filter field. This will cause all functions containing the expression "page" to be displayed.
§ The "Page\$ ()" function returns the page number.
§ The "TotalPage\$ ()" function returns the total number of pages.
Add the "Page\$ ()" function to the result area by doubleclicking.
4. If you now want to output a footer in the form of "Page 1 of 2 ", you can enter the text "of" directly in the usual way. Please consider that individual parts must be joined with a "+" and that text must be enclosed in quotation marks.
5. Now add the "TotalPages\$ ()" function by double-clicking to get the total number pages. The formula will look like this:


Figure 2.24: Formula wizard with functions and text
6. You can center this line in the report in the usual way:


Figure 2.25: Quick format "centered" alignment
7. Switch to preview mode to view the result:


Figure 2.26: Real data preview in the Designer

## Print or Export the Report

You can start the print function directly from the higher-level program, from the real data preview or via File > Export.

1. Choose File > Export.
2. The Print Options dialog will appear.
§ Under "Print target", you can change the printer or the printer configuration.
§ Select the output format (e.g. preview, printer, Excel) under "Direct to".


Figure 2.27: Output options
3. If you have selected an export format (such as PDF), choose the storage location in the following "Save As" dialog and enter a name for the file to be created in the "File name" field.


Figure 2.28: Save dialog when exporting to a different format

ゅ
combit List \& Label

## Producing a Simple Invoice

In this chapter, you will now meet functions and possibilities for designing reports.
For the next example, you will create a multi-page invoice with covering letter, invoice header, totals, footers, enclosure and printed copy.

See also:

- Create a New Print Template
- Adding a Company Logo
- Add the Address Field
- Create Invoice Header With Number and Date
- Adding the Item List
- Alignment and Formatting
- Background Color and Frame
- Invoice Footer With Totals
- Create Additional Footer Lines


## Create a New Print Template

1. Start the combit List \& Label Sample Application (see Chapter The Sample Application).
2. Choose Design > Invoice to start the Designer.
3. A file selection dialog will appear. To start a new project, choose the New button.

Note: This button is not available in Windows XP. Enter a name for the new print template in the "File name" field and then choose "Open".
To open or edit an existing project, choose Open.

## Adding a Company Logo

Use the picture object to add a logo.

1. In the "Objects" toolbar, choose the "Picture" object type.

Figure 2.29: "Objects" toolbar
2. Pull the object to the right size and select an image file (see Chapter Inserting a Company Logo Into the Report).

## Add the Address Field

To add an address, use the text object. Text objects let you place text or the contents of fields in the workspace.


Figure 2.30: Address field: it should look like this.

1. In the "Objects" toolbar, choose the "Text" object type.

Figure 2.31: "Objects" toolbar
2. Pull the object to the required size.
3. The formula wizard will now appear which you can use to define the contents of the text object.
This dialog consists of a series of tabs. On the "Variables and Functions" tab, select the variable for the company address (company) from the list of available variables and fields.
You will see an auto filter field above the list of variables. This means that you can enter "Company" to display all fields and variables containing this expression. Select the variable that you want by double-clicking and confirm your selection with OK. You have now defined the first line of the address field.


Figure 2.32: Formula wizard with variable
4. A text object can hold as many paragraphs as you want and they can all have completely different display properties. You can add more paragraphs by means of the mini toolbar which appears automatically. Choose "Append paragraph" to define an additional line.


Figure 2.33: Adding another paragraph via the mini toolbar
5. The formula wizard will now appear, which you can use to define the contents of the paragraph. Now enter the first name and the last name of the recipient. First choose the "Firstname" variable.
6. You should insert a space before choosing the "Lastname" variable to prevent the contents of the two variables from being placed end to end. A space is simply "Text". Fixed text must be enclosed in quotation marks. So now enter "+" as a joining operator followed by " " for the space.
7. Now select the "Lastname" variable. You must naturally also join this with " + ". The whole line looks like this:


Figure 2.34: Linking variables and text
8. Now continue with the other variables: street and city. You have now completed the address field.


Figure 2.35: Text field with 4 paragraphs

## Create Invoice Header With Number and Date

Use the text object again to add the text "Invoice", the invoice date and the invoice number to an invoice header.


Figure 2.36: Invoice header: it should look like this.

1. In the "Objects" toolbar, choose the "Text" object type.
2. Pull the object to the required size.
3. The formula wizard will now appear, which you can use to define the contents of the text object. Type "Invoice".
4. Add another paragraph by means of the mini toolbar that appears automatically and the "Append paragraph" (insert) button.
5. The formula wizard will appear. Type "Invoice date".
6. Before you enter the invoice date, insert a tab stop so that the invoice date is aligned to the right. You can insert a tab stop by means of the "Text" tab. You create a tab stop with the "Tab" button and you define the position and alignment with the "Properties" button.
Note: Only one tab stop can be inserted on each line. A tab stop causes the preceding text to run only as far as the tab stop. A tab stop that is right aligned will cause the text that follows it to be justified to the right. The distance from the left margin determines the position of the tab stop.

Since a tab stop is also a character, it must also be
enclosed in quotation marks. Alternatively, in this example, you can insert the tab stop with "Insert" to the existing "Invoice date" text.


Figure 2.37: Adding a tab stop to a text expression
7. Finally, position the cursor outside of the text field and insert the "Invoice_Date" variable with the "Variables and Functions" tab.
The format will be converted automatically.
Conversion with the "Date\$()" function is essential since the invoice date has the "date" type and the value that is returned is already defined as a "string" by the "Invoice date" text expression entered earlier. The return value must always be unique. The date value must therefore be converted to a string by means of a function. The "Date\$()" function is one way of doing this.


Figure 2.38: Format conversion with Date\$()
The automatic format conversion uses "\%x" as the formatting parameter. If you want a different output format, switch to the "Date Format" tab. You can choose a format here and then select the "Invoice_Date" variable.
8. Now add the invoice number in the same way.

The format will be converted automatically here as well.
|Number. - $^{*}+$ str 5 (Irvoice_Number, 0,0 )
Figure 2.39: Format conversion with Str\$()
Conversion with the "Str\$()" function is essential since the invoice number is a field with the "number" type and the value that is returned is already defined as a "string" by the "Invoice number" text expression entered earlier. The return value must always be unique. The number must therefore be converted to a string by means of a function. The "Str\$()" function is one way of doing this. This function has 3 parameters: The first parameter is the number that is to be converted to a string; the second parameter specifies the minimum length and the third parameter the number of decimal places.
9. Now increase the font size for "Invoice". To do this, select the relevant line.
10. Set the font size to 28 by means of the "font size" button in the mini toolbar that appears automatically.
11. Now increase the distance to the next line, i.e. open the paragraph properties dialog via the "Object properties" button in the mini toolbar. In this dialog, you can define the properties for each line. Set the value for the "Paragraph Spacing" property for the first line to 12 pt .


Figure 2.40: Paragraph properties dialog
12. And you now have an attractive invoice header:


Figure 2.41: Invoice header

## Adding the Item List

Use the "Report Container" object to add the item list to the report. As the name says, a report container can hold several objects: tables and freely defined content, charts and cross tabulations. You need the "Table" element for the item list, and "Free content" for the covering letter. Please see chapter Insert the Table for the Product List for an introduction about how to create tables.

1. In the "Objects" toolbar, choose the "Report Container" object type.


Figure 2.42: "Objects" toolbar
2. In the workspace, hold down the left mouse button and pull the object to the required size.
3. A selection dialog will appear for the chosen object type.

Choose the "Table" object type.


Figure 2.43: Object type selection
4. Now select the data source in the following dialog. All available tables are shown hierarchically; in other words, under the tables, you will find the related tables in each case.

Select the "Items" table because it contains the fields that we want for our item list.
5. A selection dialog will appear with all the fields in the "Items" table.
6. In this dialog, now choose the columns for the item list. Double-click one after another on the fields: "Quantity", "No", "Description1", "UnitPrice". This will add the fields to the "Columns" area. Confirm your selection with OK.


Figure 2.44: Field selection dialog
7. The table will now be displayed in the workspace. A header line with strings as the column titles and a data line with the contents of the fields will be created automatically.
List \& Label adjusts the width of the columns automatically. You can adjust the width of a column manually by moving the separating line to the right or the left with the mouse.
Note: This changes all table columns, whose separators are within $+/-2 \mathrm{~mm}$ from the mouse position. If you hold down the CTRL key, the action will only be carried out for the line on which the mouse is positioned. The "Column width modification affects next column" option (Project > Options > Workspace), lets you alter the column width while making the next column smaller.


Figure 2.45: Changing column widths with the mouse
8. In the next step, you add a new column: "Total". Select the "UnitPrice" field by clicking in the upper left corner of the field. Then choose the "Append column" button in the mini toolbar.


Figure 2.46: Select column
9. A menu will appear for the selected field type. Choose "Text" here.
10. The formula wizard will now appear, which you can use to define the contents of the field. Type "Total" and confirm with OK.
A message will now appear giving you the option of
reducing the width of the columns to allow the new column to be inserted in the visible area. Confirm this dialog. You have now added an additional column to the header line. You now need this column in the data line as well.
11. Now select the data line field containing the item price and add a new column to it.


Figure 2.47: Add a new column
The total price is now to be calculated here. The total price is calculated by entering "Quantity*ItemPrice". To do this, select the relevant variables by double-clicking them.


Figure 2.48: Multiplying two values in the formula wizard
Note: Open the list of available operators by clicking the "Operators" tab. Operators join two or more values or variables to give a new value. In this way, you can formulate arithmetic expressions (basic arithmetic operations) or logical expression. You can combine multiple operations in one expression. Please consider the calculation hierarchy of the operators used and place
them in brackets if necessary. The "+" operator has a special meaning. It is not only suitable for additions ("Number" and "Date" value types) but is also used for joining strings.

## Alignment and Formatting

Make use of the numerous formatting options to present the invoice in the form that you want. For example, change the column title and format the amounts as currency.

1. Click the column title that you want to change. The formula wizard will now appear, which you can use to change the contents of the field. Please consider that text must be enclosed in quotation marks.
2. In addition, you should also align the columns for the unit price and the total price with the column titles. To do this, hold down the Shift key and select all 4 fields by clicking in the upper left corner of the "Item Price" data line. Then choose the button for right alignment in the mini toolbar.
3. To format both amounts as currency, hold down the CTRL key and select both fields by clicking in the upper left corner of each one. Then choose the button for the formatting dialog.


Figure 2.49: Multiple field selection
4. A selection dialog will now appear in which you can select the formatting that you want for the numeric field. To do this, choose the type on the left hand side i.e. "Currency". In the right pane, you can specify the currency formatting in detail. The settings for the relevant application will be used in each case as standard. Alternatively, you can choose the system setting or a user defined setting.


Figure 2.50: Formatting a field
5. Format the field for the quantity as a number without decimal points.
6. Our item list will now look roughly like this:


Figure 2.51: Preview

## Background Color and Frame

In order to structure the invoice and make it more attractive, it's a good idea to give at least the column headers a colored background. You can also modify the background color.

1. To do this, select a column title.
2. Now open the paragraph properties dialog via the "Object properties" button in the mini toolbar. In this dialog, you can define the properties for each column and row. Hold down the CTrl key and select all columns.
3. Click the "+" sign to expand the "Background" property group.
4. For the background, choose the "Pattern/block color" property and the color, e.g. LL.Color.LightSkyBlue. You reach the LL.Colors via the arrow button and the standard color dialog via the ... button.


Figure 2.52: Object properties dialog for the table: background
5. In addition, you want to increase the spacing between the header and the data line. Since this is a line property (and not a column property), select the line.

Here you have the "Spacing" property. Choose e.g. 0.1 inch as "Bottom" to increase the spacing below the column title.


Figure 2.53: Object properties dialog for the table: margins
6. Now it only remains to make the settings for the frame. You can define different settings independently for each cell. However, it's normally a good idea to make the settings for the entire table. For this reason, the settings for the frame are included in the table properties. You will find the table properties, as standard, on the lower left when you select the table in the "Report Container" tool window. When you select the property, there is a button which you can use to open the dialog for the frame properties.


Figure 2.54: Frame properties
7. Choose "white" as the color and click "Outline" to apply the color. Please note: If you don't apply the setting but simply confirm the dialog with OK, your settings will be lost!
8. The invoice will now look roughly like this in the preview:


Figure 2.55: Preview (frame and background)

## Invoice Footer With Totals

You now need to output the net totals, the VAT and the total amount in the invoice footer. You can simply insert these values if they are held as fields in the database. In our case, however, you must first calculate the values. To do this, you use the "Sum()" aggregate function. This function totals the values that are passed to it.
Since the totals are to be output at the end of the table, you define a footer for this purpose.

1. Click the "Append Line" button in the mini toolbar and then click "Footer".


Figure 2.56: Add new line
2. You have not yet defined a line for this line type so you will be asked whether you want to use an existing line definition from one of the other line types. This wouldn't save any time at this point. Therefore, choose the "Single field or free content" option.


Figure 2.57: Applying a line definition
3. The formula wizard will open to allow you to define the contents of the first column. Enter "Subtotal", not forgetting to enclose it in quotation marks.
4. Once you have created the first line, add another column to it for the aggregation. Use the "Sum()" function for aggregating values. You can either type it in directly in the result field in the formula wizard or you can select the function from the list of aggregate functions by doubleclicking. As a parameter, enter the value to be aggregated ) in this case Quantity*ItemPrice.


Figure 2.58: Totaling with Sum()
5. The entire line should now be moved a little to the right. Hold down the CTRL key and drag the column separator on the extreme left at the outer edge of the table to the right. An empty column will be inserted automatically as a spacer on the extreme left.
6. You must also format the total field as currency, give both columns the same background color as the header line and set the "Top" margin to 0.1 inch to correspond with the "Bottom" margin of the header line.
7. As these footers are only to be output on the last page, even in the case of multiple page invoices, you must now specify the "Lastpage()" or "LastFooterThisTable()" function for the "Appearance Condition" line property.


Figure 2.59: Object dialog with 3 columns
8. You have now defined the first footer and the invoice looks like this:


Figure 2.60: Preview (footer)

## Create Additional Footer Lines

You can create as many line definitions as required for each line type (data line, footer). The different line definitions are shown in the object dialog as a tree structure.
For our example, now create two more footers for the VAT and the grand total.

| Subtotal | $\$ 5.638 .00$ |
| :--- | ---: |
| $19 \%$ vai | $\$ 1,071.22$ |
| Total | $\$ 6.709 .22$ |

Figure 2.61: Preview 3 footers

1. To add more lines in the object dialog, first select an existing line definition. You can then create a new line definition with the "Append line definition" button.


Figure 2.62: Object dialog with 3 columns
Alternatively, you can add a new line in the workspace via the context menu or with the mini toolbar.
2. A dialog will appear asking you whether you want to use
an existing line definition. It saves some time if you use the footer that you have already created as you will not have to configure the left margin, alignment or background colors. Therefore, choose the "Line definition 1" entry from the "Footer" area.
3. The line will now be inserted. Click the "Subtotal" field so that you can then change the content to "VAT 19\%" with the formula wizard. Don't forget to enclose the text in quotation marks.
4. Once you have created the column, click in the column to enter the aggregation. Instead of the total net amount, the $19 \%$ VAT is to be output here. Therefore, you simply multiply the amount by 0.19 .
Sum(Item.Quantity * Item.UnitPrice * 0.19)
5. Proceed in the same way with the third footer. Copy the existing footer, change the content and, in the formula, add the VAT to the total, like this:
Sum(Item.Quantity * Item.UnitPrice * 1.19)
6. You have now defined the first footer and the invoice looks like this:

| ADC Businessinc Dese Me, Jotn Sith 5 .ABC Btad. Spring field 12345 | ess inc In Sath <br> 12345 |  | age |  |
| :---: | :---: | :---: | :---: | :---: |
|  |  |  | Invoic Onte Number: | ice $\begin{aligned} & 7 / 20010 \\ & 47110015 \end{aligned}$ |
| Quantity | No | Descrisiont | Unitprice | Itax |
| 1.00 | Expsans | Southen Atica 5 xplowr 20 -day Sour tom Case Town to Wetoris Fals exduding itht | \$1,500.00 | \$1,50000 |
| 200 | EXPMALOT | Moldives diving trip. 14 dyys, southern Mate Ropl, Promdse Bosch ${ }^{* 2 \pi}$ exduding figt | \$1/800000 | \$ 33,80000 |
| 200 | EXPYucos | Wucatsn, Onthe Tral ofthe Mays, 2-veek round stip, pacludrg tight | 51,200,00 | \$2,40000 |
| 100 | TRPPRADA | From Gotricto Broque to Ait Nouvesu, Cey trip to Progn inctuding scopinnoditén | \$35500 | \$355.00 |
| 100 | truerant | flort onh traze | \%20300 | \$763.00 |
|  |  | $\begin{aligned} & \text { Satatal } \\ & \text { 19\% vat } \\ & \text { tatal } \end{aligned}$ |  | $\begin{aligned} & \$ 5,630.00 \\ & \$ 1,071.22 \\ & \$ 0,709.22 \end{aligned}$ |

Figure 2.63: Invoice with 3 footers

ゅ
combit List \& Label

## Creating a Simple Label

For your first label, you will now create an item label.

1. Start the combit List \& Label Sample Application
2. Choose Design > Item Labels to start the Designer.
3. A file selection dialog will appear. To start a new project, choose the New button.
Note: This button is not available in Windows XP. Enter a name for the new print template in the "File name" field and then choose "Open".
To open or edit an existing project, choose Open.
4. The standard project for this project type is displayed. This standard project is normally an empty workspace with a certain paper size and alignment.
5. Select a label format via Project > Page Setup. In the "Templates" tab, choose the template that you want from the predefined label formats of various manufacturers. This specifies automatically the size of the individual labels, how many are to appear on the sheet and how they are to be distributed. You will find more information about this in the chapter Templates for Label Formats.


Figure 2.64: Specify label format
6. A single label will now be displayed in the workspace.

See also:

- Insert Object
- Printing Labels


## Insert Object

A picture of the item is to be displayed as a picture object on the label together with two text objects (Item_No, Description and Price). Place a tab stop between the text "Item No." and the "Item_No) variable.


Figure 2.65: It should look like this

1. Begin by positioning the text object. In the "Objects" toolbar, choose the "Text" object type.

Figure 2.66: "Objects" toolbar
2. Pull the object to the required size.
3. The formula wizard will now appear, which you can use to define the contents of the text object.
4. Now follow the steps described in Chapter Add the Address Field onwards.
5. Use the picture object to add a logo. Drag the "Picture"
variable from the list of variables into the workspace. Pull the object to the required size.

## Printing Labels

You can start the print function directly from the higher-level program, from the real data preview or via File > Export.

1. Choose File > Export.
2. The output options dialog will appear.
§ Under "Print target", you can change the printer or the printer configuration.
§ Select the output format (e.g. preview, printer) under "Direct to".
3. The "Select" button lets you specify the starting position when printing the sheet of labels. In this way, you can also print sheets of labels that have already been partly used. You will find a sample label sheet for your label project in the dialog for selecting the start position. Click the label where the print is to start.

ゅ
combit List \& Label

## Effective Workspace Techniques

This chapter will provide you with useful information and the most important techniques for working efficiently with the Designer.

See also:

- General Procedures
-Arrangement and Alignment of the Objects

ゅ
combit List \& Label

# General Procedures 

See also:<br>- Choosing a Page Layout<br>- View Mode<br>- Status Line<br>- Mini Toolbar<br>- Default Settings for Font and Frame<br>- Real Data Preview

## Choosing a Page Layout

The first task in a new project is to set up the page layout that you want. Choose Project > Page Layout to specify properties such the choice of printer, paper size and orientation.
If multiple layout areas are defined, the active workspace can be selected via the "Active Design Layout" project property.

## View Mode

You can select the view mode by means of the tabs in the margin of the workspace:


Figure 3.1: Tabs for selecting the view mode
§ In layout mode, you merely see the object frame and the contents of the objects as formulas.
§ The layout preview shows the objects in the WYSIWYG format. In addition, the objects are transparent and are drawn in the color of the layer to which they are assigned.
§ Preview mode display is a real data preview, provided that this is supported by the application.

## Status Line

The status line is divided in three sections.
§ The current mouse position from upper left.
§ The active operation (e.g. selection).
§ Name, position upper left, position lower right, width, height and layer of the selected object.
$8.635,14.339$ Select $\quad$ Text $-0.72 \mathrm{Sin}, 1629 \mathrm{in}-3.52 \operatorname{din} 2.709 \mathrm{in}=2.795 \mathrm{in}, 1060 \mathrm{in}$ (Base)
Figure 3.2: Status line

## Mini Toolbar

When you select a text object or a table object, a convenient toolbar in miniature format is displayed, the mini toolbar. You can close the mini toolbar by pressing Esc.
With the mini toolbar you can apply fonts, font sizes, orientations, text colors and formatting as well as open the object dialog.
Additionally with text objects: Add paragraph. Right mouse click: Add paragraph before the last selected paragraph.
Move paragraphs upwards/downwards.
§ To select a complete paragraph, click onto the bar on the left. Hold Ctrl or Shift to select multiple paragraphs or a complete range.


Figure 3.3: Mini toolbar for text objects
Additionally with table objects: Define new line, insert new row, move selected rows/cells to the left/right, borders.
§ To select a complete line, click onto the bar on the left. Hold Ctrl or Shift to select multiple paragraphs or a complete range.
§ To select a field, click in the top left corner of the field. Hold Ctrl or Shift to select multiple fields or a complete range.
§ To select a complete column, hold Alt. Hold Ctrl or Shift to select multiple columns or a complete range.


Figure 3.4: Mini toolbar for table objects

## Default Settings for Font and Frame

Choose Project > Options > Objects.
The "Select" button under "Object font" lets you choose the default font to be used for objects. Under "Color preferences", you can specify the border and the filling for objects.


Figure 3.5: Project options for objects
New objects will be created according to these settings. However, you can change the settings later individually for each object.
When you start a new project, it's a good idea to configure these settings using suitable values to keep the effort required for making manual changes to a minimum. The settings only apply for the current project.
The "Default" parameter in the respective object properties (Font/Frame/Background) is active as standard for all objects.

Therefore, if the default values are changed later, this will also affect all objects whose font has not been changed manually.

## Real Data Preview

The real data preview function is also available directly in the Designer provided that your application supports it. In this way, you can check the layout of a printout without wasting paper in order to do so. The screen preview is accurate down to the last detail (WYSIWYG What You See Is What You Get), exactly as it would be when printed. After checking the layout, the actual print can be started from the preview without having to use the print command again. Also see chapter Output Options.


Figure 3.6: Real data preview in the Designer

ゅ
combit List \& Label

## Arrangement and Alignment of the Objects

Objects are your project's building blocks. They are generated in the workspace where they are also given a border with which their size and position can be changed. This border defines the space that the object takes up and thus also the maximum size to which the contents of the respective object can be expanded. Objects may overlap fully or partly.

See also:

- Arrangement as an Object List
-Displaying the Alignment Grid
- Aligning Objects
- Guides in the Workspace
- Size and Position of Objects
- Grouping of Objects
- Multiple Copies of Objects


## Arrangement as an Object List

The object list defines the implicit print order of the objects, thus they are automatically sequentially linked to one another. Unlinked objects are printed first, followed by the tables and finally the objects that are linked in any other way.
The most important commands for the respective objects are available via context menus as in the work space. Objects that are in a hidden layer have parentheses around the rectangle in front of their names. Locked objects are identified by a small X .


Figure 3.7: Print order in the object list

## Displaying the Alignment Grid

Choose Project > Options > Project. Here, the "Show grid" option lets you specify whether a grid is to be shown in the background of the workspace to assist with orientation. You can specify the spacing of the grid lines in each case. The "Horiz./vertical synchronized" option causes the same grid spacing in both directions.

## Aligning Objects

Via the Objects $>$ Arrange $>$ Alignment menu item, or the corresponding toolbar, you can align multiple objects with one another. At least two objects must be selected for the function to be enabled.

Figure 3.8: Toolbar for the alignment functions
§ Left, right, top, bottom: The selected objects are aligned to the border of the selection rectangle in which they are enclosed.
§ Centered: The selected objects are centered in the respective direction (horizontal or vertical) within the selection rectangle.
§ Size adjustment: The frames of the selected objects are adjusted to a common size in the respective direction (horizontal or vertical).
§ Equal shape distance: The distance between the edges of the selected object is set to be equal. The outer edges in each case are significant here. The average distance is determined relative to the edges of the objects and the objects are arranged accordingly. In the event that the selected objects overlap, i.e. the intermediate area has a negative value, the function will not spread the objects apart but instead, only the degree of overlapping will be adjusted to the average value.
§ Equal center distance: The distance between the centers of the selected object is set to be equal. The average distance is determined relative to the centers of the objects and the objects are arranged accordingly. In the event that the selected objects overlap, the function will
not spread the objects apart but instead, merely the degree of overlapping is adjusted to the average value.

## Guides in the Workspace

You can create horizontal and vertical guides. To do this, position the mouse on a ruler, press and hold the mouse button and release it again in the workspace. The new guide is aligned in the same direction as the originally selected ruler. You can then move the guides around as you wish.
A catch function helps you to position objects directly on the guide. This function does not connect the objects permanently to the guide but merely helps you when positioning objects.
If you hold down the CTRL key when you move the guide, the objects connected to the guide will be moved as well.
The options relating to guides are defined via a context menu which appears when you click the right mouse button. This lets you choose a catch range in pixels for each guide. When you come within this range, the object will be drawn to the guide. The catch function is switched off if you hold down the Ctrl function when working with objects.


Figure 3.9: Context menu for guides
You can fix the guides in the workspace and prevent them from being moved by mistake. You can also specify the position of the guide directly.

## Size and Position of Objects

You can move each selected object or change its size. If you select multiple objects, you can change them all as if a single object were selected.

1. Changing the size with the mouse: Select the object. If you position the mouse on the frame, you can change the size by pulling the frame inwards or outwards, as indicated by the arrows, while holding down the left mouse button. You must position the mouse in one corner of the frame in order to change the size both horizontally and vertically.
2. Moving with the mouse: Select the object. Hold down the mouse button and drag the object to the position that you want. If you press the Shift key, you will only be able to move the objects horizontally or vertically. The alignment remains the same.
3. Changing the size and moving with the dialog: You can also change the size and position of an object by means of the property list. You can enter values precisely here. If you double click the "Position" sub-item in the property list, a position dialog will appear which makes it even easier to enter the size and position of objects with the keyboard.
4. Using the keyboard to move objects or change their size: Select the object. Use the direction keys to move the object in the respective direction. Pressing the key once moves the object by the smallest possible unit, if you hold down the Ctrl key, the object will be moved by ten times the smallest unit.

## Grouping of Objects

You can group multiple objects that belong together and then modify them as if they were a single object. Please note that an object can only belong to one group. It is therefore not possible to combine groups to a higher-level group. In order to make a group of two or more objects, select the objects in question and then choose Group in the context menu.
To remove the grouping, choose the item Ungroup.

## Multiple Copies of Objects

If you want to place several, similar objects with the same distance on the workspace, select the object and then choose Multiple Copies in the context menu. Define the number and spacing of the objects horizontally and vertically.

ゅ
combit List \& Label

## Mastering Appearance Conditions

In this chapter, you will examine the concept of appearance conditions using a multi-page standard letter or mail merge as an example. By means of appearance conditions, you can specify when objects, elements, paragraphs, lines or columns are to be printed or displayed.
In addition, it is frequently helpful to assign objects that belong together in terms of content into what is called a layer. An appearance condition can be defined for each layer. This condition specifies the conditions under which the objects assigned to this layer are to be printed.

See also:

- Where Will the Objects be Printed?
- Working With Appearance Conditions
- Working With Layers
- Practice: Create a Mail Merge Project

ゅ
combit List \& Label

## Where Will the Objects be Printed?

If a layer has not been defined for the objects, they will be printed on each page.
As an exception to this rule, table objects, cross tabs, text and formatted text are printed consecutively starting on the 1st page.
With text and formatted text, the "Page break" property must be set to "yes".
A new page will therefore be created automatically whenever the space available in the object is no longer sufficient. This means that you do not have to create a second page; this is taken care of automatically by table objects, crosstabs, text, formatted text or by means of the "Minimum page count" project property.

ゅ
combit List \& Label

## Working With Appearance Conditions

You can use filter conditions to control precisely the data to be output. In this way, you can specify conditions not only for displaying data records but also for displaying objects.
These conditions are logical expressions whose results decide whether a certain data record or a certain object is printed or not. If the logical expression is true, the data record or the object is printed. If the logical expression is false, the data record or the object is not printed.
In order to achieve this, you define a corresponding
"Appearance condition". You will find these conditions in the properties of projects, layers, objects, elements, paragraphs, lines or columns.
For example, if you want to output the terms of payment in the footer of an item table, use the appearance condition for this line to specify that it is only to be printed on the last page. Otherwise, this footer will be printed on every page at the end of the table.

Example "Last page only": Lastpage()
Or you define a condition specifying that the company logo is only to be output if the letter is created as a PDF file, since, when outputting to the printer, the company logo is already printed on the letterheads.

## Example "For PDF output only": LL.OutputDevice="PDF"

In appearance conditions for table footer lines, you can also use the predefined "Last page only" value. Internally, this entry uses the functions "Lastpage()" or "LastFooterThisTable()".
In appearance conditions for table headers, you can also use the predefined "First page only" value. Internally, this entry uses the functions "not Lastpage()" or
"FirstHeaderThisTable()".

ゅ
combit List \& Label

## Working With Layers

If you define appearance conditions for a layer, you can then assign objects to this layer. This is very useful especially with large and complex projects as it allows you to hide or display the objects assigned to a specific layer via the checkbox in the tool window.
In this way, you avoid overlapping the different objects on the workspace which would otherwise make working on individual objects difficult.


Figure 4.1: Layers tool window

See also:

- Defining Layers
- Assigning Objects to a Layer


## Defining Layers

You define layers via Project > Layer Definitions or by double clicking in the "Layers" tool window. In the "Layers" dialog, you can define as many additional layers as you want with the "Insert line" button. Each new layer appears initially with the name "Layer". You can enter a meaningful name for the layer in the "Name" field.
§ When creating new projects, the "Base", "First page" and "Following pages" layers are automatically defined.
§ So that you can differentiate better between the different layers on the workspace, it's a good idea to give the layers different colors. This also causes the objects in the respective layers to be shown in different colors in the layout preview display mode. This color has no effect on the actual print.
§ If you delete a layer, the associated objects are automatically assigned to the base layer. At least one layer must be defined.


Figure 4.2: Layer definition dialog
In the "Condition" field you define the appearance
condition for this layer. This appearance condition then applies for all objects on the layer in question, i.e. the associated objects are only printed when the condition for the layer is met.
Typical appearance conditions:

| Condition | Explanation |
| :--- | :--- |
| no condition | The objects on this layer are always printed. |
| Page ()$=1$ | The objects on this layer are only printed on <br> the first page. |
| Page ()$<>1$ | The objects on this layer are printed from the <br> second page onwards. |

Note: The Lastpage() function in an appearance condition can only be evaluated correctly if it is used within the footnote of a table object or in an object linked to a table. In the data lines, the result of Lastpage() is always "False".

## Assigning Objects to a Layer

Once you have defined the layers, you can assign objects to them. You have two options:

1. In order to assign multiple objects to a layer, select the objects in the workspace and choose Assign to layer in the context menu.
Select the layer that you want in the dialog that appears and confirm with OK.


Figure 4.3: Dialog for assigning objects to a layer
In the workspace, the assigned objects automatically receive the color of the corresponding layer so that they can be easily differentiated from the other objects. This only affects the appearance on the workspace and not the print.
2. Alternatively, you can also copy objects into a layer. This is useful when you want to include the same objects in different layers. Example: you create one layer for each language. You then copy all objects into the language layer and translate them.
The original object remains in its original layer and a copy of the object is created in an additional layer. To do this choose Copy to Layer from the context menu for the
selected object.

ゅ
combit List \& Label

## Practice: Create a Mail Merge Project

You learned the basic procedures in Chapter $\underline{2}$. In this chapter, you will now meet other functions and possibilities for designing reports, using a serial or standard letter as an example. As opposed to the previous examples, you use the formatted text object here for the text of the mail merge and change the position of the object from the second page onwards.

See also:

- Create a New Print Template
- Adding a Company Logo
- Add the Address Field
- Adding the Date and Page Number
- Adding Formatted Text for the Letter
- Adjusting the Position of the Letter Text for Following Pages
- Assigning Objects to the Layer


## Create a New Print Template

1. Start the combit List \& Label Sample Application (see Chapter The Sample Application).
2. Choose Design > Mail Merge / Letter to start the Designer.
3. A file selection dialog will appear. To start a new project, choose the New button.

Note: This button is not available in Windows XP. Enter a name for the new print template in the "File name" field and then choose "Open".
To open or edit an existing project, choose Open.

## Adding a Company Logo

Use the picture object to add a logo.

1. In the "Objects" toolbar, choose the "Picture" object type.

Figure 4.4: "Objects" toolbar
2. Pull the object to the right size and select an image file (see Chapter 2.3.1).
3. The company logo is only to be printed for PDF output. The logo is not to be printed when outputting to the printer. This means, you select the picture object and enter the "LL.OutputDevice = "PDF" logical condition in the "Appearance Condition" object property. Also see chapter Overview of LL Variables and LL Fields.

## Add the Address Field

To add an address, use the text object. Text objects let you place text or the contents of fields in the workspace.


Figure 4.5: Address field: it should look like this.

1. In the "Objects" toolbar, choose the "Text" object type.

Figure 4.6: "Objects" toolbar
2. Pull the object to the required size.
3. The formula wizard will now appear, which you can use to define the contents of the text object.
This dialog consists of a series of tabs. On the "Variables and Functions" tab, select the variable for the company address (company) from the list of available variables and fields.
You will see an auto filter field above the list of variables. This means that you can enter "Company" to display all fields and variables containing this expression.
Select the variable that you want by double-clicking and confirm your selection with OK. You have now defined the first line of the address field.


Figure 4.7: Formula wizard with variable
4. A text object can hold as many paragraphs as you want and they can all have completely different display properties. You can add more paragraphs by means of the mini toolbar which appears automatically. Choose "Append paragraph" to define an additional line.


Figure 4.8: Adding another paragraph via the mini toolbar
5. The formula wizard will now appear, which you can use to define the contents of the paragraph. Now enter the first name and the last name of the recipient. First choose the "Firstname" variable.
6. You should insert a space before choosing the "Lastname" variable to prevent the contents of the two variables from being placed directly end to end. A space is simply "Text". Fixed text must be enclosed in quotation marks. So now enter "+" as a joining operator followed by " " for the space.
7. Now select the "Lastname" variable. You must of course also join this with "+". The whole line looks like this:


Figure 4.9: Linking variables and text
8. Now continue with the other variables: street and city. You have now completed the address field.


Figure 4.10: Text field with 4 paragraphs

## Adding the Date and Page Number

Use a text object once more to add a date and the page number to the letter.

1. In the "Objects" toolbar, choose the "Text" object type.
2. Pull the object to the required size.
3. The formula wizard will now appear, which you can use to define the contents of the text object. Switch to the "Date Format" tab.
4. Select the date format that you want. When you do so, the Now() function will be inserted automatically in the Date\$() function. This outputs the current date in the format that you have chosen.
5. Then create a further text object in the footer area for the page number. In the result area of the formula wizard, enter the Page\$() function directly to output the page number.

## Adding Formatted Text for the Letter

You output the text for the letter with the formatted text object. As opposed to the normal text object, this object also lets you change the formatting of the text within a line.

1. In the "Objects" toolbar, choose the "Formatted text" object type.


Figure 4.11: "Objects" toolbar
2. Pull the object to the required size.
3. A dialog appears where you can type in the letter in the form of continuous text.
Alternatively, you can select an RTF variable from the "Source" drop-down list if available.

Expert tip: If the current RTF object is appended to another RTF object for which the "Page break" option is enabled, the "Transfer exceeding text of ..." option is available as a data source ...". If you select this option, you cannot enter text in the RTF object because the (remaining) text will be transferred automatically from the other RTF object (e.g. for a two-column print).


Figure 4.12: Formatted text object dialog
4. If you click the formula button, the formula wizard will appear here as well, which you can use to insert variables. You want to address the recipient personally so you now create the salutation. When doing so, please make sure that you do not enter an unnecessary space character at the end of the salutation formula if the "Name" field in "Dear Sir or Madam" salutations is to be empty. One possibility for a perfect salutation formula is the use of Rtrim $\$($ ) e.g.:
«Rtrim\$(Recipient.Salutation + " " + Recipient.Lastname)»,
5. Now write the letter text. You have various formatting options at your disposal. You create a tab stop with Ctrl+tab.
Or you can create the letter text or pictures and graphics in your normal text processing program (e.g. MS Word), and insert them into this dialog with copy CTrL+C and paste CTRL+V.
6. The "WYSIWYG" (What You See Is What You Get) button shows you the hyphenation based on the object size. This is just an approximate guide and can differ slightly.
7. Page breaks are created automatically if the available
space is no longer sufficient and the "Page break" object property is set to "Yes". If you want to trigger a page break manually, you can do this by means of the context menu. Select the Pagebreak control character.


Figure 4.13: Context menu for a user defined page break

## Adjusting the Position of the Letter Text for Following Pages

Because of the address field, the text starts further down on the first page of a letter. So that the text begins at the top margin in multiple page mail merge projects, a condition can be applied to the position (top, height) of the formatted text object with which the object can be moved upwards and increased in height on following pages.

1. Select the formatted text object.
2. Select the "Position.top" property and set the Cond(Page()=1,125,50) condition.
3. Select the "Position.height" property and set the Cond(Page() $=1,150,225$ ) condition.


Figure 4.14: Specifying the position with a formula

## Assigning Objects to the Layer

So that the objects for address, company logo and date are only printed on the first page, you can assign these objects to the corresponding layer.

1. Hold down the CTrl key and select the objects for the address, company logo and date.
2. Right-click to open the context menu and choose "Assign to Layer".
3. A selection dialog will now appear in which you can select the "First page" layer. Confirm your selection with OK.
4. The objects are now assigned to the "First page" layer. This is indicated by the fact that the objects are shown in the layout preview in green, the color assigned to this layer.


Figure 4.15: Objects are shown in different colors
5. The "formatted text" object remains assigned to the base layer so that it will be printed on all pages.

ゅ
combit List \& Label

## Creating Professional Tables

In this chapter, we will extend the previous examples to include more functions, output grouped data, use appearance conditions and layers to allow multiple pages to be output and add more elements to the report container.

See also:

- Using Advanced Features With the Report Structure
- Practice: Defining the Report Structure Correctly
- Modifying the Fields and Columns
-Defining Multiple Line Layouts
-Defining Column Contents
-Defining Group Lines
- Table Layouts

ゅ
combit List \& Label

## Using Advanced Features With the Report Structure

You use the "Report Container" object to add a table. As the name says, a report container can hold several objects. Tables, charts and crosstabs can be added in any order, even as sub-elements of tables. This lets you define sub-reports with almost any relationships between tables. However, only one report container is permitted and you cannot define any separate tables, crosstabs or charts. You define new elements in the "Report Structure" tool window along with the hierarchical structure that you want. All elements and respective sub-elements of a report structure are shown here, with object type and data source [relation name, sort name], in the order in which they will later be printed (sequence plan).
To add a new element to the report container, select the "Append an element" or the "Append a sub-element" button. Sub-elements are only possible with tables.
Report Structure
783 |l|l|

- -3 Report Container
- Table: Customers [CustomeiD | + 1]
- Till Table: Ordess [Customern2Orders]

TII Table: Order Detais [Orderi2Oeder Detals]
lit Chart: Customer turnover per year
Figure 5.1: "Report Structure" tool window
Only one element is displayed at any time in the workspace. This means that with the "Report Structure" toolbar, you select the element that you want to be displayed in the workspace.

