



# **FastReport .NET User Manual (WinForms, Mono, WPF, Avalonia, ASP.NET)**

Version 2025.1

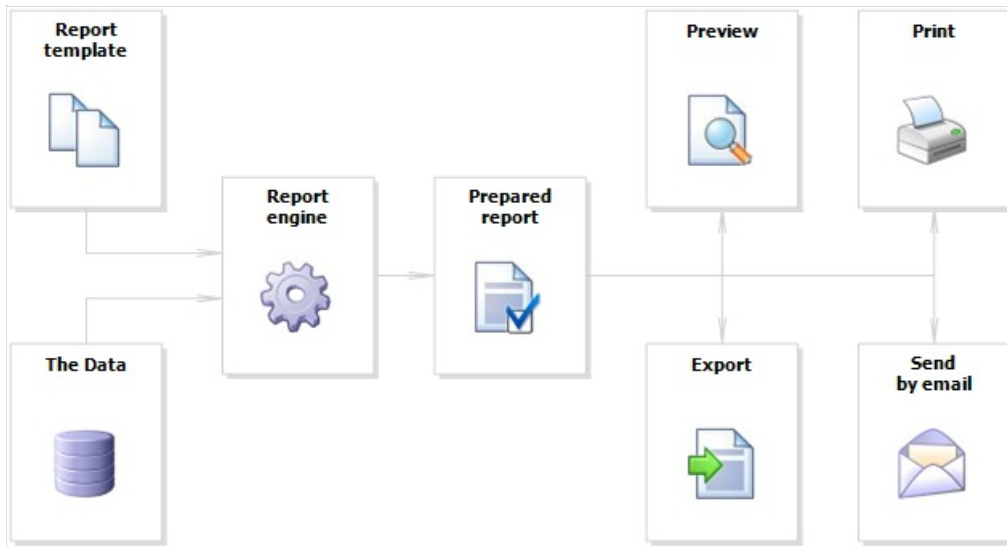
© 2008-2024 Fast Reports Inc.

# Fundamentals

In this chapter we will learn the principles of working with a report in the FastReport. We will also take a close look at report elements such as report pages, bands, and report objects.

# The report

The report building process can be represented as follows:



Report template (later-Report) - this is, what we see in the designer. Reports are saved in files with an extension .FRX. A Report can be created with the help of designer or programmatically.

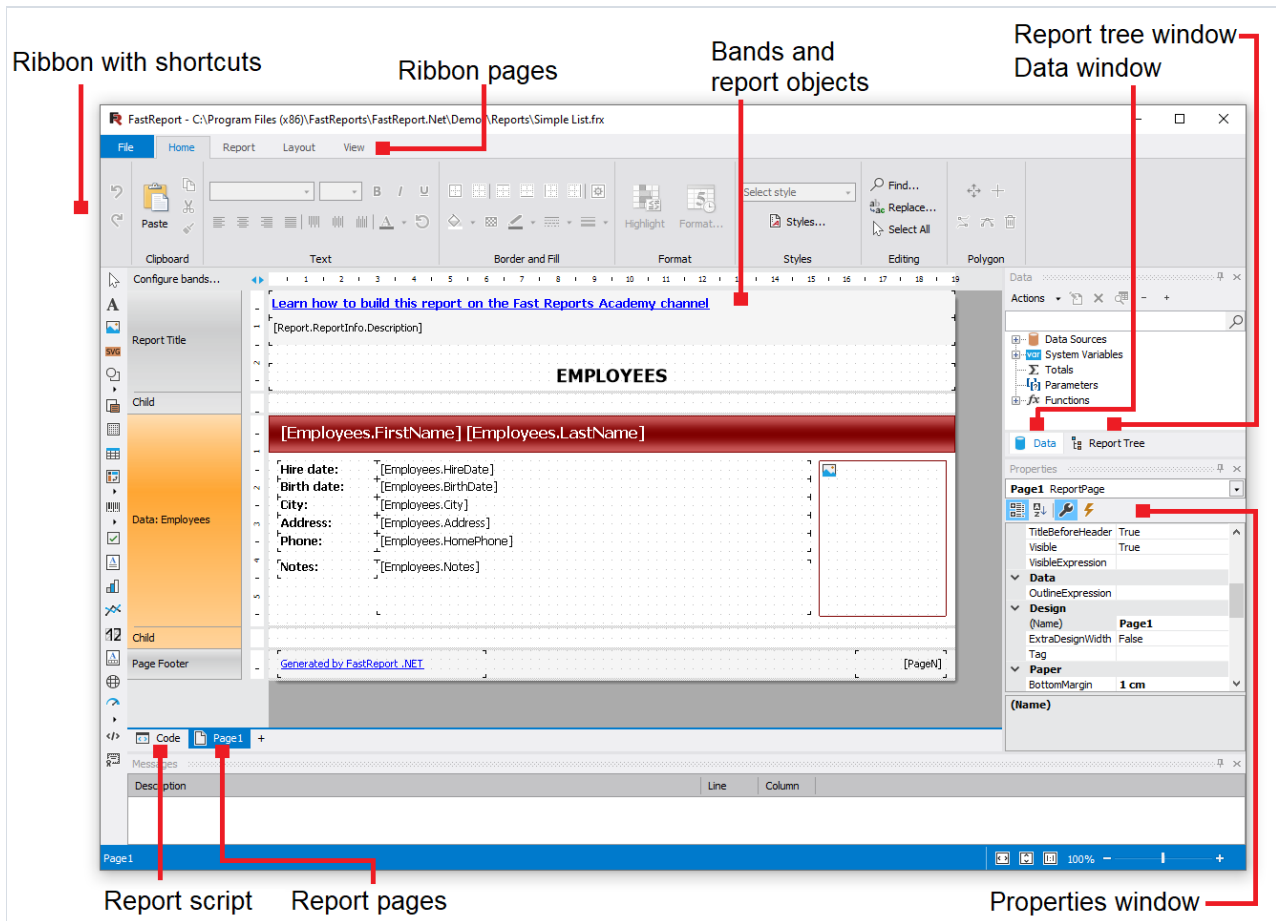
Data can be any: this is data, defined in the program, or data from DBMS, for example, MS SQL. FastReport can also work with business-logic objects (later - business-objects).

Prepared Report - this is what we see in the preview window. Prepared report can be previewed, printed, saved in one of the supported formats (.doc, .xls, .pdf and others), or can be sent by email.

# Report designer

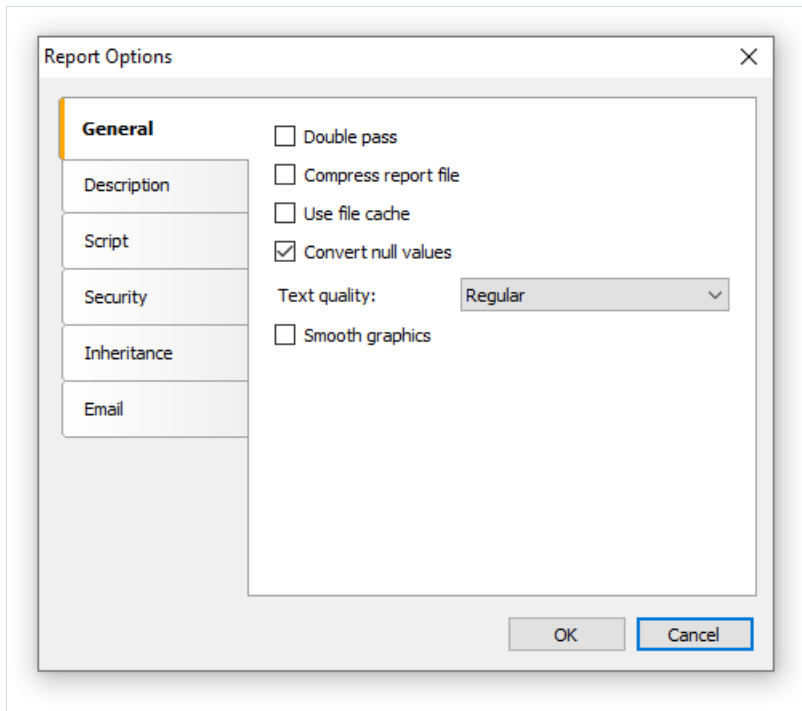
In order to create a report template, a report designer is used. A designer gives the user comfortable facilities for designing the report and allows previewing the report at the same time.

The report designer is the compound part of FastReport and does not depend on the development environment (for example, MS Visual Studio). If you are a software developer, you may include the report designer into your application. This will give your end-users the ability to either change the existing report or create a new one.



# Report options

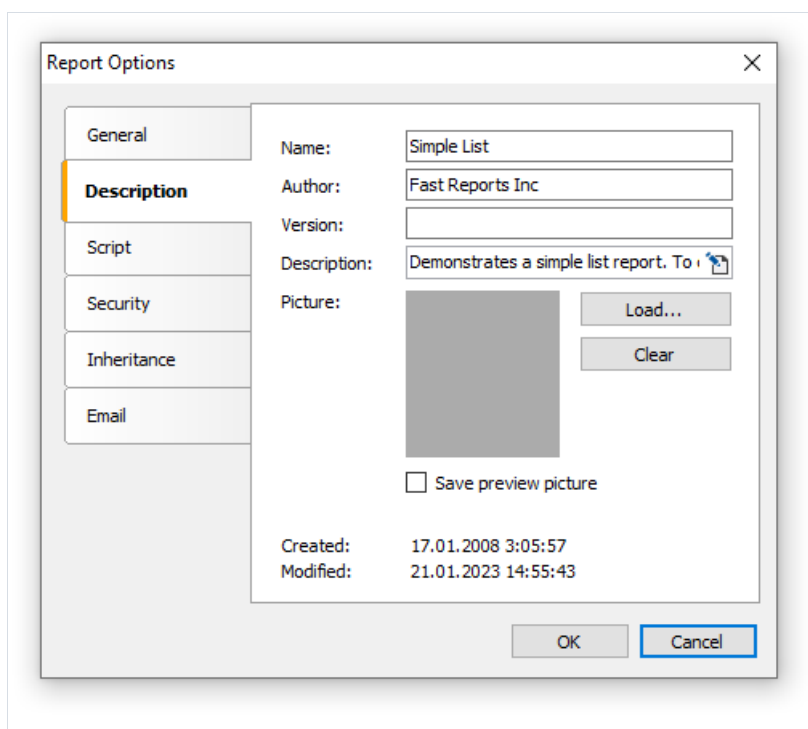
A window with report options can be called in the "Report|Options..." menu. You will see a dialogue window with several tabs:



On the "General" tab, you can control the following report parameters:

- "Double pass" parameter allows to enable two report passes. This can be necessary when you use the "total number of pages" system variable;
- "Compress report file" parameter allows saving a report in a compressed form. For compressing, zip algorithm is used, that is why you can easily extract original contents with the help of any archive;
- "Use file cache" parameter allows to save the memory when creating a report. Use this parameter if your report has got a lot of pages;
- "Convert null values" controls converting the null value data column into the default value (0, empty string, `false` - depending on the data type of a column);
- "Text quality" parameter allows choosing the mode of text displaying in the report. This mode does not affect printing of the report;
- "Smooth graphics" parameter allows to enable the smooth mode when drawing graphical objects (line, border, picture).

On the "Description" tab, you can give the description of the report. All these parameters are not obligatory, and they serve for informational purposes:



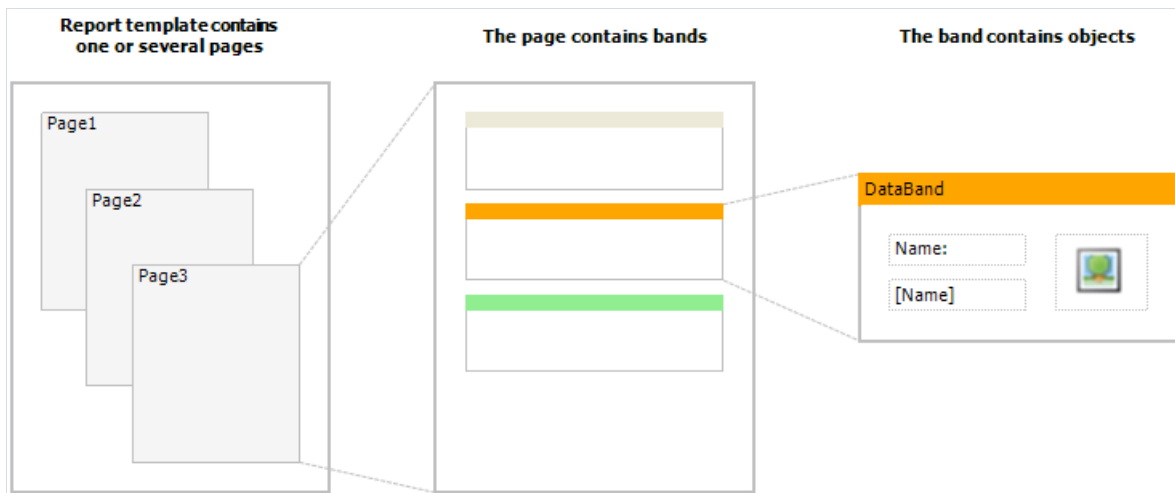
On the "Script" tab, you can choose the script language for the report. Detailed work with script can be found in the ["Script"](#) chapter.

On the "Security" tab you can give the password which will be requested when opening the report. A report which has a password, is saved in an encoded form, so do not forget your password! Restoring a report in this case will be practically impossible.

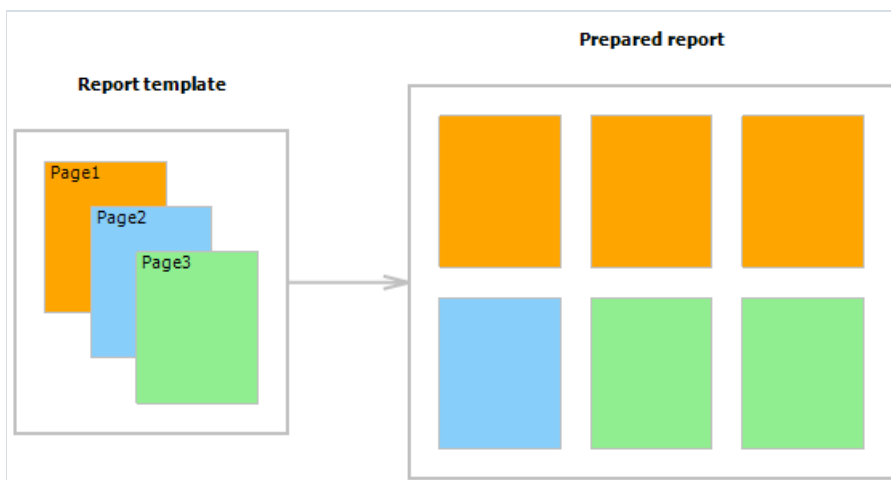
On the "Inheritance" tab, you can control report inheritance. This functionality will be looked at later.

# Report pages

Template consists of one (mostly) or several report pages. Report page, in turn, contains bands. Report objects like Text, Picture and others are placed on the band:





Report template can consist of several pages. For example, you can create a template containing title-page and a page with data. When creating such a report, the first page will be printed first, then the second page and so on. Every page of template can generate one or several pages of a prepared report – this depends on the data it contains:

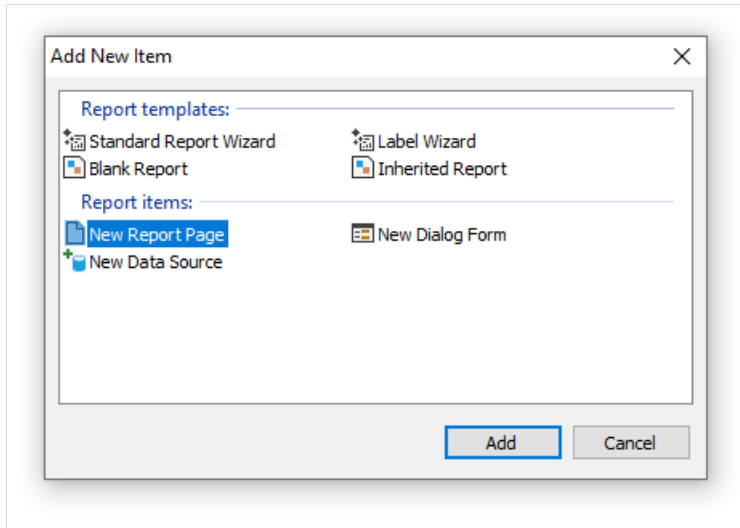



Report pages are also used when working with subreports. Contrary to other report generators, subreports in FastReport are saved in a separate template page, and not in a separate file.

Apart from report pages, a template can contain one or more dialogue forms. Dialogue forms can be used for inquiring some parameters before creating a report. Detailed work with dialogue forms will be covered in the ["Dialogue forms"](#) chapter.

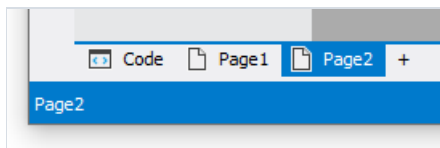
# Managing pages

When you have created a new report, it already contains one page with several bands. For adding a new page, click the  button. A page can also be added by clicking the  button and choosing "New Report Page" in the window.



In a similar way, dialogue forms can be added into the report. For this, use the  button.

Template pages are displayed in the designer as tabs:




The first tab is the report code. It can neither be moved nor deleted.

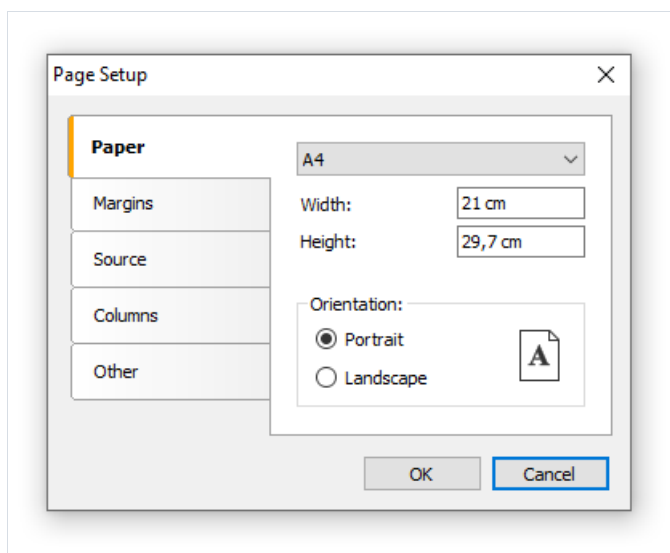
In order to switch to the needed page, simply click on its tab. Changing order of the pages can be done with the help of the mouse. For this, left click on the tab and, without leaving the mouse, move the tab to the desired place.

For deleting a page, click the  button. This button is not active if the report consists of only one page.



# Page properties

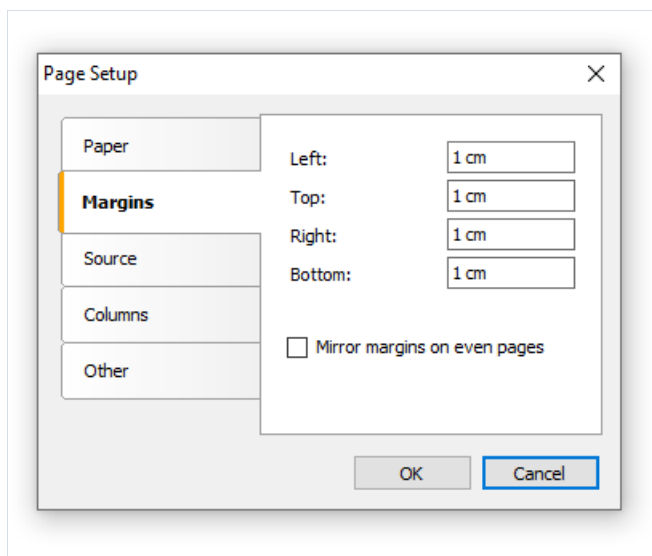
Every report page can have its own settings, such as paper size, orientation (landscape or portrait), margins, columns, paper source and others. Report template can contain several pages with different orientations and paper sizes. The window with page setup can be called by clicking the  button or by choosing the "File|Page setup..." menu item.



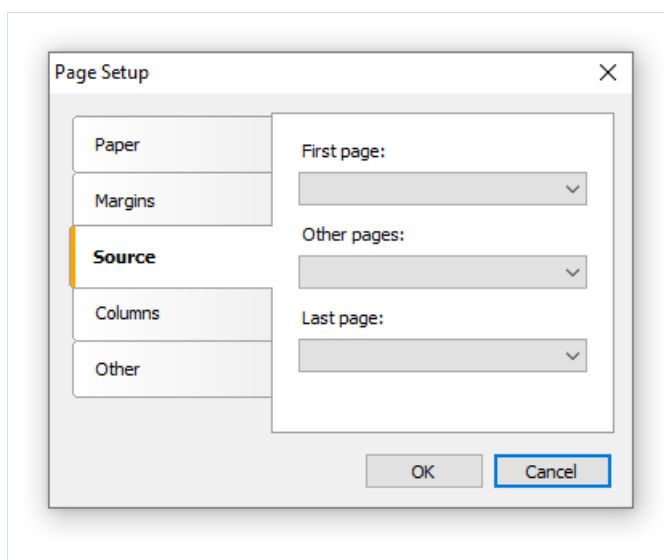
The "Paper" group allows to set the paper size and orientation. It is possible to choose one of the supported sizes, by using the drop-down list. It contains all paper sizes which are supported by the current printer.

Current printer can be configured by using "File|Printer Setup..." menu.

The "Margins" group allows to setup page margins. The "Mirror margins on even pages" options can be used to print booklets:

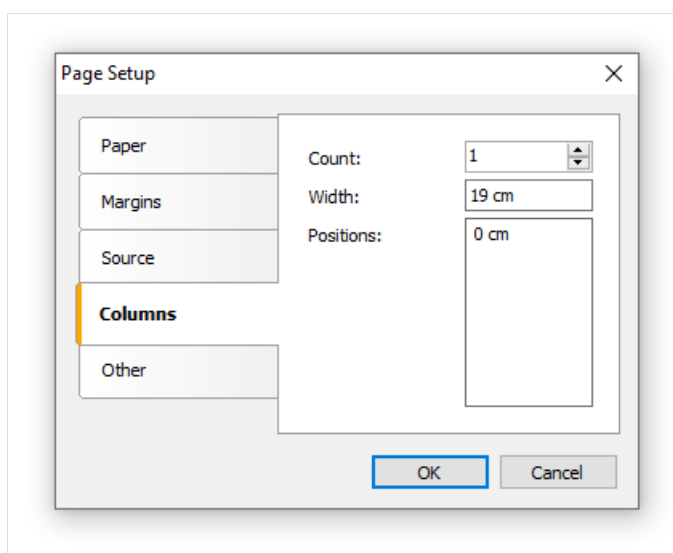


The "Source" group allows choosing the source of the paper. Note that the sources can be given separately, that of the first page of the prepared report, and that of the rest of pages:

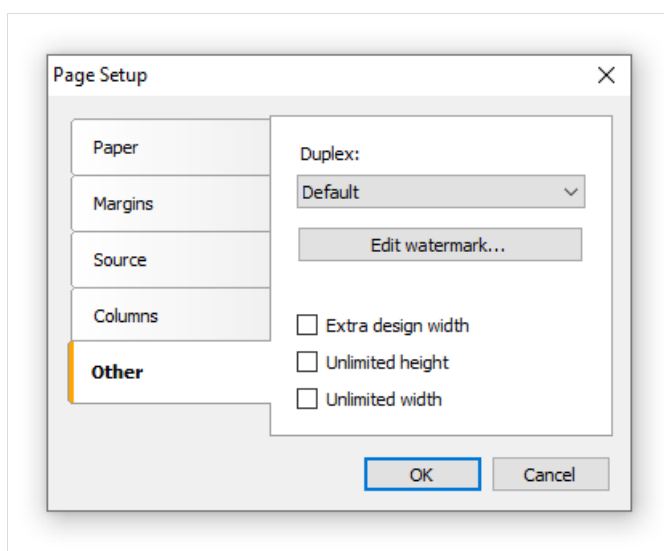


The source can be chosen in the "Print" dialog as well.

The "Columns" group allows setup the column parameters for multi-columned report. For this, the number of columns need to be indicated and (optional) correct the width of the column and the position of every column:



The "Other" group allows giving certain helpful page properties. It is possible to indicate duplex mode for duplex printing if your printer supports this mode. Here it is also possible to set the watermark, which will be printed on prepared report pages:



The "Extra design width" checkbox allows to increase the page width in the design mode. It may be useful if you work with such objects as "Table" or "Matrix".

The duplex mode can be chosen in the "Print" dialog as well.

# Bands

The band is an object which is located directly on the report page and is a container for other objects like "Text", "Picture" and others.

In all, in FastReport there are 13 types of bands. Depending on its type, the band is printed in a certain place in the report.

| Band                  | How it's printed   |
|-----------------------|--|
| <b>Report Title</b>   | It is printed once at the very beginning of the report. You can choose the order of printing - before the "Page Header" band or after it - with the help of the <code>TitleBeforeHeader</code> page property. Changing this property can be done with the help of "Properties" window. By default, property is equal to <code>true</code> , that is, report title is printed before page header. |
| <b>Report Summary</b> | It is printed once at the end of the report, after the last data row, but before the "Page Footer" band.   |
| <b>Page Header</b>    | It is printed on top of every page of the report.  |
| <b>Page Footer</b>    | It is printed at the bottom of every page of the report.   |
| <b>Column Header</b>  | This band is used when printing a multi-columned report (when the number of columns indicated in the page setup > 1). It is printed on top of every column after the Page Header band.   |
| <b>Column Footer</b>  | Printed at the bottom of every column, before the Page Footer band.  |
| <b>Data</b>           | This band is connected to the data source and is printed as many times as there are rows in the source.  |
| <b>Data Header</b>    | This band is connected to the "Data" band and is printed before the first data row.  |
| <b>Data Footer</b>    | This band is connected to the "Data" band and is printed after the last data row.  |
| <b>Group Header</b>   | It is printed at the beginning of every group, when the value of the group condition changes.  |
| <b>Group Footer</b>   | It is printed at the end of every group.   |
| <b>Child</b>          | This band can be connected to any band, including another child band. It is printed immediately after its parent.  |
| <b>Overlay</b>        | Printed as a background on every report page.  |

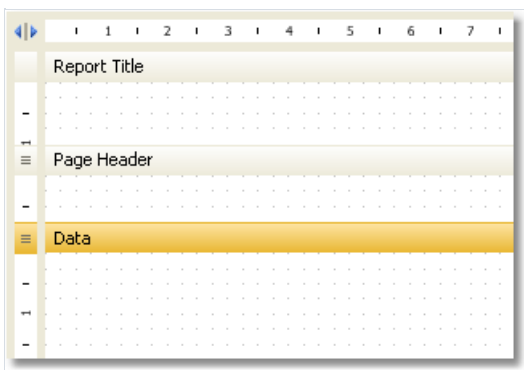
# Bands in designer

A band in the designer appears in form of a rectangular area. A band, like many other report objects, can have a border and fill (by default they are disabled). Apart from this, a band displays a grid. To set the grid mode, go the "View|Options..." menu and choose "Report page". Grid can also be enabled or disabled in the "View" menu.

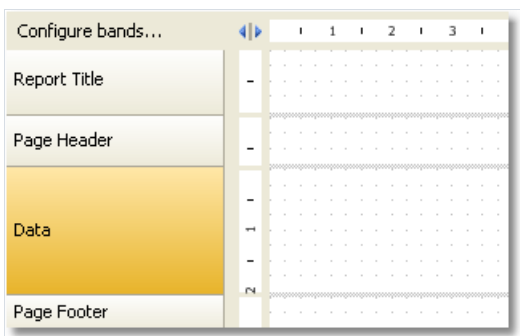
You can set the band's height in three ways:


- place the mouse pointer at the bottom of the band. The cursor shape will be changed to "horizontal splitter" and you can resize a band.
- drag the band handle on the left ruler.
- use "Properties" window to set the band's **Height** property.

The designer has two modes of displaying bands, between which you can switch at any time. In the first mode, every band has got a header, which contains the title of the band and useful information about it (for example, the name of the data source to which it is connected).



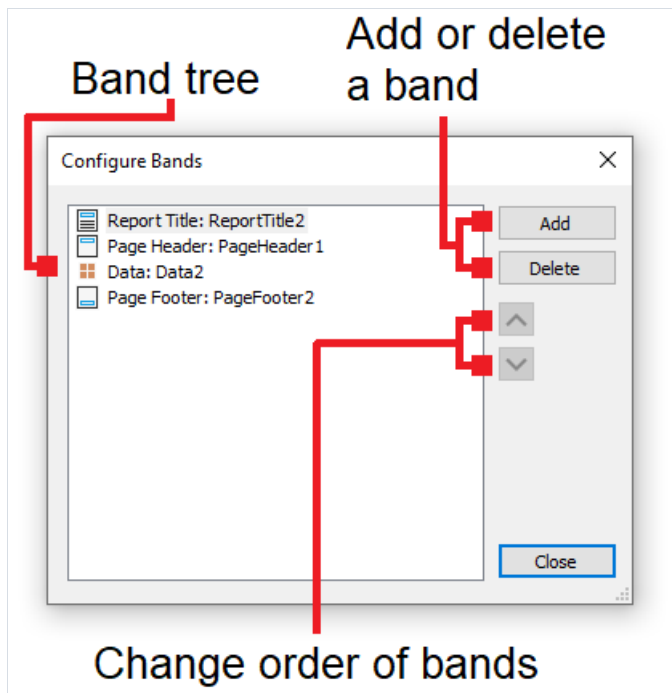
In the second mode, the band does not have a header. Instead of that, on the left side of the window, the structure of the bands is displayed. This mode helps to easily understand the structure of the report, especially if it was not created by you.



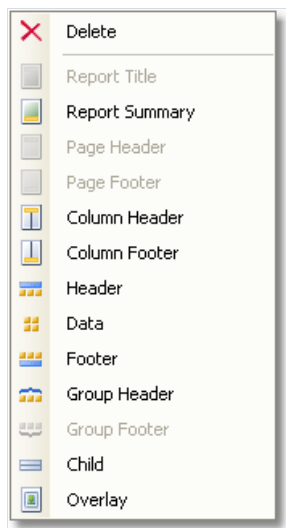
You can switch between these modes by clicking the  button.

# Configuring bands

You can configure the bands in the "Configure Bands" window. It can be called from the "Report|Configure Bands..." menu or with the help of the "Configure bands" button, placed over the bands tree:

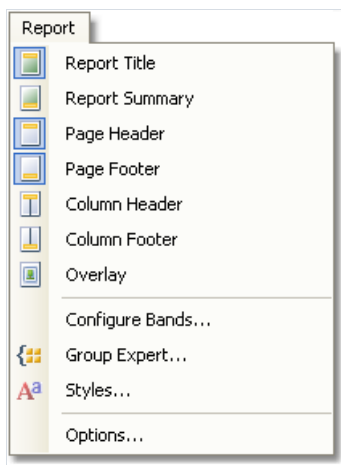


In this window, it is possible to add bands into the report, delete them or change their order. To add a band, click the "Add" button or right click on band tree. A context menu will come up containing a list of bands. A band which cannot be added is dimmed.



The "Add" operation depends on what band was chosen in the band tree. For example, adding "Data Header" and "Data Footer" bands is possible only if the "Data" band was selected beforehand.

There is also another way of configuring some bands. This can be done from the "Report" menu:



To delete a band, select it and press `Delete` key.

When configuring bands, FastReport does not allow to do operations which leads to the creation of a wrong report template. For example, you cannot delete the "Data" band, which is connected to the group - for this, the group needs to be deleted first. Another example, when deleting the "Data" band, its header and footer are deleted automatically. Also, it is not possible to delete a band if it is the only one on the page.

# Printing Bands

So, there are several bands placed on the page. How will FastReport compose a prepared report? Let us look at the following example:

|                 |   |                |
|-----------------|---|----------------|
| Report Title    | - | Report Title   |
| Page Header     | - | Page Header    |
| Data: Employees | - | Data           |
| Report Summary  | - | Report Summary |
| Page Footer     | - | Page Footer    |

The "Report Title" band will be printed first. The "Page Header" band will be printed immediately after it. Further, the "Data" band will be printed as many times as there are rows in the data source, to which the band is connected. After all the rows of the "Data" band have been printed, the "Report Summary" band is printed and at the bottom of the page - the "Page Footer" band. Printing of the report ends with this. A prepared report will be looking something like this:

|              |                |
|--------------|----------------|
| Report Title | Page Header    |
| Page Header  | Data           |
| Data         | Data           |
| Data         | Data           |
| Data         | Data           |
| Data         | Data           |
| Data         | Data           |
| Data         | Data           |
| Data         | Data           |
| Data         | Report Summary |
| Page Footer  | Page Footer    |

In the process of printing, FastReport checks if there is enough space on the current page of the prepared report, so that the band can be printed. If there isn't enough space, the following occurs:

- page footer is printed;
- a new page is added;
- page header is printed;
- continues to print the band which did not fit on the previous page.




















# Band properties

Every band has several useful properties, which affect the printing process. They can be configured by using the band's context menu. To do this, right-click on an empty space on the band, not occupied by other objects. Also, it is possible by clicking on the band header (if the classic display mode is used) or on band structure (otherwise). Another method – choose a band and change the corresponding properties in the "Properties" window.

| Property                  | Description  |
|---------------------------|--|
| <b>CanGrow, CanShrink</b> | These properties determine whether a band can grow or shrink depending on the size of the objects contained in the band. If both properties are disabled, the band will always have the size specified in the designer. Read more about this in the <a href="#">"Report Creation"</a> chapter. |
| <b>CanBreak</b>           | If the property is enabled, FastReport tries to print a part of the band's contents on the available space, that is, "break" the band. Read more about this in the <a href="#">"Report Creation"</a> chapter.  |
| <b>StartNewPage</b>       | Printing a band with such property begins on a new page. This property is usually used when printing groups; that is, every group is printed on a new page.  |
| <b>PrintOnBottom</b>      | A band with this property is printed at the bottom of the page, before the "Page Footer" band. This can be useful when printing certain documents, where the total sum is supposed to be printed at the bottom of the page.  |
| <b>RepeatOnEveryPage</b>  | The bands - "Data Header", "Data Footer", "Group Header" and "Group Footer" - have got this property. This type of band will be printed on each new page, when data printing is being done. Read more about this in the <a href="#">"Report Creation"</a> chapter.                             |

# Report objects fundamentals

A wide range of objects can be used in the report:

| Icon  | Title  | Description  |
|---|--|--|
|    | "Text" (TextObject)  | Shows one or more lines of text.   |
|    | "Picture" (PictureObject)  | Shows a picture.   |
|    | "SVG" (SVGObject)  | Shows a vector image.  |
|    | "Line" (LineObject)  | Shows a line. The line can be vertical, horizontal, or diagonal.                                   |
|    | "Shape" (ShapeObject)  | Shows one of the geometric shapes - rectangle, ellipse, triangle, etc.                             |
|    | "PolyLine" (PolyLineObject)  | Shows a polyline with a different number of vertices that supports curvature of the line segments. |
|   | "Polygon" (PolygonObject)  | Shows a polygon with varying numbers of vertices and curvature support.                            |
|  | "Rich text" (RichObject)   | Shows rich text (RTF format).  |
|  | "Barcode" (BarcodeObject)  | Shows a barcode.   |
|  | "Checkbox" (CheckBoxObject)  | Shows a checkbox that can have two states - "on" or "off".   |
|  | "Table" (TableObject)  | Shows a table consisting of rows, columns, and cells.  |
|  | "Matrix" (MatrixObject)  | Shows a matrix (also known as pivot table, cross-tab).   |
|  | "Chart"(MSChartObject)   | Shows a chart.   |
|  | "Zipcode" (ZipCodeObject)  | Shows a zip code.  |
|  | "Cellular text"(CellularTextObject)  | Shows a text, where each character is written in a separate cell.                                  |
|  | Gauge objects (LinearGauge, SimpleGauge, RadialGauge, SimpleProgressGauge) | Several objects with similar functionality are designed to visualize a value.                      |
|  | Digital Signature (DigitalSignatureObject)                                 | Shows a visible field that can be used for electronic signatures.                                  |

Objects can be used both to display information (the "Text" object) and to design a report (the "Picture", "Line",

"Shape" objects). Complex objects such as "Table" and "Matrix" can contain other simple objects.

# Common object properties

All report objects are inherited from one basic class ( `ReportComponentBase` ) and have got certain set of common properties. Before studying each object, we will look at these properties.

You can change the value of properties with the help of the "Properties" window. Some properties can be changed using the object's context menu or toolbars (for example, border and fill).

| Property                        | Description   |
|---------------------------------|---|
| <b>Left, Top, Width, Height</b> | A report object, as a rule, is a rectangle. It has coordinates (properties <code>Left</code> , <code>Top</code> ) and size (properties <code>Width</code> , <code>Height</code> ).  |
| <b>Anchor</b>                   | This property determines how the object will be changing its position and/or its size when the container on which it is laying grows or shrinks. By using <code>Anchor</code> , it can be done in such a way that, the object expands or moves synchronously with container. Read more about this property in the <a href="#">"Dynamic layout"</a> chapter. |
| <b>Dock</b>                     | This property determines on which side of the container the object will be docked. Read more about this property in the <a href="#">"Dynamic layout"</a> chapter.   |
| <b>Border, Fill</b>             | These properties control the object's border and fill. They can be changed using the "Border and Fill" toolbar.   |
| <b>CanGrow, CanShrink</b>       | These properties allow fitting the height of the object in such a way that it fits the whole text. Read more about this property in the <a href="#">"Dynamic layout"</a> chapter.   |
| <b>ShiftMode</b>                | An object, whose property is enabled, will be moving up and down, if the object above on can either grow or shrink. Read more about this property in the <a href="#">"Dynamic layout"</a> chapter.  |
| <b>GrowToBottom</b>             | An object, whose property is enabled, will be stretched to the bottom of a band. Read more about this property in the <a href="#">"Dynamic layout"</a> chapter.   |
| <b>CanBreak</b>                 | Objects "Text" and "Rich Text" have this property. It determines whether the object's contents can be broken into parts.  |
| <b>PrintOn</b>                  | This property determines on which pages the object can be printed. Read more about this property in the <a href="#">"Booklet-type report"</a> chapter.  |
| <b>Cursor</b>                   | This property determines the type of mouse cursor when it is located over an object. The property works only in the preview window.   |
| <b>Visible</b>                  | This property determines whether the object will be displayed in the report. Invisible object is never displayed in the preview window and is never printed on the printer as well.   |
| <b>Printable</b>                | This property determines whether the object will be printed on the printer. If this property is disabled, then the object will be visible in the preview window but it will not be printed.   |
| <b>Hyperlink</b>                | This property makes it possible to make the report object interactive. Read more about this property in the <a href="#">"Interactive reports"</a> chapter.  |

| Property            | Description   |
|---------------------|---|
| <b>Bookmark</b>     | This property is used together with the <code>Hyperlink</code> property. It can contain any expression. The expression will be calculated when the report will be working, and its value will be used as bookmark's name. |
| <b>Restrictions</b> | This property restricts certain operations, such as moving, resizing, deleting the object.  |
| <b>Style</b>        | You can assign the style name to this property. When this is done, the object will become like it has been indicated in the style. If the parameters of the style changes, the appearance of the object changes as well.  |

# The "Text" object

The "Text" object is the main object which you will use often. It looks like this:

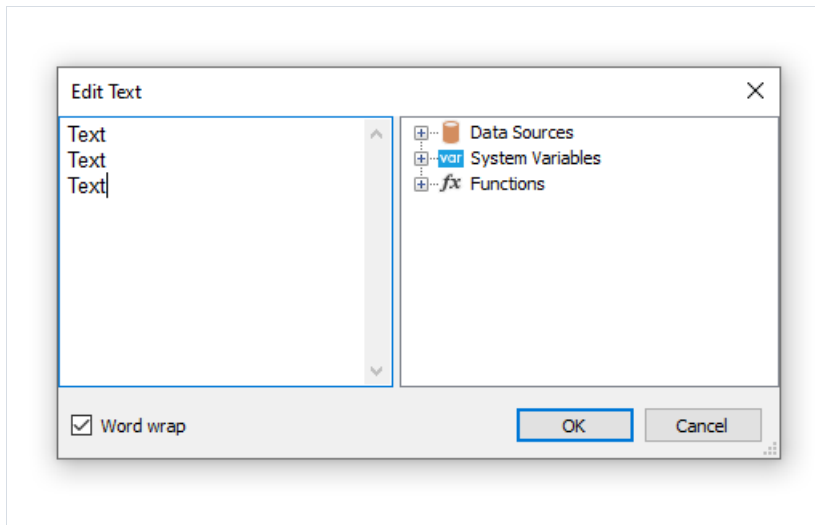


The object can display any text data, specifically:

- one or several text lines;
- data columns;
- report parameters;
- total values;
- expressions;
- any combination of the above items.

# Text editing

In order to edit an object's text, just double click on it. You will see a text editor:



There is a data tree on the right side of the editor, which elements can be added into the text. This can be done by dragging the element onto needed place by using the mouse. Another way to insert an element into the text - double click on the element, and it will be added onto the cursor's current position.

In order to save the changes and close the editor window, click the OK button or press the **Ctrl+Enter** keys.

Another method of editing a text - in-place editing. To do this, select the "Text" object and press Enter. To finish editing, click somewhere outside the objects bounds or press **Ctrl+Enter**. Press **Esc** key to cancel the changes.

When editing an object in-place, its size can be changed by using a mouse.

# Displaying the expressions

The "Text" object can contain a plain text mixed with expressions. For example:

```
Today is [Date]
```

When printing such an object, all expressions contained in the text will be calculated. So the result may look like this:

```
Today is 12.09.2010
```

As seen, expressions are identified by square brackets. This is set in the `Brackets` property, which by default contains the string `[,]`. When needed, you can use a different set of symbols, for example `<, >`, or `<!, !>`. In the last case, an expression in the text will be like this:

```
Today is <!Date!>
```

Apart from this, it is possible to disable all expressions. To do this, set the `AllowExpressions` property to false. In this case the text will be shown "as is".

Inside the square brackets, you can use any valid expression. Read more about expressions in the ["Expressions"](#) chapter. For example, an object with the following text:

```
2 * 2 = [2 * 2]
```

will be printed like this:

```
2 * 2 = 4
```

Frequent mistake - attempt to write an expression outside the square brackets. Reminding, that it is considered an expression and gets executed only that, which is located inside square brackets. For example:

```
2 * 2 = [2] * [2]
```

This text will be printed this way:

```
2 * 2 = 2 * 2
```

There may be elements inside an expression that needs own square brackets. For example, it may be a reference to a system variable (see the ["Expressions"](#) chapter for details). Let's look at the following example:



The next page: [[Page] + 1]

The text contains an expression `[Page] + 1`. `Page` is a system variable which returns the number of the current report page. It is included in own brackets. These brackets must be square brackets, regardless of the "Text" object settings.

Strict speaking, we were supposed to use two pairs of square brackets when using the `Date` system variable in the examples above:

Today is [[Date]]

However FastReport allows to leave out unnecessary pair of brackets if there is only one member in an expression.

# Displaying the data columns

You can print the data column in the following way:

```
[Datasource name.Column name]
```

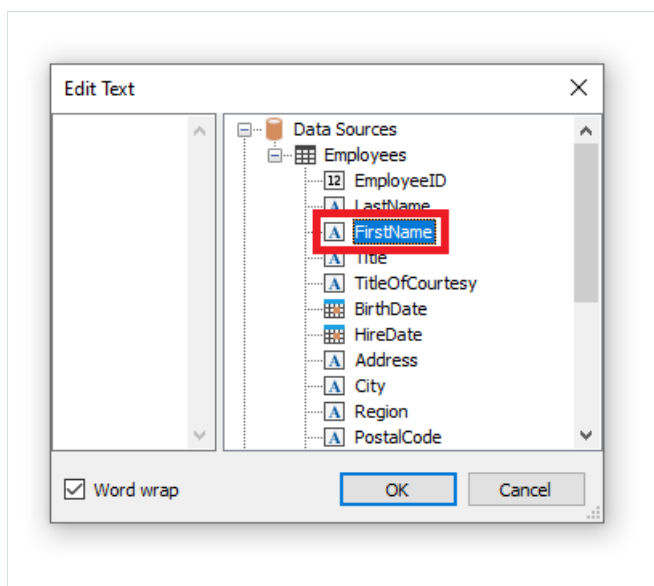
As you can see, the square brackets are used here. The data source name and data column name are separated by the period. For example:

```
[Employees.FirstName]
```

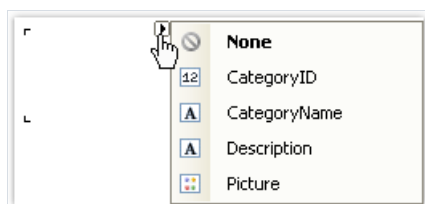
Read more about using the data columns in the ["Expressions"](#) chapter.

There are several ways to insert a data column into the "Text" object.

1. In the "Text" object's editor we write the name of the data column manually. This method is the most inconvenient as it is easy to make a mistake.
2. In the object's editor we choose the needed data column and drag&drop it into the text:



3. Click on the small button in the upper right corner of the object and choose the data column from a list:



4. Drag&drop a data column from the "Data" window into the report page. In this case the "Text" object is created which contains a link to the column.

# Support for HTML tags in text

Previously, in FastReport, in the "Text" object, it was possible to use some simple HTML tags using the `HtmlTags` property. However, the `HtmlTags` property has been replaced by a new `TextRenderType` property. The new property provides a wider range of functionality than just processing HTML tags.

`TextRenderType` property has three possible values:

- **Default** - just text, no tag conversion;
- **HtmlTags** - application of HTML tags. Their list is quite limited: `<b>`, `<i>`, `<u>`, `<strike>`, `<br>`, `<sub>`, `<sup>`, `<img>`;
- **HtmlParagraph** - allows you to adjust the line spacing, red line and all the same tags as **HtmlTags**;
- **Inline** - displays text in a simple format. For internal use only - not recommended for external use.

One of the tags available in the previous `HtmlTags` property was the `<font>` tag. This tag is now deprecated and is not guaranteed to be supported in all browsers. To this end, a new rendering engine has been introduced that allows specific CSS styles to be applied using the `style` attribute for the `<span>` tag.

Let's take a closer look at the HTML tag processing modes:

## HtmlTags

As previously mentioned, the "Text" object supports the following HTML tags:

- `<b>` - highlight text in bold

Usage example:

```
<b>FastReport</b>
```

Result:

**FastReport**

- `<i>` - Italic font style

Usage example:

```
<i>FastReport</i>
```

Result:

*FastReport*

- `<u>` - underline text

Usage example:

```
<u>FastReport</u>
```

Result:

~~FastReport~~

- `<strike>` - strikethrough text

Usage example:

```
<strike>FastReport</strike>
```

Result:

Fast  
Report

- `<br>` - line break

Usage example:

```
Fast<br>Report
```

Result:

Fast<sub>Report</sub>

- `<sub>` - displaying text in the subscript

Usage example:

```
Fast<sub>Report</sub>
```

Result:


Fast<sup>Report</sup>

- `<sup>` - displaying text in the superscript

Usage example:

```
Fast<sup>Report</sup>
```

Result:

Fast

- `<img>` - inserting an image into text

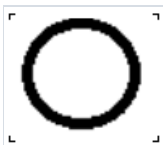
The `<img>` tag only supports the `src` attribute, which can contain a link to the image (http, https, base64), as well as the `width` and `height` attributes, which specify the image's dimensions in pixels. In this way, images can be inserted directly into the text. Image links with http and https protocols are applicable for web reports.

Usage example:

```

```

Result:



## HtmlParagraph

This mode includes a new HTML tag handler. It handles the same set of tags as `HtmlTags` and also includes a `<span>` tag. Despite the fact that this mode has almost the same set of tags, they are rendered differently. This is particularly noticeable on `<sub>` and `<sup>` tags.

The `<span>` tag provides the ability to style text using the `style` attribute. This allows you to apply various CSS styles, such as text color, font size, alignment, and other formatting options, directly to a specific piece of text. So, instead of using the legacy `<font>` tag, which limits functionality and lacks flexibility, you can now achieve the same effect by using the `<span>` tag with basic CSS styles in the `style` attribute.

Usage examples:

```
<span style="font-size:20pt;">FastReport</span>  
<span style="color:red;">Fast</span>Report  
<span style="font-family:Consolas;">FastReport</span>  
<span style="background-color:yellow;">FastReport</span>
```

Result:



## ParagraphFormat

It is worth considering the `ParagraphFormat` property. It works in conjunction with the `HtmlParagraph` property and provides settings for displaying paragraphs (line spacing, indented line). Namely:

- **FirstLineIndent** - first line indent;
- **LineSpacing** - distance between lines in centimeters;
- **LineSpacingMultiple** - multiplication factor by the value of the previous parameter. Works with the Multiple type;
- **LineSpacingType** - line spacing type:
  - Single;
  - At least;
  - Exact;
  - Multiple.

Example settings:

| ▼ ParagraphFormat   | (ParagraphFormat) |
|---------------------|-------------------|
| FirstLineIndent     | 1,25 cm           |
| LineSpacing         | 2,38 cm           |
| LineSpacingMultiple | 0,9               |
| LineSpacingType     | Multiple          |

Result:

Lorem Ipsum is simply dummy text of the printing and typesetting industry.  
 Lorem Ipsum has been the industry's standard dummy text ever since the 1500s, when an unknown printer took a galley of type and scrambled it to make a type specimen book. It has survived not only five centuries, but also the leap into electronic typesetting, remaining essentially unchanged.  
 It was popularised in the 1960s with the release of Letraset sheets containing Lorem Ipsum passages, and more recently with desktop publishing software like Aldus PageMaker including versions of Lorem Ipsum.

# Object's properties

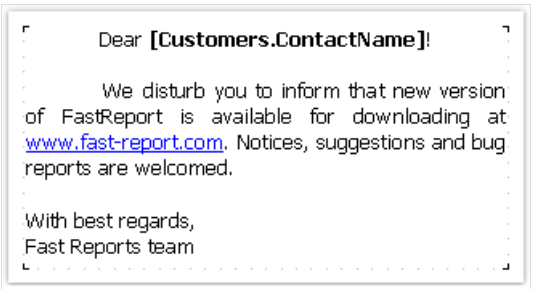
| Property                    | Description   |
|-----------------------------|---|
| <b>AllowExpressions</b>     | This property allows to turn on or off the expression handling. It is on by default.  |
| <b>Angle</b>                | This property indicates the text rotation, in degrees.  |
| <b>AutoShrink</b>           | This property allows to shrink the font size or font width automatically to fit the text.   |
| <b>AutoShrinkMinSize</b>    | This property determines the minimum size of a font, or the minimum font width ratio, if the <code>AutoShrink</code> property is used.  |
| <b>AutoWidth</b>            | This property allows to calculate the width of object automatically.  |
| <b>Brackets</b>             | This property contains a pair of symbols that designate an expression.  |
| <b>BreakTo</b>              | With this property you can organize the text flow from one text object to another.<br>For example, we have "A" and "B" text objects. The "A" object contains the long text that does not fit in the object's bounds. You can set the <code>A.BreakTo</code> to B, so the "B" object will display the part of text that does not fit in "A". |
| <b>Clip</b>                 | This property determines whether it is necessary to clip a text outside of object's bounds. It is on by default.  |
| <b>Duplicates</b>           | This property determines how the repeated values will be printed. Read more about this property in the <a href="#">"Formatting"</a> chapter.  |
| <b>FirstTabOffset</b>       | This property determines the offset of the first TAB symbol, in pixels.   |
| <b>Font</b>                 | Font settings.  |
| <b>FontWidthRatio</b>       | Use this property to make the font wider or narrower. By default the property is set to 1. To make the font wider, set the property to value > 1. To make the font narrower, set the property to value between 0 and 1.   |
| <b>HideValue</b>            | This property is of string type. It allows to hide values that are equal to the value of this property. Read more about this property in the <a href="#">"Formatting"</a> chapter.  |
| <b>HideZeros</b>            | This property allows to hide zero values. Read more about this property in the <a href="#">"Formatting"</a> chapter.  |
| <b>Highlight</b>            | This property allows to setup the conditional highlight. Read more about this in the <a href="#">"Formatting"</a> chapter.  |
| <b>HorzAlign, VertAlign</b> | These properties determine the text alignment.  |
| <b>HtmlTags</b>             | Deprecated, replaced by <code>TextRenderType</code> .   |
| <b>LineHeight</b>           | This property allows to explicitly set the height of a text line. By default it is set to 0, so the default line spacing is used.   |

| Property               | Description  |
|------------------------|--|
| <b>MergeMode</b>       | Combining text objects with the same content that are located next to each other. Contains 2 values:<br><ul style="list-style-type: none"> <li>- <code>Horizontal</code> – horizontal union;</li> <li>- <code>Vertical</code> – vertical union.</li> </ul> It is acceptable to use both separately and jointly.<br><i>Important note! The objects should be close to each other.</i> |
| <b>NullValue</b>       | The text that will be printed instead of a null value. You also need to uncheck the "Convert null values" option in the "ReportOptions..." menu.   |
| <b>Padding</b>         | This property allows to setup the padding, in pixels.  |
| <b>ParagraphFormat</b> | Settings for displaying paragraphs (line spacing, red line indentation). Read more about this property in the <a href="#">"Support for HTML tags in text"</a> chapter.   |
| <b>ParagraphOffset</b> | Paragraph offset, measured in pixels. For <code>TextRenderType.HtmlParagraph</code> , use the <code>ParagraphFormat.FirstLineIndent</code> property.   |
| <b>RightToLeft</b>     | This property indicates whether the text should be displayed in right-to-left order.   |
| <b>TabPositions</b>    | A set of tab character positions, in pixels. Negative values do not affect this property.  |
| <b>TabWidth</b>        | This property determines the width of the TAB symbol, in pixels.   |
| <b>Text</b>            | This property contains the text of the object.   |
| <b>TextFill</b>        | This property determines the text fill. Use this property editor to choose between different fill types.   |
| <b>TextOutline</b>     | Border around text for emphasis or styling.  |
| <b>TextRenderType</b>  | This property allows you to use HTML tags in the object text. Read more about this property in the <a href="#">"Support for HTML tags in text"</a> chapter.  |
| <b>Trimming</b>        | This property determines how to trim the text that does not fit inside the object's bounds. It is used only if the <code>WordWrap</code> property is set to <code>false</code> .   |
| <b>Underlines</b>      | This property allows to display a graphical line after each text line. This property can be used only if the text is top-aligned.  |
| <b>WordWrap</b>        | This property determines whether it is necessary to wrap a text by words.  |
| <b>Wysiwyg</b>         | This property changes the display mode of the "Text" object to match the screen and the final printout. This mode is also used if you use the justify-align or non-standard line height.   |



# The "Rich Text" object

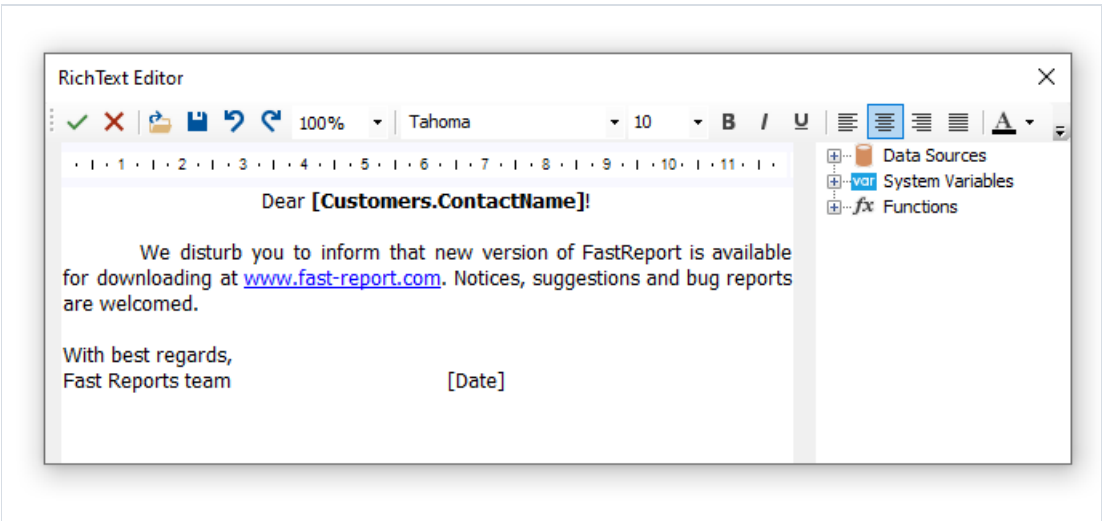
This object displays the formatted text (in the RTF format). It looks like this:




Try to use the "Text" object to display a text. When you export the report to some document formats, the "Rich Text" object will be exported as a picture.

This object supports only the solid fill type. Gradient, hatch, glass fills are not supported.

To edit the object, double click on it. You will see the editor window:



You can also use the Microsoft Word to create a text. When you have done, save the text in the .RTF format. Next, you need to open the "Rich Text" editor and load the .RTF file into it by pressing the  button.

The "Rich Text" object does not support all of the Microsoft Word formatting features.

You can display a data in this object the following ways:

- you can insert an expression in the object's text, just as you do this in the "Text" object. Insert the necessary data column into the text;
- use the `DataColumn` property to bind the object to the column.

The object has the following properties:

| Property         | Description  |
|------------------|--|
| AllowExpressions | This property allows to turn on or off the expression handling. It is on by default. |

| Property          | Description  |
|-------------------|--|
| <b>Brackets</b>   | This property contains a pair of symbols that designate an expression. |
| <b>DataColumn</b> | The data column that this object is bound to.                          |
| <b>Text</b>       | This property contains the RTF text.                                   |
| <b>Padding</b>    | The padding, in pixels.  |

# The "Picture" object

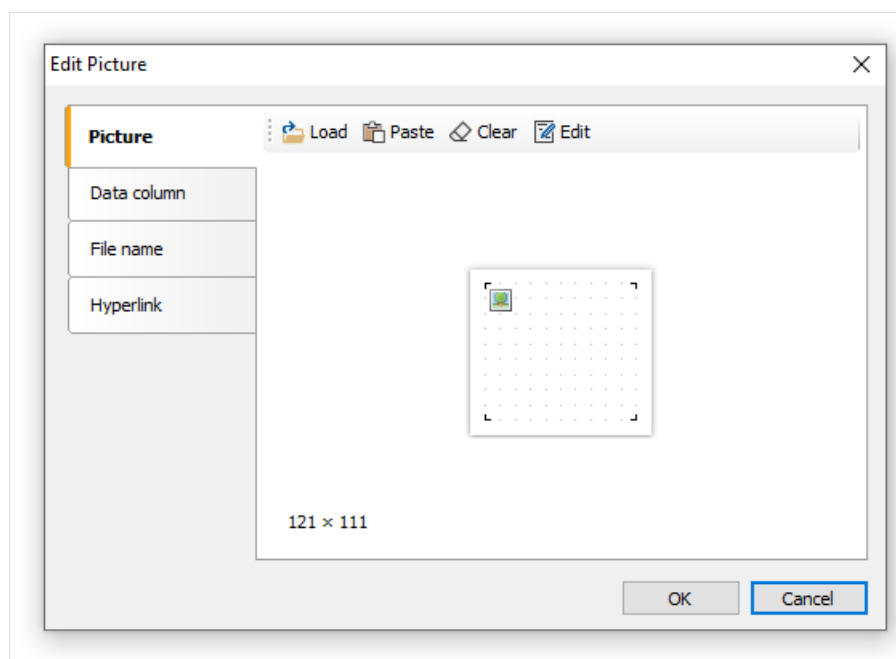
An object can display graphics in the following formats: BMP, PNG, JPG, GIF, TIFF, ICO, EMF, WMF. With the help of the "Picture" object, you can print your company logo, a photo of employee or any graphical information. The object looks like this:



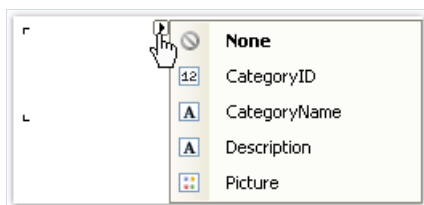
An object can show data from the following sources:

| Source                     | Description  |
|----------------------------|--|
| <b>File with a picture</b> | Picture is loaded from a file and is saved inside the report. Picture is stored in the <code>Image</code> property.  |
| <b>Data column</b>         | Picture from a data column. Name of the column is stored in the <code>DataColumn</code> property.  |
| <b>File name</b>           | Picture is loaded from a file with the given name. Name of file is stored in the <code>ImageLocation</code> property. Picture is never stored inside the report. You should distribute the picture file along with the report. |
| <b>URL</b>                 | Picture is loaded from the internet every time the report is created. Image is never stored inside the report. URL is stored in the <code>ImageLocation</code> property.   |

In order to call a picture editor, double click on the object. In the editor, you can choose the data source for the picture:



In order to bind the object to a data column, click on the small button in the upper right corner of the object and choose the data column from a list:

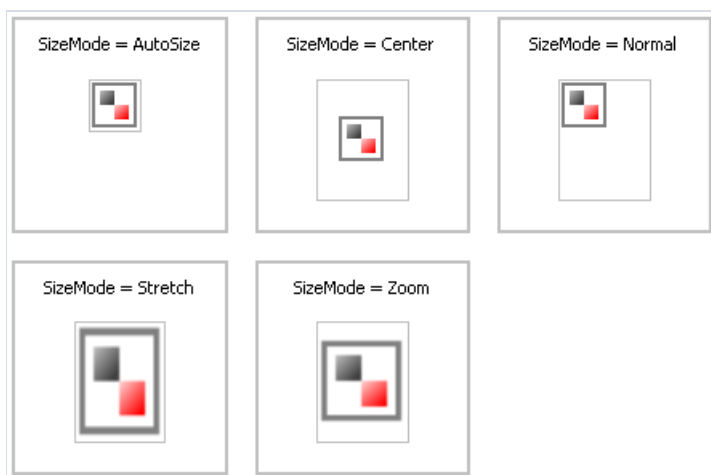


You also can drag&drop a data column from the "Data" window into the report page. In this case the "Picture" object is created which contains a link to the column. The column you drag should have the `byte[]` data type.

In the context menu of the "Picture" object you can choose the size mode:

- AutoSize - The object gets the size of the picture;
- CenterImage - The picture is centered inside the object;
- Normal - The picture is displayed in the left corner of the object;
- StretchImage - The picture is stretched to the size of the object;
- Zoom - The picture is stretched to the size of the object in accordance with the aspect ratio.

The difference between modes is shown in the following picture:



The "Picture" object has the following properties:

| Property                | Description  |
|-------------------------|--|
| <b>Angle</b>            | The rotation angle, in degrees. Possible values for this property are 0, 90, 180, 270.   |
| <b>SizeMode</b>         | The size mode.   |
| <b>Transparency</b>     | The degree of transparency of the pictures. The property can have values between 0 and 1. The value 0 (by default) means that the picture is opaque. |
| <b>TransparentColor</b> | The color which will be transparent when displaying the picture.   |
| <b>Image</b>            | The picture.   |
| <b>DataColumn</b>       | The data column that this object is bound to.  |
| <b>ImageLocation</b>    | This property can contain name of the file or URL. The picture will be loaded from this location when building the report.                           |
| <b>Padding</b>          | The padding, in pixels.  |

| Property                | Description   |
|-------------------------|---|
| <b>ShowErrorMessage</b> | Shows the "No picture" picture, in case when the picture is empty. This property makes sense to use if the picture is downloaded from the Internet. |

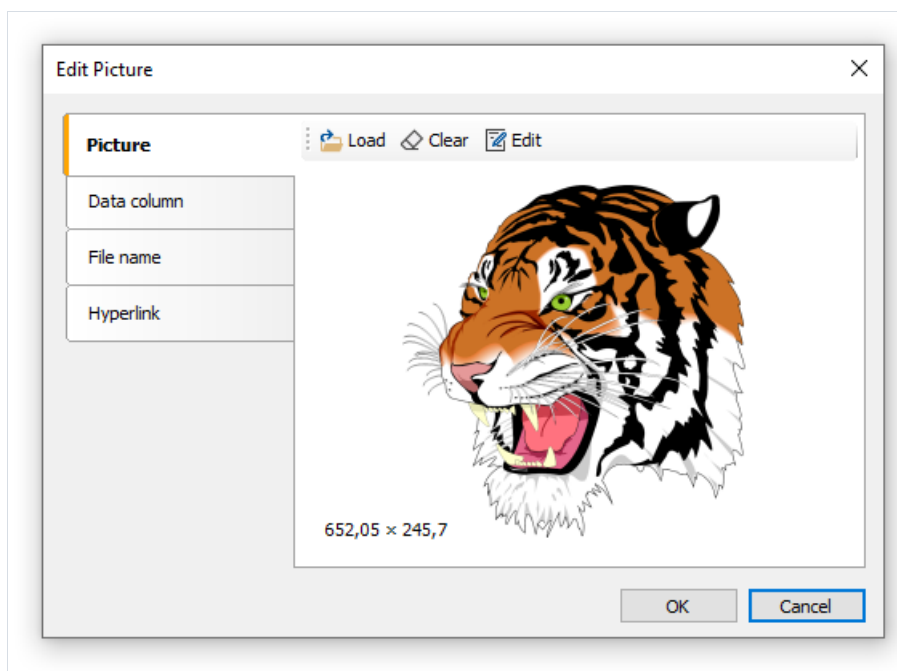
# SVG object

This object is used for displaying vector images in SVG format. An example of such an image:



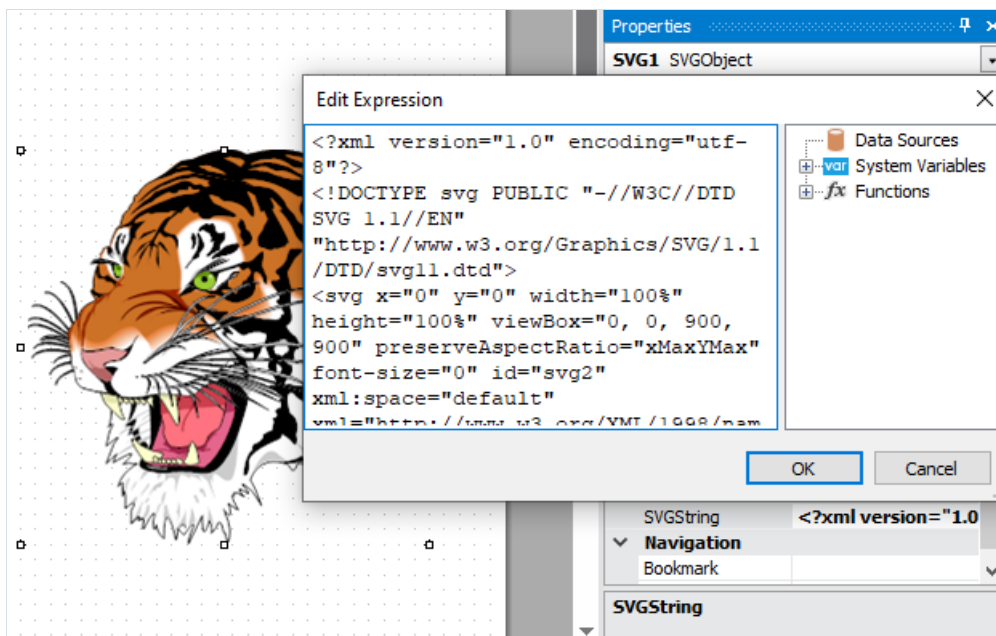
To add an SVG image, add an SVG object to your report using the following button in the object panel: 

After that click on the object twice, or select "Edit" in the context menu. This will open the Image Editor, which is similar to the Image Object Editor.



In this editor you can set the image that will be shown by the object. You can include an image in the report template (the "Open" button on the "Image" tab), select an SVG image from the data table (the "Data Field" tab), establish a connection to an external SVG file (the "File name" tab, in this case you need to distribute this file together with the template) or set a hyperlink from the address from which the image will be obtained (the "Hyperlink" tab).

If you include an image in the report template, its string representation will be stored in the `SVGString` property. The screenshot shows a fragment of such a line:

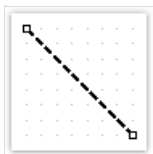


The SVG object supports the same sizing modes as the Picture object: **AutoSize** , **CenterImage** , **Normal** , **StretchImage** , **Zoom** .



A detailed description of these modes is given in the description of the "Picture" object.

# The "Line" object

The "Line" object can display horizontal, vertical or diagonal line. The object is as follows:



If possible, use the object's border instead of "Line" object. This will simplify the report and avoid possible problems with the export to different formats.

FastReport designer has convenient tools for drawing a line. In order to insert a line into a report, click the  button on the "Objects" toolbar and in the menu choose the "Line" object or "Diagonal Line". Place the mouse cursor at the location where the line will start from. Then left click and hold the mouse, in order to draw the line. After this, you can draw the line again. When all the lines have been drawn, click on the  button on the "Objects" toolbar.

An ordinary line differs from a diagonal line in that you can make it only vertical or horizontal.

Do not choose the "Double" line style for this object. This style applies only to the object's border.

The "Line" object has the following properties:

| Property                | Description  |
|-------------------------|--|
| <b>Diagonal</b>         | The property determines weather the line is diagonal. An ordinary line can be turned into a diagonal one by enabling this property.  |
| <b>StartCap, EndCap</b> | These properties allow to setup the line caps. You can use one of the following cap styles: <ul style="list-style-type: none"><li>- ellipse;</li><li>- rectangle;</li><li>- diamond;</li><li>- arrow.</li></ul> Size of the cap can be set in the <b>Width</b> , <b>Height</b> properties of the cap. You can configure caps for each end of the line. |

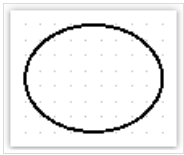



# The "Shape" object

The "Shape" object displays one of the following shapes:

- rectangle;
- rounded rectangle;
- ellipse;
- triangle;
- diamond.

The object is as follows:



In order to insert a shape into the report, click the  button on the "Objects" toolbar and choose the needed shape type.


The shape, like any other report object, has a fill and border. Contrary to the "Text" object, you cannot control each border line. Also, don't use the "Double" line style.

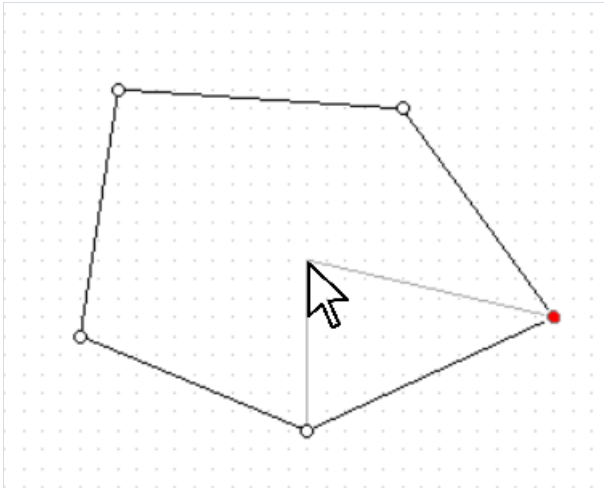
Instead of rectangular shape, use the object's borders if possible.

The "Shape" object has the following properties:

| Property     | Description  |
|--------------|--|
| <b>Shape</b> | This property determines the type of shape.  |
| <b>Curve</b> | This property is used with the "RoundRectangle" shape. It allows to set the curve. |

# Polyline and polygon objects






These objects are used to display a variety of shapes such as polygons. They are in the same category as the Shape and Line objects. To place the Polygon object, open the menu under the  and select one of the polygons. You can place a pre-made five-, six-, seven- or octagon. Alternatively, you can select Polygon to draw the shape yourself. An example of what a polygon might look like:

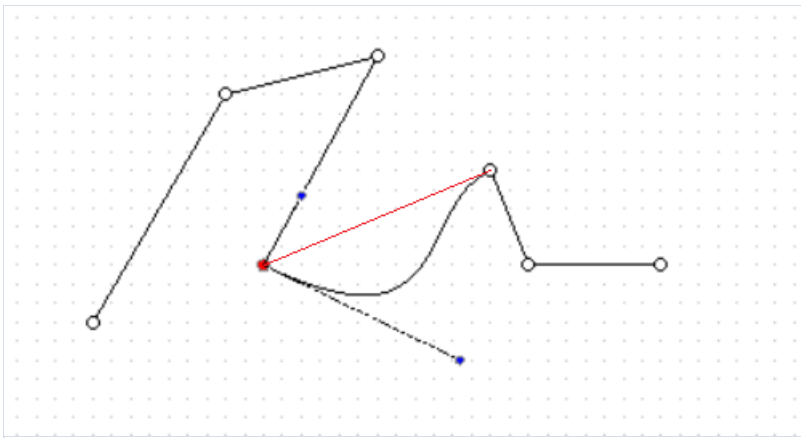


The gray lines in the image help you see how the object will look after adding a new point.

After placing the first point, you can add additional ones. After you finish drawing the polygon, press **Escape** or switch Edit mode. The editing mode panel for the Polyline and Polygon objects is located on the Home tab.

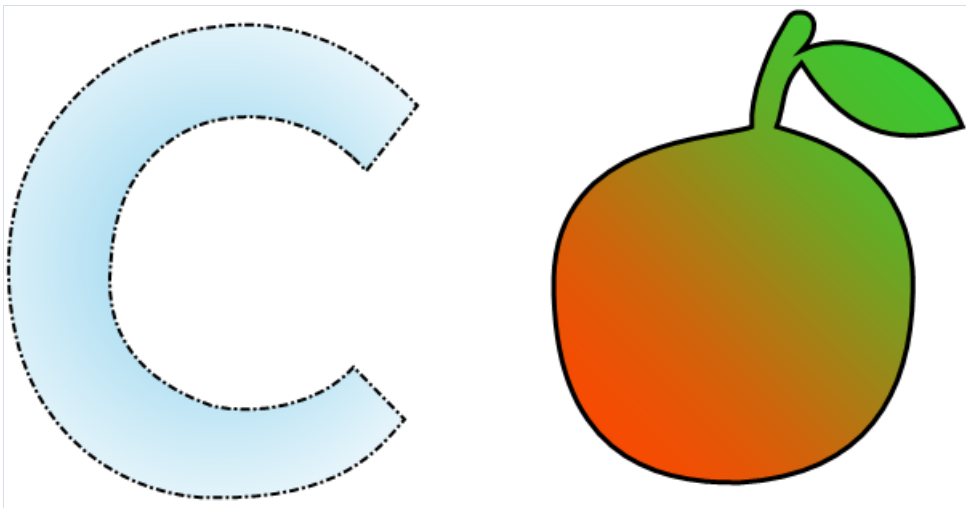
Polyline and polygon editing modes:

| Button  | Mode               | Description   |
|---|--------------------|---|
|  | Adding a point     | This mode turns on immediately after adding a polygon. To add a point, select this mode and select one of the points of the object. After that, hover over the desired location of the point and press the left mouse button. The point will be added next to the one you selected. |
|  | Moving and scaling | In this mode, you can move the entire object, as well as stretch it.  |
|  | Deleting a point   | To remove a point from a polygon or polyline, select this mode and click on the desired point. You can also select it in another mode and press the Delete key.   |
|  | Mouse pointer      | This mode allows you to view and move all points of the object that were added earlier.   |
|  | Curve anchor point | In this mode, you can add a point of a curve to the line adjacent to the selected point. This line will become a Bezier curve, and the added point will act as an anchor.   |



This is what a broken line with a set point of curvature (blue) looks like. The red line shows the state of the line before it was converted to a curve.

Both objects have the ability to adjust their borders, and the Polygon object supports the same fill modes as the rest of the objects.



# The "Barcode" object

The object displays barcodes in the report. It looks like this:



The Barcode object supports the following types of barcodes:

## Two-dimensional

| Barcode name | Length          | Allowed symbols        |
|--------------|-----------------|------------------------|
| PDF417       | No fixed length | Any                    |
| Datamatrix   | No fixed length | Any                    |
| QR Code      | No fixed length | Any                    |
| Aztec        | No fixed length | Any                    |
| MaxiCode     | No fixed length | 138 digits or 93 chars |

## EAN/UPC

| Barcode name | Length | Allowed symbols |
|--------------|--------|-----------------|
| EAN-8        | 8      | 0-9             |
| EAN-13       | 13     | 0-9             |
| UPC-A        | 12     | 0-9             |
| UPC-E0       | 6      | 0-9             |
| UPC-E1       | 6      | 0-9             |

## Post

| Barcode name       | Length          | Allowed symbols |
|--------------------|-----------------|-----------------|
| Deutsche Identcode | 12              | 0-9             |
| Deutsche Leitcode  | 14              | 0-9             |
| PostNet            | No fixed length | 0-9             |

| Barcode name            | Length          | Allowed symbols |
|-------------------------|-----------------|-----------------|
| Japan Post 4 State Code |                 |                 |
| Intelligent Mail (USPS) | No fixed length | 0-9, A-Z        |

## GS1

| Barcode name                        | Length          | Allowed symbols |
|-------------------------------------|-----------------|-----------------|
| GS1-128                             | No fixed length | 0-9, A-Z        |
| GS1 DataBar Omnidirectional         |                 |                 |
| GS1 DataBar Limited                 |                 |                 |
| GS1 DataBar Stacked                 |                 |                 |
| GS1 DataBar Stacked Omnidirectional |                 |                 |
| GS1 Datamatrix                      |                 |                 |

## Others

| Barcode name       | Length          | Allowed symbols                |
|--------------------|-----------------|--------------------------------|
| 2 of 5 Interleaved | No fixed length | 0-9                            |
| 2 of 5 Industrial  | No fixed length | 0-9                            |
| 2 of 5 Matrix      | No fixed length | 0-9                            |
| ITF-14             | 14              | 0-9                            |
| Codabar            | No fixed length | 0-9, -, \$, :, /, ., +         |
| Code 128           | No fixed length | 128 ASCII chars                |
| Code 39            | No fixed length | 0-9, A-Z, -, ., *, \$, /, +, % |
| Code 39 Extended   | No fixed length | 128 ASCII chars                |
| Code 93            | No fixed length | 0-9, A-Z, -, ., *, \$, /, +, % |
| Code 93 Extended   | No fixed length | 128 ASCII chars                |
| MSI                | No fixed length | 0-9                            |
| 2-Digit Supplement | 2               | 0-9                            |
| 5-Digit Supplement | 5               | 0-9                            |

| Barcode name | Length          | Allowed symbols  |
|--------------|-----------------|------------------|
| Plessey      | No fixed length | Hex digits (0-F) |
| Pharmacode   | No fixed length | 0-9              |

Barcode data in an object is of a string type. The string can contain any symbol, allowed for the chosen type of barcode. You can choose the type of barcode in the context menu of the "Barcode" object.

You can connect an object to data by using one of the following methods:

- set the barcode data in the `Text` property;
- bind the object to a data column using the `DataColumn` property;
- set the expression that returns the barcode data in the `Expression` property.

The Barcode object has the following properties:

| Property          | Description   |
|-------------------|---|
| <b>Barcode</b>    | This property contains barcode-specific settings. Expand this property to set these settings.   |
| <b>Angle</b>      | This property determines the rotation of a barcode, in degrees. You can use one of the following values: 0, 90, 180, 270.                                     |
| <b>Zoom</b>       | This property allows to zoom a barcode. It is used along with the <code>AutoSize</code> property.   |
| <b>AutoSize</b>   | If this property is on, the object will stretch in order to display a whole barcode. If this property is off, the barcode will stretch to to object's bounds. |
| <b>ShowText</b>   | This property determines whether it is necessary to show the human-readable text.   |
| <b>DataColumn</b> | The data column which this object is bound to.  |
| <b>Expression</b> | The expression that returns the barcode data.   |
| <b>Text</b>       | The barcode data.   |
| <b>Padding</b>    | The padding, in pixels.   |

The following properties are specific to the barcode type. To change them, select the barcode, go "Properties" window and expand the `Barcode` property.

| Property            | Description   |
|---------------------|---|
| <b>WideBarRatio</b> | This property is specific to all linear barcodes. It determines the wide-to-narrow bar ratio. For most of barcode types, the value for this property should be between 2 and 3.                                 |
| <b>CalcChecksum</b> | This property is specific to all linear barcodes. It determines whether is necessary to calculate the check sum automatically. If this property is off, you should provide the check digit in the barcode data. |

| Property               | Description  |
|------------------------|--|
| <b>AutoEncode</b>      | <p>This property is specific to the Code128 barcode. This code has three different encodings - A, B, C. You should either set the encoding explicitly in the barcode data, or set this property to true. In this case the encoding will be chosen automatically.</p> <p>Use the following control codes in the barcode data:</p> <p>&amp;A; START A / CODE A<br/> &amp;B; START B / CODE B<br/> &amp;C; START C / CODE C<br/> &amp;S; SHIFT<br/> &amp;1; FNC1<br/> &amp;2; FNC2<br/> &amp;3; FNC3<br/> &amp;4; FNC4</p> <p>If you set the <code>AutoEncode</code> property to <code>true</code>, all control codes will be ignored. Example of use the control codes: <code>&amp;C;1234&amp;B;ABC</code></p> |
| <b>AspectRatio</b>     | <p>This property is specific to the PDF417 barcode. It determines the height-to-width aspect ratio and is used to calculate the barcode size automatically (in case the <code>Columns</code> and <code>Rows</code> properties are not set).</p>  |
| <b>CodePage</b>        | <p>This property is specific to the PDF417 and Datamatrix barcode. It determines the code page which is used to convert non-ASCII chars. For example, the default windows codepage is 1251.</p>  |
| <b>Columns, Rows</b>   | <p>These properties are specific to the PDF417 barcode. They determine the number of columns and rows in a barcode. If both properties are set to 0, the size of the barcode will be calculated automatically. In this case the <code>AspectRatio</code> property is used as well.</p>   |
| <b>CompactionMode</b>  | <p>This property is specific to the PDF417 barcode. It determines the PDF417 data compaction mode.</p>   |
| <b>ErrorCorrection</b> | <p>This property is specific to the PDF417 barcode. It determines the error correction level.</p>  |
| <b>PixelSize</b>       | <p>This property is specific to the PDF417 barcode. It determines the size of barcode element, in screen pixels. As a rule, the element's height should be greater than the element width by 3 times or more.</p>  |
| <b>Encoding</b>        | <p>This property is specific to the Datamatrix barcode. It determines the Datamatrix data encoding.</p>  |
| <b>PixelSize</b>       | <p>This property is specific to the Datamatrix barcode. It determines the size of barcode element, in pixels.</p>  |
| <b>SymbolSize</b>      | <p>This property is specific to the Datamatrix barcode. It determines the size of barcode symbol.</p>  |

# PDF417




This common two-dimensional barcode is designed to encode large amounts of data. Its abbreviation stands for Portable Data File, and the number 417 was formed as a result of the addition of 4 and 17. Here 4 is four bars and four spaces, and 17 is the number of modules in the code word.

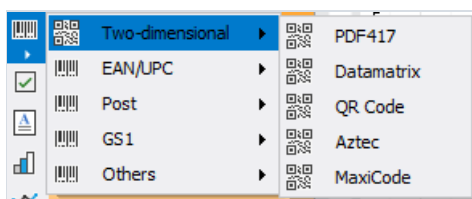
As noted above, PDF417 is a widely used barcode, along with QR Code, Maxi Code and Data Matrix. PDF417 looks like a mixture of the classic linear and matrix barcodes, but in fact it is a complex linear code. Along the edges it has bars, as in ordinary linear codes, and in the middle, the bars are arranged in lines one above the other. It turns out such a compression of the linear code due to the arrangement of bars with codes one above the other. This allows you to store large amounts of information – from 3 to 90 bars, in which you can encode up to 1859 alphabetic characters or 2725 numeric characters.

Code corruption protection provides redundancy that can cover up to 50% of the code. This is a very high number, but the size of the code will grow as well. Compared to matrix codes, PDF417 takes up several times more space when encoding the same amount of information which can be considered a disadvantage.


The scope of PDF417 is quite extensive. It is used by: transport companies for printing on passenger tickets and cargo shipments, in postal items, reporting documents, various identity cards, warehouse accounting, and in many other areas where labeling and identification is required.

To read this barcode, laser scanners are used, and they are slightly different from conventional line code scanners. As mentioned above, there are normal lines on the sides of the code, typical for linear barcodes. They are needed to identify the beginning and end of the code.

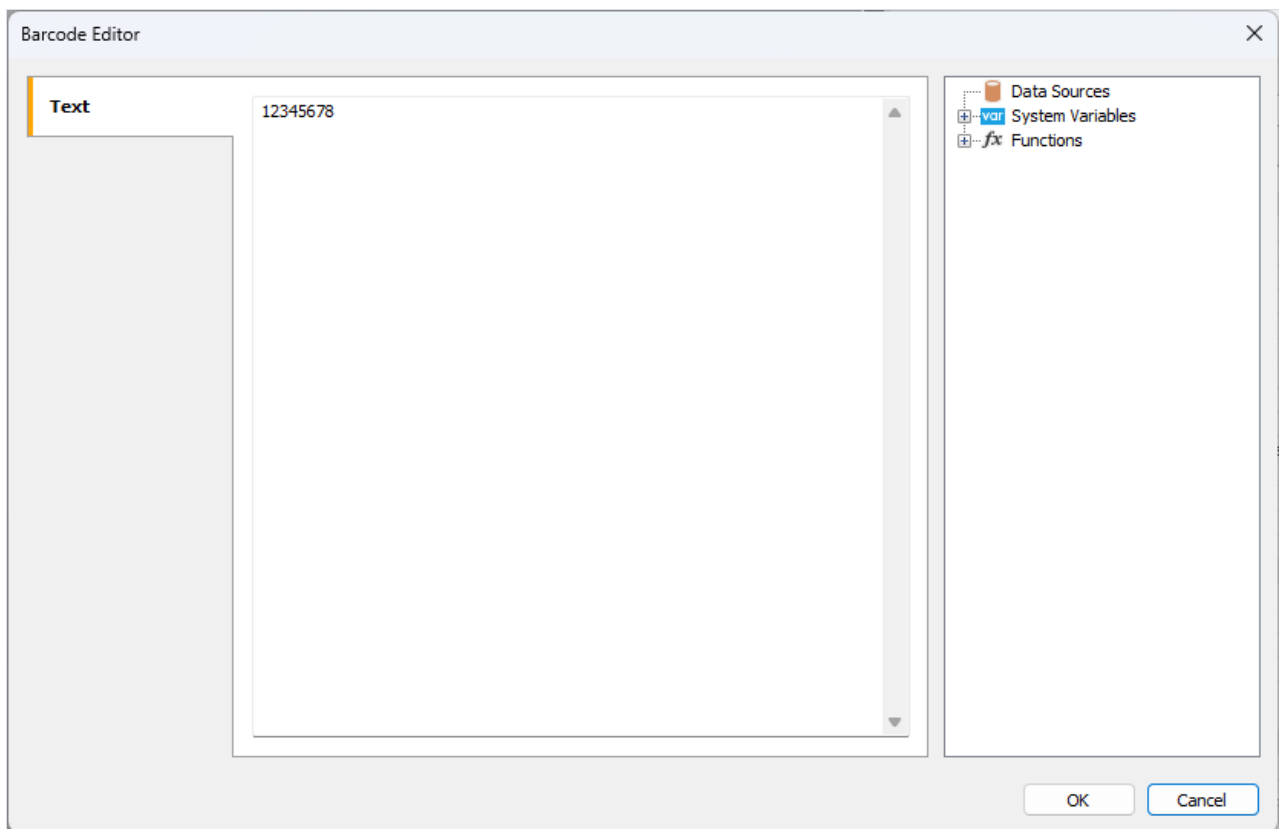
To generate a PDF417 barcode in FastReport .NET, select the Barcode object  at the Components Panel in the Report Designer. In the drop-down list, navigate to the "Two-dimensional" category, and then choose PDF417:



After selecting the barcode, place it on the Report Page.

Double-click on the added barcode to open the editor. You can also open the barcode editor by clicking the button  in the context menu of the added object, accessed by right-clicking:





If you want to hide the text under the barcode, locate the `ShowText` property in the property inspector of the corresponding barcode and set its value to `False`.


# QR code

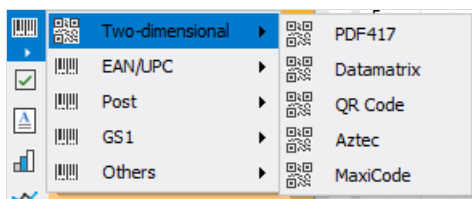
QR code is a two-dimensional barcode used to store numeric, alphanumeric, and binary information.

For accurate QR code recognition using a camera, special markers are employed at the corners and within the image area. This allows for image normalization post-scanning and conversion of dot encoding into binary numbers with checksum verification.


A QR code can contain up to 4296 characters using alphanumeric encoding.

## Object

To generate a QR code in FastReport .NET, select the Barcode object  at the Components Panel in the Report Designer. In the drop-down list, navigate to the "Two-dimensional" category, and then choose QR Code:

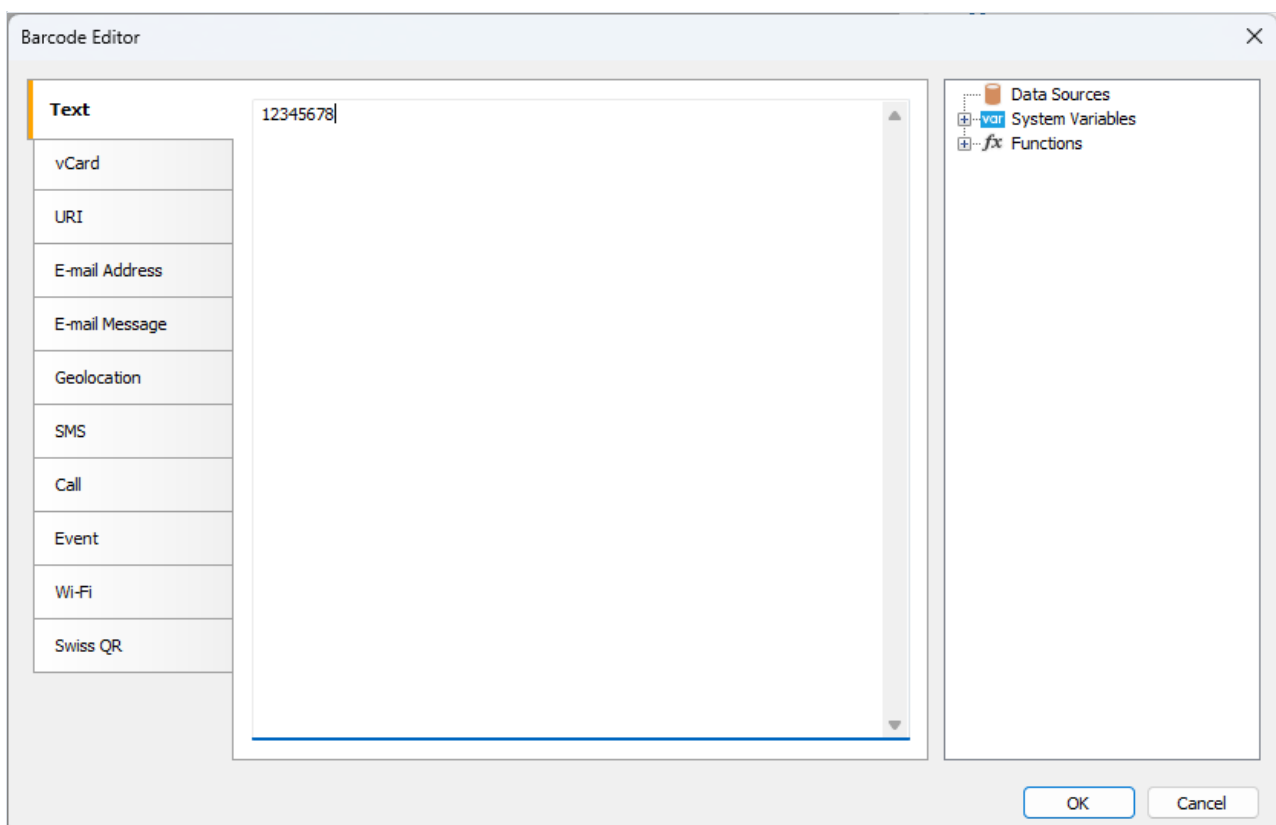


After selecting the barcode, place it on the Report Page.

Double-click on the added barcode to open the editor. You can also open the barcode editor by clicking the button  in the context menu of the added object, accessed by right-clicking.

## Editor

The QR code editor looks like a regular expression editor. There is a tree of data, parameters and functions on the right. From there, you can drag items into the text editor.



The main difference from the expression editor lies in the presence of tabs on the left side of the window. These tabs determine the type of content for the QR code.

Depending on the selected content type, a corresponding set of fields for input appears.

Barcode Editor

Text

**vCard**

URI

E-mail Address

E-mail Message

Geolocation

SMS

Call

Event

Wi-Fi

Swiss QR

First name

Anne

Phone (mobile)

Last name

Dodsworth

Phone (personal)

Title

Manager

Phone (business)

(71) 555-4444

Company

Street

7 Houndstooth Rd.

Website

Zip code

E-mail (personal)

City

E-mail (business)

Country

Data Sources

var

System Variables

fx


Functions










OK


Cancel

QR code content types

When working with the editor, the content text (the fourth column of the table) is generated automatically. Editor fields can contain any expressions, including fields from the data source (you can drag them from the tree on the right).

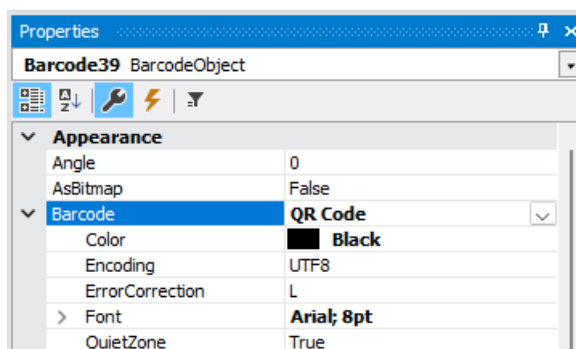
| Content type | Description              | Image example   | Content example |
|--------------|--------------------------|---|-----------------|
| Text         | Literal and numeric text |  | 12345678        |

| Content type          | Description  | Image example   | Content example   |
|-----------------------|--|---|---|
| <b>vCard</b>          | <p>Versitcard is an electronic business card presentation format. May contain the following information:</p> <ul style="list-style-type: none"> <li>- Surname;</li> <li>- Name;</li> <li>- Position;</li> <li>- Company;</li> <li>- Company website;</li> <li>- personal E-mail;</li> <li>- work E-mail;</li> <li>- Mobile phone;</li> <li>- Home phone;</li> <li>- Work phone;</li> <li>- Address;</li> <li>- Index;</li> <li>- City;</li> <li>- Country.</li> </ul> <p>Only some of the fields can be filled in.</p> |    | <pre>BEGIN:VCARD VERSION:2.1 FN:AnneDodsworth N:Dodsworth;Anne TITLE:Manager TEL;WORK;VOICE(71) 555-4444 ADR;;;7 HoundstoothRd.;;; END:VCARD</pre>                  |
| <b>URI</b>            | Uniform resource identifier. A string with a link to a file, document, image, email, website, etc.   |    | <pre>http://www.fast-report.com/en/product/fast-report-net/</pre>   |
| <b>E-mail Address</b> | E-mail address   |   | <pre>support@fast-report.com</pre>  |
| <b>E-mail Message</b> | E-mail message   |  | <pre>MATMSG:TO:support@fast-report.com;SUB:FastReport .NET question;BODY&gt;Hello, I have a question about FastReport .NET.;;</pre>                                 |
| <b>Geolocation</b>    | Coordinates for determining the geographic location  |  | <pre>geo:-50.737563,-79.490016,120</pre>  |
| <b>SMS</b>            | Text message   |  | <pre>SMSTO:(71) 555-4444:Hello, Dolly! I'm fine!</pre>  |
| <b>Call</b>           | Phone number   |  | <pre>tel:(71) 555-4444</pre>  |
| <b>Event</b>          | Event to add to the calendar. It can contain a text message additionally to the time and date.   |  | <pre>BEGIN:VEVENT SUMMARY:Team Meeting DTSTART:20240420T090000Z DTEND:20240420T100000Z DESCRIPTION:Weekly team meeting to discuss project updates. END:VEVENT</pre> |
| <b>Wi-Fi</b>          | Information for connecting to a Wi-Fi network.   |  | <pre>WIFI:T:WPA;S:Honeypot;P:youarewelcome;H:true;</pre>  |

| Content type | Description  | Image example   | Content example   |
|--------------|--|---|---|
| Swiss QR     | A special QR code containing payment information for Swiss Bill. You can read more about this barcode in <a href="#">another article</a> . |  | <div>SPC</div> <div>0200</div> <div>1</div> <div>CH4431999123000889012</div> <div>S</div> <div>Carl Ltd.</div> <div>Luber</div> <div>16</div> <div>123321</div> <div>Berlin</div> <div>GE</div> |
|              |  |   | <div>50050.00</div> <div>EUR</div> <div>S</div> <div>Sigmunt Shuld</div> <div>Lunglen</div> <div>23</div> <div>123322</div> <div>Ferburg</div> <div>GE</div> <div>NON</div>                     |
|              |  |   | <div>EPD</div>  |

## Barcode properties

Now let's look at the properties of the QR code. They are available in the object inspector under the Barcode property.



| Property               | Description   |
|------------------------|---|
| <b>Encoding</b>        | Text encoding of the barcode content, for example: UTF8, Windows_1251, CP_866, etc. Default: UTF8.  |
| <b>ErrorCorrection</b> | Error correction using the Reed-Solomon code. It can take on the following values: L (low – 7%), M (medium – 15%), Q (25%), H (high – 30%). The default is L. |
| <b>QuietZone</b>       | Determines the presence of a white border around the QR code. Default: True (enabled).  |

|              |  |
|--------------|--|
| <b>Color</b> | Determines the color of the barcode. The default is Black. |
|--------------|--|

The error correction is needed for correct data reading in case of a partially damaged code image or an image applied over it.

For example, if the redundancy is set to H (30%), the barcode shown below is read without problems:



If you want to hide the text under the barcode, locate the `ShowText` property in the property inspector of the corresponding barcode and set its value to `False`.

The `AutoSize` property is used for automatically adjusting the size of the code depending on the size of the object. If you need to manually resize the barcode (using the mouse), then you need to disable this property (set to `False`). In this case, you need to monitor the proportions yourself.


# Swiss QR

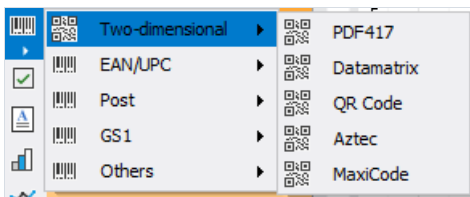
Swiss QR is one of the variants of [QR codes](#). In Switzerland, as well as in the whole world QR codes are used everywhere. However, to use these codes in the sphere of electronic payments it was decided to create its own kind of QR. Now all the payment receipts and bills in Switzerland should have a QR code with the Swiss cross in the center is a distinctive feature of the Swiss QR.

FastReport .NET supports Swiss QR code. The code itself looks almost the same as an ordinary QR with the only difference that there is a Swiss cross in its center:




The essence of this type of code is that it encrypts payment information.

To generate a Swiss QR in FastReport .NET, select the Barcode object  at the Components Panel in the Report Designer. In the drop-down list, navigate to the "Two-dimensional" category, and then choose QR Code:



After selecting the barcode, place it on the Report Page.

Double-click on the added barcode to open the editor. You can also open the barcode editor by clicking the button  in the context menu of the added object, accessed by right-clicking.

In the QR code editor, you need to switch to the Swiss QR tab. After that, the settings for the Swiss QR will be displayed:

The image shows a 'Barcode Editor' window. On the left is a sidebar with a list of data types: Text, vCard, URI, E-mail Address, E-mail Message, Geolocation, SMS, Call, Event, Wi-Fi, and Swiss QR (which is highlighted with an orange bar). The main area contains a form for the 'Swiss QR' type. The form has several sections: 
 

- IBAN Section:** Fields for IBAN (CH4431999123000889012), Type (Iban), Amount (50050), and Currency (EUR).
- Creditor Section:** Fields for Name (Carl Ltd.), Street / Nr (Luber, 16), and Zip / City / Country (123321, Berlin, GE).
- Debitor Section:** Fields for Name (Sigmunt Shuld), Street / Nr (Lunglen, 23), and Zip / City / Country (123322, Ferburg, GE).
- Additional information Section:** Fields for Reference, Type (NON), Text type, Message, Bill Information, Alternative procedure 1, and Alternative procedure 2.

 On the right side of the form is a 'Data Sources' panel with 'var System Variables' and 'fx Functions'. At the bottom right are 'OK' and 'Cancel' buttons.

Let's take a closer look at the parameters:

## Iban

In Switzerland, the IBAN (*International Bank Account Number*) standard is used to represent a bank account number. It is an international standard, which is registered in ISO under the number 13616.

Here you can choose between two types **Iban** or **QR-Iban**.

**QR-Iban** should be used for payments with QR link. In this case **QR-Iban** also corresponds to the standard ISO 13616. Every legal institution, which participates in the scheme, gets its identifier in the range of 30000 - 31999. This identifier is called QR-IID and it is included into the **QR-Iban**.

## Creditor

Fill in the details of the billing agent. Name of the organization and address.

## Reference

A link to the payer's payment, which the recipient of the payment needs.

## Type

- QRR is a QR link: a Swiss standard link with the length of 26 symbols (only figures);
- SCOR – Link of the creditor: the international standard of the link with the length from 5 to 25 symbols;
- NON – the link can be empty.

## Text type

- QR-Reference is used with the type of the reference QRR;
- ISO 13616 – is used with the reference type SCOR.

QR-reference is a replacement for the ISR link, which is used now. It helps to go to the new QR accounts with the used wound of red and orange receipts.

## Debitor



Payer data – name or name of the organization and address.

### **Additional Information**

An invoice issuer can enter any additional structured/nonstructured information for the payer.

### **Currency**

Since the payment system is Swiss, two types of currency are assumed: euro (EUR) and Swiss francs (CHF).

### **Alternative Procedure 1 and Alternative Procedure 2**

It is assumed that in the future billing agents may offer procedures alternative to bank transfers. There are two fields in Swiss QR for this purpose.

### **Amount**

Here you can specify the amount of payment with comma separator.

As a rule Swiss QR is used in the document Swiss bill. You can find an example of the report with such Swiss account in the folder [Demos/Reports/](#).

# Aztec

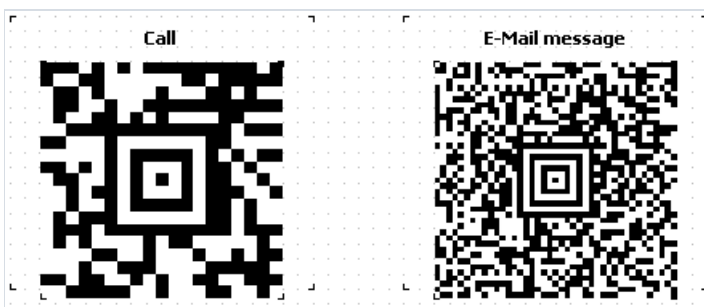
Aztec Code is one of the types of QR codes. Aztec is the name of the tribe of Indians from Central America. If you look carefully at the code, in its center you can notice a square, which looks like the Aztec pyramid, if you look at it from above. This is a special target from which you can determine the center of the code and its orientation.



Aztec Code combines the best ideas of 2D barcodes: MaxiCode, SuperCode, CodeOne, DataMatrix, DotCode and PDF417. Despite the patent, this development has become public domain. The coding standard is described in ISO/IEC 24778:2008.

The size of the code depends on the amount of information being encoded. For example, a minimum size of 15x15 pixels allows you to encode 6 bytes, that is, 12 letters or 13 digits. The maximum size of 151x151 pixels allows encoding 1914 bytes, 3067 letters or 3832 digits.

Note that the code has two display formats: Compact, where the symbol with the target consists of two squares, and Full-Range, where the symbol with the target consists of three squares. The choice of format depends on the amount of data encoded.



The advantage of this type of encoding over others is the ability to read the code in any orientation. Even the mirrored code will be easily read, due to the use of navigation markers.


Using a target in the center of the code allows you to read information even from distorted or stretched images.

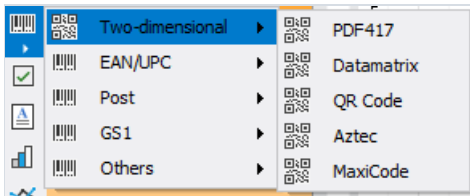
Thanks to the Reed-Solomon encoding algorithm, the Aztec Code can be read even with partial damage. In this case, redundancy is specifically embedded in the code. The user is given the opportunity to adjust the percentage of redundant code from 5 to 95, which provides a high degree of resistance to reading errors.

The layer-by-layer structure of the code makes it possible to increase the volume of stored information by increasing the coding area.


All these advantages made Aztec Code very attractive for use in transport networks as electronic tickets, for example, in air and rail transportation. In some countries, it is used in government documentation. Like other high-density codes, Aztec codes are popular in commerce, logistics, manufacturing and pharmaceuticals.

In comparison with a QR code, Aztec Code has a higher density of recording and doesn't require fields around the code. Also, the minimum size of Aztec Code is 15x15 compared to 21x21 at QR.

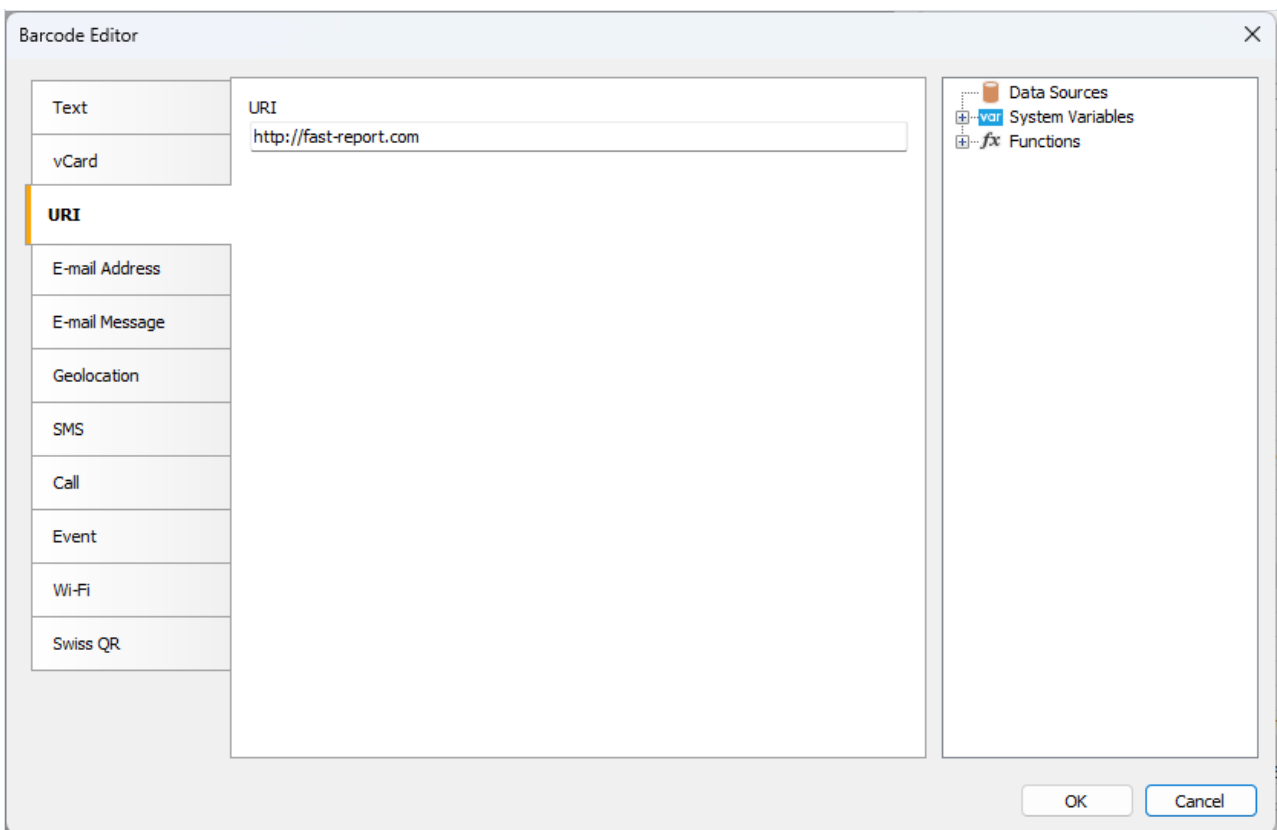
To generate an Aztec Code in FastReport .NET, select the Barcode object  at the Components Panel in the Report Designer. In the drop-down list, navigate to the "Two-dimensional" category, and then choose Aztec:



After selecting the barcode, place it on the Report Page.

Double-click on the added barcode to open the editor. You can also open the barcode editor by clicking the button  in the context menu of the added object, accessed by right-clicking.