ゅ
combit List \& Label

## Practice: Defining the Report Structure Correctly

Let us assume that you want to produce a list of all customers, showing the orders of the respective customers and all order items.
You want the result to look roughly like this:


Figure 5.2: Hierarchical invoice list
To achieve this, proceed as follows:

1. In the "Objects" toolbar, choose the "Report Container" object type.

$$
\therefore \Phi \mid \text { 因 } \backslash \square \bigcirc \text { 圆 } 1
$$

Figure 5.3: "Objects" toolbar
2. In the workspace, hold down the left mouse button and pull the object to the required size.
3. A selection dialog will appear for the chosen object type. Choose the "Table" object type.

```
Choose Object Type x
Which object would you like to insert into the report container?
The report container can hold any number of objects. The following object types are avalable.
table
    A table displogs data in a logt (e.g. address list). The selection of 'ree content' as a drta source
    enables desctiptive tents and cagtions.
Crosstab
    A crestab collects dats in two dimentions (eg. turnover per region and quarte)
* Chart
    A chart displays data graphically (e.g. pie chart).

Figure 5.4: Choosing the object type
4. You specify the data source in the following dialog. All available tables are shown hierarchically, in other words, under the tables, you will find the relational tables in each case.


Figure 5.5: Choosing the data source
For the list of invoices, you need the following structure: Customers > Orders > Order_Details.
You have 2 alternative procedures at this point:
a) You select the "Customers" table to first create the "top" table. This corresponds to a top-down procedure; meaning that you then add the "Orders" sub-table followed by the "Order details" sub-table by means of the "Report Structure" tool window.
b) Or you choose the structure that you want right from the start by selecting the "lowest" table. This
corresponds to a bottom-up procedure, meaning that you create all three tables starting by designing the "lowest" table.
5. You will be using the second method in this example. Accordingly, you select the table "Customers > Orders > Order_Details".
6. A selection dialog will appear with all the fields in the "Order_Details" table.


Figure 5.6: Field selection dialog
In this dialog, now choose the columns for this sub-table. For example, double-click the "ProductID", "Quantity", "UnitPrice" and "ProductName" fields from the "Products" table which has a 1:1 relationship. This will add the fields to the "Columns" area. You can change the order with the arrow button.
7. The "Order_Details" table will now be displayed in the workspace.
§ The selected fields are displayed in the data line, in other words, the data line contains the data.
\(\S\) In addition, a header line is automatically produced. Header lines are used mostly as column titles, i.e. the selected field names are now shown here as text.
§ List \& Label adjusts the width of the columns automatically. You can adjust the width of a column manually by moving the separating line to the right or
the left with the mouse.
Note: This changes all table columns whose separators are within \(+/-2 \mathrm{~mm}\) from the mouse position. If you hold down the Ctrl key, the action will only be carried out for the line on which the mouse is positioned. The "Column width modification affects next column" option lets you alter the column width while making the next column smaller.
8. To define the columns of the "Orders" table, double-click the table in the "Report Structure" tool window.

ゅ
combit List \& Label

\section*{Modifying the Fields and Columns}

There are two possibilities for adding additional columns to tables or for editing and formatting them in detail.

See also:
- Mini Toolbar
- Object Dialog

\section*{Mini Toolbar}

For the most frequent functions, use the mini toolbar which appears automatically. To select a single field, click the upper left corner of the field.
§ To select multiple fields, hold down the Ctrl key and click
the upper left corner of the fields.
§ To select a whole area, hold down the Shift key.
§ To select an entire column, hold down the Alt key.


Figure 5.7: Multiple field selection
You have various functions at your disposal including:
§ Font: font, font size, bold, italic, underline, color, open font settings dialog or set font to default (via right-click)
§ Properties: Alignment, edit line definition, append new line, delete selected cells, move selected columns or cells to the right or left, open formatting dialog, open frame properties dialog.

\section*{Object Dialog}

Use the object dialog for more advanced functionalities. You open this dialog via the corresponding button in the mini toolbar or by double-clicking the element in the report structure.


Figure 5.8: Object properties dialog for tables
There is a tab for each type of line where you can specify the different definitions and columns for the respective line. The following types of line are available: header line, data line, footer line, group header, group footer. A checkmark on the tab indicates that a line type has one or more line definitions. § Header lines are mostly used as titles for the columns of the table.
§ Data lines contain the formatting for the actual table rows and the data that is to be shown in the table.
§ Footer lines are displayed at the very end of the table and can hold final information about the data lines that are
output above.
§ Group header and footer lines are used to structure the data lines by means of "Intermediate headings" and "Intermediate footers".
All line types can be defined independently of one another. \(\S\) This means that the columns of a header line can have a different appearance as the data lines or footers that follow.
§ You can also create different line layouts or line definitions for the individual line types. It is then possible to activate the different line definitions with special appearance conditions as required.

ゅ
combit List \& Label

\section*{Defining Multiple Line Layouts}

You can define different layouts for each type of line.
Depending on the appearance conditions, the appropriate layout is used in each case according to the situation. For example, you can output two table lines for each data record in this way:


Figure 5.9: Two line definitions
Proceed as follows to create table lines in the object dialog.
1. First select the line type that you want to edit by clicking the relevant tab, e.g. Data Line.
2. Now choose "Insert Table Line" in the context menu. Alternatively, you can select an existing Line definition and then click the "New (Append line definition)" button.
3. In the "Choose a Table Line Definition" dialog that follows, you have the option of
§ using an already existing layout as a template for the new line definition (very useful if the layout is similar).
§ starting the field selection wizard (very useful if you want to create several columns in one operation)
§ or creating an empty line definition so that you can then add the columns by means of the object dialog.


Figure 5.10: Using an existing line layout
4. You have now created the new line. Edit the columns as described in Defining Column Contents. Change the order of the line with the arrow button or with Drag \& Drop.
5. You can then specify the appearance of the new line as a whole. Various properties are available for this including: § "Appearance Condition" with which you can specify when the line is to be printed. This is useful if you define multiple line layouts that are to be printed depending on certain values. The familiar dialog for defining logical expressions opens up here.

Example line 1: Subtotal not on last page
Appearance condition: not Lastpage()
Example line 2: Grand total only on last page
Appearance condition: Lastpage()
§ Name of the line e.g. "data first line". This makes it easier to find the line in complex layouts.
§ Display in Designer: with this property, you can hide the lines in the workspace this is very useful if you have a lot of line definitions.
§ Spacing (margins): here you define the top, bottom, right
and left spacing of the line. The "top" or "bottom" values cause a corresponding space between the individual table rows. With the "left" and "right" spacing values, you can specify the margin in relation to the table object, i.e. you can indent lines or columns.
§ The "Default Font" property sets the font for the entire table row. Newly inserted columns appear initially in this font.
§ Outline Level (index level) of the bookmark in preview mode or for PDF export.
Also see chapters Overview of Properties and Defining Group Lines.


Figure 5.11: Line properties

ゅ
combit List \& Label

\section*{Defining Column Contents}

You can define as many columns as you want for each line. You must only make sure that these columns can be displayed within the width defined for the table.
The individual columns are shown in the object dialog as a tree structure. The buttons let you edit, delete, cut, copy, insert and move the selected columns. You can also move columns outside of the line definitions by using Drag \& Drop. Proceed as follows to create new columns in the object dialog:
1. First select the line in which you want to insert a new column.
2. Now choose "Append column" in the context menu. (Alt+Ins). Alternatively, you can select an existing Column definition and then click the "New (Append column)" button or the small downwards arrow next to this button to specify the type.
3. Each column has a certain type. Various properties are available for the column type including: text, drawing, barcode, RTF text, chart, gauge, HTML text and OLE container. When selecting a field, this data type will be set automatically.
4. To define the contents, the familiar formula wizard will appear in which you can define the column contents in the form of expressions. You will find more information about this under Variables, Formulae and Expressions.
5. Now define the column's properties. Each column in a line can be edited and formatted separately. Select the column that you want in the tree structure in the object dialog.


Figure 5.12: Column properties
To select multiple columns, hold down the Alt or the Shift key. Various properties are available including:
§ Formatting e.g. as number or currency.
§ Name of the column: this will help you to maintain an overview with complex expressions. If you change the name directly in the tree structure, your change will also be applied as "content" where appropriate.
§ You can specify when this column is to be printed with an "appearance condition". This is useful if you define multiple columns that are to be printed depending on certain values. The familiar dialog for defining logical expressions opens up here.
§ Rotation of the content in increments of \(90^{\circ}\).
§ Background, frame, font, vertical and horizontal alignment.
§ A fixed height for the field irrespective of the content.
§ The column width.
Also see chapter Overview of Properties.

ゅ
combit List \& Label

\section*{Defining Group Lines}

Group lines are a special type of line. They are used to group together the data lines that are to be printed.
You can use the "Group by" line property to specify how the data is to be grouped. This means that the line is printed whenever the result of the expression changes from one data line to the next. If you don't enter an expression, the line will not be printed and the property is highlighted in red in the property window.
A group header is printed accordingly before the data line is output, e.g. "Item group XYZ" group heading.
A group footer appears after the condition of the "Group By" property has changed, in other words, after outputting the data line. Group footers are suitable e.g. for totals of data within a group.
Example: Grouping by the first letter of the "Item.No" field.
1. Create a new line definition on the "Group header" tab. Enter the following expression for the content of the column:
"Main group: ' + Left\$ (Item.No,1)"
The result of the expression "Left\$ (Item.No,1)" is the first character of the "Item.No" variable.
Whenever the first letter of "Item.No" changes, the text "Main group: " and the first letter in each case will be printed.
2. Enter "Left\$ (Item.No,1)" as the condition for the "Group By" property.
With each new first letter, a corresponding intermediate header will be printed in the list.
3. Multiple line layouts are possible with group headers as well. In this way, you can produce hierarchically structured intermediate headers. For example, you can define a line
layout that is produced, as in the above example, based on the first letter of the "Item.No" variable. In addition, you define a second line layout that produces intermediate headers based on the first three characters of "Item.No". Create a new line grouping for the sub-group header in the same way. Enter "Left\$ (Item.No,3)" as the condition for the "Group By" property.
Enter the "sub-group" for the content of the column: ' + Left\$ (Artikel.Nr,3)":
When the first 3 letters change, a corresponding intermediate header will be printed in the list.


Figure 5.13: Group lines in an article list
Along with the properties of the "normal" lines, you also have the following at your disposal:
§ Group sums: You can set sum variables to 0 here once they have been output in order to produce group sums.
§ Page break before outputting a group header or break after outputting a group footer.
§ The option of always displaying the group header additionally at the start of the table if the group has been separated by a page break.

Also see chapter Overview of Properties.


Figure 5.14: Group Headers

ゅ
combit List \& Label

\section*{Table Layouts}

There are various properties and functions which you can use to influence the layout of a table. Also see chapters Creating Statistical Reports With Footers, Drilldown Reports (Increase Detail Level), Different Layout Techniques and Overview of Properties.

See also:
- Align Columns
- Fixed Size
- Printing Header Lines and Footer Lines Again
- Defining the Size of the Table Variably
- Forcing a New Page
- Creating a Conditional Page Break
- Keeping Lines Together
- Defining Totals and Counters
- Outputting Free Content Before and After a Table
- Defining Columns Across Multiple Lines

\section*{Align Columns}

There are various ways of simplifying the use of table objects.
§ If you hold down the Ctrl key when reducing the size of a table, all columns will be automatically reduced in size by the same factor.
§ To align (sub) tables exactly with one another, additional tick marks can be shown on the ruler by means of an element property.


Figure 5.15: Tooltip for a column separator
§ List \& Label adjusts the width of the columns automatically. You can adjust the width of a column manually by moving the separating line to the right or the left with the mouse. This changes all table columns, whose separators are within \(+/-2 \mathrm{~mm}\) from the mouse position. If you hold down the Ctrl key, the action will only be carried out for the line on which the mouse is positioned. If the "Column width modification affects next column" option is enabled (Project > Options > Workspace), you can alter the column width while making the next column smaller.
§ You can hide Line Types (header, data, footer, group lines) in the workspace. To do this, select the table object and use Visible Line Types in the context menu or the corresponding menu item View> Visible Line Types.

\section*{Fixed Size}

The "Fixed Size" property lets you specify that the size of the table is not to be adjusted automatically when fewer data lines are printed than the available space in the table object. This property is useful to ensure that footers are always printed at the bottom of the page, e.g. if the page number is output in the footer. If the property is disabled, the end of the table automatically moves upwards (and the footer therefore also).

\section*{Printing Header Lines and Footer Lines Again}

If the print of a table is continued on the following page, the header lines of this table and the outer table will be printed again. To suppress repeated printing of the header lines on the following page, use the FirstHeaderThisTable() function as an appearance condition.
This functionality is also available for footer lines, here you use the LastFooterThisTable() function as the appearance condition. This ensures that footer lines are only output on the last page of the table in the event that the print is continued on the following page due to lack of space. You will find more information about this in the chapter Overview of Functions.

\section*{Defining the Size of the Table Variably}

You can define the height and width of the report containers variably to avoid data being truncated when the page format is changed (e.g. from portrait to landscape).
To do this, select the report container in the "Report
Structure" tool window and use the LL.Device.Page variables and the UnitFromSCM() function to specify the height and width.
\begin{tabular}{|c|c|c|}
\hline EPosition & \multicolumn{2}{|l|}{[0.39, 0.39, 7.86, 11.29 in]} \\
\hline Left & UnitFromSCM(10000) & [0.39 in] \\
\hline Top & UnitFromSCM(10000) & [0.39 in] \\
\hline Width & LL.Device.Page.Size.cx - UnitFromSCM(20000) & [7.47 in] \\
\hline Height & UL.Device.Page-Sizecy - UnitfromSCM (20000) & \(\sqrt{\text { a }}\) \\
\hline
\end{tabular}

Figure 5.16: Size of the report container defined variably

\section*{Forcing a New Page}

With complex projects containing hierarchical tables, it is sometimes wise to create a page break before outputting a line of the "top" table.
Alternatively, you may want to have a new page if, after outputting a data line of the "top" table, there is not enough room for the following data lines of the sub-table.
You can handle both cases with the "Pagebreak Condition" property.
For example, in the case of a hierarchical table, in order to output each data line of the main table on a new page, select the main table in the "Report Structure" tool window and set the "Pagebreak Condition" property to True.


Figure 5.17: Active page break condition

\section*{Creating a Conditional Page Break}

Let s assume that you want to create a continuous table. However, you want to avoid printing data lines from the main table at the bottom of the table unless there is enough room for at least 3 data lines from the sub-table.
You do not want it to look like this:


Figure 5.18: Data lines from the main table without following line

You can avoid this by means of the RemainingTableSpace() function. This function returns a value showing the available space. If you set the parameter to "True", the value will be returned as \(1 / 1000 \mathrm{~mm}\).

If you want to specify that a new page should be started before outputting the data line of the main table if the space remaining is less than 3 cm , enter the following formula for the "Pagebreak Condition" property:
RemainingTableSpace(True)<30000

\section*{Keeping Lines Together}

If the print of a table is continued on the following page due to lack of space, you can decide whether the lines are to be separated or kept together if possible.
This option is useful for multi-line data lines or for invoice footer lines containing totals. You can use this option with data lines, footer lines, group footers and group headers

\section*{Defining Totals and Counters}

You define totals and counters with sum variables or the corresponding aggregate functions. You will find more information about this in chapters Using Functions, Overview of Functions and Working With Sum Variables.

\section*{Outputting Free Content Before and After a Table}

You can output free text before and after a table. To do this, use the RTF object and link the object to the project as free text via the report container.
Produce a first page with a covering letter for the invoice that we created in Chapter Producing a Simple Invoice and an enclosure with the General Terms and Conditions.
1. To add a new element to the report container, select the "Append an element" or the "Append a sub-element" button in the "Report Structure" tool window.
2. A selection dialog will appear for the chosen object type. Choose the "Table" object type.
3. In the following dialog, select "Free content" as the data source.


Figure 5.19: Add free content to the report container
4. Then add a column via the properties dialog for the table object. In our case, we want to create the covering letter
as formatted text. Therefore, click the small arrow on the right of the button and choose the Formatted Text option.


Figure 5.20: Specifying formatted text for the content of the column
5. A dialog appears where you can type in the covering letter in the form of continuous text. You will find detailed instructions for working with formatted text in Chapter Practice: Create a.
6. Please don't forget to remove the (column) frame for free content. The best way is to remove the frame via the "Default Frame" property in the element properties.
7. You may also have to change the layer condition as the covering letter will now be output on the first page.
8. If the General Terms and Conditions are also to be output at the end of the invoice, you must add another free content object to the container after the item table. Use the Formatted Text object here as well or include the content via a linked file.
Use the LoadFile\$() function to load a linked file and enter the path of the file as the parameter as follows LoadFile\$ (ProjectPath\$()+"\gtc.txt").


Figure 5.21: Including content from a file with LoadFile\$()

\section*{Defining Columns Across Multiple Lines}

Let us assume that you want to output the contents of a column across two lines. You can achieve this by specifying a negative value for the "Frame.Top.Distance" property, e.g. -0.2 inch. The content will then be moved upwards by this distance. Please note, however, that this will cause content to be overlapped and that the export function cannot always apply the layout 1:1.


Figure 5.22: Column contents across two lines

ゅ
combit List \& Label

\section*{Producing Analyses}

In List \& Label, you can make use of charts, gauges, crosstabs, footer lines in tables or drilldown reports to analyze your data.

See also:
- Creating Charts
- Creating Gauges
-Creating a Crosstab
- Creating Statistical Reports With Footers
-Drilldown Reports (Increase Detail Level)
- Multi-Column Reports

ゅ
combit List \& Label

\section*{Creating Charts}

This object is used to evaluate and display data graphically in diagrams. This gives you an overview of your data and lets you recognize anomalies immediately.
For example, you can analyze sales trends, illustrate percentage shares and show data in multiple dimensions.
You have a wide range of different types of diagrams at your disposal:
§ Pie chart
§ Bar chart (also displayed as cylinders, pyramids, cones, octahedrons)
§Simple (e.g. sales per customer)
§ Multi-row (e.g. sales to various customers over the years, scaled by customer)
§ Clustered (e.g. sales to various customers over the years, grouped by year)
§ Stacked (e.g. percentage of sales to various customers stacked over the years)
§ 100\% stacked (e.g. respective sales percentages for various customers over the years)
§ Lines/symbols
§ Simple
§ Multi-row
§ Stacked
§ 100\% stacked
See also:
- Inserting a Chart Object
- Creating a Pie Chart
- Creating a Multi-Row Bar Chart
-100\% Stacked Bar Chart
- Multi-Row Line Chart
- Using Series to Determine the Values
-Special Fields With Diagrams

\section*{Inserting a Chart Object}

There are various ways of outputting chart objects:
1. A chart as an element in the report container. Add the object via the "Report Structure" tool window. If you have not yet added a report container to the workspace, select "Report Container" in the "Objects" toolbar and pull the object to the right size in the workspace while holding down the left mouse button. A selection dialog will appear for the chosen object type. Choose the "Chart" object type.
2. You can output charts and gauges in a table cell. To do this, select the relevant entry by means of the context menu in the object dialog for the table. If you want to output the aggregated data, a good way of doing this is to use a footer line.


Figure 6.1: Chart object in a footer line
3. In the following dialog, now select the data source. All available tables are shown hierarchically, in other words, under the tables you will find the relational tables in each
case.
To evaluate sales per country, for example, choose the "Customers > Orders > Order Details" table in the List \& Label Sample Application so that you have all three tables at your disposal. The "Customers" table contains the country, the "Orders" table the order date and the "Order_Details" table the sales.
4. The chart object dialog is displayed. In the drop down lists in the top left you can select the base type and the corresponding sub type; three types with up to nine sub types are available:
§ Pie: Pie, Circle.
§ Bar/Ribbon: Simple (also 3D), Multi-Row, Clustered (also
3D), Stacked (also 3D), 100\% Stacked (also 3D). § Line/Symbol: Simple, Multi-Row, Stacked, 100\% Stacked. The properties are defined in the tabs. You can click directly into the live preview (e.g. onto the title or axis label) to quickly jump to the corresponding property.


Figure 6.2: Chart object dialog

\section*{Creating a Pie Chart}

Let's assume that you want to evaluate the sales per country. The pie chart is the right choice for this. It lets you read off the percentages immediately. Proceed as follows in the List \& Label Sample Application:
1. As the data source, select the "Customers > Orders > Order_Details" table.
2. For the diagram type, choose Pie > Pie.
3. You should first specify the coordinate values for the data source, i.e. the values that define the individual pie segments, e.g. Customers.Country.
Data Source Pie Segment Diagram Obyet Colons


Figure 6.3: Definition of the data source in the pie chart object
4. Switch to the "Pie Segment" tab to specify the coordinate values for size of the pie segment, i.e. the sales. Doubleclick the "Coordinate Value" property. Now select the aggregate function that you want for the contents in the "Coordinate Value" dialog that follows. You want to create a sales evaluation so choose the "Sum" function.


Figure 6.4: Wizard for creating the coordinate value formula
5. In the upper part of the dialog, you can specify the contents by clicking the formula button to start the formula wizard. In the Sample Application, the sales per order value is not supplied directly as a field so you must calculate it using the "Order_Details.Quantity * Order_Details.UnitPrice" formula.
6. The "Label on Object" property is already set to "Yes" so that a label with the percentage value is shown on the pie segments. Define the value as "percent" without decimal places by means of the "Format" property.


Figure 6.5: Definition of the pie segment in the pie chart object
7. The "Explosion Offset" property lets you specify a distance to the center for the pie segment. With the "Arclndex" chart field, which numbers the pie segments according to their size, you can even display the largest
pie segment with a greater offset. Example:
Cond (LL.ChartObject.ArcIndex=1,50,10)
8. On the "Diagram" tab, select the general diagram options. Various properties are available including:
§ The degree of perspective, e.g. strong.
§ The color mode, e.g. single color
Also see chapter Overview of Properties.
9. On the "Object" tab, select the general layout options for the entire chart object. Various properties are available for this including:
§ Title, e.g. "Customer turnover per country"
§ Position of the title
§ Background including filling, border and shadow, e.g. border = transparent
Also see chapter Overview of Properties.
10. On the "Colors" tab, you can specify the colors for the display:
§ Assigned Colors: You can assign fixed colors to particular axis values. If you click the "New" button, you can create a new assignment e.g. Customers.Country = "Germany".
§ Unassigned Colors: Specifies the colors and color sequences for the data rows that are not specified by the "Assigned Colors".
11. The pie chart now looks like this:

Customer turnover per country


Figure 6.6: Pie chart

\section*{Creating a Multi-Row Bar Chart}

Let's assume that you want to evaluate the sales for various countries over the years, scaled by country. A multi row bar chart is perfect for this. You get a diagram in which you can see the turnover achieved in the respective country for each quarter. Proceed as follows in the List \& Label Sample Application:
1. As the data source, select the "Customers > Orders > Order_Details" table.
2. Choose Bar/Ribbon > Multi-Row (3D) as the diagram type.
3. First specify the coordinate values for the category axis, i.e. the values of the x-axis. In the Sample Application, the order year is not supplied directly as a field so you must calculate it using the "Year\$(Orders.OrderDate" formula. If you want to evaluate the data by quarter and year, simply enter "Orders.OrderDate" as the coordinate value and then choose an appropriate date format by means of the "Coordinate Label > Format" property, e.g. "Q\%q \%y". Then change the "Sort Coordinates" property to "No" so that the values are not sorted by quarter.
4. Type "Year" as the text for the "Axis Label".
\begin{tabular}{|c|c|c|c|c|c|c|}
\hline Category Aus (x) & Value Axss (y) & Diagram & Object & Colors & & \\
\hline (18) At t*3 & & & & & & 0 \\
\hline \multicolumn{7}{|l|}{EData} \\
\hline \multicolumn{5}{|l|}{Coordinate Value} & \multicolumn{2}{|l|}{YearYOrders.OrderDate)} \\
\hline \multicolumn{5}{|l|}{Sort Coordmates} & \multicolumn{2}{|l|}{Yes} \\
\hline \multicolumn{5}{|l|}{Numbes of Sample Resords} & \multicolumn{2}{|l|}{19} \\
\hline \multicolumn{5}{|l|}{Filter} & \multicolumn{2}{|l|}{Nofilter (All Dasta)} \\
\hline \multicolumn{5}{|l|}{Round Seart and End Values} & \multicolumn{2}{|l|}{Yes} \\
\hline \multicolumn{7}{|l|}{EAxil Label} \\
\hline \multicolumn{5}{|l|}{回Aus Label} & \multicolumn{2}{|l|}{'Vear*} \\
\hline \multicolumn{5}{|l|}{BlCebrinute label} & \multicolumn{2}{|l|}{UL.ChartObject.AoalCoordinate} \\
\hline \multicolumn{5}{|l|}{tegend} & \multicolumn{2}{|l|}{At Aoss} \\
\hline
\end{tabular}

Figure 6.7: Definition of the category axis
5. Now specify the coordinate values for the series axis, i.e. the values of the y-axis. Select the "Customers.Country" field via the formula wizard.


Figure 6.8: Definition of the series axis
6. Now specify the coordinate values for the value axis (zaxis), i.e. the height of the bars representing the turnover. Double-click the "Coordinate Value" property. Now select the aggregate function that you want for the contents in the "Coordinate Value" dialog that follows. You want to create a sales evaluation so choose the "Sum" function.


Figure 6.9: Wizard for creating the coordinate value formula
7. In the upper part of the dialog, you can specify the contents by clicking the formula button to invoke the formula wizard. In the Sample Application, the sales per
order value is not supplied directly as a field so you must calculate it using the "Order_Details.Quantity * Order_Details.UnitPrice" formula.


Figure 6.10: Definition of the value axis
8. Various other properties are available on this tab including the following layout options: § Maximum Value Automatic: You can limit the height of the displayed area, e.g. to cater for "anomalies".
§ Presentation: The data can be presented in various ways: cylinders, bars, pyramids, ribbons, octahedrons, cones
§ Thickness of the bars
§ Zebra mode for the background
Also see chapter Overview of Properties.
9. On the "Diagram" tab, select the general diagram options. Various properties are available including:
§ The degree of perspective, e.g. strong.
§ Color Mode: Specifies which axis determines the color, e.g. the y-axis values.

Also see chapter Overview of Properties.


Figure 6.11: Definition of the diagram options
10. On the "Object" tab, select the general layout options for the entire diagram. Various properties are available for this including:
§ Title, e.g. "Customer turnover per country"
§ Background including filling, border and shadow, e.g. border \(=\) transparent
Also see chapter Overview of Properties.
11. On the "Colors" tab, you can specify the colors for the display:
§ Assigned Colors: You can assign fixed colors to particular axis values as a condition, e.g. Customers.Country = "Germany".
§ Unassigned Colors: Specifies the colors and color sequences for the data rows that are not specified by the "Assigned Colors".
12. The multi-row bar chart now looks like this:

Customer turnover per country


Figure 6.12: Multi row bar chart

\section*{100\% Stacked Bar Chart}

The pie chart in the first example gave you an overview of the percentages for the entire evaluation period. But in order to be able to recognize trends, it would be good to see how the percentages have changed during the course of the evaluation period. The \(100 \%\) stacked bar chart can be used for precisely these types of applications. The respective percentage of the length of the bars relates directly to the turnover percentage of the respective country.
Proceed as follows in the List \& Label Sample Application:
1. As the data source, select the "Customers \(>\) Orders \(>\) Order_Details" table.
2. Choose Bar/Ribbon \(>100 \%\) stacked as the diagram type.
3. First specify the coordinate values for the category axis, i.e. the values of the x-axis. In the Sample Application, the order year is not supplied directly as a field so you must calculate it using the "Year\$(Orders.OrderDate" formula.
4. Now specify the coordinate values for the series axis, i.e. the values of the y-axis. Select the "Customers.Country" field via the formula wizard.
5. Specify the coordinate values for the value axis (z-axis), i.e. calculate the turnover with the "Order_Details.Quantity * Order_Details.UnitPrice" formula.
6. On the "Diagram" tab, choose "Left to Right" for the
"Alignment" to create a horizontal diagram.
7. The multi-row bar chart now looks like this:


Figure 6.13: 100\% Stacked bar chart

\section*{Multi-Row Line Chart}

A line diagram offers an alternative to a multi-row bar chart. You can read off the values faster here.

Customer turnover per year


Figure 6.14: Multi-row line chart
Proceed as follows in the List \& Label Sample Application:
1. As the data source, select the "Customers > Orders > Order_Details" table.
2. Choose Line/Symbol > Multi-Row as the diagram type.
3. First specify the coordinate values for the category axis, i.e. the values of the x-axis. In the Sample Application, the order year is not supplied directly as a field so you must calculate it using the "Year\$(Orders.OrderDate" formula.
4. Now specify the coordinate values for the series axis, i.e. the values of the y-axis. Select the "Customers.Country" field via the formula wizard.
5. Specify the coordinate values for the value axis (z-axis), i.e. calculate the turnover with the "Order_Details.Quantity * Order_Details.UnitPrice" formula.

\section*{Using Series to Determine the Values}

With three-axis diagrams, you can also determine the values of the series axis ( \(y\)-axis) by means of rows. This means that you define the different rows (e.g. measured value/target value/actual value) with a single data record and can show them parallel e.g. in a bar chart.
As an example, we will create a diagram which shows the currency percentages of the 3 economic areas. Data for APAC, EMEA and NAFTA is supplied as rows.
Currency shares of the economic areas per year in percent


Figure 6.15: Example of a diagram using rows
Proceed as follows in the List \& Label Sample Application:
1. Select the "Sales" table as the data source.
2. Choose Bar/Ribbon > \(100 \%\) stacked as the diagram type.
3. First specify the coordinate values for the category axis, i.e. the values of the x-axis. Select the "Sales.Year" field with the formula wizard. Remove the 2 decimal places using the "Str\$(Sales.Year, 0,0 )" formula.
4. Now specify the coordinate values for the series axis, i.e. the values of the \(y\)-axis. Select the "Use rows as data source" entry from the drop-down list above the properties.


Figure 6.16: Option for using rows to determine the value
This option changes the properties of the series axis and displays a dialog for defining the rows when you click the "Row Definitions" property. Create the individual rows choosing "Sales.APAC", "Sales.EMEA" or "Sales.NAFTA" in each case as the coordinate value.


Figure 6.17: Row definitions dialog

\section*{Special Fields With Diagrams}

Various fields are available for diagrams including:
§ LL.ChartObject.ArcIndex: returns the index of the current pie segment. The largest pie segment has index 1 , the second largest has index 2 and so on.
This means, for example, that you can assign a greater explosion offset to the largest pie segment:
Cond(LL.ChartObject.ArcIndex \(=1,30,10\) )
§ LL.ChartObject.ArcPerc: returns the percentage share of the current pie segment. This field is especially useful for labeling the pie segment.
§ LL.ChartObject.ArcTotal: returns the absolute value of the total data volume with pie charts.
§ LL.ChartObject.ArcTypelsOthers: returns True, if the current pie segment is the "other" segment.
This means, for example, that you can assign a greater explosion offset to the "other" pie segment: Cond(LL.ChartObject.ArcTypelsOthers,30,10)
§ LL.ChartObject.ArcValue: returns the absolute value of the current pie segment. This field is especially useful for labeling the pie segment.
§ LL.ChartObject.AxisCoordinate: returns the coordinate value and can be used for labeling the axes.
You will find more information about this in the chapter Overview of Functions.

ゅ
combit List \& Label

\section*{Creating Gauges}

Gauges are a good way of presenting actual values. Let's assume that you want to output the current turnover in relation to the target value. A gauge is the right choice for this. It provides you with the value at a glance.

See also:
- Inserting a Gauge
- Specify Properties

\section*{Inserting a Gauge}

There are various ways of outputting gauges:
1. A gauge as an object. Select the "Gauge" in the "Objects" toolbar and pull the object to the right size in the workspace while holding down the left mouse button.
2. You can output gauges in a table cell. To do this, select the "Gauge" entry by means of the context menu in the tables object dialog. If you want to output the aggregated data, a good way of doing this is to use a footer line.


Figure 6.18: Gauge object in a footer line

\section*{Specify Properties}
1. Choose the type, form, shadow and pointer properties for the gauge using the drop-down lists.
2. The "Value" property lets you determine the value that the pointer is to display, e.g. the customer's turnover or rating.
3. You can also make use of various other layout options including:
§ Appearance: Filling, pointer options, glass properties. § Lettering: Rotation angle, white space before and after the scale range, tickmarks, scale labels, signal ranges, text fields
§ Values: Minimum and maximum vales of the scale Also see the chapter Overview of Properties.


Figure 6.19: Gauge properties
4. Indicate the optimal area, e.g., by specifying the color of the signal range.


Figure 6.20: Gauge with colored signal ranges
To do this, set the "Signal Ranges" property to "Show" and click the "..." button to open the dialog for defining the regions. In this dialog, click the "New" button to create the respective ranges with start and end values and assign the colors that you want.


Figure 6.21: Signal range definition dialog
5. You can also define text areas in the same way, i.e. you can output text to any position, as you wish.


Figure 6.22: Gauge with text area
To do this, set the "Labels" property to "Show" and click the "..." button to open the dialog for defining the regions. In this dialog, click the "New" button to create the respective ranges with position, rotation, frame size, background, font and formatting. You specify the position in relation to the area of the gauge (measured from left to right). For example, a vertical and horizontal position of \(50 \%\) each positions the label precisely in the middle.


Figure 6.23: Text label definition dialog

ゅ
combit List \& Label

\section*{Creating a Crosstab}

Crosstabs are used for evaluating and presenting data in multiple dimensions. Crosstabs (or contingency tables) are tables containing information about the frequency of the occurrence of combinations of certain characteristics.
These frequencies are extended by their marginal totals which form "contingencies." With a three-dimensional crosstab, (three characteristics), the table includes an additional column grouping.
For example, you can examine turnover trends per year and region, evaluating sales according to quantities and customers, and create marginal totals for quarters and years.
A normal ("flat") table has the attribute names in the first row and the occurrences of these attributes in all other rows. A crosstab is different. The titles of both columns and rows receive characteristic occurrences and, at the point of intersection of the respective column and row, a value is shown that depends on the characteristics specified for the column and row in each case.

Customer turnover per year and quarter
\begin{tabular}{|c|c|c|c|c|c|c|c|c|}
\hline & Callahan & Davolio & Fuller & King & Leverling & Peacock & Suyama & Total \\
\hline ALFKI & --- & 1342 & --- & --- & --- & 1208 & 1086 & 3636 \\
\hline ANATR & --- & --- & --. & 89 & 800 & 514 & --- & 1403 \\
\hline ANTON & --- & 957 & --- & 2963 & 403 & 2157 & --- & 6480 \\
\hline AROUT & 899 & 2596 & --- & --- & --- & 1704 & 480 & 5679 \\
\hline BERGS & 1489 & --- & 613 & --- & 3429 & --- & --- & 5531 \\
\hline Total & 2388 & 4895 & 613 & 3052 & 4632 & 5583 & 1566 & 22729 \\
\hline
\end{tabular}

Figure 6.24: Example of a three-dimensional crosstab
For the schematic presentation of two-dimensional crosstabs, the 3D multi-row bar chart is the best choice. You can find more information about this in chapter Creating Charts.

See also:
- Creating a Crosstab Object
-Defining Groupings
- Defining Cell Properties
- The Layout Option and Wrapping Behavior - Special Functions

\section*{Creating a Crosstab Object}

Let's assume that you want to examine the development of turnover per year, quarter and country. Proceed as follows in the List \& Label Sample Application:
1. Crosstabs are elements in the report container. Therefore, you add these objects in the "Report Structure" tool window. If you have not yet added a report container to the workspace, select "Report Container" in the "Objects" toolbar and pull the object to the right size in the workspace while holding down the left mouse button.
2. A selection dialog will appear for the chosen object type. Choose the "Crosstab" object type.
3. In the following dialog, now select the data source. All available tables are shown hierarchically, in other words, under the tables you will find the related tables in each case.
For our turnover analysis, choose the "Customers > Orders > Order Details" table in the List \& Label Sample Application so that you have all three tables at your disposal. The "Customers" table contains the country, the "Orders" table the order date and the "Order Details" table the turnover.
4. A wizard appears which will lead you through the 3 configuration dialogs for crosstabs.

\section*{Defining Groupings}

In the wizard's first dialog, or alternatively on the "Axis
Definition" tab, you first define the grouping for the rows and columns, i.e. the characteristics.
1. In the "Rows" pane, click on the "Insert a row grouping" button.
2. In the formula wizard, you now enter the field or the expression for the row grouping e.g. Customers.Country. You have now created a row grouping and the data will be grouped by this characteristic.
3. In the "Columns" pane, click on the "Insert a column grouping" button.
4. In the formula wizard, you now enter the field or the expression for the column grouping.
5. Since you first want to group the data by year, you must enter an expression here that returns the year of the order date. You have the Year() function in the formula wizard at your disposal; i.e. you select this function from the list and insert the order date as the parameter by double-clicking. The formula looks like this: Year(Orders.OrderDate).
6. Since we also want to examine the data at another level, insert an additional column grouping via the "Insert a column grouping" button.
7. Now enter an expression to return the quarter of the order date. You can use the Quarter() function in the formula wizard for this. The formula then looks like this: Quarter(Orders.OrderDate).

Note: You can change the order of the groupings with the arrow button. The column at the bottom is the inner grouping.

To swap lines and rows (Pivot function) use the button on the lower right on the "Axis Definition" tab. This button is only available in the object dialog, not in the wizard.


Figure 6.25:Swap all rows and lines
8. You have now created the groupings and you can go on to define the value for the intersection of the respective columns and rows. Click on the "Edit group result formula" button located under "Value of the result cells".
9. Now select the aggregate function that you want for the contents in the "Cell Contents" dialog that appears. You want to create a sales evaluation so choose the "Sum" function. In the upper part of the dialog, you can specify the contents by clicking the formula button to start the formula wizard.


Figure 6.26: Formula wizard for defining the cell content
10. In the Sample Application, the sales per order value is not supplied directly as a field so you must calculate it using the "Order_Details.Quantity * Order_Details.UnitPrice" formula.


Figure 6.27: Definition of the axes of a three-dimensional crosstab

\section*{Defining Cell Properties}

In the wizard's second dialog or, alternatively, on the "Cell Definition" tab, you edit the properties of the different cells. You can select the cells directly in the drawing in the upper pane of the dialog and then edit their properties. To select multiple cells, hold down the Ctrl key or you can draw a border around the cells with the mouse.
1. Assuming that you want to prefix the number of the quarter with a "Q" as the title of a column. Select the respective column title and then double-click on the "Displayed Contents" property.


Figure 6.28: Cell definition for the crosstab
2. With this property, you can now specify the text that is to be displayed in this cell (independent of the value that you have defined for this column grouping).
Now define either a suitable formula, e.g. "Q " +

Str\$(Quarter(Orders.OrderDate)) in the formula wizard, or use the "Format" property. Then remove the "Quarter\$()" here, i.e. only the date field remains in the field, and format the value by means of the property.
To do this, click the "Formatting" property, choose "Date" as the formatting type, and finally select the "Userdefined" entry from the drop-down list. At the end of the list you will find an example for formatting a quarter plus the number of the year. Since we don't need the number of the year, shorten the formula's string to "Q\%q".


Figure 6.29: Formatting for date values
3. This cell is now formatted and you can go on to format all other cells in the same way. Various properties are available including:
\(\S\) Rotation of the content in increments of \(90^{\circ}\)
§ Background
§ Frame
§ Font
§ Vertical and horizontal alignment
§ Maximum width, minimum width and minimum height Also see chapter Overview of Properties.

\section*{The Layout Option and Wrapping Behavior}

In the wizard's third dialog or, alternatively, on the "Properties" tab, you edit the layout properties and specify the wrapping behavior.
Various layout properties are available including:
§ Background
§ Default frame
§ Minimum size (\%) and minimum height
In addition, as crosstabs are often wider and higher than the specified page format, you can also specify the wrapping behavior for columns and rows. List \& Label creates as many pages (shadow pages) as necessary. The row labels are repeated on all pages as standard while the column labels are not repeated.
Various wrapping properties are available including:
§ Repeat Labels: Specifies whether the labels of columns or rows are to be printed again in the case of a page break.
§ Break Level: Specifies the optimum break level, e.g. "0".
This corresponds to the lowest group, i.e. the quarter.
§ Column > Page Break on Shadow Pages: If the cross table is too wide, the wrapped parts are printed on shadow pages. A shadow page does not count as a "real" page and therefore does not have a page number. The default setting specifies that the wrapped parts are to be output below the table.
Also see chapter Overview of Properties.


Figure 6.30: Crosstab properties

\section*{Special Functions}

Various additional functions are available in crosstabs including:
§ Crosstab.Value() returns the content of the cell (as a value).
§ Crosstab.Cells.Avg() returns the average value of the volume of data.
§ Crosstab.Col\$() or Crosstab.Row\$() returns the description of the column or the row for the current cell.
With this, you can, for example, assign a particular color to the background of a column or row. The following example sets the background color to orange for all cells in a row where the cell descriptor is "Germany":
Cond(Crosstab.Row\$()="Germany",LL.Color.Orange,LL.Color.


Figure 6.31: Coloring a particular row
§ Crosstab.Cells.Max() or Crosstab.Cells.Min() returns the largest or smallest value in the entire crosstab. With this, you can, for example, emphasize the largest or smallest value of the volume of data or perform calculations. The following example sets the background color of the cell with the largest value to green:
Cond(Crosstab.Value=Crosstab.Cells.Max(),LL.Color.Green, Cond(Crosstab.Row\$()="Germany",LL.Color.Orange,LL.Color.
\begin{tabular}{|c|c|c|c|c|c|c|c|c|c|}
\hline \multirow[t]{3}{*}{} & \multicolumn{9}{|c|}{Customer turnover per year and quarter} \\
\hline & & & \multicolumn{3}{|l|}{2009} & \multicolumn{3}{|l|}{2010} & Total \\
\hline & 03 & 04 & 01 & Q2 & 03 & 04 & Q1 & 02 & Total \\
\hline Germamx & & & & & 1,086 & 1,208 & 851 & 491 & 3656 \\
\hline Mexico & & 492 & & 3,038 & 2,562 & 1,277 & & 514 & 7883 \\
\hline Sweden & 2,102 & & 3,429 & & & & & & 5531 \\
\hline UK & & 480 & 1,352 & & 2,143 & 1,704 & & & 5679 \\
\hline Tetal & 2102 & 972 & 4781 & 3038 & 5791 & 4189 & 851 & 1006 & 22729 \\
\hline
\end{tabular}

Figure 6.32: Coloring a particular cell
§ Crosstab.Col() or Crosstab.Row() returns the index of the column or the row for the current cell. Here, for example, you can set the background color of alternate rows thereby producing a zebra pattern. Example:
Cond(Odd(Crosstab.Row()),LL.Color.LightGray,LL.Color.White Customer turnover per year and quarter
\begin{tabular}{|c|c|c|c|c|c|c|c|c|c|}
\hline \multirow[t]{3}{*}{} & \multicolumn{2}{|c|}{\multirow[b]{2}{*}{2008}} & \multicolumn{4}{|c|}{\multirow[b]{2}{*}{2009}} & \multicolumn{2}{|c|}{\multirow[t]{2}{*}{2010}} & \multirow[b]{3}{*}{Total} \\
\hline & & & & & & & & & \\
\hline & 03 & 04 & 01 & Q2 & Q3 & 04 & 01 & Q2 & \\
\hline Gemany & & & & & 1,086 & 1,208 & 851 & 491 & 3636 \\
\hline Mexico & & 492 & & 3.038 & 2,562 & 1,277 & & 514 & 7883 \\
\hline Sweden & 2,102 & & 3,429 & & & & & & 5531 \\
\hline UK & & 480 & 1,352 & & 2,143 & 1,704 & & & 5679 \\
\hline Total & 2102 & 972 & 4781 & 3038 & 5791 & 4189 & 851 & 1006 & 22729 \\
\hline
\end{tabular}

Figure 6.33: Creating a zebra pattern
§ Join\$() returns a collection of strings, separated by a particular character. For example, you can output the individual turnover amounts in addition to the total turnover. Example:
Fstr\$(Sum(Order_Details.Quantity*Order_Details.UnitPrice)," \#\#,\#\#\#,\#\#\#") + "ף["+
Join\$(Fstr\$(Sum(Order_Details.Quantity*Order_Details.UnitPI \#\#,\#\#\#,\#\#\#"))+"]"


Figure 6.34: Display detailed information
§ You can use Total() for calculations across all cells.
Otherwise, calculations are always made across all values

\section*{that affect the respective cell.} Also see chapter Overview of Functions.

ゅ
combit List \& Label

\section*{Creating Statistical Reports With Footers}

When you enable the "Data Lines.Suppress" object property in tables, all data lines are completely suppressed. This option is particularly useful in combination with the "Force Sums" option. The latter option specifies that totals are also calculated when a data line is not printed. By combining both options, you can output footer lines with totals and produce interesting statistics in this way.
Let's assume that you want to output the turnover per country:

\section*{Customer turnover}
\begin{tabular}{|l|r|}
\hline Mexico & \(\$ 172.80\) \\
\hline Sweden & \(\$ 380.00\) \\
\hline Denmark & \(\$ 140.00\) \\
\hline USA & \(\$ 616.00\) \\
\hline Italy & \(\$ 320.25\) \\
\hline
\end{tabular}

Figure 6.35: Creating statistical reports with footers
Proceed as follows in the List \& Label Sample Application:
1. Create a new element in the report container and choose "Table" as the object type.
2. In the following dialog, now select the data source. All available tables are shown hierarchically, in other words, under the tables you will find the relational tables in each case.
To evaluate sales per country, for example, choose the "Orders > Order Details" table so that you have both tables at your disposal. The "Customers" table has a 1:1 relationship with the "Orders" table so you don't need to select it. The turnover is held in the "Order_Details" table.


Figure 6.36: Hierarchical tables for statistical reports
3. Create a data line with the "OrderID" field in the "Orders.Order_Details" table. Although the data line is not output (it is suppressed), List \& Label still needs a field so that the table can be printed at all.
4. Define the actual statistic as a footer line, i.e. with the country name in the first column, and total the turnover in the second column. Now calculate the total again with the "Sum(Order_Details.Quantity * Order_Details.UnitPrice)" formula.


Figure 6.37: Creating footer lines for a statistical report
5. In the "Orders" table, create a data line with the "Country" field from the linked table "Customers". Although the data line is not output (it is suppressed), List \& Label still needs a field so that the table can be printed at all.
6. Define the grand total across all countries as a footer line, i.e. with "Total" in the first column, and total the turnover again in the second column.
7. Now select the "Orders" table in the "Report Structure"

\section*{tool window and set the＂Data Lines．Suppress＂property to ＂Yes＂．}

8．Finally，also select the＂Orders＂table and set the＂Data Lines．Suppress＂property to＂Yes＂here as well．
```