In the barcode editor, you can choose a template for encoding information. All templates except the Swiss QR template can be used in Aztec Code. For example, a website address:

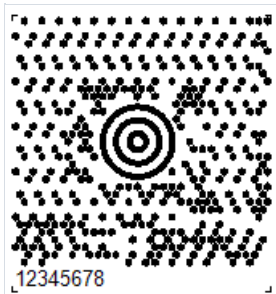


In the end the code will look like this:



# MaxiCode

The two-dimensional MaxiCode barcode has a unique design that cannot be confused with others. It is easily identifiable by its circular target, facilitating accurate scanning positioning. Encoding elements are depicted as dots, arranged in a hexagonal (cellular) configuration, forming unique combinations. This grid comprises 888 cells, each carrying specific information.



MaxiCode allows encoding 138 digital characters or 93 alphabetic characters. The barcode is fixed at 1.11 x 1.054 inches. It provides read error correction for code corruption based on Reed-Solomon code. Since most of these barcodes are on packages, they must be resistant to damage. Error correction allows up to 1/8 of the code to be corrupted.


Each Maxicode barcode includes two main messages: primary and secondary. The primary message includes a zip code, country code and class, while the secondary message contains address data.

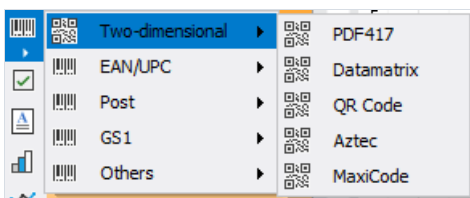
To read the Maxicode barcode a 2D barcode scanner is required, preferably one that performs keyboard emulation and is powered by USB port, so no external power supply is required.

The main application of MaxiCode is marking of packages, pallets, etc. It ensures that critical information is available at all times when the shipment is being processed.


MaxiCode was designed to be included in an existing manifest of delivery systems. Its compact size also allows the use of MaxiCode to replace less dense characters such as linear barcodes.

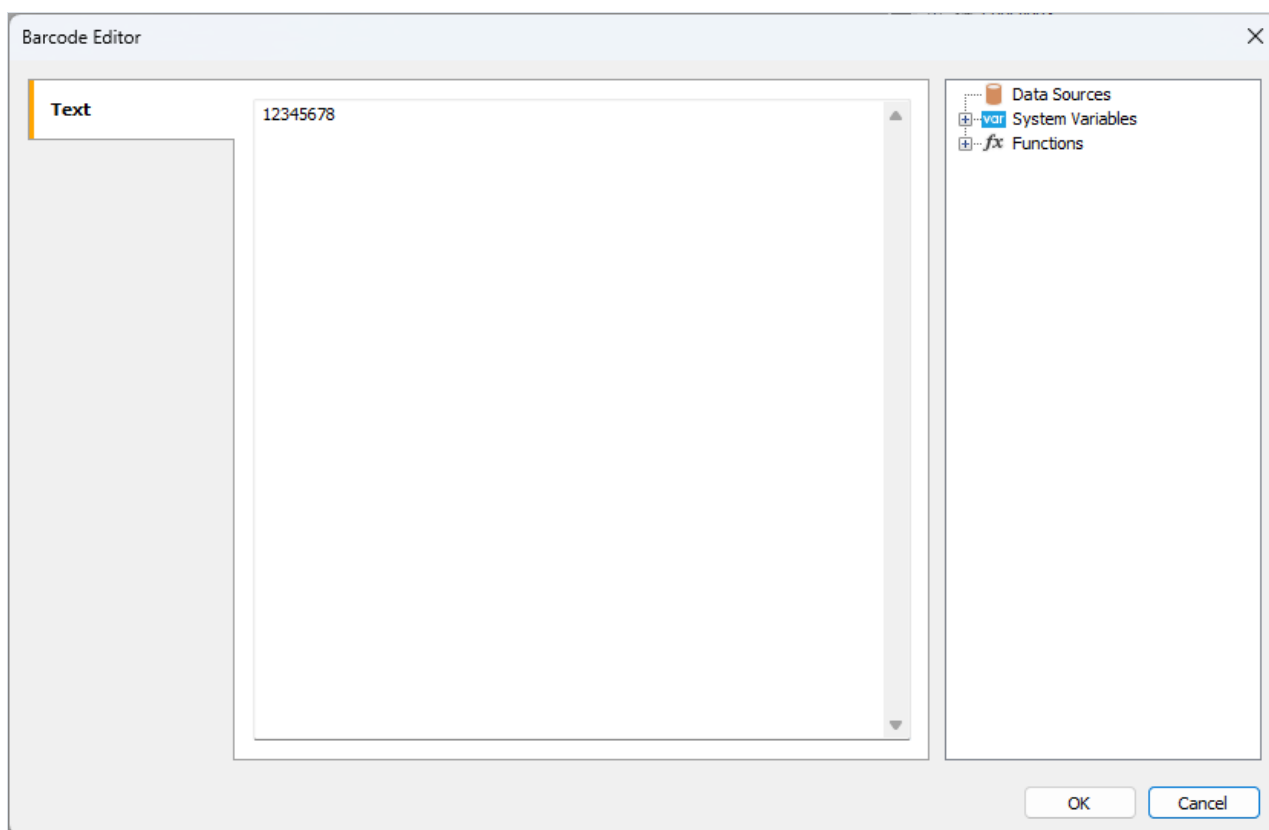
One of the key features of MaxiCode is that it can be read at high speed in a large field of view. This means that this code can be successfully used in automated information processing systems.

To generate a MaxiCode barcode in FastReport .NET, select the Barcode object  at the Components Panel in the Report Designer. In the drop-down list, navigate to the "Two-dimensional" category, and then choose MaxiCode:



After selecting the barcode, place it on the Report Page.

Double-click on the added barcode to open the editor. You can also open the barcode editor by clicking the button  in the context menu of the added object, accessed by right-clicking:



Like all other barcodes in FastReport, MaxiCode can be displayed with or without text information under the code. The `ShowText` property is responsible for this.

# EAN-8

EAN-8 (European Article Number), also known as GTIN-8, is a short 8-digit barcode. The numbers are divided into 2 blocks of 4 numbers. The first 2 digits identify the country of origin of the goods, then 5 digits are the encoded information, and the last digits are the checksum to verify the integrity of the data.

This code was created based on EAN-13 by reducing the size of the code to 8 digits. This adjustment allows for the placement of the code on small packages. As long as there is a country identifier in the code, it is considered international.

EAN-8 is used in trade to identify products and equipment.

Each digit is encoded using 7 units, which consist of vertical lines and spaces. Let's assign 0 to represent a space and 1 to represent a line. The encoding of the numbers can be represented as follows:

- 0 – 0001101;
- 1 – 0011001;
- 2 – 0010011;
- 3 – 0111101;
- 4 – 0100011;
- 5 – 0110001;
- 6 – 0101111;
- 7 – 0111011;
- 8 – 0110111;
- 9 – 0001011.

Let's see in this example:



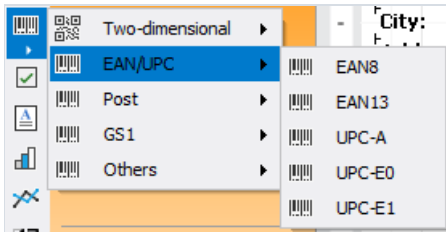
Remember, the code begins with 101, followed by the encoding for each digit starting from 3 and so forth. As mentioned earlier, the code is divided into two parts. The separator line, along with the starting and ending lines of the code, is long and represented by the 01010 pattern.

Upon closer examination, you'll observe that the lines following the separator do not match the numbers listed above. This is because a different encoding scheme is utilized for the second part:


- 0 – 1110010;
- 1 – 1100110;
- 2 – 1101100;
- 3 – 1000010;
- 4 – 1011100;
- 5 – 1001110;
- 6 – 1010000;
- 7 – 1000100;
- 8 – 1001000;
- 9 – 1110100.

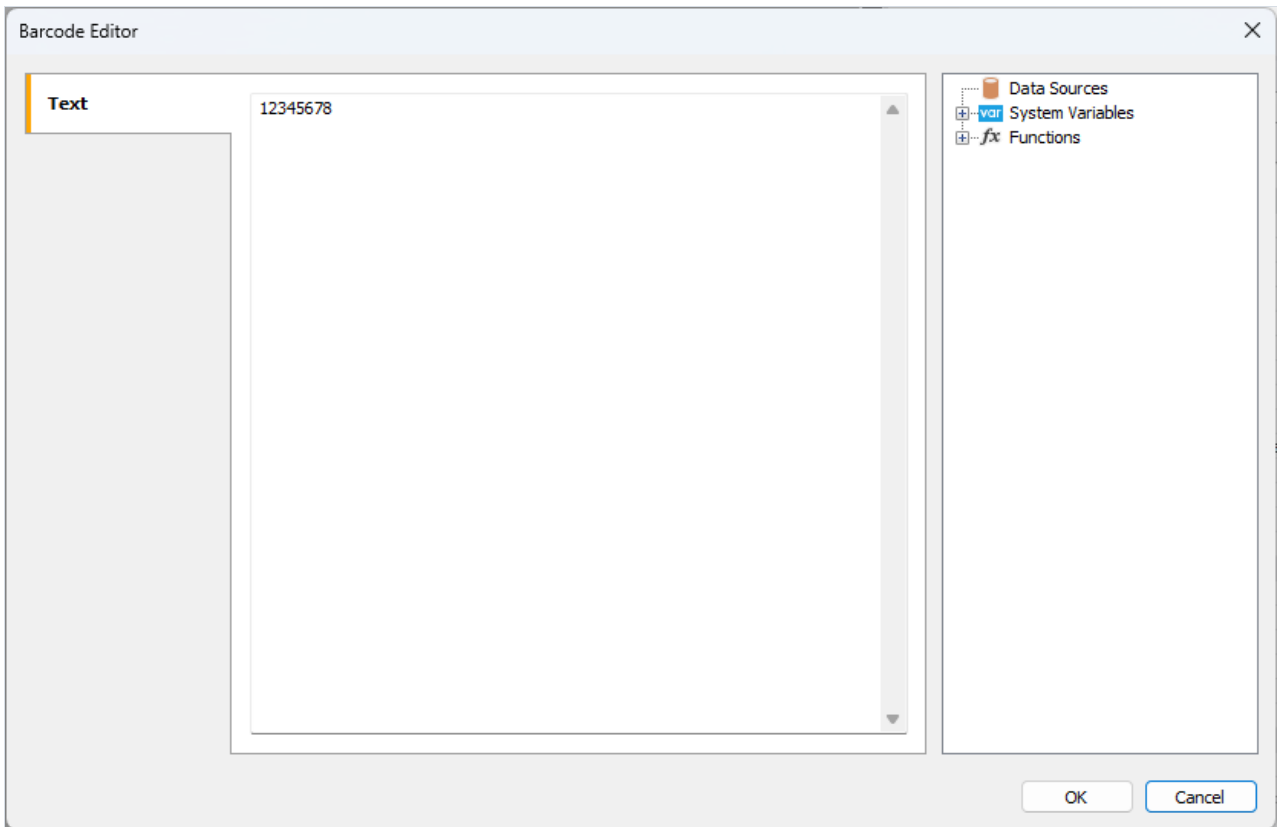
To generate an EAN-8 barcode in FastReport .NET, select the Barcode object  at the Components Panel in the

Report Designer. In the drop-down list, navigate to the "EAN/UPC" category, and then choose EAN8:

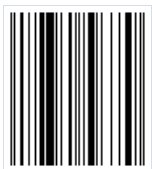


After selecting the barcode, place it on the Report Page.

Double-click on the added barcode to open the editor. You can also open the barcode editor by clicking the button  in the context menu of the added object, accessed by right-clicking:



If you want to hide the text under the barcode, locate the `ShowText` property in the property inspector of the corresponding barcode and set its value to `False` :



# EAN-13



The EAN-13 (European Article Number) barcode is the most commonly used linear barcode symbology. It encodes 12 digits of information, the last 13th digit of the code is a checksum for checking the integrity of the code. Only digits are allowed.

The physical structure of the code is as follows:

- the first 2-3 digits are used to encode the country of the product manufacturer;
- next 4-5 digits are used to encode the manufacturer of the goods;
- the remaining 3-5 digits are the product number at the enterprise.


As already mentioned, the 13th digit is the checksum. It is calculated automatically based on the previous 12 digits using a special algorithm. Thanks to this digit, you can determine whether the code has been read correctly.

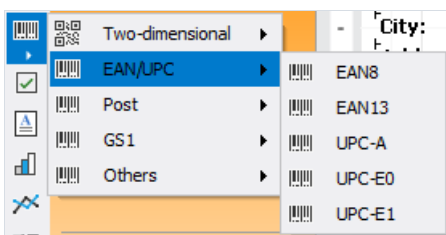
When examining the code's appearance, you'll notice a sequence of two bars at both the beginning and end. Representing the code as a binary sequence, with bars as 1 and spaces as 0, results in the pattern 101. Additionally, the barcode is symmetrically divided into two parts at its midpoint, marked by the same sequence.

Each character is encoded in 7 elements (bars and spaces). There are three special tables with code sequences for digits. Moreover, the first and the second halves of the code use different versions of such tables.


This code can be read in either left-to-right or right-to-left sequence, adding to its versatility. Its simplicity and ease of reading have quickly made it popular in the retail industry for product labeling.

The low capacity of this code (the length of the encoded sequence) is often mentioned as the disadvantage.

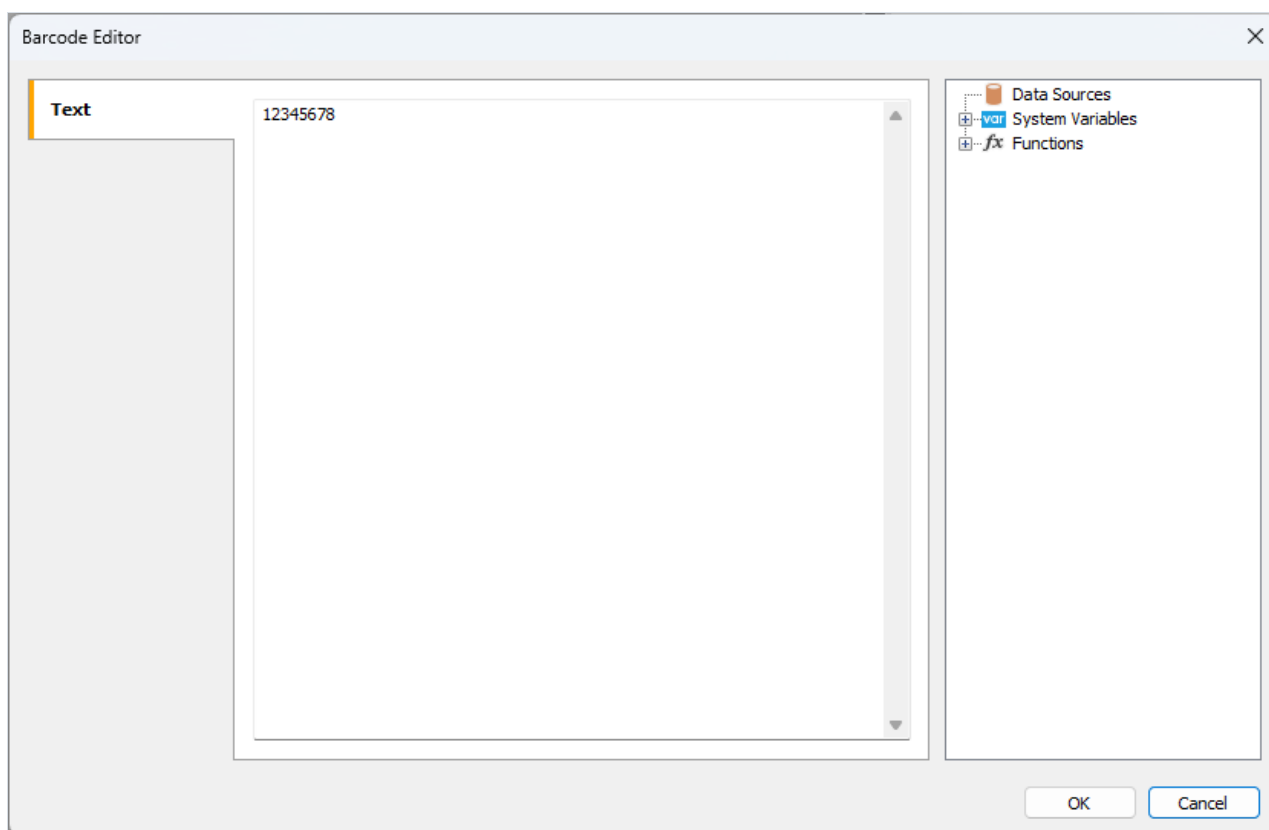
To generate an EAN-13 barcode in FastReport .NET, select the Barcode object  at the Components Panel in the Report Designer. In the drop-down list, navigate to the "EAN/UPC" category, and then choose EAN13:



After selecting the barcode, place it on the Report Page.

Double-click on the added barcode to open the editor. You can also open the barcode editor by clicking the button  in the context menu of the added object, accessed by right-clicking:





If you want to hide the text under the barcode, locate the `ShowText` property in the property inspector of the corresponding barcode and set its value to `False` :



# UPC-A



Universal Product Code is a linear barcode that allows you to encode 12 characters. It was developed jointly by the Uniform Grocery Product Code Council and IBM in 1973. Its structure and purpose are similar to the well-known EAN-13 code. UPCs are designed for North America, while EANs are for Europe.

Code structure:

- start character, indicating the beginning of the code;
- a prefix that indicates the type of product – 1 character;
- manufacturer code – 5 characters;
- product code – 5 characters;
- check digit – 1 character. Calculated from the previous 11 digits using the Modulo 10 formula;
- stop character, indicating the end of the code.

There should be an empty space of about 9 modules before and after the code. This is necessary to ensure that the code is recognized by the scanner.


The code length is 12 characters, of which only 11 are encoded data, and one more is a check digit.

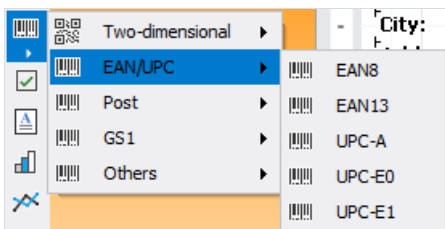
Each character is encoded with two bars and two spaces. A bar or space can be 1, 2, 3, or 4 modules wide (one module is 0.33mm).

The UPC-A barcode is very common in the US and Canada. It is used in supermarkets to label products.


This barcode has earned a great deal of popularity due to its compact size, ease of scanning, and the presence of a check digit to protect against reading errors.

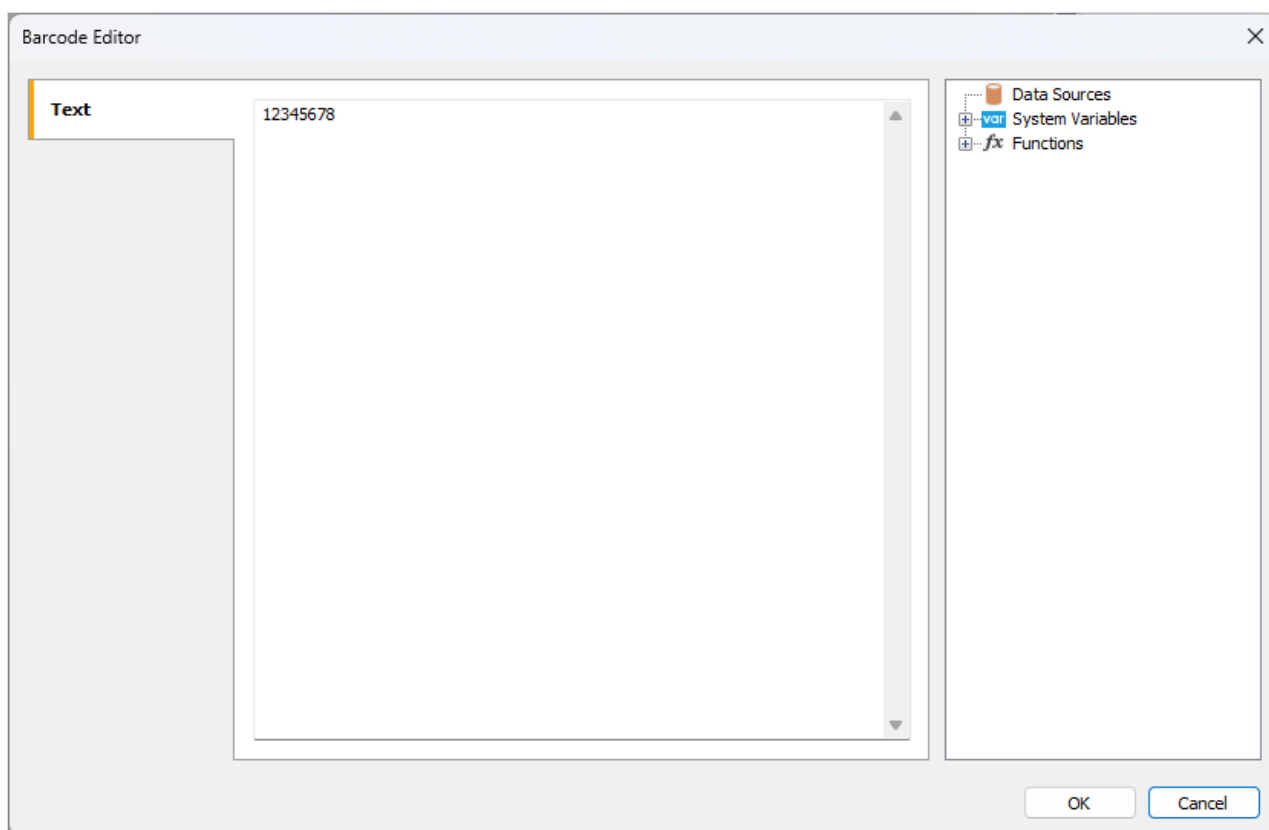
The disadvantages of the code include the ability to encode only numbers, as well as the small size of the code, which limits the scope of its application.

To generate an UPC-A barcode in FastReport .NET, select the Barcode object  at the Components Panel in the Report Designer. In the drop-down list, navigate to the "EAN/UPC" category, and then choose UPC-A:



After selecting the barcode, place it on the Report Page.

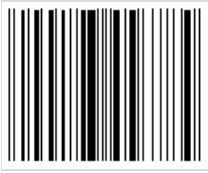
Double-click on the added barcode to open the editor. You can also open the barcode editor by clicking the button  in the context menu of the added object, accessed by right-clicking:



All objects of the Barcode type have a set of properties. Here are the most commonly used ones:

| Property                      | Description  |
|-------------------------------|--|
| <b>Angle</b>                  | Allows you to set the rotation of the object to one of the fixed angles – 0, 90, 180, or 270 degrees.  |
| <b>Zoom</b>                   | Sets the scaling of the barcode. This property is used only with the <b>AutoSize</b> property.   |
| <b>AutoSize</b>               | If this property is enabled, the object will be stretched to show the entire barcode. If the property is disabled, the barcode will be stretched to the size of the object.                        |
| <b>ShowText</b>               | Determines whether to show the text at the bottom of the barcode.  |
| <b>DataColumn</b>             | The data field from which to load the text of the object.  |
| <b>Expression</b>             | An expression that returns the text of the object.   |
| <b>Text</b>                   | The text of the object.  |
| <b>Padding</b>                | Allows you to set the padding from the edges of the object in pixels.  |
| <b>WideBarRatio</b>           | This property is available for all linear barcodes. It defines the relative size of the barcode's wide bars.   |
| <b>CalcChecksum</b>           | This property is available for many linear barcodes. It determines whether to calculate the checksum automatically. If this property is disabled, the checksum must be present in the object text. |
| <b>DrawVerticalBearerBars</b> | If this property is enabled, the side lines will be displayed for the object.  |

If the `ShowText` property is disabled, the barcode will look like this:



# UPC-E



UPC-E is a linear digital barcode designed to encode product information in retail. As you know, the UPC standard is based on EAN. Namely, the American UPC-A is an analog of the European EAN-13. The main difference is only the length of the code: 12 characters versus 13. EAN-13 has a shortened version EAN-8, created for more compact placement on small packages. Similarly, UPC-A has a "lite" version of UPC-E, consisting of 6 characters. As a rule, the product and item codes used in UPC-A contain many zeros. By removing zeros from these codes, it was possible to reduce the barcode length from 12 to 6 characters without losing information.


The structure of the code:

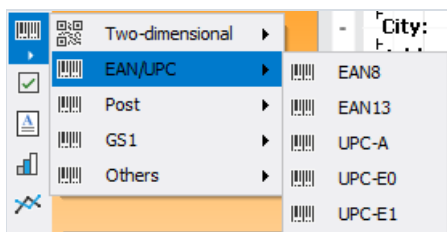
- the first character prefix denotes the number system 0 or 1. This determines how the code will be interpreted;
- then 6 characters of the manufacturer and product code (in UPC-A, they are separated);
- the last character is a checksum digit to verify the integrity of the code.

UPC-E can actually encode 6 to 8 characters. When encoding a minimum set of 6 characters, the first character denoting the number system, as well as the last character, the checksum digit, are excluded. When encoding 7 characters, only the checksum digit is excluded. The eight-character code includes the entire code structure described above. The shortened version of UPC lacks start and stop characters—in favor of compactness.


Depending on which number system is selected, the checksum is calculated automatically. Thus, the UPC-E standard is divided into UPC-E0 and UPC-E1. It is important to note that UPC-E can only encode GTIN-12 structures with a leading 0 and a sequence of zeros in the code. This sequence of zeros will eventually be eliminated in the UPC-E code.

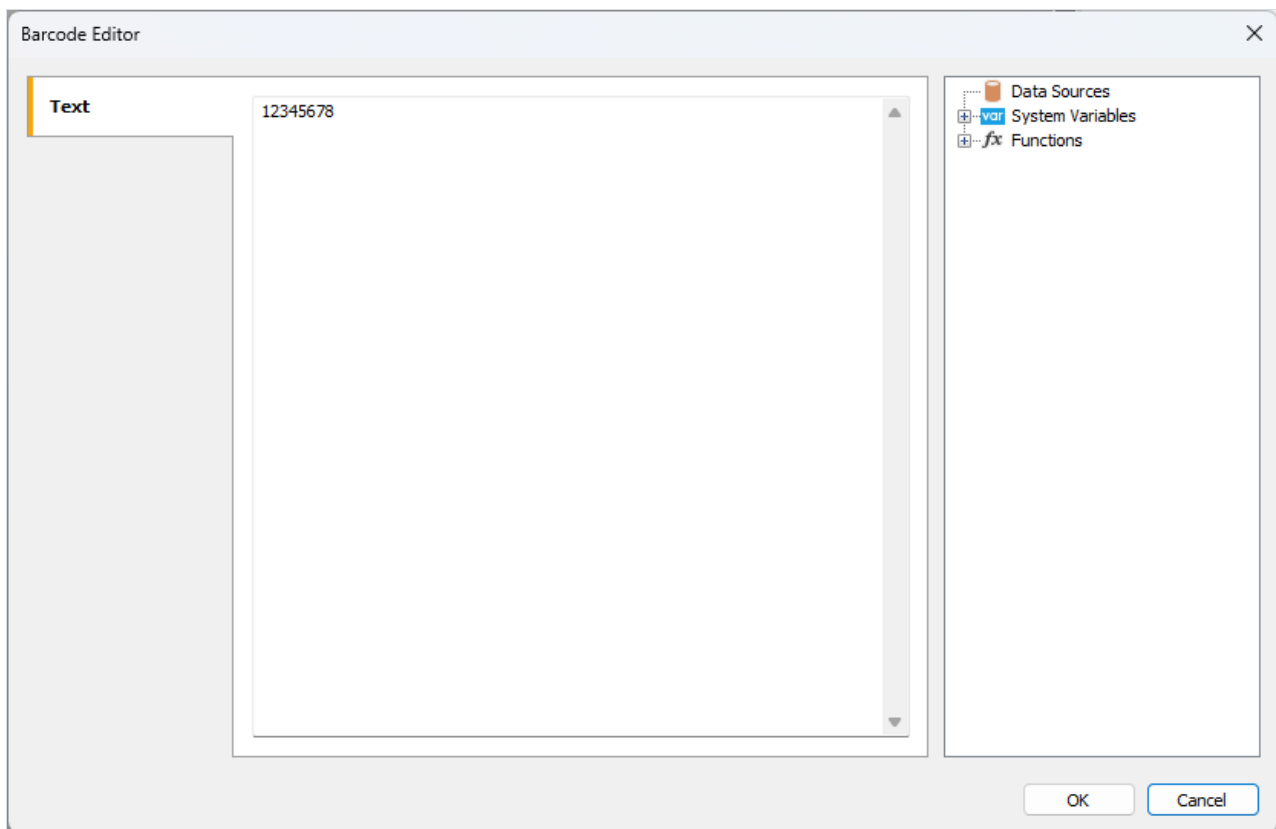
As in UPC-A, each data character in UPC-E is encoded with two bars and two spaces. A bar or space can be 1, 2, 3, or 4 modules wide (one module is 0.33 mm).

To generate an UPC-E0 or UPC-E1 barcode in FastReport .NET, select the Barcode object  at the Components Panel in the Report Designer. In the drop-down list, navigate to the "EAN/UPC" category, and then choose UPC-E0 or UPC-E1:



After selecting the barcode, place it on the Report Page.

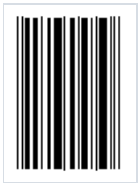
Double-click on the added barcode to open the editor. You can also open the barcode editor by clicking the button  in the context menu of the added object, accessed by right-clicking:



All objects of the Barcode type have a set of properties. Here are the most commonly used ones:

| Property                      | Description  |
|-------------------------------|--|
| <b>Angle</b>                  | Allows you to set the rotation of the object to one of the fixed angles – 0, 90, 180, or 270 degrees.  |
| <b>Zoom</b>                   | Sets the scaling of the barcode. This property is used only with the <code>AutoSize</code> property.   |
| <b>AutoSize</b>               | If this property is enabled, the object will be stretched to show the entire barcode. If the property is disabled, the barcode will be stretched to the size of the object.                        |
| <b>ShowText</b>               | Determines whether to show the text at the bottom of the barcode.  |
| <b>DataColumn</b>             | The data field from which to load the text of the object.  |
| <b>Expression</b>             | An expression that returns the text of the object.   |
| <b>Text</b>                   | The text of the object.  |
| <b>Padding</b>                | Allows you to set the padding from the edges of the object in pixels.  |
| <b>WideBarRatio</b>           | This property is available for all linear barcodes. It defines the relative size of the barcode's wide bars.   |
| <b>CalcChecksum</b>           | This property is available for many linear barcodes. It determines whether to calculate the checksum automatically. If this property is disabled, the checksum must be present in the object text. |
| <b>DrawVerticalBearerBars</b> | If this property is enabled, the side lines will be displayed for the object.  |

If the `ShowText` property is disabled, the barcode will look like this:



# Deutsche Post Identcode



The Deutsche Post Identcode linear barcode is based on the popular 2 of 5 Interleaved barcode and uses its symbology. The differences lie in the purpose of the code. Identcode was created specifically for Deutsche Post to identify the postal sender. Based on the required size of the sender data, the code length has been reduced to 12 digits compared to 14 for Interleaved. However, the actual length of the encoded data is 13 characters, since the 13th is used for the checksum, which is necessary to verify the correctness of the code reading.

Identcode has the following structure:


- start character;
- 2 digits for the primary distribution center identifier;
- 3 digits for the customer ID;
- 6 digits for the zip code;
- check digit;
- stop character.

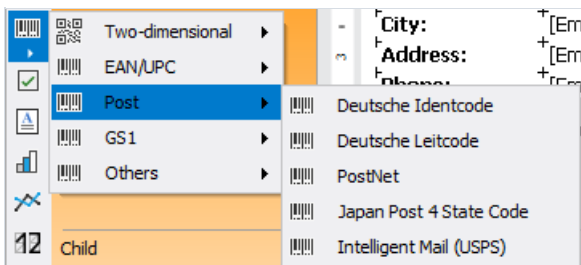
If we represent bars and spaces as binary characters, then the start character is designated as follows: 1010. This character helps determine the reading direction.

The 2 of 5 Interleaved code table is used to encode data digits. Each data character consists of 5 bars or 5 spaces, of which two must be wide and the rest narrow. It is important to note that all codes of the 2 of 5 family use both bars and spaces for encoding. Therefore, the code length is always even, since one set of 5 bars and 5 spaces between them encodes two digits at once. This ensures the high density and compactness of the code.


After 11 data digits, a checksum follows, which is calculated using the Modulo 10 algorithm.

The code ends with a stop character, which is encoded as a sequence of 101.

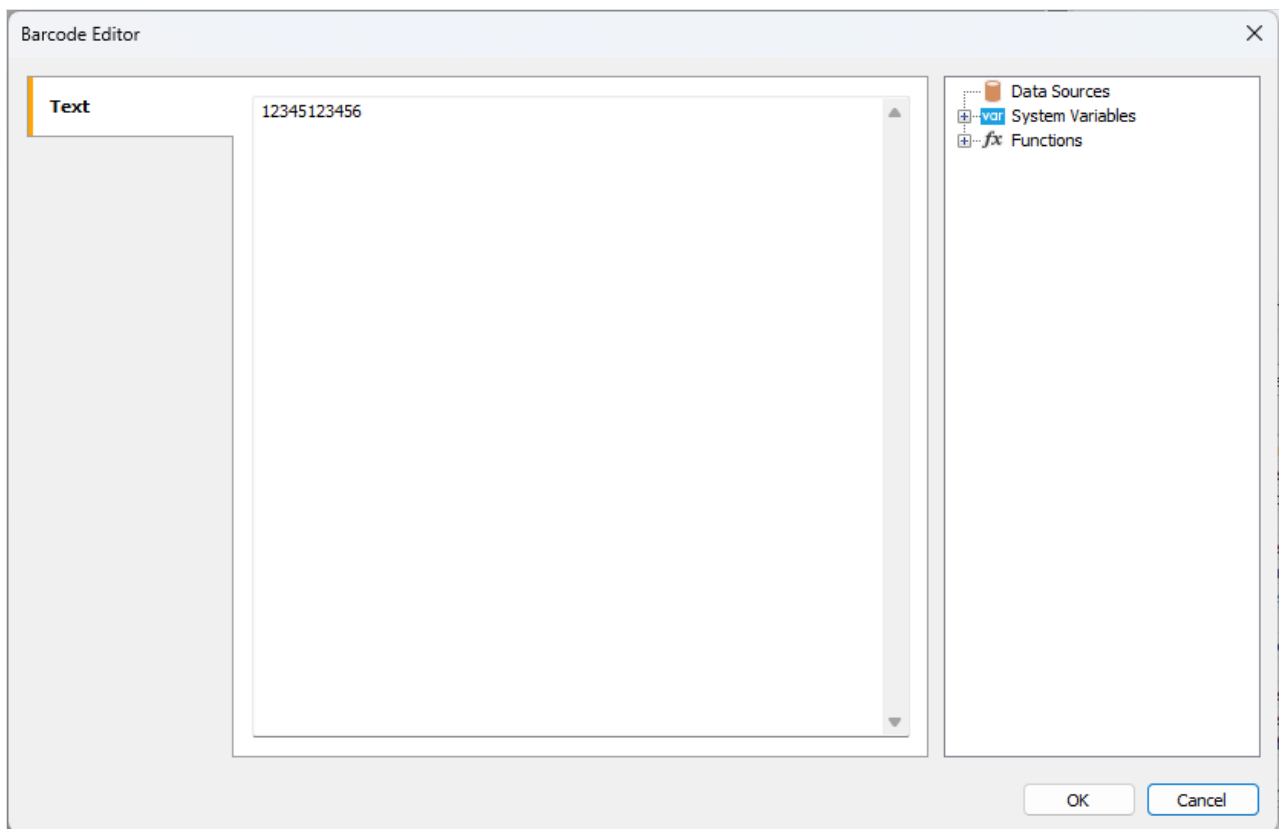
To generate a Deutsche Post Identcode barcode in FastReport .NET, select the Barcode object  at the Components Panel in the Report Designer. In the drop-down list, navigate to the "Post" category, and then choose Deutsche Identcode:



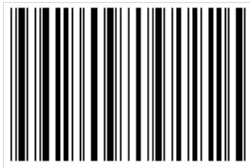
After selecting the barcode, place it on the Report Page.

Double-click on the added barcode to open the editor. You can also open the barcode editor by clicking the button  in the context menu of the added object, accessed by right-clicking:





If you want to hide the text under the barcode, locate the `ShowText` property in the property inspector of the corresponding barcode and set its value to `False` :



# Deutsche Post Leitcode




Deutsche Post Leitcode linear barcode, or simply LeitCode, is based on the widely known 2 of 5 Interleaved code. It allows you to encode 13 digital characters (0-9) and is used in the German postal service (Deutsche Post) and DHL to automate the sorting of mail items. A distinctive feature of Leitcode is its well-defined data structure, which distinguishes it from 2 of 5 Interleaved barcodes.

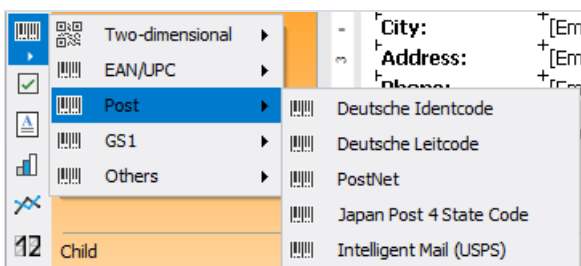
Code structure:

- start character;
- 1-5 – ZIP code;
- 6-8 – street number;
- 9-11 – house number;
- 12-13 – product code;
- 14 – check digit;
- stop character.


Another difference between Leitcode and 2 of 5 Interleaved is the use of the Modulo 10 algorithm to calculate the mandatory checksum.

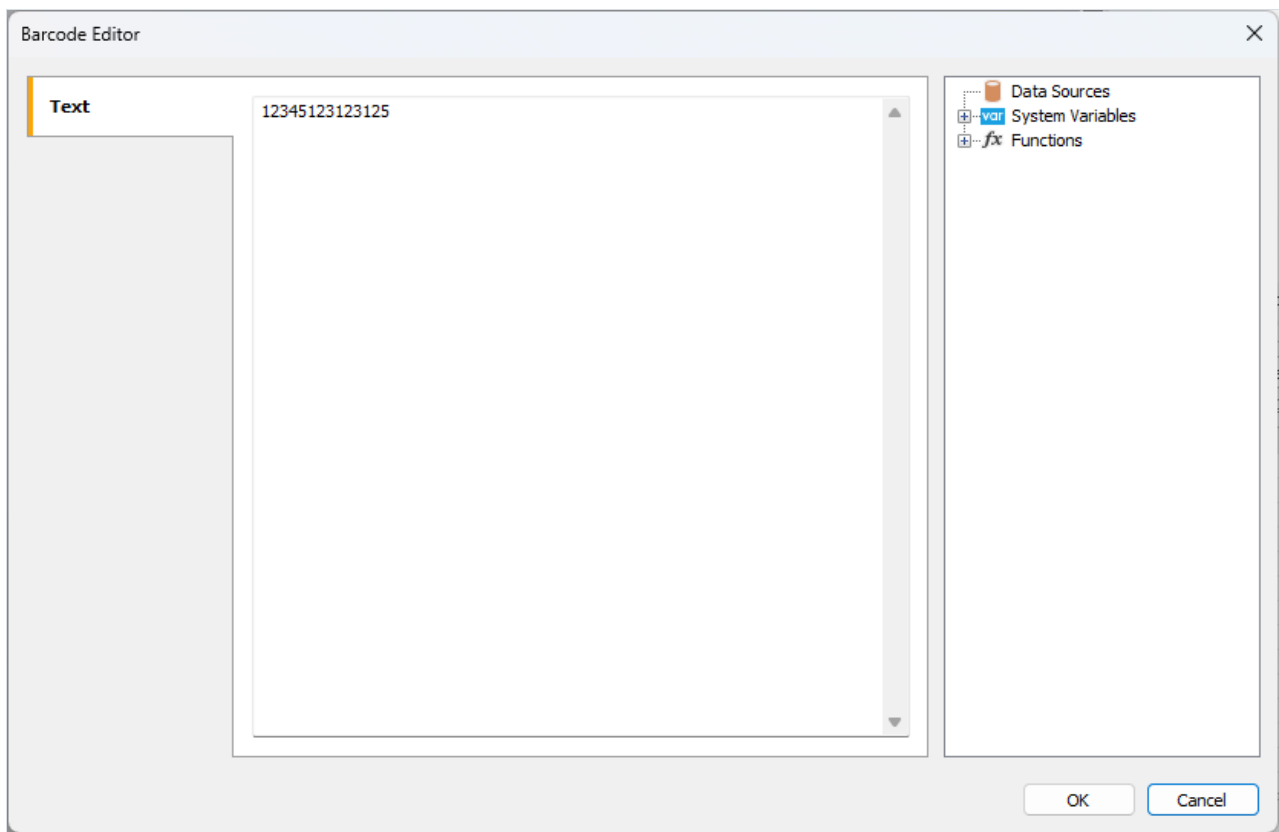
As in the case with 2 of 5 barcodes, Leitcode uses both bars and spaces to encode data. This approach has helped reduce the size of the code. The data character in Leitcode is encoded with five modules (bars/spaces), where two bars are wide and three are narrow. The same applies to spaces. It is thanks to this combination that the barcode got its name 2 of 5.

To generate a Deutsche Post Leitcode barcode in FastReport .NET, select the Barcode object  at the Components Panel in the Report Designer. In the drop-down list, navigate to the "Post" category, and then choose Deutsche Leitcode:

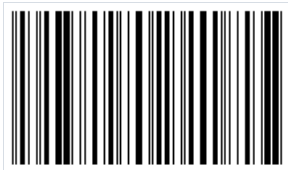


After selecting the barcode, place it on the Report Page.

Double-click on the added barcode to open the editor. You can also open the barcode editor by clicking the button  in the context menu of the added object, accessed by right-clicking:



If you want to hide the text under the barcode, locate the `ShowText` property in the property inspector of the corresponding barcode and set its value to `False` :



# PostNet



This linear numeric barcode was created specifically for use in the United States Postal Service and is designed for machine mail sorting. The appearance of this barcode is very different from the familiar UPC and EAN, which are used to label products in retail. The bars have different heights, and overall the code also has a small height for easy placement on the envelope.

The length of the ZIP code in the US was originally 5 characters, but in 1983, it was increased to 9 characters due to insufficient capacity. But then the code was expanded by 2 more characters. As a result, PostNet has three sizes:

- 5 characters;
- 9 characters;
- 11 characters.

Code structure:

- start character;
- data;
- check character;
- stop character.


Each character is encoded with 5 bars. The start and stop characters are denoted by a single long bar.

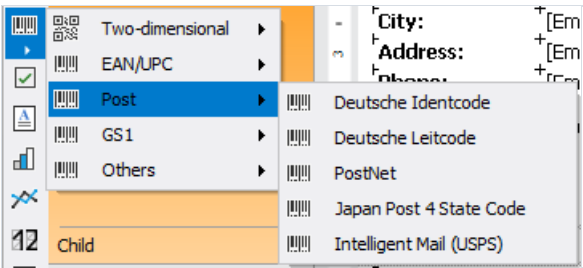
Here is the list of the correspondence between numbers and barcodes:

- 1.
- 2.
- 3.
- 4.
- 5.
- 6.
- 7.
- 8.
- 9.
- 10.


Due to the rather large number of bars required to encode a single character, the PostNet barcode is quite large. In addition, it allows you to encode only numbers. These shortcomings have led to a decline in its popularity, and it is currently being gradually replaced by a more modern solution – Intelligent Mail.

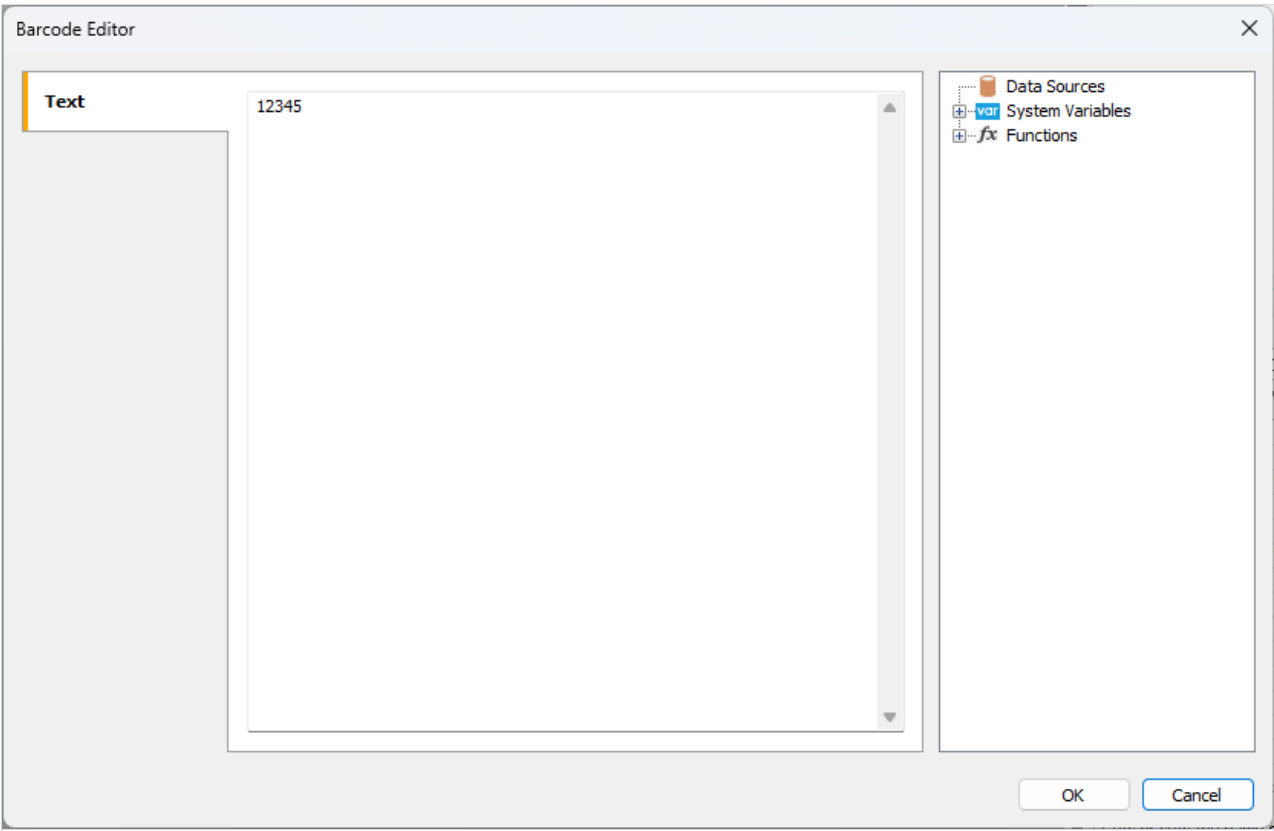
There are no restrictions on the number of characters for the PostNet barcode in FastReport .NET. However, it is worth remembering that the standard provides three options for the code size, which were discussed earlier.

To generate a PostNet barcode in FastReport .NET, select the Barcode object  at the Components Panel in the Report Designer. In the drop-down list, navigate to the "Post" category, and then choose PostNet:



After selecting the barcode, place it on the Report Page.

Double-click on the added barcode to open the editor. You can also open the barcode editor by clicking the button  in the context menu of the added object, accessed by right-clicking:



Like all barcodes in FastReport .NET, the PostNet code has several properties that can be edited in the object property inspector:

| Property | Description   |
|----------|---|
| Angle    | Allows you to set the rotation of the object to one of the fixed angles – 0, 90, 180, or 270 degrees.   |
| Zoom     | Sets the scaling of the barcode. This property is used only with the <code>AutoSize</code> property.  |
| AutoSize | If this property is enabled, the object will be stretched to show the entire barcode. If the property is disabled, the barcode will be stretched to the size of the object. |
| ShowText | Determines whether to show the text at the bottom of the barcode.   |

| Property                      | Description  |
|-------------------------------|--|
| <b>DataColumn</b>             | The data field from which to load the text of the object.  |
| <b>Expression</b>             | An expression that returns the text of the object.   |
| <b>Text</b>                   | The text of the object.  |
| <b>Padding</b>                | Allows you to set the padding from the edges of the object in pixels.  |
| <b>WideBarRatio</b>           | This property is available for all linear barcodes. It defines the relative size of the barcode's wide bars.   |
| <b>CalcChecksum</b>           | This property is available for many linear barcodes. It determines whether to calculate the checksum automatically. If this property is disabled, the checksum must be present in the object text. |
| <b>DrawVerticalBearerBars</b> | If this property is enabled, the side lines will be displayed for the object.  |

# Intelligent Mail

IMb (Intelligent Mail barcode), also known as USPS OneCode, is a height-modulated barcode that encodes up to 31 digits of mail sending data in 65 vertical bars, using symbols with 4 states. This symbolism uses four different states of "bars", allowing for encoding more information in a single barcode.



The Intelligent Mail standard was created on the basis of the POSTNET and PLANET standards, which were used in the USA for mail earlier.

These standards allow for the encoding of zip codes and serve mainly for mail sorting and tracking.

POSTNET is able to encode a five-digit postal code, a 4-digit plus code and a 2-digit delivery point code.

POSTNET barcodes have variable lengths ranging from 32 to 62 bars, while PLANET barcodes consist of either 62 or 72 bars. They are modulated in height (vertical columns have different lengths) and have two states (short and long bars). Each digit of encoded data is represented by a group of five bars. In POSTNET barcodes, each group of five bars contains exactly two full bars, while in PLANET barcodes there are three.

IMb technology effectively combines the capabilities of PLANET and POSTNET in one barcode. It allows mail users to use a single barcode to participate in several mail services at the same time, extends the ability of mail users to track individual mail items and provides greater visibility of the mail flow. The use of this barcode allows the Postal Service to provide multiple services to the mailing industry and additional mail tracking features, as well as to track mail performance and reduce costs. Like POSTNET, IMb has a checksum to check the integrity of the code and possible recovery of damaged code.

Compared to POSTNET, IMb has a much larger data capacity (31 characters versus 11). In addition to the routing code, the Intelligent Mail barcode includes four additional fields: barcode identifier, service type identifier (STID), mail program identifier (MID) and serial number. These additional fields allow mail users to define the class of mail, identify the services they want to purchase (e.g., tracking and correcting the address), and allow mail clients to uniquely identify mail items.

The disadvantages include a sufficiently long code and the ability to encode only numbers.

Intelligent Mail allows encoding the following information:

| Information                            | Description  |
|--|--|
| Barcode Identifier (2 characters)      | Assigned by the United States Postal Service.                                  |
| Service type identifier (3 characters) | Mail class or other service.   |
| Sender ID (6 or 9 characters)          | Company ID assigned by the United States Postal Service.                       |
| Serial number (6 or 9 characters)      | The sender is assigned a number to identify a specific recipient or household. |

## Information


## Description

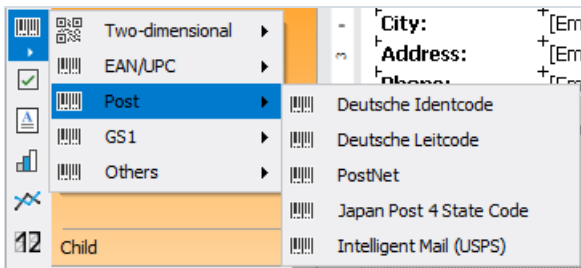
**Postal code of the delivery point (11 characters)**

Not required field.


Hatches have different height, direction (up, down) and thickness.

Such a barcode can be printed not only on an envelope but also directly on a document. It can then be read through a special transparent window on the envelope.

To generate an Intelligent Mail barcode in FastReport .NET, select the Barcode object  at the Components Panel in the Report Designer. In the drop-down list, navigate to the "Post" category, and then choose Intelligent Mail (USPS):



After selecting the barcode, place it on the Report Page.

Double-click on the added barcode to open the editor. You can also open the barcode editor by clicking the button  in the context menu of the added object, accessed by right-clicking.

In FastReport, the minimum length of the code is 20 characters. This is because the Postal Code of the delivery point, which is not required, begins with 21 characters.

Changing just one digit completely changes the barcode:





# GS1-128

This barcode is also known as EAN-128 and UCC-128. It is a one-dimensional linear barcode, with the ability to encode both numbers and letters. It is most popular in the packaging labeling field, but is also used in other areas. The GS1 Logistic Label standard describes a logistic shipping label that contains the GS1-128 code.

A set of Code-128 identifiers known as Application Identifiers (AI) is used to build the code. Each AI denotes a certain type of information and is indicated in brackets, for example, (37) . Knowing the data type enclosed in parentheses allows you to correctly interpret subsequent data.

First, the identifier code is indicated in brackets, then its value. Further, without spaces, the following identifier code and value. Thus, the code will have a value:

(01)12646846874672(10)ABC11(15)100420

Code length is limited to 48 identifiers excluding parentheses.