E Data Lines
Force Sums Yes
Keep Rows Together
@ Zebre Pattem Pattern/Block Color
⿴囗十|ooter Lines
田Group Footer Lines
$\square$ Group Header Lines

```

Figure 6．38：Suppressed data lines for statistics

ゅ
combit List \& Label

\section*{Drilldown Reports (Increase Detail Level)}

Navigation in hierarchical data is known as drilling down. Drilling down makes it possible to "zoom in" to examine the data at different levels of detail. Different print templates are linked to each other to achieve this. This property is only available with hierarchical tables.


Figure 6.39: Example of a report with an open drilldown report

These reports make it possible for anyone to find the information they are looking for quickly, even with very large and complex data inventories.
Only one level is printed to start with (e.g. customers). A new detail report opens (e.g. orders) when you click on a customer.
This drilldown report can be opened in the context menu either in the same window (navigation via the green arrow button in the preview window), or in a new foreground or background tab.
The drilldown function is only available in the preview. You can export any drilldown report to another format from the preview, e.g. PDF.
Drilldown reports can be embedded in the preview file to allow them to be sent or saved as a complete unit. You will find the respective option "Embed Drilldown Reports" in the project properties.
A drilldown link relates either to a single field or an entire table row. A whole series of links can be associated with each of these elements, e.g., to present the data in different ways. Proceed as follows to create a drilldown report:
1. In the "Objects" toolbar, choose the "Report Container" object type. In the workspace, hold down the left mouse button and pull the object to the required size. A selection dialog will appear for the chosen object type. Choose the "Table" object type.
2. In the following dialog, now select the data source. All available tables are shown hierarchically, in other words, under the tables you will find the related tables in each case.
In order to be able to open a sub-report in drilldown mode, you must select a table here that also has a sub-table! In the Sample Application, select the "Customers" table, for example, as it has "Orders" as a sub-table.
3. Now define the columns of the table with the wizard, i.e. CustomerID, CompanyName, ContactName, City.
4. In the object dialog for the table, now define an additional
column for the drilldown link. Enter the text "Show..." as the content.
5. Now open the dialog for creating the drilldown link by means of the "Drilldown Links" column property.


Figure 6.40: Data line with additional column for a drilldown link
6. Create a new drilldown link with the "Insert new link" button in the dialog that appears.


Figure 6.41: Dialogs for defining the linked drilldown report
7. A dialog appears where you can create the print template for the sub-report. Select the "Create a new project" option and enter the name for the print template.
8. A second instance of the Designer opens for you to create the print template. Proceed as usual to create this sub-
report. I.e. you create a report container, choose the "Table" object type, choose "Orders" as the data source and define the columns of the orders table that are to be shown in the report.
9. Once you have completed the report, close this second Designer instance.
10. Back in the "Drilldown Links" dialog, you now define the "Menu Text" property for this link. If you have more than one link, its text will be shown in a context menu. The text will also be used as the title for a tab if multiple drilldown reports are shown in a preview window.
11. You have now finished the drilldown report and you can display it in the preview.

ゅ
combit List \& Label

\section*{Multi-Column Reports}

With complex projects containing many different diagrams, it may be a good idea to present them over several columns.


Figure 6.42: Multi-column report container
To make this possible, the "Column Count" property is provided in the report container and also in tables. You can define up to five columns whereby the report container and the included tables can have different column count values.
Creation is easy:
1. Select the "Report Container" object in the "Report Structure" tool window.
2. Set the "Column Count" object property to "2".
3. If you select individual objects, you will then have various properties for controlling column breaks at your disposal: § Column Break Before: A column break will be performed before the object is output.
§ Column Break Condition: If the result is "True" when a data line is output in a multi-column table, a column break will be triggered. Tip: the "LL.CurrentTableColumn"
field returns the index of the current column.
(Page) Break Before: A page break is performed before the object is output. If you have multiple multi-column objects, a page break is triggered automatically after an object if the column counts for the objects are different (e.g. 2-column table followed by a 3-column table) and if there would be insufficient room for the object that follows.

ゅ
combit List \& Label

\section*{Advanced Functions}

In this chapter, we will concern ourselves with topics that you will probably only use very rarely. Nevertheless, the possibilities offered by linking objects and by sum and userdefined variables provide you with an important and useful tool for producing sophisticated printed outputs.

See also:
- Linking Objects
- Working With Sum Variables
- Working With User Variables
- Project Includes
- Displaying HTML Pages
- Adding OLE Documents
- Adding Form Template Objects
- Importing Objects

ゅ
combit List \& Label

\section*{Linking Objects}

By interlinking objects, you can influence the order of printing causing some objects to be printed after others thereby overlaying them in the event that they overlap ("sequential (temporal) linking"). Another possibility is to cause the size and position of some objects to be adjusted automatically to correspond to changes to other objects ("spatial linking"). Designer differentiates between three kinds of interlinking:
§ Sequential
§ Individual size and position adaptation
§ At end, keep size
When linking objects, there is a hierarchy: the main object and the attached (interlinked) object.

See also:
- Object List
- Creating Interlinks
- Sequential Interlinking
- The Individual Size and Position Adaptations
-The "at end, keep size" Interlink

\section*{Object List}

The object list defines the implicit print order of the objects, thus they are automatically sequentially interlinked with one another. Unlinked objects are printed first, followed by the tables and finally the objects that are linked in any other way.


Figure 7.1: Object list in the tool window
Example: In the above object list, the objects "company logo", "addressee", "invoice data", "dummy rectangle" and "page count" are printed first. The table "item list" comes next followed by the interlinked object "payment terms".

This means that if an object is to be printed over another object, it must be printed after this object. Sequential interlinking is therefore always necessary if an object that is not interlinked is to be printed after the interlinked objects, if an object is to be printed over a report container or if the contents are only available after another object has been printed.

\section*{Creating Interlinks}

You create interlinks or edit existing ones in the "Object List" dialog. This dialog opens up via the Object > Object List (Num followed by * on the number pad) or by double-clicking the "Objects" tool window.


Figure 7.2: Defining interlinks in the object list
You can interlink the selected object with other objects via the "Link with..." button. The link is shown in the object list in the form of a tree structure in which you can see the defined interlinks. You can also interlink several objects with one object.
When you select the interlinked object, you can choose from the three different kinds of interlink in the lower part of the dialog.
The "Detach link" button lets you remove an existing
interlink. The interlinked object will then be shown in the object list as an independent object.

\section*{Sequential Interlinking}

Sequential, or temporal interlinking makes sense if the content of the interlinked object can only be filled once the main object has been printed or if an object is to be printed over another object.

Example 1: You are printing an article list and want to output the number range of the articles on this page.


Figure 7.3: Range of article numbers at the top of the list But the "Article To" object only knows the last article number on the page once the table has been printed. Therefore, the "Article To" text object must be interlinked sequentially with the "Article List" table.
Select the "Article To" object in the object dialog and interlink it with the "Article List" table. The "Sequential" interlink type is already selected.
You must also interlink the table with the "Article From" object. Sequential interlinking is sufficient here as you don't want to change the size or position of the object.


Figure 7.4: Object list with sequential interlinking
Example 2: You want to output "Copy" over a table.
You therefore create a text object containing "Copy". Because objects that are not interlinked are printed first and tables are printed last, the text object must be linked sequentially with the table. Otherwise it would be printed before the table and therefore under the object.
\begin{tabular}{|c|c|c|c|c|}
\hline Quantly & No & Descnption & Unitfrice & Total \\
\hline 1.00 & EXPSAO1 & Southem Africa Eplorer 20-day tour from Cape Town to Victona Falls excluding fight & \$1,500.00 & \$1,500 00 \\
\hline 200 & EXPMALO1 & Maldves diving trip \(\leqslant\) thes. southern Male A! Beach "w \(\square\) & \$ \(\$ 1,800.00\) & \$3,80000 \\
\hline 200 & Exprucot & Yucatan, Oflo fhs of the & \$1,200.00 & \$2,400.00 \\
\hline 1.00 & TRPPRA01 & Fros Gotb to Baroque to Ant Noung Cty trip to Prague including accorrmodation & \$35500 & \$35600 \\
\hline 1.00 & TRVERA01 & Flight only Erazil & \$783.00 & \$783.00 \\
\hline
\end{tabular}

Figure 7.5: Text object over the list
Select the text object in the object dialog and interlink it with the "Article List" table. The "Sequential" interlink type is already selected.


Figure 7.6: Object list with sequential interlinking

\section*{The Individual Size and Position Adaptations}

Individual size and position adaptations are spatial interlinks, i.e. the size and position of the main object determines the size and position of the linked object. This automatically causes sequential interlinking as well.
If the size or position of the main object changes because the variables that it contains take up less room as the object provides, the linked objects adjust their size automatically to these changes.
Two kinds of interlinking are therefore available:
§ Position adaptation: If the position of a main object
changes, the position of the linked object changes too. You have three options here:
§ Relative to begin: The interlinked object moves in relation to the upper left corner of the parent object.
§ Relative to end: The interlinked object moves in relation to the lower right corner of the parent object.
§ To end: The upper edge of the interlinked object starts at the end of the main object, irrespective of its original position. This causes an implicit size change on the first page on which the child object is printed.
§ Size adaptation: If the size of a main object changes, the
position of the linked object changes too. You have two
options here:
§ Proportional: The size of the linked object changes exactly like that of the parent object. E.g. if the main object becomes 10 mm shorter, the interlinked object will also become 10 mm shorter.
§ Inverse: The size of the interlinked object is adjusted
inversely to the size of the main object. E.g. if the main object becomes 10 mm shorter, the interlinked object will become 10 mm longer.
You can specify whether the interlinking is to be horizontal and/or vertical in each case:
§ Vertical interlinking: The linked object adjusts its position or size to changes in the vertical position or height of the main object.
§ Horizontal interlinking: The linked object adjusts its position or size to changes in the horizontal position or width of the main object.

\section*{Example of vertical, relative to end position} adjustment: You print an invoice and want to output a closing text after the table.
You therefore create a text object and position it below the table.

> Sunshine agency

\section*{Lighteright Tin Ltd.
Dear Mr John Iron \\ 10 Steelestreet}

Metal City XY 12345

\begin{tabular}{|c|c|c|c|c|}
\hline Oumsty & No & Destription & UntPrice & Tax \\
\hline \multirow[t]{2}{*}{1.00} & EXPSA01 & Southem Atrica Explocer. 20-day tour from Cape Tomento Vktoria Falls excluding fight & \$1,500.00 & \$1,500.00 \\
\hline & & & & \[
\$ 0.00
\]
\[
50.00
\] \\
\hline
\end{tabular}

Figure 7.7: Text object below the list
Select the text object in the object dialog and interlink it with the table. Choose "Individual size and position adaptation" as the type of interlink. On the "Vertical Interlink" tab, enable the "Vertical" check box for the position adaptation and choose the "Relative to end" option so that the interlinked text object adjusts its size depending on the position of the lower right corner of the table.

\(\Rightarrow\) print time


Figure 7.8: Diagram of vertical, relative to end position adjustment
If the table becomes smaller, the text object moves upwards proportionally. It doesn't matter where the table ends, the text object will always be output after the table in the specified size.


We thank you for your order
Figure 7.9: Closing text below the list

> Example of vertical, relative to end position adjustment and vertical inverse size adjustment: On a multi-page invoice, the item table is to start on the first page below the address. On following pages it is to start at the upper page margin.
1. Create an invisible frame by inserting a rectangular
object.
2. The upper edge of the object is positioned precisely where the invoice table is to begin on the following pages. The lower edge is positioned precisely where the invoice table is to begin on the first page.
3. Assign the "Page( \()=1\) " appearance condition to the rectangle so that it is only printed on the first page.
4. Create a table. This begins directly below the rectangular object.


Figure 7.10: Invisible rectangular object above the table
5. Now interlink the table with the rectangle (main object) and select "Individual size and position adaptation" as the type of interlink.


Figure 7.11: Spatial interlinks in the object list
On the "Vertical Interlink" tab, choose the position option "Relative to end" and the size option "Inverse".


Figure 7.12: Interlink options in the object list


Figure 7.13: Diagram: Vertical, relative to end position adaptation and vertical inverse size adaptation
The table now changes its position based on the lower right corner of the rectangle (main object) and adjusts its height inversely in proportion.
The Page()=1 appearance condition prevents the rectangle from being printed on the second page. Therefore, it "shrinks" by 80 mm on the second page, and the table moves upwards proportionally and becomes 80 mm larger.


Figure 7.14: Position and size adaptation on the second page
Example: position adaptation, vertical to end: The interlinked object changes its position based on the main object. The size is also adjusted implicitly because the upper edge of the interlinked object changes based on the main object but the position of the main object does not change. The interlinked object must overlap the main object. The main object must be larger than the interlinked object. If the main object becomes smaller, the interlinked object only changes its position from above and increases its size. If the main object becomes larger, the linked object shrinks (the main object must be in the foreground).


Figure 7.15: Diagram: Position adaptation vertical, to end
Example: position adaptation, vertical, relative to begin: The interlinked object changes its position based on the upper left corner of the main object. This type of interlink
is the exact opposite of "vertical, relative to end". Example: The main object moves upwards due to the "Alignment bottom = True" setting in the Designer and the interlinked object follows this position adaptation in an upwards direction.


Figure 7.16: Diagram: Position adaptation, vertical, relative to begin

\section*{The "at end, keep size" Interlink}

This kind of interlink is similar to position adaptation. But here, the main object's available space is taken into consideration and the interlinked object always keeps its size. In other words, the object is always output within the boundary of the main object. If the available space is not sufficient, a page break will be triggered.
The interlinked object must overlap the main object in the Designer. It is crucial that the main object is always larger than the interlinked object. The interlinked object always tries to occupy the space remaining from the original size of the main object and, if the interlinked object is larger, this leads to an infinite loop because there is never enough space.


Figure 7.17: Function of the "At end, keep size" type of interlink

Example: Assuming you want to output a scanned signature after a text. The size of the signature must not change and it must be output within the border of the text object.

You therefore create a picture object and position it directly on the formatted text object. The height of the picture object is less than that of the table object.

Select the picture object in the object dialog and interlink it with the formatted text object. Choose "At end, keep size" as the type of interlink.
Irrespective of where the text ends, the picture object will always be output after the text in the specified size. If there is no longer sufficient space after the text, the picture object will be output on the next page so that the size can be kept. Dear Mr. Iron,
thank you for choosing Sunshine Agency for your trip. We are happy to send you your travel documerts today.
Enclosed you vill find your flight tickets, a travel guide and the rouke description with the
rental car to the hotel. Your personal travel attendant Michaela soleil will expect you on location.
10. Liblemir \(\begin{aligned} & \text { is about the booking, you can contact us by infoegsunshine- } \\ & 17531906010 \text {. }\end{aligned}\)

Your tr avel team from Sunshine Agency
Figure 7.18: The interlinked picture object overlaps the main object in the Designer
```