The list of supported data type identifiers is quite large, here is an example of some of them:

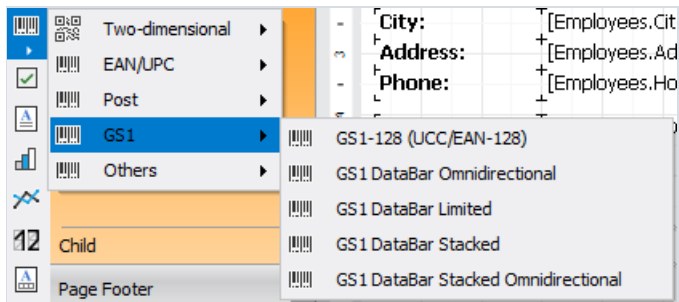
| Code | Description  | Format   |
|------|--|----------|
| 00   | Serial Shipping Container Code (SSCC)                                  | N2+N18   |
| 01   | Global Trade Item Number (GTIN)  | N2+N14   |
| 02   | Identification of trade items contained in a logistic unit             | N2+N14   |
| 10   | Batch or lot number  | N2+X..20 |
| 11   | Production date (YYMMDD)   | N2+N6    |
| 12   | Due date for amount on payment slip (YYMMDD)                           | N2+N6    |
| 13   | Packaging date (YYMMDD)  | N2+N6    |
| 15   | Best before date (YYMMDD)  | N2+N6    |
| 37   | Count of trade items or trade item pieces contained in a logistic unit | N2+N..8  |
| 240  | Additional product identification assigned by the manufacturer         | N3+X..30 |
| 8002 | Cellular mobile telephone identifier                                   | N4+X..20 |

If we use the above example when generating this barcode in FastReport .NET, we will get:




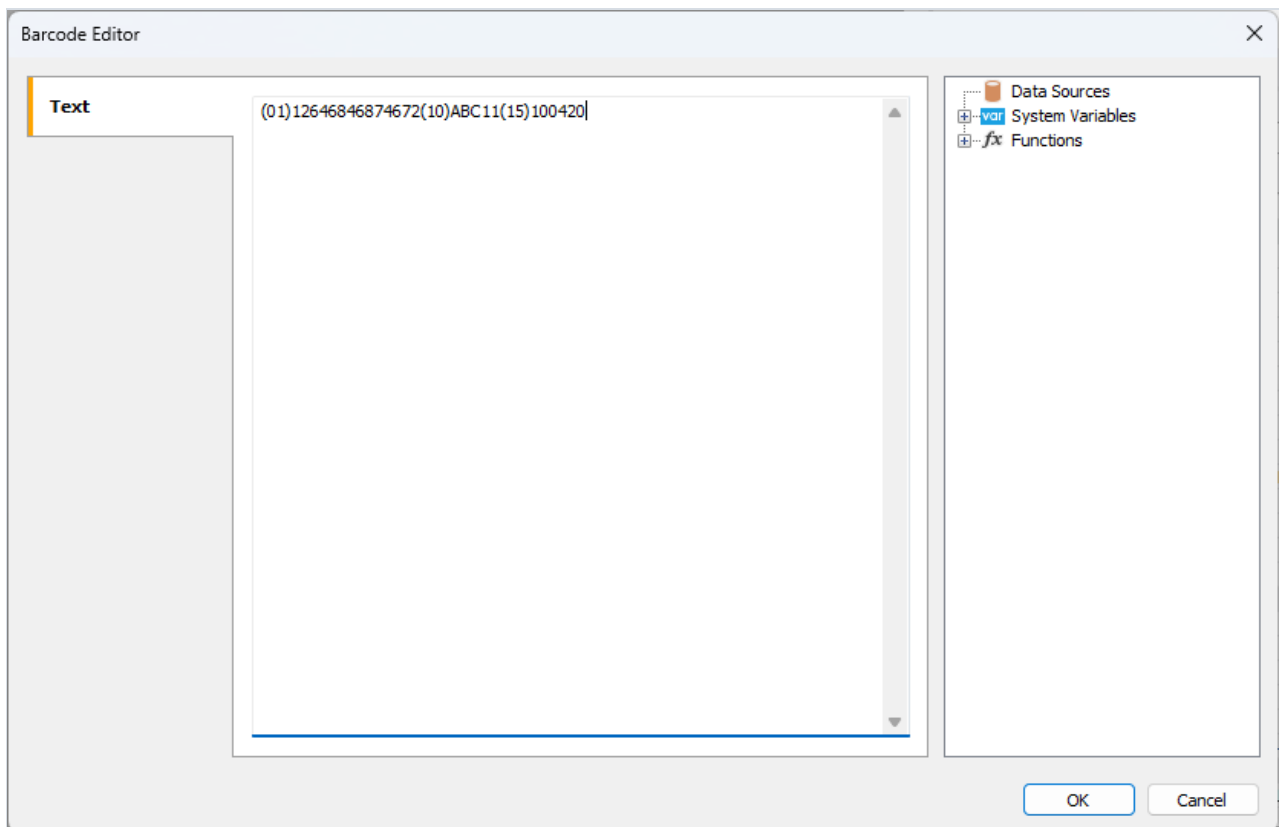
To generate a GS1-128 barcode in FastReport .NET, select the Barcode object  at the Components Panel in the

Report Designer. In the drop-down list, navigate to the "GS1" category, and then choose GS1-128 (UCC/EAN-128):



After selecting the barcode, place it on the Report Page.

Double-click on the added barcode to open the editor. You can also open the barcode editor by clicking the button  in the context menu of the added object, accessed by right-clicking:



# Interleaved 2 of 5



The Interleaved 2 of 5 (ITF 2/5) barcode is a classic linear code for encoding numerical information of any length. It was developed by David Allais in 1972.

A distinctive feature of this barcode is the mandatory requirement of an even number of encoded characters. This is because characters are encoded in pairs. If the code has an odd number of digits, a zero is added before the barcode.

The first character of the pair is encoded with bars, and the second with spaces. The bars and spaces can be narrow or wide. Each character is encoded with five bars or spaces. Two bars are wide, and three are narrow. The same applies to the spaces.


Thanks to this encoding method, the barcode has a high density and takes up less space than, for example, Code 39.

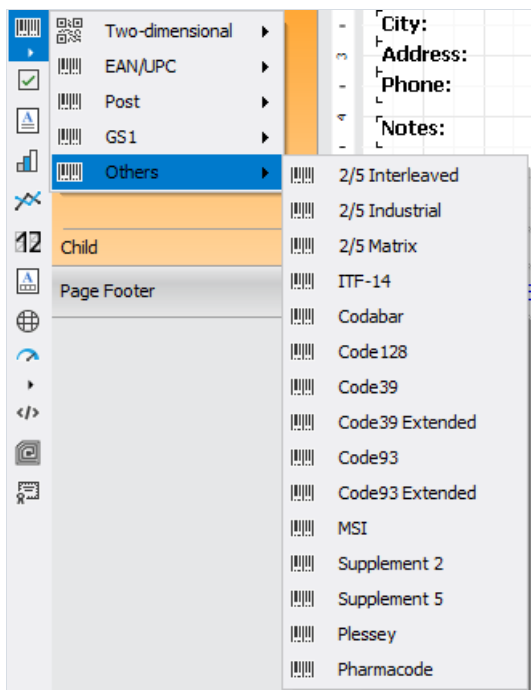
The Interleaved 2 of 5 barcode is used for product labeling in warehouses, as well as in shipping.

Code structure:


- start character – the code start;
- encoded data;
- optional check digit;
- stop character.

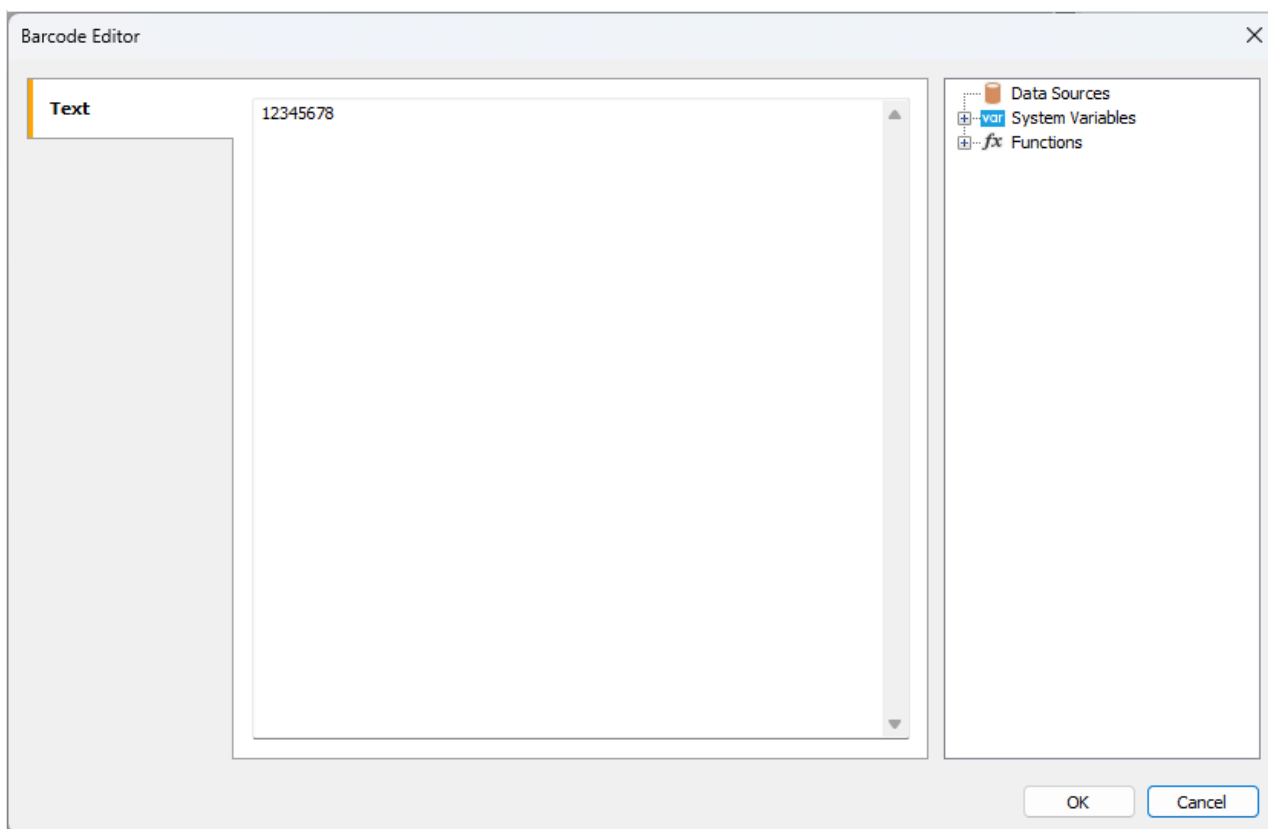
The barcode provides self-checking, so the presence of a check digit is not mandatory.

To generate an Interleaved 2 of 5 barcode in FastReport .NET, select the Barcode object  at the Components Panel in the Report Designer. In the drop-down list, navigate to the "Others" category, and then choose 2/5 Interleaved:

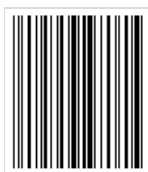


After selecting the barcode, place it on the Report Page.

Double-click on the added barcode to open the editor. You can also open the barcode editor by clicking the button  in the context menu of the added object, accessed by right-clicking:



If you want to hide the text under the barcode, locate the `ShowText` property in the property inspector of the corresponding barcode and set its value to `False` :



# Industrial 2 of 5



Industrial 2 of 5 (also known as Standard 2 of 5) is a low-density linear barcode designed to encode numbers of arbitrary length. Only lines of varying thickness are used for encoding, while spaces serve only as separators between them. Due to this, the barcode has a low density and, consequently, a larger size.


Each character is encoded with five lines, two of which have a larger thickness. This feature is what gives the code its name.

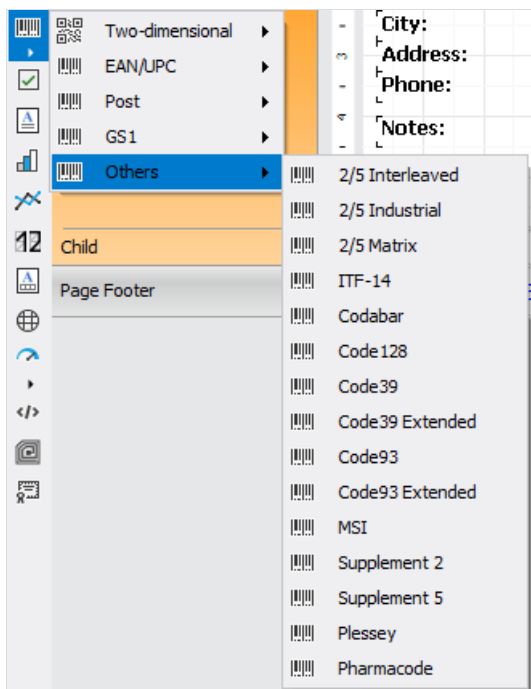
Currently, this barcode is considered outdated. It was used in warehouse inventory management, photo labs, and for numbering airline tickets.

Industrial 2 of 5 provides for a check digit modulo 10, but it is not mandatory.


The standard 2 of 5 barcode has the following physical structure:

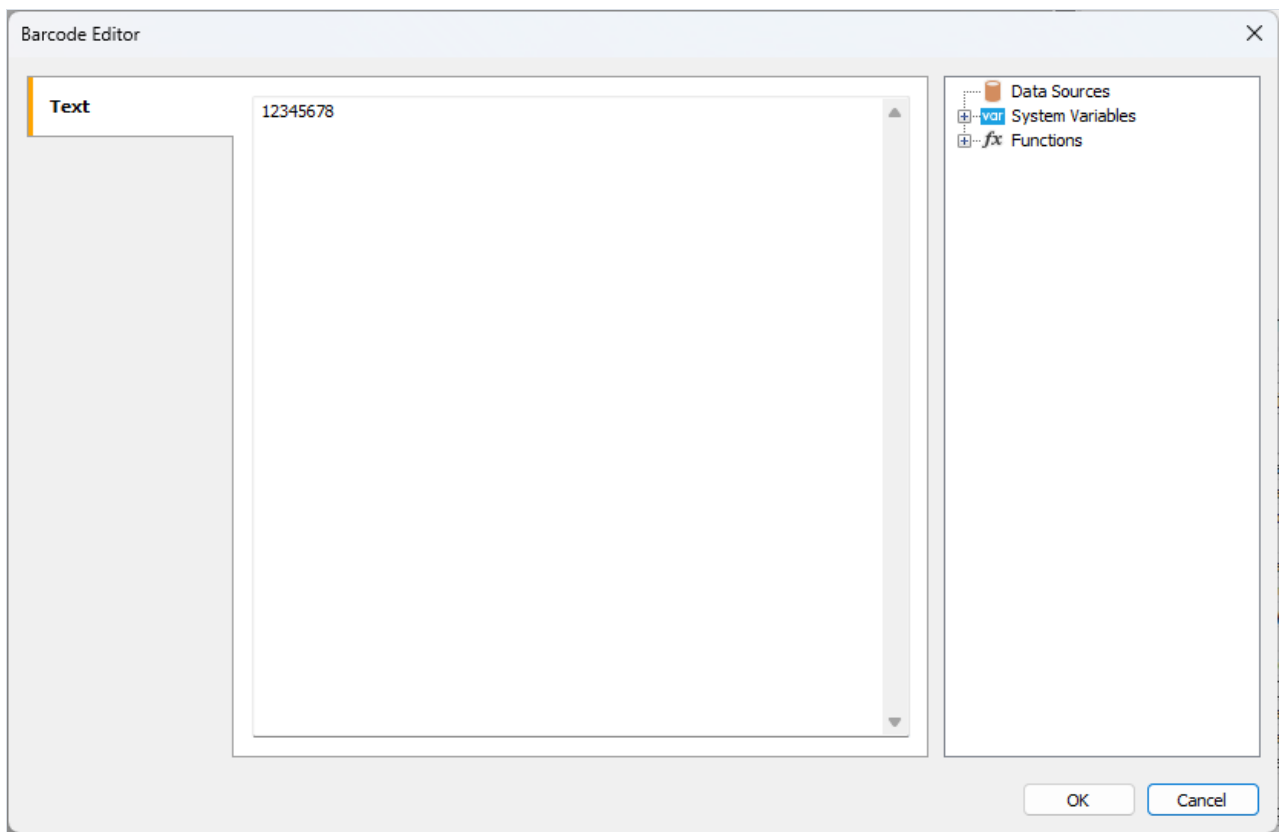
- start character;
- data characters;
- optional check character;
- stop character.

To generate an Industrial 2 of 5 barcode in FastReport .NET, select the Barcode object  at the Components Panel in the Report Designer. In the drop-down list, navigate to the "Others" category, and then choose 2/5 Industrial:

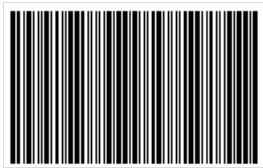


After selecting the barcode, place it on the Report Page.

Double-click on the added barcode to open the editor. You can also open the barcode editor by clicking the button  in the context menu of the added object, accessed by right-clicking:



If you want to hide the text under the barcode, locate the `ShowText` property in the property inspector of the corresponding barcode and set its value to `False` :



## 2 of 5 Matrix



The 2 of 5 Matrix is a high-density linear barcode designed to encode numerical data (0-9) of arbitrary length. It was developed by Identicon in 1968. This type of code provides high density due to the use of spaces between lines for encoding information, resulting in both lines and spaces being used in the encoding process.


The code has built-in self-checking and supports bidirectional reading, meaning it can be read from left to right or right to left.

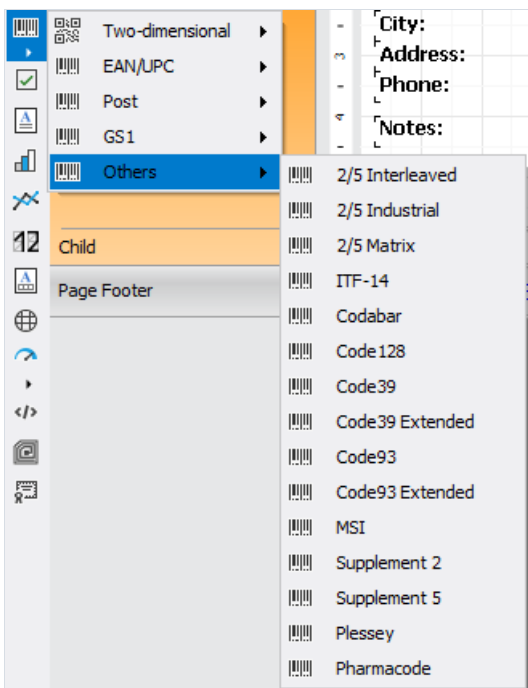
Code structure:

- start character;
- data;
- optional verification character. Calculated using the formula  $\text{mod } 10$  ;
- stop character.


The character consists of five lines, two of which are wide. It is a symbology with two widths, meaning that it defines the width of a thin line, and anything wider is considered a wide line. This allows printing codes with a large tolerance on not-so-high-quality equipment.

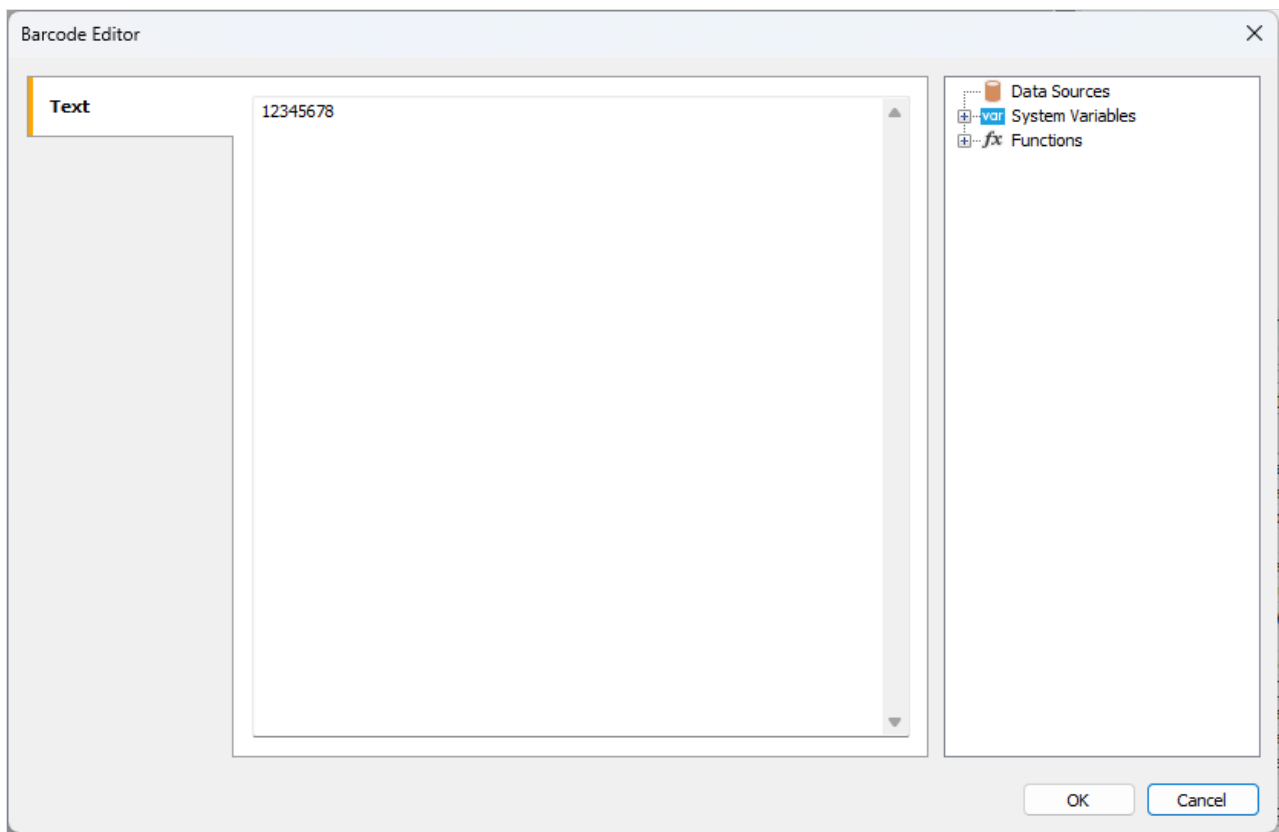
2 of 5 Matrix is used for labeling goods in warehouses, marking airline tickets, and in photo labs.

To generate a 2 of 5 Matrix barcode in FastReport .NET, select the Barcode object  at the Components Panel in the Report Designer. In the drop-down list, navigate to the "Others" category, and then choose 2/5 Matrix:

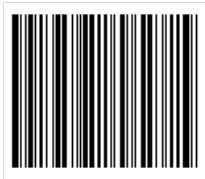


After selecting the barcode, place it on the Report Page.

Double-click on the added barcode to open the editor. You can also open the barcode editor by clicking the button  in the context menu of the added object, accessed by right-clicking:



If you want to hide the text under the barcode, locate the `ShowText` property in the property inspector of the corresponding barcode and set its value to `False` :





# ITF-14



ITF-14 is a two-digit numeric code, also known as a high-density code. It encodes only an even number of digits. In this type of barcode, each odd digit is represented by a dark line and each even digit is represented by a space between them. To encode an odd number of digits, the leftmost (most significant) digit needs to be padded with a zero.

The barcode has the following structure:


- the first character – indicator denotes a level of packaging for a specific carton. This one-digit prefix can vary from 0 to 8 (for example, 1 – carton, 2 – case, and so on);
- next 2-3 digits – country regional code (prefix), where this number is registered;
- next 4-5 digits represent a registration number of the enterprise within the national organization;
- next group of digits denotes a serial number of production within the enterprise;
- the last 13th digit is a checksum or check digit. It is calculated from the previous twelve according to the Module 10 algorithm.

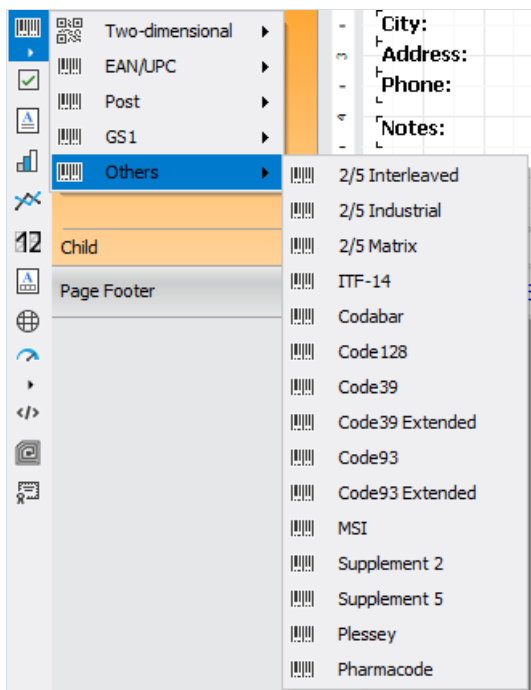
ITF-14 barcode is used for automating warehouse accounting of goods placed in individual or group transport packaging. It allows the computer inventory system to determine not only the type of product in the package but also its quantity.

ITF-14 is commonly used for printing on corrugated cardboard and for marking cardboard boxes, cases, or pallets. These barcodes are widely used by retailers, manufacturers, and distributors for precise logistics control and inventory management. In addition, they can be used for baggage identification at airports, numbering of air tickets, and identification of postal items.


Since the ITF-14 barcode is intended for marking goods in transport packages, it does not provide for processing at checkout terminals.

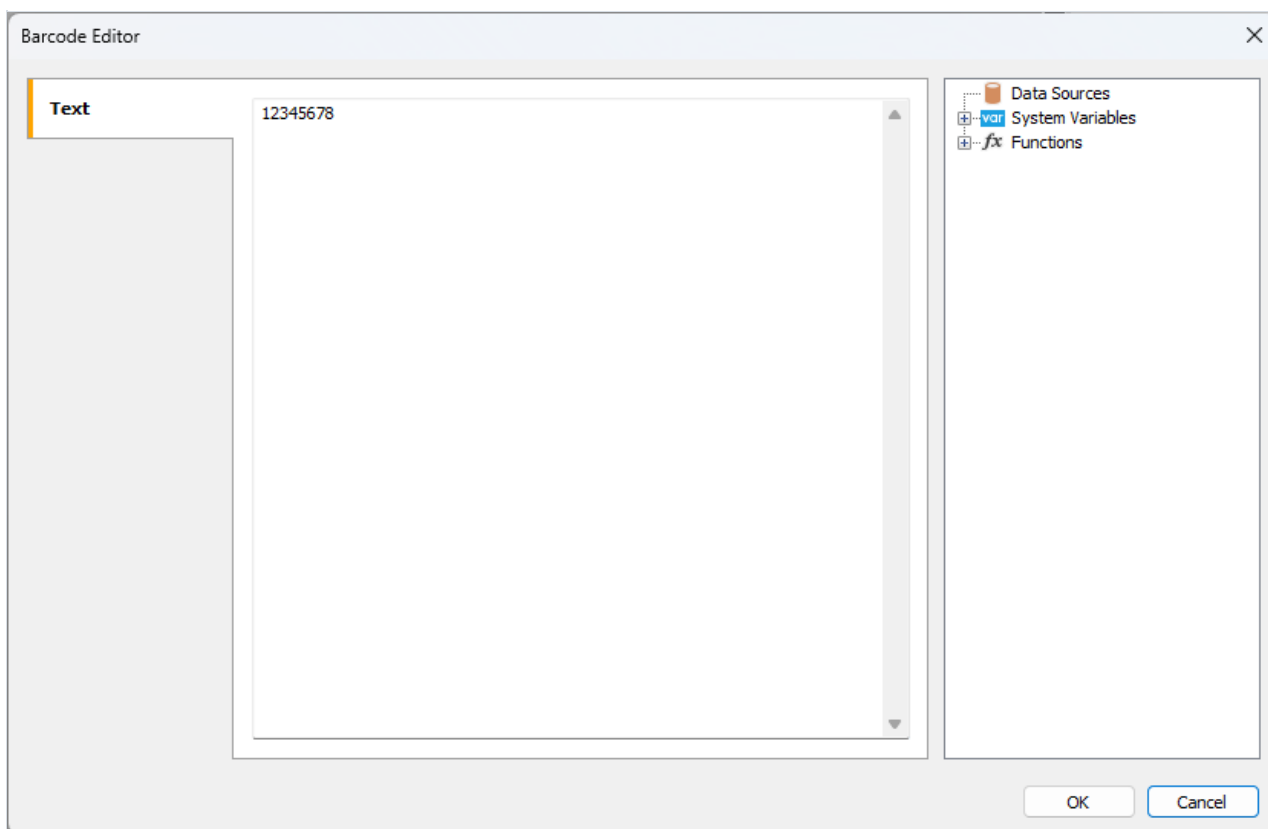
The thick black frame around the barcode is called the Bearer Bar. This bar equalizes the pressure created by the printing plate over the entire surface of the barcode and improves readability, reducing the likelihood of incomplete character scanning. ITF-14 can be with visible or hidden vertical bearer bars.

To generate an ITF-14 barcode in FastReport .NET, select the Barcode object  at the Components Panel in the Report Designer. In the drop-down list, navigate to the "Others" category, and then choose ITF-14:



After selecting the barcode, place it on the Report Page.

Double-click on the added barcode to open the editor. You can also open the barcode editor by clicking the button  in the context menu of the added object, accessed by right-clicking:



Let's look at the properties of the Barcode object:

| Property     | Description   |
|--------------|---|
| <b>Angle</b> | Allows you to set the rotation of the object to one of the fixed angles – 0, 90, 180, or 270 degrees. |
| <b>Zoom</b>  | Sets the scaling of the barcode. This property is used only with the <code>AutoSize</code> property.  |

| Property                      | Description  |
|-------------------------------|--|
| <b>AutoSize</b>               | If this property is enabled, the object will be stretched to show the entire barcode. If the property is disabled, the barcode will be stretched to the size of the object.                        |
| <b>ShowText</b>               | Determines whether to show the text at the bottom of the barcode.  |
| <b>DataColumn</b>             | The data field from which to load the text of the object.  |
| <b>Expression</b>             | An expression that returns the text of the object.   |
| <b>Text</b>                   | The text of the object.  |
| <b>Padding</b>                | Allows you to set the padding from the edges of the object in pixels.  |
| <b>WideBarRatio</b>           | This property is available for all linear barcodes. It defines the relative size of the barcode's wide bars.   |
| <b>CalcChecksum</b>           | This property is available for many linear barcodes. It determines whether to calculate the checksum automatically. If this property is disabled, the checksum must be present in the object text. |
| <b>DrawVerticalBearerBars</b> | If this property is enabled, the side lines will be displayed for the object.  |

If the `DrawVerticalBearerBars` is disabled, the barcode will look like this:



# Codabar



Codabar is a variable-length linear barcode that is most commonly used for encoding serial numbers. It was created in 1972 by Pitney Bowes. Its bars ensure an accurate reading of the code even if the code is printed on a dot-matrix printer.

Symbology:

- numbers 0-9;
- letters A, B, C, D;
- special characters `-`, `$`, `/`, `.`, `+`.


Letters can only be used as a start and finish character for the code. They can be used as additional encoded information.

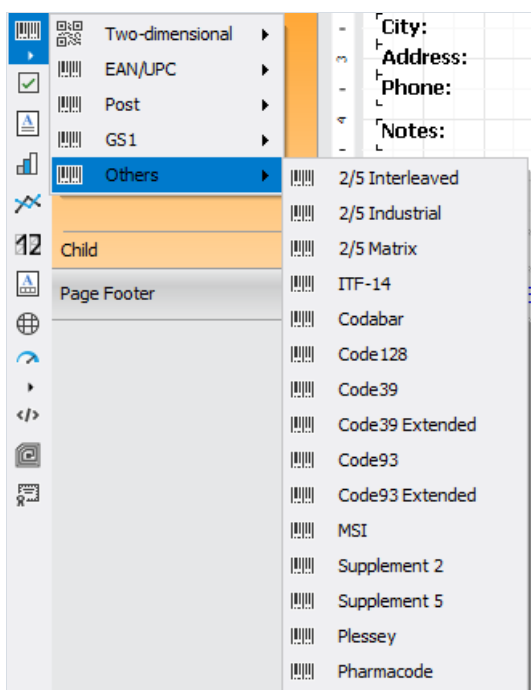
The code structure is extremely simple:

- start character – indicates the beginning of reading data;
- encoded data;
- stop character – indicates the end of the code.


A check digit character is not provided, as the code has built-in self-checking.

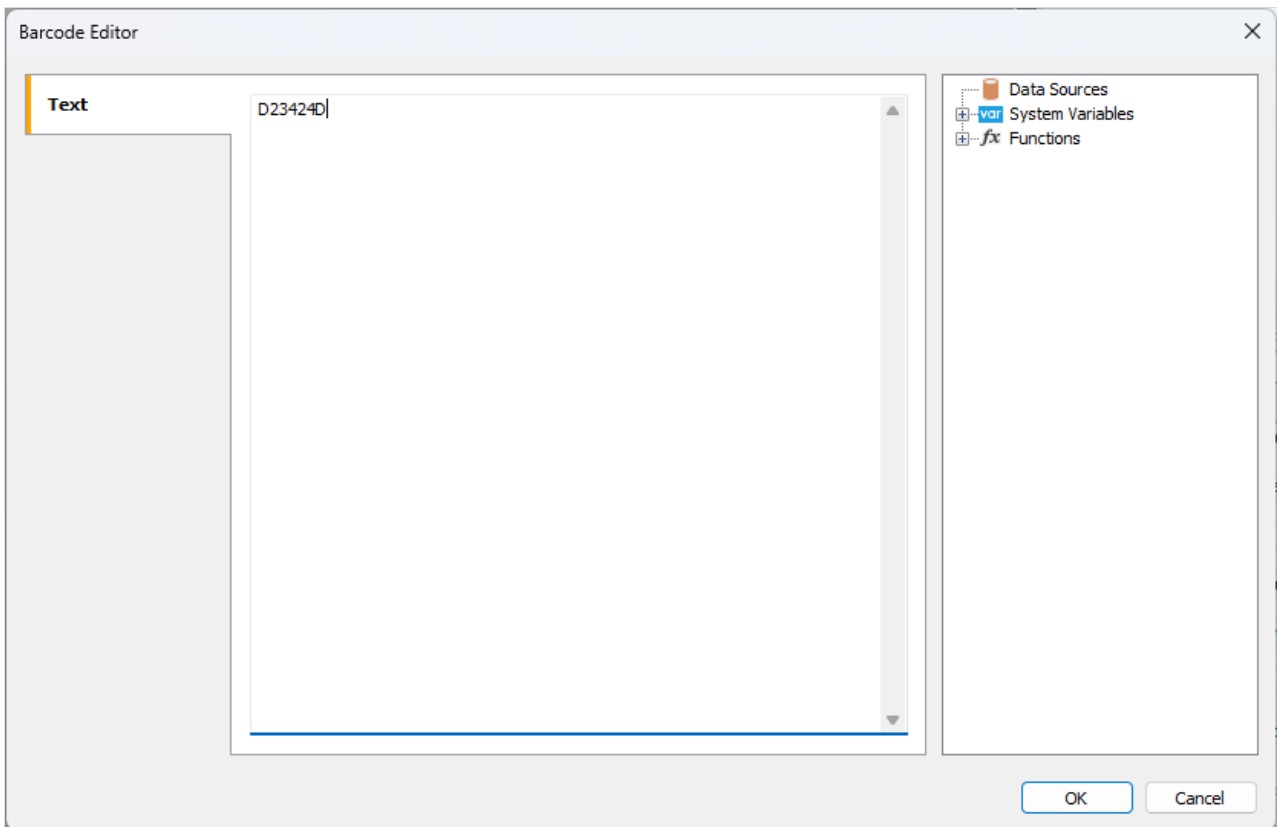
Each character in Codabar is encoded with 4 lines and 3 spaces.

To generate a Codabar barcode in FastReport .NET, select the Barcode object  at the Components Panel in the Report Designer. In the drop-down list, navigate to the "Others" category, and then choose Codabar:



After selecting the barcode, place it on the Report Page.

Double-click on the added barcode to open the editor. You can also open the barcode editor by clicking the button  in the context menu of the added object, accessed by right-clicking:



If you want to hide the text under the barcode, locate the `ShowText` property in the property inspector of the corresponding barcode and set its value to `False` :



# Code 128




Code 128 linear barcode has become widespread in many areas. From its name, it can be understood that it allows 128 characters to be encoded. However, this is not limited to numbers only, as it is also capable of encoding letters and some special characters. This is its main advantage and difference from EAN standard codes.

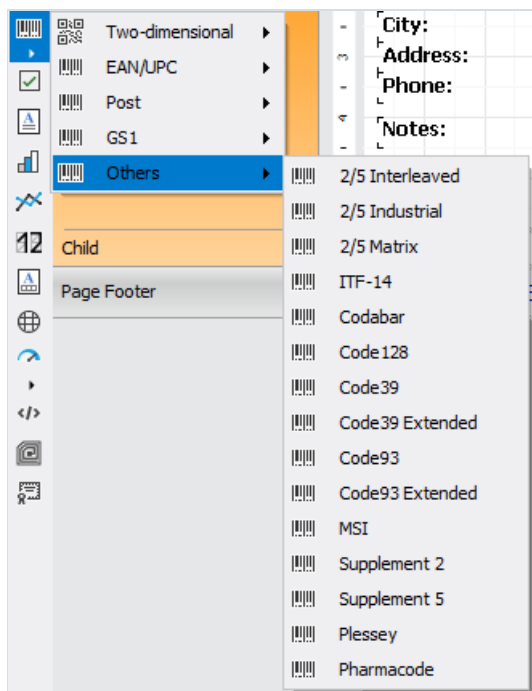
The structure of the code, like most linear barcodes, is simple. On both sides, the code has an empty space, which allows you to clearly define the beginning of the code. Next, the beginning of the code is indicated by a special character – a sequence of lines and spaces. After the start character, the encoded data follows. Then, a checksum character to verify the integrity of the code. Stop code, signaling the end of the encoded data, completes it.

Different information is encoded using different sets of characters for letters, numbers, and special characters. Each character is encoded with three lines and three spaces. Lines and spaces can vary in width. The width is determined by the number of modules – from 1 to 4.


As a result, Code 128 has the following advantages:

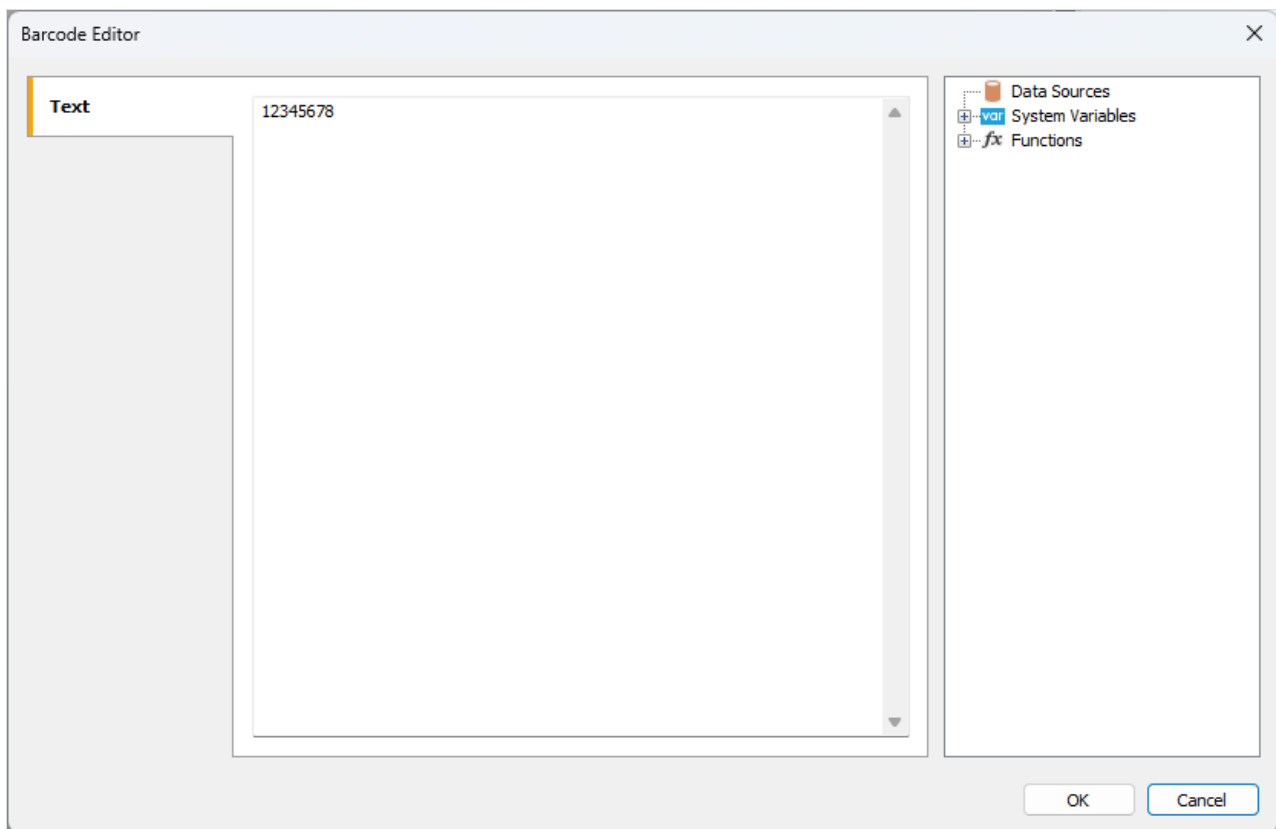
- both numbers and letters (uppercase) can be encoded;
- a purely digital code is very compact due to the use of double packaging.

To generate a Code 128 barcode in FastReport .NET, select the Barcode object  at the Components Panel in the Report Designer. In the drop-down list, navigate to the "Others" category, and then choose Code128:



After selecting the barcode, place it on the Report Page.

Double-click on the added barcode to open the editor. You can also open the barcode editor by clicking the button  in the context menu of the added object, accessed by right-clicking:



If you want to hide the text under the barcode, locate the `ShowText` property in the property inspector of the corresponding barcode and set its value to `False` :



# Code 39



Code 39 is a linear barcode developed in 1975 by Intermec. This code gained great popularity due to its ability to encode alphanumeric data. It is still widely used today.

An important feature of this code is the ability to encode data of arbitrary length. The width of the code is limited only by the scanner's reading capability.


Each character in the code is represented by 9 elements: 5 bars and 4 spaces. This large number of elements is due to built-in self-checking. This means that each character in the code is checked, and there is no need for a check digit. However, adding a check digit is still allowed for extra barcode integrity assurance.

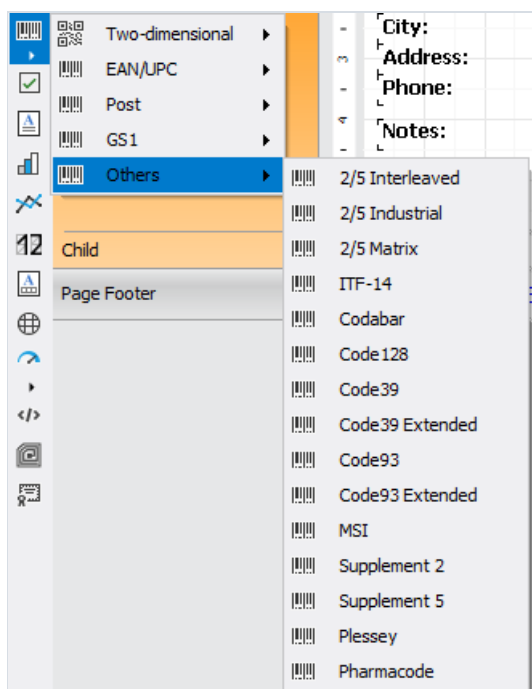
As mentioned above, this barcode allows the encoding of alphanumeric information. The following characters are allowed:

- Latin uppercase letters (A-Z);
- digits (0-9);
- special characters ( - , . \$ / + % : ).

Code 39 barcode has the following structure:


- start character encoded as \* . Indicates the beginning of the code;
- encoded data;
- optional checksum digit;
- stop character, also encoded as \* .

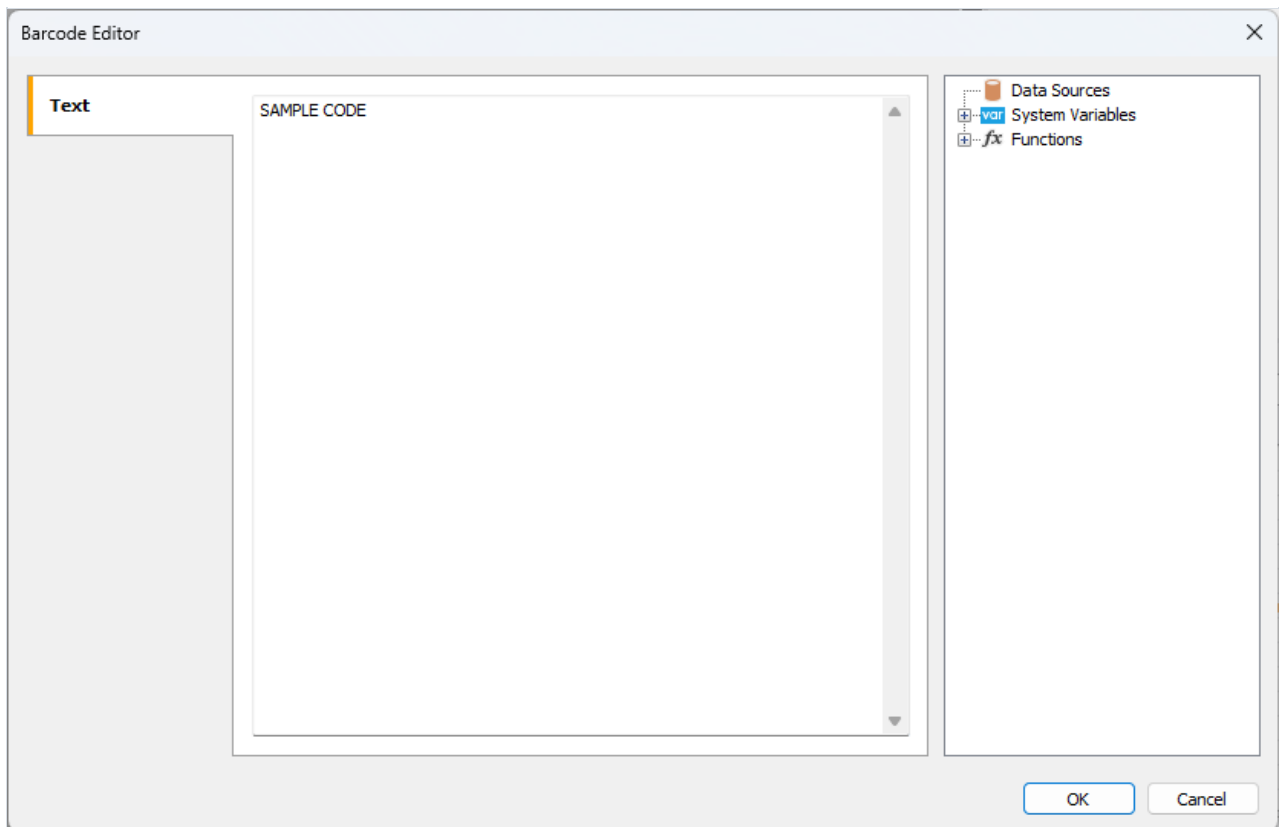
To generate a Code 39 barcode in FastReport .NET, select the Barcode object  at the Components Panel in the Report Designer. In the drop-down list, navigate to the "Others" category, and then choose Code39:





After selecting the barcode, place it on the Report Page.

Double-click on the added barcode to open the editor. You can also open the barcode editor by clicking the button  in the context menu of the added object, accessed by right-clicking:



If you want to hide the text under the barcode, locate the `ShowText` property in the property inspector of the corresponding barcode and set its value to `False` :



# Code 39 Extended



As the name suggests, this is an extended version of Code 39. It supports the full ASCII character set, including uppercase letters A-Z, lowercase a-z, and a large set of special characters. Like regular Code 39, the extended version can contain a check digit, which is calculated using the modulo 43 formula.


The barcode allows the encoding of alphanumeric information. The following characters are allowed:

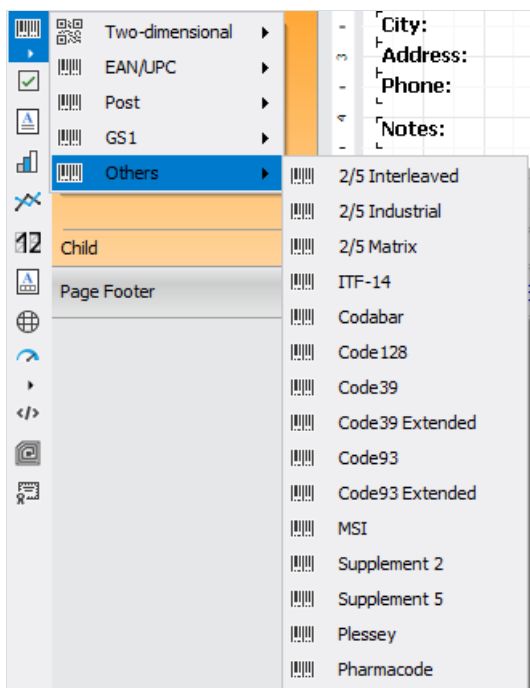
- Latin letters in uppercase (A-Z) and lowercase;
- digits (0-9);
- special characters.

Code 39 Extended has the following structure:

- start character encoded as \* ;
- encoded data;
- optional checksum digit;
- stop character, also encoded as \* .

For creating a lowercase letter, two characters are used: + and the corresponding uppercase letter. To encode some special characters, a combination of two characters is also used: % and an uppercase letter, or / and an uppercase letter.

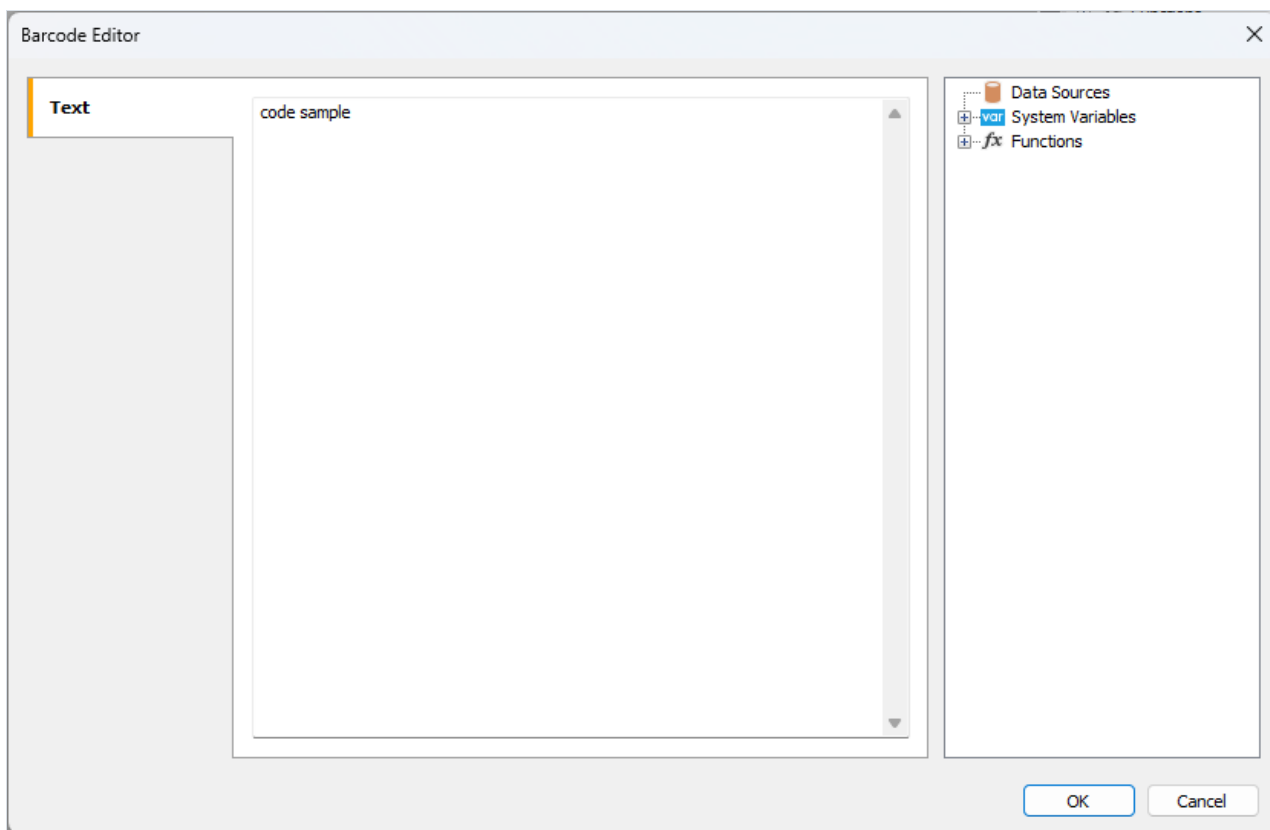
To generate a Code 39 Extended barcode in FastReport .NET, select the Barcode object  at the Components Panel in the Report Designer. In the drop-down list, navigate to the "Others" category, and then choose Code39 Extended:



After selecting the barcode, place it on the Report Page.

Double-click on the added barcode to open the editor. You can also open the barcode editor by clicking the button

 in the context menu of the added object, accessed by right-clicking:



If you want to hide the text under the barcode, locate the `ShowText` property in the property inspector of the corresponding barcode and set its value to `False` :



# Code 93




Code 93 (AIM-BC5-2000 specification) is a linear barcode that allows the encoding of both numbers and letters. It was developed in 1982 by Intermec and based on Code 39. The main advantage over its predecessor is its more compact size.

Code 93 is used by Canada Post to encode information about mail items, as well as in the automotive industry.

The length of the code is limited to 48 characters, of which: 26 uppercase letters, 10 digits, 7 special characters, and 5 service characters, such as the start and end of the code, as well as characters for extending the symbology. This code can represent the full set of ASCII characters using a combination of two service characters.


Code structure:

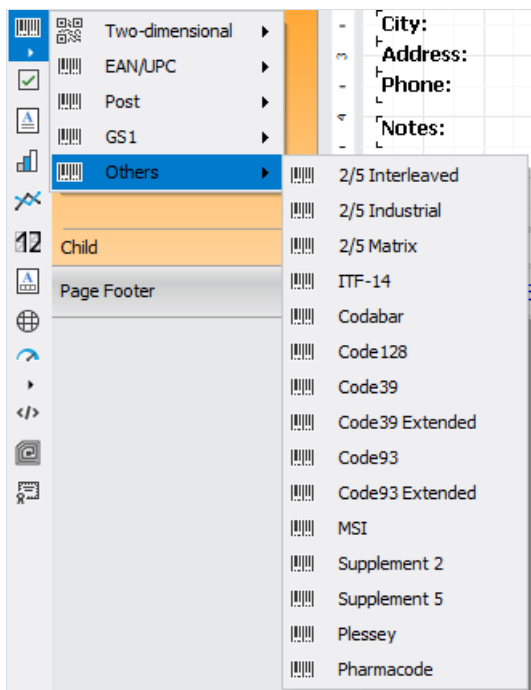
- start character;
- encoded information;
- a check digit calculated using the "C" modulo;
- a check digit calculated using the "K" modulo;
- stop character.

The start character is represented as .


Each data character consists of three lines and three spaces. The width of a line or space can range from 1 to 4 modules.

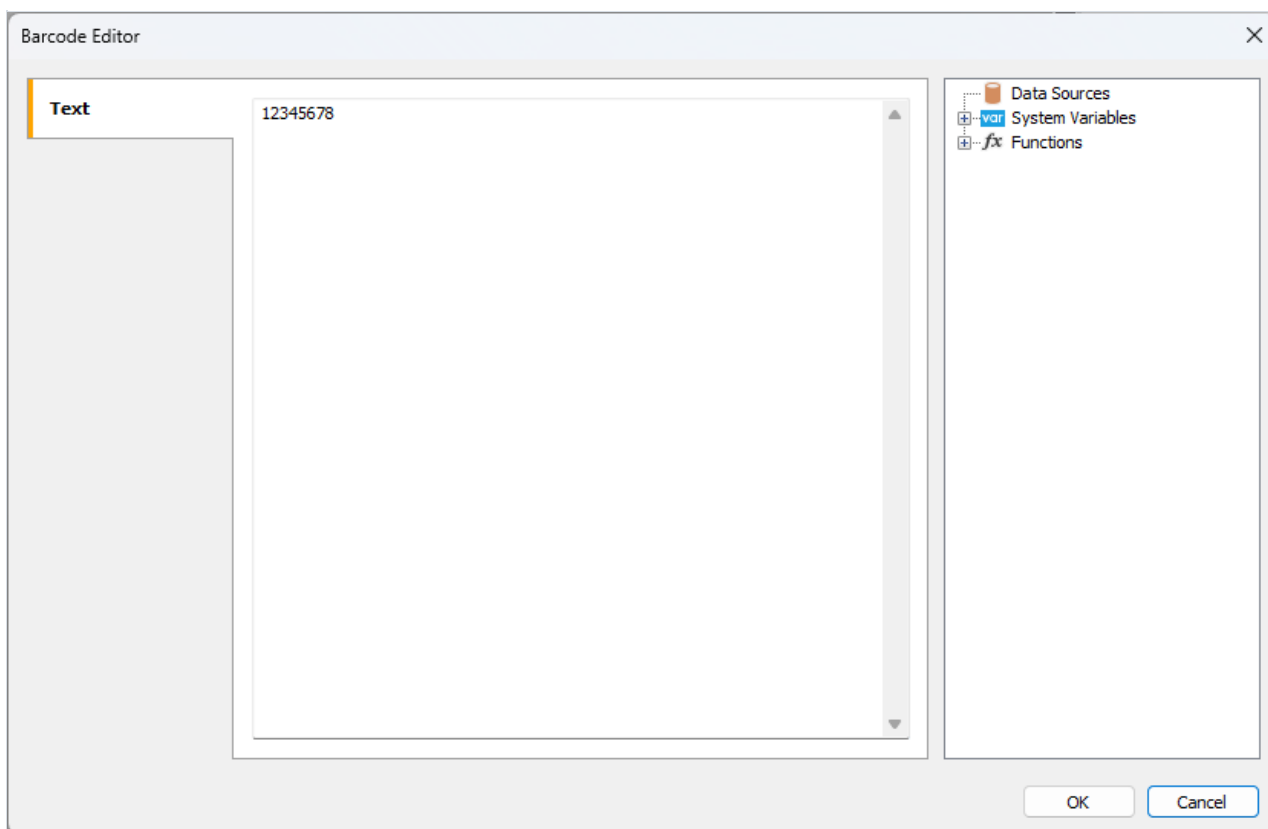
Checksum digits are not displayed in the text below the code. The checksum consists of two digits, which are calculated in different ways: modulo "C" and modulo "K". The calculation formula is the remainder modulo 47 of the sum of the weighted data values. The difference between "C" and "K" lies in the methods of weighting the values.

To generate a Code 93 barcode in FastReport .NET, select the Barcode object  at the Components Panel in the Report Designer. In the drop-down list, navigate to the "Others" category, and then choose Code93:



After selecting the barcode, place it on the Report Page.

Double-click on the added barcode to open the editor. You can also open the barcode editor by clicking the button  in the context menu of the added object, accessed by right-clicking:



If you want to hide the text under the barcode, locate the `ShowText` property in the property inspector of the corresponding barcode and set its value to `False` :



# Code 93 Extended



Code 93 Extended is an extended version of the Code 93 barcode. Unlike the basic version, it is capable of encoding 128 ASCII characters instead of 48.

Set of characters:


- digits 0-9;
- capital letters A-Z;
- special characters: \$, %, /, +.

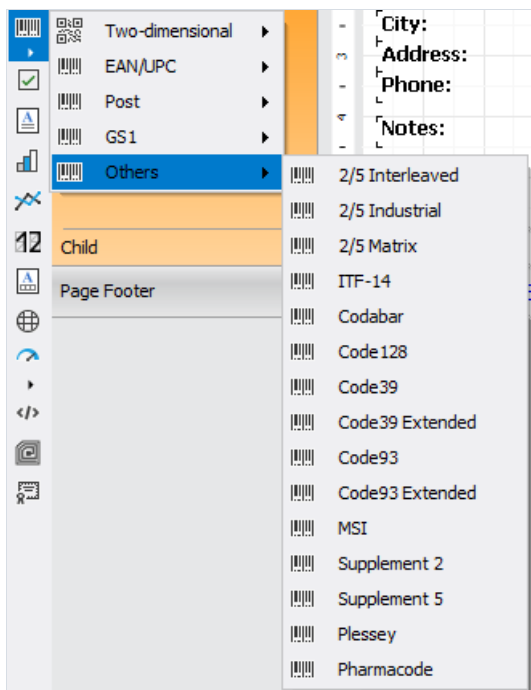
Code structure:

- start character \*;
- the encoded information – a data character consists of three lines and three spaces. The width of a line or space can be from 1 to 4 modules;
- a check digit calculated using the "C" modulo;
- a check digit calculated using the "K" modulo;
- stop character \*.


The check digit numbers are not displayed in the text under the code. The checksum consists of two digits, which are calculated in different ways: modulo "C" and modulo "K". The calculation formula is the remainder modulo 47 of the sum of the data value weights. The difference between "C" and "K" is the methods of weighting the values.

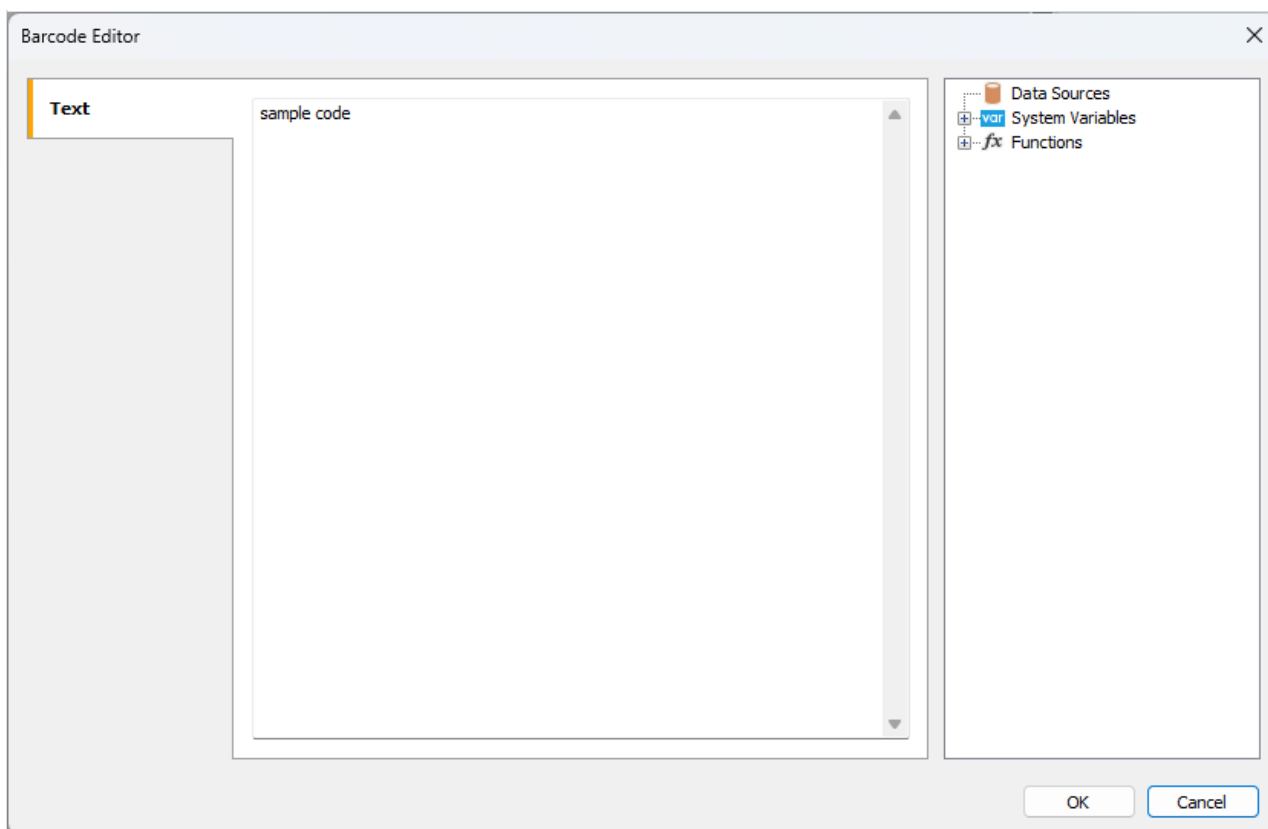
As mentioned earlier, Code 93 Extended allows the encoding of 128 characters. However, additional characters ( \$, /, %, + ) must be used to expand the standard Code 93 character set. Thus, to encode lowercase letters, a combination of the + character and the corresponding uppercase letter is required. The remaining modifier characters are needed to expand the set of special characters.

To generate a Code 93 Extended barcode in FastReport .NET, select the Barcode object  at the Components Panel in the Report Designer. In the drop-down list, navigate to the "Others" category, and then choose Code93 Extended:



After selecting the barcode, place it on the Report Page.

Double-click on the added barcode to open the editor. You can also open the barcode editor by clicking the button  in the context menu of the added object, accessed by right-clicking:



If you want to hide the text under the barcode, locate the `ShowText` property in the property inspector of the corresponding barcode and set its value to `False` :



# MSI



MSI linear barcode was created in 1971 on the basis of Plessey barcode and is essentially its improved version. It can only encode numbers from 0 to 9. Code structure:


- start character indicating the beginning of data reading (110);
- data;
- optional check character;
- stop symbol indicating the end of data reading (1001).

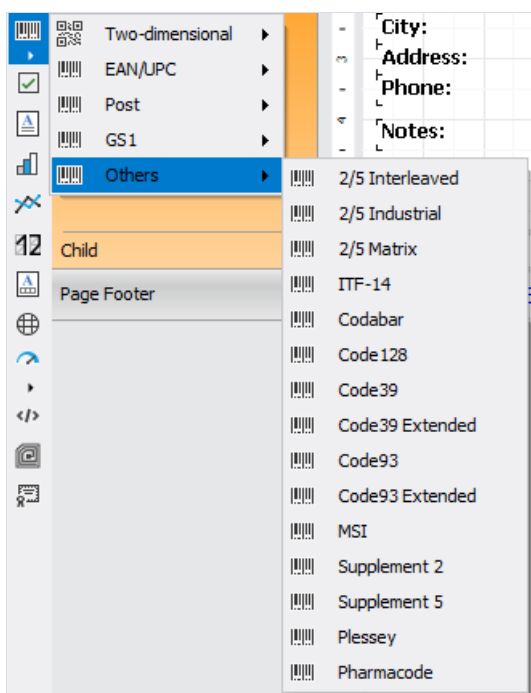
Each character in the code is represented in binary form with bars and spaces. Bar is 1 and space is 0.

The control character can be calculated by one of four types: Modulo 10, Modulo 11, Modulo 1010, Modulo 1110. The most common type is Modulo 10. By a certain algorithm is calculated the control character. The data read by the scanner is added up by the algorithm and the result is compared with the control character. If the result is positive, the code is considered to have been read correctly.


MSI – code of any length. It is limited only by the capabilities of the scanner.

Now this barcode is considered obsolete and practically is not used. Previously, it was used to mark goods in warehouses and supermarkets.

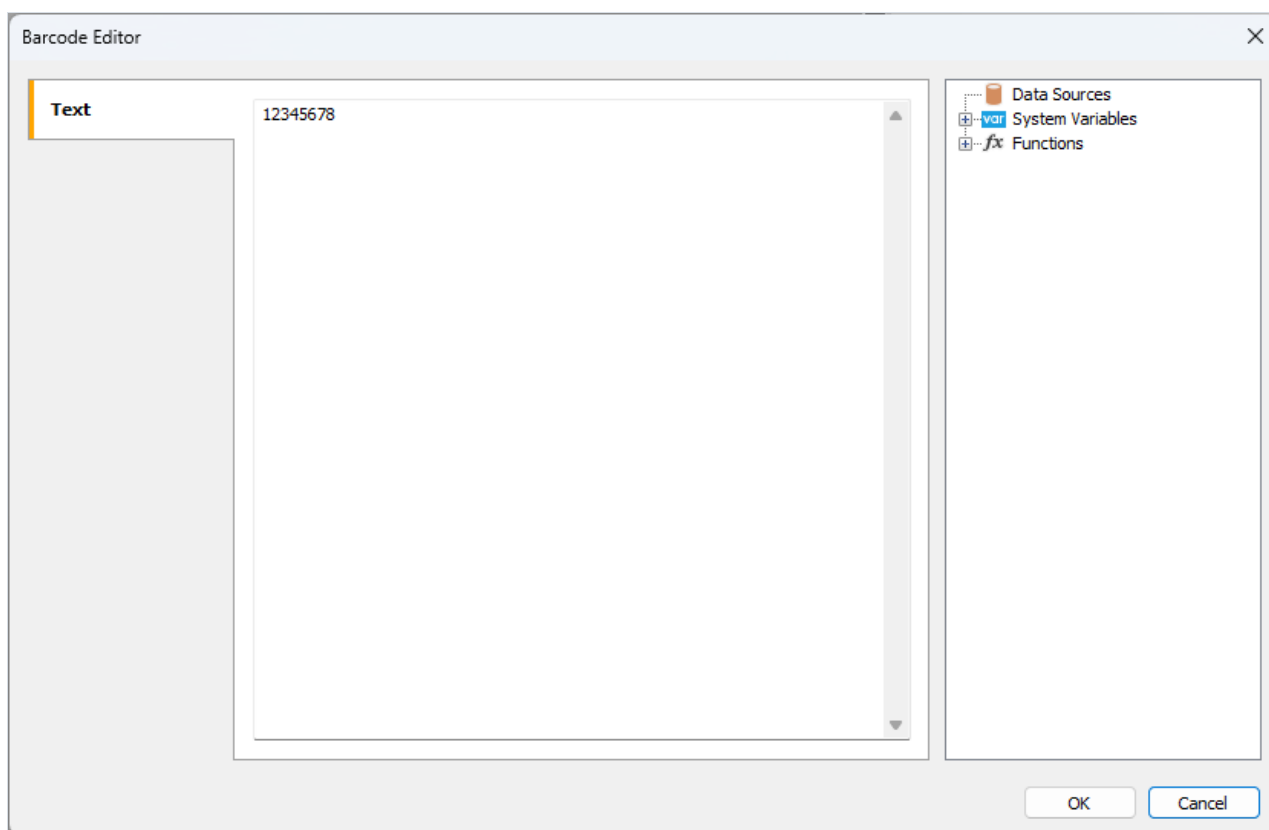
To generate a MSI barcode in FastReport .NET, select the Barcode object  at the Components Panel in the Report Designer. In the drop-down list, navigate to the "Others" category, and then choose MSI:



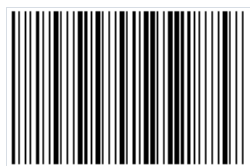
After selecting the barcode, place it on the Report Page.

Double-click on the added barcode to open the editor. You can also open the barcode editor by clicking the button  in the context menu of the added object, accessed by right-clicking:





If you want to hide the text under the barcode, locate the `ShowText` property in the property inspector of the corresponding barcode and set its value to `False` :



# Supplement

Common one-dimensional barcodes, such as EAN in Europe and UPC in the US, can be extended with an additional code placed to the right of the main code.

UPC-A, UPC-E, EAN-13, and EAN-8 can include an additional barcode to the right of the main barcode. This second barcode, which is usually not as tall as the main barcode, is used to encode additional information for newspapers, books, and other periodicals. The supplemental barcode can encode 2 or 5 digits of information.

The two-digit supplemental code is typically used to encode issue number information in periodicals (magazines, newspapers).



The supplemental code allows to omit issue number from the main barcode, keeping it the same for all issues. Thus, there is no need to scan the large code when only issue information is needed.

Five-digit supplemental code is often used on books and contains information about the suggested retail price of the item. Of the five characters, the first one denotes the currency code, and the remaining 4 – the price.



There are specific codes to indicate no-price conditions:

- code 90000 – retail price for the book is undetermined;
- code 99991 – the book is distributed free of charge.

However, the supplemental code can also store other information for internal use within the publishing house.

Each digit is encoded with 7 modules (bars and spaces).

Code structure:

- start character (1011 if we draw an analogy between bars and digits);
- first data character;
- separator (01);
- second data character or 4 characters in case of a 5-character code.


There is no explicit check character and stop character. After the separator, an admissible number of characters is read.

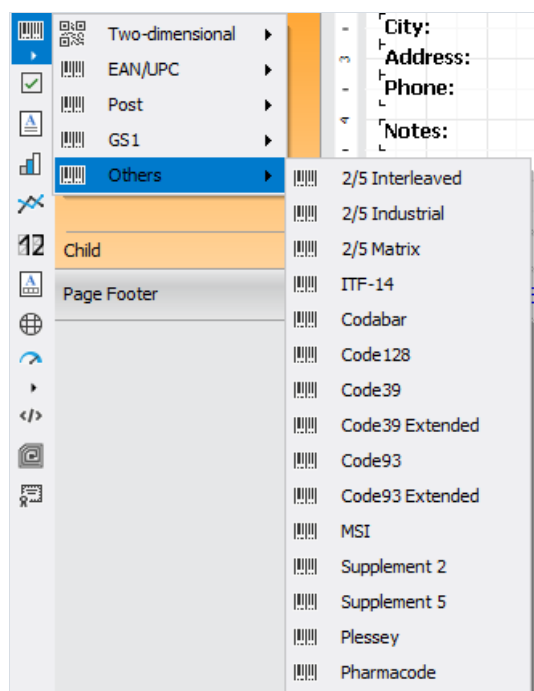
Data is encoded according to the "left even" and "left odd" sets used in EAN. Depending on the size of the supplemental barcode, different character parity patterns are used. The parity of the character determines the checksum.

If the supplemental barcode is two-digit, then the two-digit number which is formed by adding the first and second numbers must be divided by 4. Next, if the remainder from the division is an even number, then the first character is encoded with even parity, and the second with odd parity. This means that the even set of values will be used for encoding the first digit, and for the second – the odd set.

Now, when scanning, the parity of the read value will be determined. If it does not match the calculated parity, which is expected – it means that the barcode has been read incorrectly.


In the case of a five-digit code, the checksum calculation is more complicated. It is assumed that the last digit of the code is in an odd position. Starting from the last digit and up to the first in turn, even and odd positions are assigned. Then, the sum of all odd digits is taken and multiplied by 3. The sum of all even digits is multiplied by 9. Then, the unit of measurement is taken from the sum of the two previous calculations, that is, the far-right digit. This check digit is used to determine the parity pattern in a special table.

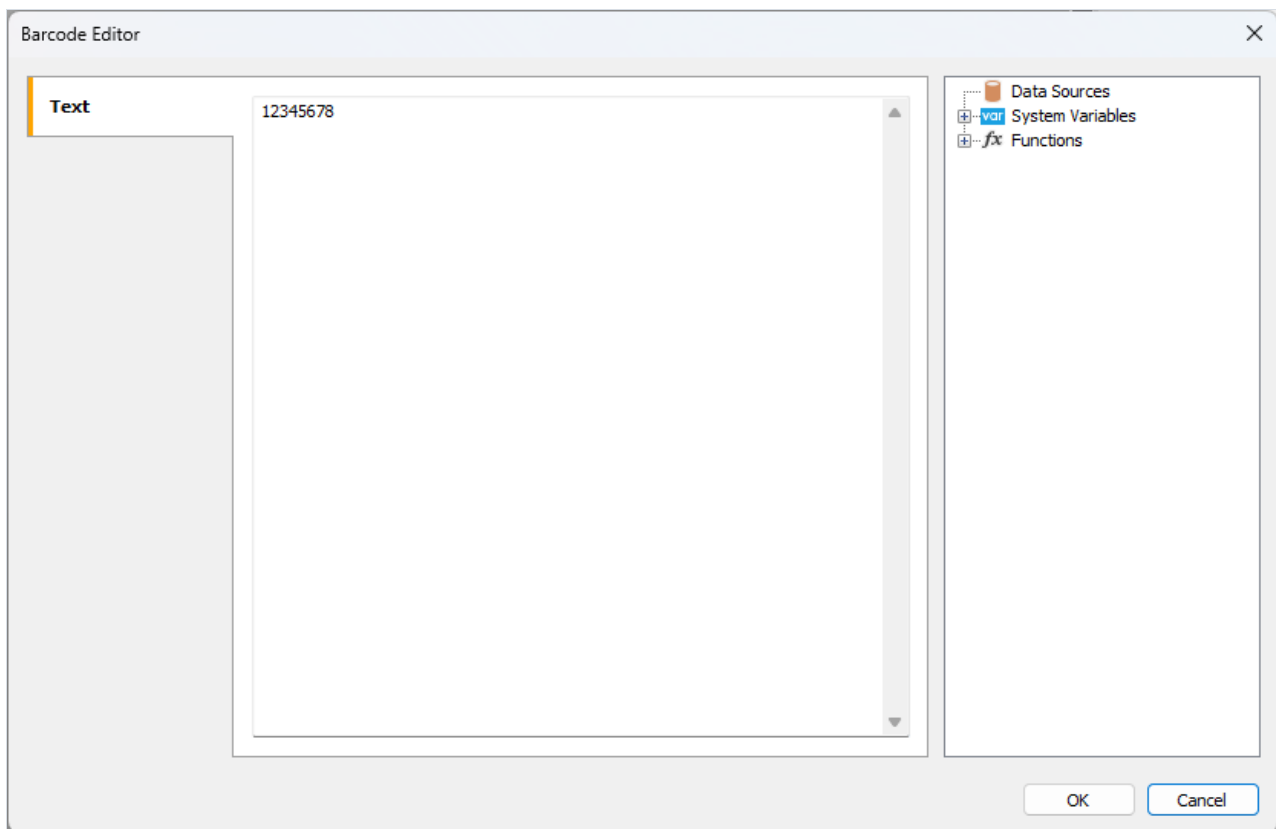
To generate a Supplement 2 (for a two-digit code) or Supplement 5 (for a five-digit code) barcode in FastReport .NET, select the Barcode object  at the Components Panel in the Report Designer. In the drop-down list, navigate to the "Others" category, and then choose Supplement 2 or Supplement 5:



After selecting the barcode, place it on the Report Page. Position the supplemental code next to the main code:



Double-click on the added barcode to open the editor. You can also open the barcode editor by clicking the button  in the context menu of the added object, accessed by right-clicking:



Like all barcodes in FastReport .NET Supplement, it has several properties that you can edit in the object property inspector:

| Property                      | Description  |
|-------------------------------|--|
| <b>Angle</b>                  | Allows you to set the rotation of the object to one of the fixed angles – 0, 90, 180, or 270 degrees.  |
| <b>Zoom</b>                   | Sets the scaling of the barcode. This property is used only with the <b>AutoSize</b> property.   |
| <b>AutoSize</b>               | If this property is enabled, the object will be stretched to show the entire barcode. If the property is disabled, the barcode will be stretched to the size of the object.                        |
| <b>ShowText</b>               | Determines whether to show the text at the bottom of the barcode.  |
| <b>DataColumn</b>             | The data field from which to load the text of the object.  |
| <b>Expression</b>             | An expression that returns the text of the object.   |
| <b>Text</b>                   | The text of the object.  |
| <b>Padding</b>                | Allows you to set the padding from the edges of the object in pixels.  |
| <b>WideBarRatio</b>           | This property is available for all linear barcodes. It defines the relative size of the barcode's wide bars.   |
| <b>CalcChecksum</b>           | This property is available for many linear barcodes. It determines whether to calculate the checksum automatically. If this property is disabled, the checksum must be present in the object text. |
| <b>DrawVerticalBearerBars</b> | If this property is enabled, the side lines will be displayed for the object.  |

# Plessey

The Plessey barcode was developed in 1971 by the Plessey Company. This is a classic one-dimensional linear barcode, which was mainly used for labeling goods on store shelves and in warehouse control. The main advantage of this code at the time of its creation was the ease of printing on a dot matrix printer. It is currently considered obsolete and hardly ever found.

Plessey allows you to encode hexadecimal digits (0-F). Each digit is represented by four bits – bars. "Zero" is a thin bar, "one" is a thick one. In addition to numbers, letters A-F can also be encoded. The barcode has a start code, encoded data, a checksum code, a finish mark and a reverse reading code (the code can be read in any direction).




MSI has become one of the variations of Plessey. Unlike regular Plessey, it allows you to encode only numbers, and there is no code for reverse reading. MSI supports several kinds of checksum code, for example: Mod-10, Mod-11, Mod-1010, Mod-1110.

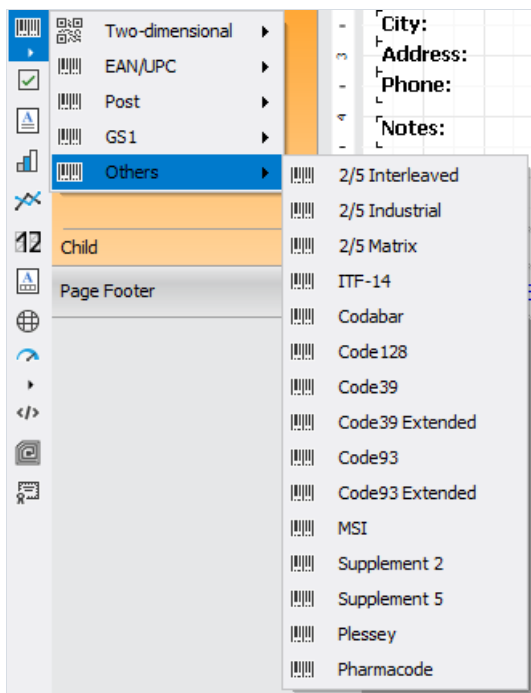


Both MSI and regular Plessey do not limit the length of the code, but a too long code may simply not fit the package and the reading scanner is not designed for long length, too.


This is what such a long code would look like:



To generate a Plessey barcode in FastReport .NET, select the Barcode object  at the Components Panel in the Report Designer. In the drop-down list, navigate to the "Others" category, and then choose Plessey:



After selecting the barcode, place it on the Report Page.

Double-click on the added barcode to open the editor. You can also open the barcode editor by clicking the button  in the context menu of the added object, accessed by right-clicking.

# Pharmacode

Pharmacode is a binary code that was developed by German company LAETUS GMBH specifically for pharmaceutical packaging. It is a subset of CODE39. This code is widely used in the pharmaceutical industry as part of the product packaging control system.

As part of an automated packaging system, Pharmacode allows easy scanning and recording of pharmaceutical shipments using universal identifiers. Also, with the help of scanners, it is easy to determine if a batch of drugs has been mixed with another batch.

The Pharmacode barcode ensures that the code is read, despite possible printing errors. Also, to ensure that the rest of the package, except the code, is printed correctly, Pharmacode can be printed in different colors (code and background) as opposed to barcodes intended for reading by laser or laser emulation. This is possible because Pharmacode is scanned with special white LAETUS scanners. This makes Pharmacode a very practical format for printing on packaging or documents that do not contain black ink.

As mentioned above, Pharmacode can be printed in different colors. Both the code and the background color can be different from white and black. There are special specifications for the combination of code and background colors used depending on the scanner type for reading. For example, standard black and white scanners only accept contrasting code and background colors, while special scanners that recognize color have no strict limitations.


Unlike other 1D barcodes, Pharmacode stores data in a binary system, not in a decimal one. In addition, Pharmacode can only represent single integers from 3 to 131070. The minimum number of lines is 2 for number 3, and the maximum value is 16 for 131070. The uniqueness of Pharmacode also lies in its right-to-left reading orientation, unlike other linear barcodes that have start and stop symbols. Reading the code from left to right would result in a completely different sequence of numbers.

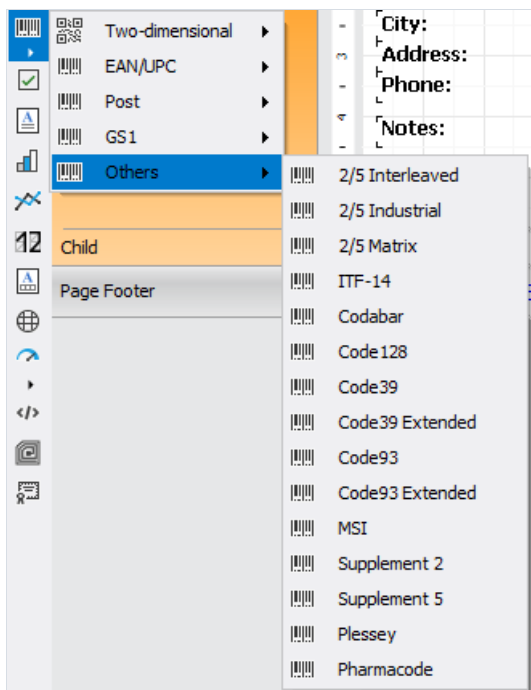
The Pharmacode standard is regulated by the LAETUS developer and is described in the PharmaCode Guide document.

Here is an example of the Pharmacode:




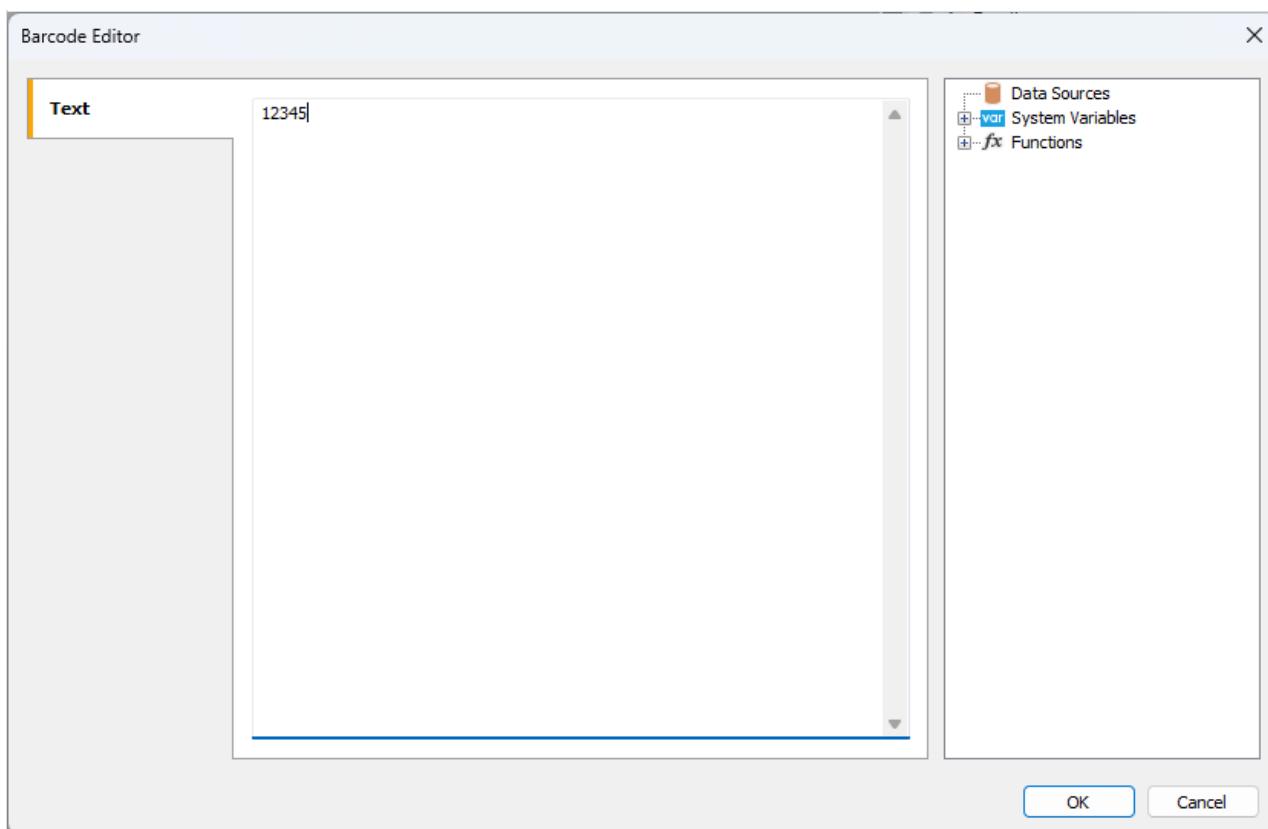
FastReport .NET allows you to create such codes in your reports. You can design the package in the generator immediately with the barcode.

To generate a Pharmacode barcode in FastReport .NET, select the Barcode object  at the Components Panel in the Report Designer. In the drop-down list, navigate to the "Others" category, and then choose Pharmacode:



After selecting the barcode, place it on the Report Page.

Double-click on the added barcode to open the editor. You can also open the barcode editor by clicking the button  in the context menu of the added object, accessed by right-clicking:

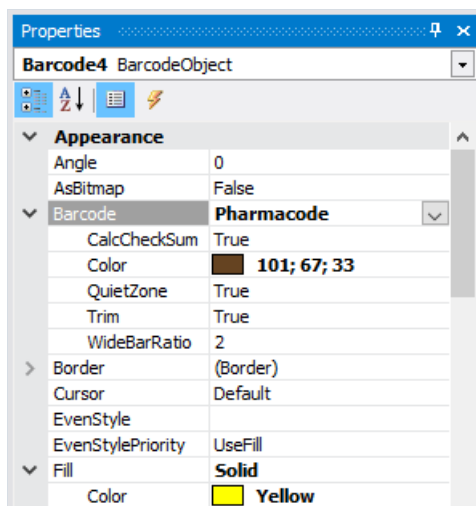


A numeric sequence may be entered as the code value, a function, a report variable and a database value may be specified.

In barcode properties, you can change the line spacing ( `WideBarRatio` ), code height ( `Height` ), and number display under the code ( `ShowText` ).

By default, the barcode is black on a white background. You can change the code color in `Barcode` -> `Color` property. And the background color is in the `Fill` -> `Color` property:



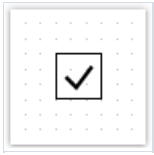


As a result of color adjustment, Pharmacode may look like this:



# The "CheckBox" object

The object displays the checkbox in the report. It looks as follows:



The object can display two states: "Checked" and "Unchecked". Use the following ways to handle the object's state:

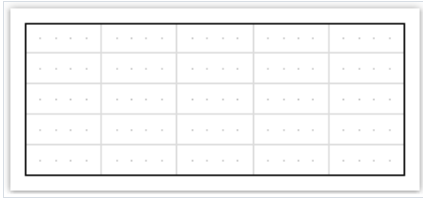
- set the state in the `Checked` property;
- bind the object to a data column using the `DataColumn` property;
- set the expression that returns the `true` or `false` in the `Expression` property.

The "CheckBox" object has the following properties:

| Property                              | Description   |
|---------------------------------------|---|
| <b>CheckedSymbol, UncheckedSymbol</b> | These properties determine the symbol that is shown in the object, depending on the object's state.   |
| <b>CheckColor</b>                     | This property determines the color of the check symbol.   |
| <b>CheckWidthRatio</b>                | Use this property to set the check width ratio. The width of the check symbol depends on the size of the object. You can use values in the 0.2 - 2 range. |
| <b>HideIfUnchecked</b>                | This property allows to hide the object if it is unchecked.   |
| <b>Checked</b>                        | This property controls the state of the object.   |
| <b>DataColumn</b>                     | The data column which this object is bound to. The type of column should be either bool or int.   |
| <b>Expression</b>                     | The expression that returns the <code>true</code> or <code>false</code> .   |

# The "Table" object

The "Table" object is made up of rows, columns and cells. It is a simplified analog of Microsoft Excel table. It looks like this:



You can learn more about this object in the ["Creating reports"](#) chapter.

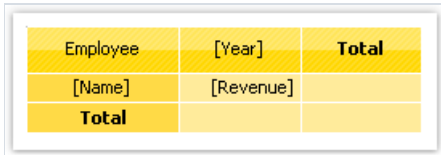
The "Table" object has the following properties:

| Property             | Description  |
|----------------------|--|
| <b>ColumnCount</b>   | Use this property to quickly set the number of columns. If columns in a table are few, they get added, and when they are more, they get deleted.                                       |
| <b>RowCount</b>      | Use this property to quickly set the number of rows. If rows in a table are few, they get added, and when they are more, they get deleted.   |
| <b>FixedColumns</b>  | The property determines how many columns in the table are fixed. Fixed columns form the table header. Printing of the header is controlled by the <code>RepeatHeaders</code> property. |
| <b>FixedRows</b>     | The property determines how many rows in the table are fixed. Fixed rows form the table header. Printing of the header is controlled by the <code>RepeatHeaders</code> property.       |
| <b>RepeatHeaders</b> | The property allows printing the table header on every new page. This property works only for tables which are formed dynamically.   |

# The "Matrix" object

The "Matrix" object is, like the "Table" object, made up of rows, columns and cells. At the same time, it is not known beforehand how many rows and columns will be in the matrix - this depends on the data to which it is connected.

The object looks like this:



|          |           |       |
|----------|-----------|-------|
| Employee | [Year]    | Total |
| [Name]   | [Revenue] |       |
| Total    |           |       |

You can learn more about this object in the ["Creating reports"](#) chapter.

The "Matrix" object has the following properties:

| Property               | Description   |
|------------------------|---|
| <b>RepeatHeaders</b>   | If matrix is divided on several pages, this property allows printing matrix header on each new page.  |
| <b>CellsSideBySide</b> | This property determines how matrix cells will be located if the matrix has several data cell levels.<br>Possible variants: <ul style="list-style-type: none"><li>- the cells are displayed side by side;</li><li>- the cells are displayed under each other.</li></ul> |
| <b>Style</b>           | Using this property you can set a style for the whole matrix. You can choose one from predefined styles.  |
| <b>AutoSize</b>        | This property allows to calculate the matrix size automatically. Disable it if you want to control the object size manually.  |
| <b>DataSource</b>      | The property allows connecting the matrix to the data source. This property is set up automatically when you drag data column to the matrix. However, if you use expressions in cells, check that this property was set up correctly.                                   |
| <b>Filter</b>          | This property contains expression for data filtering which will be applicable to data source of the matrix (see <a href="#">DataSource</a> property).   |

# Advanced Matrix Object

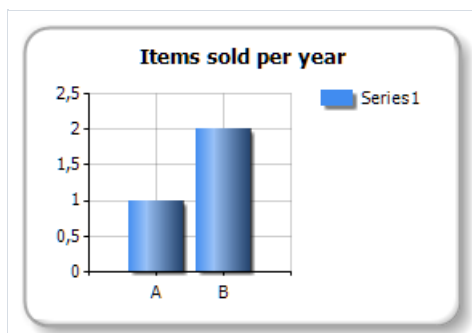
This object, like the "Matrix" object, allows you to build summary reports.

|                            | Top 5 customers     |                     | Total                 |
|----------------------------|---------------------|---------------------|-----------------------|
|                            | Total               | Others (84)         |                       |
| ► Beverages (12)           | \$144 007,88        | \$192 684,30        | <b>\$336 692,18</b>   |
| ► Condiments (12)          | \$123 407,20        | \$105 665,89        | <b>\$229 073,09</b>   |
| ► Confections (13)         | \$54 823,13         | \$112 615,09        | <b>\$167 438,23</b>   |
| ► Dairy Products (10)      | \$103 074,28        | \$166 433,01        | <b>\$269 507,29</b>   |
| ► Grains/Cereals (7)       | \$31 368,01         | \$64 376,58         | <b>\$95 744,59</b>    |
| ▼ Meat/Poultry (6)         | \$269 495,38        | \$113 526,98        | <b>\$383 022,36</b>   |
| 1. Alice Mutton            | \$13 428,48         | \$19 269,90         | <b>\$32 698,38</b>    |
| 2. Mishi Kobe Niku         | \$1 319,20          | \$5 907,30          | <b>\$7 226,50</b>     |
| 3. Pâté chinois            | \$7 137,60          | \$10 288,80         | <b>\$17 426,40</b>    |
| 4. Perth Pasties           | \$6 916,56          | \$13 657,61         | <b>\$20 574,17</b>    |
| 5. Thüringer Rostbratwurst | \$239 795,40        | \$60 573,28         | <b>\$300 368,67</b>   |
| 6. Tourtière               | \$898,14            | \$3 830,10          | <b>\$4 728,24</b>     |
| ► Produce (5)              | \$67 154,22         | \$71 955,36         | <b>\$139 109,58</b>   |
| ► Seafood (12)             | \$31 732,55         | \$99 591,19         | <b>\$131 323,74</b>   |
| <b>Total</b>               | <b>\$825 062,63</b> | <b>\$926 848,41</b> | <b>\$1 751 911,04</b> |

You can read more about this object in the chapter ["Advanced Matrix" object](#).

# The "Chart" object

The "MS Chart" object allows to display charts. There are more than 30 different series types available - bars, columns, areas, lines, bubbles, pie, circular, financial, pyramidal, ranges. The object looks like this:



You can learn more about this object in the ["Report creation"](#) chapter.

The "Chart" object has the following properties:

| Property  | Description  |
|---|--|
| <b>Chart</b>  | Reference to Microsoft Chart object.   |
| <b>AlignXValues</b>   | This property allows to align X values in different chart series (by inserting empty values). It is used if the chart contains two series or more.   |
| <b>AutoSeriesColumn,</b><br><b>AutoSeriesColor,</b><br><b>AutoSeriesSortOrder</b> | These properties allows to set up automatically created series. Read more about this in the <a href="#">"Report creation"</a> chapter.               |
| <b>DataSource</b>   | The property allows connecting the chart to the data source.   |
| <b>Filter</b>   | This property contains expression for data filtering which will be applicable to data source of the chart (see <a href="#">DataSource</a> property). |

# The "Zip Code" object

The "Zip Code" object allows to print a zip code on envelopes. It may display numeric characters (0-9).

The object is as follows:



You can connect an object to data by using one of the following methods:

- set the zipcode data in the `Text` property;
- bind the object to a data column using the `DataColumn` property;
- set the expression that returns the zipcode data in the `Expression` property.

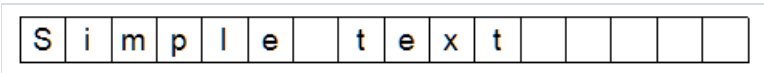
The "Zip Code" object has the following properties:

| Property                           | Description  |
|------------------------------------|--|
| <b>SegmentCount</b>                | The number of segments in a code. This property is set to 6 by default.                              |
| <b>SegmentWidth, SegmentHeight</b> | The size of a single code segment. The default size is 0.5x1cm.                                      |
| <b>Spacing</b>                     | This property determines a distance between two segment's origins. The default value is 0.9cm.       |
| <b>ShowGrid</b>                    | Determines whether it is necessary to display a grid.  |
| <b>ShowMarkers</b>                 | Determines whether it is necessary to display the markers (bold horizontal lines above the zipcode). |
| <b>DataColumn</b>                  | The data column which this object is bound to.   |
| <b>Expression</b>                  | The expression that returns the zipcode data.  |
| <b>Text</b>                        | The text containing a zipcode.   |

# The "Cellular Text" object

This object can display each character of a text in its individual cell. It is often used to print some forms in financial applications.

The object is as follows:



In fact, this object is directly inherited from the "Text" object. You may connect it to data in the same manner. For example, you may invoke the object's editor and type the following text:

[Employees.FirstName]

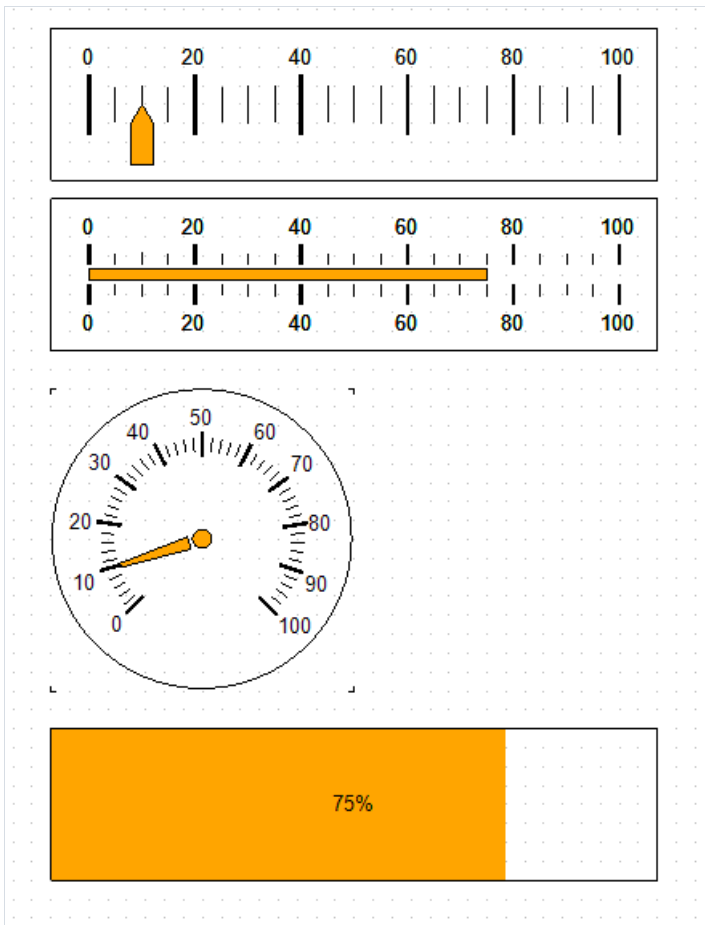
The "Cellular Text" object has the following properties:

| Property                    | Description  |
|-----------------------------|--|
| CellWidth,<br>CellHeight    | These properties determine the size of a single cell. If both properties are 0 (by default), the cell size will be calculated automatically, depending on the font used. |
| HorzSpacing,<br>VertSpacing | These properties determine the horizontal and vertical gap between adjacent cells.   |




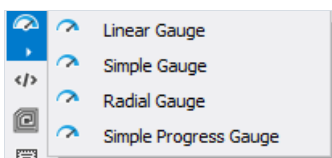
# Gauge objects


These objects are intended to visually display any value. Here's what the four different gauge types currently supported look like:

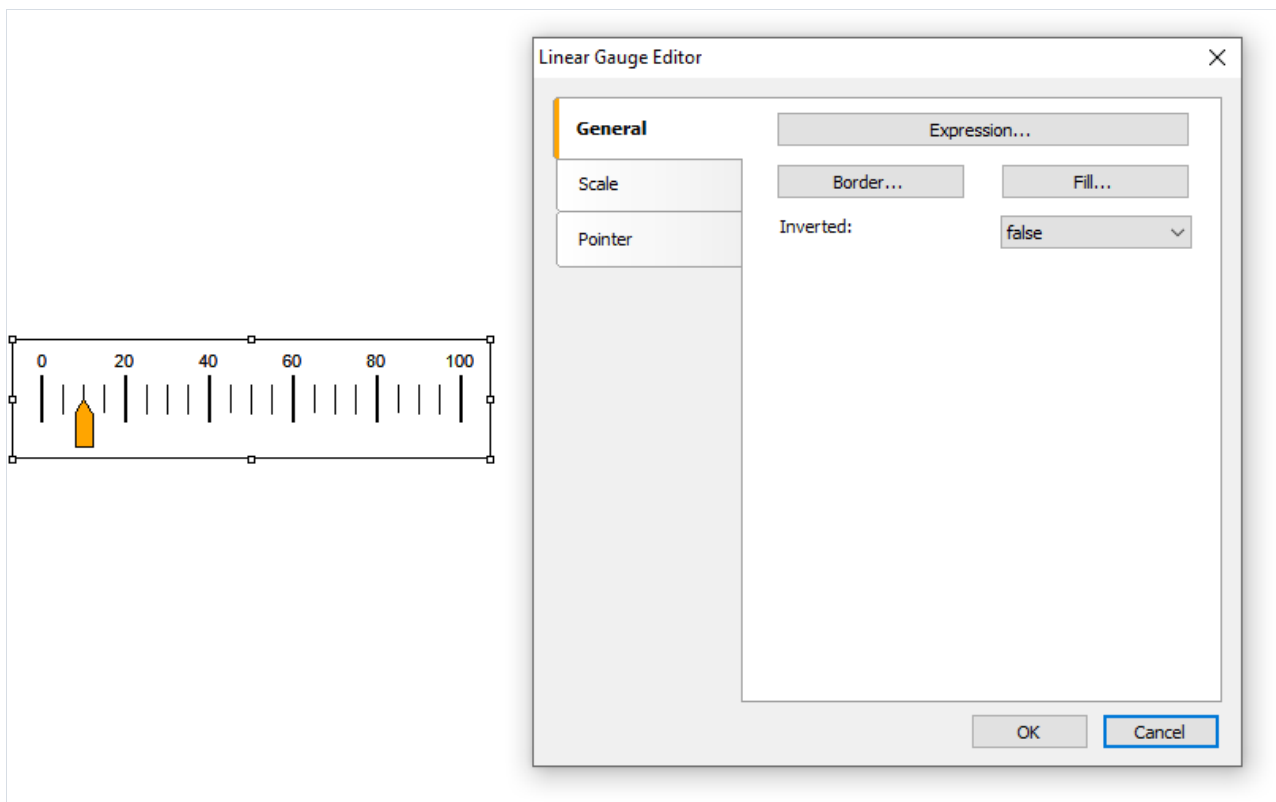


- Linear Gauge;
- Simple Gauge;
- Radial Gauge;
- Simple Progress Gauge.

To add the Gauge object to your report, select one of the options from the submenu that opens when you click  button:



You can change some aspects of the gauge appearance, such as the color of the pointer. To edit the gauge parameters, double-click on it with the left mouse button, or from the context menu using the button .



The Expression button will open a text editor where you can enter the gauge value yourself, or compose an expression for it and connect the gauge to the data.

# The "Digital Signature" object

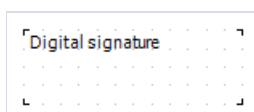
The digital signature - cipher that guarantee uniqueness and originality, allowing to unequivocally establish authorship and protect against document changes. Thanks to reliable encryption algorithms, such signatures are no worse than handwritten, and even better, more reliable.

In the current version two types of signatures are available:

1. **Signing field** (signature field) - implies the presence of a special field in the document, by clicking on which, the user will be able to attach his certificate.

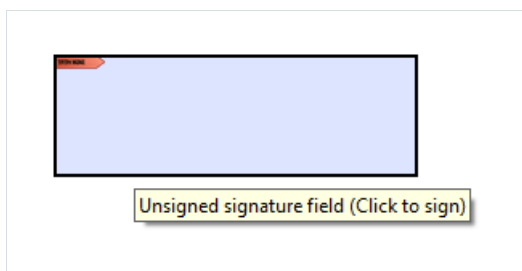
An object that implements this feature is added by pressing this key: 

It is called Digital Signature. When placing this control on the report page, it looks like this:



In the report view it is invisible. Its functionality is limited solely to PDF export. That is, you will see this field when viewing a PDF file in Acrobat Reader.

After export, the field will look like this:



Click on the signature field and see the window for choosing a certificate to sign the document:



2. **Invisible signature** – then sign with a certificate. It is not visible visually, but in document properties one can get information about the signatory, authenticity of the signature, version of the document at the moment of signing and other information.

In order to sign an exported PDF file with an invisible signature, you do not need to add the "Digital Signature" object to the report page. You need to enable the signing option in the export settings, and select a certificate there as well:

Export to PDF

Export

Options

Vector Graphics

Information

Security

**Digital signature**

Viewer

Sign Information

☒ Sign document

Location

Reason

Contact info

Certificate Information

Certificate path

C:\Certificate.p12

Certificate password

\*\*\*\*\*

☐ Save password

☐ Open after export

OK

Cancel


# Your first report in FastReport

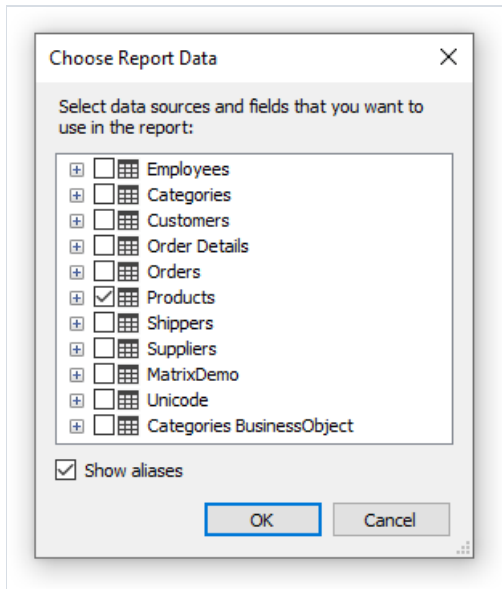
Let us create a simple report, which prints a list of products. We will use the Products table, which can be found in the demo data base as our data source.

Assuming that, you will be performing the actions written below in the demonstration program, Demo.exe, from which the report designer can be called.