Dear Mr, Smith,

```
thark you for choosing Sunshine Agency for your trip. We are happy to send you your travel documents today.

Enclosed you will find your flight tickets, a travel guide and the route description with the rental car to the hotel. Your personal travel attendant Michaela Soleil vill expect you on location.

If you have any questions about the booking, you can contact us by infog sunshineagency, de or by phone 07531906010 .

Sunny greetings
Your travel team from Sunshine Agency
t. lioglenir

Figure 7.19: The signature is output at the end without changing the size
In this way you can also position multiple objects after each other (e.g. charts, pictures etc.).The "Base object" of the chain is the first object in the interlink hierarchy with an active page break.

ゅ
combit List \& Label

\section*{Working With Sum Variables}

Sum variables offer another way of creating totals and counters and work fundamentally across tables.
They are therefore a good choice whenever you want to create totals across different table hierarchies.
In all other cases, we recommend the use of the aggregate functions Sum(), Count() and Countlf() for totals and counters. Aggregate functions are always table-specific. You can even produce statistical analyses directly with aggregate functions e.g. Median(), Variance(), StdDeviation(). You will find a list of all functions in the "Aggregate functions" function category in the formula wizard.
Sum variables can be used to create totals over data sets, e.g. to add up the "Item.UnitPrice" fields in a table in order to calculate the total price. Such totals are permitted for all numeric variables or for expressions that return a numeric value as the result.
But sum variables are also a convenient way of defining a counter which can be updated accordingly for each data record that is printed.
You can create a total across all data sets of a printed page (page totals) or across the entire project (grand totals).


Figure 7.20: Totaling with sum variables.
Proceed as follows to define the variables:
1. Choose Project > Sum Variables or the "Edit sum variables" button in the formula wizard.
2. In the dialog that follows, create a new sum variable via the "Insert a new variable" button.
3. An input dialog appears where you can enter a description of the new variable. Give the variable a meaningful name, the "@" character will be added automatically as a prefix.
4. Click the "Edit" button to open the formula wizard and assign a field or an expression to the new sum variable. For example, select the numeric field "Item.UnitPrice" if you want to add up the "Item.UnitPrice" column. You can, however, also perform aggregations with complex expressions provided that the result is a numeric value. For instance, you can add up the gross price from the net price and the VAT. Enter the following expression, for example, in the "Sums over" field:
Item.UnitPrice+ Item.UnitPrice* (Item.VAT/100)
5. If you don't want to add up any values but merely want to create a counter or a numeration, the definition is simple: In the "Sums over" field, simply enter the value that is to be added to the existing counter.
The simplest case is a consecutive number that is increased by 1 for each data record. Just enter the value "1".
6. The "Page sum" checkbox lets you specify whether the totals are to be set to 0 at the end of a page. In this way, you can define page totals and counters.
7. Once you have defined which sums are to be stored in which sum variables, you can use these sum variables in your objects. In the formula wizard, you will find the sum variables at the end of the variable list in the "Sum variables" folder.

ゅ
combit List \& Label

\section*{Working With User Variables}

User variables are a way of saving values and expressions for later use. This saves you having to enter them anew each time if they are frequently needed in precisely this form or if user-defined data is to be output repeatedly. They are, so to speak, "formula building blocks".
You can then store these user variables in project includes if they are also to be used in other projects.
Note: User variables cannot be used within appearance conditions for layers.
You can also use the SetVar() and GetVar() functions if you only want to save values in the variable repository for later use. You will find more information about functions in the chapter Overview of Functions.
Proceed as follows to define user variables:
1. Choose Project > User Variables or the "Edit user variables" button in the formula wizard.
2. In the dialog that follows, create a new user variable via the "Insert a new variable" button.
3. An entry dialog appears where you can enter a description of the new variable. Give the variable a meaningful name, the "@" character will be added automatically as a prefix.
4. Click the "Edit" button to open the formula wizard and assign a field or an expression to the new user variable.


Figure 7.21: Edit user variables
5. You can use the arrow to specify the evaluation order. The variables are evaluated from top to bottom.
6. Once you have defined the user variables, you can use them in your objects. In the formula wizard, you will find the user variables at the end of the variable list in the "User-defined variables" folder.

ゅ
combit List \& Label

\section*{Project Includes}

If you design several similar projects, it's a good idea to include other projects as "Includes" to avoid having to create elements that occur repeatedly in each new project. In this way, you can easily include a letter head, for example, and any changes can be made centrally, e.g. if the design changes.
For example, if the address of your company changes, it can be updated in a single project instead of having to change hundreds of individual projects.
Or user variables that are used repeatedly (e.g. complex formulas in address fields) can be stored in includes.
You must consider the following restrictions: (1) Includes may not contain other includes. (2) Elements can be interlinked with elements in includes; the element in the include is always the parent element because it is printed first.
Totals and user variables are read and used in the same way. When designing includes, please make sure that you do not cause any overlaps (e.g. use a project with a sum variable as a include which is already contained in the current project). Via Project > Include, you can add other projects as includes. You can see the objects belonging to these projects but you cannot change them in the current project.


Figure 7.22:
Dialog for project includes
You can specify the name, visibility and appearance condition for each include by means of the property list. The buttons let you specify the (print) order. Project includes are always inserted at the beginning of the object list.

ゅ
combit List \& Label

\section*{Displaying HTML Pages}

HTML objects are used for displaying HTML content. You specify the HTML page by specifying the file name (e.g. combit.htm) or the URL (e.g. www.combit.net) in the object properties, or by means of the formula wizard. Also see the chapter Overview of Properties.

ゅ
combit List \& Label

\section*{Adding OLE Documents}
- Use the OLE container object to embed OLE server documents. In this way, you can embed documents from other applications (e.g. Word, Excel, Visio, MapPoint) in a report. Only the first page will be displayed as there is no standard for multi-page OLE objects.
You select the object type via the standard Windows dialog "Insert Object". Here you can choose an existing file ("Create from File") or create a new file.
Also see chapter Overview of Properties.

ゅ
combit List \& Label

\section*{Adding Form Template Objects}

Form templates are images, i.e. scanned forms, which you place in the background of your workspace as a template. This lets you position objects in a project precisely to fit the form. Although form templates are shown in the workspace, they are not printed and cannot be modified.
The best way of positioning them is by means of the property list.
To place a form template in the background of your workspace, use the form template object which you can insert via the Objects > Insert > Form Template menu item. Once you have created the template, it's best to use the objects tool window to select it. Form templates cannot be selected by clicking in the workspace.

ゅ
combit List \& Label

\section*{Importing Objects}

With File > Import, you can insert a copy of all objects belonging to another project to the project that is currently loaded.

ゅ
combit List \& Label

\section*{Different Layout Techniques}

You can influence the layout of your report in many different ways. In this chapter, we will examine the possibilities offered by different layout regions, define a multi-page report and control the page break behavior.

See also:
- Specifying the Page Layout
- Layout Regions

ゅ
combit List \& Label

\section*{Specifying the Page Layout}

13 Your first task in a new project is to set up the page layout that you want. Choose Project > Page Layout to specify properties such the choice of printer, paper size and orientation. There are different layout options depending on the project mode (label or list).
With multi-page projects, it is sometimes a good idea to choose different layout settings, e.g. printer, page size, orientation, paper bin, for the different pages. You will find more information about this in the chapter Different Layout Techniques.


Figure 8.1: Definition of the layout regions

See also:
-Printer Settings
- Export Media
- Templates for Label Formats
- Defining Your Own Label Format

\section*{Printer Settings}

By means of the properties, you can make different settings for each layout region for printer, page size, orientation, duplex print, number of copies, sort copies and paper bin
(e.g. first page on company letterhead and normal paper for the following pages).
The printer settings (and changed export format options) are saved in a special file (e.g. Article_List.Isp). If the respective file is not available when printing, the current default Windows printer is used.

See also:
- Size Adjustment
- Use Physical Page
- Force Paper Size
combit List \& Label

\section*{Size Adjustment}

Specifies whether the project is to be adjusted to fit the page when different printers are used when printing from the preview or whether the scale is to be kept.
combit List \& Label

\section*{Use Physical Page}

Specifies whether the whole physical page, including the non printable margin area, is to be available in the Designer. This is sometimes necessary in order to position labels correctly, e.g. if you use sheets of labels without margins. The nonprintable page margins are shown as hatched areas in the preview.
This enables you to use the complete page when defining the layout of your project but, of course, the printer cannot print these margins. If you place objects on such projects, you must still consider the non-printable margins. If this property is set to "False", only the area that can actually be printed is shown in the workspace.
combit List \& Label

\section*{Force Paper Size}

If there is no printer definition file, the application tries to force the page size set during design (e.g. Letter) as far as possible. This is only possible, however, if the selected printer supports either exactly this size or the "user-defined" option. If this is not the case, it will first check whether the printer's default size is large enough, otherwise it will choose the next largest size.

\section*{Export Media}

This list shows the various export possibilities offered by List \& Label.


Figure 8.2: Default setting for the output format
With the two buttons on the upper right, you can define a selected format as the default value for the later print and specify the options for this format. These options are saved in a special file (e.g. Article_List.Isp).

\section*{Templates for Label Formats}

Via the "Templates" tab in the page layout for labels, you can make your selection from numerous predefined label formats from different manufacturers. This automatically specifies the size of the individual labels, how many are to appear on the sheet and how they are to be distributed.

\(\square\) Qnly show templates for curent page retting?
(2)

> OK Cancel

Figure 8.3: Definition of the label size

\section*{Defining Your Own Label Format}

You can also define your own label formats if you can't find the layout that you want among the templates. You can make the required settings with the "Layout Definition" region property; there is a special dialog for this:
§ Offset: The offset specifies the horizontal or vertical distance of the upper left label to the margin of the chosen page region (physical/printable)
Note: In the screen display, the upper left corner of the workspace always starts at coordinates 0/0 irrespective of the chosen page size and specified offset. However, you will see the effect of the offset in the preview or when printing.


Figure 8.4: Defining customized label layouts
§ Size: This value defines the size (horizontal=width / vertical=height) of the label.
§ Distance: The distance to the next label is specified here.

With single-column labels, only the vertical distance must be entered.
§ Number: This option specifies the number of labels per page (horizontal number = number of columns per page, vertical number = number of rows per page).
§ Print Order: Specifies the order of the print if multiple labels are printed on a page. Possible values: 0 (horizontal), 1 (vertical), 2 (horizontal bottom up), 3 (vertical bottom up).
The default is to print labels row by row from the upper left to the lower right (horizontal). However, in the case of sheets of labels that have already been started, it is possible that the first label row has already been used. This causes a loss of stiffness in the upper part of the sheet. Many printers have problems feeding in sheets that have already been started, resulting in a paper jam. In this case, it helps to print the labels in reverse, from the bottom upwards instead of from the top downwards. In this way, the upper row of labels on the sheet is always printed as the last and the sheet retains the stiffness required for feeding in.

See also:
- Saving your own label formats in the label template list

\section*{Saving your own label formats in the label template list}

To save your own label formats, you can edit the file "cmll1601.inf".
Layout of a label definition (all measurements in 1/1000 mm):
\(<\mathrm{A}><\mathrm{B}\rangle,<\mathrm{C}\rangle=<\mathrm{D}\rangle,<\mathrm{E}\rangle,<\mathrm{F}\rangle,<\mathrm{G}\rangle,<\mathrm{H}\rangle,<\mathrm{l}\rangle,<\mathrm{J}\rangle\), <K>
A: code, B: description, C: page size, D: label width, E: label height, F: horizontal distance between labels, G: vertical distance between labels, H: number horizontal, I: number vertical, J: margin left and right, K: margin top and bottom
e.g. 3420 universal labels, \(70 \times 16.9 \mathrm{~mm}=70000,16900,0\), \(0,3,17,0,4850\)

ゅ
combit List \& Label

\section*{Layout Regions}

With multi-page projects, it is sometimes a good idea to choose different layout settings, e.g. printer, page size, orientation, paper bin, for the different regions. You define the layout regions with Projects > Page Setup. You create a new region with the relevant button on the "Layout" tab and then define the properties for this region. You will find a detailed explanation of the properties in chapter Overview of Properties.

See also:
- Active Design Layout
-Practice: Report With Different Page Orientations
- Practice: Managing Issues (Copies)
-Practice: Payment Form on the Last Page

\section*{Active Design Layout}

If you define different layouts, you can decide which layout is to be displayed as the workspace. The "Active Design Layout" project property lets you choose from all defined layout regions.


Figure 8.5: Selecting the active design layout

\section*{Practice: Report With Different Page Orientations}

Let's assume that you want to change the page orientation within a report: the first section with the bar chart is to be printed in portrait mode, the second section with the cross tab in landscape mode and the remaining section with the pie charts in portrait mode again.
The report should look like this:


Figure 8.6: Report with different layout regions
Proceed as follows:
1. Position the report container on the workspace and create the bar charts, pie charts and the cross tab. You can find more information about this in the chapter Producing Analyses.
2. Since you want to change the page orientation, you must also adjust the height and the width of the report
container. If you don the this, the crosstab data will be truncated when the report container is positioned in portrait mode.
Select the report container in the "Report Structure" tool window and use the LL.Device.Page variables and the UnitFromSCM() function to specify the height and width:

\section*{EPosition}
[0.39, 0.39, 7.86, 11.29 in]
\begin{tabular}{|c|c|c|}
\hline - & & \\
\hline Left & UnitFromSCM(10000) & [0.39 in] \\
\hline Top & UnitFromSCM(10000) & [0.39 in] \\
\hline Width & LL.Device.Page.Size.cx - UnitFromSCM(20000) & [7.47 in] \\
\hline Height & LL.Device.Page.Sizecy - UnitFromSCM(20000) & (a) \\
\hline
\end{tabular}

Figure 8.7: Size of the report container defined variably
3. Create headings by adding a Table > Free Content element in each case.
4. Enter "Orders per Customer" as the name of the crosstab's heading. This element name will then be available later in the "LL.CurrentContainerltem" field.
5. So that the titles are always printed at the beginning of a page, set the "Pagebreak before" property to "Yes" in each case. This will produce a page break before outputting the element.
6. Now define the layout regions. Select Project > Page Setup.
7. The dialog for defining the layout will now appear. The default region "Standard Layout" is always the last area with the "True" condition and cannot be renamed. Leave "Portrait" as the orientation for this layout.


Figure 8.8: Layout definition for a particular element
8. Create a new region with the "New" button. Enter a meaningful name for this layout in the "Description" property, e.g. "Landscape".
9. Then specify the specific properties for this layout. Change the "Orientation" property to "Landscape".
10. You can specify when this layout region is to be used as a "Condition." In this example, the mode must change when the element with the name "Orders per Customer" is printed. The formula for the logical condition is therefore: LL.CurrentContainerltem = "Orders per Customer".

\section*{Practice: Managing Issues (Copies)}

Let's assume that you want to produce two copies of an invoice. The first copy is to be printed on a company letterhead in paper bin 1. The second copy is to be printed on the cheaper paper in paper bin 2.
1. In the project properties, enter " 2 " in the "Number of issues" property.
2. Open the dialog for the report container via Project > Page Setup.
3. Create a new region "Original". In this case, use the IssueIndex() function as the "Condition". This function returns the number of the issue. So you define the logical condition "IssueIndex()=1". Select paper bin 1 for this layout.
4. Create a layout called "Copy" and define the logical condition "IssueIndex()=2". Select paper bin 2 for this layout.


Figure 8.9: Definition of the layout regions

If you want to endorse the second copy additionally with the text "Copy" over the invoice table, carry out the following steps:
1. Create a text object containing "Copy".
2. Because objects that are not interlinked are printed first and tables are printed last, the text object must be linked sequentially with the table. Otherwise it would be printed before the table and therefore under the object.
3. Select the text object in the object dialog and interlink it with the Report Container. The "Sequential" interlink type is already selected.


Figure 8.10: Object list with sequential interlinking
4. Now select the text object again and set the value of the "Display condition for issue print" object property to Issuelndex()=2. This text object will now only be printed on the second copy.
\begin{tabular}{|c|c|c|c|c|}
\hline Cuartsy & No & Description & UnitPrice & Total \\
\hline 1.00 & EXPSADI & Southern Africa Explorer 20-day tour from Cape Town to Victona Falls excluding fight & \$1,500.00 & \$1,500.00 \\
\hline 200 & EXPMALO1 & \begin{tabular}{l}
Maldives diving trip \(<4\) s. southern Male At \\
Beach \({ }^{\prime 2}\) ench \\
ng) gint
\end{tabular} & \$1,800,00 & \$3,600.00 \\
\hline 200 & EXPYUC01 & Yucatan, C PThe 1 of the May itionerrg indtp. & \$1,20000 & \$2,400,00 \\
\hline 100 & TRPPRA01 & Fro coil to Baroque to Ant Noum Cey trip to Prague including accommodation & \$355.00 & \$35500 \\
\hline 1.00 & TRVBRA01 & Fight only Erazi & \$783.00 & \$783.00 \\
\hline
\end{tabular}

Figure 8.11: Text object over the list

\section*{Practice: Payment Form on the Last Page}

Let's assume that you want to print a payment form on the last page of a multi-page invoice. The payment form is preprinted on a special paper in paper bin 2 in the lower range of the page. The field contents have to be printed exactly at the correct positions.
Proceed as follows:
1. Open the dialog for the layout regions by Project > Page Setup.
2. Create a new region "Last page". Use the function Lastpage() as a condition that returns the value "True" if the last page is printed. Choose paper bin 2 for this region so the last page will always be printed on the special payment form preprint from paper bin 2.
3. Position the text objects for the field contents of the payment form exactly at the correct position. You can put a scanned-in picture of the payment form in the background by choosing/clicking Objects > Insert > Form Template.
4. Set the appearance condition of all text objects to Lastpage() to make sure they are only printed on the last page.
5. You have to link all text fields with the table as the function Lastpage() can only be evaluated correctly in tables, layout regions or objects linked to tables. Links are created and edited in the "Object List" dialog. Open this dialog by choosing/clicking Objects > Object List. Select the table object and link via text object of the payment form to it by clicking "Link with...". The interlink
type "Sequential" is already selected. The sequential interlink is sufficient here as no changes in object position or object size are desired.
Repeat these steps for all text objects.
6. Now you have to avoid that the fields of the payment form are printed on top of the table if the table ends on the last page in the area of the payment order. Therefore create a placeholder by inserting a rectangle object without border and filling. The rectangle has the exact height of the payment form and has to overlap the table necessarily! Select the table in the object dialog and link the rectangle to it. Choose "At end, keep size" as interlink type. No matter where the table ends, the rectangle is always output in the selected size after the text. If there is not enough space after the table, the rectangle is output on the next page and a page break is triggered.


Figure 8.12: Invoice with fields and placeholders for payment order

ゅ
combit List \& Label

\section*{Output Options}

There are two ways of printing projects: Start the print from the higher-level application or via the real data preview in the Designer (if supported by the application).

See also:
- Output Options
- Real Data Preview
- Export in another Format (PDF, XLS ...)
, Print Sample in the Designer

ゅ
combit List \& Label

\section*{Output Options}

You can start the print function directly from the higher-level program, via a menu item or from the preview.
If you start the print from the higher-level program, the print settings dialog will normally appear once you have selected the project to print.


Figure 9.1: Output options
You have various configuration options here:
§ Under "Print Target", you can change the printer settings. If you have defined different layout regions in the page layout dialog, you can also change the printer settings here for the various regions. Enter the number of copies under "Issues".
§ Select the output format (e.g. preview, printer, PDF) under "Direct to".
§ The "Preview" option lets you view the output first on the
screen as it would be printed. In this way, you can check the result before printing.
§ If you select the "Save options permanently" setting, your choice of printer and output format will be saved as default values for this print template.
§ With "First Page", you specify the page with which the print is to start.
§ "Pages" lets you select certain pages or a page range for printing, e.g. 1, 3-4, 10-.
§ The "Print" drop-down list lets you restrict the output to even/odd pages or the pages that you have selected above.
§ When printing labels, you have an extra "Select" button which you can use to specify the position where you want to start printing the sheet of labels.
In this way, you can also print sheets of labels that have already been partly used. You will find a sample label sheet for your label project in the dialog for selecting the start position. Click the label where the print is to start. Please take note of the selected print order. You can print not only in rows from upper left to lower right but also column for column or in the reverse direction. The labels will be printed in the specified direction starting with the selected label.


Figure 9.2: Additional print settings for labels

ゅ
combit List \& Label

\section*{Real Data Preview}

The real data preview function is also available directly in the Designer provided that your application supports it. In this way, you can check the layout of a printout without wasting paper in order to do so. The screen preview is accurate down to the last detail (WYSIWYG What You See Is What You Get), exactly as it would be when printed. After checking the layout, the actual print can be started from the preview without having to use the print command again.
§ On the "Pages" tab, the individual pages are shown as miniatures to let you navigate quickly and directly. Via Project > Options > Preview, you can specify the maximum number of pages that are to be displayed in the real data preview.
§ You will find the table of contents on the "Outline" tab. With Project > Options > Project, you can specify the maximum folder depth of lines and objects to be used as bookmarks in the preview. Table rows and objects have the "Outline level" property for defining the table of contents.
§ The "Number of copies" specified in the print options is ignored in the preview as this is only relevant for the actual print.
§ The preview window contains a toolbar which you can use to select the different preview functions.


Figure 9.3: Real data preview in the Designer

ゅ
combit List \& Label

\section*{Export in another Format (PDF, XLS ...)}

You can also output a print in different file formats.
To do this, choose the relevant output medium in the Print Options > Print Setup dialog, which you reach via File > Export or via "Save as" from the preview.
Please consider that, due to the format, the layout cannot always be taken over 1:1 because there are specific restrictions when converting to these formats.
Many formats have various possibilities for configuration. You can make use of these by clicking the "Options" button. § For example, with Excel formats, you can reduce the number of columns and rows to the minimum required by means of the "Only data from table object(s)" option.
§ With PDF exports, you will find the option for creating a file that is PDF/A compliant. You will find more information about creating a PDF table of contents in Chapter Levels in the Table of Contents.


Figure 9.4: Output settings for PDF
When you start the print, choose the storage location in the following "Save As" dialog and enter a name for the file to be created in the "File name" field.
§ By checking the option "Open the file in the registered application after the output", you can display the file after creation directly in the respective program (e.g. Excel).
§ The option "Send exported files by email" lets you send the files directly by email.
§ You can add a digital signature to your files by means of the "Digitally sign created files" option (not available in all applications).


Figure 9.5: Output options

ゅ
combit List \& Label

\section*{Print Sample in the Designer}

Choose File> Print Sample for a test print of your current project. In the print sample, fixed text appears as it is defined in the project; variables and fields however are replaced by a predefined sample text or by a single repeated sample data record.

ゅ
combit List \& Label

\section*{Variables, Formulae and Expressions}

Information such as a return address line on an address label or a heading over a list can be entered directly into the project as "fixed text". Fixed text is printed exactly as it is held in the project.
Alternatively, this information can be taken over dynamically from a higher-level program. Such information is entered into the projects as "variables" or "fields".
Fields are the data that changes for each line of a table. Variables remain the same throughout the page. For example, you might choose the TELEPHONE field for the contents of a column of a table. The different telephone numbers for the data records in the database are printed in this column. In other words, variables and fields are placeholders.
With these two kinds of information, attractive projects can already be designed that are adequate for many purposes. However, List \& Label Designer offers much more. With the aid of formulas and expressions, the information held in variables and fixed text can be joined or modified in almost any conceivable way. The "formulas" and "expressions" make this possible. In formulas and expressions, fixed text and variables can be used in "functions" and joined by "operators".
For example, with projects for printing address labels, you can use an expression to automatically add the text "PO Box" to a PO Box number held in a variable called POBOX. In this way, not just the number alone will be printed on the label but something like "PO Box 111111" instead.
Or, consider this: The net price of an article is held in a variable called PRICE. However, you want to print the price
including VAT in your list. A formula that calculates the VAT from the net price and then adds it on will help you here. The gross price will then be printed.

See also:
- Variable List
- The Elements of an Expression
- Working With Functions
- Working With Operators

ゅ
combit List \& Label

\section*{Variable List}

The variable list displays all variables available in the current project; for list type projects, all available fields are also displayed.
The hierarchical list differentiates between variables, fields, database tables, user variables and sum variables.
User-defined variables and fields can also be structured hierarchically. The contents of variables normally remain unchanged at least throughout a page; fields change from table row to table row.
If you want to assign variables or fields to existing objects, you can simply select the variables and fields that you want in the list and drag them to the object with the mouse (Drag \& Drop). List \& Label inserts them automatically where possible. If you drag a variable to a free area on the workspace, a new text object will be created there. The size relates to the size of the last object whose size was changed.


Figure 10.1: Variable list

ゅ
combit List \& Label

\section*{The Elements of an Expression}

Fixed text, variables, formulae, functions, operators and the like \(\geqslant\) generally referred to below as "Elements of expressions" are all inserted and combined by means of a common dialog.
The formula wizard helps you with your entries in several ways:
§ Function syntax display: A tooltip appears describing the chosen function; it lists the required parameters and shows the result type.
§ Auto complete: When you type a letter, the available functions, fields and variables are listed that begin with this letter. Within functions, suitable values are suggested for parameters.
§ Syntax coloring: Functions, parameters, operators and comments are shown in different colors.
§ Automatic type conversion: Variable and field types are converted automatically when inserted in existing expressions to ensure that the data type corresponds to that expected.


Figure 10.2: Autocomplete in the formula wizard

See also:
- Different Expression Modes
- The Tabs
- The Editing Line
- Inserting Variables
- Insert Fixed Text
- Inserting Comments

\section*{Different Expression Modes}

Please note that there are two ways of writing expressions. Which mode is being used is set by the application.
On the one hand, there is the normal expression mode, in which you can enter names of variables and functions without brackets. Fixed text must be enclosed in quotation marks. The individual variables must be joined with the "+" operator.
On the other hand, there is the extended mode, in which you can enter fixed text without quotation marks. You must enclose variables with "<" and ">" and functions with chevrons ("«" and "»").
«» In this mode, you can insert the chevrons by clicking the "Insert chevrons" button (e.g. if you want to enter a function directly". You can also use Alt+174/175. It is not necessary to use an operator to join individual operators in this mode. The extended mode is easier to use.


Figure 10.3: Extended mode

\section*{The Tabs}

This dialog consists of a series of tabs, each containing different elements to be edited.
\begin{tabular}{ll}
\hline Tab & Contains the elements \\
\hline Variables and & \begin{tabular}{l} 
The variables and functions available for this \\
object type
\end{tabular} \\
\hline Functions & \begin{tabular}{l} 
Special dialog for defining IF-THEN-ELSE \\
conditions
\end{tabular} \\
\hline Textition & \begin{tabular}{l} 
Dialog for entering fixed text and options for \\
setting tab stops (only text objects)
\end{tabular} \\
\hline Date Format & Available date formats \\
\hline \begin{tabular}{l} 
Number \\
Format
\end{tabular} & Available number formats \\
\hline Operators & Available logical operators
\end{tabular}

You will find an "Insert" button on each of these tabs with which you can add the selected element to the editing line. A double-click on the relevant element has the same effect. In addition, you can also add the elements to the editing line with Drag \& Drop (also in and from the function list).
For the different elements of an expression (variables, text, functions etc.), certain rules apply for the way of writing and for joining individual elements to give an expression. A wizard integrated in the dialog makes sure that these rules are observed. For this reason, you should always add the different elements to the editing line by means of the respective tab in this dialog. Thus, you should use the "Variables and Functions" tab to enter variables and the "Text" tab to enter text etc.
Experienced users can also enter the expression that they want directly in the edit box or modify the text that is there (e.g. put something in brackets).

\section*{The Editing Line}

The editing line contains the expression that you have compiled by means of the various tabs, entered directly or created with Drag \& Drop.
The expression is checked continuously as you create it to make sure that the syntax is correct. Any syntax errors are shown in the information pane under the editing line, together with an explanation of the cause of the error. The syntax checker will normally produce an error until the expression is complete. Don't let this worry you. When the expression is complete, the resulting text should be shown with the Designer's sample data.
To make complex expressions clearer, you can split them across several lines with Ctrl+Return. This has no effect on the result.
With the different buttons on the right next to the input field
you can
§ mark brackets belonging to the formula expression.
\(\S\) mark the expression between matching brackets.
§ edit sum and user variables.
§ undo the last operation.
§ redo the last undo operation.

\section*{Inserting Variables}

There are different data types for variables: "string", "number", "date", "Boolean" (logical values), "picture" and "barcode". The data type is important if you want to use variables as parameters in functions as they normally only accept certain data types. Thus you can only multiply a numeric value with a numeric value.
The "Variables and Functions" tab includes an overview of all available variables, an icon indicating the data type in front of the variable as well as the available functions.
You can filter the variables by means of the input field above the variable list.
To add a variable, double-click the variable that you want, use the "Insert" button or drag the variable to the editing line (also via the function list). The variable in question will be added to the editing line in the correct syntax.
Repeat the above steps to add more variables to your expression. If you want to have spaces between the individual variables, e.g. to separate FIRST_NAME and NAME, make sure that you enter this space in the editing line.


Figure 10.4: Joining variables and free text
You can also insert variables by "dragging" the one you want to the target object in the workspace with Drag \& Drop. The variable is then automatically added to the object as a new line.

\section*{Insert Fixed Text}

Another important element in expressions is fixed text, with which you can prefix a variable with an identifier, e.g. "Telephone: 1234567".
With the "Text" tab, you can insert free text in your expression, set tab stops and page breaks.
Enter the text that you want and click "Insert" to add your entry to the editing line. The text will be placed automatically in quotation marks.


Figure 10.5: Fixed text in the formula wizard
In the above example, the fixed text "Name" is first inserted via the "Text" tab followed by the variables "Customer.Title", "Customer.Firstname" and "Customer.Lastname" by means of the "Variables and Functions" tab. The "Name:" text will then be printed first followed by title, first name and last name from the database.

Please consider that spaces that are to appear between variables or between variables and text, e.g. as separators, also count as "fixed text".
Depending on the mode, variables and fixed text cannot be simply placed together but must be joined by the "+" joining operator. In this example, the fixed text "Name" is joined to the "Customer.Title" variable with the "+" operator.

See also:
- Inserting Linefeed
- Inserting Tab Stops

\section*{Inserting Linefeed}

With the "Linefeed" button, you insert a line break ("9") into your text line.

Linefeed
Figure 10.6: Insert line break by clicking the button
However, such a break only has an effect if you have specified that breaks are allowed for the object in question (line of a text object or column of a table object). In this case, the words that don't fit in the line/column are continued on a new line causing the lines below to be moved down by one line. (Caution: If the text contains just one long word, it will not be broken but will be truncated instead). With text objects, the value of the "Line Wrap" property in the property list for the respective paragraph must be "Wrap".
With table objects, the value of the "Fit" property in the property list for the respective column must be "Wrap".

\section*{Inserting Tab Stops}

Tab stops are only allowed in text objects. Therefore this button is not displayed in table objects.

Iab
Figure 10.7: Insert tab stop by clicking the button
Just one tab stop per line may be inserted between text elements or variables. Its position can be specified in the "Tab Properties" dialog which appears when the button is pressed. You will find more information under Text Objects.

\section*{Inserting Comments}

You can add comments to formulas provided that this is supported by the application. You can do this in two ways:
§ "/* <text> */" for comments in the middle of a formula
\(\S " / *<t e x t>"\) for comments at the end of the formula. With this variation, all the remainder of the formula becomes a comment \(\geqslant\) not just the line.


Figure 10.8: Inserting comments

ゅ
combit List \& Label

\section*{Working With Functions}

Functions open up countless possibilities and make defining expressions really interesting. With the aid of these functions, you can calculate values, influence the results of variables or their appearance, convert value types and perform many more tasks.

See also:
- Notation of Functions
- Value Types
- Overview of the Functions
- Using Functions

\section*{Notation of Functions}

Functions all use the same notation which is based on the BASIC programming language: return value \(=\) function(arguments)
You only specify the function and the arguments. Whether you use capitals or small letters is irrelevant for the names of the functions, but not for the arguments. Capitals or small letters are especially important when variables are used as arguments.
List \& Label evaluates the expression and interprets it replacing the "function(arguments)" part by the "return value".
I.e., the return value.is produced from the function(arguments) input line.
The elements have the following meanings:
\begin{tabular}{ll}
\hline Element & Meaning \\
\hline Function() & \begin{tabular}{l} 
The name of the respective function in its \\
correct syntax. The brackets () for the \\
arguments belong to the function name. The \\
brackets must always be present even if a \\
function does not have any arguments.
\end{tabular} \\
\hline Arguments & \begin{tabular}{l} 
The values that a function uses in order to \\
produce the return value. The arguments follow \\
the name of the function immediately without \\
any spaces in between. A function can have \\
zero, one or more than one argument(s).
\end{tabular} \\
& \begin{tabular}{l} 
Functions usually expect arguments of a certain \\
value type (see below). It is important that the \\
value types of the arguments conform to the \\
types expected by the function.
\end{tabular} \\
\hline Return value & \begin{tabular}{l} 
The value that List \& Label returns as the result \\
of a function. The type of the return value \\
depends on the function in question or the \\
value types of the arguments.
\end{tabular} \\
\hline
\end{tabular}

\section*{Value Types}
\begin{tabular}{ll}
\hline Value type & Explanation \\
\hline Boolean & \begin{tabular}{l} 
The logical values "True" or "False". If the \\
condition is met, the result is true otherwise \\
false.
\end{tabular} \\
\hline String & \begin{tabular}{l} 
Any string. This string can contain letters, digits \\
and special characters. It must be placed in \\
quotation marks ("") so that it can be \\
differentiated from names of variables.
\end{tabular} \\
\hline Date & \begin{tabular}{l} 
Date values according to the Julian calendar. \\
\hline Number \\
\begin{tabular}{l} 
A string containing only the digits 0 - 9, the \\
decimal point and the minus sign, other \\
characters are not permitted. Number strings \\
do not have to be enclosed in quotation marks.
\end{tabular} \\
\hline Barcode
\end{tabular} \begin{tabular}{l} 
A string that is made up of the characters used \\
for barcodes.
\end{tabular} \\
\hline Picture & One of the supported picture formats. \\
\hline RTF & Formatted text \\
\hline
\end{tabular}

\section*{Overview of the Functions}

You will find an overview of the available functions on the "Variables and Functions" tab. You will also see an explanation of the currently selected function. The explanation informs you of the nature of the function and the type of the arguments that it expects (parameters).
If no arguments are given for a function, this means that the function does not expect any (apart from the empty brackets). Otherwise, the function expects exactly the number of arguments as shown. Arguments that appear in square brackets ([ ]) are optional, i.e. they can be omitted. The argument "All" means that the argument can be any of the following types (Boolean, string, date, number, picture, barcode, RTF).
You will find a more detailed explanation of all functions and parameters under Overview of Properties.


Figure 10.9: List of functions with help text and tooltip
The functions available on the "Variables and Functions" tab are sorted alphabetically or shown by function group. There are the following function groups:
§ Numerical functions
§ Mathematic functions
§ Date functions
§ Character functions
§ Miscellaneous functions
§ Aggregate functions
§ Barcode functions
§ Conversion functions
§ Binary functions
§ Drawing functions
§ Project and print-dependent functions
§ Logical functions
§ Currency functions
You can filter the functions by means of the input field above the list of functions.
When you select a function, you are shown a short
explanation in the information pane at the bottom. To add the function to the editing line, double-click the function that you want or use the "Insert" button.

\section*{Using Functions}

Examples of some selected functions are explained below.

See also:
- Convert a Number to a String With Str\$()
- Convert a String to a Number With Val().
- Convert a String to a Date With Date()
- Convert a String to a Barcode With Barcode()
- Convert a String to a Picture With Drawing()
- Truncate Strings
- Formatting a Data Value With Date\$()
- Formatting a Number Value With FStr\$()
- Formatting Currencies With LocCurrL\$() or LocCurr\$()
- Page Numbers With Page\$() or Page()
- Counting Values With Count()
- Only Count Certain Values With Countlf()
- Totaling With Sum()
- Obtaining User Input via a Dialog With AskString\$()
- Appearance Conditions With Lastpage()
- Logical Conditions With Cond()
- Working With Null Values

\section*{Convert a Number to a String With Str\$()}

The Str\$ function converts a number to a string.
The (optional) second parameter specifies the length of the string. However, if this number is too large for this format, the resulting string may be longer than you want. If the number is too small, the result will be padded with spaces, according to the sign, on the right (negative) or the left (positive).
The (optional) third parameter specifies the number of decimal places. If it is positive, the number is shown as a floating point number or in scientific notation if it is negative. If the third parameter is not specified, the number of decimal places is set to 0 with integers and, for reasons of compatibility, to 5 with floating decimal values. Examples:

Str\$ (Constant.Pi()) result: "3.14159"
Str\$ (Constant.Pi(),3) result: "3.14159"
Str\$ (Constant.Pi(),3,0) result: " 3"
Str\$ (-Constant. Pi(),12,-3) result: "-3.141e+00"
Str\$ (Page()) result: "5.000000"
Str\$ (Page(),10) result: " 5"
Str\$ (Page(),-10) result: "5
To enter the expressions, proceed as follows:
1. Locate the function in the alphabetical list or filter the functions by entering Str\$ in the filter field above the list.
2. Double-click the "Str\$ ()" function to add it to the editing line. This also adds placeholders for the parameters that are expected or accepted by the function. The first of these parameters is selected automatically and you will be prompted to replace this placeholder with a valid value. It's a good idea to first replace all the function's
parameters with the respective values before you go on to define the expression.
Functions are also accepted as values for most parameters. List \& Label Designer takes care of the correct syntax provided that you also use the "Functions" tab for entering them.
3. The "Constant. Pi()" function was inserted here as a parameter \{number\}.
4. To insert a variable and a function at the same time, use the mouse to drag the variable that you want to the relevant function folder (e.g. "Numerical functions"). The folder opens up automatically and you can select the function that you want with the mouse. If you move the mouse upwards or downwards, the list scrolls automatically in the corresponding direction. If you "drop" the variable onto a function, the function will be inserted with the selected variable as the first parameter.
combit List \& Label

\section*{Convert a String to a Number With Val().}

The Val() function converts a string to a number. If there is an error, the result will be 0 . The decimal point character must always be given as "."
Example:
Val ("3.141") result: 3.14
The "LocVal()" functions converts a string to a number and presents the result in a format that is valid for the country. Examples:

LocVal ("12.00","DEU") result: 12,00
LocVal ("12.00","USA") result: 1200,00
combit List \& Label

\section*{Convert a String to a Date With Date()}

The Date() function converts a string to a date. When doing so, the separator is evaluated accordingly:
Example:
Date ("04.07.1776") result: 04.07.1776
combit List \& Label

\section*{Convert a String to a Barcode With Barcode()}

The Barcode() function converts a string to a barcode. This function can only be used in a table or barcode object. For the second parameter, the wizard offers you the possible barcode types as autocomplete options. Some barcodes have special formats which must be adhered to. You will find detailed information about this in chapter Barcode Objects. Example:

Barcode ("Hello World","GS1 128")
combit List \& Label

\section*{Convert a String to a Picture With Drawing()}

The "Drawing()" function converts a string to an image file. Example:

Drawing("sunshine.gif")

\section*{Truncate Strings}

The "Left\$()" function shortens a string from the left by a specified number of characters. The "Right\$()" function shortens a string from the right and the "Mid\$()" function cuts out part of the string.
The second parameter specifies the maximum number of places in the result.
Examples:
Left\$ ("combit", 1) result: "c"
Mid\$("combit",1,2) result: "om"
Right\$("combit",3) result: "bit"
The "StrPos()" and "StrRPos()" functions return the position of the \(\mathrm{n}^{\text {th }}\) occurrence of a search string in a string. You can supply a third parameter specifying which occurrence of the search string is to be returned. The first character of the string corresponds to position 0 . This means that with this function, you can extract a substring from the string, e.g. from the first space onwards.
Example:
Left\$ ("John Smith",StrPos("John Smith"," ")) result: "John"
The "Rtrim\$()" function removes spaces at the end of a string, the "Atrim\$()" function removes them from the beginning and the end of a string.
Example:
RTrim\$ ("Hello World ") result: "Hello World"

\section*{Formatting a Data Value With Date\$()}

You can format date values with the Date\$() function. To avoid having to enter the formatting parameters yourself, you can select the parameters from a list on the "Date Format" tab.
In this way, for example, you can specify whether days or months are to be written in words or whether the year is to be output with two or four digits etc. You should generally proceed by first selecting the date format that you want from the "Date Format" tab and then insert the value to be formatted or the expression as a parameter. In the format list, you will see the respective formatting instructions on the left and the respective result on the right. The "Now()" function, which returns the current date, is selected automatically as the date value. But if you want to format a different date value, simply replace "Now()" with the value that you want. You will find more information about the "Date\$ ()" function under Overview of Functions.


Figure 10.10: Example with Date\$()

\section*{Formatting a Number Value With FStr\$()}

You can format numerical values with the FStr\$() function. To avoid having to enter the formatting parameters yourself, you can select the parameters from a list on the "Number Format" tab.
In this way you can specify the number of positions before and after the decimal point, leading zeros and similar. You should generally proceed by first selecting the number format that you want from the "Number Format" tab and then insert the value to be formatted or the expression as a parameter. With complex expressions containing calculations, make sure that you format the result and not a value in the calculation formula. Otherwise you will not be able to perform the calculation.
In the format list, you will see the respective formatting instructions on the left and the respective result on the right. You can add the number format that you want to the editing line by double-clicking or with "Insert".
Insert the number value to be formatted as a parameter. You will find more information about the "FStr\$ ()" function under Overview of Functions.


Figure 10.11: Example with Fstr\$()

\section*{Formatting Currencies With LocCurrL\$() or LocCurr\$()}

The "LocCurrL\$()" function returns a string with the currency format and symbol that is valid for the country. Insert the number value to be formatted as a parameter. The (optional) second parameter is the ISO 3166-
Country code for the country whose currency format is to be used.
Example:
LocCurrL\$ (1000) result: "\$1,000.00 "
The "LocCurr\$()" function returns a string with the currency format that is valid for the country but without the currency symbol.
Example:
LocCurr\$ (1000) result: "1,000.00"
You will find more information about this function under Overview of Functions.
combit List \& Label

\section*{Page Numbers With Page\$() or Page()}

The "Page\$()" function returns the page number of the page being printed as a string.
Example:
"Page " + Page\$ () + "/" + TotalPages\$ () result: Page
1/3
The "Page()" function returns the page number of the page being printed as a number, thus giving you the opportunity of using an expression or a formula, e.g. in conditions.
Example:
Cond(Page()>1, "Page " + Page\$())
combit List \& Label

\section*{Counting Values With Count()}

The "Count()" function counts the number of values of the first argument.
The first parameter specifies the values to be counted. The (optional) second parameter specifies whether the values used for the calculation are to be deleted after outputting. Examples:

Count (Order_Details.ProductID)
Count (1)

\section*{Only Count Certain Values With Countlf()}

The "CountIf()" function counts the number of values that satisfy the condition. You should also use the "Distinct()" function if multiple occurrences of values are only to be counted once.
The first parameter specifies the expression for the compare.
The (optional) second parameter specifies whether the values used for the calculation are to be deleted after outputting (default: True).
Examples:
Countlf (Customers.Region="D")
Countlf (Distinct(Customers.Region="D")) multiple occurrences of values are only counted once
Countlf (IsNull (Orders.OrderDate)) counts all values whose content is empty
combit List \& Label

\section*{Totaling With Sum()}

The "Sum()" function adds up the values of the first argument.
The (optional) second parameter specifies whether the values used for the calculation are to be deleted after outputting (default: True).
Example:
Sum (Order_Details.UnitPrice)

\section*{Obtaining User Input via a Dialog With AskString\$()}

You can use the "AskString\$()" function to obtain information from the user during the print process. A dialog appears when printing in which the user is required to enter the information that you need.
The first parameter contains the text for the request that is to appear in the dialog.
With the second parameter, you can specify whether the user request is only to appear once when printing starts (default: False), or whether the information is to be requested for each individual data record (True).
The third parameter contains the string that you want to display as a recommended value in the dialog's input field.
The last parameter specifies the maximum number of characters that the user may enter. Example:
AskString\$ ("Insert Subject",False,"Your request from " + Date\$(Now()))


Figure 10.12: Input dialog with AskString\$()

\section*{Appearance Conditions With Lastpage()}

The "Lastpage()" Boolean function returns whether the actual page is the last page, i.e. the result of this function is "True" or "False".
This function can only be used in footer lines of tables, as a condition in a layout region condition, or in objects linked to a table. In all other cases, Lastpage() is always "False".
combit List \& Label

\section*{Logical Conditions With Cond()}

The "Cond()" and "If()" functions let you formulate a wide variety of conditions. The first argument is a logical expression that is evaluated for truth. If the expression is true, the second argument is returned as the result. If the expression is false, the third argument is returned as the result.
A simple example: Let us assume that you want to output the total of the article prices on a page in an invoice footer line. You also want to output the grand total of article prices on the last page.


Figure 10.13: Example of the use of Cond()
You enter this function in the formula wizard either directly in the editing line or via the "Condition" tab.
1. As a "condition" ( \(1^{\text {st }}\) parameter), enter the expression that is to be tested for TRUE or FALSE. In our example, we want to use the "not Lastpage()" function to determine whether
this is the last page.
2. As the "expression, if condition is TRUE" ( \(2^{\text {nd }}\) parameter), enter the expression that is to apply if the above condition is TRUE. In our example, the condition is true if this is not the last page and in this case the page total is to be output (parameter of the "Sum()" function is "True").
3. As the "expression, if condition is FALSE" (3 \(3^{\text {rd }}\) parameter), enter the expression that is to apply if the above condition is FALSE. In our example, the condition is false if this is the last page and in this case the grand total is to be output (parameter of the "Sum()" function is "False").
4. If you have entered this function via the "Condition" tab, you can use the "Insert" button to add the finished condition to your editing line.
combit List \& Label

\section*{Working With Null Values}

There are various functions for working with Null values (undefined field contents).
If there are Null values in an expression, the entire expression can become Null. To prevent this, use the "NullSafe()" function for fields that could be empty (e.g.
salutation or title). This function checks to see whether the field value is Null and returns a substitute value if this is the case; otherwise, it returns the result of the expression.

\section*{Example:}

Cond (Empty(COMPANY),NulISafe(SALUTATION) + " " + FIRSTNAME+ " " + NAME)

With the "IsNull()" function, you can check whether the value passed to the function or the result of the expression is Null, i.e. an empty field.

Example:
Cond (Empty(COMPANY) or IsNull(COMPANY),FIRSTNAME + " " + NAME)

You can set a Null value with the "Null()" function.

ゅ
combit List \& Label

\section*{Working With Operators}

Open the list of available operators by clicking the "Operators" tab. The operators are used to join variables and free text to give more complex conditions and to perform compares or calculations.
On the "Operators" tab, you will find the respective operators in the column on the extreme left; the syntax is shown in the middle column and the value types with which the respective operator can be used in the right column.
Operators join two or more values or variables to give a new value. In this way, you can formulate arithmetic expressions (basic arithmetic operations) or logical expressions. The value type of the result of an expression depends on the value type of the individual elements of the expression. The "+" operator has a special meaning. It is not only used for additions ("Number" and "Date" (value types) but also for joining strings and fixed text ("string" value type).
You can combine multiple operations in one expression. The normal rules of precedence apply in this case: Logical operators are evaluated before arithmetic operators which are evaluated before relational operators. You must use brackets if you want a different processing order. The "innermost" brackets are evaluated first.
The general processing hierarchy is
\begin{tabular}{ll}
\hline Priority & Operator \\
\hline 1 & brackets () \\
\hline 2 & functions \\
\hline 3 & logical operators \\
\hline 4 & arithmetic operators \\
\hline 5 & relational operators \\
\hline
\end{tabular}


Figure 10.14: Example of multiplying two fields

See also:
- Arithmetic Operators
- Relational Operators
- Logical Operators

\section*{Arithmetic Operators}

The familiar rules of precedence multiplication and division before addition and subtraction \(\geqslant\) - apply for arithmetic operators. The "Modulo" operator is evaluated first followed by multiplication/division ("*" and "/") and finally addition and subtraction ("+" and "-"). Example: NET_PRICE+(NET_PRICE*0.19)
\begin{tabular}{lll}
\hline Operator & Meaning & Data types \\
\hline+ & addition & string, date, number \\
\hline- & subtraction & date, number \\
\hline\(*\) & multiplication & number \\
\hline\(/\) & division & number \\
\hline\(\%\) & \begin{tabular}{l} 
modulo (remainder \\
division)
\end{tabular} & with number \\
\hline
\end{tabular}

\section*{Relational Operators}

Relational operators consist of two values of the same data type which are compared with one another returning a true/false value. The result (return value) is the Boolean value true or false. Example: Page()<>1
\begin{tabular}{lll}
\hline Operator & Meaning & Data types \\
\hline\(>\) & greater & string, number, date \\
\hline\(>=\) & greater or equal & string, number, date \\
\hline\(<\) & less than & string, number, date \\
\hline\(<=\) & less than or equal & string, number, date \\
\hline\(=\) & equal & string, number, date \\
\hline\(==\) & equal & string, number, date \\
\hline\(<>\) & not equal & string, number, date \\
\hline\(!=\) & not equal & string, number, date \\
\hline
\end{tabular}

\section*{Logical Operators}

A logical operator is a function that returns a true/false value. The result (return value) is the Boolean value true or false.
Depending on the type of logical operator, the compound expression is true if both joined expressions are true (AND conjunction) or if at least one of the joined expressions is true (OR conjunction).
The rules of precedence are: Negations are evaluated first, then the "logical AND" and finally the "logical OR".
Example: Zip code >=70000 AND zip code \(<=80000\)
\begin{tabular}{lll}
\hline Operator & Meaning & Data type \\
\hline NOT or .NOT. & negation & Boolean \\
\hline AND or .AND. & logical AND & Boolean \\
\hline OR or .OR. & logical OR & Boolean \\
\hline XOR or .XOR. & logical exclusive OR & Boolean \\
\hline
\end{tabular}

ゅ
combit List \& Label

\section*{Overview of LL Variables and LL Fields}

List \& Label provides several variables and fields automatically depending on the application. You will find the variables and fields in the "LL" sub-folders in the variable list.

See also:
- Overview of Variables
- Overview of Fields

ゅ
combit List \& Label

\section*{Overview of Variables}
\begin{tabular}{|c|c|}
\hline Name & Explanation \\
\hline LL.Color.* & Color value of the corresponding color. \\
\hline LL.Device.Name & Identification of the output device (printer). \\
\hline LL.Device.Page.Name & Designation of the selected paper size (example "A4"). \\
\hline LL.Device.Page.Size.cx & \begin{tabular}{l}
Physical page width of the output device in the project \(\rangle\) unit of measure. Can be used in formulas to adjust the size of objects dynamically to fit larger output formats. \\
Example: In the property list, set the Position.Left property to the value 0 and the Position.Right property to LL.Device.Page.Size.cx. The object will now always occupy the total page width.
\end{tabular} \\
\hline LL.Device.Page.Size.cy & Physical page height in the projects unit of measure. \\
\hline LL.Device.PrintableArea.Offset.cx & \begin{tabular}{l}
Width of the unprintable left margin in the project s unit of measure. Example: In the property list, set the Position.Left property to the value LL.Device.PrintableArea.Offset.cx. \\
The object will now always be positioned precisely at the left margin of the printer's printable area.
\end{tabular} \\
\hline LL.Device.PrintableArea.Offset.cy & Height of the unprintable top margin in the project s unit of measure. \\
\hline LL.Device.PrintableArea.Size.cx & Printable page width in the project unit of measure. \\
\hline LL.Device.PrintableArea.Size.cy & Printable page height in the project s unit of measure. \\
\hline LL.CountData (not with multiple tables) & Number of data records transferred by the program. This number also includes the data records that were not printed because of their filter condition. This counter is incremented for each data record. \\
\hline
\end{tabular}
\begin{tabular}{ll}
\begin{tabular}{l} 
LL.CountDataThisPage \\
(not with multiple tables)
\end{tabular} & \begin{tabular}{l} 
Number of data records transferred \\
by the program on the current page. \\
This number also includes the data \\
records that were not printed \\
because of their filter condition. This \\
counter is incremented for each data \\
record.
\end{tabular} \\
\hline \begin{tabular}{ll} 
LL.CountPrintedData \\
(not with multiple tables)
\end{tabular} & Number of records actually printed. \\
\hline LL.CountPrintedDataThisPage & \begin{tabular}{l} 
Number of data records actually \\
printed on the current page.
\end{tabular} \\
\hline LL.CurrentContainerltem & \begin{tabular}{l} 
Value of an element's "Name" \\
property in the report container. Is \\
used e.g. for display and layout
\end{tabular} \\
region conditions.
\end{tabular}

ゅ
combit List \& Label

\section*{Overview of Fields}
\begin{tabular}{|c|c|}
\hline Name & Explanation \\
\hline LL.ChartObject.ArcIndex (only with pie charts) & Index of the pie segment. The pie segments are sorted by size. The largest pie segment has index 1. \\
\hline LL.ChartObject.ArcPerc (only with pie charts) & Size of the pie segment in percent. \\
\hline LL.ChartObject.ArcTotal (only with pie charts) & Absolute value of the entire data volume. \\
\hline LL.ChartObject.ArcTypelsOthers (only with pie charts) & True, if the current pie segment is the "other" segment. \\
\hline LL.ChartObject.ArcValue (only with pie charts) & Absolute value of the current pie segment. \\
\hline LL.ChartObject.AxisCoordinate (only with charts) & Coordinate value (can be used in axis labels). \\
\hline LL.CurrentRelation (only with multiple tables) & Description of the current relationship to the higher-level table. \\
\hline LL.CurrentSortOrder (only with multiple tables) & Description of the current sort order in the table. \\
\hline LL.CurrentTable (only with multiple tables) & Identification of the table currently in use. \\
\hline LL.CurrentTablePath (only with multiple tables) & \begin{tabular}{l}
Identification of the table currently in use (hierarchically with higher-level tables), e.g. \\
Customers.Orders.Order Details.
\end{tabular} \\
\hline LL.FCountData & Number of data records transferred by the program. This number also includes the data records that were not printed because of their filter condition. \\
\hline LL.FCountDataThisPage & Number of data records transferred by the program on the current page. This number also includes the data records that were not printed because of their filter condition. \\
\hline LL.FCountPrintedData & Number of records actually printed. \\
\hline LL.FCountPrintedDataThisPage & Number of data records actually printed on the current page. \\
\hline LL.Relations.* (only with multiple tables) & Available relationships. \\
\hline
\end{tabular}

LL.Tables.* (only with multiple tables)

ゅ
combit List \& Label

\section*{Overview of Functions}

All functions available in List \& Label are listed here alphabetically. In the formula wizard, you also have an additional list sorted by functional group.

\section*{Abs}

\section*{Purpose:}

Calculate the absolute value of a number. A negative value will be returned as positive and a positive value will remain unchanged.
Parameter: Number
Return value:
Number
Example:
Abs (-3) \(=3\)
\(\operatorname{Abs}(3.12)=3.12\)

\section*{AddDays}

\section*{Purpose:}

Adds the given number of days to the date, or subtracts the number of days when a negative value is entered.
Parameter:
Date
Number
Return value:
Date

\section*{AddHours}

\section*{Purpose:}

Adds the given number of hours to the date, or subtracts the number of hours when a negative value is entered.
Parameter:
Date
Number
Return value:
Date

\section*{AddMinutes}

\section*{Purpose:}

Adds the given number of minutes to the date, or subtracts the number of minutes when a negative value is entered.
Parameter:
Date
Number
Return value:
Date

\section*{AddMonths}

\section*{Purpose:}

Adds the given number of months to the date, or subtracts the number of months when a negative value is entered.
Parameter:
Date
Number
Return value: Date

\section*{AddSeconds}

\section*{Purpose:}

Adds the given number of seconds to the date, or subtracts the number of seconds when a negative value is entered.
Parameter:
Date
Number
Return value: Date

\section*{AddWeeks}

\section*{Purpose:}

Adds the given number of weeks to the date, or subtracts the number of weeks when a negative value is entered.
Parameter:
Date
Number
Return value:
Date

\section*{AddYears}

\section*{Purpose:}

Adds the given number of years to the date, or subtracts the number of years when a negative value is entered.
Parameter:
Date
Number
Return value:
Date

\section*{Alias\$}

\section*{Purpose:}

Returns the value that is specified for the key (first parameter) in the key/value-pairs (second parameter).
Parameter:

String
String List of values (Form: <key=value>|
[<key=value>]. To be able to use "|" or "=" in the value or key, place a " \(\backslash\) " infront of it.

String (optional) Default if the value cannot be found.
Return value:
String
Example:
Alias\$("USA", "DEU=Deutschland|USA=United States of America|GB=United Kingdom") Result: United States of America

\section*{ArcCos}

\section*{Purpose:}

Calculates the arccosine of the value.

Parameter:
Number Value
Number (optional) Mode ( \(0=\) Degree, \(1=\) Radian). Default: 0.
Return value:
Number
Example:
ArcCos (0)
Result: 90

\section*{ArcSin}

\section*{Purpose:}

Calculates the arcsine of the value.
Parameter:
Number Value
Number (optional) Mode (0=Degree, 1=Radian). Default: 0.

\section*{Return value:}

Number
Example:
ArcSin (0.5) Result: 30,00

\section*{ArcTan}

\section*{Purpose:}

Calculates the arccotangent of the value.

\section*{Parameter:}

Number Value
Number (optional) Mode (0=Degree, 1=Radian). Default: 0.

\section*{Return value:}

Number
Example:
ArcTan (1) Result: 45,00

\section*{Asc}

\section*{Purpose:}

Returns the ASCII-Code of the first character of the string. Parameter: String Return value: Number
Example: Asc("A") Result: 65

\section*{AskString\$}

\section*{Purpose:}

With this function, information can be requested from the user during printing. A typical example of use for this function would be in a project for a bank transfer form. Information that remains constant, such as name and bank details of the sender, can be integrated directly into the project as fixed text or variables. The transfer amount, however, will almost always be different. With the function AskString \(\$(\) ), this information can be requested from the user during printing.
At print time, a dialog will appear in which the needed information can be entered.
The dialog allows the entered value to be carried over. Abort with "Cancel".
With the button "All", the entered value will be automatically used for all future result for the AskString\$ function during thus print job. This is useful when the value remains constant over all records.

\section*{Parameter:}

The first parameter contains some descriptive text that will appear in the dialog. Since this is a formula, fixed text must be entered in quotation marks, for example "Transfer amount: This first parameter must be entered, all remaining parameters are optional. If no other parameter(s) is/are entered, the first string is also the default setting for the user input.
Boolean (optional) The second parameter allows you to define whether the dialog should be shown once prior to printing (default, FALSE), or if the dialog should be shown for each record (TRUE).

\section*{String}
(optional) The third parameter contains the string that appears as the recommended value for the user input. Since this is a formula, fixed text must be entered in quotation marks,

Number \(\begin{array}{r}\text { for example "50.00 USD". } \\ \begin{array}{r}\text { (optional) The last parameter defines the number of } \\ \text { characters that can be entered by the user. A value of } 16 \text {, for } \\ \text { example, allows the user to enter a maximum of } 16 \text { characters. }\end{array}\end{array}\)
Return value:
String
Example:
AskString\$("Transfer amount",True,"50.00 USD",16) Opens a dialog with the title "Transfer amount \(\rangle\), a recommended value of "50.00 USD \(\geqslant\) and a maximum of 16 characters. Since the second parameter is TRUE, the dialog will be shown for each record to be printed.

\section*{AskStringChoice\$}

\section*{Purpose:}

Prompts the user to choose a value for the specified variable from a combobox at print time.
Parameter:
String Text, which is displayed and should specify what is to be entered.
Boolean (optional) Sets whether the dialog should be shown once prior to printing (default, FALSE), or if the dialog should be shown for each record (TRUE).
String
(optional) The combobox entries. The single entries of the combo box are separated by "|". If one of the entries is '***' (three asterisks), the text is editable. So a new value which may be different from the list items can be entered.

\section*{Number (optional) Maximum length (Default: 8192 characters). Return value: String Example: \\ AskStringChoice\$ ("Document \\ type".F.,"Offer|Invoice|Delivery note|***)}

\section*{ATrim\$}

\section*{Purpose:}

Removes spaces from the beginning and end of a string. Parameter:

\section*{String}

Return value:
String
Example:
Atrim\$(" combit GmbH ") Result: combit GmbH

\section*{Purpose:}

Generates the mean of the set of values that is produced by the first argument.
Parameter:
Number
Boolean
Expression of the value to be calculated.
(optional) TRUE: After the output, the values which were stored for the calculation are deleted. (default: TRUE). Please note that the stored calculation values are generally deleted for every (sub)table end. The second parameter only decides whether the values are already deleted within the table.

\section*{Return value:}

Number
Example:
Avg(Order_Details.Quantity*Order_Details.UnitPrice)

\section*{Barcode}

\section*{Purpose:}

This function converts a string to a barcode.

Parameter:
String
String

Barcode value (contents)
Barcode type. The possible barcode types will be listed by the auto-complete function of the wizard. If the barcode cannot be correctly interpreted it will not be printed. Some barcodes require special formats that must be used. Further information can be found in Chapter Supported Barcode.

Return value:
Barcode

\section*{Example:}

Barcode(Upper\$(Name),"3of9")

\section*{Barcode\$}

\section*{Purpose:}

Returns the text contents of a barcode.
Parameter:
Barcode
Return value:
String
Example:
Barcode\$(BC_3OF9) Result: "Item 4711"

\section*{BarcodeType\$}

\section*{Purpose:}

Returns the type of the barcode as a string.
Parameter:
Barcode
Return value:
String

\section*{BasedStr\$}

\section*{Purpose:}

Returns the value to any radix.

Parameter:
Number Value.
Number Radix (2 to 36).
Number (optional) Minimum length of the string (without optional prefix). 0 for the minimal length (Default).
Boolean (optional) Defines if a prefix ('Ob' for radix 2, '0o' for radix 8, '0x' for radix 16) is inserted before the string (Default: False).
Return value:
String
Example:
BasedStr\$ (1,2,1,True) Result: 0b1

\section*{BinaryAND}

\section*{Purpose:}

Links the two (integer) parameters binary with 'and' and returns the result.
Parameter:
Number Value.
Number Value.
Return value: String
Example:
BinaryAND \((01,10) \quad\) Result: 0
BinaryAND \((10,11)\)
Result: 10

\section*{BinaryNOT}

\section*{Purpose:}

Negates the value binary and returns the result.
Parameter:
Number Value.
Return value:
String
Example:
BinaryNOT (10) Result: 5
(ten equivalent 1010, five equivalent 0101)

\section*{BinaryOR}

\section*{Purpose:}

Links the two (integer) parameters binary with 'or' and returns the result.
Parameter:
Number Value.
Number Value.
Return value:
String
Example:
BinaryOR \((01,10)\)
BinaryOR \((10,11)\)
Result: 11
Result: 11

\section*{BinarySHL}

\section*{Purpose:}

Shifts the value binary to the left.
Parameter:
Number Value.
Number Number of bits, the value is shifted.
Return value:
String
Example:
BinarySHL (1,1) Result: 2

\section*{BinarySHR}

\section*{Purpose:}

Shifts the value binary to the right.
Parameter:
Number Value.
Number Number of bits, the value is shifted.
Return value:
String
Example:
BinarySHR \((2,1) \quad\) Result: 1,00

\section*{BinaryXOR}

\section*{Purpose:}

Links the two (integer) parameters binary with 'exclusive or' and returns the result.
Parameter:
Number Value.
Number Value.
Return value:
String
Example:
BinaryXOR \((01,10)\)
Result: 11
BinaryXOR \((10,11)\)
Result: 1

\section*{BMPMapToGray}

\section*{Purpose:}

Converts the picture to greyscales.
Parameter:
Picture or String
Return value:
Picture
Example:
BMPMapToGray ("sunshine.gif")

\section*{BMPRotate}

\section*{Purpose:}

Rotates a picture by the given degree.
Parameter:
Picture or String
Number
Number
Rotation angle (optional) Mode ( \(0=\) Degree, \(1=\) Radian)
Return value:
Picture
Example:
BMPRotate(Article.Picture,90)

\section*{Capitalize\$}

\section*{Purpose:}

Returns a string in which the first letter of the individual words is a capital letter and the rest are small letters.
Parameter(s):
String
Return value:
String
Example:
Capitalize\$ (Product.Category) Result: Tea, Coffee, And Soft Drinks

\section*{Case\$}

\section*{Purpose:}

Converts a number, dependant upon the value, into a string. Assignment is made with a formatting string that contains the replacement string for the number values in ascending order.

\section*{Parameter:}

Number Number to be converted (n). The n-th value of the string will be copied to the return value string. If enough values do not exist, the string will remain empty.
String Collection of strings separated by a particular character. If a third parameter does not exist, this is the "|" character, otherwise the first character of this parameter.
String (optional) Separator for the formatting string (default: "|")

\section*{Return value:}

\section*{String}

\section*{Example:}

Case\$(Page(),"0|I|II|II||IV|V|VI|VII|VIII|IX|X")
Result: "III", if Page() = 3

\section*{Ceil}

\section*{Purpose:}

Calculates the next bigger integer based on the given value. See also function Floor().

\section*{Parameter:}

Number Value.
Return value:
Number
Example:
Ceil(5.6)
Result: 6

\section*{Century}

\section*{Purpose:}

Returns the century of the date.
Parameter:
Date
Boolean (optional) sets whether the calculation should be carried out 'simply' (century starts with year 0) or 'historically' (century starts with year 1). Default: False
Return value:
Number
Example:
Str\$(Century(Today()),0,0)
Str\$(Century(Date("01.01.2000")),0,0)
Result: 21

20
Str\$(Century(Date("01.01.2001")),0,0)
Result:
21
Str\$(Century(Date("01.01.2000"),.T.),0,0) Result: 21

\section*{CheckMod10}

\section*{Purpose:}

Calculate the modulo 10 checksum digit of the string. Parameter:

String A string of digits.
Return value:
Number

\section*{Chr\$}

\section*{Purpose:}

Converts a number to a character. This character has the entered number as its ASCII-Code. For multibyte character sets, the highword is the lead byte, for Unicode, the value is the Unicode code point.
Parameter:
Number
Number (optional) Defines the type of the parameter. \(0=\) multibyte character sets, 1=Unicode. Default is Unicode.

\section*{Return value:}

\section*{String}

Example:
Chr\$(64) Result: "@"

\section*{ChrSubst\$}

\section*{Purpose:}

Searches a string for a string that is contained in the second parameter. Every occurrence of this string will be replaced by the string defined in the third parameter. If no third parameter exists, the strings will be removed.
Parameter:
String
String
String
(optional)
Return value:
String
Example:
ChrSubst\$("Otto", "Oo", "_") Result: "_tt_" ChrSubst\$("Normalconsū̄er", "aeiou","??"') Result: "N?? rm??lc??ns??m??r"
ChrSubst\$("Normalconsumer", "aeiou") Result: "Nrmlcnsmr"
ChrSubst\$("3.1415926535",".",",") Result:
"3,1415926535"

\section*{Cond}

\section*{Purpose:}

Allows to define conditions. The first parameter is a logical expression that will be evaluated as "True of "False . If the first expression is "True , the second expression will be returned as the result. If the expression is "False , the third expression will be returned as the result. If no third expression is entered, the return value will assume the following standard values, dependent upon its type:
\begin{tabular}{|l|l|}
\hline 2. Argument Type & \begin{tabular}{l} 
Return value if expression \\
= False
\end{tabular} \\
\hline Boolean & False \\
\hline String & "" (empty String) \\
\hline Date & Julian Date value 0 \\
\hline Number & 0 \\
\hline Picture & "" (empty String) \\
\hline Barcode & "" (empty String) \\
\hline
\end{tabular}

\section*{Parameter:}

\section*{Boolean}

All
All
(optional) The third parameter must be the same type as the second parameter.

\section*{Return value:}

\section*{All}

\section*{Example:}

Cond(COUNTRY<>"USA",COUNTRY_LONG)
Cond(PRICE=0,"on request",Str\$(PRICE,0,2))
Cond(empty(COMPANY),SALUTATION,COMPANY)

\section*{Constant.Pi}

\section*{Purpose:}

Returns the value of Pi.
Parameter: -
Return value:
Number
Example:
Constant. Pi()\(\quad\) Result: 3,14159 (depending on the number of decimals)

\section*{Contains}

\section*{Purpose:}

Evaluates if a string contains another string (second parameter).
Parameter:
String
String
Return value:
Boolean
Example:
Contains("Itemnumber: 12345", "1234") Result: True
combit List \& Label

\section*{Continued}

\section*{Purpose:}

Indicates that a text or RTF-object had a page break. This means that the current page is a result of the page break.
Parameter:

\section*{Return value:} Boolean

\section*{Cos}

\section*{Purpose:}

Calculates the cosine of the value.

Parameter:
Number Value
Number (optional) Mode (0=Degree, 1=Radian). Default: 0.
Return value:
Number
Example:
Cos (90) Result: 0

\section*{Count}

\section*{Purpose:}

Counts the number of values of the first argument. With this function, all Null values in the argument are included in the count. Use the CountIf() function when you want to disregard Null values.
Hint: Sum variables (see Working With Sum Variables) are an alternative way of creating counters. Sum variables are principally applicable to whole tables. Aggregate functions principally table specific.

\section*{Parameter:}

AII
Values to count (sets the value to count). Needed to define the table (subtable) for which the records shall be counted.

\section*{Boolean (optional) TRUE: The values which were stored for the} calculation are deleted after output. (default: TRUE). Please note that the stored calculation values are generally deleted for every (sub)table end. The second parameter only decides whether the values will be already deleted within the table.

\section*{Return value:}

Number
Example:
Count(Order_Details.ProductID)
NthLargest(Ārticle.Price,Count(Distinct(Artikel.Stkpreis), Tru True)
Calculates the second-smallest value, only taking repeated values into account once.

\section*{Countlf}

\section*{Purpose:}

Counts the number of values that comply with the condition. Use the function Distinct() when repeated values are only to be counted once.

\section*{Parameter:}

Boolean Expression for the comparison.
Boolean (optional) TRUE: The values which were stored for the calculation are deleted after output. (default: TRUE). Please note that the stored calculation values are generally deleted for every (sub)table end. The second parameter only decides whether the values will be already deleted within the table.
Return value:
Number
Example:
CountIf(Customers.Region="EMEA")
Countlf(Distinct(Customers.Region="EMEA")) counts repeated values once only.
Countlf (IsNull (Orders.OrderDate)) counts all values with empty fields.

\section*{Crosstab.Cells.Avg}

\section*{Purpose:}

Returns the average of the cell contents. Only available in crosstab objects.
Parameter:
Boolean (optional) True: Only defined values are entered into the calculation (Default: False). Defined values: if you are, for example, analyzing customers and quarters, the quarters without turnover constitute an undefined value and can be treated separately in the calculation.
Number (optional) Row layer ( \(0=\) bottom layer or innermost group, \(1=\) next lowest, ...). Default: 0.

Number
(optional) Column layer ( \(0=\) bottom layer or innermost group, \(1=\) next lowest, ...). Default: 0.

\section*{Return value:}

Number

\section*{Crosstab.Cells.Max}

\section*{Purpose:}

Returns the largest value of the cell contents. Only available in crosstab objects. For the parameters and their meaning, see function Crosstab.Cells.Avg().

\section*{Crosstab.Cells.Min}

\section*{Purpose:}

Returns the smallest value of the cell contents. Only available in crosstab objects. For the parameters and their meaning, see function Crosstab.Cells.Avg().

\section*{Crosstab.Col\$}

\section*{Purpose:}

Returns the column header for the cell currently being output. Only available in crosstab objects.

Number (optional) Column layer ( \(0=\) lowest layer or innermost group, \(1=\) next lowest, ...). Default: 0.

\section*{Return value:}

String

\section*{Crosstab.Col}

\section*{Purpose:}

Returns the column index for the cell currently being output. Only available in crosstab objects.
Parameter:
Boolean (optional) True: Iayer (only cells in this layer count), Default: False.
Return value:
Number

\section*{Crosstab.Row\$}

\section*{Purpose:}

Returns the row header for the cell currently being output. Only available in crosstab objects.
Parameter:
Number (optional) Row layer ( \(0=\) lowest layer or innermost group, \(1=\) next lowest, ...). Default: 0.

Return value:
String

\section*{Crosstab.Row}

\section*{Purpose:}

Returns the row index for the cell currently being output. Only available in crosstab objects.
Parameter:
Boolean (optional) True: layer (only cells in this layer count), Default: False.

\section*{Return value:}

Number
combit List \& Label

\section*{Crosstab.Value}

\section*{Purpose:}

Returns the cell's content.
Parameter: Return value:

Number

\section*{CStr\$}

\section*{Purpose:}

Formats a number according to a format string. This is identical to the formatting information for the function printf() in the language \(C\). The first parameter is a number of double precision, and the conversion operator can assume i.e. the following values: 'f', 'g', 'G', 'e', 'E'.
Parameter:
Number
String format string in C-notation, i.e. '\%<format>f'.
Return value:
String
Example:
CStr(Pi,"\%5.1f") Result: " 3.1"
CStr(100*Pi,"num: \%g") Result: "num:
3.141593e+02"

\section*{Date}

\section*{Purpose:}

Converts a string to a date.
- If the string Contains a dot ".", it will be read in the "d.m.y" format (German).
- If the string contains a diagonal slash "/", it will be read in the "m/d/y" format (US English).
- If the string contains a dash "-", if will be read in the "y-m-d" format (ANSI).
- If the input cannot be correctly interpreted, then the date represents a value that is larger than all other values, (1e100). The return value can be evaluated for correctness using "JulianToDate(1e100)".
- When one or two digits represent the year, all values under 30 will be applied to the \(21^{\text {st }}\) century ( \(20 x x\) ) and all values over 30 will be applied to the \(20^{\text {th }}\) century (19xx).

\section*{Parameter:}

\section*{String}

Return value:
Date

\section*{Example:}

Date("17.10.2007")
Date("10/17/2007")
Date("2007-10-17")

\section*{Date\$}

\section*{Purpose:}

Converts a date, using a format string, into an appropriately formatted string.
Composition of the format string: this is a normal string into which placeholders can be embedded.
\begin{tabular}{|c|c|}
\hline Place holder & Description \\
\hline \%d & Day (1..31) \\
\hline \(\%<n>d\) & Day to <n> digits \\
\hline \% \(0<n>d\) & Day to <n> digits, filled on left with '0's \\
\hline \%w & Weekday (1..7) \\
\hline \%<n>w & Weekday to <n> digits \\
\hline \%0<n>w & Weekday to <n> digits, filled on left with '0's \\
\hline \%m & Month (1..12) \\
\hline \%<n>m & Month to <n> digits \\
\hline \%0<n>m & Month to <n> digits, filled on left with '0's \\
\hline \%y & Year \\
\hline \% \(<n>y\) & Year, to <n> digits \\
\hline \% \(0<n>y\) & Year, to <n> digits, filled on left with '0's \\
\hline \%q & Quarter \\
\hline \%D & Weekday, written out \\