# Example 1. Creating a report manually

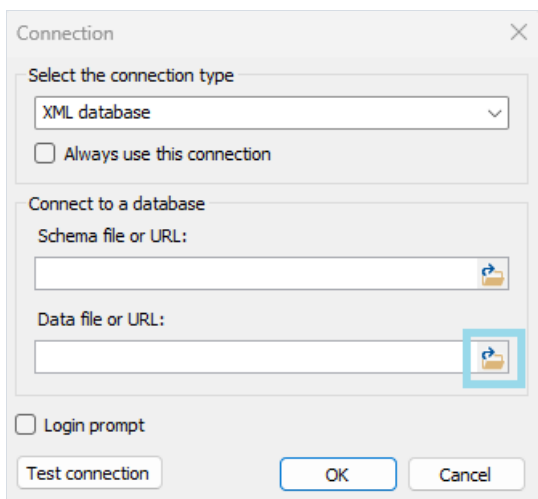
In this example, we will create a report manually. For this, we will do the following:

- press the  button on the toolbar, and in the "Add New Item" window, choose "Blank report";
- in the "Data" menu, choose the "Choose Report Data..." item and check the "Products" data source:

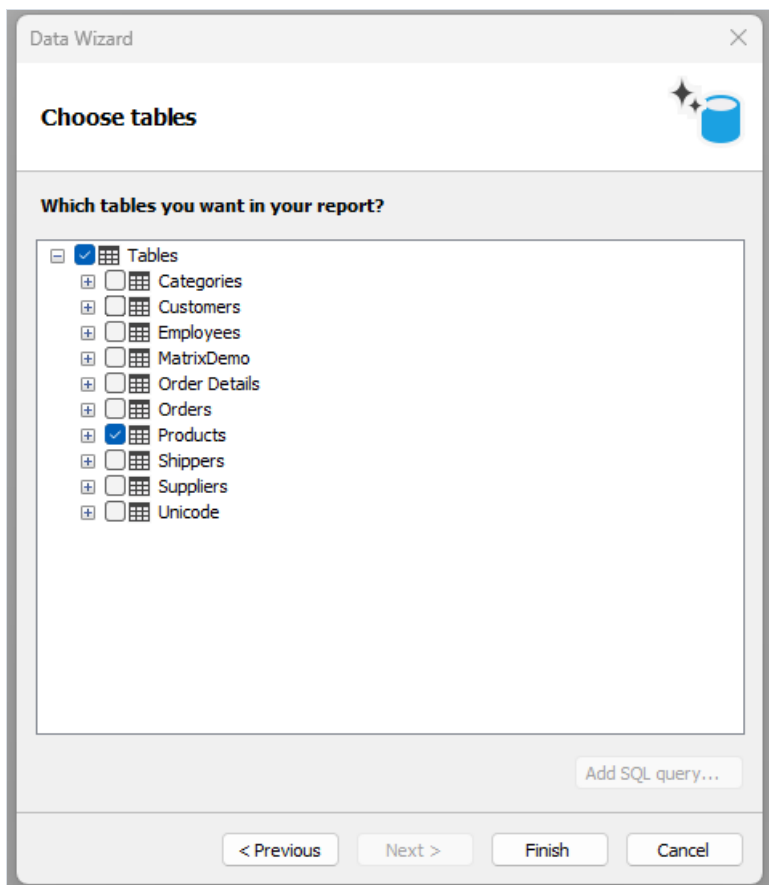


The second option is how you can add your data from an XML file:

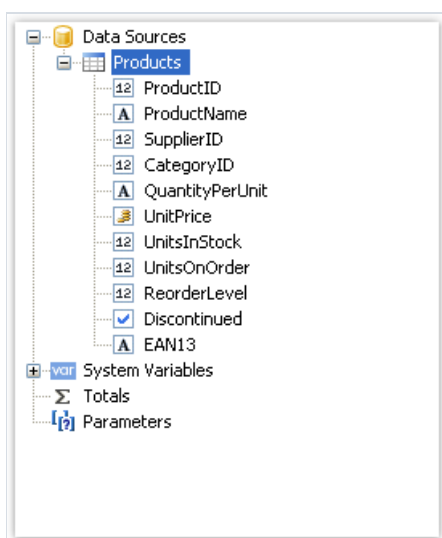
- in the "Data" menu, select "Add Data Source...". The Data Wizard form opens;
- click on the "New Connection..." button;
- select the connection type: XML database. Click on the icon of the data file or URL:



- in the Open File dialog box, navigate to the location of the file, for example C:\Program Files (x86)\Fast Reports\2024.2.12\FastReport .NET Trial\Demos\Reports, then select nwind.xml;
- to check the connection to the database, click on the "Test connection" button. If the connection test is completed successfully, click OK to close the Connection window;
- in the Data Wizard, click the Next button;
- expand the root node of the "Tables", then select the "Products" table and click Finish:



- switch to the "Data" service window (if it is not on the screen, it can be shown by choosing the "Data|Show Data Window" menu item). Expand the "Data Sources" item, then the "Products" item:

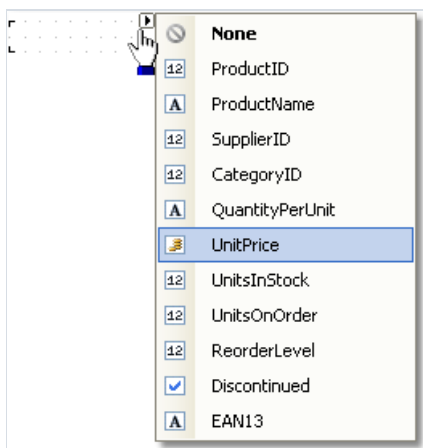


- drag the **ProductName** data column onto the "Data" band. FastReport creates a "Text" object, which is connected to this column and a header for it;
- we will create the **UnitPrice** data column by using another method. For this, press the "Text" button on the "Objects" toolbar:



- leave the mouse and drag its pointer onto the "Data" band - you will see that FastReport offers to insert an object. Choose the needed position and click the mouse to insert the object;
- place the mouse pointer on the object and click the small button in the right corner of the object. You will see


a list of data columns. Choose the **UnitPrice** item from the list:



- create the "Text" object - header for the **UnitPrice** column. Place in on the "Page Header" band. Double click the object and write the text "Unit Price";
- create the "Text" object - report title. Place it on the "Report Title" band and write the text "PRODUCTS";
- set "Bold" as font style for all objects that are placed on the "Page Header" and "Report Title" bands. For this, select objects by pressing Shift, and press the **B** button on the "Text" toolbar. After this, the report will be looking like this:

|                |                        |                      |
|----------------|------------------------|----------------------|
| Report Title   |                        | <b>PRODUCTS</b>      |
| Page Header    | <b>ProductName</b>     | <b>Unit Price</b>    |
| Data: Products | [Products.ProductName] | [Products.UnitPrice] |
| Page Footer    |                        |                      |

Ways to run the report:

- using the button  on the control panel;
- using the keyboard shortcut Ctrl+P.

The report will be built and shown in the preview window:

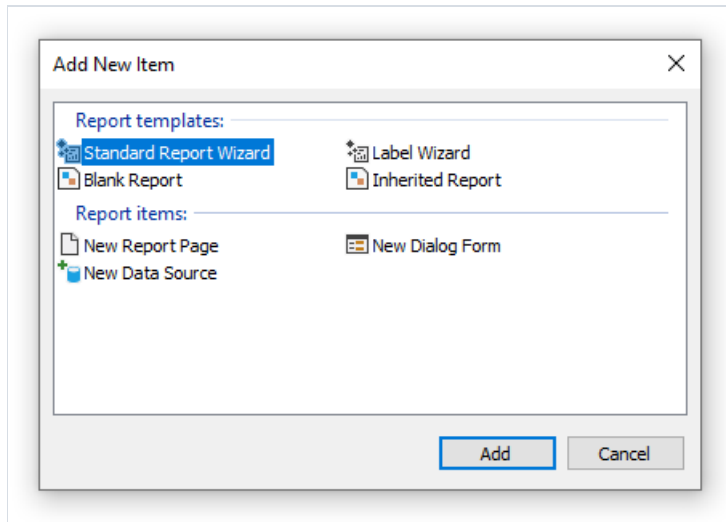
| <b>PRODUCTS</b>              |                   |
|------------------------------|-------------------|
| <b>ProductName</b>           | <b>Unit Price</b> |
| Chai                         | 18                |
| Chang                        | 19                |
| Aniseed Syrup                | 10                |
| Chef Anton's Cajun Seasoning | 22                |
| Chef Anton's Gumbo Mix       | 21,35             |
| Grandma's Boysenberry Spread | 25                |
| Uncle Bob's Organic Dried    | 30                |
| Northwoods Cranberry Sauce   | 40                |
| Mishi Kobe Niku              | 97                |
| Ikkura                       | 31                |



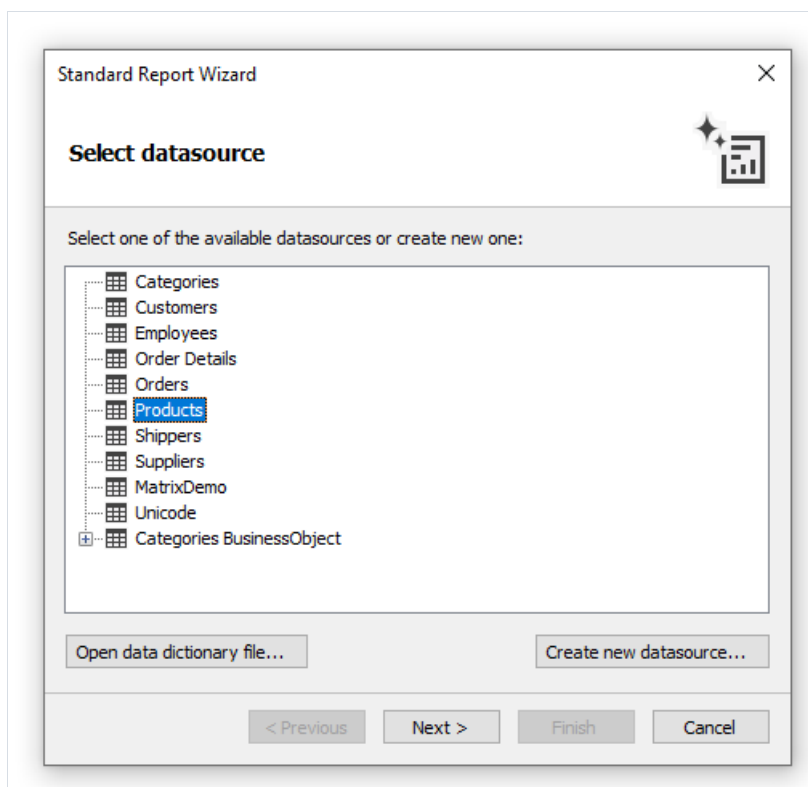
## Example 2. Creating a report with the wizard

In this example, we will create a report with the help of the "Standard Report Wizard". For this, do the following:

- press the  button on the toolbar and in the "Add New Item" window, choose "Standard Report Wizard":



- on the first step of the wizard, choose the "Products" table and click the "Next" button:



- on the second step of the wizard, choose the `ProductName` and `UnitPrice` data columns:

Standard Report Wizard

### Select data columns

Select data columns that you want to show in a report.

Available columns:

- ProductID
- SupplierID
- CategoryID
- QuantityPerUnit
- UnitsInStock
- UnitsOnOrder
- ReorderLevel
- ☒ Discontinued
- EAN13

Selected columns:

- ProductName
- UnitPrice

< Previous   **Next >**   Finish   Cancel

- the rest of the steps can be skipped, click the "Next" button;
- on the last step of the wizard, choose "Blue" style and click the "Finish" button:

Standard Report Wizard

### Select style

Select the report style.

Standard

Gray

Green

Sand

**Blue**

CUSTOMERS


| ID    | Company Name                       |
|-------|------------------------------------|
| A     |                                    |
| ALFIO | Alfreda Fullerton                  |
| ANATR | Ana Trujillo Emparedados y helados |
| ANTON | Antonio Moreno Taqueria            |
| AROUT | Around the Horn                    |
| B     |                                    |
| BERRG | Berglund's smörgåsar               |
| BIALS | Bialas Szwedzkie                   |
| BOLID | Bonadetti's pizza                  |
| BOLID | Bonadetti's pizza                  |
| BONAP | Bonaparte                          |
| BOTTU | Bottum's Market                    |
| BOTTU | Bottum's Market                    |
| C     |                                    |
| CACTU | Cactus Comidas para llevar         |
| CEMTC | Centro comercial Modocums          |
| CHOPS | Chop-Shop Chinese                  |
| COMUS | Comércio Uniao                     |

Page | Page

< Previous   Next >   Finish   Cancel

FastReport will create the following report:

|                |   |
|----------------|---|
| Report Title   | Products                                    |
| Page Header    | ProductName UnitPrice                       |
| Data: Products | [Products.ProductName] [Products.UnitPrice] |
| Page Footer    | [PageN]                                     |

To run a report, click the  button on a toolbar. The report will be built and shown in the preview window:

### Products

| ProductName                     | UnitPrice |
|---------------------------------|-----------|
| Chai                            | 18        |
| Chang                           | 19        |
| Aniseed Syrup                   | 10        |
| Chef Anton's Cajun Seasoning    | 22        |
| Chef Anton's Gumbo Mix          | 21,35     |
| Grandma's Boysenberry Spread    | 25        |
| Uncle Bob's Organic Dried Pears | 30        |
| Northwoods Cranberry Sauce      | 40        |
| Mishi Kobe Niku                 | 97        |
| Ikura                           | 31        |
| Que Pasa                        | 21        |

# Report creation

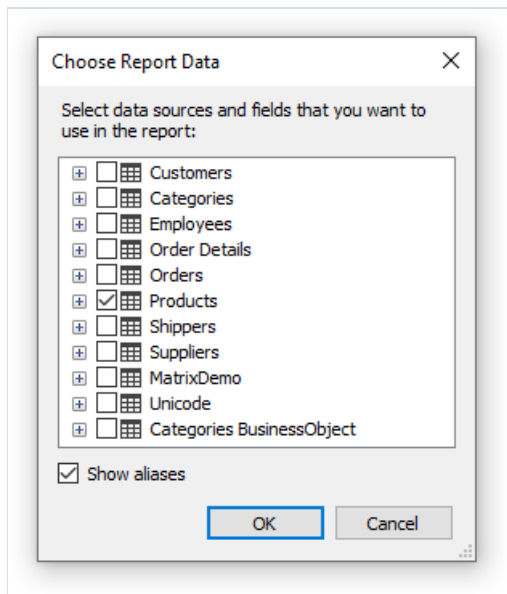
In this chapter we will look at the methods of creating common types of reports. In order to create any report, as a rule, you need to do the following:

1. Choose or create data, which will be used in the report.
2. Create the report structure, by adding the needed bands into the report.
3. Connect the band to a data source.
4. Place the "Text" objects on the bands to print data.
5. Setup the appearance, formatting.

# Choosing data for a report

Before you start building a report, you need to choose the data which will be printed in the report. You can do this in two ways:

- you can choose one of the data sources, which was registered in the report by a programming method. This can be done in the "Data|Choose Report Data..." menu, by marking the needed data source:



- you create a new data source in the "Data|New Data Source..." menu.

Read more about data sources in the ["Data sources"](#) chapter.

Just after you have chosen the data source, it appears in the "Data" window. Now you can use this source in the report. Many reports use only one data source. For reports of "master-detail" type, you need to choose two data sources, related to each other (you can read more about relations in the ["Data sources"](#) chapter). Several data sources can also be needed in a report, which prints data from related sources.

# Dynamic layout

It is necessary often to print a text whose size is not known when creating a report. For example, this can be a description of goods. In this case, the following tasks will need to be solved:

- calculate the height of the object, such that it encloses the whole text;
- calculate the height of the band, such that it encloses the object with a variable amount of texts;
- move or change the height of other objects, which are contained on the band, such that, they do not disturb the general design of the report.

These tasks can be solved by using some object and band properties:

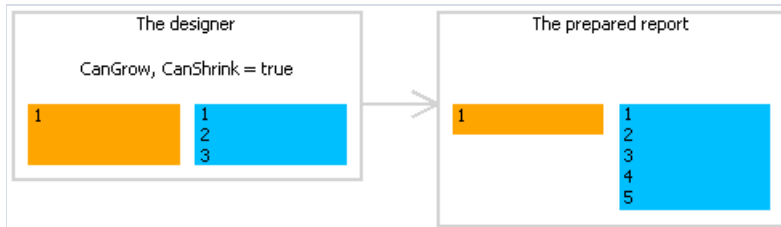
- `CanGrow` and `CanShrink` properties allow calculating the height of the object automatically;
- `ShiftMode` property allows moving objects that are located under the objects that expand;
- `GrowToBottom` property allows resizing an object to the bottom edge of the band;
- `Anchor` and `Dock` properties allow controlling the size of objects depending on the size of the band.

All these properties will be looked at below.

# CanGrow, CanShrink properties

Every band and report object has these properties. They determine whether an object can grow or shrink depending on the size of its contents. If both properties are disabled, the object always has the size specified in the designer.

These properties are very useful, if it is needed to print a text whose size is not known when designing. In order for an object to accommodate the entire text, it needs to have the `CanGrow` and `CanShrink` properties enabled:

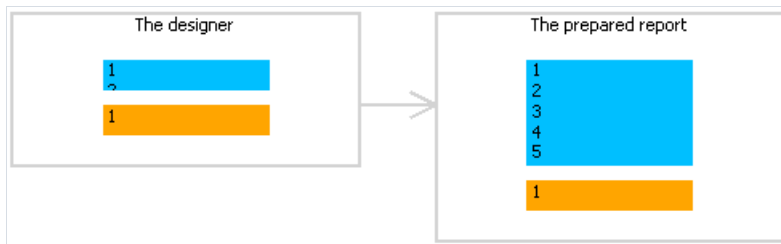


The following objects can affect the height of a band:

- "Text";
- "Rich Text";
- "Picture" (with `AutoSize` property enabled);
- "Table".

# ShiftMode property

Every report object has this property. This property is accessible only in the "Properties" window. An object, whose **ShiftMode** property is enabled, will be moving up and down, if the object above on can either grow or shrink.



The **ShiftMode** property can have one of the following values:

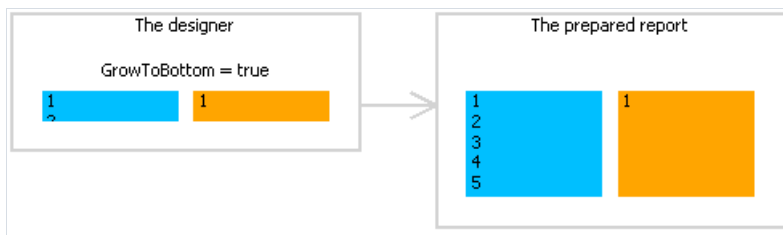
- Always (by default) - meaning that the object needs to shift always;
- Never - implies that the object does not need to shift;
- WhenOverlapped - implies that the object needs to shift in that case, if the expanding object is located exactly over it (that is, both objects overlap horizontally).

This property is convenient to use when printing information in a table form, when several cells of the table are located on top of each other and can have a variable amount of text.



# GrowToBottom property

Every report object has this property. When printing an object with this property, it stretches up to the bottom edge of a band:



This is needed when printing information in a table form. In a table row there can be several objects which can stretch. This property makes it possible to set all objects' height to the maximum height of the band.

# Anchor property

Every report object has this property. It determines how the object will be changing its position and/or its size when the container on which it is laying will be changing its size. By using **Anchor** , it can be done in such a way that, the object expands or moves synchronously with its container.

The container, being referred to, in many cases will be the band. But this is not a must - this can also be the "Table" or "Matrix" objects.

The **Anchor** property can have one of the following values, and also any combination of them:

| Value         | Description  |
|---------------|--|
| <b>Left</b>   | Anchors the left edge of the object. When the container's size will be changing, the object will not be moving left/right.   |
| <b>Top</b>    | Anchors the top edge of the object .When the container's height will be changing, the object will not be moving up/down.   |
| <b>Right</b>  | Anchors the right edge of the object .When the container's width will be changing , the distance between the right edge of the object and the container will be constant. If the left edge of the container is anchored as well, then the object will be growing and shrinking synchronously with container. |
| <b>Bottom</b> | Anchors the bottom edge of the object. When the container's height will be changing, the distance between the bottom edge of the object and the container will be constant. If the top edge of the object is anchored as well, the object will be growing and shrinking synchronously with container.        |

By default, the value of this property is **Left, Top** . This means that, when the container's size will be changing, the object will not be changing. In the table below, combinations of some frequent used values are given:

| Value                           | Description  |
|---------------------------------|--|
| <b>Left, Top</b>                | Value by default. The object does not change when the size of the container changes.   |
| <b>Left, Bottom</b>             | The object moves up/down when the height of the container changes. The position of the object in relation to the bottom edge of the container does not change. |
| <b>Left, Top, Bottom</b>        | When the height of the container is changing, the height of the object synchronously changes with it.  |
| <b>Left, Top, Right, Bottom</b> | When the width and the height of the container are changing, the object grows or shrinks synchronously with it.  |

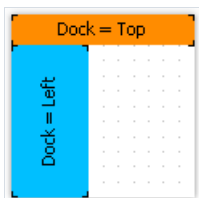
# Dock property

Every report object has this property. This property determines on which side of the container the object will be docked.

The **Dock** property can have one of the following values:

| Value         | Description  |
|---------------|--|
| <b>None</b>   | Value by default. The object is not docked.  |
| <b>Left</b>   | The object is docked to the left edge of the container. The height of the object will be equal to the height of the container*.  |
| <b>Top</b>    | The object is docked to the top edge of the container. The width of the object will be equal to the width of the container*.     |
| <b>Right</b>  | The object is docked to the right edge of the container. The height of the object will be equal to the height of the container*. |
| <b>Bottom</b> | The object is docked to the lower edge of the container. The width of the object will be equal to the width of the container*.   |
| <b>Fill</b>   | The object occupies all the free space of the container.   |

\* this is not quite so, if several objects have been docked at the same time. The figure below shows two objects, the first one has been docked to the top edge of the container and the second - to the left:



As seen, the height of the second object is equal to height of the free space, which remains after docking the first object.

The docking behavior depends on the object's creation order. You can change the order in the context menu of an object. To do this, select either the "Bring to front" or "Send to back" menu items.

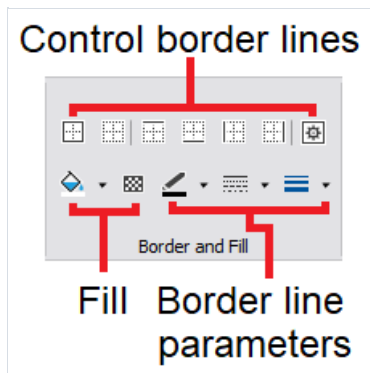
# Formatting


In this section, we will look at the following questions:

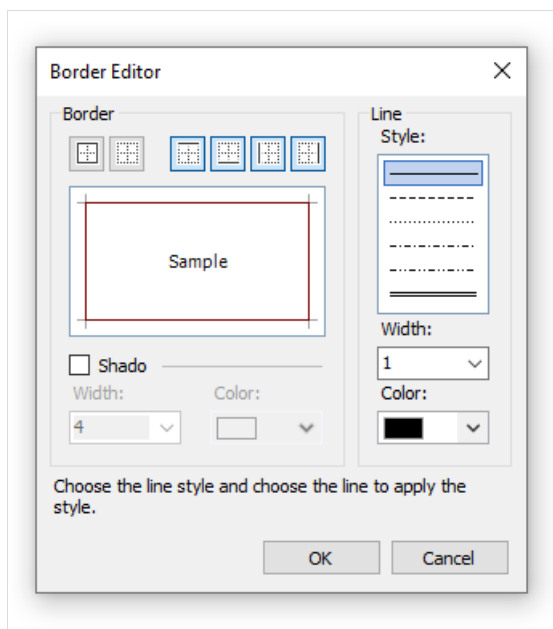
- changing an object's appearance;
- changing the format of printing values;
- automatically changing the appearance of an object when fulfilling some kind of condition;
- hide unnecessary values;
- highlighting even data rows in different colors.



# Border and fill

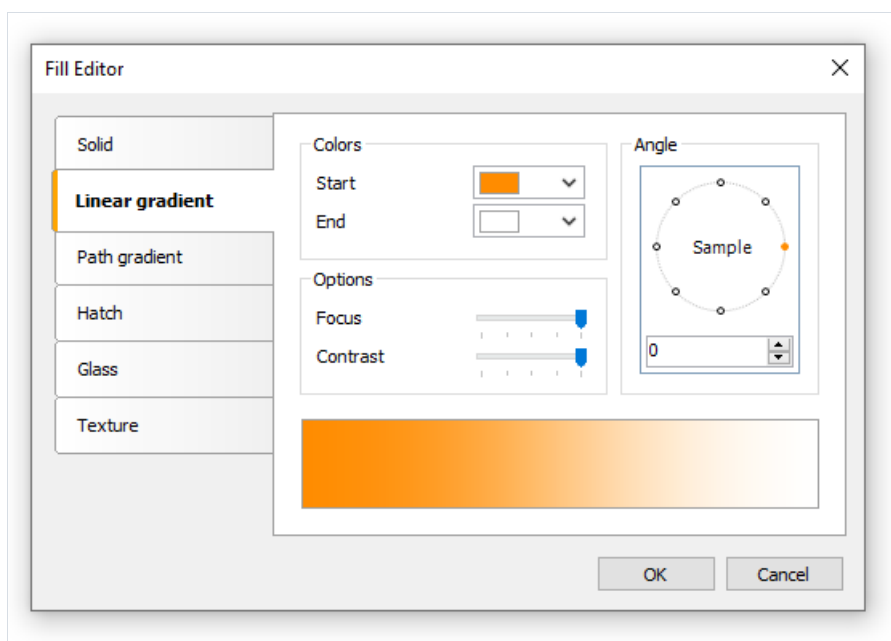
Almost all report objects have the border and fill. To work with these properties, use the "Border and Fill" toolbar:



The object's border consists of four lines. Each line can have different width, color and style. The toolbar buttons affect all lines of frame. The  button displays a dialog which allows to set up each line separately:

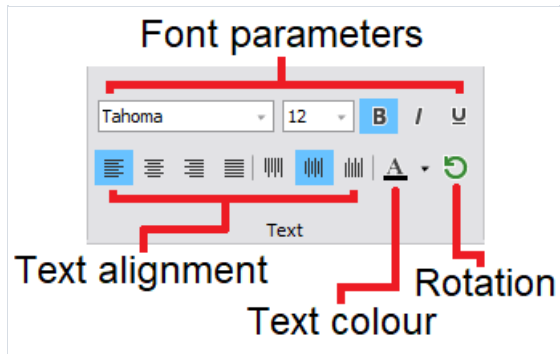


To work with fill, there are two buttons on the toolbar. The  button allows to choose a color for the solid fill type. The  button displays a dialog which allows to choose between different fill types:



# Text formatting


To change the "Text" object appearance, use the "Text" toolbar:

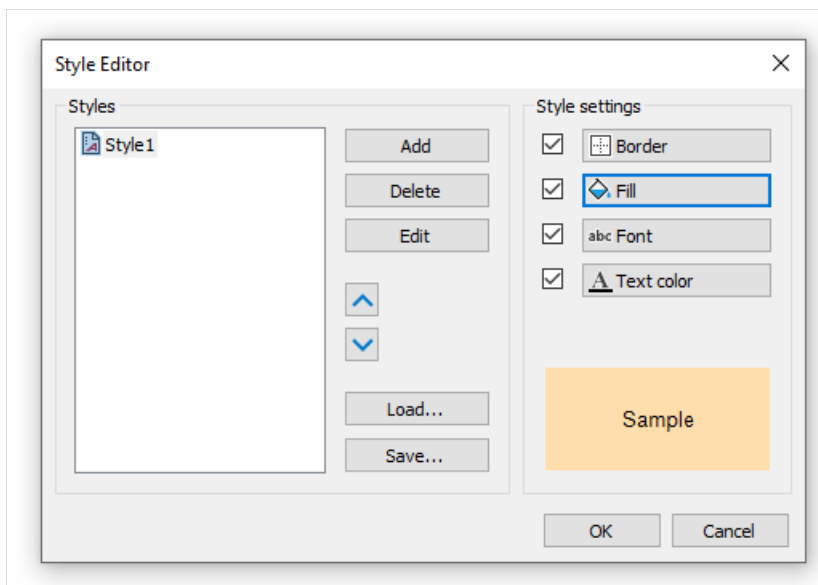


# Styles

To set up the object appearance, you may use styles. Style is a set of the following properties:

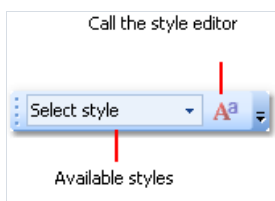
- border;
- fill;
- font;
- text color.

The list of styles is stored in a report. You can control it either from the "Report|Styles..." menu or by the  button in the "Style" toolbar:



You can set an object's style in the following ways:

- set the "Style" property in the "Properties" window;
- use the "Style" toolbar:



If the toolbar is not present on the screen, enable it in the "View|Toolbars" menu.

When you set the object's style, the object's appearance will be changed according the style settings. When you change the style settings, the object with that style will change automatically.

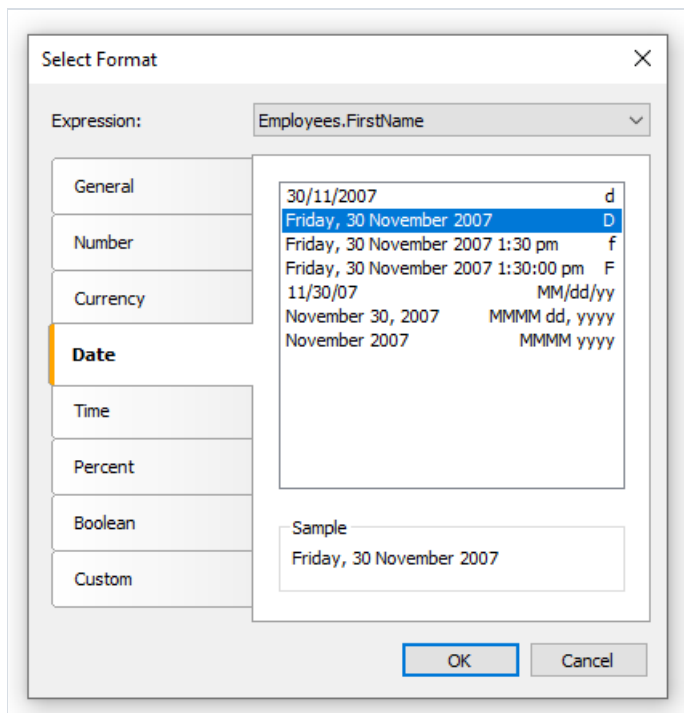


# Data formatting

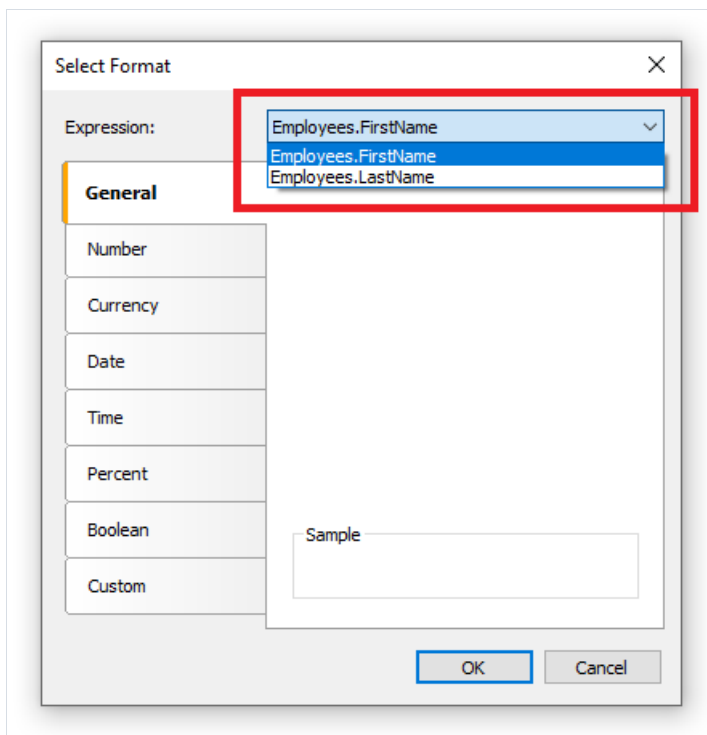
To print a textual data in a report, the "Text" object is used. It applies the default formatting to all data that comes from data source. For example, the data source columns of type `DateTime` will be printed in the following way (it depends on your system's regional settings):

4/2/2024 6:04:52 PM

If you need to print the date part only, you have to set up the data formatting. To do this, right-click the "Text" object to show its context menu. In the menu, choose the "Format..." item. You will see the format editor window:




You may choose one of the available formatting types or set up own formatting string. To do this, select the "Custom" formatting. If the "Text" object contains several data columns or expressions, you may choose appropriate format for each of them. To do this, select the expression in the top of the window, then choose the format:

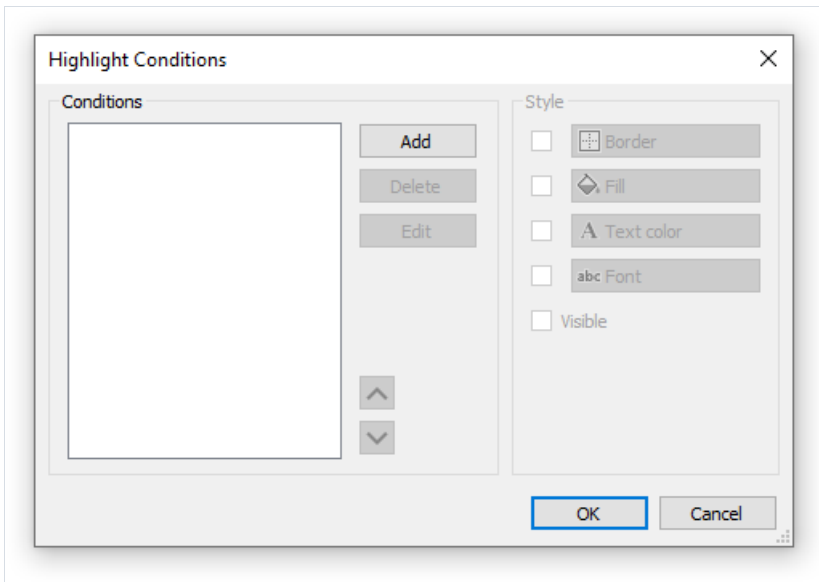


You may also format the data using the `String.Format` method. Get the help on this method in the MSDN.

```
Today is [String.Format("{0:d}", [Date])]
```

# Conditional highlighting

There is an possibility to change the "Text" object's appearance depending on the given conditions. For example, an object can be highlighted with red color if it has a negative value. This feature is called "conditional highlighting". To set up it, select the "Text" object and click the  button on the "Text" toolbar. You will see the following dialog window:

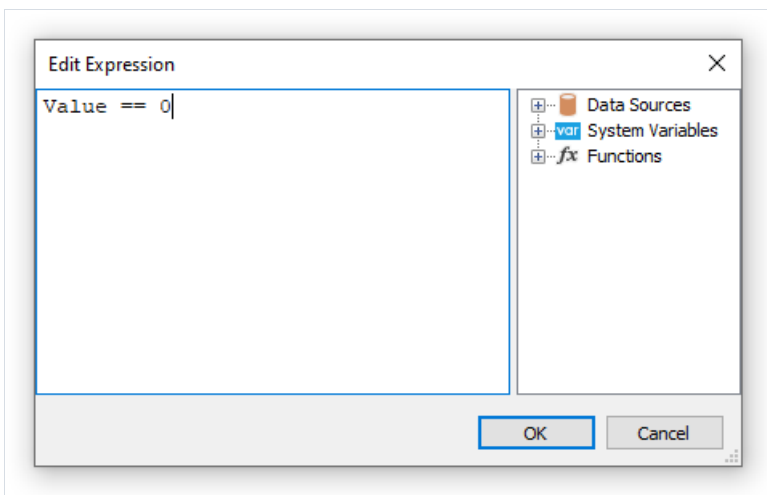


It is possible to define one or several conditions and set up the style for every condition. Style can contain one or several settings:

- fill;
- text color;
- font;
- object's visibility.

You can indicate, which settings need to be changed when the condition is met. For this, check the needed setting using the checkbox. By default, a new style contains one setting - the text color.

In order to create a new condition, click the "Add" button. You will see an expression editor:



Here, it is possible to write any expression which returns a boolean result. In many cases you will use the `Value` variable, which contains the currently printing value.

Let us look at the following example: we have a "Text" object, in which we print the amount of products in stock:

```
[Products.UnitsInStock]
```

We want to paint the object red, if the amount of products = 0. For this, we create the following condition:

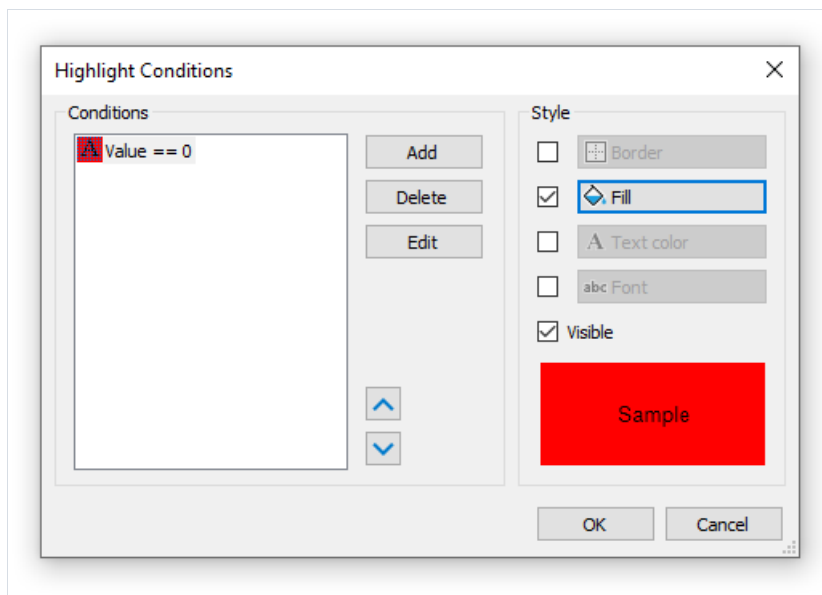
```
Value == 0
```

In the given case, we used the `Value` variable, which has got a printed value. If there are several expressions in an object, then this variable will have the value of the last expression. Instead of `Value`, you can use a data column:

```
[Products.UnitsInStock] == 0
```

The expression is written in C# style. This is so, if the chosen report language is C#. For VisualBasic.NET you must use the single `=` sign. The report language can be changed in the "Report|Options..." menu.

Configure the style for the given condition in such a way that only fill can be used, and choose the red color:



When printing an object which has a zero value, it will be red. Let us make our example more complex, we will add another condition. If the units in stock is less than 10, it must be printed yellow. To do this, open the condition editor and click the "Add" button. The second condition will be like this:



```
Value < 10
```

In case where several conditions have been indicated, FastReport checks all the conditions, starting from the first one. If a certain condition is met, FastReport applies its style settings to the object, and the process stops. It is important to put the conditions in a correct order. The order which we have seen in this example is correct:

1. Value == 0
2. Value < 10

If we swap conditions, then the highlighting will work wrongly.

1. Value < 10
2. Value == 0

In the given case, the `Value==0` will not be executed, because when the value is zero, then the first condition will be met. In order to change the order of the conditions, use the  and  buttons.

# Hiding zero values

The "Text" object has the `HideZeros` property which can be used to hide zero values. Lets look at an object with the following contents:

```
Total elements: [CountOfElements]
```

If the value of variable `CountOfElements` is equal to 0, and the property `HideZeros` is set to `true`, the object will be printed as follows:

```
Total elements:
```

The "Text" object also has the `HideValue` property which can be used to hide the value of an expression which is equal to the given value. For example, if the property value is `0`, then all the zero fields will be hidden. This property can also be used for hiding zero dates. As a rule, it's a date like "1/1/0001" or "1/1/1900". In this case the value of `HideValue` property must be like this:

```
1/1/1900 0:00:00 AM
```

As you can see, apart from the date, you need to indicate time as well. This is necessary because the value of the date in .NET contains time also.

Important note: this mechanism depends on the regional settings of your system, which can be set in the control panel. This happens because FastReport compares strings using the `ToString()` method. This method converts an expression value into a string. In relation with this, be careful when building reports which can be launched on a computer with different regional settings.

Finally, the `NullValue` property of the "Text" object allows to print some text instead of a null value. It is often used to print the dash instead of a null value. Lets look at an object with the following contents:

```
Total elements: [CountOfElements]
```

If the value of variable `CountOfElements` is `null`, and the property `NullValue` is set to `--`, the object will be printed as follows:

```
Total elements: --
```

# Hide duplicate values

The "Text" object has **Duplicates** property which allows to control how duplicate values will be printed. This property can be used if the "Text" object is on the "Data" band. The values considered duplicate if they are printed in the near by data rows.

The **Duplicates** property can have one of the following values:

- Show - show the duplicates (by default);
- Hide - hide the object with duplicate value;
- Clear - clear the object's text, but show the object;
- Merge - merge several objects with the same value.

The difference between these modes is shown in the figure below:



# Highlight odd/even data rows

In order to improve the appearance of a report, you can highlight even data rows in different colors. This can be done by using the `EvenStyle` property of the band or its objects. The property contains a style name, which will be used to highlight even band rows.

It is preferable to use the `EvenStyle` property of the object instead of the band. This avoids possible problems when exporting the report.

In order to configure the highlighting, do the following:

1. Define the style, which will be used for highlighting the rows. This can be done in the "Report|Styles..." menu.
2. Indicate the name of the new style in the `EvenStyle` property of the band or its objects.

By default, objects use only a fill property of the style given in the `EvenStyle` property. This behavior is defined in the `EvenStylePriority` property - by default it is `UseFill`. If you need to use the rest of the style parameters, set this property to `UseAll`.

A ready report, which uses this technique, can look like this:

| Product name       | Unit price |
|--------------------|------------|
| Chai               | 18,00      |
| Chang              | 19,00      |
| Chartreuse verte   | 18,00      |
| Côte de Blaye      | 263,50     |
| Guaraná Fantástica | 4,50       |
| Ipoh Coffee        | 46,00      |
|                    |            |



## Report with one "Data" band

This type of a report is more often required. It allows printing a list of rows from the data source. For example, this can be a customer list.

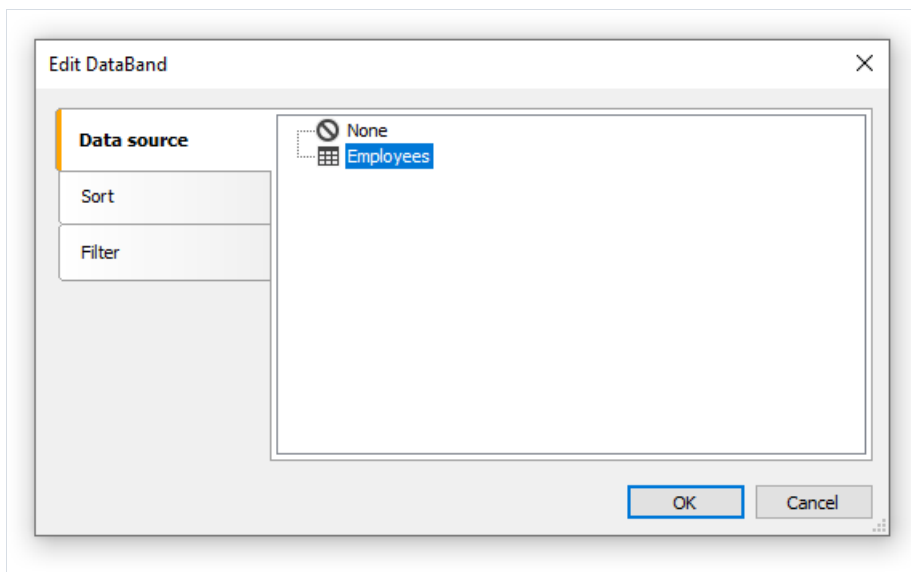
# Connecting a band to data source

To print a data from the data source, you will need the "Data" band, which is supposed to be connected to the data source. The band will be printed as many times as there are rows in the data source.

If the "Data" band is not connected to the source, it gets printed once.

When you create a new report, it already contains several empty bands, including the "Data" band. This band can also be added into the report from the "Configure Bands" window, by choosing the "Report|Configure Bands..." menu item.

In order to connect a band to data, double click it. Choose data source in the editor window and click "OK":



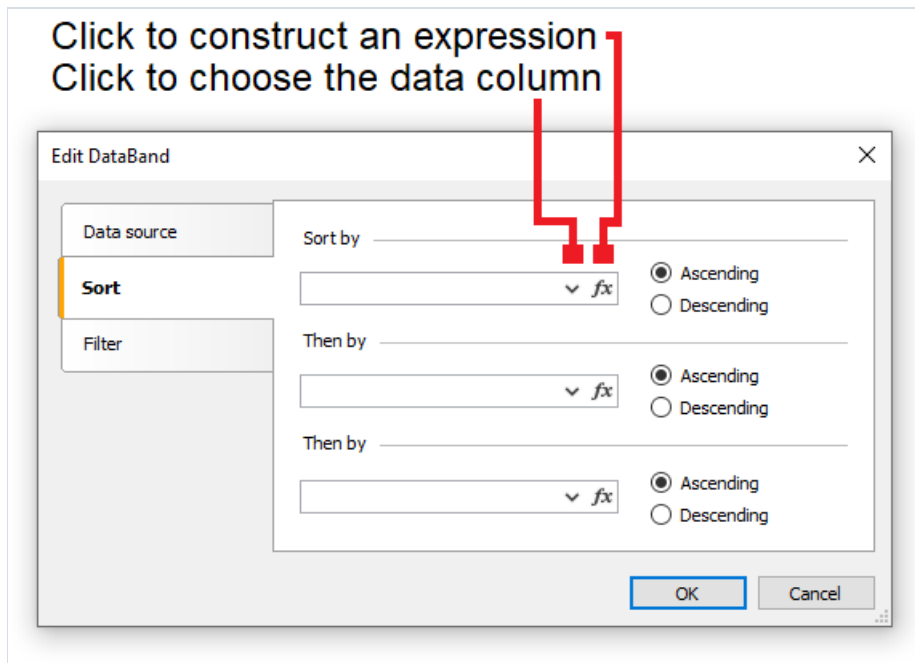
# Printing the text

After you have connected the band to a data source, you can place the "Text" object on the band, which will display the information from a data column. The fastest method to do this - drag a data column from the "Data" window and drop it on the band. Read more about the "Text" object in the ["The Text object"](#) chapter.

# Sorting the data

By default, the "Data" band prints data in natural order. Often it is needed to sort the data before printing. For example, a list of customers can be comfortably presented by sorting it in an alphabetical order.

You can control sorting in the "Data" band editor. In order to call the editor, double click on a free space at the band:

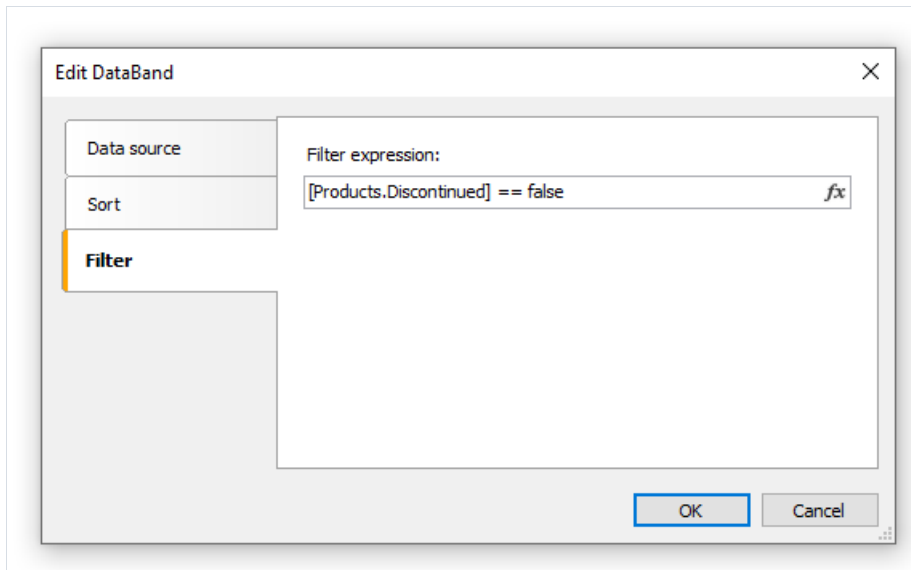


As a sort criteria, you can use either a data column or an expression. You can indicate several (not more than three) sorting conditions. This can be needed, for example, if you want to sort the list of customers by their cities, and after that, by customer's name. For each condition, you can choose the order of sorting - ascending or descending.

Another method of sorting data - use the SQL query as a data source. The query will be executed on the data server and return a sorted rows.

# Filtering the data

In order to filter a row, which is printed in the "Data" band, call its editor and switch to the "Filter" tab:



As a filter expression, you can indicate any correct expression. More details about expressions can be found in the ["Expressions"](#) chapter.

In the example above the following filter is used:

```
[Products.Discontinued] == false
```

This means that, all the data rows whose **Disconnected** flag is equal to **false** will be chosen.

You can use complex filter condition:

```
[Products.Discontinued] == false && [Products.UnitPrice] < 10
```

This means that, all the data rows whose **Disconnected** flag is equal to **false**, and whose price is less than 10 will be chosen.

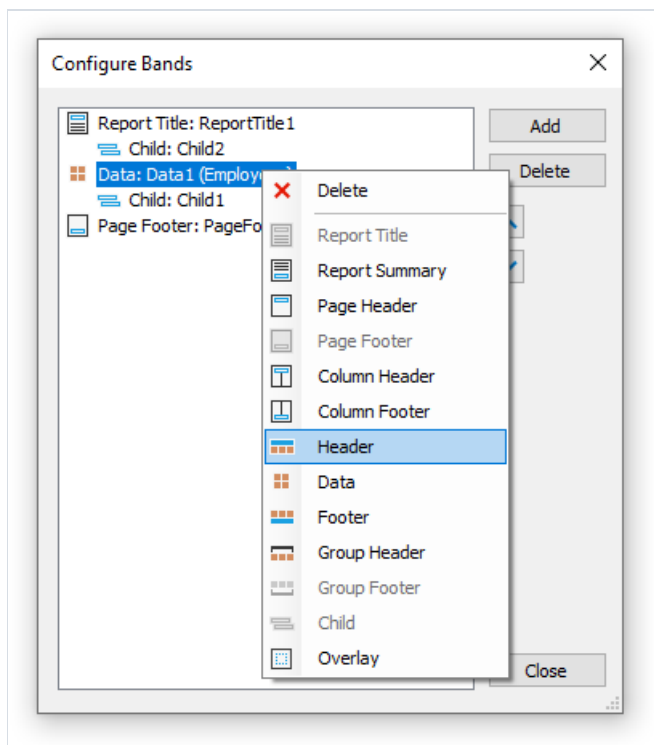
This filtration method supposes that, the data source contains all rows, part of which will be filtered. If the data source contains a large amount of rows, this can seriously slow down the report. In this case you can use SQL query as a data source, in which you can perform the needed filtration. The query will be executed on the data server and return only those rows which are needed in the report.

You also may use dialogue forms to perform data filtering. See more details in the ["Dialogue forms"](#) chapter.

# Data header and footer

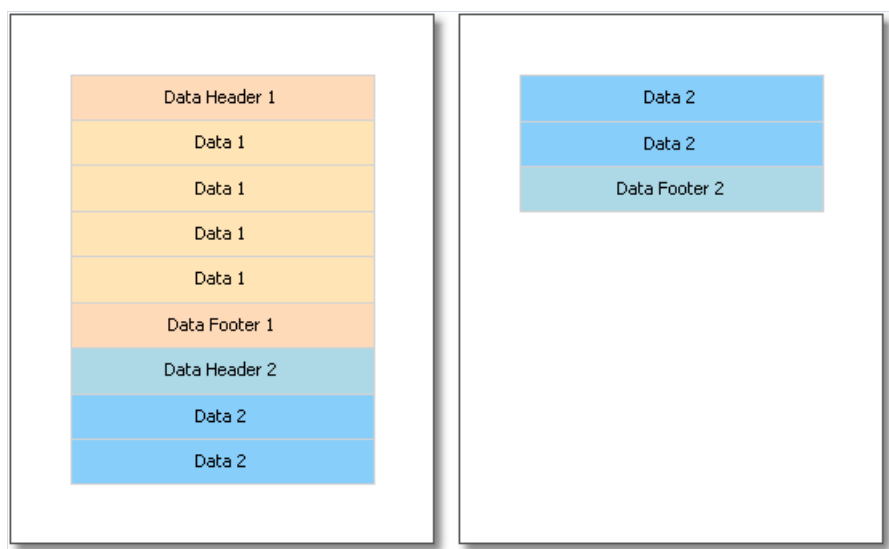
A "Data" band can contain a header and a footer. The header will be printed once before data, the footer will be printed after the output of all data.

In order to add a header and footer to a "Data" band, choose the "Report|Configure Bands..." menu item. In the window, select the "Data" band and right click the mouse. In the context menu choose the "Header" and/or "Footer" items:



These bands can be useful in the following situations:

- when printing several lists on one page ("master-master" reports). Every data band in this case can have its own header and footer:



- when printing one list, if the list does not fit on one page of the prepared report. By using the "Repeat On Every Page" property of the data header/footer, you can print these bands on every page of the report:

|             |             |
|-------------|-------------|
| Data Header | Data Header |
| Data        | Data        |
| Data        | Data        |
| Data        | Data Footer |
| Data        |             |
| Data        |             |
| Data        |             |
| Data        |             |
| Data Footer |             |

# Breaking data and keeping it together

In this section, we will look at two modes of data printing - "Break" and "Keep together".

In an regular band printing mode, FastReport checks if there is enough space on the current page to print a band. If there is not, the band is printed wholly on the next page. If the `CanBreak` property of the band is enabled, FastReport will try to print the part of the band on the available space, that is, "break" it.

An attempt to break a band can be either successful or not. It depends on the type of object which has been placed on the band, and its settings. The following objects can be broken:

- "Text";
- "Rich Text";
- "Table".

These objects have the `CanBreak` property as well. If it is enabled, then the object can be broken. Non-breakable objects are always displayed wholly, there, where they have enough place.

In the figure below, how a band can be broken is shown.



Break algorithm does not always work correctly. The artifacts can occur in a situation, when there are several objects with different font size on a band.

The goal of band breaking is to save the space on the printed sheet. Data keeping's goal is contrary: display a set of bands wholly on one sheet. In this case there will be a lot of unused space on the sheets, but the data is printed in a way that it is comfortable to percept.

The "keep together" mechanism allows keeping a set of bands together on one page (or column, if the report has columns). If, when printing, kept data reaches the end of the page, FastReport relocates all data which has been printed already onto a new page.

You can use the "keep together" in the following cases:

- printing all the rows of the "Data" band together;
- printing all the elements of a group (header, data, footer) together;
- printing the row of the master data source together with all detail rows (in the "master-detail" report);
- printing the report header or the data header together with at least one data row;
- printing the report footer or data footer together with at least one data row;
- printing the parent and child bands together.

Let us look at the use of "keep together" mechanism.

To keep together all data rows or group elements (header, data, footer), enable the `KeepTogether` property. This property is used in the "Data" and "Group Header" bands. The figure below shows how data is printed with and without keeping together:





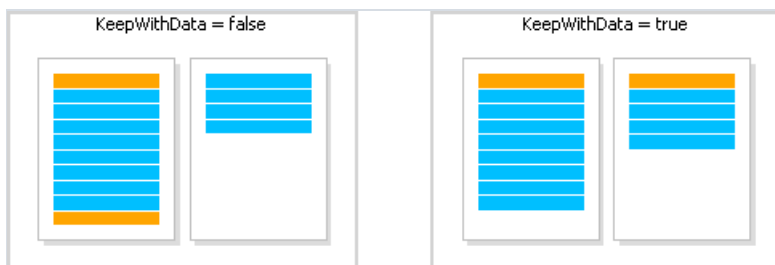
To keep master data row together with detail data rows, enable the `KeepDetail` property of the "Data" band. This property is used in a report of "master-detail" type:



To prevent "hanging" headers and footers, use the `KeepWithData` property. The following bands have got this property:

- report header;
- report footer;
- data header;
- data footer;
- group header;
- group footer.

This property allows to keep header/footer with at least one data row:

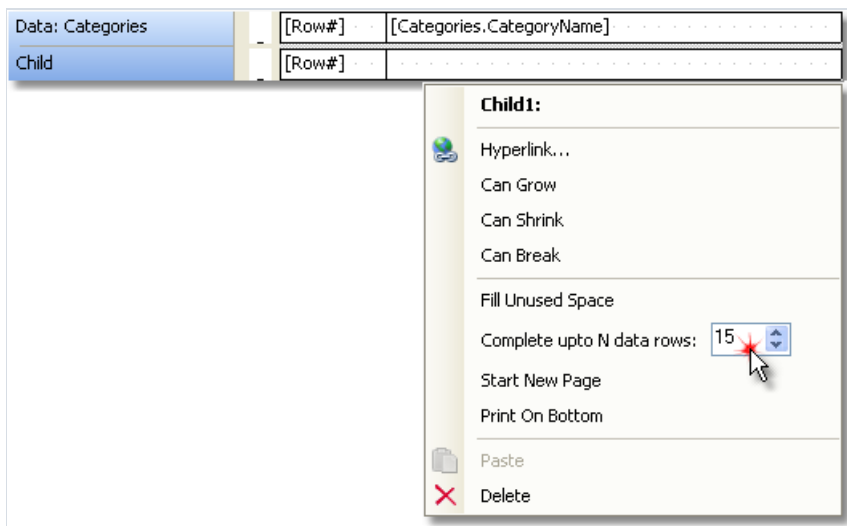


In order to keep a band and its child band together, enable the `KeepChild` property.

# Printing empty data rows

Often enough, when printing on pre-printed forms, a certain amount of data rows is supposed to be printed. If the actual data is less than needed amount of rows, then empty rows need to be printed. This can be done with the help of the "Child" band, by attaching it to the "Data" band.

The "Child" band has got a `CompleteToNRows` property. If this property is set to value greater than 0, the band will be used for additional data rows up to the indicated amount. For example, we need to print 15 rows, but there are only 8 rows in the data source. In this case, the "Child" band will be printed 7 times.



The prepared report will look like this:

|    |                |
|----|----------------|
| 1  | Beverages      |
| 2  | Condiments     |
| 3  | Confections    |
| 4  | Dairy Products |
| 5  | Grains/Cereals |
| 6  | Meat/Poultry   |
| 7  | Produce        |
| 8  | Seafood        |
| 9  |                |
| 10 |                |
| 11 |                |
| 12 |                |
| 13 |                |
| 14 |                |
| 15 |                |

If the data source has more rows than indicated in the `CompleteToNRows` property, then an empty row will not be printed.

Another way to print an empty row is to fill the free space on a page. In this case, the "Child" band is attached to the bands of either the "Data Footer" or "Group Footer" types and fills the free space on the page. The footer band will be printed at the bottom of the page.

In order to print an empty row this way, attach the "Child" band to the footer band and enable its `FillUnusedSpace` property. You will see that the child band is now displayed above the band it is attached to. In the figure below, the "Child" band is attached to the "Report Summary" band:

|                  |                           |
|------------------|---------------------------|
| Data: Categories | [Categories.CategoryName] |
| Child            |                           |
| Report Summary   | report summary            |


When we run such a report, we will see the following:

|                |
|----------------|
| Beverages      |
| Condiments     |
| Confections    |
| Dairy Products |
| Grains/Cereals |
| Meat/Poultry   |
| Produce        |
| Seafood        |
|                |
|                |
|                |
|                |
|                |
| report summary |

# Printing "No data" text

When the data band is connected to an empty data source, it will not be printed. Sometimes it is required to print some text like "No data" instead of just an empty page. To do this:

- add a child band to the data band;
- set the child band's `PrintIfDatabandEmpty` property to `true` (it can be done in the "Properties" window);
- put the "Text" object on a child band and write the "No data to display" text in it.



| Page Header     |   | CompanyName             | Country             |
|-----------------|---|-------------------------|---------------------|
| Data: Customers | - | [Customers.CompanyName] | [Customers.Country] |
| Child           | - | No data to display      |                     |

The report will be printed in the following way:

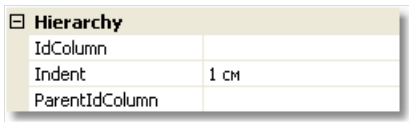
- if the data source has some data rows, the data band will be printed, together with all its related bands (data header/data footer);
- if the data source is empty, only the child band with "No data to display" text will be printed.

# Printing hierarchy

A "Data" band allows printing a hierarchical list. For this, one band and one data source are used. Hierarchy must be defined in the data source with the help of two data columns:

1. Key column. This is the data row identifier.
2. Column which contains the key of this item's parent.

In order to print such a source in a hierarchical form, you need to set the following "Data" band properties. This can be done in the "Properties" window:



|                  |      |
|------------------|------|
| <b>Hierarchy</b> |      |
| IdColumn         |      |
| Indent           | 1 cm |
| ParentIdColumn   |      |

- indicate the key column in the `IdColumn` property;
- indicate the column containing parent value, in the `ParentIdColumn` property;
- indicate the hierarchy indent in the `Indent` property.

Let us look at an example of how to print a hierarchy of employees from the "Employees" demo table. The table has got two columns which we need:

- `EmployeeID` column is the key and contains the employee ID;
- `ReportsTo` column contains the ID of "parent" employee.

Create a report that looks like the following:

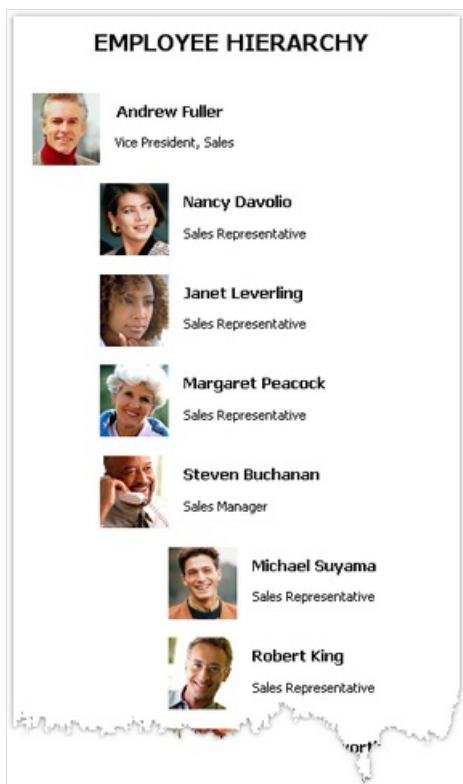


Set the "Data" band properties, which is responsible for the hierarchy, in the following way:



|                  |                      |
|------------------|----------------------|
| <b>Hierarchy</b> |                      |
| IdColumn         | Employees.EmployeeID |
| Indent           | 1,5 cm               |
| ParentIdColumn   | Employees.ReportsTo  |

When we run a report, we will see the following:

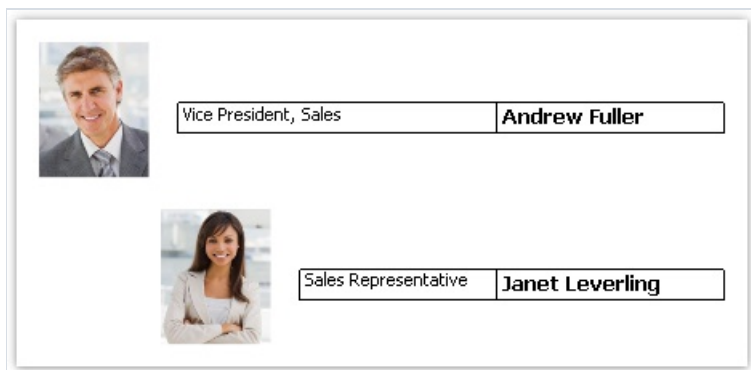


When printing hierarchy, FastReport shifts the band to the right (by a value indicated in the **Indent** property), and also decreases the band width by the same value. This allows you to use the **Anchor** property of band's objects.

Here is possible values for this property that can be used in this case:

- Left, Top (by default) - the object is moved with the band;
- Right, Top - the object stays in its original position;
- Left, Right, Top - the right side of the object stays at its original position, the left side is moved with the band.

It allows you to get some useful effects:



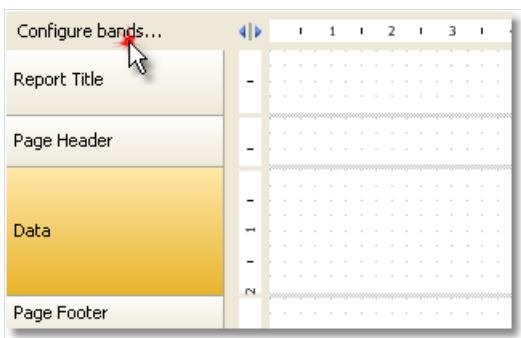
In this example, the picture object has **Anchor** property set to **Left, Top**; the object with job title is anchored to **Left, Right, Top**; the object with the name is anchored to **Right, Top**.

# Master-detail report

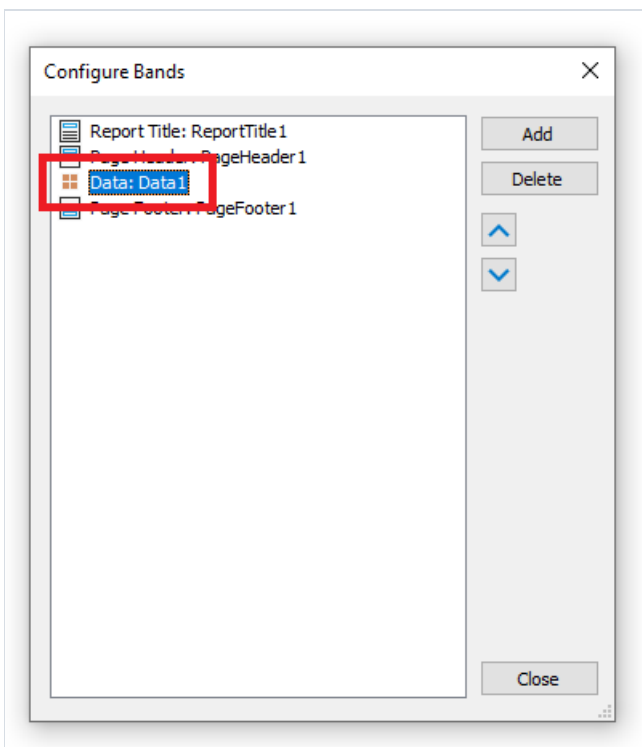
By using two "Data" bands, it is easy to create a report of the "master-detail" type. In this report, two data sources, between which there is a relation, are used. One row of the master source can correspond with several rows of the detail source. More details on relations can be found in the ["Data"](#) chapter.

It is necessary to place a band in a report in such a way that, the master band contains the detail band inside it. This can be done in the "Configure bands" window, which can be called in the "Report|Configure Bands..." menu.

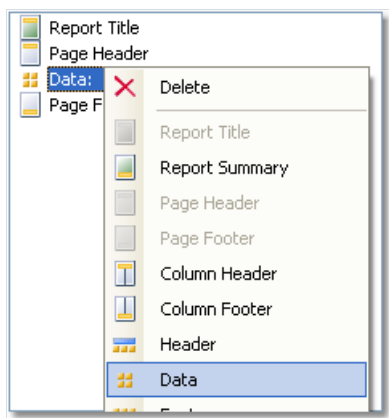
Let us look at the creation of a master-detail report from a scratch. For this, we will run the report designer and create a new empty report. It already contains one "Data" band:



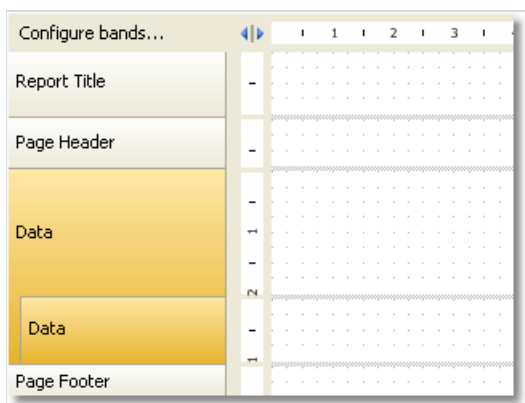
In order to add a detail data band, call the "Configure Bands" window. This can be done by pressing the "Configure bands..." button, shown in the figure, or by choosing the "Report|Configure Bands..." menu item. In the configurations window, the band structure is displayed:



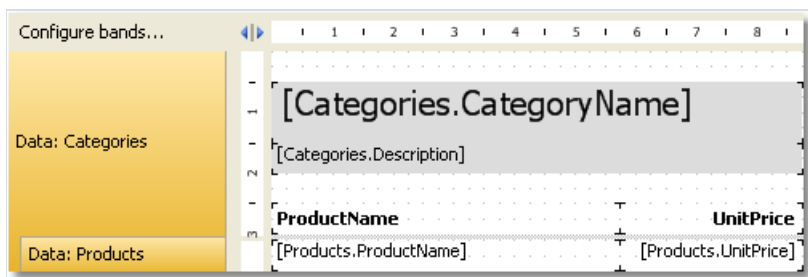
Select the "Data" band, as shown in the figure, and right click the mouse in order to show the context menu (or press the "Add" button in the lower part of the window). In the window which will open, select "Data" band:



After this, a nested "Data" is added to the selected band. Close the window by pressing the "Close" button. You will see that the report template changes in the following way:



Nested data bands are clearly seen on band structures on the left part of the window. After this, you need to connect the band to the corresponding data source and place data columns on the bands. We will be using two data sources - Categories and Products - from the demo data base which comes with FastReport:



If we run the report, we will see the following:



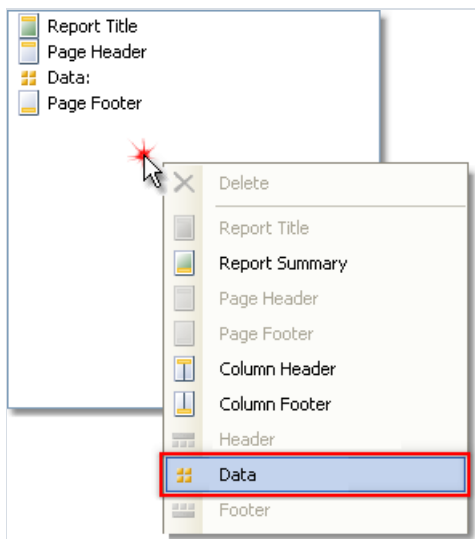
| Beverages  |           |
|--|-----------|
| Soft drinks, coffees, teas, beers, and ales                |           |
| ProductName  | UnitPrice |
| Chai   | 18,00p.   |
| Chang  | 19,00p.   |
| Guaraná Fantástica   | 4,50p.    |
| Sasquatch Ale  | 14,00p.   |
| Steeleye Stout   | 18,00p.   |
| Côte de Blaye  | 263,50p.  |
| Chartreuse verte   | 18,00p.   |
| Ippoh Coffee   | 46,00p.   |
| Laughing Lumberjack Lager                                  | 14,00p.   |
| Outback Lager  | 15,00p.   |
| Rhönbräu Klosterbier                                       | 7,75p.    |
| Lakkaikööri  | 18,00p.   |
| Condiments   |           |
| Sweet and savory sauces, relishes, spreads, and seasonings |           |
| ProductName  | UnitPrice |
| Aniseed Syrup  | 10,00p.   |
| Chef Anton's Cajun Seasoning                               | 22,00p.   |
| Chef Anton's Gumbo Mix                                     | 21,35p.   |

In this way, you can create a master-detail report type with unlimited nested data, for example, master-detail-subdetail. Another method, which is used for the creation of master-detail report type, connected with the use of nested reports. Nested reports will be looked at in the ["Subreports"](#) section.

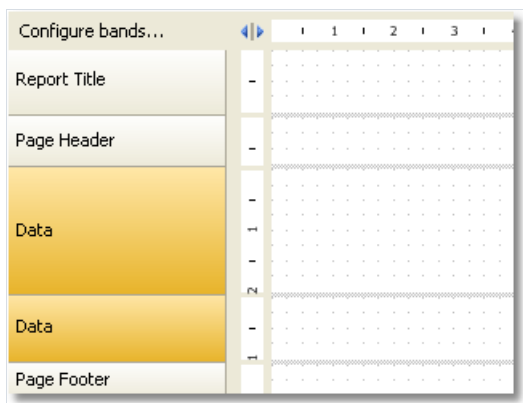
# Master-master report

On the report page, you can print several simple lists. This can be done, by placing on the page two or several "Data" bands. Contrary to master-detail report, where bands are nested into each other and prints data from related sources, in a report of this type both bands and data sources do not depend on each other.

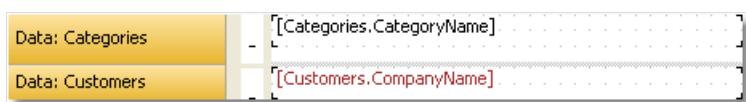
We will show by an example, how to create a report which prints two lists on a page - Categories table and Customers table. We will create a new report and add into it needed data sources. In order to add a second "Data" band, call the "Configure Bands" window.



Right click on an empty place of the list, as shown in the figure, and select "Data" band in the context menu. This creates a new independent "Data" band. The report template will be like this:



Now, we will connect the band to the data source and place several data columns on it:



If we run the report, we will see the following:

Beverages

Condiments

Confections

Dairy Products

Grains/Cereals

Meat/Poultry

Produce

Seafood

Alfreds Futterkiste

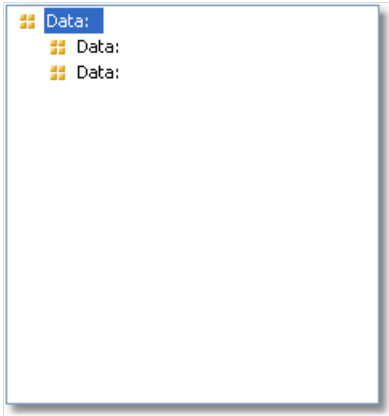
Ana Trujillo Emparedados y helados

Antonio Moreno Taquería

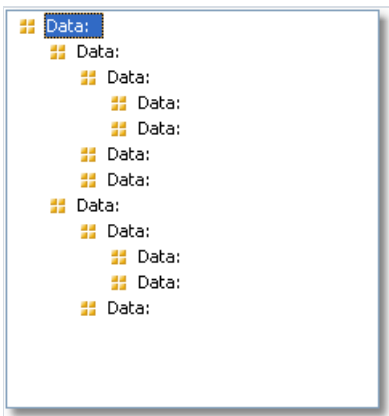
Around the Horn

# Master-detail-detail report

A "Data" band can contain one or several nested "Data" bands. This allows building a master-detail-detail report type. To do this, call the "Configure Bands" window, right click on the master "Data" band and add a detail "Data" band to it. Repeat this procedure in order to add the second detail band:



In this way, it is possible to add an unlimited number of detail bands to the master "Data" band. An example report structure can be like this (this is just an example; it only demonstrates the abilities of FastReport):



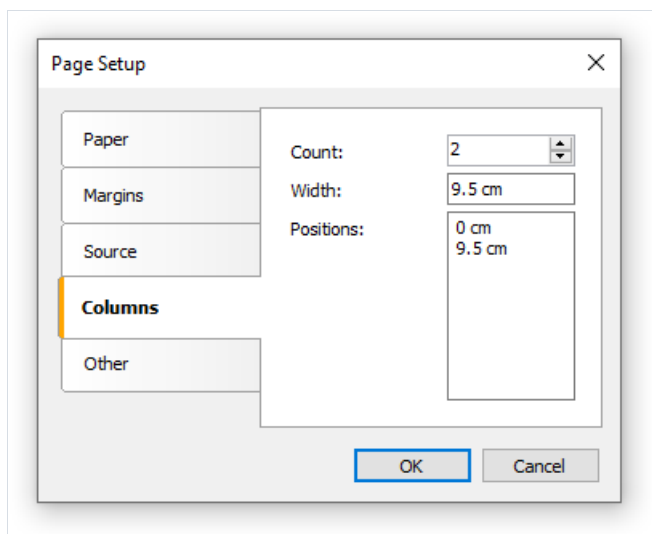
# Multicolumn reports

An ordinary report prints data as long as it has not reached the end of the page. After this, a new page is formed and printing continues on it. A report with columns prints data in several columns. When the end of the page has been reached, printing continues in a new column on the same page. In this sense, an ordinary report can be seen as a report with one column.

In FastReport there are two methods of printing columns.

# Page columns

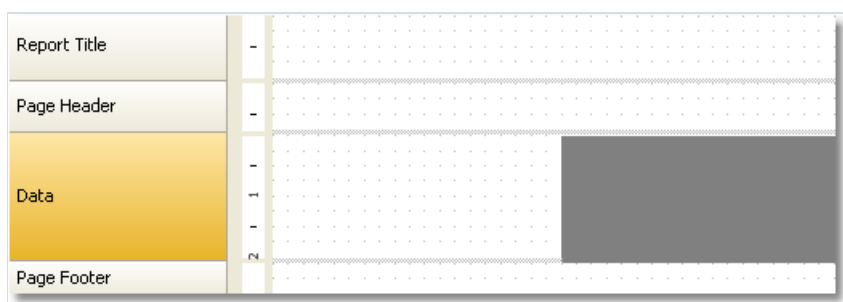
The first method is based on setting the number of columns of the report page. This is done in the "Page Setup" window on the "Columns" tab:



As seen, you can set the following column parameters:

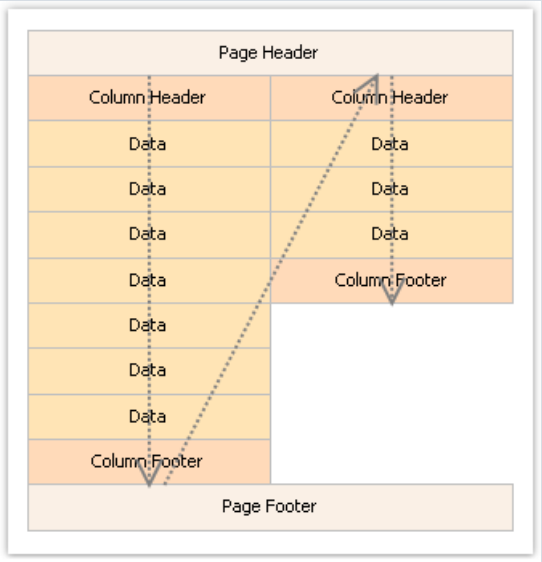
- column count;
- column width;
- the position of every column.

In order to transform an ordinary report into a report with columns, you need to set only the number of columns on the page. The rest of the parameters FastReport will calculate on its own. When you enable columns, the mode of bands in the designer changes:



The area shown in grey should never be used for placing objects on it. It is used to print next columns' objects.

For working with columns, the "Column Header" and "Column Footer" bands are used. As seen from their names, they print at the top and the bottom of every column respectively. The following figure demonstrates the bands printing order in the report with columns:



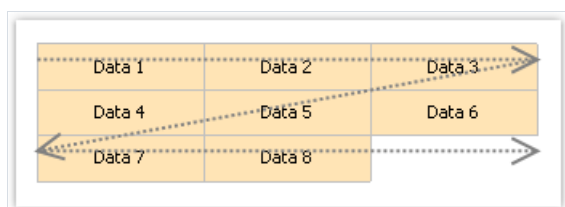
# Data band columns

Another method of printing a multicolumn report is based on using the "Data" band columns. The rest of the bands continue to be printed in one column.

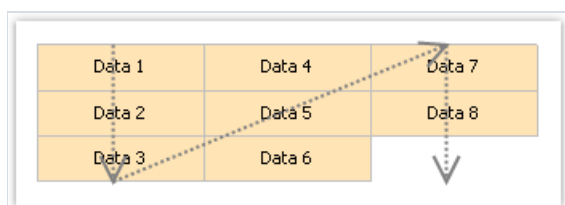
Column parameters can be configured using the `Columns` property, which can be changed in the "Properties" window. You can set up the following parameters:

- number of columns;
- column width;
- column printing mode. You can select either of the two modes - `AcrossThenDown` and `DownThenAcross` ;
- the minimum number of rows in one column, if the chosen mode is `DownThenAcross` .

Column band can be printed in either of the two modes. In the `AcrossThenDown` mode (the default one), columns are printed in the following way:



In the `DownThenAcross` mode, column printing occurs in the following way:



In this mode, FastReport calculates the number of data rows in a column in such a way that, columns are filled equally. You can also set the minimum number of rows in a column with the `Columns.MinRowCount` property.




# "Booklet"-type reports

When printing a report in form of a booklet, you will probably face with the following demands:

- separate report page - cover, table of contents, report contents, back cover;
- different page margins for even and odd pages;
- different header and footer on even and odd pages.

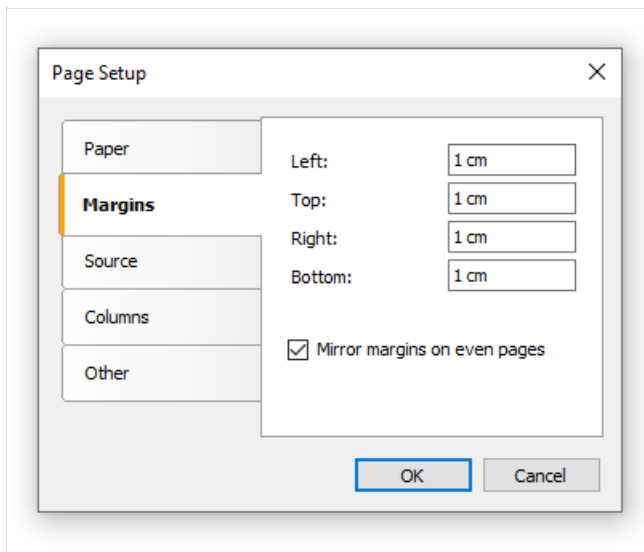
# Adding page into a report

You can add any number of pages into the report template. On each page you can place a separate report. To add a new page, click the  button on the toolbar. A page can also be added, by pressing the "Add New..." button and selecting the "Report page" item in the window.

For creating the "Table of Contents" section, you can use the technique described in the ["Interactive reports"](#) chapter.

# Page settings

In the "Page Setup" window, you can indicate that FastReport should mirror the left and the right margin for even pages:



If you need to start the page on an odd page number, set the `StartOnOddPage` property to `true`. When needed, FastReport prints the empty page before starting to print the indicated page.

# Printing on odd/even pages

All report objects have the `PrintOn` property. Using this property, you can print different objects on odd and even pages.

This property can be set in the "Properties" window.

This property determines on which pages the object can be printed. The property can have one of the following values or any combination of it:

- `FirstPage`;
- `LastPage` - the report must be double-pass;
- `OddPages`;
- `EvenPages`;
- `RepeatedBand`. This value refers to a band with the `RepeatOnEveryPage` property set to true;
- `SinglePage` - the report must be double-pass.

By default, the value of this property equals to

`FirstPage, LastPage, OddPages, EvenPages, RepeatedBand, SinglePage`. This means that the object will be printed on all pages of the report. In case the report has single page only, the object's visibility is determined by the `SinglePage` value only.

We will give several typical examples of using this property:

| Property value                                      | Where the object will be printed   |
|---|--|
| <b>FirstPage</b>                                    | Only on the first page.  |
| <b>LastPage, OddPages, EvenPages, RepeatedBand</b>  | On all pages except the first.   |
| <b>FirstPage, OddPages, EvenPages, RepeatedBand</b> | On all pages except the last.  |
| <b>RepeatedBand</b>                                 | Only on bands with the <code>RepeatOnEveryPage</code> property is set to <code>true</code> . |
| <b>FirstPage, LastPage, OddPages, EvenPages</b>     | On all bands except the repeated one.  |
| <b>FirstPage, LastPage, OddPages, RepeatedBand</b>  | Only on odd pages.   |
| <b>FirstPage, LastPage, EvenPages, RepeatedBand</b> | Only on even pages.  |

Note: Even and odd pages are determined by the page number in the rendered report, that counts pages starting from 0. For example, the second page displayed in the preview will be odd because its number in the rendered report is 1.

For example, to print different text on odd and even pages, put two "Text" objects on a band and setup them in the following way:

- the first object will be printed on odd pages. Set its `PrintOn` property to

- `FirstPage, LastPage, OddPages, RepeatedBand` (i.e. all values except `EvenPages` ).
- the second object will be printed on even pages. Set its `PrintOn` property to `FirstPage, LastPage, EvenPages, RepeatedBand` (i.e. all values except `OddPages` ).

These objects will never be printed at the same time. You could place them on top of each other.

All bands have the same property. To print different bands on odd and even pages, use the "Child" band. You can attach it to any band; this can be done in the "Configure Bands" window. Setup the main band and its child in the following way:

- the main band will be printed on odd pages. Set its `PrintOn` property to `FirstPage, LastPage, OddPages, RepeatedBand` (i.e. all values except `EvenPages` ).
- the child band will be printed on even pages. Set its `PrintOn` property to `FirstPage, LastPage, EvenPages, RepeatedBand` (i.e. all values except `OddPages` ).

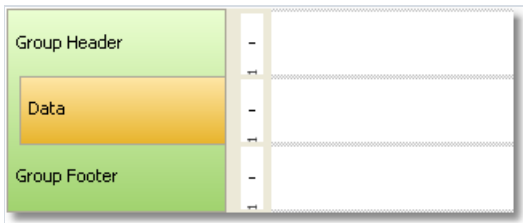
Bands can have different height, appearance and contents. Look at the following example which prints different page headers:

|             |   |  |         |
|-------------|---|--|---------|
| Page Header | - | PrintOn = FirstPage, LastPage, OddPages  | [PageN] |
| Child       | - | PrintOn = FirstPage, LastPage, EvenPages | [PageN] |

# Groups and totals

Earlier we looked at the "Master-detail" report type, which printed data from two related sources. FastReport allows creating a report which looks in the same way, but uses one data source. For this, groups are used.

A group is a set of three bands: "Group header", "Data" and "Group footer". In designer, this looks as follows:



A group always contains a header and data. Group footer is optional, you can delete it.

In order to use a group, you should set the group condition for the group header, and connect the data source to the "Data" band. The condition can be any expression, but as a rule, this is one of the data source columns. Group printing is done in the following way:

1. group header is printed;
2. data row is printed;
3. checks if the grouping condition has changed;
4. if the condition has not changed, next data row is printed (p.2);
5. if the condition has changed, the group footer is printed, and starts printing a new group (p.1).

Assuming that we have a Products table with the following data:

| CategoryName | ProductName                      |
|--------------|----------------------------------|
| Beverages    | Côte de Blaye                    |
| Beverages    | Chartreuse verte                 |
| Beverages    | Steeleye Stout                   |
| Beverages    | Guaraná Fantástica               |
| Beverages    | Sasquatch Ale                    |
| Beverages    | Rhönbräu Klosterbier             |
| Beverages    | Lakalikööri                      |
| Beverages    | Outback Lager                    |
| Beverages    | Ipoh Coffee                      |
| Beverages    | Laughing Lumberjack Lager        |
| Beverages    | Chang                            |
| Beverages    | Chai                             |
| Condiments   | Original Frankfurter grüne Soße  |
| Condiments   | Sirop d'érable                   |
| Condiments   | Chef Anton's Gumbo Mix           |
| Condiments   | Northwoods Cranberry Sauce       |
| Condiments   | Grandma's Boysenberry Spread     |
| Condiments   | Chef Anton's Cajun Seasoning     |
| Condiments   | Aniseed Syrup                    |
| Condiments   | Louisiana Hot Spiced Okra        |
| Condiments   | Veggie-spread                    |
| Condiments   | Louisiana Fiery Hot Pepper Sauce |
| Condiments   | Gula Malacca                     |
| Condiments   | Genen Shouyu                     |

Data can be grouped on the **CategoryName** column. This column will be printed in the group header. The data itself is presented by the **ProductName** field. The report will be as follows:

|                |   |                                      |
|----------------|---|--------------------------------------|
| Group Header:  | - | [[Products.Categories.CategoryName]] |
| CategoryName   | - |                                      |
| Data: Products | - | [Products.ProductName]               |
| Group Footer   | - |                                      |

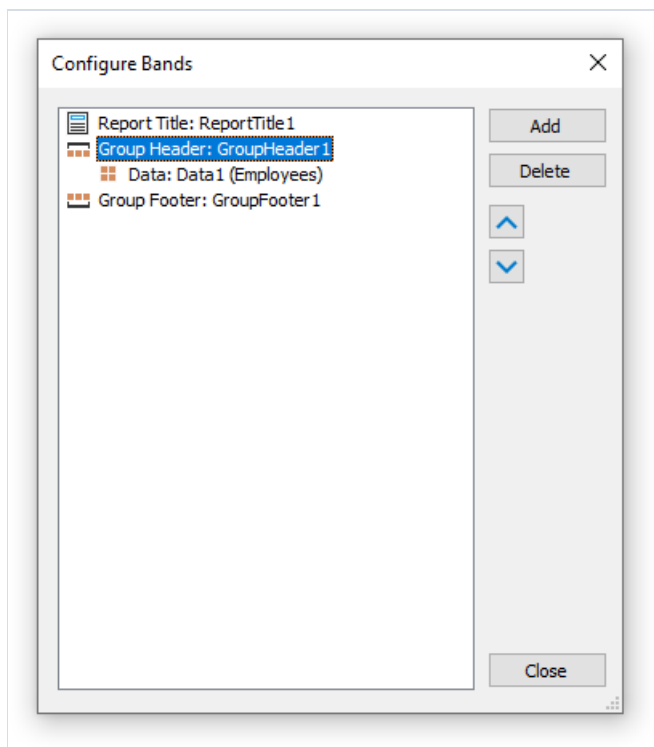
If we run the report, the following will be seen:

| Beverages                        |
|----------------------------------|
| Côte de Blaye                    |
| Chartreuse verte                 |
| Steeleye Stout                   |
| Guaraná Fantástica               |
| Sasquatch Ale                    |
| Rhönbräu Klosterbier             |
| Lakalikööri                      |
| Outback Lager                    |
| Ipoh Coffee                      |
| Laughing Lumberjack Lager        |
| Chang                            |
| Chai                             |
| Condiments                       |
| Original Frankfurter grüne Soße  |
| Sirop d'érable                   |
| Chef Anton's Gumbo Mix           |
| Northwoods Cranberry Sauce       |
| Grandma's Boysenberry Spread     |
| Chef Anton's Cajun Seasoning     |
| Aniseed Syrup                    |
| Louisiana Hot Spiced Okra        |
| Veggie-spread                    |
| Louisiana Fiery Hot Pepper Sauce |
| Gula Malacca                     |
| Genen Shouyu                     |

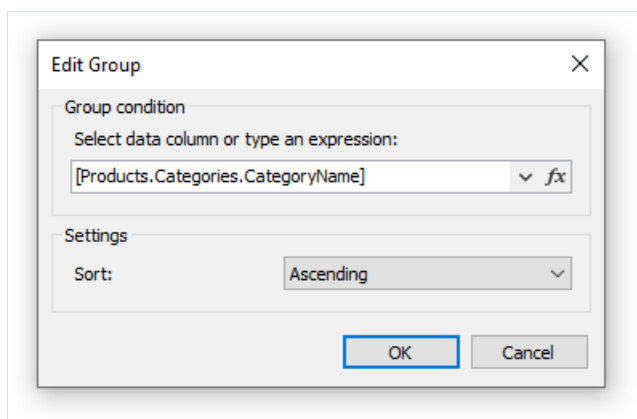
# Creating groups

Adding a group into a report can be done by using two methods.

First method: you add the "Group header" band in the "Configure Bands" window. To do this, press the "Add" button and select the "Group header" band. FastReport adds the group to the available "Data" band or will create a whole group, if such a band is not in the report:



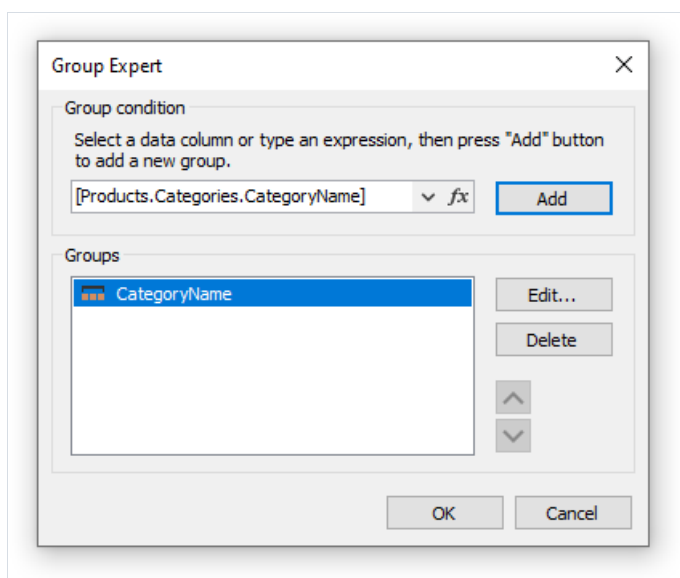
In order to configure a group, double click the "Group header" band. You will see the group header editor window:



You need to set the group condition. This can be any expression or data source column. Also choose the sorting. By default, data is sorted in ascending order.

Second method: you use the wizard, which can be called from the "Report|Group Expert..." menu. In order to create a group, enter the group condition and press the "Add" button:





The wizard will add all the elements of the group into the report. Also it creates the "Text" object on the group header, in which the group condition is printed:



# Sorting the data

For correct working of the group, it is needed to fulfill the following condition: data source must be sorted on that column which is used in the group condition. If this condition is not fulfilled, you will see a lot of groups containing 1-2 data rows:

|                                 |
|---------------------------------|
| <b>Beverages</b>                |
| Sasquatch Ale                   |
| Steeleye Stout                  |
| <b>Seafood</b>                  |
| Inlagd Sill                     |
| Gravad lax                      |
| <b>Beverages</b>                |
| Côte de Blaye                   |
| Chartreuse verte                |
| <b>Seafood</b>                  |
| Boston Crab Meat                |
| Jack's New England Clam Chowder |

Fortunately, there is a possibility of sorting the data source in two ways.

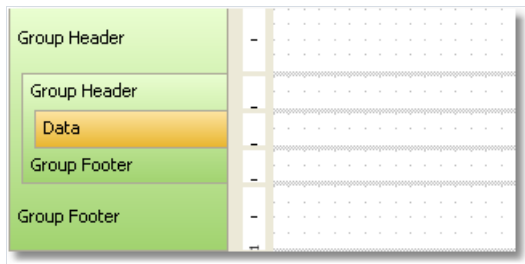
- you can set up the data sorting in the group editor. Data source will be automatically sorted on group condition;
- you can set up the sorting in the "Data" band editor.

Both methods are equivalent, however it is comfortable to use the first method. When creating groups, you set up the data grouping and sorting in one dialog.

In certain situations, the first method should not be used. Assuming that, we will set grouping on the first letter of the product's name. In such case, the product will be sorted only on the first letter, which is not acceptable. You should use the second method and indicate sorting on the full name of the product.

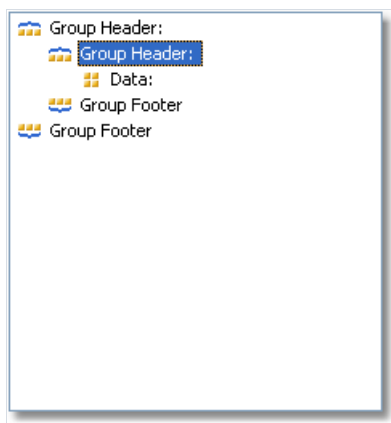
# Nested groups

A nested group has several "Group header" bands. The last band contains the "Data" band:



Every group header has its own group condition.

Creating a nested group can be done in the same way like an ordinary. In the first case, you create a simple group and add the nested group in the "Configure Bands" window. For this, select the existing "Group header" band, press the "Add" button and add another "Group header" band:



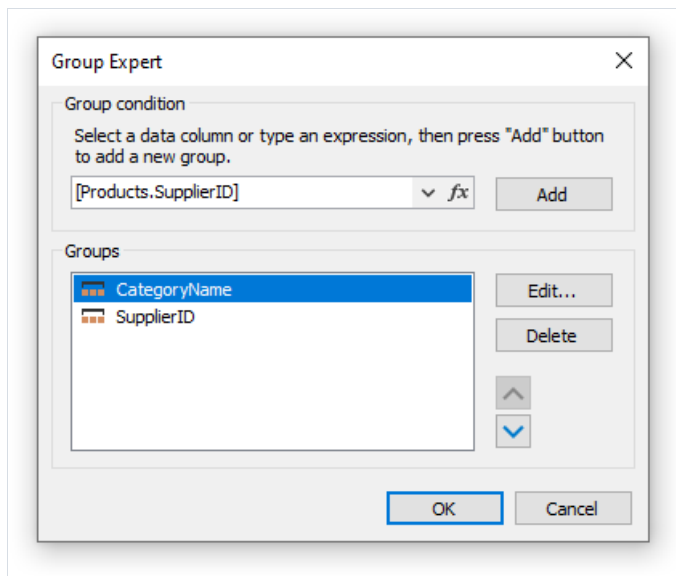
After this, call the editor of the added group and set up the group condition.



In the second case, you use the group expert which we have looked at already. Set the grouping condition and click the "Add" button. The wizard will add the new group to the existing one.

Printing of the nested group does not differ much from printing of an ordinary group. When printing data, FastReport will check all group conditions of all groups. If the condition changes, the corresponding group finishes and a new group starts printing.

# Managing groups

For managing groups, the group expert can be used. It can be called from the "Report|Group Expert..." menu:



With the help of the wizard you can either add or delete a group, and change the grouping order as well. For changing the grouping order, the buttons  and  are used. With the help of the "Edit..." button, you can change the group condition of the chosen group.

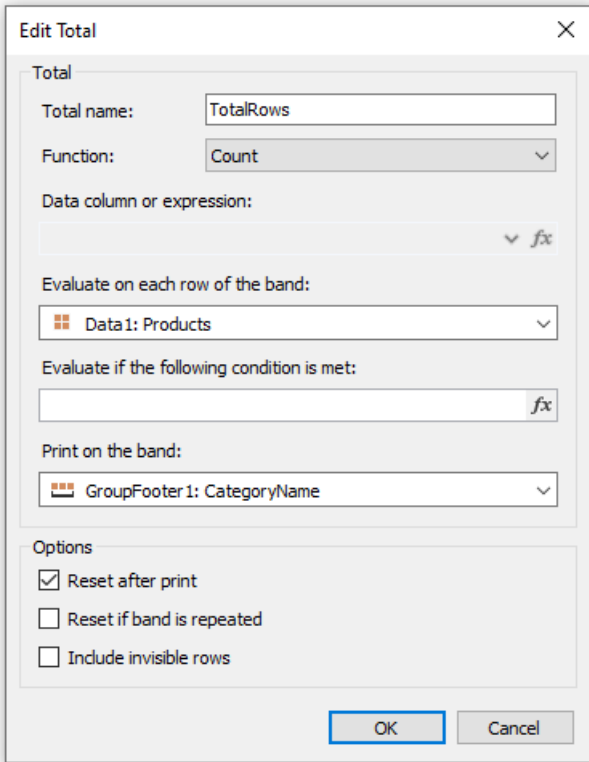
# Printing total values

Grouping is often used for printing some total values in every group. For example, this can be the number of rows in the group, or sum from one of the data columns. For printing such values, the total is used. The use of totals is described in the "Data" chapter.

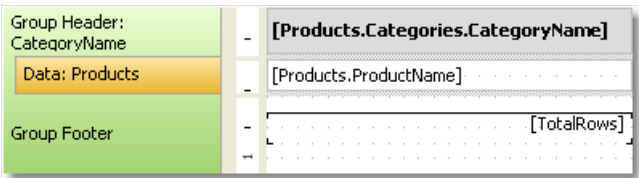
In order to print the total value in the group, you need to do the following:

- create a total, by selecting the "Actions|New total..." item in the "Data" window;
- choose the group's data band in the "Evaluate on each row of the band" combobox;
- choose the group footer in the "Print on the band" combobox;
- place the "Text" object, which prints the total value, on the group footer.

For example, for printing the number of rows in every group, configure the total as follows:



In order to display the value of the total, drag it onto the group footer:



Prepared report will be like this:

**Meat/Poultry**

Alice Mutton

Perth Pasties

Thüringer Rostbratwurst

Pâté chinois

Tourtière

Mishi Kobe Niku

6

**Produce**

Rössle Sauerkraut

Uncle Bob's Organic Dried Pears

Manjimup Dried Apples

Longlife Tofu

Tofu

5

# Repeating the header and footer

Group header and footer have the `RepeatOnEveryPage` property. It can be useful if the group does not fit on one page of a prepared report. By using this property, you can print the group header/footer on each page, where the group is printed. When printing such header/footer, FastReport sets its "Repeated" flag. This can be used for printing different objects on an ordinary group header and on repeat, for example, printing the "continue..." text on a new page. For this, use the `PrintOn` property of the "Text" object (see more details in the "Booklet" report type section).

In order to print different texts, place two objects on the group header, one on top of the other:

- the first object will be printed on ordinary headers. Set its `PrintOn` property value to `FirstPage, LastPage, OddPages, EvenPages` (that is all values except the `RepeatedBand`);
- the second object will be printed only on the headers which are repeated. Set its `PrintOn` property to `RepeatedBand`. Add the "Text" object with "continue..." text on the header.

The report will be printed as follows:

|              |                   |
|--------------|-------------------|
| Group header | Group - continued |
| Data         | Data              |
| Data         | Data              |
| Data         | Data              |
| Group footer | Data              |
| Group header | Data              |
| Data         | Data              |
| Data         | Data              |
| Data         | Group footer      |

Group footer can also be repeated on every page:

|                          |       |
|--------------------------|-------|
| <b>R</b>                 |       |
| Raclette Courdavault     | 55,00 |
| Ravioli Angelo           | 19,50 |
| Rhönbräu Klosterbier     | 7,75  |
| Röd Kaviar               | 15,00 |
| Røgede sild              | 9,50  |
|                          |       |
| <b>R</b>                 |       |
| Rössle Sauerkraut        | 45,60 |
| <b>Total products: 6</b> |       |

In this report, the group footer has two objects, placed one on top of the other:





# Group properties

The "Group Header" band has some useful properties.

The `StartNewPage` property allows forming a new page before printing the group. As a result, each group will be placed on a new page.

The new page will not be added before the first group. This is done in order to avoid an empty first page.

The `ResetPageNumber` property allows resetting the page number when printing the group. Usually it is used together with the `StartNewPage` property. As a result, if both of these properties are enabled, every group will be printed on a new page, and will have its own page numeration.

# Subreports

Sometimes at a certain place of the main report, extra data needs to be shown, this can be a separate report with a very complicated structure. You can try to solve this task by using FastReport's rich collection of bands. However, in certain cases, it is preferable to use the "Subreport" object.

The "Subreport" object is an ordinary report object, which can be placed on one of the bands. When you do this, FastReport adds an extra page into the report and connects it with the subreport. On this page it is possible to create an extra report, having any structure.

When printing the report, in which there is the "Subreport" object, the following will be done:

1. the main report will be printed, while the "Subreport" object is not met;
2. the subreport bands will be printed;
3. printing of the main report will continue.

Since subreport is formed on the sheet of the main report, it cannot contain the following bands: "Report header/footer", "Page header/footer", "Column header/footer", "Overlay".

# Printing modes

Subreport can be printed in two modes.

In the first printing mode, bands and objects of the subreport are printed on the page of the main report. There are some limitations:

- "Subreport" object must be located in lower border of the band;
- Never place other objects under the "Subreport" object. When the report will be working, such objects will be overlapping with the objects of the subreport:



To place other objects under the subreport, use the "Child" band. Place the objects in the following way:



The second printing mode differs in that subreport's objects are printed on the band, which contains the "Subreport" object. You can enable this mode from the context menu of the "Subreport" object. To do this, select the "Print on Parent" item. This mode does not put a limit on placing of the objects. Apart from that, in this mode, a parent band can either grow or shrink depending on how much data has been printed in the subreport.

The only problem with the second mode is that there may be a lot of data in the subreport. When printing it, the parent band will have a big height. In order to print such a band correctly, it is required to break its contents ( `CanBreak` property). The break algorithm does not provide 100% quality and in some cases it can lead to the displacement of objects.

## Side-by-side subreports

By placing two "Subreport" objects side-by-side on the same band, you can print two independent data lists. When printing such a report, FastReport acts in the following way:

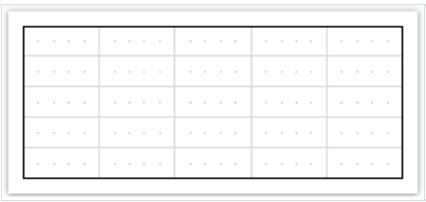
- prints the main report until, the "Subreport" object is not met;
- prints the first subreport;
- turn to the page where the printing of subreport started from, and prints the following subreport;
- after all the subreports have been printed, the main report continues to be printed from where the longest subreport has stopped.

# Nested subreports

On the page of a subreport, you can place another "Subreport" object, so the subreport becomes nested. The number of nesting levels formally is not restricted; however do not get carried away by that. Multiple nesting is very difficult to understand. If you have the possibility, use data bands for printing nested data. A "Data" band can have one or several nested "Data" bands. If you need to print a report of the master-detail type or master-detail-subdetail, there is no need to use the subreport.

# Table-type reports

The "Table" object is made up of rows, columns and cells. It is a simplified analog of Microsoft Excel table. It looks in the following way:

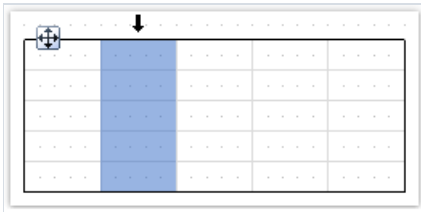


|   |   |   |   |   |
|---|---|---|---|---|
| . | . | . | . | . |
| . | . | . | . | . |
| . | . | . | . | . |
| . | . | . | . | . |
| . | . | . | . | . |

# Configuring columns

You can add or delete columns with the help of the context menu. For that:

- select the table or any of its elements and place the cursor on the needed column. The cursor's form changes to a small black arrow:



- left click, in order to select the column;
- right click in order to show the column's context menu;
- if you need to select several adjacent columns, left click and, without leaving it, move the mouse to the right or to the left, in order to select adjacent columns.

Column's context menu can also be called in the "Report Tree" window. Open the window, select the needed column and right click the mouse.

# Managing the size of the column

You can set up the width of the column by using one of the following methods:

- select the table or any of its elements and place the cursor on the border between two columns. The form of a cursor changes into a horizontal splitter:



- select a column and indicate the needed width in the `Width` properties. This property is accessible in the "Properties" window.

You can also enable the `AutoSize` column property. When running the report, the width of the column will be calculated automatically. In order to limit the width of the column, you can indicate the `MinWidth` and `MaxWidth` properties.

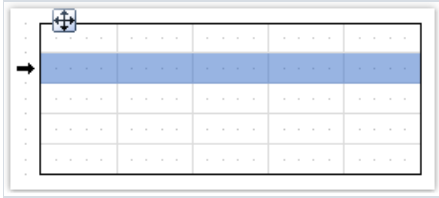
Width of the column should never be larger than the page's width.



# Configuring rows

Rows are configured in the same way. In order to select a row, do the following:

- select the table or any of its elements and place the cursor to the left of the needed row. The cursor's form changes to a small black arrow:



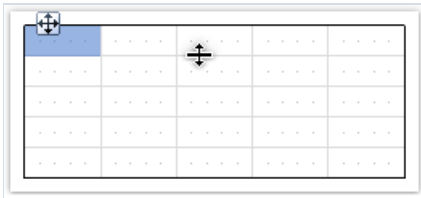
- left click the mouse, in order to select the row;
- right click the mouse in order to show the row's context menu.

If you need to select several adjacent rows, then left click the mouse and, without leaving it, move the mouse to right or to the left, so as to select the adjacent rows.

# Managing the size of the row

You can set up the height of the row by using one of the following methods:

- select the table or any of its elements and place the cursor on the border between two rows. The cursor's form changes into a vertical splitter:



Left click and move the mouse, in order to change the size of the row.

- select the row and indicate the needed height in the `Height` property. This property is accessible in the "Properties" windows.

You can also enable the row's `AutoSize` property. When the report is run, the height of the row will be calculated automatically. In order to limit the height of the row, you can use the minimum height ( `MinHeight` ) and Maximum height ( `MaxHeight` ) properties:

Height of the row should never be larger than the page's height.

# Configuring cells

Cells are text objects. In essence, the cells class is inherited from the "Text" object. Everything which has been said above about the "Text" object applies to the table's cells as well.

Editing the cells' text can be done just like the "Text" object. Besides, you can drag and drop an element from the "Data" window into the cells.

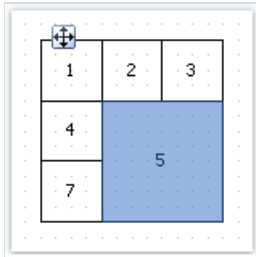
Border and fill of a cell can be configured with the help of the "Border and Fill" toolbar.

In order to call a cell's context menu, just right click on the cell.

# Joining and splitting cells

You can join adjacent cells of the table. As a result, there will be one big cell. In order to do this:

- select the first cell with the help of a mouse;
- left click and, without leaving it, move the mouse in order to select the group of cells;
- on the selected region, right click the mouse in order to show the context menu of the cells;
- in the context menu, choose the "Join cells" item.



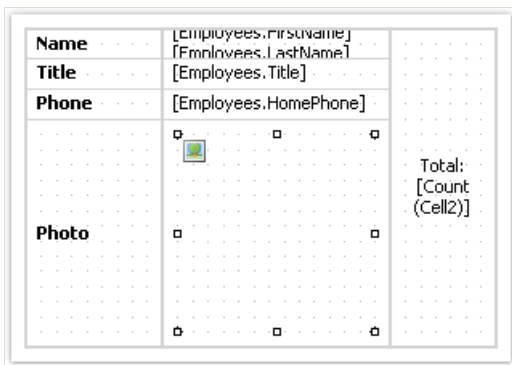
In order to split a cell, call its context menu and choose the "Split cell" item.

# Inserting objects in cells

In the cells, you can insert other objects, for example, the picture. The following objects can never be added into the cells:

- "Table";
- "Matrix";
- "Subreport".

In order to add an object into the cell, simply drag it inside the cell. You can freely move an object between cells, and also take it back beyond the table's boundary.



A cell serves as a container to objects placed into it. This means that, you can use the **Dock** and **Anchor** properties of an object inside the cell. This allows changing the size of the object when the size of the cell is changing.

# Printing a table

A table can be printed in two modes:

In the first mode, the table is printed inside the band which it belongs to, and looks just the same as in the designer. In this mode, the table does not split across pages if its width is bigger than the width of the report page. This is the mode of printing by default.

The second mode is dynamic. In this mode, the table is built with the help of script. During this, the resulting table can be different from the initial table, just like the prepared report of FastReport differs from the report template. In dynamic mode, the table can split across pages if it does not fit on the report page.