\hline \%M & Month, written out \\
\hline \[
\begin{array}{ll}
\text { "\%e", } \\
\text { <n>e" }
\end{array}
\] & Year in the local calendar (Japan: Emperor's year) \\
\hline \[
\begin{array}{ll}
\hline \% \mathrm{Fg} \text { ", } \\
\text { <n>g" }
\end{array}
\] & Era of the local calendar (Japan: Emperor sera) \\
\hline "\%g", "\%1g" & Single letter, Latin letters \\
\hline "\%gg", "\%2g" & Single letter, localized \\
\hline \[
\begin{aligned}
& \text { "\%ggg", } \\
& \text { "\%3g" } \\
& \hline
\end{aligned}
\] & Long name, localized \\
\hline
\end{tabular}
\begin{tabular}{ll}
\begin{tabular}{l} 
"\%gggg", \\
"\%4g"
\end{tabular} & Long name, Latin letters \\
\hline "\%x" & Localized date, short form \\
\hline\(\% \mathrm{H}\) & Hours in 24h-format \\
\hline\(\% \mathrm{~h}\) & Hours in 12h-format \\
\hline\(\%<\mathrm{n}>\mathrm{h}\) & Hours in \(<\mathrm{n}>\) digits \\
\hline\(\% \mathrm{i}\) & Minutes \\
\hline\(\%<\mathrm{n}>\mathrm{i}\) & Minutes in \(<\mathrm{n}>\) digits \\
\hline\(\% \mathrm{~s}\) & Seconds \\
\hline\(\%<\mathrm{n}>\mathrm{s}\) & Seconds in \(<\mathrm{n}>\) digits \\
\hline\(\% \mathrm{PM}\) & AM or PM
\end{tabular}

As long as one of the above formats is used, the optional third parameter can be used to set the locale. If the second parameter contains a valid ISO 3166-Country code, the third parameter can be used to set either the short " 0 or long " \(1>\) format.

\section*{Parameter:}

Date
String
String

Value to be formatted.
(optional) Format description or ISO 3166-Country code.
(optional) ISO 3166-Country code or date format.

\section*{String}

\section*{Example:}

Date\$(Today(),"Date: \%D, \%d/\%m/\%y") Result: "Date:
Thursday, 8/11/2007"
Date\$(Today(),"\%2wthWeek; \%D, \%2d/\%2m/\%4y")
Result: "45th Week, Thursday, 8/11/2007"
Date\$(Today(),"\%D, \%3d/\%02m/\%4y") Result:
"Thursday, 8/11/2007"
Date\$ (Now(),"\%02h:\%02i:\%02s \%PM") Result:
"04:03:50 PM"
combit List \& Label

\section*{DateDiff}

\section*{Purpose:}

Returns the difference between two dates in days.
Parameter:

Date
Date

First date value
Second date value

Return value:
Number
Example:
DateDiff(Date("01.01.2009"),Date("01.03.2009"))
Result: 59
combit List \& Label

\section*{DateDiff\$}

\section*{Purpose:}

Returns the difference between two dates in days as string.
Parameter:

Date
Date
String
Return value:
String
Example:
DateDiff\$(Date("01/01/2009"),Date("03/01/2009"))
Result: 2 Months

\section*{DateHMS}

\section*{Purpose:}

Converts three numbers for hour, minute and second into a date.
Parameter:

Number
Number
Number
Return value:
Date

Hour
Minute
Second
combit List \& Label

\section*{DateInLeapYear}

\section*{Purpose:}

Checks if the given date is in leap year or not. The calculation is made according to the proleptic gregorian calendar.
Parameter:
Date
Return value:
Boolean
Example:
DateInLeapYear("01.01.2012") Result: True

\section*{DatelnRange}

\section*{Purpose:}

Evaluates if the date falls within the entered time interval:
Minimum Date: JulianToDate(0)
Maximum Date: JulianToDate(1e100)
Parameter:
Date
Date
Date to be evaluated.
Date
Lower limit of the test interval.
Upper limit of the test interval.
Return value:
Boolean
Example:
DateInRange(Date("2007.10.20"),Date("2007.2.29"),Today Result: True

\section*{DateToJulian}

\section*{Purpose:}

Calculates the Julian value of a date. Each day (even those in the past) are assigned a unique number.
Parameter:
Date
Return value:
Number
Example:
DateToJulian(Today()) Result: 2453992

\section*{DateYMD}

\section*{Purpose:}

Converts three numbers for day, month and year into a date.
Parameter:
Number Year
Number Month
Number Day
Return value:
Date
Example:
DateYMD(2009, 11, 1) Result: 01.11.2009

\section*{Day}

\section*{Purpose:}

Determines the day (1...31) of the month and returns it as a number.
Parameter: Date
Return value:
Number
Example:
Day(Date("17.10.2009")) Result: 17

\section*{Day\$}

\section*{Purpose:}

Determines the day (1 31) of the month of a date and returns it as a string.
Parameter:
Date
Return value:
String
Example:
Day\$(Date("17.10.2009")) Result: "17"

\section*{Decade}

\section*{Purpose:}

Returns the decade of the date. Hint: The value is always relative to the start of the century (1..10)!
Parameter:
Date
Boolean (optional) sets whether the calculation should be carried out 'simply' (decade starts with year 0) or 'historically' (decade starts with year 1). Default: False

\section*{Return value:}

Number
Example:
Str\$(Decade(Date("01.01.2009")),0,0)
Result: 1
Str\$(Decade(Date("01.01.2000")),0,0)
Result:
10
Str\$(Decade(Date("01.01.2000"),.T.),0,0) Result: 1

\section*{Distinct}

\section*{Purpose:}

Affects the higher order aggregate function (e.g. Sum(), Avg(), Count()...) and causes equal values only to be used once in the calculation.
Parameter:
All
Return value:
All
Example:
Countlf(Distinct(Customers.Region="EMEA"))

\section*{Dow}

\section*{Purpose:}

Returns the day of the week to a number(1...7), 1=Sunday, \(2=\) Monday, ...
Parameter:
Date
Return value:
Number
Example:
Dow(Date("04.07.1776")) Result: 4 (Thursday).

\section*{Dow\$}

\section*{Purpose:}

Returns the day of the week as a string in accordance with the country settings, "Sunday", "Monday", ...
Parameter:
Date
Return value:
String
Example:
Dow\$(Date("04.07.1776")) Result: "Thursday"
combit List \& Label

\section*{Drawing}

\section*{Purpose:}

Converts a string type file path into a picture.
Parameter:
String
Return value:
Drawing
combit List \& Label

\section*{Drawing\$}

\section*{Purpose:}

Converts a picture into a string type file path.
Parameter:
Picture
Return value:
String

\section*{DrawingHeightSCM}

\section*{Purpose:}

Returns the height of the Picture in SCM units (1/1000 mm ).
Parameter:
Picture
Return value:
Number

\section*{DrawingWidthSCM}

\section*{Purpose:}

Returns the width of the Picture in SCM units (1/1000 mm ).
Parameter:
Picture
Return value:
Number

\section*{Empty}

\section*{Purpose:}

Evaluates if a string is empty. If it is empty, "True will be the return value, otherwise "False . Useful, for example, to determine if the field "ADDRESS is empty, and if it is, in combination with the IF-THEN-ELSE condition cond(), either print the contents of the field "ADDRESS or "POBOX
The third parameter allows the removal of leading and trailing spaces. If this is evaluated as "True, a string consisting only of multiple spaces will be recognized as empty.
Parameter:
String
Boolean (optional)
Return value:
Boolean
Example:
Empty("xyz") Result: False
Empty("")
Result: True

\section*{EndsWith}

\section*{Purpose:}

Checks whether the string in the first argument ends with the string in the second argument.
Parameter(s):
String
String
Boolean Specifies whether capitals/small letters are relevant. Default value: False
Return value:
Boolean
Examples:
EndsWith ("Hallo World","rld") Result: True
EndsWith ("Hallo World","llo") Result: False

\section*{Evaluate}

\section*{Purpose:}

Evaluates the expression passed as parameter.
Parameter:
String
Return value:
All
Example:
Str\$(Evaluate("3*4"),0,0) Result: 12
Str\$(Evaluate("4-3"),0,0) Result: 1

\section*{Even}

\section*{Purpose:}

Evaluates if a number is even. If the number is even, "True \(\uparrow\) will be returned, otherwise "False

\section*{Parameter:}

Number
Return value:

\section*{Boolean}

Example:
"Page number "+Cond(Even(Page()),"even","odd")

\section*{Exists}

\section*{Purpose:}

Checks if a variable or field is defined. Is often used in connection with GetValue() and Evaluate().

\section*{Parameter:}

String
Return value:
Boolean
Example:
Exists("CustomerID") Result: False
If(Exists("Customer.Status"),Evaluate("Customer.Status"),"। customer status")
combit List \& Label

\section*{Exp}

\section*{Purpose:}

Calculates the exponential ( \(e^{x}\) ).
Parameter:
Number
Return value:
Number
Example:
Exp(3) Result: 20.08553692

\section*{Exp10}

\section*{Purpose:}

Calculates 10 raised to the power of number (10ㅈ).
Parameter:
Number
Return value:
Number
Example:
Exp10(3) Result: 1000

\section*{FirstHeaderThisTable}

\section*{Purpose:}

Returns whether the header of the table is being output for the first time. The function can be used as an appearance condition for the header to prevent it being printed more than once if the table continues onto the next page due to space limitations. The header is then only printed at the beginning of the table.
Parameter:
-
Return value: Boolean

\section*{Floor}

\section*{Purpose:}

Calculates the next smaller integer based on the given value. See also function Ceil().
Parameter:
Number
Return value:
Number
Example:
Floor(5.6) Result: 5

\section*{Frac}

\section*{Purpose:}

Calculates the fractional part of a number Parameter: Number
Return value:
Number
Example:
Frac(Pi) Result: 0.1415926535

\section*{FStr\$}

\section*{Purpose:}

Formats a number according to the format string.
These consist of the following characters ("if negative" refers to the value to be formatted):
\begin{tabular}{ll}
\hline\(*\) & Digit or '*'-Prefix \\
\hline\(\$\) & Local currency symbol \\
\hline- & Digit or sign, if negative \\
\hline+ & Digit or sign \\
\hline\((\) & Digit or '('-Prefix if negative \\
\hline\()\) & ')'-Postfix if negative \\
\hline\(\#\) & Digit or space prefix \\
\hline\(\&\) & Digit or '0' \\
\hline. & Decimal point \\
\hline & Comma, or space prefix \\
\hline
\end{tabular}
\(\overline{\text { A prefix is a sign that precedes a value, }}\) when needed. The expression \(\operatorname{FStr\$ (1,~"***")~results~in~"**1".~The~value~}\) " 1 " is preceded by the characters "**".
A Postfix is a character that, when needed, is placed after a number.
These formatting characters can be combined as needed. If the number is too large for the desired format, a "* string will be returned.
With the third (optional) parameter, additional formatting can be accomplished.
\begin{tabular}{ll}
\hline Value & Description \\
\hline 1 & \begin{tabular}{l} 
Removal of leading spaces. \\
The use is similar to the functions RTrim\$() \\
and LTrim\$().
\end{tabular} \\
\hline 2 & Empty string if value Null. \\
\hline 3 & \begin{tabular}{l} 
Removal of leading spaces and empty \\
strings when value is 0
\end{tabular} \\
\hline
\end{tabular}
Parameter:
NumberString Format stringNumberReturn value:
String
Example:
FStr(3.142, "\#") Result: "3"
FStr(5003.1,"\#,\#\#\#.\&\&") Result: "5.003,10"
FStr\$(3.142,"\#.\#\#\#") Result: "3,142"
FStr\$(3.142,".\#\#\#\#\#\#") Result:
FStr\$(3.142,"(\#.\#\#\#)") Result: " 3,142 "
FStr\$(-3.142,"(\#.\#\#\#)") ..... Result: "(3,142)"
FStr\$(3.142,"+\#.\#\#\#") Result: "+3,142"
FStr\$(3.142,"-\#.\#\#\#") ..... Result: " 3,142"
FStr\$(-3.142,"-\#.\#\#\#") ..... Result: "-3,142"
FStr\$(3.142,"\&\&\&.\&\&\&") Result: "003,142"FStr\$(3.142,"***.***") Result: "**3,142"
FStr\$(3.142,"\$\$\$.\$\$\$") Result: "\$\$3,142"
FStr\$(3.142,"\#\#\#.***") Result: " 3,142"
FStr\$(5003.1,"\#,\#\#\#.\&\&") ..... Result: "5.003,10"
FStr\$(3.142,"\#\#\#\#\#") Result: " 3"

\section*{GeometricAvg}

\section*{Purpose:}

Calculates the geometric average of the set of values that result from the first parameter / formula.
Parameter:
Number Expression for the value to be averaged.
Boolean (optional) True: The values which were stored for the calculation are deleted after output. (default: True). Please note that the stored calculation values are generally deleted for every (sub)table end. The second parameter only decides whether the values are already deleted within the table.
Return value:
Number

\section*{GetValue}

\section*{Purpose:}

Returns the value of a variable or field. Is often used in connection with Exists().
Parameter:
String
Return value:
All
Example:
Str\$(GetValue("Customers.CustomerID"),0,0) Result:
1234
If(Exists("Customer.Status"),Evaluate("Customer.Status"),"। customer status")

\section*{GetVar}

\section*{Purpose:}

Fetches a value that was set with the SetVar() function from the variable repository. The purpose of these functions is to provide a simple buffer for values. You shouldn't execute complex nesting with GetVar/SetVar or combine both functions with each other - especially for header, footer and group lines unexpected effects can occur here.
Parameter(s):
All
Return value:
All
Example:
GetVar ("Page")
result: contents of
SetVar

\section*{Hour}

\section*{Purpose:}

Determines the hour of the date and returns it in number format. If the parameter is not used, the hour of the print time will be returned.

\section*{Parameter:}

Date (optional)
Return value:
Number
Example:
A condition can evaluate if the current hour has the value "10 . The value of the current hour must be determined and then compared to the value " 10 .
Hour( \()=10\)

\section*{Purpose:}

Calculates a color value in the HSL color space (Hue, Saturation, Lightness)
Parameter:

Number
Number
Number
Return value:
Number

Hue \([0-360]\left(0^{\circ}=\right.\) red, \(120^{\circ}=\) green, \(240^{\circ}=\) blue \()\)
Saturation [0-1]
Lightness [0-1] (0=no lightness, \(1=\) full lightness)

\section*{Hyperlink\$}

\section*{Purpose:}

The function Hyperlink\$ creates a hyperlink text that can be inserted using an export module. When available, the hyperlink will only be embedded if the third parameter returns a result of "True \(\rangle\) (=default).
To optionally use a hyperlink only on a HTML page, you need to use the List \& Label variable LL. OutputDevice: Hyperlink\$("combit","http://www.combit.net",LL.OutputDev If an object text contains the string:
<!--begin:hyperlink="target"-->"Display text"<!--end:hyperlink-->
then a hyperlink will be automatically created in the exported HTML page. The hyperlink function automatically creates a string with the correct syntax.

\section*{Parameter:}

String
String
Boolean
Return value:
String
Example:
Hyperlink\$("combit","http://www.combit.net")
combit List \& Label

If
see Cond

\section*{Issuelndex}

\section*{Purpose:}

Returns the Issue Index (1..) for display and layout region conditions, if multiple issues are selected in the project parameters
Parameter:
Number
Return value:
Number

\section*{Int}

\section*{Purpose:}

Calculates the integer value of a number. The value will be truncated.
Parameter:
Number
Return value:
Number
Example
Int \((3,1)\) Result: 3

\section*{IsNull}

\section*{Purpose:}

Checks whether the transferred value or the result of the expression is Null, e.g. an empty date field.
Parameter: All

\section*{Return value:} Boolean

\section*{IsNulIOrEmpty}

\section*{Purpose:}

Checks whether a string is empty or Null.
Parameter(s):
String
Boolean (optional) If the value is True, then spaces at the beginning and end of the string are removed. Default value: False
Return value:
String
Example:
IsNulIOrEmpty ( \(\rangle\), True)
Result: True

\section*{Join\$}

\section*{Purpose:}

Collection of strings separated by a particular character. Parameter:

String
Collection of strings separated by a particular character.
String (optional) Separator for the formatting string (default: ";")
Number (optional) Maximum number of values ('...' will be appended). Default: all values.
Boolean (optional) True: The values which were stored are deleted after output. (Default: True). Please note that the stored values are generally deleted for every (sub)table end. The second parameter only decides whether the values will be already deleted within the table.

\section*{Return value: \\ String}

\section*{JulianToDate}

\section*{Purpose:}

Interprets a number as a Julian date (each day is assigned a unique number) and returns the appropriate date.
Parameter:
Number
Return value:
Date
Example:
JulianToDate(2454347) Result: 09/04/2007

\section*{LangCase\$}

\section*{Purpose:}

Returns one of the substrings according to the language that is set.
Parameter(s):
String
String that is returned if a localization string is not found. The localization string must be a valid ISO 639 language code.
String String with translation substrings, separated by "I" (or by an optional third argument). Translation texts must be formatted as follows "ISO 639 language code = translation text|[ISO 639 language code \(=\) translation text \(\geqslant\) ]. If the characters "|" or "=" are to be used in the value or the key, they must be preceded by "\", e.g. "USA=He\=llo".
String
(optional) separator (default: "|")
Return value:
String
Example:
LangCase\$("Hallo","USA=Hello|ESP=Hóla")
result: " Hóla " (with Spanish systems)

\section*{LastFooterThisTable}

\section*{Purpose:}

Returns whether the footer of the current table is being output for the last time. This function can be used as an appearance condition for the footer, in order to prevent the footer being printed if the table is continued on the next page due to space limitations. The footer is then only printed on the last page of the table.
Parameter:
-
Return value:
Boolean

\section*{Lastpage}

\section*{Purpose:}

Returns if the current page is also the last page. This
function can only be used in the footer lines of tables, in objects linked with tables or in the layout regions condition! In all other cases, the result of Lastpage() is always False.

\section*{Parameter:}

Return value:

\section*{Boolean}

\section*{Example:}

Cond(Lastpage(),"Total sum","Subtotal")

\section*{Left\$}

\section*{Purpose:}

Reduces a string from the right so that only the number of characters set under Number remain. If the original string is already small enough, it is not affected.
Parameter:
String The value to be shortened
Number maximum number of positions of the result
Boolean (optional) True: The cut off value is ended with "..." (Default: False). With numbers \(<3\) the setting is ignored.

\section*{Return value:}

\section*{String}

\section*{Examples:}

If you had a customer database that contains, amongst other things, the field NAME for the surname. You now wish to search for all customers whose surname starts with "C". To do this, you must first identify the starting letters.

Left\$(NAME, 1)
NAME string. Left\$("combit", 2)
Left\$("combit", 4,True)
Left\$("combit", 2, True)

Result: the first letter of the

Result: "co"
Result: "c
Result: "co"
combit List \& Label

Len

\section*{Purpose:}

Returns the number of characters in a string.
Parameter:
String
Return value:
Number
Example:
Len("1234"+"12") Result: 6

\section*{LoadFile\$}

\section*{Purpose:}

Outputs the contents of the file as a string.
Parameter:
String
String (optional) When the file is not available, the value set here is used.
Return value:
String
Example:
LoadFile\$("C:\log.txt","File not found!")

\section*{Locale\$}

\section*{Purpose:}

Returns information about the country settings, for example currency, decimals, separators, language and country code. The code for the appropriate country is entered in the second parameter, if no second parameter is used the default country settings will be used.

\section*{Parameter:}

Number Index of Locale Entry
String
(optional) ISO 3166-Country code
Return value:
String
Example:
Locale\$(42,"USA") Result: "Monday"
Possible constants for index entry:
http://msdn.microsoft.com/en-us/library/bb507201.aspx

\section*{LocCurr\$}

\section*{Purpose:}

Returns a string with the valid currency format without the currency symbol for the entered country.
Parameter:
Number Value to be formatted
String (optional) ISO 3166-Country code
Return value:
String
Example:
LocCurr\$(123,"USA") Result: "123.00"

\section*{LocCurrL\$}

\section*{Purpose:}

Returns a string with the valid currency format and currency symbol for the entered country.
Parameter:
Number Value to be formatted
String (optional) ISO 3166-Country code
Return value: String
Example:
LocCurr\$(123,"USA") Result: "\$123.00 "

\section*{LocDate\$}

\section*{Purpose:}

Returns a string with the valid date format for the entered country.
Parameter:
Date Value to be formatted
String (optional) ISO 3166-Country code
Number (optional) Format
Return value:
String
Example
LocDate\$(Date("17.11.2007"),"USA") Result:
"11/17/2007"

\section*{LocDateTime}

\section*{Purpose}

Converts the string into a date (with time if required). The string is expected to be in the relevant format for the country.
Parameter:

String
String

Date
(optional) ISO 3166-Country code

Return value:
Date

\section*{Example}

LocDateTime("04.07.1776","DEU") Result: "07/04/1776"

\section*{LocNumber\$}

\section*{Purpose:}

Returns a string with the valid number format for the entered country.
Parameter:
Number Value to be formatted
String (optional) ISO 3166-Country code
Return value:
String
Example:
LocNumber\$(123,"USA")
Result:
"123.00"

\section*{LocTime\$}

\section*{Purpose:}

Returns a string with the valid time format for the entered country.
Parameter:
Date Value to be formatted
String (optional) ISO 3166-Country code
Number (optional) Format
Return value:
String
Example:
LocTime\$ (Now(),"USA")
Result: 9:05:22 AM"

\section*{LocVal}

\section*{Purpose:}

Interprets the string as a number and returns its value
(while respecting any localized decimal or 1000
separators).
Parameter:
String Number (as string)
String (optional) ISO 3166-Country code
Return value:
Number
Example:
LocVal ("12,00","USA")
Result: 1200,00

\section*{Purpose:}

Calculates the natural logarithm \(\ln (x)\).
Parameter:
Number
Return value:
Number
Example:
Log(Exp(1)) Result: 1

\section*{Log10}

\section*{Purpose:}

Calculates the base-10 logarithm \(\log (x)\).
Parameter:
Number
Return value:
Number
Example:
Log10(1000) Result: 3

\section*{Lower\$}

\section*{Purpose:}

Converts the characters of a string into lower case letters.
Parameter:
String
Return value:
String
Example:
Lower\$("George") Result: "george"

\section*{LTrim\$}

\section*{Purpose:}

Removes the leading spaces of a string.
Parameter:
String
Return value:

\section*{String}

Example:
LTrim\$(" George") Result: "George"
combit List \& Label

\section*{Max}

\section*{Purpose:}

Returns the largest of the two values.
Parameter:
Number or Date
Number or Date
Return value:
Number or Date

\section*{Maximum}

\section*{Purpose:}

Calculates the maximum of the set of values that result from the first parameter / formula.
Parameter:
Number
Boolean (optional) TRUE: The values which were stored for the calculation are deleted after output. (default: TRUE). Please note that the stored calculation values are generally deleted for every (sub)table end. The second parameter only decides whether the values are already deleted within the table.

\section*{Return value:}

Number
Example:
Maximum(Order_Details.ProductID@Products.ProductID:Un

\section*{Median}

\section*{Purpose:}

Calculates the median of the set of values that result from the first parameter / formula.
Parameter:
Number Expression for the value to be averaged.
Boolean (optional) The values which were stored for the calculation are deleted after output. (default: TRUE). Please note that the stored calculation values are generally deleted for every (sub)table end. The second parameter only decides whether the values are already deleted within the table.
Return value:
Number
Example:
Median(UnitsInStock)

\section*{Mid\$}

\section*{Purpose:}

Returns a part of a string. The desired number of characters starting at the starting position will be returned.
If the third parameter is not used, the string will be returned from the starting position to the end.
The first character of the string has the Position 0.
Parameter:

\section*{String}

Number Starting position
Number (optional) Number of characters to be displayed.
Return value:
String
Example:
Mid\$("Normalconsumer",6) Result: "consumer"
Mid\$("Normalconsumer",6,30) Result: "consumer"
Mid\$("Normalconsumer",6,3) Result: "con"
Mid\$(Name,0,6)
Result: "Normal"
combit List \& Label

\section*{Min}

\section*{Purpose:}

Returns the smallest of the two values.
Parameter:
Number or Date
Number or Date
Return value:
Number or Date

\section*{Minimum}

\section*{Purpose:}

Calculates the minimum of the set of values that result from the first parameter / formula.
Parameter:
Number
Boolean (optional) TRUE: The values which were stored for the calculation are deleted after output. (default: TRUE). Please note that the stored calculation values are generally deleted for every (sub)table end. The second parameter only decides whether the values are already deleted within the table.
Return value:
Number
Example:
Minimum(Order_Details.ProductID@Products.ProductID:Uni

\section*{Minute}

Determines the minute of the entered date, and returns the result as a number. If the parameter is not used, the minute of the time of printing will be returned.
Parameter:
Date (optional)
Return value:
Number

\section*{Mode}

\section*{Purpose:}

Calculates the mode (most common value) of the set of values that result from the first parameter / formula.
Parameter:
Number Expression for the value to be examined.
Boolean (optional) The values which were stored for the calculation are deleted after output. (default: TRUE). Please note that the stored calculation values are generally deleted for every (sub)table end. The second parameter only decides whether the values are already deleted within the table.
Return value:
Number

\section*{Month}

\section*{Purpose:}

Determines and returns the month (1...12) as a number. Parameter: Date
Return value:
Number
Example:
Month(Date("2007.10.17")) Result: 10

\section*{Month\$}

\section*{Purpose:}

Determines and returns the month (1...12) as a string. Parameter: Date
Return value:
String
Example:
Month\$(Date("2007.10.17")) Result: "10"
combit List \& Label

Now

\section*{Purpose:}

Returns the current date and time.
Parameter:
Return value: Date

\section*{NthLargest}

\section*{Purpose:}

Calculates the nth-largest value of the set of values that result from the first parameter / formula.
Parameter:
Number
Number <n>, i.e. the index for the value which is to be returned (1-based).
Boolean (optional) TRUE: The values which were stored for the calculation are deleted after output. (default: TRUE). Please note that the stored calculation values are generally deleted for every (sub)table end. The second parameter only decides whether the values are already deleted within the table.
Return value:
Number
Example:
NthLargest(Order_Details.ProductID,2) calculates the
2-largest number

\section*{NthLargestIndex}

\section*{Purpose:}

Calculates the index of the nth-largest value of the set of values that result from the first parameter / formula.
Parameter:
Number
Number <n>, i.e. the index for the value which is to be returned (1-based).
Boolean (optional) TRUE: The values which were stored for the calculation are deleted after output. (default: TRUE). Please note that the stored calculation values are generally deleted for every (sub)table end. The second parameter only decides whether the values are already deleted within the table.
Return value:
Number
Example:
NthLargestIndex(Order_Details.ProductID,2)

\section*{NthValue}

\section*{Purpose:}

Calculates the nth value of the set of values that result from the first parameter / formula.
Parameter:
All
Number <n>, i.e. the index for the value which is to be produced, calculated e.g. with NthLargestIndex().
Boolean (optional) TRUE: The values which were stored for the calculation are deleted after output. (default: TRUE). Please note that the stored calculation values are generally deleted for every (sub)table end. The second parameter only decides whether the values are already deleted within the table.
Return value:
Number
Example:
NthValue(NthLargestIndex(Order_Details.ProductID,2))
combit List \& Label

Null

\section*{Purpose:}

Returns a Null value (value not available).
Parameter:
Return value: All

\section*{NullSafe}

\section*{Purpose:}

Checks if the parameter is Null and returns a substitute value if it is, otherwise it returns the value of the parameter.
Parameter:
All
Return value:
AII

\section*{NumInRange}

\section*{Purpose:}

Evaluates if a number falls within the desired range. Parameter:

Number
Number Upper limit
Number Lower limit
Return value:

\section*{Boolean}

\section*{Example:}

NumInRange(Page(),1,10) Result: True, if page number is between 1 and 10 .

\section*{Odd}

\section*{Purpose:}

Evaluates if a number is odd. If the number is odd "True will be returned, otherwise "False \(\geqslant\).
Parameter:
Number
Return value:
Boolean
Example:
"Page number "+Cond(Odd(Page()),"odd","even")

\section*{Ord}

\section*{Purpose:}

Returns the ASCII value of the first character.
Parameter:
String
Return value:
Number
Example:
Ord("combit") Result: 99

\section*{Page}

\section*{Purpose:}

Returns the current page number.
Parameter:
Return value:
Number
Example:
Case(Odd(Page()),"Even","Odd")+" page number"

\section*{Page\$}

\section*{Purpose:}

Returns the page number of the printed page as a string. Parameter:

Return value: String
Example:
"Page "+Page\$()+"/"+TotalPages\$() Result: Page 1/3
combit List \& Label

\section*{Pow}

\section*{Purpose:}

Corresponds to the function (Base) ^ (Exponent).
Parameter:
Number Base
Number Exponent
Return value:
Number
Example:
Pow \((2,3) \quad\) Result: 8

\section*{Previous}

\section*{Purpose:}

Returns the previous value of the variable, field or formula, i.e. the value it had for the last record.

Parameter:
All Variable, field or formula
Return value:
All
Example:
Previous(NAME) Result: "consumer"

\section*{PreviousUsed}

\section*{Purpose:}

Returns the value the given variable or expression had when it was last evaluated.
Parameter:
All Variable, field or formula
Return value:
All
Example:
PreviousUsed(NAME) Result: "Mustermann"

\section*{ProjectParameter\$}

\section*{Purpose:}

Returns the value of a project parameter. Available parameters:
\begin{tabular}{ll}
\hline LL.FAX.Queue & Print queue \\
\hline LL.FAX.RecipName & Recipient name \\
\hline LL.FAX.RecipNumber & Recipient fax number \\
\hline LL.FAX.SenderBillingCode & Sender billing code \\
\hline LL.FAX.SenderCompany & Sender company \\
\hline LL.FAX.SenderDept & Sender department \\
\hline LL.FAX.SenderName & Sender name \\
\hline LL.MAIL.To & Mail address \\
\hline LL.MAIL.CC & \begin{tabular}{l} 
Mail address for \\
carbon copy
\end{tabular} \\
\hline
\end{tabular}
LL.MAIL.BCC Mail address for blind carbon copy
\begin{tabular}{ll}
\hline LL.MAIL.Subject & Subject line \\
\hline LL.MAIL.From & Sender mail address \\
\hline LL.MAIL.ReplyTo & Reply To mail address \\
\hline LL.MinPageCount & Minimum page count. \\
\hline LL.ProjectDescription & Project Description \\
\hline LL.SlideShow.TransformationID & \begin{tabular}{l} 
Default transition \\
effect for the preview \\
's slideshow mode.
\end{tabular} \\
\hline LL.MAIL.ShowDialog & \begin{tabular}{l} 
Show mail dialog \\
before sending
\end{tabular} \\
\hline
\end{tabular}

\section*{Parameter:}

\section*{Return value:}

\section*{String}

\section*{Example: \\ ProjectParameter\$("LL.ProjectDescription") Result:"Article list"}

\section*{ProjectPath\$}

\section*{Purpose:}

Returns the path of the project file, optionally including the file name (otherwise with " \(\backslash\) " at the end) Parameter:

Boolean True: Sets that the path is returned including the file name (Default: False).
Return value:
String
Example:
ProjectPath\$() Result: C:\Program Files\LL\
ProjectPath\$(True) Result: C:\Program
Files\LL\crosstab.Isr

\section*{Quarter}

\section*{Purpose:}

Returns the quarter of the year (1..4)
Parameter:
Date
Boolean (optional) sets whether the quarter calculation should be returned relative to the year (1..4) or in absolute terms since 1.1.0001 (1..). Default: False (relative).

Return value:
Number
Example:
Str\$(Quarter(Date("01.01.2008")),0,0)
Str\$(Quarter(Date("01.05.2008")),0,0)
Str\$(Quarter(Date("01.05.2008"),.T.),0,0)
Result: 1

8030

Result: 2
Result:

\section*{RainbowColor}

\section*{Purpose:}

Calculates a color value between violet and red corresponding to the value of the first parameter e.g. for rainbow colors in crosstabs.
Parameter:
Number Value to be displayed.
Number Limiting value for violet.
Number Limiting value for red.
Return value:
Number

\section*{RegExMatch\$}

\section*{Purpose:}

Returns the part of the string that corresponds to the regular expression or the group passed in the third parameter.
The regular expression corresponds to Pearl 5 Syntax, which in most details equals the regular expression syntax of the Visual Basic Scripting engine.
Parameter:

\section*{String}

String
Number
Return value:

\section*{String}

Example:
Division of the "STREET" field to street and number:
"Street: " + RegExMatch\$(STREET,"((?:\w* )+)(\d+[\w
]*\$)",1) "Number: " + RegExMatch\$(STREET,"((?:\w* )+)
(\d+[\w ]*\$)",2)
RegExMatch\$("test1234xyz0815", "[0-9]+") Result:
"1234"

\section*{RegExSubst\$}

\section*{Purpose:}

Replaces the substrings of the first argument with a new value if they match the regular expression.
Parameter(s):

String
String
String

The string to be checked
Regular expression
Replacement expression (can contain " 10 " for the entire match or " \(\backslash 1\) " \(\uparrow\) " \(\backslash 9\) " for the respective group.
Boolean (optional) Specifies whether only the first occurrence is to be replaced. Default value: False

\section*{Return value:}

\section*{String}

\section*{Example:}

RegExSubSt\$("1234xyz6789","[0-9]+", "a") result:
"axyza"
RegExSubSt\$("1234xyz6789","[0-9]+", "a") result:
"axyz6789"

\section*{RemainingTableSpace}

\section*{Purpose:}

Returns the space available to data and group lines in a table object. The parameter defines the unit of the return value. The function can be used to carry out conditional page breaks before group lines, e.g. "Page break before only \(5 \%\) space is left \(\widehat{\text { p }}\).

\section*{Parameter:}

Boolean (optional) TRUE: the value is in units which are independent of the system (SCM-units), FALSE: the value is a percentage of the entire table size (default: FALSE).
Return value:
Number

\section*{Rep\$}

\section*{Purpose:}

Returns a string that contains the appropriate number of strings defined in the first parameter.
Parameter:
String
Number
Return value:

\section*{String}

Example:
Rep\$("-",10) Result: "
Rep\$("+-",5) Result: "+-+-+-+-+-"

\section*{RGB}

\section*{Purpose:}

Calculates the color value using the relative red, green and blue saturation values (between 0 and 255). No saturation has the value 0, full saturation the value 255 . This function can be used to set the font color using a formula.

\section*{Parameter:}

Number red saturation
Number green saturation
Number blue saturation
Return value:
Number
Example:
Cond(Amount<0, RGB(255,0,0), RGB(0,0,0) Result: red for negative amounts

\section*{Right\$}

\section*{Purpose:}

Reduces a string from the left so that only the number of characters set under Number remain. If the original string is already small enough, it is not affected.
Parameter:
String
Number
Boolean (optional) True: The cut off value starts with "..." (Default: False). If Number < 3 the setting is ignored.
Return value:

\section*{String}

Example:
Right\$("normalconsumer", 8) Result: "consumer" Right\$("normalconsumer", 11,.T.) Result: "...consumer"

\section*{Round}

\section*{Purpose:}

Rounds a value to the entered number of decimal places. Default is 0 .
Parameter:
Number
Number (optional)
Return value:
Number
Example:
Round(3.1454,2) Result: 3,15
Round(3.1454) Result: 3

\section*{RTrim\$}

\section*{Purpose:}

Removes spaces from the end of a string. Parameter:

String
Return value:
String
Example:
RTrim\$("John ") Result: "John"

\section*{Second}

Determines the second of the entered date and returns the result as a number. If the parameter is not used, the second of the print time will be returned.

Parameter:
Date (optional)
Return value:
Number

\section*{SetVar}

Saves a value in the variable repository for later use with the GetVar() function. The purpose of these functions is to provide a simple buffer for values. You shouldn't execute complex nesting with GetVar/SetVar or combine both functions with each other - especially for header, footer and group lines unexpected effects can occur here.
Parameter(s):
String
All
Return value: All

\section*{Example:}

SetVar ("Page", Page())

\section*{Sign}

\section*{Purpose:}

Returns the sign of the value (+1 for a positive value, -1 for a negative value or 0 if the value is 0 ).
Parameter:
Number Value
Return value:
Number
Example:
Sign (-3) Result: -1

\section*{Sin}

\section*{Purpose:}

Calculates the sine of the value.

\section*{Parameter:}

Number Value
Number (optional) Mode (0=Degree, 1=Radian). Default: 0.

\section*{Return value:}

Number
Example:
Sin (90) Result: 1
combit List \& Label

\section*{Sqrt}

\section*{Purpose:}

Calculates the square root of a number.
Parameter:
Number
Return value:
Number
Example:
Sqrt(4) Result: 2

\section*{StartsWith}

\section*{Purpose:}

Checks whether the string in the first argument begins with the string in the second argument.
Parameter(s):

\section*{String}

String
Boolean Specifies whether capitals/small letters
are relevant. Default value: False
Return value:
Boolean
Examples:
StartsWith ("Hello World","hel")
Result: True
StartsWith ("Hello World","rld")
Result: False

\section*{StdDeviation}

\section*{Purpose:}

Calculates the standard deviation of the set of values that result from the first parameter / formula.
Parameter:
Number
Boolean (optional) TRUE: The values which were stored for the calculation are deleted after output. (default: TRUE). Please note that the stored calculation values are generally deleted for every (sub)table end. The second parameter only decides whether the values are already deleted within the table.

\section*{Return value:}

Number
Example:
StdDeviation(Order_Details.Quantity*Order_Details.UnitPric

\section*{Str\$}

\section*{Purpose:}

Converts a number into a string. The number will be formatted with 6 decimal places that may be rounded. The length is variable.

\section*{Parameter:}

Number
Number (optional) Defines the length of the string (default:6). If the number is too large for this format, the resulting string may then be longer than desired. If the number is too small, spaces will be attached dependent upon the prefix, right (negative) or left (positive).
Number
(optional) Defines the precision (number of decimal places). If the number is positive, the number will be displayed as a floating-point number, if negative in scientific format.

\section*{Return value:}

\section*{String}

\section*{Example:}

Str\$(Pi)
Str\$(Pi,3)
Str\$(Pi,3,0)
Str\$(-Pi,12,-3)
Str\$(Page())
Str\$(Page(),10)
Str\$(Page(),-10)

Result: "3.141593"
Result: " 3"
Result: " 3"
Result: "-3.141e+00"
Result: "5.000000"
Result: " 5"
Result: "5 "

\section*{StrPos}

\section*{Purpose:}

Returns the position of the nth appearance of a search string. The third parameter determines which appearance will be returned. Default is 1 .
The first character in the string has the position 0.
-1 as return value signifies the search string no longer appears.
Parameter:
String
String
Search string
Number (optional)
Return value:
Number
Example:
StrPos("Normalconsumer","or")
Result: 1
StrPos("Normalconsumer","r")
StrPos("Normalconsumer","r",1)
StrPos("Normalconsumer","r",2) Result: 2
Result: 2
Result: 13

\section*{StrRPos}

\section*{Purpose:}

Returns the position of a search string within a string.
This is a backwards search. The third parameter, determines which appearance will be returned. Default is 1.

The first character in the string has the position 0.
-1 as return value signifies the search string no longer appears.

\section*{Parameter:}

\section*{String}

String Search string
Number (optional)
Return value:
Number

\section*{Examples:}

StrRPos("Normalconsumer","or")
StrRPos("Normalconsumer","r")
StrRPos("Normalconsumer","r",1)
StrRPos("Normalconsumer","r",2)

Result: 1
Result: 13
Result: 13
Result: 2

\section*{StrSubst\$}

\section*{Purpose:}

Searches a string for the appearance of a search string and replaces it with the string contained in the third parameter (replacement string). If no third parameter is used, the string located using the search string will be deleted.
Parameter:
String
String
String
Return value:

\section*{String}

Example:
Assume that you want to print address labels that contain the company name. You do not have much space available on the label and cannot afford to completely print long company names, for example, "Forrer Construction, Incorporated
With the expression StrSubst\$(COMPANY,"Incorporated","Inc.") every appearance of "Incorporated" in the COMPANY field will be replaced with "Inc."

\section*{Sum}

\section*{Purpose:}

Calculates the sum of the parameter / formula in the parameter.
Hint: Sum variables (see Working With Sum Variables) are an alternative way of creating sums and counters. Sum variables are principally applicable to whole tables. Aggregate functions principally table specific.
Parameter:
Number
Boolean (optional) TRUE: The values which were stored for the calculation are deleted after output. (default: TRUE). Please note that the stored calculation values are generally deleted for every (sub)table end. The second parameter only decides whether the values are already deleted within the table.

\section*{Return value:}

Number
Example:
Sum (Order_Details.UnitPrice)

\section*{Tan}

\section*{Purpose:}

Calculates the tangent of the value.

\section*{Parameter:}

Number Value
Number (optional) Mode (0=Degree, 1=Radian). Default: 0 .

\section*{Return value:}

Number
Example:
Tan (45) Result: 1,00

\section*{Time\$}

\section*{Purpose:}

Returns the current time in string format. The following formats are available:
Placeholder Description
\begin{tabular}{ll}
\hline\(\% \mathrm{~h}\) & Hours in 24 hour format \\
\hline\(\% \mathrm{H}\) & Hours in 12 hour format \\
\hline\(\% \mathrm{~m}\) & Minutes \\
\hline\(\% \mathrm{~s}\) & Seconds \\
\hline\(\% \mathrm{P}\) & \begin{tabular}{l} 
Display the part of day \\
(A.M. / P.M.)
\end{tabular} \\
\hline\(\% \mathrm{p}\) & \begin{tabular}{l} 
Display the part of day \\
(a.m. / p.m.)
\end{tabular} \\
\hline
\end{tabular}

\section*{Parameter:}

String
Return value:
String
Example:
Time\$("\%02h:\%02m:\%02s") Result: "18:30:45"

\section*{Today}

\section*{Purpose:}

Returns the current date.
Parameter:
Return value:
Date
Example:
Date\$(Today(),"\%D, \%m.\%d.\%4y")
Result: "Friday, 11/8/2007"
combit List \& Label

\section*{Token\$}

\section*{See Case\$}

\section*{ToRTF\$}

\section*{Purpose:}

Returns a string in RTF-format. This is necessary because some strings may contain one of the specially defined RTFformat symbols. ('\', '\{' or '\}'). For compatibility reasons, this function only processes the passed string if the optional second parameter is explicitly set to True.

\section*{Parameter: \\ String \\ Boolean \\ Return value: \\ String \\ Example:}

If, for example, the field PRODCODE could contain one of the characters, then the text should be inserted in the following way:
"<<SALUTATION>> <<NAME>>, You have received our product <<PRODUCT>>, Code <<ToRTF\$(PRODCODE)>>..."

\section*{Total\$}

\section*{Purpose:}

Sets that the expression in the argument is calculated for the whole crosstable-object.
Parameter:
All
Return value:
All
Example:
Sum(sales)/Total(Sum(sales))*100

\section*{TotalPages\$}

\section*{Purpose:}

Returns the total number of pages. The returned string is replaced by the total number of pages when printing. Please note when using this function that the timing behavior of the print process can be affected. A progress bar may reach \(100 \%\) faster, but because of further processing of the output, there may be a delay before the actual printout is produced. No calculations may be performed with the result of this function.
Parameter:
Return value: String
Example:
"Page "+Page\$()+"/"+TotalPages\$() Result: Page 1/3

\section*{Translate\$}

\section*{Purpose:}

Translates the text in the argument provided that it is held in the dictionary transferred by the application.
Parameter(s):
String
Return value:
String
Example:
Translate\$("Page \{0\} of \{1\}", Page\$(), TotalPages\$())
Result in German e.g.: Seite 1 von 2

\section*{UnitFromSCM}

\section*{Purpose:}

Converts a SCM-Unit (1/1000 mm) to the print unit (inch/mm). Important for the definitions of property values independent of the selected print unit.
Parameter:
Number
Return value:
Number
Example:
Cond(Page()=1,UnitFromSCM(100000),UnitFromSCM(2000 ( Result: 10 cm for Page 1, 2 cm for the other pages.

\section*{Upper\$}

\section*{Purpose:}

Converts the characters of a string to capital letters.

Upper\$("Otto") Result: "OTTO"

\section*{Val}

\section*{Purpose:}

The string is interpreted and returned as a number. If an error occurs, the return value is 0 . The decimal sign must always be entered as ".".
Parameter:
String
Return value:
Number
Example:
Val("3.141")
Result: 3.141
Val("3,141")
Result: 3
Val("3.141e2")
Val(ChrSubst\$("3,141", ",", "."))
Result: 314.2
Result: 3.141

\section*{Variance}

\section*{Purpose:}

Calculates the variance of the set of values that result from the first parameter / formula.
Parameter:
Number
Boolean (optional) TRUE: The values which were stored for the calculation are deleted after output. (default: TRUE). Please note that the stored calculation values are generally deleted for every (sub)table end. The second parameter only decides whether the values are already deleted within the table.

\section*{Return value:}

Number
Example:
Variance(Order_Details.Quantity*Order_Details.UnitPrice)
combit List \& Label

\section*{Woy}

\section*{Purpose:}

Returns the week number of a given date.
The optional second parameter determines the setting for the first week of the year.
0 Week with the first
working day
1 Week of January, 1
2 First week with at least 4
days
\(3 \quad\) First week with 7 days
4 Week with the first
Monday
Parameter:
Date
Number
Return value:
(optional)
Number

\section*{Year}

\section*{Purpose:}

Determines the year of a date and returns it as a number. Parameter: Date
Return value:
Number
Example:
Year(Today())
Result: 2010
Year\$(Date("1.1.2010"))

\section*{Year\$}

\section*{Purpose:}

Determines the year of a date and returns it as a string. Parameter:

\section*{Date}

Return value:
String
Example:
Year\$(Today())
Year\$(Date("1.1.2010"))

ゅ
combit List \& Label

\section*{Overview of Properties}

All of the properties for projects and objects are described centrally here.
Properties are defined by means of the respective property lists. If you select multiple objects, you can set their common properties at the same time. You can specify values in different ways depending on the property.
§ Open a drop down list of values by means of an "arrow down" button.
Example: Appearance condition, font color, font. At the end of the list of values, you will almost always find the "Formula" entry.
§ You can set the value with a formula via the formula button or the "Formula" entry in the list of values.
Example: If you want to set the font color to red for negative values, set the default "property" for the font to "False" and define the "Font color" property using a formula, e.g.:
Cond(Item.UnitPrice< 0,LL.Color.Red,LL.Color.Black)
§ Open a configuration dialog with the "..." button.
For example, there are dialogs for the following properties:
formatting, font, frames, position, label format.
§ Enter the value directly in the property fields.
Example: Project description in the project properties.
§ Set a file path with the open dialog.
Example: Name of the project include file or the image file.

See also:
- Project Properties
- Common Object Properties
- Text Objects
- Line Objects
- Rectangle Objects
- Circles and Ellipses
- Picture Objects
- Barcode Objects
- Report Container Object
- Table Objects
- Chart Objects
- Gauge Objects
- Crosstab Objects
- Formatted Text Objects
- Form Control Objects
- HTML Text Objects
- OLE Container
- Form Template Objects

ゅ
combit List \& Label

\section*{Project Properties}

The project's property window is displayed if no object is selected in the workspace.
The project properties are also available as fields (see Overview of Fields) and can be evaluated with the ProjectParameter\$() function.

See also:
- General Settings
- Fax and Mail Variables

\section*{General Settings}

See also:
- Project Description
- Active Design Layout
- Number of Issues
- Display Condition for Issue Print
- Embedding Drilldown Reports
- Minimum Page Count
- Transition Effects for Slideshow

\section*{Project Description}

You can enter a description for the respective project in the "Project description" field. This description is then shown in the File > Open dialog making it easier for you to find the project that you want. Alternatively, you can also enter the description in the File > Save As dialog.


Figure 13.1: Project properties
combit List \& Label

\section*{Active Design Layout}

With multi-page projects, it is sometimes a good idea to choose different layout settings, e.g. page size, orientation for the different pages. You specify which layout setting is to be shown in the workspace by means of the "Active design layout" field.
combit List \& Label

\section*{Number of Issues}

Specifies the number of issues (copies) for printing and previewing. In addition, it also enables the Issuelndex() function for display and layout region conditions.
If you specify multiple issues, you will then have the "Display condition for issue print" property which you can use for printing of objects conditionally for the different copies.
combit List \& Label

\section*{Display Condition for Issue Print}

Allows print conditions to be set for the pages of the different issues, e.g. if the last page containing the GTC should be suppressed when printing the copy.
Example: If (IssueIndex()=2, not Lastpage(), True)
combit List \& Label

\section*{Embedding Drilldown Reports}

Drilldown reports can be embedded in the preview file to allow them to be sent or saved as a complete unit.
combit List \& Label

\section*{Minimum Page Count}

With index card projects, this property specifies the minimum number of pages that are to be printed automatically. For example, if you want to output a four-page form with different layouts for each of the four pages, you create a layer for each page and position the objects on these layers as required for the output. Specify "4" as the minimum page count.
With list projects, the number entered here determines the page number on which the output of the table/report container will start. For example, if you need a covering sheet, you can assign the "Following pages" layer to the table and design the "First page" layer as you wish. Then specify "2" as the minimum page count.
combit List \& Label

\section*{Transition Effects for Slideshow}

Here you specify the default values for the kind of page transition in the preview's slideshow mode.

\section*{Fax and Mail Variables}

You send faxes of List \& Label documents by selecting the respective fax (printer) driver in the print process. A fax program must be installed in order to be able to send faxes. If the fax is to be sent via the Windows fax driver, the fax parameters (at least the fax number) must be specified in the project properties. Enter the respective variables in the "Fax Parameters" area.
If the fax is to be sent via a different fax (printer) driver, you enter the fax number and other field information (as far as supported) by means of commands (e.g. DvISE commands for Tobit David). You enter these commands directly in a text field in the print project. Doing this suppresses the recipient dialog during printing because all information is already embedded in the document. You will find the precise procedure in the documentation for your fax software. You can also send List \& Label documents directly by email. You also define the required email variables in the project's property window. Make the email settings (SMTP, MAPI, XMAPI) under "combit Mail" in the Windows Control Panel.

ゅ
combit List \& Label

\section*{Common Object Properties}

You specify most of the object properties in the property list and/or in additional dialogs. Each object type has its own individual properties. However, there are a number of attributes that are common to all objects, such as size, position, name and, appearance condition. These properties are described here centrally and explained in more detail in the sections that follow.

See also:
- Locked
- Name
- Display Condition for Issue Print
- Appearance Condition
- Levels in the Table of Contents
- Page Break Before Outputting Object
- Export as Picture
- Position
- Font
- Color
- Background / Filling
- Pattern
- Frame
- Format
- Content

\section*{Locked}

Locks the object to prevent it from being selected unintentionally by clicking. This property is only relevant during design and has no effect on the later print. If you set "Locked" to "True", the object in question can no longer be selected in the workspace and will be marked with a small red logo. This property is not available for sub-tables.


Figure 13.2: Locked objects in the object list
Note: You can select a locked object as usual in the object list thereby making it editable again. Since "locked" is only relevant during the project s design phase, there is no way in which you can determine the value of the property by means of a formula.
\begin{tabular}{llll}
\hline Property & Description & Value & Description \\
\hline Locked & \begin{tabular}{l} 
Locks the object and prevents \\
it from being selected \\
unintentionally by clicking in \\
the workspace.
\end{tabular} & \begin{tabular}{l} 
locked \\
not locked
\end{tabular} \\
& & \\
\hline
\end{tabular}

\section*{Name}

When you add a new object to the workspace, a description of the object, made up of the type of the object (e.g. "Text") and its coordinates, appears in the right section of the status line. This is the default name for this object.
However, if your project has a large number of similar objects, these identifiers can easily become confusing. For this reason, you can give your objects meaningful names by means of the Objects tool window or with the object's property list. You do this by simply clicking once on the existing name and then changing it.
Alternatively, you can enter a new name in the object name input field via Objects > Object list or you can change it in the property list.
\begin{tabular}{llll}
\hline Property & Description & Value & Description \\
\hline Name & Name of the object & Name &
\end{tabular}

If you have enabled the Options > Workspace > Object info option, the object name will also be shown in the tooltip that appears.

\section*{Display Condition for Issue Print}

This option enables conditional printing of objects for the different issues. This property is only available if you have defined multiple issues in the project properties. The Issuelndex() function lets you specify the index of the issue, e.g. Issuelndex()=2. You will find more information about managing issues in chapter Project Properties.
\begin{tabular}{llll}
\hline Property & Description & Value & Description \\
\hline Display & Enables conditional printing of & True & Display \\
condition & objects for the different issues, & False & Hide \\
for & issue & e.g. Issuelndex()=2. & Formula \\
print & & & Formula \\
\hline
\end{tabular}

\section*{Appearance Condition}

You can assign an appearance condition to each object. This specifies under which conditions the object is to be printed. You will find a guide to defining such conditions under Variables, Formulae and Expressions.
\begin{tabular}{|c|c|c|c|}
\hline Property & Description & Value & Description \\
\hline \multirow[t]{17}{*}{Appearance condition} & \multirow[t]{2}{*}{Appearance condition for printing. There is no output if the result is false.} & True & Always show \\
\hline & & False & Never \\
\hline & In appearance conditions & FirstHeaderThisTable() & show \\
\hline & for table footers, you can & LastFooterThisTable() & First page \\
\hline & also use the predefined & Formula & only \\
\hline & "Last page only" value. & & \\
\hline & Internally, this entry uses & & Last page \\
\hline & the functions Lastpage() or & & only \\
\hline & LastFooterThisTable(). & & \\
\hline & In appearance conditions & & Formula \\
\hline & for table headers, you can & & \\
\hline & also use the predefined & & \\
\hline & "First page only" value. & & \\
\hline & Internally, this entry uses & & \\
\hline & the functions not & & \\
\hline & Lastpage() or & & \\
\hline & FirstHeaderThisTable(). & & \\
\hline
\end{tabular}

\section*{Levels in the Table of Contents}

Specifies the outline level (index level) of the bookmark in preview mode or for PDF export ( \(0=\) not in index). You can set the maximum folder depth via Project > Options > Project.
\begin{tabular}{llll}
\hline \begin{tabular}{l} 
Levels in \\
the table \\
of \\
contents
\end{tabular} & \begin{tabular}{l} 
Specifies the index level of the \\
bookmark ( \(0=\) not in index).
\end{tabular} & \begin{tabular}{l} 
Number \\
Formula
\end{tabular} & \begin{tabular}{l} 
Formula \\
wizard
\end{tabular} \\
\hline Text & Name of the bookmark. & Text & \begin{tabular}{l} 
Formula \\
wizard
\end{tabular} \\
\hline
\end{tabular}

\section*{Page Break Before Outputting Object}

Each object can trigger a page break before it is printed, i.e. the object begins on a new page.
\begin{tabular}{llll}
\hline Property & Description & Value & Description \\
\hline \begin{tabular}{lll} 
Page break \\
before
\end{tabular} & \begin{tabular}{l} 
If the condition returns "True", \\
a page break will be triggered \\
before printing the object.
\end{tabular} & False & Break \\
Formula & \begin{tabular}{l} 
Formula \\
wizard
\end{tabular} \\
\hline
\end{tabular}

\section*{Export as Picture}

For exporting objects in picture format if a vector-based export does not give the desired results, or in order to achieve a better representation.
\begin{tabular}{lllll}
\hline Property & Description & Value & Description \\
\hline Export as & If the result is "True", the & True & Yes \\
picture & \begin{tabular}{l} 
object will be exported as a \\
picture.
\end{tabular} & \begin{tabular}{ll} 
False & No \\
&
\end{tabular} & & \begin{tabular}{l} 
Formula \\
wizard
\end{tabular} \\
\hline
\end{tabular}

\section*{Position}


\section*{Font}

\section*{If the default value is set to "Yes", the default font will be} used.
There is also a dialog for defining the values.
\begin{tabular}{|c|c|c|c|}
\hline Property & Description & Value & Description \\
\hline Font & You can define the font properties in a dialog. If the default value is set to "True", the default font will used. & (A) & Font dialog \\
\hline Default value & The default font will be used instead of the set values. & True False Formula & Default font No Formula wizard \\
\hline Name & Selected font. All installed fonts will be displayed. & List Formula & Font Formula wizard \\
\hline Character set & Specifies the country version of the character set. All available character sets are displayed. & Number & Character set \\
\hline Size & Font size in points. Lists all available sizes for the selected font. & \begin{tabular}{l}
Number \\
Formula
\end{tabular} & Default size Formula wizard \\
\hline Width & Sets the width of the font. 0 means standard width, otherwise the average character width will be specified. & Number Formula & Width Formula wizard \\
\hline Bold & Turns the "bold" text property on and off & True False Formula & \begin{tabular}{l}
Yes \\
No \\
Formula wizard
\end{tabular} \\
\hline Italic & Turns the "italic" text property on and off & True False Formula & Yes No Formula wizard \\
\hline
\end{tabular}
\begin{tabular}{llllll}
\hline Underline & \begin{tabular}{l} 
Turns the "underline" \\
property on and off
\end{tabular} & text & \begin{tabular}{l} 
True \\
False \\
Formula
\end{tabular} & \begin{tabular}{l} 
Yes \\
No \\
Formula \\
wizard
\end{tabular} \\
\hline Strike out & \begin{tabular}{l} 
Turns the "strike \\
property on and off
\end{tabular} & text & \begin{tabular}{l} 
True \\
False
\end{tabular} & \begin{tabular}{l} 
Yes \\
No
\end{tabular} \\
& & & Formula \\
& & \begin{tabular}{l} 
Formula \\
wizard
\end{tabular} \\
\hline Color & Font color (see chapter Color) & & \\
\hline
\end{tabular}

\section*{Color}

The color property lets you specify the color of the font or the background. For background colors, you must also set the "Background" property to a value > 0, e.g. to "Pattern/Block color".
There is also a dialog for defining the values.
\begin{tabular}{|c|c|c|c|}
\hline Property & Description & Value & Description \\
\hline \multirow[t]{2}{*}{Color} & You can define the color in a color dialog. & & Color dialog \\
\hline & \begin{tabular}{l}
In the dialog, you can choose the color from a list of fixed predefined colors or specify your own color by means of a formula or a function ("Formula" entry at the end of the list). \\
(1) With the HSL() function, you define the color by specifying the hue value ( 0 360 ), the saturation value ( \(0-\) 1) and the lightness value (01). \\
(2) The RGB() function defines a color by means of red, green and blue values. Each color portion can have a value between 0 and 255 .
\end{tabular} & & Selection of predefined colors and formula wizard \\
\hline
\end{tabular}

\section*{Background / Filling}

The background/filling property lets you specify a block color or a gradient.
\begin{tabular}{|c|c|c|c|}
\hline Property & Description & Value & Description \\
\hline \multirow[t]{8}{*}{Filling / Background} & \multirow[t]{7}{*}{Select the kind of gradient that you want and specify the properties for color, mid color, end color and fading-in color, depending on background.} & 0 & Transparent \\
\hline & & 1 & Pattern/block color \\
\hline & & 2 & Horiz. \\
\hline & & 3 & gradient \\
\hline & & 4 & Vert. gradient \\
\hline & & 5 & Horiz. 2-part gradient \\
\hline & & 6 & Vert. 2-part gradient \\
\hline & Value 7 only with tables, charts, rectangles or circles. & \begin{tabular}{l}
\[
7
\] \\
Formula
\end{tabular} & Partly transparent Picture Formula wizard \\
\hline Color & Font color (see chapter Color) & & \\
\hline
\end{tabular}

\section*{Pattern}
\begin{tabular}{llll} 
The pattern property lets you specify the texture of a color. \\
\hline Property & Description & Value & Description \\
\hline Pattern & Choose a pattern here from & Selection a \\
& wide range of predefined & of \\
& patterns. Each pattern is & predefined \\
& represented by a number. You & pattern \\
& can specify your own & and \\
& pattern/number by means of a & formula \\
& formula or a function & wizard \\
& ("Formula" entry at the end of & \\
& the list). This property is only & \\
& evaluated if "Filling" or & \\
& "Background" is set to & \\
& "Pattern/Block color". & \\
\hline
\end{tabular}

\section*{Frame}

The "Frame" property group defines the frame properties and distances from the frames.
There is also a dialog for defining the values:


Figure 13.3: Dialog for the frame properties
\begin{tabular}{llll}
\hline Property & Description & Value & Description \\
\hline \begin{tabular}{l} 
Frame \\
(default \\
value)
\end{tabular} & \begin{tabular}{l} 
You can define frame \\
properties and distances in a \\
dialog.
\end{tabular} & \(\ldots\) & Frame dialog \\
& \begin{tabular}{l} 
To apply the selected type of \\
line, color or width, click one \\
of the default settings, the
\end{tabular} & & \\
& \begin{tabular}{ll} 
lines of the preview or use \\
the buttons.
\end{tabular} & & \\
\hline \begin{tabular}{l} 
Default \\
frame \\
setting \\
(with table \\
cells)
\end{tabular} & \begin{tabular}{l} 
If set to True, the default \\
frame defined in the table \\
object will be used.
\end{tabular} & \begin{tabular}{l} 
False
\end{tabular} & \begin{tabular}{l} 
Lines lines \\
Layout
\end{tabular} \\
\hline \begin{tabular}{l} 
Describes the layout of the \\
frame lines (only relevant for \\
multi-line frames).
\end{tabular} & 0 & \begin{tabular}{l} 
Formula \\
wizard
\end{tabular} \\
& \begin{tabular}{lll} 
Lay
\end{tabular} & \begin{tabular}{l} 
tial \\
Horiz.
\end{tabular} \\
& & &
\end{tabular}
\begin{tabular}{lllll} 
& & \begin{tabular}{l}
2 \\
Formula
\end{tabular} & \begin{tabular}{l} 
priority \\
Vert. priority \\
Formula \\
wizard
\end{tabular} \\
\hline \begin{tabular}{llll} 
Left/Top \\
Right/Bottom
\end{tabular} & \begin{tabular}{l} 
Settings for the \\
frame line.
\end{tabular} & respective & & \\
\hline Distance & \begin{tabular}{l} 
Distance between \\
and frame.
\end{tabular} & & content & \begin{tabular}{l} 
Number \\
Formula
\end{tabular} \\
& & & \begin{tabular}{l} 
Formula \\
wizard
\end{tabular} \\
\hline Lines & Visibility of the frame line. & \begin{tabular}{l} 
True \\
False \\
Formula
\end{tabular} & \begin{tabular}{l} 
Lines \\
No lines \\
Formula \\
wizard
\end{tabular} \\
\hline Color & Line color (see chapter Color). & & \\
\hline Line type & Line type. & Line & \begin{tabular}{l} 
Selection of \\
predefined \\
lines (20)
\end{tabular} \\
& & & Formula & \begin{tabular}{l} 
Formula \\
wizard
\end{tabular} \\
\hline Width & Line width. & & Number & Formula
\end{tabular} \begin{tabular}{l} 
Formula \\
\end{tabular}

\section*{Format}

The format property is an alternative to formatting with the functions Date\$() and FStr\$() in the formula dialog. This property can be found, for example, in text, crosstab and table fields. Note that the formatting will affect the expression's result. If you only wish to format certain parts of an expression (e.g. for text and numbers within one expression) use the functions Date\$(), LocCurrL\$ or FStr\$() in the formula dialog.
With the format editor you can set the format for numbers, currency, date, time, date and time, percentage, angle and date-/time difference.
By default, the respective application settings are used. Alternatively select the system setting or a custom setting. If no application setting is passed by the application, the application setting is the same as the system setting.


Figure 13.4: Formatting dialog

\section*{Content}

Many objects cannot be defined solely by means of the property list. They contain sub-objects (or "content"), such as text objects consisting of several paragraphs.
The "Contents" property (if available) opens up a dialog. You will find a description of the respective content dialog accompanying the description of the individual objects.

ゅ
combit List \& Label

\section*{Text Objects}
(4) Text objects let you place text in the workspace. A text object can hold as many paragraphs as you want and they can all have completely different display properties. These paragraphs and their properties present the contents of the text object.
In the paragraph properties dialog, you can edit the individual paragraphs that make up the text object and fill them with content.

See also:
- Object Properties
- Paragraph Properties

\section*{Object Properties}
\begin{tabular}{|c|c|c|c|}
\hline Property & Description & Value & Description \\
\hline Bottom aligned & Bottom aligned within the object's border. If this option is enabled, the object's text will be output at the lower margin of the object, or otherwise at the upper margin. For this to be possible, the paragraphs must not be larger than the object otherwise the text will be truncated as usual at the bottom or wrapped. This option is very useful e.g. if text is to be output at the lower margin of a page and the length is not known. & True False Formula & \begin{tabular}{l}
Yes \\
No \\
Formula wizard
\end{tabular} \\
\hline Rotation & Rotates the object
anticlockwise. Please note that
only TrueType fonts can be
rotated. & \[
\begin{aligned}
& \hline 0 \\
& 1 \\
& 2 \\
& 3 \\
& \text { Formula }
\end{aligned}
\] & \[
\begin{aligned}
& \hline 0^{\circ} \\
& 90^{\circ} \\
& 180^{\circ} \\
& 270^{\circ} \\
& \text { Formula } \\
& \text { wizard }
\end{aligned}
\] \\
\hline Page break & Specifies whether the object can trigger a page break. If this property is enabled, the content will be wrapped to the next page automatically if it exceeds the size of the object. This is an interesting option e.g. with text objects that are to cover several pages. With labels, the next label will only be started when all objects have been printed as a result of this option in the previous label. You might not be able to set this property if page & True False Formula & \begin{tabular}{l}
Yes \\
No \\
Formula wizard
\end{tabular} \\
\hline
\end{tabular}
breaks are not supported by the higher-level program.

\section*{Paragraph Properties}
\begin{tabular}{|c|c|c|c|}
\hline Property & Description & Value & Description \\
\hline Paragraph spacing & \(\begin{array}{ll}\text { Distance to } & \text { next paragraph } \\ \text { ("Paragraph } & \text { spacing"). You }\end{array}\) specify the distance in points: To achieve line spacing of 1.5 with a font size of 10 points, enter 5 points. Negative values are also allowed. You should always make the settings under Options > Objects > Object font. & Number Formula & Formula wizard \\
\hline Alignment & You can specify the alignment in the same way as in your text processing program. & \[
\begin{aligned}
& \hline 0 \\
& 1 \\
& 2 \\
& \text { Formula }
\end{aligned}
\] & \begin{tabular}{l}
Left \\
Centered \\
Right \\
Formula \\
wizard
\end{tabular} \\
\hline Justified & \begin{tabular}{l}
Block text is justified both right and left. This property only takes effect if the line in question is wrapped at the end of the line. In other words, the line must be longer than the available space, it must consist of more than one word and the "Line wrap" option must be enabled. \\
The last line is presented according to the "Alignment" property.
\end{tabular} & True False Formula & \begin{tabular}{l}
Yes \\
No Formula wizard
\end{tabular} \\
\hline Format & Please consider that the formatting relates to the result of the entire expression. Use the Date\$() and FStr\$() functions in the formula wizard if you only want to format part of the expression (e.g. with text and number within an expression). & True False & Dialog \\
\hline Blank optimization & The blank optimization option lets you remove unwanted blanks (leading, embedded and & True False Formula & \begin{tabular}{l}
Yes \\
No Formula
\end{tabular} \\
\hline
\end{tabular}
trailing).
Consider the following situation: In a label project, you are printing a line with the variables <SALUTATION> <FIRSTNAME> <NAME>
whereby the individual variables are separated by blanks in each case. If there is no salutation for a data record, this variable remains empty; the following blank would however be printed. First name and name would be shifted by one position (leading blank).
If the "Firstname" variable is empty, there would be two spaces (embedded blanks) between "Salutation" and "Name".
If all three variables were empty, both blanks would remain (trailing blanks). This means that the line is not empty and would therefore not be suppressed automatically.
The "Blank Optimization" option helps in such cases. It can remove leading, embedded and trailing blanks automatically. Multiple embedded blanks are reduced automatically to a single blank.
\begin{tabular}{|c|c|c|c|}
\hline Inerasable & \begin{tabular}{l}
Lines that are completely empty are suppressed automatically. In this case, the following lines move upwards. This is normally correct but can be unwanted in some cases, e.g. when filling out forms. \\
With the "Untraceable" option, the line in question remains, even if it is empty after inserting the variables.
\end{tabular} & \begin{tabular}{l}
True \\
False \\
Formula
\end{tabular} & \begin{tabular}{l}
Yes \\
No Formula wizard
\end{tabular} \\
\hline Line wrap & \begin{tabular}{l}
Specifies the behavior if the text is too long for a line. \\
So that long words are not truncated with value "1" (wrap), you can use the "Force wrap"
\end{tabular} & \[
\begin{aligned}
& \hline 0 \\
& 1 \\
& 2 \\
& 3 \\
& 4
\end{aligned}
\] & Truncate Wrap Shrink Compress Optimal fit \\
\hline
\end{tabular}
\begin{tabular}{|c|c|c|c|}
\hline & \begin{tabular}{l}
option to ensure that a break occurs after the last suitable character. \\
Value "3" (compress) reduces the character spacing and should only be used to a limited extent in order to guarantee legibility. \\
With value "4" (optimal fit) the font size is varied in such a way as to fill out the object as completely as possible.
\end{tabular} & Formula & Formula wizard \\
\hline & \begin{tabular}{ll} 
Page & With this property, \\
\hline break & \begin{tabular}{l} 
you can specify that \\
the paragraph is not \\
to be separated in the \\
\\
\\
\\
event that a page \\
break is triggered.
\end{tabular}
\end{tabular} & \begin{tabular}{l}
True False \\
Formula
\end{tabular} & Break Keep together Formula wizard \\
\hline & \begin{tabular}{ll} 
Force & If a long word cannot \\
wrap & \begin{tabular}{l} 
be wrapped, a line \\
break will be forced \\
after the last suitable \\
character.
\end{tabular} \\
&
\end{tabular} & True False Formula & \begin{tabular}{l}
Yes \\
No \\
Formula wizard
\end{tabular} \\
\hline Line spacing & \begin{tabular}{l}
Line spacing defines the distance between the paragraph's individual lines. \\
You specify the distance in points: To achieve line spacing of 1.5 with a font size of 10 points, enter 5 points. Negative values are also allowed. \\
You should always make the settings under Options > Objects \(>\) Object font.
\end{tabular} & Number Formula & Formula wizard \\
\hline
\end{tabular}

ゅ
combit List \& Label

\section*{Line Objects}

\section*{You define lines by their alignment, width and type.}

See also:
- Object Properties

\section*{Object Properties}
\begin{tabular}{|c|c|c|c|}
\hline Property & Description & Value & Description \\
\hline Alignment & Bottom aligned within the object's rectangle. If you hold down the Shift key when changing the size with the mouse, the line will be aligned either vertically or horizontally. & \[
\begin{aligned}
& \hline 0 \\
& 1 \\
& 2 \\
& 3 \\
& \text { Formula }
\end{aligned}
\] & \begin{tabular}{l}
Diagonal \} \\
Diagonal / \\
Horizontal \\
Vertical \\
Formula \\
wizard
\end{tabular} \\
\hline Width & Width of line in the unit of measure of the workspace. & Number Formula & Formula wizard \\
\hline Line type & Line type selection. & \begin{tabular}{l}
\[
\begin{aligned}
& 0,1,3, \\
& 4
\end{aligned}
\] \\
Formula
\end{tabular} & \begin{tabular}{l}
4 \\
predefined lines Formula wizard
\end{tabular} \\
\hline
\end{tabular}

ゅ
combit List \& Label

\section*{Rectangle Objects}

You define rectangles by their border, rounding and shadow.

See also:
, Object Properties

\section*{Object Properties}
\begin{tabular}{|c|c|c|c|}
\hline Property & Description & Value & Description \\
\hline \multirow[t]{3}{*}{Border} & Here you specify whether the rectangle is to have a border. & \[
\begin{aligned}
& \hline 0 \\
& 1 \\
& \text { Formula }
\end{aligned}
\] & Transparent Pattern/color Formula wizard \\
\hline & Color Color of the border. & & \\
\hline & \begin{tabular}{ll} 
Width & \begin{tabular}{l} 
Width of the border in \\
the unit of measure of \\
the workspace
\end{tabular}
\end{tabular} & Number Formula & Formula wizard \\
\hline Rounding & \begin{tabular}{l}
Rounding factor for the corners of the rectangle in \% of the short edge of the rectangle. \\
\(0 \%\) means square cornered; \(100 \%\) means: the short edge of the rectangle is completely round (elliptical).
\end{tabular} & Number Formula & Formula wizard \\
\hline \multirow[t]{4}{*}{Shadow} & Here you define whether the rectangle is to have a shadow. & \[
\begin{aligned}
& \hline 0 \\
& 1 \\
& \text { Formula }
\end{aligned}
\] & Transparent Pattern/color Formula wizard \\
\hline & Pattern Shadow pattern. & & \\
\hline & Color Shadow color. & & \\
\hline & \begin{tabular}{ll} 
Width & \begin{tabular}{l} 
Width of the shadow \\
in the unit of measure \\
of the workspace.
\end{tabular}
\end{tabular} & Number Formula & Formula wizard \\
\hline
\end{tabular}

ゅ
combit List \& Label

\section*{Circles and Ellipses}

\section*{You define circles and ellipses by their borders and filling.}

See also:
, Object Properties

\section*{Object Properties}
\begin{tabular}{|c|c|c|c|}
\hline Property & Description & Value & Description \\
\hline Pie (Flat) & This property lets you specify that the ellipse is always displayed as a circle (centered in the object rectangle). & True False Formula & \begin{tabular}{l}
Yes \\
No \\
Formula wizard
\end{tabular} \\
\hline Border & Here you define whether the ellipse is to have a border. & \[
\begin{aligned}
& \hline 0 \\
& 1 \\
& \text { Formula }
\end{aligned}
\] & Transparent Pattern/color Formula wizard \\
\hline & Color Color of the border. & & \\
\hline & \begin{tabular}{ll} 
Width & \begin{tabular}{l} 
Width of the shadow, \\
in the unit of measure \\
of the workspace.
\end{tabular}
\end{tabular} & Number Formula & Formula wizard \\
\hline
\end{tabular}

ゅ
combit List \& Label

\section*{Picture Objects}

Picture objects are used to display the content of fixed files or variables.
The following formats are available: WMF, EMF, BMP, DIB, PCX, SCR, TIFF, GIF, JPEG, PCD, PNG and ICO. As a general rule, you should use the RGB color space (not CYMK). List \& Label itself supports transparency in PNG files by using the corresponding Windows functions. In our experience the majority of printer drivers do not support transparency so that reports with e.g. partly transparent PNG files should thoroughly be tested on the actual hard-software combination. If that is not possible we recommend doing without the alpha channel.
If the image is held in a file or a variable, you can select the data source by double-clicking the object.

See also:
, Object Properties

\section*{Object Properties}
\begin{tabular}{|c|c|c|c|c|}
\hline Property & \multicolumn{2}{|l|}{Description} & Value & Description \\
\hline Data source & \multicolumn{2}{|l|}{Select the method of determining the image data source, i.e. via file name, formula or variable name.} & File name Formula Variable & Formula wizard \\
\hline & File name & Fixed file name: is evaluated if you have selected "File name" as the data source property. You then select the image file that you want by means of the file selection dialog. In this dialog, you can also insert the picture into the project by enabling the ("Embed image in project file" checkbox option. This option copies the image to the project thus making it available even without the external file. In this case, "(embedded)" will be shown as the file name. & 0 & \begin{tabular}{ll} 
Open & file \\
dialog
\end{tabular} \\
\hline & & \begin{tabular}{ll} 
Relative & \begin{tabular}{l} 
The path is \\
path \\
relative to \\
the project \\
path.
\end{tabular} \\
&
\end{tabular} & True False & \[
\begin{aligned}
& \text { Yes } \\
& \text { No }
\end{aligned}
\] \\
\hline & Formula & If you have selected "Formula" as the data source property, the file name is derived from a formula. The formula must return a & Formula & Formula wizard \\
\hline
\end{tabular}
\begin{tabular}{|c|c|c|c|c|}
\hline & & \begin{tabular}{l}
"picture" value type. \\
You can also enter a valid file \\
name. \\
However, this must first be converted to the "picture" type with the Drawing() function.
\end{tabular} & & \\
\hline & Variable & \begin{tabular}{l}
If you have selected "Variable" as the data source property, the file name is taken from a variable. \\
Select the variable that you want from the drop-down box. The drop-down box lists all "picture" variables defined in your application.
\end{tabular} & Variable & \\
\hline & Properties & Depending on your application, a dialog may open up at this point to allow you to define more properties. & 9090 & Opens dialog \\
\hline & Save as JPEG & Embed the image as a JPEG file if possible (not all file types support this option). & True False Formula & \begin{tabular}{l}
Yes \\
No \\
Formula wizard
\end{tabular} \\
\hline Original size & & Specifies whether the picture is to be drawn in the original size (if it can be determined), or if the size of the picture stame is to be adjusted to fit. & True False Not defined & \begin{tabular}{l}
Yes \\
No \\
Not \\
defined
\end{tabular} \\
\hline & Keep proportions & With this option you can specify whether the height/width proportions are to be kept (True) when inserting the picture or if the picture's frame is to be adjusted, possibly leading to distortion (False). & True False Formula & \begin{tabular}{l}
Yes \\
No \\
Formula \\
wizard
\end{tabular} \\
\hline & Alignment & Describes how the image is to be arranged in the available area. & \[
\begin{aligned}
& \hline 0 \\
& 1
\end{aligned}
\] & \begin{tabular}{l}
Centered \\
Next \\
each ot
\end{tabular} \\
\hline
\end{tabular}
(tiled)
Left top
Left
bottom
Right top
Right bottom
Left
Right
Top
Bottom

ゅ
combit List \& Label

\section*{Barcode Objects}
 serial numbers and many other purposes. A barcode normally consists of a series of bars and spaces in different thicknesses whereby, depending on the code, the spaces between the bars also hold information.

See also:
- Object Properties
- Barcode Content
- Supported Barcode Formats

\section*{Object Properties}
\begin{tabular}{|c|c|c|c|}
\hline Property & Description & Value & Description \\
\hline Bar width & The narrowest bar width in SCM units ( \(1 / 1000 \mathrm{~mm}\) ). \(0=\) automatic adjustment. Not supported by all barcodes. & Formula & Formula wizard \\
\hline & Orientation Orientation of the barcode within the object's frame & \[
\begin{aligned}
& \hline 0 \\
& 1 \\
& 2 \\
& \text { Formula }
\end{aligned}
\] & \begin{tabular}{l}
Left \\
Centered Right Formula wizard
\end{tabular} \\
\hline Bar color & Color of the selected barcode. & & \\
\hline Rotation & Rotates the object anticlockwise & \[
\begin{aligned}
& \hline 0 \\
& 1 \\
& 2 \\
& 3 \\
& \text { Formula }
\end{aligned}
\] & \[
\begin{aligned}
& 0^{\circ} \\
& 90^{\circ} \\
& 180^{\circ} \\
& 270^{\circ} \\
& \text { Formula } \\
& \text { wizard }
\end{aligned}
\] \\
\hline Optimum size & Sets the optimum size for the barcode. This property has an effect when the size changes and can be used with the following formats: EAN 13 (all), DPLeitcode, DP-Identcode, German Parcel, Postnet and FIM. & True False Formula & \begin{tabular}{l}
Yes \\
No \\
Formula wizard
\end{tabular} \\
\hline Font & Font for the barcode text. Is only evaluated if the "Show text" property is enabled. & (7) & Font dialog \\
\hline Bar width ratio & The relationship of the different widths of bars or spaces. Not supported by all barcodes. & Formula & Formula wizard \\
\hline Show text & This property lets you specify whether the content of the barcode is to be printed as text. & True False Formula & \begin{tabular}{l}
Yes \\
No Formula wizard
\end{tabular} \\
\hline
\end{tabular}

\section*{Barcode Content}

The content dialog for the barcode object lets you define the barcode more precisely.
§ Choose "Text" if you want to print fixed text as a barcode. Enter the value to be printed in the first part of the input field. Choose the type of barcode that you want in the second field.
\(\S\) There are additional configuration options for many barcodes, e.g. Maxicode. You can edit these options in a further dialog.
§ Choose "Formula" if you want to use a formula as a barcode, and define a valid formula expression with the "Edit" button. The formula must return a "barcode" value type. You can, of course, also enter variables. However, they must first be converted to the "barcode" type with the Barcode() function.
§ Choose "Variable" if you want to print a variable as a barcode. In the drop-down box, you can choose from all available barcode variables.

\section*{Supported Barcode Formats}

List \& Label supports a range of barcode formats. There is normally no need for special printers, fonts etc. \(\rangle\) the barcodes are printed directly by List \& Label.

See also:
- Overview of the General 1-D Codes
- Overview of the General 2-D Codes
- Overview of Post Codes (1-D and 2-D Codes)

\title{
Overview of the General 1-D Codes
}
\begin{tabular}{|c|c|}
\hline Name & Formats and permitted characters \\
\hline \[
\begin{aligned}
& \text { GTIN-13, EAN- } \\
& \text { 13, UCC-13, } \\
& \text { JAN-13 }
\end{aligned}
\] & \begin{tabular}{l}
Formats: \\
cc|nnnnn|aaaaa \\
(normal EAN13) \\
cc|nnnnn|aaaaa|xx \\
(EAN13 for newspap \\
cc|nnnnn|aaaaa|xxxxx \\
ppp|nnnn|aaaaa \\
(normal EAN13) \\
ppp|nnnn|aaaaa|xx \\
(EAN13 for newspar \\
ppp|nnnn|aaaaa|xxxxx \\
(EAN13 for books, "Bookland \\
with \\
cc = country code \\
ppp \(=\) product code \\
nnnn, nnnnn \(=\) country cod \\
aaaaa \(=\) article code \\
| = character code chr(124) \\
xx, xxxxx = supplemental c
\end{tabular} \\
\hline & \begin{tabular}{l}
Permitted characters: [0-9] \\
The check digit will be calculated and attached autc character is 7 bar-widths wide, a code should have a mi \((12 * 7+11) * 0.3 \mathrm{~mm}=2.85 \mathrm{~cm}\). \\
Ideal size (bar symbol) nominal size SC2: \\
Width: 31.4 mm , Height: 24.5 mm \\
Minimum offset that should be kept free around the symb left: 3.6 mm , top: 0.3 mm , right: 2.3 mm , bottom: 0.0 mm The text may partially exceed this area.
\end{tabular} \\
\hline EAN-14, UCC-14 & Format: nnnnnnnnnnnnnnn (14 digits) Permitted characters: [0-9] \\
\hline GTIN-8, EAN-8, UCC-8, JAN-8 & \begin{tabular}{l}
Formats: nnnnnnn, nn|nnnnn \\
(|= character code chr(124)) \\
Permitted characters: [0-9] \\
Each character is also 7 bar widths wide, a code shou minimum width of \((8 * 7+11) * 0.3 \mathrm{~mm}=2.01 \mathrm{~cm}\). \\
Ideal size (bar symbol) nominal size SC2: \\
\(\mathrm{dx}: 22.1 \mathrm{~mm}\), dy: 19.9 mm \\
Minimum offset that should be kept free around the symb left: 2.3 mm , top: 0.3 mm , right: 2.3 mm , bottom: 0.0 r otherwise 0.3 mm ) \\
The text may partially exceed this area.
\end{tabular} \\
\hline UPC-A & Format: c|nnnnn|aaaaa, cnnnnnaaaaa \\
\hline
\end{tabular}
\begin{tabular}{ll} 
with \(\quad\) & \(\mathrm{c}=\) number system \\
& nnnnn \(=\) company code \\
& aaaaa = article code \\
& \(\mid=\) character code chr(124)
\end{tabular}

Permitted characters: [0-9]
The check digit will be calculated and attached autc character is also 7 bar widths wide, a code should then \(h\) width of \((13 * 7+6) * 0.3 \mathrm{~mm}=2.88 \mathrm{~cm}\).
\begin{tabular}{|c|c|}
\hline UPC-E &  \\
\hline 2-of-5 Industrial & ```
Format: any
Permitted characters: [0-9]
A code is (14*number of characters+18) bar widths wide.
``` \\
\hline \[
\begin{aligned}
& \text { 2-of-5 } \\
& \text { Interleaved (ITF) }
\end{aligned}
\] & \begin{tabular}{l}
Format: any, must have an even number of characters Permitted characters: [0-9] \\
A code is (9*number of characters+9) bar widths wide.
\end{tabular} \\
\hline 2-of-5 Matrix & ```
Format: any
Permitted characters: [0-9]
A code is (10*number of characters+18) bar widths wide.
``` \\
\hline 2-of-5 Datalogic & ```
Format: any
Permitted characters: [0-9]
A code is (10*number of characters+11) bar widths wide.
``` \\
\hline Codabar & \begin{tabular}{l}
Format: fnnnnnf \\
Permitted characters: \(\mathrm{f}=\) frame code [A-D], \(\mathrm{n}=[0-9]\), \([-\$:\) Every character is either \(2 * 3+6 * 1\) (characters ' 0 '..' \({ }^{\prime}\) ', '-', (characters ':', '/', '.', '+', 'A'..'D') bar widths wide. The ch frame code will not be printed with the text.
\end{tabular} \\
\hline CODE11 & \begin{tabular}{l}
Format: any \\
Permitted characters: [0-9],[-] \\
Code 11 has, depending on the length, 1 or 2 check dig calculates only 1 instead of 2 check digits if the length maximum of 10 characters.
\end{tabular} \\
\hline Code39, 3-of-9, Alpha39 & \begin{tabular}{l}
Format: any \\
Permitted characters: [A-Z], [0-9], [-./\$\%+*]
\end{tabular} \\
\hline Extended code 39 & \begin{tabular}{l}
Format: any \\
Permitted characters: any \\
The expanded code can be activated by a combination code: for example: '+A' -> 'a'. Every character is 16 baı
\end{tabular} \\
\hline
\end{tabular}
text has (16*number of characters 1) bars.
\begin{tabular}{|c|c|}
\hline Code 39 with CRC & \begin{tabular}{l}
Format: any \\
Permitted characters: [A-Z], [0-9], [-./\$\%+*]
\end{tabular} \\
\hline Code 93 (simple and extended) & \begin{tabular}{l}
Code 93 is an extension of Code 39 and covers the com of the ASCII Allowed characters, including NULL. This mus as chr\$(255). \\
It contains two check digits that are automatically gene Label. \\
The characters consist of 9 bar widths, that each havi spaces. There are two options for the extended code: \\
a) transfer of the shift character from the host program as \\
b) transfer of the desired character, L\&L adds the a character.
\end{tabular} \\
\hline Code128 & \begin{tabular}{l}
Format: any \\
Permitted characters: any \\
Determine the code set that should be used: \\
Use one of the following codes as a start character: \\
chr\$(135) start with code A \\
chr\$(136) start with code B \\
chr\$(137) start with code C \\
In order to switch between different code sets within tr \\
can use the usual control characters: \\
Starting from code A to \\
B: chr\$(132) \\
C: chr\$(131) \\
Starting from code B to \\
A: chr\$(133) \\
C: chr\$(131) \\
Starting from code C \\
A: chr\$(133) \\
B: chr\$(132) \\
Example: \\
<Subset B> "RL" <Subset C> "04432476" <Subset B> "( \\
Barcode(Chr\$(136)+"RL"+chr\$(131)+"04432476"+chr\$(: \\
"Code 128")
\end{tabular} \\
\hline \[
\begin{aligned}
& \text { GS1 128, } \\
& \text { EAN128 }
\end{aligned}
\] & \begin{tabular}{l}
The special FNC1 character following the start character the EAN128 code. Special characters must be replaced as \\
NUL: chr\$ (255) \\
FNC1: chr\$ (254) \\
FNC2: chr\$ (253) \\
FNC3: chr\$ (252) \\
FNC4: chr\$ (251)
\end{tabular} \\
\hline
\end{tabular}

GS1 DataBar Format: nnnnnnnnnnnnn (13 digits)
(Limited, Permitted characters: [0-9] Stacked, Stacked Omnidirectional, Stacked

\section*{Truncated)}
\begin{tabular}{ll}
\hline \begin{tabular}{l} 
GS1 DataBar \\
Expanded
\end{tabular} & \begin{tabular}{l} 
Format: Data begins with Al. Max. 74 numeric/4] \\
characters \\
Permitted characters: any
\end{tabular} \\
\hline IM (4CB/4- & \begin{tabular}{l} 
Intelligent Mail Barcode (US Postal Services). Alternative r \\
Solution or 4-State Customer Barcode). \\
Format: 20, 25, 29 or 31 digits \\
Permitted characters: [0-9]
\end{tabular} \\
\hline ISBN & \begin{tabular}{l} 
Format: nnn|nnnnnnnnn (12-digit, no check digit) \\
Permitted characters: [0-9]
\end{tabular} \\
\hline MSI & \begin{tabular}{l} 
Format: any \\
Permitted characters: [0-9], [A-F]
\end{tabular} \\
\hline Pharma-Zentral- & \begin{tabular}{l} 
Format: nnnnnn (6 digits) \\
Permitted characters: [0-9]
\end{tabular} \\
\hline Nummer & \begin{tabular}{l} 
Format: \(\{n n\} n n n n n n n n n n n n n n n ~(17 ~ o r ~ 19 ~ d i g i t s) ~\) \\
Permitted characters: [0-9]
\end{tabular} \\
\hline
\end{tabular}

\title{
Overview of the General 2-D Codes
}
\begin{tabular}{|c|c|}
\hline Name & Formats and permitted characters \\
\hline Aztec & Format: any Permitted characters: any \\
\hline Datamatrix & \begin{tabular}{l}
In order to enter non-printable characters (binary data) in the barcode text, they must be packaged in a special string. The data is inserted using the \(\sim d N N N\) string, whereby NNN stands for ASCII-Code. \\
Example: \\
DEA~d065~d015~d000~d247~d220 \\
(~d065 stands for "A") \\
To represent an EAN data matrix, you can code the special FNC1 character as \(\sim 1\). \\
The Datamatrix symbology uses the ECC 200 error correction code. Further options like size, module format and encoding can be set in the barcode's options dialog. Click the options symbol in the barcode content dialog to enter this dialog.
\end{tabular} \\
\hline PDF417 & \begin{tabular}{l}
Can display all available and non-printable characters. \\
In order to enter non-printable characters (binary data) in the barcode text, they must be packaged in a special string. The data is inserted using the "\{binary:nn\}" string, whereby nn stands for any sequence of twocharacter hexadecimal numbers. This is especially important if Maxicodes are to be created according to UPS specifications; the special characters needed for this can be entered in this way: \\
In order to pack a Null and a Backspace (BS) character in the data, use "\{binary:0008\}" (corresponds to "\{binary:00\} \{binary:08\}"). Use "Hallo\{binary:OdOa\}World" to include a line break.
\end{tabular} \\
\hline
\end{tabular}
QR Code Format: any

Permitted characters: all characters
In order to enter non-printable characters (binary data) in the barcode text, they must be packaged in a special string. The data is inserted using the \(\sim\) dNNN string, whereby NNN stands for ASCII-Code.
Example: ~d065 stands for the letter "A".

\section*{Overview of Post Codes (1-D and 2-D Codes)}
\begin{tabular}{|c|c|}
\hline Name & Formats and permitted characters \\
\hline DP-Identcode & \begin{tabular}{l}
Formats: nn.nnnnnn.nnn, nn.nnnnn.nnnn, nn.nnnn.nnnnn, nn.nnn.nnnnnn \\
Permitted characters: [0-9] \\
A code is ( \(9 *\) number of characters+9) bar widths wide. \\
Width: \(32.0 \mathrm{~mm}-585 \mathrm{~mm}\) (at least 5 mm light zone right and left). Height: 25 mm . Check digit is calculated automatically; relation: 4:9; special 2 of 5 IL \(\geqslant\) code.
\end{tabular} \\
\hline DP-Leitcode & \begin{tabular}{l}
Format: nnnnn.nnn.nnn.nn \\
Permitted characters: [0-9] \\
A code is ( 9 *number of characters+9) bar widths wide. \\
Width: \(37.25 \mathrm{~mm}-67-5 \mathrm{~mm}\) (at least 5 mm light zone right and left). Height: 25 mm . \\
Check digit is calculated automatically; relation: 4:9; special \(\geqslant 2\) of 5 IL \(\geqslant\) code.
\end{tabular} \\
\hline FIM & \begin{tabular}{l}
Formats: A, B, C \\
Permitted characters: [A-C] \\
Minimum size: \(1 / 2^{\prime \prime}\) * 5/8". \\
The FIM barcode is always printed in the size specified by the US Postal Office. This means that it might extend beyond the available object border.
\end{tabular} \\
\hline German Parcel & \begin{tabular}{l}
Format: any, must have an even number of characters. \\
Permitted characters: [0-9] \\
A code is (14*number of characters+18) bar widths wide. \\
Relation: 1:2
\end{tabular} \\
\hline Japanese Postcode & \begin{tabular}{l}
Japanese Postcode. \\
Format: Postcode as nnn-nnnn, then max. 13 \\
character address \\
Permitted characters: \(n=[0-9]\), address \(=[A-\) Z], [0-9], [-]
\end{tabular} \\
\hline
\end{tabular}
\begin{tabular}{|c|c|}
\hline Maxicode & \begin{tabular}{l}
Can display all available and non-printable characters. \\
In order to enter non-printable characters (binary data) in the barcode text, they must be packaged in a special string. The data is inserted using the "\{binary:nn\}" string, whereby nn stands for any sequence of twocharacter hexadecimal numbers. This is especially important if Maxicodes are to be created according to UPS specifications; the special characters needed for this can be entered in this way. \\
Example: in order to pack a Null and a Backspace (BS) character in the data, use " \{binary:0008\}" (corresponds to "\{binary:00\} \{binary:08\}"). \\
Example: use "Hallo\{binary:OdOa\}world" to include a line break.
\end{tabular} \\
\hline Maxicode/UPS & \begin{tabular}{l}
Format: Formatting according to UPS specifications \\
Permitted characters: all characters
\end{tabular} \\
\hline Postnet & \begin{tabular}{l}
Formats: nnnnn, nnnnn-nnnn, nnnnn-nnnnnn Permitted characters: [0-9] \\
Minimum size: \(1.245^{*}\) * 4/16" (10-digits). \\
Bar distance at least 1/24". \\
Error correction digit will be appended automatically. \\
This bar code is automatically printed in the right size if the object is larger than the maximum size of the barcode.
\end{tabular} \\
\hline RM4SCC, KIX® Royal Mail with CRC & \begin{tabular}{l}
Permitted characters: [A-Z], [0-9], [a-z] \\
Permitted characters: [A-Z], [0-9], [a-z] \\
Format: Either just the post code is coded (e.g. LU17 8XE) or the post code with a supplementary "Delivery Point" (e.g. LU17 8XE 2B). The maximum number of characters that can be used is therefore limited to 9.
\end{tabular} \\
\hline
\end{tabular}

ゅ
combit List \& Label

\section*{Report Container Object}
- A report container can hold multiple table objects, cross tab objects and chart objects.

See also:
- Object Properties
- Element Properties

\section*{Object Properties}
\begin{tabular}{|c|c|c|c|}
\hline Property & Description & Value & Description \\
\hline Page break before & If the condition returns "True", a page break will be triggered before printing the report container. & True False Formula & \begin{tabular}{l}
Break \\
No break Formula wizard
\end{tabular} \\
\hline Default font & Default font setting for the elements. & (7) & Font dialog \\
\hline \multirow[t]{2}{*}{Column count} & \begin{tabular}{l}
Number of columns in the container. \\
Note: \\
the \\
"LL.CurrentTableColumn" field returns the index of the current column.
\end{tabular} & \[
\begin{aligned}
& \hline 1 \\
& 2 \\
& 3 \\
& 4 \\
& 5 \\
& \text { Formula }
\end{aligned}
\] & \begin{tabular}{l}
1-column \\
2-column \\
3-column \\
4-column \\
5-column \\
Formula \\
wizard
\end{tabular} \\
\hline & Distance \begin{tabular}{lll} 
Spacing of & of \\
columns \\
container.
\end{tabular}\(\quad\)\begin{tabular}{c} 
the
\end{tabular} & Number Formula & Formula wizard \\
\hline
\end{tabular}

\section*{Element Properties}

To display the properties of the elements, mark the element in the "Report Structure" tool window. Also see chapter Common Object Properties.
\begin{tabular}{|c|c|c|c|}
\hline Property & Description & Value & Description \\
\hline Name & Name of the element (stored in the "LL.CurrentContainerltem" variable). & Name & \\
\hline Separator ticks (only tables) & To align (sub) tables exactly with one another, additional tick marks can be shown on the ruler. & True False & \[
\begin{aligned}
& \text { Yes } \\
& \text { No }
\end{aligned}
\] \\
\hline Spacing (only charts and crosstabs) & Distance of the object (including fr container's print area. & frame) from & \(m\) the report \\
\hline Distance before & Distance from the previous element (not required if the element starts at the beginning of the page. & Number Formula & Formula wizard \\
\hline Output height (only charts) & Height of the object (including frame). & & \\
\hline Sort order (only tables and charts) & Sort order for the table. All available sort orders are offered in a selection list (stored in the "LL.SortStrategy" variable). & List & \\
\hline Pagebreak before & If the condition returns "True", a page break will be triggered before printing the object. If you have multiple multi-column objects, a page break is triggered automatically after an object if the column counts for the objects are different (e.g. 2column table followed by a 3 column table) and if there would & \begin{tabular}{l}
True \\
False \\
Formula
\end{tabular} & Break
No break Formula wizard \\
\hline
\end{tabular}
\begin{tabular}{|c|c|c|c|}
\hline & be insufficient room for the following object. & & \\
\hline \multirow[t]{4}{*}{Column count} & \begin{tabular}{l}
Number of columns in the object. \\
If you have multiple multicolumn objects, a page break is triggered automatically after an object if the column counts for the objects are different (e.g. 2column table followed by a 3column table) and if there would be insufficient room for the following object. \\
Note: \\
"LL.CurrentTableColumn" field returns the index of the current column.
\end{tabular} & \[
\begin{aligned}
& \hline 0 \\
& 1 \\
& 2 \\
& 3 \\
& 4 \\
& 5 \\
& \text { Formula }
\end{aligned}
\] & Container default 1-column 2-column 3-column 4-column 5-column Formula wizard \\
\hline & \begin{tabular}{llll} 
Distance & \begin{tabular}{l} 
Spacing \\
columns \\
object.
\end{tabular} & \begin{tabular}{l} 
of \\
in
\end{tabular} & the
\end{tabular} & Number Formula & Formula wizard \\
\hline & \begin{tabular}{ll} 
Column & A column break \\
break & will be triggered if \\
condition & the result of the \\
(only & condition is "True". \\
tables) & \\
\hline
\end{tabular} & True False Formula & \begin{tabular}{l}
Yes \\
No Formula wizard
\end{tabular} \\
\hline & \begin{tabular}{ll} 
Column & \begin{tabular}{l} 
A column break \\
break \\
before
\end{tabular} \\
\begin{tabular}{l} 
will be performed \\
before the object \\
is output.
\end{tabular}
\end{tabular} & True False Formula & \begin{tabular}{l}
Yes \\
No \\
Formula wizard
\end{tabular} \\
\hline
\end{tabular}

ゅ
combit List \& Label

\section*{Table Objects}

Table objects are elements of the report container.
See also:
- Object Properties
- Line Properties
- Group Line Properties
- Column Properties

\section*{Object Properties}

\section*{Also see chapters Common Object Properties and Report Container Object.}

only drawn as far as the last data line. This property is only available with "fixed size" tables.
\begin{tabular}{|c|c|c|c|}
\hline Background & \multicolumn{3}{|l|}{Color and type of the background of the table. (Only available in the report container for certain applications).} \\
\hline Default frame & Default value for the table's frame. & & \\
\hline Default font & Default value for the table's font (not available for sub-tables). & ( A & Font dialog \\
\hline Pagebreak condition & \begin{tabular}{l}
With this property, you can specify a condition that causes a page break after a data line as soon as the condition is met. \\
If "Pagebreak condition" is set to -True \({ }^{2}\), a page break will be triggered after each line. \(\langle\) False specifies that a page break is only to be triggered when necessary.
\end{tabular} & True False Formula & \begin{tabular}{l}
Yes \\
No \\
Formula wizard
\end{tabular} \\
\hline \multirow[t]{4}{*}{Column count} & Number of columns in the table. Note: "LL.CurrentTableColumn" field returns the index of the current column. & \[
\begin{aligned}
& \hline 0 \\
& 1 \\
& 2 \\
& 3 \\
& 4 \\
& 5 \\
& \text { Formula }
\end{aligned}
\] & Container default 1-column 2-column 3-column 4-column 5-column Formula wizard \\
\hline & \begin{tabular}{ll} 
Distance & \begin{tabular}{l} 
Spacing of columns \\
in the table.
\end{tabular}
\end{tabular} & Number Formula & Formula wizard \\
\hline & \begin{tabular}{ll} 
Column & If the result is \\
break & "True" when a data \\
condition & \begin{tabular}{l} 
line is output, a \\
\\
\\
\\
column break will \\
be triggered.
\end{tabular}
\end{tabular} & True False Formula & \begin{tabular}{l}
Break \\
No break Formula wizard
\end{tabular} \\
\hline & \begin{tabular}{ll} 
Column & A column break will \\
break & be performed \\
before & \begin{tabular}{l} 
before the object is \\
output.
\end{tabular}
\end{tabular} & True False Formula & \begin{tabular}{l}
Break \\
No break Formula wizard
\end{tabular} \\
\hline Pagebreak before & \begin{tabular}{l}
If the condition returns "True", a page break will be triggered before printing the object. \\
If you have multiple multi-column
\end{tabular} & True False Formula & \begin{tabular}{l}
Break \\
No break Formula wizard
\end{tabular} \\
\hline
\end{tabular}
objects, a page break is triggered automatically after an object if the column counts for the objects are different (e.g. 2-column table followed by a 3-column table) and if there would be insufficient room for the following object.
\begin{tabular}{|c|c|c|c|c|}
\hline \multicolumn{5}{|l|}{Data lines} \\
\hline & Suppress & \begin{tabular}{l}
When you enable the \\
"Data lines.Suppress" \\
object property in tables, all data lines are completely \\
suppressed. This option \\
particularly useful in combination with the "Force Sums" option. The latter option specifies that totals are also calculated when a data line is not printed. By combining both options and using groups and sum variables, you can print more interesting statistics.
\end{tabular} & True False Formula & \begin{tabular}{l}
Yes \\
No \\
Formula wizard
\end{tabular} \\
\hline & Force sums & Sum variables are calculated even if you suppress data lines. & True False Formula & \begin{tabular}{l}
Yes \\
No \\
Formula wizard
\end{tabular} \\
\hline & Zebra pattern & The "Zebra pattern" option in the "Data line" field specifies whether data lines are to have alternating background colors. This can improve readability, especially with & & \\
\hline
\end{tabular}
\begin{tabular}{|c|c|c|c|c|}
\hline & \multirow[b]{2}{*}{Keep rows together} & large tables. & & \\
\hline & & In the event of a page break, data lines are kept together as far as possible, i.e. they are printed together on the next page. & True False Formula & Yes No Formula wizard \\
\hline \multicolumn{5}{|l|}{Footer lines} \\
\hline & Keep rows together & In the event of a page break, footer lines are kept together as far as possible, i.e. they are printed together on the next page. & True False Formula & Yes No Formula wizard \\
\hline \multicolumn{5}{|l|}{Group footer lines} \\
\hline & Also empty groups & Output group footer lines even if the groups are empty. & True False Formula & \begin{tabular}{l}
Yes \\
No \\
Formula wizard
\end{tabular} \\
\hline & Keep rows together & In the event of a page break, group footer lines are kept together as far as possible, i.e. they are printed together on the next page. & True False Formula & \begin{tabular}{l}
Yes \\
No \\
Formula wizard
\end{tabular} \\
\hline \multicolumn{5}{|l|}{Group header lines} \\
\hline & Keep together & Where possible, a group header line will not be separated from the following data line because of a page break. & True False Formula & \begin{tabular}{l}
Yes \\
No \\
Formula wizard
\end{tabular} \\
\hline & Keep rows together & In the event of a page break, group header lines are kept together as far as possible, i.e. they are printed & True False Formula & Yes No Formula wizard \\
\hline
\end{tabular}
together on the next page.

\section*{Line Properties}
\begin{tabular}{|c|c|c|c|}
\hline Property & Description & Value & Description \\
\hline Display in Designer & With this property, you can hide lines in the workspace this is very useful if you have a lot of line definitions. & True False Formula & \begin{tabular}{l}
Yes \\
No \\
Formula wizard
\end{tabular} \\
\hline Default font & You can set the default font for the entire table row. Newly inserted columns are created with this font. & (4] & Font dialog \\
\hline Ignore empty lines & Ignore lines where all fields are empty. & \begin{tabular}{l}
True \\
False \\
Formula
\end{tabular} & \begin{tabular}{l}
Yes \\
No \\
Formula \\
wizard
\end{tabular} \\
\hline Spacing & Here you define the top, bottom, right and left spacing of the line. The "top" or "bottom" values result in a corresponding space between the individual table rows. With the "left" and "right" spacing values, you can specify the margin in relation to the table object, i.e. you can indent lines or columns. & Formula & Formula wizard \\
\hline
\end{tabular}

\section*{Group Line Properties}
\begin{tabular}{|c|c|c|c|}
\hline Property & Description & Value & Description \\
\hline Group sums & \begin{tabular}{l}
The selected sum variables are set to "0" when the condition for the group line is met. \\
This setting is useful to create so-called group sub-totals, for example to add up the prices of all articles in a particular article group.
\end{tabular} & ...) & Dialog \\
\hline Group by & This expression represents a key. Whenever the result of the expression changes, a group change is triggered. & Formula & Formula wizard \\
\hline Repeat as header & Outputs the group header again after a page break. & True False Formula & \begin{tabular}{l}
Yes \\
No Formula wizard
\end{tabular} \\
\hline Break before & \begin{tabular}{l}
A page break will be triggered before outputting the group header, i.e. each group begins on a new page. If several group lines are printed with this option at the same time, they will appear one after the other on the new page. \\
The RemainingTableSpace() function lets you perform conditional breaks before outputting group headers, e.g. "If there are only 2 inches left on the page then break before outputting the object". You will find more information in chapter Overview of Functions.
\end{tabular} & True False Formula & \begin{tabular}{l}
Yes \\
No Formula wizard
\end{tabular} \\
\hline Break after & A page break will be triggered after outputting the group footer lines, i.e. each group & True False Formula & \begin{tabular}{l}
Yes \\
No \\
Formula
\end{tabular} \\
\hline
\end{tabular}

\section*{Column Properties}

The column properties correspond with the properties of the respective object type, with some table-related restrictions.
Text and RTF text columns are special cases. A column property lets you switch between these two text variations later on. The property list also changes accordingly depending on this property.
Also see chapter Common Object Properties.
\begin{tabular}{|c|c|c|c|}
\hline Property & Description & Value & Description \\
\hline Options (only with some field types) & Opens the "Content" dialog for the relevant object type. & 匀 & Content dialog \\
\hline Drilldown links & Opens the dialog for editing the drilldown links. A drilldown report can also be started from the preview. & (9) & Drilldown dialog \\
\hline Link & \begin{tabular}{l}
Link that is accessed when you click the object (in the real data preview, with PDF or HTML export). \\
Example: \\
file://c:\users\public\x.log or URL
\end{tabular} & Link Formula & Formula wizard \\
\hline Save as
JPEG
(only with
pictures) & Embed the image as a JPEG file if possible (not all file types support this option). & True False Formula & \begin{tabular}{l}
Yes \\
No \\
Formula wizard
\end{tabular} \\
\hline Rotation & Rotates the object anticlockwise. For example, you can rotate the column title or barcode by \(90^{\circ}\) with this function. & \[
\begin{aligned}
& \hline 0 \\
& 1 \\
& 2 \\
& 3 \\
& \text { Formula }
\end{aligned}
\] & \[
\begin{aligned}
& 0^{\circ} \\
& 90^{\circ} \\
& 180^{\circ} \\
& 270^{\circ} \\
& \text { Formula } \\
& \text { wizard }
\end{aligned}
\] \\
\hline Format & Please consider that the formatting relates to the result of the entire expression. Use the & True False & Dialog \\
\hline
\end{tabular}

Date\$() and FStr\$() functions in the formula wizard if you only want to format part of the expression (e.g. with text and number within an expression).
\begin{tabular}{|c|c|c|c|}
\hline Bar width (only with barcodes) & The narrowest bar width in SCM units ( \(1 / 1000 \mathrm{~mm}\) ). \(0=\) automatic adjustment. Not supported by all barcodes. & Formula & Formula wizard \\
\hline & Orientation \begin{tabular}{l} 
Orientation of the \\
barcode within the \\
object's frame
\end{tabular} & \[
\begin{aligned}
& \hline 0 \\
& 1 \\
& 2 \\
& \text { Formula }
\end{aligned}
\] & \begin{tabular}{l}
Left \\
Centered \\
Right Formula
\end{tabular} \\
\hline Bar color
(only with
barcodes) & Color of the barcode. & & \\
\hline Bar \begin{tabular}{l} 
width \\
ratio \\
with \\
(only \\
barcodes)
\end{tabular} & The relationship of the different widths of bars or spaces. Not supported by all barcodes. & Formula & Formula wizard \\
\hline Show text (only with barcodes) & This property lets you specify whether the content of the barcode is to be printed as text. & \begin{tabular}{l}
True \\
False Formula
\end{tabular} & \begin{tabular}{l}
Yes \\
No Formula wizard
\end{tabular} \\
\hline Background & The background of the columns. & & \\
\hline Text format & For presenting the text column differently. & True False & Normal text RTF Text \\
\hline Alignment (pictures) & Describes how the image is to be arranged in the available area. & \[
\begin{aligned}
& \hline 0 \\
& 1 \\
& 2 \\
& 3 \\
& 4 \\
& 5 \\
& 6 \\
& 7 \\
& 8 \\
& 9
\end{aligned}
\] & \begin{tabular}{l}
Centered \\
Next to each other \\
(tiled) \\
Left top \\
Left \\
bottom \\
Right top \\
Right \\
bottom \\
Left \\
Right \\
Top \\
Bottom
\end{tabular} \\
\hline \begin{tabular}{l}
Vert. \\
Alignment (barcodes)
\end{tabular} & Vertical alignment of the content in the available space. & \[
\begin{aligned}
& \hline 0 \\
& 1 \\
& 2 \\
& \text { Formula }
\end{aligned}
\] & \begin{tabular}{l}
Top \\
Centered \\
Bottom \\
Formula \\
wizard
\end{tabular} \\
\hline
\end{tabular}
\begin{tabular}{|c|c|c|c|}
\hline Alignment (text) & Text alignment. Decimal means that numbers are aligned by their decimal points. & \[
\begin{aligned}
& \hline 0 \\
& 1 \\
& 2 \\
& 3 \\
& \text { Formula }
\end{aligned}
\] & \begin{tabular}{l}
Left \\
Centered Right Decimal Formula wizard
\end{tabular} \\
\hline & Decimal The position of the position decimal point within the field, measured from the left edge of the previous frame. & Number Formula & Formula wizard \\
\hline Justified & Text is aligned to the right and the left (block text). & True False Formula & \begin{tabular}{l}
Yes \\
No \\
Formula wizard
\end{tabular} \\
\hline Width & The width of the column. You will get an error message if the sum of the column widths exceeds the total width of the table object. & Number Formula & Formula wizard \\
\hline Fit & \begin{tabular}{l}
Specifies the behavior if the text is too long for one line. \\
To prevent long words from being truncated with value "1" (wrap), you can use the "Force wrap" option to ensure that a break occurs after the last suitable character. Value "3" (compress) reduces the character spacing and should only be used to a limited extent in order to guarantee legibility.
\end{tabular} & \begin{tabular}{l}
0 \\
1 \\
3 \\
Formula
\end{tabular} & \begin{tabular}{l}
Truncate \\
Wrap \\
Shrink \\
Compress \\
Formula \\
wizard
\end{tabular} \\
\hline & \begin{tabular}{l}
Widow/ Prevents widow and Orphan orphan lines. \\
control The last line of a paragraph is referred to as a widow line if it is also the first line of a new column or page. If a new page or column is triggered after the first line of a new paragraph, this line would appear alone at the end of the page or column. In this case, it is referred to as an orphan line.
\end{tabular} & True False Formula & \begin{tabular}{l}
Yes \\
No \\
Formula wizard
\end{tabular} \\
\hline
\end{tabular}
\begin{tabular}{|c|c|c|c|}
\hline & wrap be wrapped, a line break will be forced after the last suitable character. & False Formula & No Formula wizard \\
\hline Height (not text or RTF text). & Fixed height of the field, the content is scaled ( 0 : no fixed height). The highest column determines the overall height of a table row. & Number Formula & Formula wizard \\
\hline Fixed height (text or RTF text). & Fixed height of the field irrespective of the content. Excess text is discarded (0: no fixed height). The highest column determines the overall height of a table row. & Number Formula & Formula wizard \\
\hline Blank Optimization & Leading and double spaces are removed. & True False Formula & \begin{tabular}{l}
Yes \\
No Formula wizard
\end{tabular} \\
\hline Line spacing (only with text) & Spacing between the text lines. & Number Formula & Formula wizard \\
\hline Frame & Specifies the frame
characteristics and margins for
the individual table cells.
Together with the selected font,
the "top" and "bottom" cell
margins determine the height of
the table rows. & Number Formula & \begin{tabular}{l}
Dialog \\
Formula wizard
\end{tabular} \\
\hline
\end{tabular}

ゅ
combit List \& Label

\section*{Chart Objects}
lan Chart objects are elements of the report container.

See also:
- Object Properties
- Special Fields
- Pie Chart: Data Source
- Pie Chart: Pie Segment
, Pie Chart: Diagram
- Bars/Lines: Category and Series Axes
- Bars/Lines: Value Axis Settings
- Bars/Lines: Diagram
- Object (All Diagram Types)
- Colors (All Diagram Types)
combit List \& Label

\section*{Object Properties}

See chapters Common Object Properties and Report Container Object.
combit List \& Label

\section*{Special Fields}
\begin{tabular}{ll}
\hline Fields & Purpose \\
\hline LL.ChartObject.ArcIndex() & \begin{tabular}{l} 
Returns the index of the \\
current pie segment. \\
The largest pie segment \\
has index 1, the second \\
largest has index 2 and \\
so on.
\end{tabular} \\
\hline LL.ChartObject.ArcPerc() & \begin{tabular}{l} 
Returns the percentage \\
share of the current pie \\
segment.
\end{tabular} \\
\hline LL.ChartObject.ArcTotal() & \begin{tabular}{l} 
Returns the absolute \\
value of the total data \\
volume with pie charts.
\end{tabular} \\
\hline LL.ChartObject.ArcTypelsOthers() & \begin{tabular}{l} 
Returns True, if the \\
current pie segment is \\
the "other" segment.
\end{tabular} \\
\hline LL.ChartObject.ArcValue() & \begin{tabular}{l} 
Value of the pie \\
segment.
\end{tabular} \\
\hline LL.ChartObject.AxisCoordinate() & \begin{tabular}{l} 
Describes the content of \\
the source value.
\end{tabular} \\
\hline
\end{tabular}

\section*{Pie Chart: Data Source}

On the "Data source" tab, you can specify the coordinate values for the data.
\begin{tabular}{|c|c|c|c|}
\hline Property & Description & Value & Description \\
\hline Coordinate value & Choose the data source for the pie segments (name of the customer or employee, name of the product category, month or quarter in the case of dates etc.). & Formula & Formula wizard \\
\hline Minimum share & Especially if you have a lot of values that have small shares, it's sometimes a good idea to group them together under "others". You can define threshold values here which specify when individual segments are to be grouped together in one segment. & Number & Formula wizard \\
\hline Sort coordinates & Specifies whether the pie
segments are to be sorted
(alphanumeric or alphabetical). & True False Formula & \begin{tabular}{l}
Yes \\
No Formula wizard
\end{tabular} \\
\hline Number of sample records & The Designer does not yet have the "real" data that appears in your chart when it is printed. So that you can still get a picture in the Designer preview of how your chart will look, you can use this option to specify the number of pie segments that are to be displayed in the Designer. & Number & Formula wizard \\
\hline Filter & You can define a filter condition here. Only data records that fulfill the condition will then be used for the diagram. All data will be used if you select "True". & True False Formula & True False Formula wizard \\
\hline Coordinate & Defines the text for the coordinate label on the pie & Formula & Formula wizard \\
\hline
\end{tabular}


\section*{Pie Chart: Pie Segment}

The "Pie Segment" tab lets you make settings for calculating and presenting the pie segment.
\begin{tabular}{lllll}
\hline Property & Description & Value & Description \\
\hline \begin{tabular}{l} 
Coordinate \\
value
\end{tabular} & \begin{tabular}{l} 
Here you specify the formula for \\
the coordinate value that \\
determines the size of the pie \\
segment (total turnover, average \\
turnover, number of sales etc.).
\end{tabular} & Number & \begin{tabular}{l} 
Formula \\
wizard
\end{tabular} \\
\hline Width & Pie width in percent. & \begin{tabular}{l} 
Formula \\
Explosion \\
offset
\end{tabular} & \begin{tabular}{l} 
The individual pie segments are \\
accentuated by bringing them \\
forward out of the pie. The value \\
describes the distance by which \\
the pie segment is to be raised (as \\
a percentage of the pie's radius).
\end{tabular} & \begin{tabular}{l} 
Formula \\
wizard
\end{tabular} \\
\hline \begin{tabular}{lll} 
Label \\
object
\end{tabular} & \begin{tabular}{l} 
Specifies whether a label is to be \\
output on the pie segment.
\end{tabular} & \begin{tabular}{l} 
0
\end{tabular} & \begin{tabular}{l} 
No \\
Yes
\end{tabular} \\
\hline
\end{tabular}

\section*{Pie Chart: Diagram}

The "Diagram" tab lets you make settings for the appearance of the pie.
\begin{tabular}{|c|c|c|c|}
\hline Property & Description & Value & Description \\
\hline Color mode & The pie segments are marked in different colors so that the individual values can be differentiated more easily. Color settings by means of the "Colors" tab. & \[
\begin{aligned}
& \hline 0 \\
& 1 \\
& \text { Formula }
\end{aligned}
\] & Monochrome pie Colored pie Formula wizard \\
\hline Perspective & This property lets you choose whether the diagram is to be created with a slight or a strong perspective. Alternatively, you can also use a simple parallel projection. & \[
\begin{aligned}
& \hline 0 \\
& 1 \\
& 2 \\
& \text { Formula }
\end{aligned}
\] & None Slight perspective Strong perspective Formula wizard \\
\hline \multirow[t]{2}{*}{Perspective gradient} & The perspective gradient produces a brightness gradient across the surface of the pie chart. Gradient in percent. & Number & Formula wizard \\
\hline & Accentuate Raise the edge of frame the pie & True False Formula & \begin{tabular}{l}
Yes \\
No \\
Formula wizard
\end{tabular} \\
\hline \(X \quad\) axis
rotation
angle & \begin{tabular}{l}
The rotation angle upwards around the x-axis in degrees, maximum \(90^{\circ}\) (vertical). \\
Determines the horizontal positioning of the pie. You can also specify this angle by means of the rotation buttons that appear on the workspace when you select a diagram.
\end{tabular} & Number & Formula wizard \\
\hline Y axis
rotation
angle & \begin{tabular}{l}
The rotation angle in degrees around the center of the pie, anticlockwise. \\
You can also specify this angle by
\end{tabular} & Number & Formula wizard \\
\hline
\end{tabular}
\begin{tabular}{|c|c|c|c|}
\hline & means of the rotation buttons that appear on the workspace when you select a diagram. & & \\
\hline \multirow[t]{3}{*}{Separator lines} & Display separator lines between & True & Yes \\
\hline & the pie segments. & False & No \\
\hline & & Formula & Formula \\
\hline
\end{tabular}

\section*{Bars/Lines: Category and Series Axes}

If you have decided in favor of a three-axis diagram, you have both of these axes at your disposal (x axis and \(x\) axis). With two-axis charts (e.g. a simple bar chart), you only need the category axis (x axis).
\begin{tabular}{|c|c|c|c|}
\hline Property & Description & Value & Description \\
\hline Coordinate value & Here you choose the data source for the coordinate, e.g. "name" with persons or "month" with dates. & Formula & Formula wizard \\
\hline Sort coordinates & Specifies whether the coordinates are to be sorted (alphanumeric or alphabetic). & True False Formula & \begin{tabular}{l}
Yes \\
No \\
Formula wizard
\end{tabular} \\
\hline Number of sample records & The Designer does not yet have the "real" data that appears in your chart when it is printed. So that you can still get a picture in the Designer preview of how your chart will look, you can use this option to specify the number of sample records that are to be displayed in the Designer. & Number & Formula wizard \\
\hline Filter & You can define a filter condition here. Only data records that fulfill the condition will then be used for the diagram. All data will be used if you select "True". & True False Formula & True False Formula wizard \\
\hline \multirow[t]{2}{*}{Axis label} & Defines the text for the axis label. & Formula & Formula wizard \\
\hline & Rotation \begin{tabular}{l} 
Rotation of the axis \\
label in degrees.
\end{tabular} & Number & Formula wizard \\
\hline \multirow[t]{2}{*}{Coordinate label} & Defines the text for the coordinate label or the legend. & Formula & Formula wizard \\
\hline & \begin{tabular}{lll} 
Rotation & \begin{tabular}{l} 
Rotation \\
coordinate
\end{tabular} label in \\
& degrees (with long
\end{tabular} & Number & Formula wizard \\
\hline
\end{tabular}
texts).
\begin{tabular}{|c|c|}
\hline & \begin{tabular}{llll} 
Fixed & The font size is to be & True & Yes \\
font size & fixed (otherwise it will & False & No \\
& become smaller if & Formula & Formula \\
& necessary in order to \\
& prevent overlapping). & & wizard
\end{tabular} \\
\hline Legend & \begin{tabular}{ll} 
Position of the legend. & None \\
When you select "At axis", the & At axis \\
values are placed directly on the & Top \\
axis. Otherwise the label is & Left \\
placed on the right, left, upper or & Right \\
lower side of the chart. & Bottom
\end{tabular} \\
\hline Use series to determine the values & \begin{tabular}{l}
For the series axis (y-axis) in a three-axis diagram, you can also specify the values by means of rows instead of formulas. \\
This means that you define the different rows (e.g. measured value/target value/actual value) with a single data record and can show them parallel e.g. in a line diagram. \\
Select the "Use rows as data source" entry from the drop-down list above the properties. This option changes the properties of the series axis and you also have the "Row definition" property. You define the individual rows by opening the "Row Definitions" dialog. You can define the properties differently for each row and move the rows with the arrow button.
\end{tabular} \\
\hline
\end{tabular}

\section*{Bars/Lines: Value Axis Settings}

On the "Value axis" tab, you can make settings for calculating and presenting the value axis of a bar chart or line diagram.
Primary axis/secondary axis: These charts support a second value axis. The second axis is enabled on the "Diagram" tab. Use the drop-down box to switch to the properties for the respective axis.
\begin{tabular}{|c|c|c|c|}
\hline Property & Description & Value & Description \\
\hline Coordinate value & Here you specify the formula for the coordinate value (total turnover, average turnover, number of sales etc.). & Formula & Formula wizard \\
\hline \multirow[t]{4}{*}{Axis scale} & \multirow[t]{4}{*}{Type of axis scaling.} & 0 & None (linear) \\
\hline & & 1 & Logarithmic, base 10 (decimal) \\
\hline & & 2 & Logarithmic, base (binary) \\
\hline & & Formula & Formula wizard \\
\hline \multirow[t]{11}{*}{\begin{tabular}{l}
Maximum \\
Value \\
Automatic
\end{tabular}} & \multirow[t]{11}{*}{\begin{tabular}{l}
Do you want the value axis to continue until it reaches a certain maximum value or do you want to determine the end value automatically? \\
You can limit the maximum height of the displayed area e.g. to cater for "anomalies . If your values contain extremely high peaks, you can cap them by setting a maximum value and show the progression of the "small" values more clearly. If you keep the default value "No", the diagram will be adapted so
\end{tabular}} & True & Yes \\
\hline & & False & No \\
\hline & & Formula & Formula wizard \\
\hline & & & \\
\hline & & & \\
\hline & & & \\
\hline & & & \\
\hline & & & \\
\hline & & & \\
\hline & & & \\
\hline & & & \\
\hline
\end{tabular}

\begin{tabular}{|c|c|c|c|}
\hline & coordinate label in degrees. & & wizard \\
\hline & \begin{tabular}{ll} 
Fixed font & The font size is to be \\
size
\end{tabular} \begin{tabular}{l} 
fixed (otherwise it \\
\begin{tabular}{l} 
will become smaller \\
as necessary in order \\
to \\
overlapping).
\end{tabular} \\
\end{tabular} & True False Formula & \begin{tabular}{l}
Yes \\
No \\
Formula wizard
\end{tabular} \\
\hline Coordinate tick distance & Calculation of the distance between two coordinate ticks. & True False Formula & Automatic Manual Formula wizard \\
\hline Legends & Position of the legend for this axis. If you select "At axis", the values will be placed directly on the axis. & & None At axis \\
\hline Number of ticks & The number of subdivisions separated by tick marks between the main coordinate markings. & Number & Formula wizard \\
\hline
\end{tabular}

\section*{Bars/Lines: Diagram}

The "Diagram" tab lets you make settings for the appearance of bar charts and line diagrams.
\begin{tabular}{|c|c|c|c|}
\hline Property & Description & Value & Description \\
\hline Secondary axis & Supports a secondary axis on the left hand side. The axis properties are set on the "Value Axis" tab. You select the respective axis from a dropdown box. & \begin{tabular}{l}
True \\
False \\
Formula
\end{tabular} & \begin{tabular}{l}
Yes \\
No \\
Formula \\
wizard
\end{tabular} \\
\hline & \begin{tabular}{ll} 
Axis & Specifies which \\
assignment \\
& \begin{tabular}{l} 
value axis the \\
value is assigned \\
to.
\end{tabular} \\
&
\end{tabular} & \begin{tabular}{l}
\[
\begin{aligned}
& \hline 0 \\
& 1
\end{aligned}
\] \\
Formula
\end{tabular} & Primary axis Secondary axis Formula wizard \\
\hline Alignment & Alignment of the graphic elements, e.g. for a horizontal bar chart. & & Left to right Bottom to top \\
\hline Illuminated & Specifies whether the chart is to be illuminated. & True False Formula & \begin{tabular}{l}
Yes \\
No Formula wizard
\end{tabular} \\
\hline Color mode & Specifies which axis determines the color. & \[
\begin{aligned}
& \hline 0 \\
& 1 \\
& 2 \\
& \text { Formula }
\end{aligned}
\] & \begin{tabular}{l}
Monochrome \\
\(x\) axis \\
\(y\) axis \\
Formula \\
wizard
\end{tabular} \\
\hline Background color & Background behind the diagram. & & Transparent Pattern/block color \\
\hline Isotropic & Specifies that both data axes (x and \(y\) axis) use the same units. & \begin{tabular}{l}
True \\
False \\
Formula
\end{tabular} & \begin{tabular}{l}
Yes \\
No Formula wizard
\end{tabular} \\
\hline Perspective & This property lets you choose whether the diagram is to be created with a slight or a strong & \[
\begin{aligned}
& \hline 0 \\
& 1
\end{aligned}
\] & None Slight perspective \\
\hline
\end{tabular}
\begin{tabular}{|c|c|c|c|}
\hline & perspective. Alternatively, you can also use a simple parallel projection. & 2
Formula & Strong perspective Formula wizard \\
\hline \(X \quad\) axis
rotation
angle & \begin{tabular}{l}
The rotation angle upwards around the \(x\) axis in degrees, maximum \(90^{\circ}\) (vertical). \\
Determines the horizontal positioning of the diagram in the available space. You can also specify this angle by means of the rotation buttons that appear on the workspace when you select a diagram.
\end{tabular} & Number & Formula wizard \\
\hline \[
\begin{aligned}
& \mathrm{Y} / \mathrm{Z} \text { axis } \\
& \text { rotation } \\
& \text { angle }
\end{aligned}
\] & The rotation angle in degrees around the center diagram, anticlockwise. You can also specify this angle by means of the rotation buttons that appear on the workspace when you select a diagram. & Number & Formula wizard \\
\hline Separator lines & Display separator lines between the pie segments. & True False Formula & \begin{tabular}{l}
Yes \\
No \\
Formula \\
wizard
\end{tabular} \\
\hline
\end{tabular}

\section*{Object (All Diagram Types)}

On the "Object" tab you will find the settings for the title and the background.
\begin{tabular}{|c|c|c|c|}
\hline Property & Description & Value & Description \\
\hline Title & You can specify the title of your diagram here. It will then be displayed at the upper margin of the object. You can also select a formula with the formula button. The "Font" button lets you change the font for the title. Click it with the left mouse button to open a font selection dialog; a right-click resets the font to the default value for the object. & Formula & Formula wizard \\
\hline Title position & Position of the diagram's title. & \[
\begin{aligned}
& \hline 0 \\
& 1 \\
& \text { Formula }
\end{aligned}
\] & Top Bottom Formula wizard \\
\hline Background & Select the color that you want to use for the background of the available area. You can also make it transparent. You can select a color in the upper dropdown box or click "..." to open a standard color selection dialog. & True False Formula & \begin{tabular}{l}
Yes \\
No \\
Formula wizard
\end{tabular} \\
\hline & \begin{tabular}{ll} 
Filling & \begin{tabular}{l} 
Filling for \\
available area
\end{tabular}
\end{tabular} & & \begin{tabular}{l}
Transparent \\
Pattern/block \\
color \\
Horiz. \\
Gradient \\
Vert. \\
Gradient \\
Horiz. 2-part \\
gradient \\
Vert. 2-part \\
gradient \\
Partly
\end{tabular} \\
\hline
\end{tabular}
\begin{tabular}{llll} 
& & \begin{tabular}{l} 
transparent \\
Picture
\end{tabular} \\
\hline Border & \begin{tabular}{l} 
Border for the \\
available area
\end{tabular} & \begin{tabular}{l} 
Transparent \\
Pattern/block \\
color
\end{tabular} \\
\hline Shadow & \begin{tabular}{l} 
Shadow for the \\
available area
\end{tabular} & \begin{tabular}{l} 
Transparent \\
Pattern/block \\
color
\end{tabular} \\
\hline Rounding & \begin{tabular}{l} 
Rounding factor in Number \\
percent for the \\
corners of the \\
available armula \\
\(0=\) rectangular area: \\
(square corners), \\
100=elliptical (short \\
edge is round)
\end{tabular} & \begin{tabular}{l} 
wizard
\end{tabular} \\
\hline
\end{tabular}

\section*{Colors (All Diagram Types)}
\begin{tabular}{llll}
\hline Property & Description & Value & Description \\
\hline \begin{tabular}{ll} 
Assigned \\
colors
\end{tabular} & You can assign fixed & Color & \begin{tabular}{l} 
Formula \\
wizard
\end{tabular} \\
& colors to particular axis \\
values. If you click the & & \\
& "New" button, you can & & \\
& create a new \\
& assignment. & & \\
\hline \begin{tabular}{l} 
Unassigned \\
colors
\end{tabular} & \begin{tabular}{l} 
Specifies the colors and color \\
sequences for the data rows \\
that are not specified by the \\
"Assigned colors".
\end{tabular} & Color \\
& & dialog \\
\hline
\end{tabular}

ゅ
combit List \& Label

\section*{Gauge Objects}
© Gauges can be placed in table lines or as an object.
See also:
- Object Properties
- Content
combit List \& Label

\section*{Object Properties}

See chapter Common Object Properties.

\section*{Content}

\section*{You define the appearance of the gauge on the "Content" tab.}
\begin{tabular}{llll}
\hline Property & Description & Value & Description \\
\hline Background & \begin{tabular}{l} 
Filling properties (background \\
of the gauge).
\end{tabular} & \\
\hline & \begin{tabular}{l} 
Color. Color of the \\
background.
\end{tabular} & Color \\
\hline & \begin{tabular}{l} 
Fading-in factor. Value \\
ranging from 0 (transparent) \\
to 100 (opaque).
\end{tabular} & & \begin{tabular}{l} 
Formula \\
wizard
\end{tabular} \\
\hline
\end{tabular}

Pointer
options
\begin{tabular}{|c|c|c|c|}
\hline & \multicolumn{2}{|l|}{Color. Color of the pointer.} & Color \\
\hline & Size factor. Specifies the size of the pointer in relation to the scale range. & Value & Formula wizard \\
\hline & Display Range. Describes the percentage range of the available area (the radius or the width) in which this element is displayed. & Value & Formula wizard \\
\hline \multicolumn{4}{|l|}{Glass properties} \\
\hline & Color. Glass color. & & Color \\
\hline & Fading-in factor. Value ranging from 0 (transparent) to 100 (opaque). & Value & Formula wizard \\
\hline Scale range rotation angle & Clockwise rotation angle for the scale \(\left(0^{\circ}=\right.\) down \()\). & Value & Formula wizard \\
\hline White space before scale range & Specifies the free space between the beginning of the gauge and the beginning of the scale (max. 50\%). & Value & Formula wizard \\
\hline White space after scale & Specifies the free space between the end of the & Value & Formula wizard \\
\hline
\end{tabular}
\begin{tabular}{|c|c|c|c|}
\hline range & gauge and the end of the scale (max. 50\%). & & \\
\hline \multirow[t]{9}{*}{Tickmarks} & Scale type & \[
\begin{aligned}
& \hline 0 \\
& 1 \\
& 2 \\
& 3
\end{aligned}
\] & None Scale type 1 Scale type 2 Scale type 3 \\
\hline & \begin{tabular}{llr} 
Display & range. & The \\
"Minimum" & and & "Maximum" \\
values & describe & the \\
percentage & range of & the \\
available & area for the \\
tickmark. & &
\end{tabular} & Value & Formula wizard \\
\hline & Color. Color of the tickmark. & & Color \\
\hline & Tickmark width. Specifies the width of the scale tickmark (percentage value). & Value & Formula wizard \\
\hline & Detail level. Specifies the maximum tickmark level to be output. \(0=\) top level only. & Value & Formula wizard \\
\hline & Filling. & & \\
\hline & Intermediate tick position. Specifies where the intermediate tickmarks are to be placed. & & \\
\hline & Calculation type. Calculation type of the coordinate tickmark distance (ticks). & True False Value & Automatic Manual Formula wizard \\
\hline & Distance. The user-defined distance between two coordinate ticks. & Value & Formula wizard \\
\hline \multirow[t]{4}{*}{Scale labels} & Scale label properties. & True False & Show Hide \\
\hline & Display range. The "Minimum" and "Maximum" values describe the percentage range of the available area for the tickmark. & Value & Formula wizard \\
\hline & Rotated. Specifies whether the font is rotated. & True False & \[
\begin{aligned}
& \text { Yes } \\
& \text { No }
\end{aligned}
\] \\
\hline & Size adjustment. Specifies whether the font size may be reduced so that the lettering & True False & \[
\begin{aligned}
& \text { Yes } \\
& \text { No }
\end{aligned}
\] \\
\hline
\end{tabular}
fits the area.
Alignment. Text alignment.
\begin{tabular}{|c|c|c|c|}
\hline & Alignment. Text alignment. & \[
\begin{aligned}
& \hline 0 \\
& 1 \\
& 2 \\
& \text { Value }
\end{aligned}
\] & \begin{tabular}{l}
Left \\
Centered Right Formula wizard
\end{tabular} \\
\hline Signal ranges & The properties of the signal ranges. You can give the ranges different colors, e.g. to signalize an optimum range. You define the start value, start color, end value and end value of the different ranges in a dialog. & True False & Show Hide Dialog \\
\hline & Display range. The "Minimum" and "Maximum" values describe the percentage range of the available area for the tickmark. & Value & Formula wizard \\
\hline Scale Labels & \begin{tabular}{l}
List of labels. You can open a dialog to define the rotation, frame size, background, font, format and position of the individual labels. \\
You specify the position in relation to the area of the gauge (measured from left to right). For example, a vertical and horizontal position of 50\% each will position the label precisely in the middle.
\end{tabular} & True False & Show Hide Dialog \\
\hline Minimum value & Minimum value of the scale. & Value & Formula wizard \\
\hline & Fit to distance The minimum value is adjusted automatically to fit the coordinate tickmark distance. & True False & \[
\begin{aligned}
& \text { Yes } \\
& \text { No }
\end{aligned}
\] \\
\hline Maximum value & Maximum value of the scale. & Value & Formula wizard \\
\hline & Fit to distance The maximum value is adjusted automatically to fit the coordinate tickmark distance. & True False & \[
\begin{aligned}
& \text { Yes } \\
& \text { No }
\end{aligned}
\] \\
\hline Start value & Defines where the bar for displaying the value starts. 'Automatic' means: if the scale is exceeding 0 , it is 0 , & True False & Automatic Manual \\
\hline
\end{tabular}
otherwise it is the minimum
value.
Value \begin{tabular}{lll} 
Value to be indicated by the Value & \begin{tabular}{l} 
Formula \\
pointer.
\end{tabular} & wizard
\end{tabular}

ゅ
combit List \& Label

\section*{Crosstab Objects}

長 Crosstab objects are elements of the report container.
See also:
- Object Properties
- Special Functions
- Cell Properties
- Properties for the Crosstab Area
combit List \& Label

\section*{Object Properties}

See chapters Common Object Properties and Report Container Object.

\section*{Special Functions}
\begin{tabular}{ll}
\hline Function & Purpose \\
\hline \begin{tabular}{l} 
Crosstab.Cells.Max() \\
or \\
Crosstab.Cells.Min()
\end{tabular} & \begin{tabular}{l} 
Returns the largest or smallest \\
value in the entire crosstab.
\end{tabular} \\
\hline Crosstab.Cells.Avg() & \begin{tabular}{l} 
Returns the average value for \\
the entire crosstab.
\end{tabular} \\
\hline \begin{tabular}{l} 
Crosstab.Col\$() \\
or Crosstab.Row \(\$()\)
\end{tabular} & \begin{tabular}{l} 
Returns the description of the \\
column or the row for the \\
current cell.
\end{tabular} \\
\hline Crosstab.Value() & \begin{tabular}{l} 
Returns the content of the cell \\
(as a number).
\end{tabular} \\
\hline Total() & \begin{tabular}{l} 
You can use this function for \\
calculations across all cells. \\
Otherwise, calculations are \\
always made across all values \\
that affect the respective cell.
\end{tabular} \\
\hline Join\$() & \begin{tabular}{l} 
Returns a collection of strings, \\
separated by a particular \\
character.
\end{tabular} \\
\hline
\end{tabular}

\section*{Cell Properties}

Select the respective column in the object dialog on the "Cell definition" tab. To select multiple cells, hold down the CTRL key or you can draw a border around the cells with the mouse.
\begin{tabular}{|c|c|c|c|}
\hline Property & Description & Value & Description \\
\hline Value & Formula for the value of the cell. This will be evaluated by the Crosstab. Cells functions. & Formula & Formula wizard \\
\hline Displayed contents & Text to be displayed in the cells. This can differ from the value specified in the "Value" property, e.g. if it is formatted. & Formula & Formula wizard \\
\hline Link URL & Link that is accessed when clicked (only real data preview, PDF or HTML export). Example: file://c:\users\public\x.log or URL & Link & Link Formula wizard \\
\hline Rotation & Rotates the object anticlockwise. For example, you can rotate the column title by \(90^{\circ}\) with this function. & \[
\begin{aligned}
& \hline 0 \\
& 1 \\
& 2 \\
& 3 \\
& \text { Formula }
\end{aligned}
\] & \[
\begin{aligned}
& \hline 0^{\circ} \\
& 90^{\circ} \\
& 180^{\circ} \\
& 270^{\circ} \\
& \text { Formula } \\
& \text { wizard }
\end{aligned}
\] \\
\hline Vert. alignment & Vertical alignment of the content in the available space. & \[
\begin{aligned}
& \hline 0 \\
& 1 \\
& 2 \\
& \text { Formula }
\end{aligned}
\] & Top Centered Bottom Formula wizard \\
\hline Alignment (text) & Text alignment Decimal means that numbers are aligned by their decimal points. & \[
\begin{aligned}
& \hline 0 \\
& 1 \\
& 2 \\
& 3 \\
& \text { Formula }
\end{aligned}
\] & \begin{tabular}{l}
Left \\
Centered Right Decimal Formula wizard
\end{tabular} \\
\hline & \begin{tabular}{llcc} 
Decimal & Position & of & the \\
position & decimal & point & (only
\end{tabular} & Number Formula & Formula \\
\hline
\end{tabular}
\begin{tabular}{|c|c|c|c|}
\hline & valid with decimal alignment, negative means: from the right.) & & wizard \\
\hline Blank Optimization & Leading and double spaces are removed. & True False Formula & \begin{tabular}{l}
Yes \\
No \\
Formula \\
wizard
\end{tabular} \\
\hline Maximum width & \begin{tabular}{l}
Specifies the maximum width for a cell. A line break will be triggered if the text is wider. \\
You can use this property in combination with the Join\$() function to display the individual values in one cell.
\end{tabular} & Number Formula & Formula wizard \\
\hline Minimum width & Sets the minimum width for the type of cell. & Number Formula & Formula wizard \\
\hline Minimum height & Sets the minimum height for the type of cell. & Number Formula & Formula wizard \\
\hline
\end{tabular}

\section*{Properties for the Crosstab Area}
\begin{tabular}{|c|c|c|c|}
\hline Property & Description & Value & Description \\
\hline Link URL & Link that is accessed when clicked (only real data preview, PDF or HTML export). Example: file://c:\users\public\x.log or URL & Link & Formula wizard \\
\hline Minimum size & Specifies the minimum size to which the crosstab may be reduced in order to avoid a horizontal page break. \(50=\) reduced by up to \(50 \%\) to avoid a page break; 100=keep original size. & Number & Formula wizard \\
\hline Minimum height & Specifies the minimum height that must be available for the object. A page break will be triggered if less space is available. & Number & Formula wizard \\
\hline \multirow[t]{4}{*}{Columns} & Specifies the column properties in the event of a page break & Formula & Formula wizard \\
\hline & \begin{tabular}{ll} 
Repeat & \begin{tabular}{l} 
Specifies whether row \\
labels \\
labels are to be \\
repeated if there is a \\
column break.
\end{tabular}
\end{tabular} & True False Formula & \begin{tabular}{l}
Yes \\
No \\
Formula wizard
\end{tabular} \\
\hline & \begin{tabular}{ll} 
Page & True: If the cross \\
Break on & table is too wide, the \\
Shadow & wrapped parts are \\
Pages & printed on shadow \\
& pages. A shadow \\
& page does not count \\
& as a "real" page and \\
& therefore does not \\
& have a page number. \\
& False: The wrapped \\
& parts are output \\
below the table.
\end{tabular} & True False Formula & \begin{tabular}{l}
Yes \\
No Formula wizard
\end{tabular} \\
\hline & \begin{tabular}{ll} 
Distance & Distance from the \\
before & previous element.
\end{tabular} & Number Formula & Formula \\
\hline
\end{tabular}
wizard
\begin{tabular}{|c|c|c|c|c|}
\hline & Break level & Specifies the ideal break level. 0=inner group, i.e. the bottom line of the column definitions. & Number Formula & Formula wizard \\
\hline & Force & Forces a break after each corresponding group. & True False Formula & \begin{tabular}{l}
Yes \\
No \\
Formula \\
wizard
\end{tabular} \\
\hline Rows & Repeat labels & Specifies whether column labels are to be repeated if there is a line break. & True False Formula & \begin{tabular}{l}
Yes \\
No \\
Formula wizard
\end{tabular} \\
\hline & Break level & Specifies the ideal break level. \(0=\) inner group, i.e. the bottom line of the line definitions. & Number Formula & Formula wizard \\
\hline & Force & Forces a break after each corresponding group. & True False Formula & \begin{tabular}{l}
Yes \\
No \\
Formula \\
wizard
\end{tabular} \\
\hline
\end{tabular}

ゅ
combit List \& Label

\section*{Formatted Text Objects}
(园 As opposed to the normal text object, with this object, you can also change the formatting of the text within a line.
Note: Normal text objects can be printed considerably faster. You should therefore only use formatted text objects if you need particular formatting that you cannot achieve or cannot achieve without difficulty - in normal text objects.

See also:
, Object Properties

\section*{Object Properties}
\begin{tabular}{|c|c|c|c|}
\hline Property & Description & Value & Description \\
\hline Rotation & Rotates the object anticlockwise. & \[
\begin{aligned}
& \hline 0 \\
& 1 \\
& 2 \\
& 3 \\
& \text { Formula }
\end{aligned}
\] & \[
\begin{aligned}
& \hline 0^{\circ} \\
& 90^{\circ} \\
& 180^{\circ} \\
& 270^{\circ} \\
& \text { Formula } \\
& \text { wizard }
\end{aligned}
\] \\
\hline Page break & \begin{tabular}{l}
Specifies whether the object can trigger a page break or a text overflow. \\
If this property is enabled, the content will be wrapped to the next page automatically (or to another interlinked RTF object) if it exceeds the size of the object. This is an interesting option e.g. with RTF objects that are to cover several pages. With labels, the next label will only be started when all objects have been printed as a result of this option in the previous label. You might not be able to set this property if page breaks are not supported by the higher-level program.
\end{tabular} & True False Formula & \begin{tabular}{l}
Yes \\
No \\
Formula wizard
\end{tabular} \\
\hline
\end{tabular}

ゅ
combit List \& Label

\section*{Form Control Objects}
\& The user can fill out form control objects directly in the List \& Label preview and in the PDF format. He can also trigger actions such as send by email. You control the element's basic behavior by selecting a type. The properties that you have at your disposal change depending on the type that you select.
Also see chapter Common Object Properties.
\begin{tabular}{lllllll}
\hline Property & Description & & & Value & Description \\
\hline Type & Specifies & the type & of & the & 0 & Edit \\
& element. & & & & 1 & Checkbox \\
& & & & 2 & Combobox \\
& & & & 3 & Button \\
\hline
\end{tabular}

Tooltip
Tooltip that is to appear

See also:
, Edit
- Checkbox
- Combobox
- Button

\section*{Edit}
\begin{tabular}{|c|c|c|c|}
\hline Property & Description & Value & Description \\
\hline Force input & Specifies whether the user must make an entry. & True False Formula & \begin{tabular}{l}
Yes \\
No Formula wizard
\end{tabular} \\
\hline Field name & Specifies the field name for a possible data export via XML/XFDF. Free text must be enclosed in quotation marks. & & Name \\
\hline \multirow[t]{2}{*}{Validation expression} & \begin{tabular}{l}
Regular expression for validating the input. \\
Examples: \\
Field not empty: ".+" \\
Simple email validation: \\
"^.+@.+\..\{2,3\}\$"
\end{tabular} & & Formula wizard \\
\hline & \begin{tabular}{ll} 
Error \\
message & \begin{tabular}{l} 
Message to be \\
displayed if validation
\end{tabular} \\
& fails.
\end{tabular} & & Formula wizard \\
\hline Value & Default value for the input field. & & Formula wizard \\
\hline Alignment (text) & Text alignment Decimal means that numbers are aligned by their decimal points. & \[
\begin{aligned}
& \hline 0 \\
& 1 \\
& 2 \\
& \text { Formula }
\end{aligned}
\] & \begin{tabular}{l}
Left \\
Centered Right Formula wizard
\end{tabular} \\
\hline Background & Specifies whether the field is to have a colored background. & \[
\begin{aligned}
& \hline 0 \\
& 1 \\
& \text { Formula }
\end{aligned}
\] & Transparent Color Formula wizard \\
\hline & Color Background color & & \\
\hline Multi-line & Specifies whether the input field can consist of multiple lines. If you select single-line, characters can still be entered and the input field scrolls automatically. However, in this case, the excess characters will & True False Formula & \begin{tabular}{l}
Yes \\
No Formula wizard
\end{tabular} \\
\hline
\end{tabular}
\begin{tabular}{|c|c|c|c|c|}
\hline & \multicolumn{2}{|l|}{\multirow[t]{2}{*}{\begin{tabular}{l}
be truncated when the field is output. \\
Specifies whether the object is to have a border.
\end{tabular}}} & & \\
\hline \multirow[t]{3}{*}{Border} & & & True False & \\
\hline & Color & Color of the border. & & \\
\hline & Width & Width of the shadow, in the unit of measure of the workspace. & Number Formula & Formula wizard \\
\hline
\end{tabular}

\section*{Checkbox}
\begin{tabular}{llll}
\hline Property & Description & Value & Description \\
\hline Field name & \begin{tabular}{l} 
Specifies the field name for a \\
possible data export via \\
\\
\\
XML/XFDF. Free text must be \\
enclosed in quotation marks.
\end{tabular} & Name & \\
\hline Value & \begin{tabular}{l} 
Default value for the input \\
field.
\end{tabular} & & \\
\hline Background & See edit type & & \\
\hline Border & See edit type & 2 & Cross \\
\hline Type & Appearance of the checkbox. & 1 & \begin{tabular}{l} 
Filled \\
\end{tabular} \\
& & 3 & Formula \\
& & \begin{tabular}{l} 
Formula \\
wizard
\end{tabular} \\
\hline
\end{tabular}

\section*{Combobox}
\begin{tabular}{|c|c|c|c|}
\hline Property & Description & Value & Description \\
\hline Force input & Specifies whether the user must make an entry. & True False Formula & \begin{tabular}{l}
Yes \\
No \\
Formula wizard
\end{tabular} \\
\hline Items & Default entries available in the combobox & List & List of
items for
selection \\
\hline Field name & Specifies the field name for a possible data export via XML/XFDF. Free text must be enclosed in quotation marks. & Formula & Name \\
\hline \multirow[t]{2}{*}{Validation expression only with variable text (Editable=yes)} & Regular expression for validating the input. & & Formula wizard \\
\hline & Error
message \begin{tabular}{l} 
Message to \\
displayed \\
validation fails.
\end{tabular}\(\quad\)\begin{tabular}{c} 
be \\
if
\end{tabular} & & Formula wizard \\
\hline Editable & Specifies whether the user may enter other values that are not included as selection items. & True False Formula & \begin{tabular}{l}
Yes \\
No Formula wizard
\end{tabular} \\
\hline Value & Default value for the input field. & & Formula wizard \\
\hline Alignment & See edit type & & \\
\hline Background & See edit type & & \\
\hline Border & See edit type & & \\
\hline
\end{tabular}

\section*{Button}
\begin{tabular}{|c|c|c|c|}
\hline Property & Description & Value & Description \\
\hline Action & \begin{tabular}{l}
Specifies the possible behavior. Only available with buttons. \\
0 : Send as mail: You can set default values for the normal email-relevant fields. \\
1: Save: Saves the preview file or the input data in the chosen format. \\
2: Send: via HTTP POST \\
3: URL: Internet address to be accessed when clicked. The object is transparent which means that you can place it on top of other objects to create links for all objects.
\end{tabular} & \[
\begin{aligned}
& \hline 0 \\
& 1 \\
& 2 \\
& 3
\end{aligned}
\] & Send as
mail
Save
Send via
HTTP POST
Link \\
\hline Text & Button label & Formula & Formula wizard \\
\hline
\end{tabular}

ゅ
combit List \& Label

\section*{HTML Text Objects}

HTML objects are used for displaying HTML content. List \& Label uses Microsoft's WebBrowser control and its DrawToDC() method for displaying HTML content. Unfortunately, not all pages are displayed correctly with this method. Microsoft does not name any concrete restrictions as any such restrictions are considered version-specific and can therefore change at any time (in future versions).
According to our experience, there should be no problems with simple web sites or simple HTML streams. Therefore, avoid complex scripts, redirects and exotic CSS tags. If in doubt, we advise you to contact Microsoft directly.
Displaying a HTML page works more reliable if the page is displayed in the so-called "Quirks" mode and not in the standard mode. That can be forced for a page by removing the element "<!DOCTYPE..." for example.

See also:
- Object Properties
- Object Content
combit List \& Label

\section*{Object Properties}

See chapter Common Object Properties.

\section*{Object Content}
\begin{tabular}{ll}
\hline Property & Description \\
\hline File & \begin{tabular}{l} 
Choose this option if you want to display the content \\
of a HTML file that you have saved. The "Open" \\
button displays a file selection dialog with which you \\
can locate the file that you want. The file must be \\
located on a local drive or network.
\end{tabular} \\
\hline URL & \begin{tabular}{l} 
This option lets you display the content of web sites \\
(e.g. www.combit.net). These web sites are loaded \\
from the Internet or intranet online during run time, \\
which means that you must always have an active \\
Internet connection.
\end{tabular} \\
\hline Formula \(\quad\)\begin{tabular}{l} 
If your application provides certain content in HTML \\
format, you can also select it here. In this case, please \\
consult the documentation for your application.
\end{tabular} \\
\hline Fit \begin{tabular}{ll} 
to If you enable this option, the entire content of the \\
object & \begin{tabular}{l} 
If \\
object will be adjusted to fit the size of the object. \\
Otherwise, the width will be adjusted and the output \\
will be spread across several pages.
\end{tabular} \\
\hline
\end{tabular}
\end{tabular}

ゅ
combit List \& Label

\section*{OLE Container}

OLE containers are used for including OLE Server
Documents in your project.

See also:
, Object Properties
combit List \& Label

\section*{Object Properties}

See chapter Common Object Properties.

ゅ
combit List \& Label

\section*{Form Template Objects}

Form templates are placed in the background of the workspace as a template so that other objects can be aligned to them. This is helpful when designing complex forms based on templates. The form template is a special case as it is not printed.

See also:
, Object Properties

\section*{Object Properties}
\begin{tabular}{|c|c|c|c|}
\hline Property & Description & Value & Description \\
\hline \multirow[t]{2}{*}{File name} & Choose the file containing the form template that you want. & \[
0
\] & Open file
dialog \\
\hline & \begin{tabular}{ll} 
Relative & The path is relative to \\
path & the project path.
\end{tabular} & True False & \[
\begin{aligned}
& \text { Yes } \\
& \text { No }
\end{aligned}
\] \\
\hline Visible in preview & Specifies whether the template is to be visible in the preview. & True False Formula & Yes No Formula wizard \\
\hline Fade color & The selected color will be linked with or to the template image in order to lighten the image when displaying. & \[
\stackrel{. .1}{\square}
\] & Color dialog Selection of predefined colors and formula wizard \\
\hline Keep proportions & With this option you can specific whether the height/width proportions are to be kept (True) when inserting the graphic or if the frame is to be adjusted, possibly leading to distortion (False). & True False Formula & \begin{tabular}{l}
Yes \\
No Formula wizard
\end{tabular} \\
\hline
\end{tabular}```