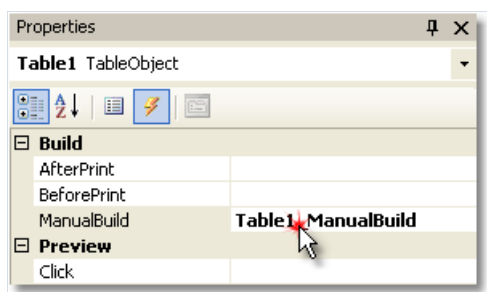
In the dynamic mode, the table does not get printed on the band on which it was placed. Instead of this, the table itself generates a set of bands, which contain parts of the resulting table. This mode of work imposes the following limits:

- never put other objects under the table or near it. Instead of this, use the "Child" band;
- never put two "Table" objects on one band.

Let us look at the dynamic mode in details.

This mode is connected with programming and needs higher qualifications from the report developer.

Formation of the table is done with the help of script. In order to create a script, select the "Table" object, in the "Properties" window click the "Events" button and double click on **ManualBuild** event:



When this is done, an empty event handler is added into the report code:

```
private void Table1_ManualBuild(object sender, EventArgs e)
{
}
```

In this mode, the source table is used as a template. In the event code, you can print rows and columns from the source table as many times as it is needed. During this, the resulting table will be formed, which can contain an unlimited number of rows and columns. Such a table can split across pages if it does not fit on the report page.

For printing a table, the following methods of the "Table" object are used:

| Method          | Parameters | Description   |
|-----------------|------------|---|
| <b>PrintRow</b> | int index  | Prints the row with the specified index. Row numbering starts from 0. |

| Method              | Parameters    | Description   |
|---------------------|---------------|---|
| <b>PrintColumn</b>  | int index     | Prints the column with the specified index. Column numbering starts from 0. |
| <b>PrintRows</b>    | int[] indices | Prints several rows of the table.   |
| <b>PrintRows</b>    | -             | Prints all rows of the table.   |
| <b>PrintColumns</b> | int[] indices | Prints several columns of the table.  |
| <b>PrintColumns</b> | -             | Prints all columns of the table.  |
| <b>PageBreak</b>    | -             | Inserts a page break before printing the next column or row.                |

Printing a table can be done by using one of the following methods:

The first method - printing from top to bottom, then from left to right. This method fits better a table with a variable amount of rows. You must call the methods in the following order:

- `PrintRow(row index)` ;
- one or more calls of the `PrintColumn(column index)` or `PrintColumns(columns indices)` methods for printing the indicated columns;
- or one call of the `PrintColumns()` method for printing all columns;
- repeat this sequence in order to print all the needed rows of the table.

Every row of the table must contain the same number of columns. Keep it in mind, when using the `PrintColumn(int index)` and `PrintColumns(int [] indices)` methods.

The second method - printing from left to right, then from top to bottom. This method is better for printing a table with a variable number of columns. You must call the methods in the following sequence:

- `PrintColumn(column index)` ;
- one or several calls of the `PrintRow(row number)` or `PrintRows(rows indices)` for printing the indicated rows;
- or one call of the `PrintRows()` method for printing all rows;
- repeat this sequence in order to print all the needed columns of the table.

Every column of the table must contain the same number of rows. Keep it in mind, when using the `PrintRow(int index)` and `PrintRows(int[] indices)` methods.

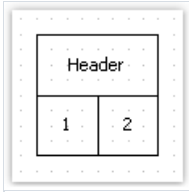
Violation of the order of calling the printing methods leads to errors when executing the report. One of the errors - attempting to print the table with the help of the following code:

```
Table1.PrintRows();
Table1.PrintColumns();
```

This sequence of methods is not correct. You must start printing the table with either the `PrintRow` or `PrintColumn` method.

# Printing complex headers

Here we are talking about headers which contain spanned cells. When printing rows or columns of the table, which has got spanned cells, the cells automatically increase in size. We will show this in the next example:



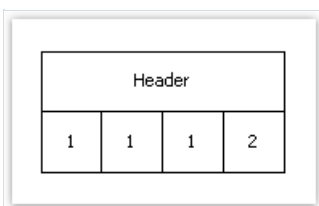
| Header |   |
|--------|---|
| 1      | 2 |

We will create a `ManualBuild` event handler, which will be printing the first column 3 times and the second column 1 time:

```
private void Table1_ManualBuild(object sender, EventArgs e)
{
    // printing row 1 and columns 0, 0, 0, 1
    Table1.PrintRow(0);
    Table1.PrintColumn(0);
    Table1.PrintColumn(0);
    Table1.PrintColumn(0);
    Table1.PrintColumn(1);
    // printing row 1 and columns 0, 0, 0, 1
    Table1.PrintRow(1);
    Table1.PrintColumn(0);
    Table1.PrintColumn(0);
    Table1.PrintColumn(0);
    Table1.PrintColumn(1);
}
```

Pay attention that we have printed the same number of columns in every row. If this rule is violated, then we will get an unpredictable result.

As a result of executing this code, we will get the following:



| Header |   |   |   |
|--------|---|---|---|
| 1      | 1 | 1 | 2 |

As seen, the header cell is spanned automatically. We will make the code a little more complex, so that two groups of columns can be printed:



```

private void Table1_ManualBuild(object sender, EventArgs e)
{
    // print 0 row and two groups of 0, 0, 0, 1 columns
    Table1.PrintRow(0);
    // group 1
    Table1.PrintColumn(0);
    Table1.PrintColumn(0);
    Table1.PrintColumn(0);
    Table1.PrintColumn(1);
    // group 2
    Table1.PrintColumn(0);
    Table1.PrintColumn(0);
    Table1.PrintColumn(0);
    Table1.PrintColumn(1);

    // print 1 row and two groups of 0, 0, 0, 1 columns
    Table1.PrintRow(1);
    // group 1
    Table1.PrintColumn(0);
    Table1.PrintColumn(0);
    Table1.PrintColumn(0);
    Table1.PrintColumn(1);
    // group 2
    Table1.PrintColumn(0);
    Table1.PrintColumn(0);
    Table1.PrintColumn(0);
    Table1.PrintColumn(1);
}

```

When we run the report, we will see the following result:

| Header |   |   |   | Header |   |   |   |
|--------|---|---|---|--------|---|---|---|
| 1      | 1 | 1 | 2 | 1      | 1 | 1 | 2 |

When printing the second column with the following code:

```
Table1.PrintColumn(1);
```

the header is finished and further printing of the first column starts a new header:

```

// group 2
Table1.PrintColumn(0);

```

# Using totals

In the dynamic mode of the "Table" object, the following total functions are supported:

| Function     | Parameters     | Description   |
|--------------|----------------|---|
| <b>Sum</b>   | TableCell cell | Returns the sum of the values contained in cell.          |
| <b>Min</b>   | TableCell cell | Returns the minimum from the values contained in a cell.  |
| <b>Max</b>   | TableCell cell | Returns the maximum from the values contained in a cell.  |
| <b>Avg</b>   | TableCell cell | Returns the average of the values contained in a cell.    |
| <b>Count</b> | TableCell cell | Returns the number of rows, contained the specified cell. |

In an ordinary printing mode (not dynamic) these functions will not work.

In order to use the total function, place it in the cell of the table. For example, the following function calculates the sum of the values contained in a cell named "Cell1":

```
[Sum(Cell1)]
```

During this, all the cells located higher and on the left of the current cell (in which we are calculating the sum) are analyzed.

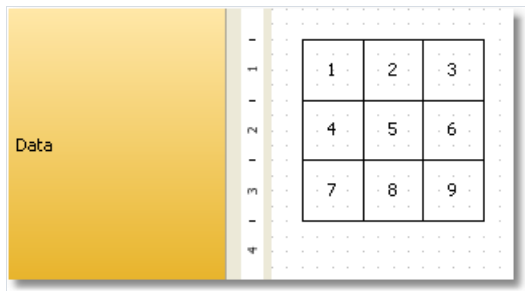
# Table layout

The table which is built dynamically, can be automatically splitted across pages. This behavior is controlled by the `Layout` property of the table. You can choose one of the following values:

| Value                 | Description  |
|-----------------------|--|
| <b>AcrossThenDown</b> | Table is rendered across then down.                  |
| <b>DownThenAcross</b> | Table is rendered down then across.                  |
| <b>Wrapped</b>        | Wide table is wrapped and rendered on the same page. |

# Examples

Let us look at printing tables for example. As a template, we will use the following report:



|   |   |   |
|---|---|---|
| 1 | 2 | 3 |
| 4 | 5 | 6 |
| 7 | 8 | 9 |

To start, select the table and create an event handler for `ManualBuild` event.

## Example 1. Printing the whole table from top to bottom

```
private void Table1_ManualBuild(object sender, EventArgs e)
{
    // printing row 0 and all of its columns
    Table1.PrintRow(0);
    Table1.PrintColumns();
    // printing row 1 and all of its columns
    Table1.PrintRow(1);
    Table1.PrintColumns();
    // Printing row 2 and all of its columns
    Table1.PrintRow(2);
    Table1.PrintColumns();
}
```

As a result, the following table will be printed, which does not differ from the template:

|   |   |   |
|---|---|---|
| 1 | 2 | 3 |
| 4 | 5 | 6 |
| 7 | 8 | 9 |

## Example 2. Printing the table from top to bottom with a repeating row

```
private void Table1_ManualBuild(object sender, EventArgs e)
{
    // Printing row 0 and all of its columns
    Table1.PrintRow(0);
    Table1.PrintColumns();
    // Printing 3 copies of row 1 and all of its columns
    for (int i = 0; i < 3; i++)
    {
        Table1.PrintRow(1);
        Table1.PrintColumns();
    }
    // printing row 2 and all of its columns
    Table1.PrintRow(2);
    Table1.PrintColumns();
}
```

In this example, the middle row gets printed 3 times. As a result, we get the following:

|   |   |   |
|---|---|---|
| 1 | 2 | 3 |
| 4 | 5 | 6 |
| 4 | 5 | 6 |
| 4 | 5 | 6 |
| 7 | 8 | 9 |

## Example 3. Printing the whole table from left to right

```
private void Table1_ManualBuild(object sender, EventArgs e)
{
    // printing column 0 and all of its rows
    Table1.PrintColumn(0);
    Table1.PrintRows();
    // printing column 1 and all of its rows
    Table1.PrintColumn(1);
    Table1.PrintRows();
    // printing column 2 and all of its rows
    Table1.PrintColumn(2);
    Table1.PrintRows();
}
```

As a result, the following table will be printed, which does not differ from the template:

|   |   |   |
|---|---|---|
| 1 | 2 | 3 |
| 4 | 5 | 6 |
| 7 | 8 | 9 |

## Example 4. Printing a table from left to right with a repeating column

```
private void Table1_ManualBuild(object sender, EventArgs e)
{
    // printing column 0 and all of its rows
    Table1.PrintColumn(0);
    Table1.PrintRows();
    // printing 3 copies of column 1 and all of its rows
    for (int i = 0; i < 3; i++)
    {
        Table1.PrintColumn(1);
        Table1.PrintRows();
    }
    // printing column 2 and all of its rows
    Table1.PrintColumn(2);
    Table1.PrintRows();
}
```

In this example, the middle column of the table gets printed 3 times. As a result, we get the following:

|   |   |   |   |   |
|---|---|---|---|---|
| 1 | 2 | 2 | 2 | 3 |
| 4 | 5 | 5 | 5 | 6 |
| 7 | 8 | 8 | 8 | 9 |



## Example 5. Printing a table with repeating rows and columns

```
private void Table1_ManualBuild(object sender, EventArgs e)
{
    // ----- printing row 0
    Table1.PrintRow(0);
    // printing column 0
    Table1.PrintColumn(0);
    // printing 3 copies of column 1
    for (int i = 0; i < 3; i++)
        Table1.PrintColumn(1);
    // printing column 2
    Table1.PrintColumn(2);

    // ----- printing 3 copies of row 1
    for (int j = 0; j < 3; j++)
    {
        Table1.PrintRow(1);
        // printing column 0
        Table1.PrintColumn(0);
        // printing 3 copies of column 1
        for (int i = 0; i < 3; i++)
            Table1.PrintColumn(1);
        // printing column 2
        Table1.PrintColumn(2);
    }
    // ----- printing row 2
    Table1.PrintRow(2);
    // printing column 0
    Table1.PrintColumn(0);
    // printing 3 copies of column 1
    for (int i = 0; i < 3; i++)
        Table1.PrintColumn(1);
    // printing column 2
    Table1.PrintColumn(2);
}
```

Pay attention that we printed the same number of columns in every row. If this rule is violated, then we will get an unpredictable result.

In this example, the middle row and middle column of the table were printed 3 times. And as a result, we get the following:

|   |   |   |   |   |
|---|---|---|---|---|
| 1 | 2 | 2 | 2 | 3 |
| 4 | 5 | 5 | 5 | 6 |
| 4 | 5 | 5 | 5 | 6 |
| 4 | 5 | 5 | 5 | 6 |
| 7 | 8 | 8 | 8 | 9 |

## Example 6. Using the data source

In all the examples considered, we printed a table, which contains an ordinary text. In this example we will show how to form a table using data source. For this, we will create a table having the following form:

| Product Name           | Unit Price           | Units In Stock          |
|------------------------|----------------------|-------------------------|
| [Products.ProductName] | [Products.UnitPrice] | [Products.UnitsInStock] |
|                        |                      |                         |

We will create the `ManualBuild` event handler, which will be doing the following:

- get the data source, defined in the report;
- initialize it (fill it with data);
- print the table's rows as many times as there are rows in the data source.

Here is the code of the handler:

```
private void Table1_ManualBuild(object sender, EventArgs e)
{
    // get the data source by its name
    DataSourceBase rowData = Report.GetDataSource("Products");
    // initialize it
    rowData.Init();

    // printing the table header
    Table1.PrintRow(0);
    Table1.PrintColumns();

    // loop through the data source rows
    while (rowData.HasMoreRows)
    {
        // printing the table row
        Table1.PrintRow(1);
        Table1.PrintColumns();

        // select the next data row
        rowData.Next();
    }

    // printing the table footer
    Table1.PrintRow(2);
    Table1.PrintColumns();
}
```

If we run the report, we will get the following:

| Product Name                    | Unit Price | Units In Stock |
|---------------------------------|------------|----------------|
| Chai                            | 18,00      | 39             |
| Chang                           | 19,00      | 17             |
| Aniseed Syrup                   | 10,00      | 13             |
| Chef Anton's Cajun Seasoning    | 22,00      | 53             |
| Chef Anton's Gumbo Mix          | 21,35      | 0              |
| Grandma's Boysenberry Spread    | 25,00      | 120            |
| Uncle Bob's Organic Dried Pears | 30,00      | 15             |
| Northwoods Cranberry Sauce      | 40,00      | 6              |



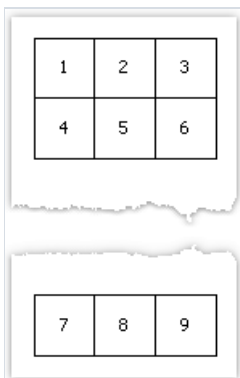
## Example 7. Inserting page breaks

Using the `PageBreak` method of the "Table" object, you can insert a page break when printing a table. Call it before you print a row/column.

We will use the Example 1 to demonstrate how the `PageBreak` method works. Let us print the third row on a new page.

```
private void Table1_ManualBuild(object sender, EventArgs e)
{
    // print the row 0 with all its columns
    Table1.PrintRow(0);
    Table1.PrintColumns();
    // print the row 1 with all its columns
    Table1.PrintRow(1);
    Table1.PrintColumns();
    // insert page break before the row 2
    Table1.PageBreak();
    // print the row 2 with all its columns
    Table1.PrintRow(2);
    Table1.PrintColumns();
}
```

As a result, we get the following:



|   |   |   |
|---|---|---|
| 1 | 2 | 3 |
| 4 | 5 | 6 |

|   |   |   |
|---|---|---|
| 7 | 8 | 9 |
|---|---|---|

## Example 8. Printing totals

We will look at the use of the total function in Example 6. We will modify it in the following way:

| Product Name           | Unit Price           | Units In Stock          |
|------------------------|----------------------|-------------------------|
| [Products.ProductName] | [Products.UnitPrice] | [Products.UnitsInStock] |
|                        | Total:               | [Sum(Cell8)]            |

The cell Cell8, which we will summarize

Summary value

If we run the report, we will get the following:

|                                 |       |      |
|---------------------------------|-------|------|
| Outback Lager                   | 15,00 | 15   |
| Fløtemysost                     | 21,50 | 26   |
| Mozzarella di Giovanni          | 34,80 | 14   |
| Rød Kaviar                      | 15,00 | 101  |
| Longlife Tofu                   | 10,00 | 4    |
| Rhönbräu Klosterbier            | 7,75  | 125  |
| Lakkalikööri                    | 18,00 | 57   |
| Original Frankfurter grüne Soße | 13,00 | 32   |
| Total:                          |       | 3119 |

# Matrix-type reports

The "Matrix" object is a variety table and like the "Table" object, is made up of rows, columns and cells. At the same time, it is not known before hand how many rows and columns will be in the matrix - this depends on the data to which it is connected.

The object looks like this:

| Employee | [Year]    | Total |
|----------|-----------|-------|
| [Name]   | [Revenue] |       |
| Total    |           |       |

When printing, the matrix fills up the values and grows up and down. The result can be as follows:

| Employee        | 1999        | 2000        | 2001        | 2002       | Total       |
|-----------------|-------------|-------------|-------------|------------|-------------|
| Andrew Fuller   | \$3,900.00  | \$2,100.00  |             | \$1,800.00 | \$7,800.00  |
| Janet Leverling | \$6,100.00  | \$3,200.00  |             |            | \$9,300.00  |
| Nancy Davolio   | \$3,300.00  | \$2,700.00  | \$3,100.00  | \$1,700.00 | \$10,800.00 |
| Steven Buchanan |             | \$3,999.00  | \$8,100.00  |            | \$12,099.00 |
| Total           | \$13,300.00 | \$11,999.00 | \$11,200.00 | \$3,500.00 | \$39,999.00 |

# A few theory

Let us look at the elements of a matrix:

|   | 1  | 2  | 3  | 4  |
|---|----|----|----|----|
| a | a1 | a2 | a3 | a4 |
| b | b1 | b2 | b3 | b4 |

In the figure, we see a matrix with 2 rows and 4 columns. Here a, b - row header, 1, 2, 3, 4 - column header, a1...a4, b1...b4 - cells. In order to build such a report, only one data source will be needed, which has got 3 columns and contains the following data:

```
a 1 a1
a 2 a2
a 3 a3
a 4 a4
b 1 b1
b 2 b2
b 3 b3
b 4 b4
```

As seen, the first column represents the matrix row, the second - matrix column, and the third - contents of the cells at the intersection of rows and columns with the indicated number. When creating a report, FastReport creates a matrix in the memory and fills it with data. During this, the matrix dynamically increases, if the row or column with the given number doesn't exist yet.

A header can have more than one level. Let us look at the following example:

|   | 10    |       | 20    |       |
|---|-------|-------|-------|-------|
|   | 1     | 2     | 1     | 2     |
| a | a10.1 | a10.2 | a20.1 | a20.2 |
| b | b10.1 | b10.2 | b20.1 | b20.2 |

In this example, a column is compound, that is, it has got two values. This report requires the following data:

```

a  10  1  a10.1
a  10  2  a10.2
a  20  1  a20.1
a  20  2  a20.2
b  10  1  b10.1
b  10  2  b10.2
b  20  1  b20.1
b  20  2  b20.2

```

Here, the first column represents the row, the second and the third represent the matrix column. The last data column contains the value of the cell.

The next matrix element - subtotal and grand total, the next figure demonstrates it:

|       | 10          |             |                         | 20          |             |                         | Total         |
|-------|-------------|-------------|-------------------------|-------------|-------------|-------------------------|---------------|
|       | 1           | 2           | Total                   | 1           | 2           | Total                   |               |
| a     | a10.1       | a10.2       | a10.1+a10.2             | a20.1       | a20.2       | a20.1+a20.2             | sum(a)        |
| b     | b10.1       | b10.2       | b10.1+b10.2             | b20.1       | b20.2       | b20.1+b20.2             | sum(b)        |
| Total | a10.1+b10.1 | a10.2+b10.2 | a10.1+b10.1+a10.2+b10.2 | a20.1+b20.1 | a20.2+b20.2 | a20.1+b20.1+a20.2+b20.2 | sum(a)+sum(b) |

This report is built on the same data used in the previous example. Rows that shown grey in the figure, are calculated automatically.

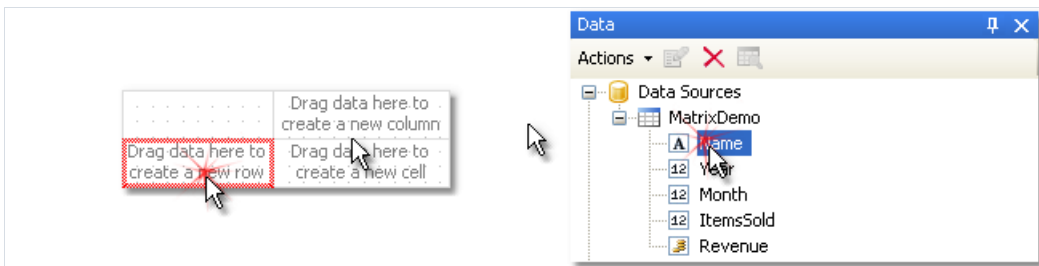


# Configuring the matrix

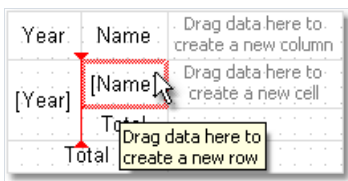
After you have placed a new "Matrix" object on a sheet, it will be as follows:



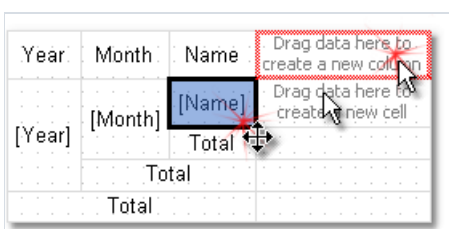
Matrix can be configured with the help of the mouse. To do this, drag and drop data source columns from the "Data" window onto the matrix, to create rows, columns and cells. The matrix highlights a red frame to a place where the new data will be placed:



If the matrix contains some elements already, then when placing a new element, an indicator will be shown. In the given case, the new data will be placed between the **Year** and **Name** elements:



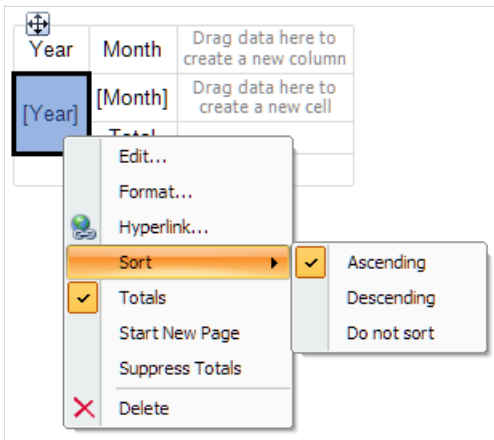
You can also change the order of the existing elements. To do this, click on the element's border (shown with black) and drag it to the needed place:



To delete an element, select it with the mouse, and press the Delete key.

# Configuring headers

To configure the header element, select it and right-click to display its context menu:



By default, data in the matrix header is sorted in ascending order. You can change the order of sorting, by selecting the "Sort" item.

Ordinarily, every item in the matrix header has a total (this is a cell with a "Total" text). You can delete the total, selecting it and pressing the `Delete` key. In order to enable total again, select an element to which it belongs, and choose the "Total" item in its context menu.

The "Start New Page" menu item tells the matrix to insert page breaks after printing each header value. For example, if you enable page breaks for the `Year` item (as shown in the picture above), every year value will be printed on its own page.

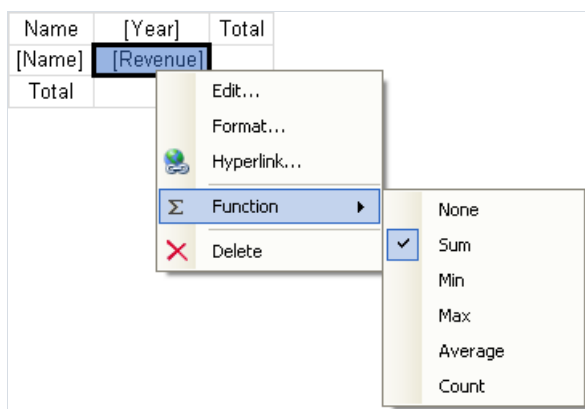
The "Suppress Totals" item allows to suppress totals in case when the group (on which the total value is calculated) contains only one value.

# Configuring cells

For a matrix cell, you can choose a function which will be used when calculating the total. A list of functions which can be used is given below:

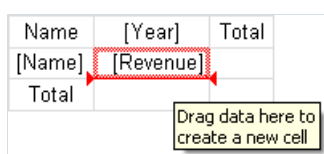
| Function      | Description                                       |
|---------------|---|
| None          | Cell value is not processed.                      |
| Sum           | Returns the sum of the values in the matrix cell. |
| Min           | Returns the minimum value.                        |
| Max           | Returns the maximum value.                        |
| Average       | Returns the average value.                        |
| Count         | Returns number of nonempty values.                |
| CountDistinct | Returns number of distinct values.                |

By default, the Sum function is used. You can change this by selecting a cell and choosing the "Function" item in its context menu:



Choose the "None" function, if you don't intend to print the total in the given cell.

In a matrix, there can be one or several data cells. In case the matrix has several cells, they can be arranged side-by-side or on top of each other. The `CellsSideBySide` property of the matrix controls how the cells are arranged. It can be changed from the context menu of the "Matrix" object. You can also choose the arrangement order when adding the second cell in the matrix. When doing this, look at the red indicator which shows where the second value will be placed:



After you have added the second value, the rest of the values will be added in the chosen order.

# Styling the matrix

In order to change the appearance of a matrix cell, click on the needed cell. With the help of the toolbar you can set up the font, border and fill. In order to change the appearance of several cells at once, select the group of cells. To do this, select the top left cell, and, without leaving the mouse, move the mouse so as to select the group:

A screenshot of a matrix table with a selection rectangle. The table has 4 columns and 4 rows. The first row contains 'Name', '[Year]', 'Total', and 'Total'. The second row contains '[Month]', 'Total', and an empty cell. The third row contains '[Name]', '[Revenue]', and an empty cell. The fourth row contains 'Total' and three empty cells. A blue selection rectangle covers the first three columns of the first three rows. A mouse cursor is positioned at the bottom-right corner of this selection area.

|         |           |       |       |
|---------|-----------|-------|-------|
| Name    | [Year]    | Total | Total |
| [Month] | Total     |       |       |
| [Name]  | [Revenue] |       |       |
| Total   |           |       |       |

You can use styles in order to change the appearance of the whole matrix. To do this, invoke the context menu of the "Matrix" object and choose the style:

# Row and column size management

Since the "Matrix" object is a kind of "Table" object, it allows setting row and column sizes in the same way.

By default, a matrix has got the `AutoSize` mode enabled. In this mode, the matrix calculates the column/row sizes automatically. You can as well manage the size of the object manually. To do this, disable the `AutoSize` property of the matrix. Rows and columns have the same property, you can use it if autosize of the matrix is disabled.

In order to limit the minimum and maximum width of a column, select a column and set up its `MinWidth` and `MaxWidth` properties.

In order to limit the minimum and maximum height of a row, select a row and set up its `MinHeight` and `MaxHeight` properties.

# Examples

Let us look at examples of using the "Matrix" object. For a start create a new report and put the "Matrix" object on the "Report Title" band. You can also use the "Data" band - in that case there is no need to connect the band to a data source. In the given case, it does not matter on which of the two bands you will put the matrix, since both bands will be printed once when the report is started. The report looking as follows:



Do not put "Matrix" object on bands which will be printed on every new page - "Page header", "Page footer", etc. The matrix in this case will be created every time when the band will be printed, which will lead to stack overflow.

Most examples will be using the "MatrixDemo" table, which is bundled with the FastReport package. This table contains the following data:

| Name            | Year | Month | ItemsSold | Revenue |
|-----------------|------|-------|-----------|---------|
| Nancy Davolio   | 1999 | 2     | 1         | 1000    |
| Nancy Davolio   | 1999 | 11    | 1         | 1100    |
| Nancy Davolio   | 1999 | 12    | 1         | 1200    |
| Nancy Davolio   | 2000 | 1     | 1         | 1300    |
| Nancy Davolio   | 2000 | 2     | 2         | 1400    |
| Nancy Davolio   | 2001 | 2     | 2         | 1500    |
| Nancy Davolio   | 2001 | 3     | 2         | 1600    |
| Nancy Davolio   | 2002 | 1     | 2         | 1700    |
| Andrew Fuller   | 2002 | 1     | 2         | 1800    |
| Andrew Fuller   | 1999 | 10    | 2         | 1900    |
| Andrew Fuller   | 1999 | 11    | 2         | 2000    |
| Andrew Fuller   | 2000 | 2     | 2         | 2100    |
| Janet Leverling | 1999 | 10    | 3         | 3000    |
| Janet Leverling | 1999 | 11    | 3         | 3100    |

| Name            | Year | Month | ItemsSold | Revenue |
|-----------------|------|-------|-----------|---------|
| Janet Leverling | 2000 | 3     | 3         | 3200    |
| Steven Buchanan | 2001 | 1     | 3         | 4000    |
| Steven Buchanan | 2001 | 2     | 4         | 4100    |
| Steven Buchanan | 2000 | 1     | 4         | 3999    |

# Example 1. Simple matrix

The matrix will contain one value in a row and a column as well as one data cell. In order to build a matrix you need to add "MatrixDemo" data columns in the following way:

- add **Year** data column to the row header;
- add **Name** data column to the column header;
- add **Revenue** data column to the matrix cell.

After that the matrix will look as follows:

| Year   | [Name]    | Total |
|--------|-----------|-------|
| [Year] | [Revenue] |       |
| Total  |           |       |

Let us improve the matrix appearance:

- choose "Orange" style for the matrix;
- choose "Tahoma, 8" font for all matrix cells;
- select the word "Total" with bold type;
- choose "Glass" type filling for the cells in upper row;
- disable the autosize of the matrix and increase the size of rows and columns.

After that the matrix will have the following view:

| Year   | [Name]    | Total |
|--------|-----------|-------|
| [Year] | [Revenue] |       |
| Total  |           |       |

Run the report and you will see the following result:

| Year  | Andrew Fuller | Janet Leverling | Nancy Davolio | Steven Buchanan | Total |
|-------|---------------|-----------------|---------------|-----------------|-------|
| 1999  | 3900          | 6100            | 3300          |                 | 13300 |
| 2000  | 2100          | 3200            | 2700          | 3999            | 11999 |
| 2001  |               |                 | 3100          | 8100            | 11200 |
| 2002  | 1800          |                 | 1700          |                 | 3500  |
| Total | 7800          | 9300            | 10800         | 12099           | 39999 |

There is lack of the following things in the matrix:

- there is no title for the **Name** data column;
- sums are not printed in currency format;

You can add a title for **Name** data column in the following way:

- the text "Year/Employee" can be put into the left upper corner of the matrix;
- diagonal line and the second "Text" object can be placed there, as shown below:

| Year/Employee | [Name]    | Total |
|---------------|-----------|-------|
| [Year]        | [Revenue] |       |
| Total         |           |       |

- enable the matrix title. To do this, choose the "Show Title" item in the context menu of the "Matrix" object. Any text can be included in the title:



| Employee |           |       |
|----------|-----------|-------|
| Year     | [Name]    | Total |
| [Year]   | [Revenue] |       |
| Total    |           |       |

In order to set the data formatting, select the whole cell area, as shown in the figure below, and set format by selecting the "Format..." item in the context menu:

| Employee |           |       |
|----------|-----------|-------|
| Year     | [Name]    | Total |
| [Year]   | [Revenue] |       |
| Total    |           |       |

After that, a prepared report will be as follows:

| Employee |               |                 |               |                 |             |
|----------|---------------|-----------------|---------------|-----------------|-------------|
| Year     | Andrew Fuller | Janet Leverling | Nancy Davolio | Steven Buchanan | Total       |
| 1999     | \$3,900.00    | \$6,100.00      | \$3,300.00    |                 | \$13,300.00 |
| 2000     | \$2,100.00    | \$3,200.00      | \$2,700.00    | \$3,999.00      | \$11,999.00 |
| 2001     |               |                 | \$3,100.00    | \$8,100.00      | \$11,200.00 |
| 2002     | \$1,800.00    |                 | \$1,700.00    |                 | \$3,500.00  |
| Total    | \$7,800.00    | \$9,300.00      | \$10,800.00   | \$12,099.00     | \$39,999.00 |

## Example 2. Multilevel headers

The matrix will have one value in a row, two values in a column and one data cell. We will get the previous example as a base and add a new item into it:

- we will add the **Month** data column to the row header to the right of **Year** item.

After adding a new item, improve the appearance of the matrix. It is also necessary to set up formatting for cells. After that, the matrix will be as follows:


|        |         | Employee  |       |
|--------|---------|-----------|-------|
| Year   | Month   | [Name]    | Total |
| [Year] | [Month] | [Revenue] |       |
|        | Total   |           |       |
| Total  |         |           |       |

Run the report and you will have the following result:

|       |       | Employee      |                 |               |                 |             |
|-------|-------|---------------|-----------------|---------------|-----------------|-------------|
| Year  | Month | Andrew Fuller | Janet Leverling | Nancy Davolio | Steven Buchanan | Total       |
| 1999  | 2     |               |                 | \$1,000.00    |                 | \$1,000.00  |
|       | 10    | \$1,900.00    | \$3,000.00      |               |                 | \$4,900.00  |
|       | 11    | \$2,000.00    | \$3,100.00      | \$1,100.00    |                 | \$6,200.00  |
|       | 12    |               |                 | \$1,200.00    |                 | \$1,200.00  |
|       | Total | \$3,900.00    | \$6,100.00      | \$3,300.00    | \$0.00          | \$13,300.00 |
| 2000  | 1     |               |                 | \$1,300.00    | \$3,999.00      | \$5,299.00  |
|       | 2     | \$2,100.00    |                 | \$1,400.00    |                 | \$3,500.00  |
|       | 3     |               | \$3,200.00      |               |                 | \$3,200.00  |
|       | Total | \$2,100.00    | \$3,200.00      | \$2,700.00    | \$3,999.00      | \$11,999.00 |
| 2001  | 1     |               |                 |               | \$4,000.00      | \$4,000.00  |
|       | 2     |               |                 | \$1,500.00    | \$4,100.00      | \$5,600.00  |
|       | 3     |               |                 | \$1,600.00    |                 | \$1,600.00  |
|       | Total | \$0.00        | \$0.00          | \$3,100.00    | \$8,100.00      | \$11,200.00 |
| 2002  | 1     | \$1,800.00    |                 | \$1,700.00    |                 | \$3,500.00  |
|       | Total | \$1,800.00    | \$0.00          | \$1,700.00    | \$0.00          | \$3,500.00  |
| Total |       | \$7,800.00    | \$9,300.00      | \$10,800.00   | \$12,099.00     | \$39,999.00 |

## Example 3. Printing the name of the month

In the previous example, there were numbers of months printed in a matrix. This occurred, because the `Month` data column contains the number of the month, not its name. In order to print the name of the month, do the following:

- choose a cell where the number of the month is printed. In our case it is a cell with a name "Cell8";
- in the "Property" window press the  button and double click `BeforePrint` event;
- FastReport will add an empty event handler in the report script. Write the following code:

```
private void Cell8_BeforePrint(object sender, EventArgs e)
{
    string[] month Names = newstring[] {
        "January", "February", "March", "April",
        "May", "June", "July", "August",
        "September", "October", "November", "December" };
    // Cell8 is a cell that prints the month number.
    // Cell8.Value is a value printed in the cell (i.e. the month number).
    // This value is of System.Object type, so we need to cast it to int
    Cell8.Text = monthNames[(int)Cell8.Value - 1];
}
```

When you run the report, you will have the following result:

| Year | Month        | Andrew Fuller | Janet Leverli |
|------|--------------|---------------|---------------|
| 1999 | February     |               |               |
|      | October      | \$1,900.00    | \$2,000.00    |
|      | November     | \$2,000.00    |               |
|      | December     |               |               |
|      | <b>Total</b> | \$3,900.00    |               |
| 2000 | January      |               |               |
|      | February     |               |               |
|      | March        |               |               |

## Example 4. Conditional highlighting

You can set conditional highlighting for matrix cells, just like for "Text" object. More details about this can be found in the ["Conditional highlighting"](#) section.

Let us look at Example 2, and see how to highlight an amount more than 3000 in red. For this, select the cell with

Revenue text and press the  button on "Text" toolbar. In the conditions editor, add the following condition:

```
Value > 3000
```

Choose red text color for the condition. A prepared report will be as follows:

|       |       | Employee      |                 |               |                 |             |
|-------|-------|---------------|-----------------|---------------|-----------------|-------------|
| Year  | Month | Andrew Fuller | Janet Leverling | Nancy Davolio | Steven Buchanan | Total       |
| 1999  | 2     |               |                 | \$1,000.00    |                 | \$1,000.00  |
|       | 10    | \$1,900.00    | \$3,000.00      |               |                 | \$4,900.00  |
|       | 11    | \$2,000.00    | \$3,100.00      | \$1,100.00    |                 | \$6,200.00  |
|       | 12    |               |                 | \$1,200.00    |                 | \$1,200.00  |
|       | Total | \$3,900.00    | \$6,100.00      | \$3,300.00    | \$0.00          | \$13,300.00 |
| 2000  | 1     |               |                 | \$1,300.00    | \$3,999.00      | \$5,299.00  |
|       | 2     | \$2,100.00    |                 | \$1,400.00    |                 | \$3,500.00  |
|       | 3     |               | \$3,200.00      |               |                 | \$3,200.00  |
|       | Total | \$2,100.00    | \$3,200.00      | \$2,700.00    | \$3,999.00      | \$11,999.00 |
| 2001  | 1     |               |                 |               | \$4,000.00      | \$4,000.00  |
|       | 2     |               |                 | \$1,500.00    | \$4,100.00      | \$5,600.00  |
|       | 3     |               |                 | \$1,600.00    |                 | \$1,600.00  |
|       | Total | \$0.00        | \$0.00          | \$3,100.00    | \$8,100.00      | \$11,200.00 |
| 2002  | 1     | \$1,800.00    |                 | \$1,700.00    |                 | \$3,500.00  |
|       | Total | \$1,800.00    | \$0.00          | \$1,700.00    | \$0.00          | \$3,500.00  |
| Total |       | \$7,800.00    | \$9,300.00      | \$10,800.00   | \$12,099.00     | \$39,999.00 |

As seen, total values are not highlighted. This occurred, because we chose highlight condition for only one cell. To highlight the rest of the values, it is needed to set up the highlight for all matrix cells.

In this example we used conditional highlight which depends on cell value itself. Besides, you can highlight a cell depending on values from matrix headers. We will show by the following example, how to highlight cells, which are belongs to 2000 year, in red. For this, select matrix cells as shown in the figure below:

|        |         | Employee  |       |
|--------|---------|-----------|-------|
| Year   | Month   | [Name]    | Total |
| [Year] | [Month] | [Revenue] |       |
|        | Total   |           |       |
| Total  |         |           |       |

Set the following highlight condition:

```
(int)Matrix1.RowValues[0] == 2000
```

In this case "Matrix1" is a name of our matrix. The `RowValues` property of the matrix has got an `object[]` type and contains an array of values from the row header of the current printed row. Number of values in array is equal to number of levels in a header. There are two values in our example, the first one is "Year" and the second one is "Month".

Do not highlight the last row. `RowValues` property has an undetermined value for it and will cause an error when building the report.

When we run the report, we will have the following result:

|       |       | Employee      |                 |               |                 |             |
|-------|-------|---------------|-----------------|---------------|-----------------|-------------|
| Year  | Month | Andrew Fuller | Janet Leverling | Nancy Davolio | Steven Buchanan | Total       |
| 1999  | 2     |               |                 | \$1,000.00    |                 | \$1,000.00  |
|       | 10    | \$1,900.00    | \$3,000.00      |               |                 | \$4,900.00  |
|       | 11    | \$2,000.00    | \$3,100.00      | \$1,100.00    |                 | \$6,200.00  |
|       | 12    |               |                 | \$1,200.00    |                 | \$1,200.00  |
|       | Total | \$3,900.00    | \$6,100.00      | \$3,300.00    | \$0.00          | \$13,300.00 |
| 2000  | 1     |               |                 | \$1,300.00    | \$3,999.00      | \$5,299.00  |
|       | 2     | \$2,100.00    |                 | \$1,400.00    |                 | \$3,500.00  |
|       | 3     |               | \$3,200.00      |               |                 | \$3,200.00  |
|       | Total | \$2,100.00    | \$3,200.00      | \$2,700.00    | \$3,999.00      | \$11,999.00 |
| 2001  | 1     |               |                 |               | \$4,000.00      | \$4,000.00  |
|       | 2     |               |                 | \$1,500.00    | \$4,100.00      | \$5,600.00  |
|       | 3     |               |                 | \$1,600.00    |                 | \$1,600.00  |
|       | Total | \$0.00        | \$0.00          | \$3,100.00    | \$8,100.00      | \$11,200.00 |
| 2002  | 1     | \$1,800.00    |                 | \$1,700.00    |                 | \$3,500.00  |
|       | Total | \$1,800.00    | \$0.00          | \$1,700.00    | \$0.00          | \$3,500.00  |
| Total |       | \$7,800.00    | \$9,300.00      | \$10,800.00   | \$12,099.00     | \$39,999.00 |

You can also use matrix's `ColumnValues` property for column values reference.

## Example 5. Highlighting even rows

To improve the appearance of a matrix, you can highlight even rows or columns with other color. We will use the Example 2 to show how to do it.

Select the whole area of matrix data as it is shown in the figure:

|        |         | Employee  |       |
|--------|---------|-----------|-------|
| Year   | Month   | [Name]    | Total |
| [Year] | [Month] | [Revenue] |       |
|        | Total   |           |       |
| Total  |         |           |       |

Call the conditional highlighting editor. Add the following condition:

```
Matrix1.RowIndex % 2 != 0
```

and choose background color a little darker than the previous one. In this example "Matrix1" is a name of our matrix. The `RowIndex` property of the matrix returns the number of the current printed line.

For column highlighting, use the matrix `ColumnIndex` property in the same way.

When we run the report, we will see the following:

|       |       | Employee      |                 |               |                 |             |
|-------|-------|---------------|-----------------|---------------|-----------------|-------------|
| Year  | Month | Andrew Fuller | Janet Leverling | Nancy Davolio | Steven Buchanan | Total       |
| 1999  | 2     |               |                 | \$1,000.00    |                 | \$1,000.00  |
|       | 10    | \$1,900.00    | \$3,000.00      |               |                 | \$4,900.00  |
|       | 11    | \$2,000.00    | \$3,100.00      | \$1,100.00    |                 | \$6,200.00  |
|       | 12    |               |                 | \$1,200.00    |                 | \$1,200.00  |
|       | Total | \$3,900.00    | \$6,100.00      | \$3,300.00    | \$0.00          | \$13,300.00 |
| 2000  | 1     |               |                 | \$1,300.00    | \$3,999.00      | \$5,299.00  |
|       | 2     | \$2,100.00    |                 | \$1,400.00    |                 | \$3,500.00  |
|       | 3     |               | \$3,200.00      |               |                 | \$3,200.00  |
|       | Total | \$2,100.00    | \$3,200.00      | \$2,700.00    | \$3,999.00      | \$11,999.00 |
| 2001  | 1     |               |                 |               | \$4,000.00      | \$4,000.00  |
|       | 2     |               |                 | \$1,500.00    | \$4,100.00      | \$5,600.00  |
|       | 3     |               |                 | \$1,600.00    |                 | \$1,600.00  |
|       | Total | \$0.00        | \$0.00          | \$3,100.00    | \$8,100.00      | \$11,200.00 |
| 2002  | 1     | \$1,800.00    |                 | \$1,700.00    |                 | \$3,500.00  |
|       | Total | \$1,800.00    | \$0.00          | \$1,700.00    | \$0.00          | \$3,500.00  |
| Total |       | \$7,800.00    | \$9,300.00      | \$10,800.00   | \$12,099.00     | \$39,999.00 |

## Example 6. Using Expressions

In previous examples we created matrix by dragging columns from the "Data" window. You can also use expressions for this purpose. In order to insert an expression into a matrix do the following:

- add any element from "Data" window into matrix. It can be any element, for example, a system variable `Date`
  - we just use it to create a matrix element;
- double click the element and select the needed expression in the expressions editor window.

If your matrix has expressions instead of data field, you have to check that the matrix `DataSource` property was set up correctly. When working with data columns, this property is filled automatically when you drag a column onto a matrix.

Let us consider an example on how to use expressions. For this, we will use the "Order Details" table as a data source, which contains a list of sold products, grouped by employees. There are several relations in this table, which gives an access to the name of an employee, product name and its category.

Our matrix will show each employee's sales, categorized by products. In order to build the matrix, do the following:

- add the `[Order Details.Products.Categories.CategoryName]` data column to the column header;
- add any item to the row header in order to create a matrix element. Then set the following expression for the header element:

```
[Order Details.Orders.Employees.FirstName] + " " + [Order Details.Orders.Employees.LastName]
```

- add any item to data cell in order to create a matrix element. Then set the following expression for the cell:

```
[Order Details.UnitPrice] * [Order Details.Quantity] * (decimal)(1 - [Order Details.Discount])
```

Why did we indicate such a long data column as a name of an employee though we could get a name from the `Employees.FirstName` ? We did that because the matrix is connected to "Order Details" data source. Using relations between this data source and other tables, it is easy to refer to its columns (more details about relations can be read in the "Data" chapter). If we refer directly to the `Employees.FirstName` data column, we will get a name of the first employee in a table.

Set up the matrix appearance. After that it will look as the follows:

| Employee   | [CategoryName]   | Total |
|--|--|-------|
| [Order Details.Orders.Employees.FirstName] + " " + [Order Details.Orders.Employees.LastName] | [[[Order Details.UnitPrice] * [Order Details.Quantity] * (decimal)(1 - [Order Details.Discount])]] |       |
| Total  |  |       |

When we run the report, we will see quite a big matrix which occupies 2 sheets:

| Employee         | Beverages | Condiments | Confections | Dairy Products | Grains/Cereals |
|------------------|-----------|------------|-------------|----------------|----------------|
| Andrew Fuller    | 40 248,25 | 14 850,67  | 21 455,69   | 23 812,55      | 11 111,11      |
| Anne Dodsworth   | 19 642,56 | 10 125,55  | 8 053,16    | 21 101,13      | 1 234,56       |
| Janet Leverling  | 44 757,41 | 13 381,64  | 33 622,40   | 32 320,84      | 23 456,78      |
| Laura Callahan   | 17 897,85 | 14 637,66  | 21 699,91   | 21 269,47      | 12 345,67      |
| Margaret Peacock | 50 308,21 | 23 314,87  | 27 768,73   | 35 987,65      | 12 345,67      |
| Michael Suyama   | 9 450,20  | 4 648,47   | 6 859,63    | 12 345,67      | 1 234,56       |
| Nancy Davolio    | 46 599,36 | 13 561,56  | 28 568,21   | 32 320,84      | 23 456,78      |
| Robert King      | 27 963,83 | 8 851,38   | 12 345,67   | 21 101,13      | 1 234,56       |
| Steven Buchanan  | 11 000,53 | 12 345,67  | 23 456,78   | 32 320,84      | 23 456,78      |



## Example 7. Pictures in cells

Matrix cells are inherited from the "Text" object and can display text data. If it is not enough, you can put any object into the cell. Let us see how to display a picture in a matrix.

For this we will take Example 6 as a basis. Let's add a photo of an employee ( `Employees.Photo` data column) and the category picture ( `Categories.Picture` data column). Do the following:



- select the cell containing the employee's name and increase its size;
- add the "Picture" object to this cell;
- in order to show an employee's photo, bind the "Picture" object to the following data column (this can be done in the object editor):

```
Order.Details.Orders.Employees.Photo
```

- select the cell containing the category name and increase its size;
- add the "Picture" object to this cell;
- in order to show a category picture, bind the "Picture" object to the following data column (this can be done in the object editor):

```
Order.Details.Products.Categories.Picture
```

After that the matrix will look as follows:

| [CategoryName] |   | Total  |  |
|----------------|---|--|--|
| Employee       |  | Total  |  |
| [Order]        |  | [[Order.Details.UnitPrice] * [Order.Details.Quantity] * (decimal)(1 - [Order.Details.Discount])] |  |
| Total          |   |  |  |

When we run the report, we will see the following:

| Employee        | Beverages | Condiments | Confections |
|-----------------|-----------|------------|-------------|
| Andrew Fuller   | 40 248,25 | 14 850,67  |             |
| Anne Dodsworth  | 19 642,56 | 10 125,6   |             |
| Janet Leverling | 44 700,00 |            |             |

## Example 8. Objects in cells

Using objects inserted into matrix cells you can have various visual effects. We will show in the following example, how to draw a simple scale indicating employee's sales level.

The matrix will use the "MatrixDemo" data source. To build a matrix, add data columns in the following way:

- add **Year** data column to the row header;
- add **Name** data column to the column header;
- add **Revenue** data column to the matrix cell.

Set the appearance of the matrix in the following way:

| Employee | [Year]    | Total |
|----------|-----------|-------|
| [Name]   | [Revenue] |       |
| Total    |           |       |

Now let us add three "Shape" objects to the cell with **Revenue** value. These objects will serve as indicators in the following way:

- if value in a cell is less than 100, only one object of red color will be shown;
- if value in a cell is less than 3000, two objects of yellow color will be shown;
- if value in a cell is more or equal to 3000, three objects of green color will be shown.

Now the matrix looks like this:

| Employee | [Year] | Total     |
|----------|--------|-----------|
| [Name]   | ■■■    | [Revenue] |
| Total    |        |           |

To control objects, we will use an event handler for a matrix cell. For this, select the **Revenue** cell, and create the **BeforePrint** event handler using the "Properties" window. Write the following code in the handler:

```

private void Cell4_BeforePrint(object sender, EventArgs e)
{
    // In our example, a cell has the Cell4 name.
    // Get cell value which is in the Cell4.Value property.
    // Some cells in our matrix will be empty. We'll take it into account (null check).
    // The value should be cast to decimal type, because data source column
    // [MatrixDemo.Revenue] is of System.Decimal type.
    decimal value = Cell4.Value == null ? 0 : (decimal)Cell4.Value;

    // Switch shape objects on or off depending on the value:
    // value < 100 - one object is visible;
    // value < 3000 - two objects are visible;
    // value >= 3000 - all objects are visible
    Shape1.Visible = true;
    Shape2.Visible = value >= 100;
    Shape3.Visible = value >= 3000;

    // Choose the color of objects:
    // value < 100 - red color;
    // value < 3000 - yellow color;
    // value >= 3000 - green color
    Color color = Color.Red;
    if (value >= 100)
        color = Color.Yellow;
    if (value >= 3000)
        color = Color.GreenYellow;

    // Set the objects' color
    Shape1.Fill = new SolidFill(color);
    Shape2.Fill = new SolidFill(color);
    Shape3.Fill = new SolidFill(color);
}

```

When we run the report, we will see the following:

| Employee        | 1999         | 2000         | 2001         | 2002        | Total     |
|-----------------|--------------|--------------|--------------|-------------|-----------|
| Andrew Fuller   | ■■■ 3 900,00 | ■■ 2 100,00  | ■            | ■■ 1 800,00 | 7 800,00  |
| Janet Leverling | ■■■ 6 100,00 | ■■■ 3 200,00 | ■            | ■           | 9 300,00  |
| Nancy Davolio   | ■■■ 3 300,00 | ■■ 2 700,00  | ■■■ 3 100,00 | ■■ 1 700,00 | 10 800,00 |
| Steven Buchanan | ■            | ■■■ 3 999,00 | ■■■ 8 100,00 | ■           | 12 099,00 |
| Total           | 13 300,00    | 11 999,00    | 11 200,00    | 3 500,00    | 39 999,00 |

## Example 9. Filling a matrix manually

In all the examples we have looked at, the matrix was filled with data automatically because it was connected to data source. Data source for the matrix is indicated in the `DataSource` property. Though we did not set the value of this property manually, it occurred implicitly while adding data columns to the matrix.

Using script it is possible to fill in the matrix manually. For this, it is needed to create the `ManualBuild` event handler of the matrix. Call `AddValue` method in the handler code to add a value. Let us show how to create a matrix which will print a 10x10 table of the following kind:


|     | 1 | 2 | 3 | ... |
|-----|---|---|---|-----|
| 1   | 1 |   |   |     |
| 2   |   | 2 |   |     |
| 3   |   |   | 3 |     |
| ... |   |   |   | ... |

Do the following:

- add an empty matrix into the report;
- put any element from the "Data" window into the row, column and cell of the matrix. Then call expression editor by double clicking the matrix element and clear an expression;
- clear the `DataSource` property of the matrix.

These steps are required to create a "dummy" matrix which has one row, column and cell. As a result the matrix will be as follows:

|       |   |       |
|-------|---|-------|
|       | □ | Total |
| □     | □ |       |
| Total |   |       |

Now create a `ManualBuild` event handler. For that, select the matrix, go "Properties" window and press the  button. Double click the `ManualBuild` event and FastReport will create an empty event handler. Write the following code in it:

```

private void Matrix1_ManualBuild(object sender, EventArgs e)
{
    // Our matrix has one level in row, column and cell.
    // Create 3 arrays of object[] type, each with one element
    // (per number of levels).
    object[] columnValues = new object[1];
    object[] rowValues = new object[1];
    object[] cellValues = new object[1];

    for (int i = 1; i <= 10; i++)
    {
        // Filling arrays
        columnValues[0] = i;
        rowValues[0] = i;
        cellValues[0] = i;

        // Adding data into the matrix
        Matrix1.AddValue(columnValues, rowValues, cellValues);
    }
}

```

In a handler, you should use the `AddValue` method of the "Matrix" object in order to fill it with data. This method has three parameters each of which is an array of `System.Object` type. The first parameter is a column value, the second one is the row value, and the third one is the cell value. Note that the number of values in every array should comply with the object's settings! In our case an object has one level in column, row and cell, correspondingly we supply one value for columns, one for rows and one for cells.

When we run the report, we will see the following:

|       | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | Total |
|-------|---|---|---|---|---|---|---|---|---|----|-------|
| 1     | 1 |   |   |   |   |   |   |   |   |    | 1     |
| 2     |   | 2 |   |   |   |   |   |   |   |    | 2     |
| 3     |   |   | 3 |   |   |   |   |   |   |    | 3     |
| 4     |   |   |   | 4 |   |   |   |   |   |    | 4     |
| 5     |   |   |   |   | 5 |   |   |   |   |    | 5     |
| 6     |   |   |   |   |   | 6 |   |   |   |    | 6     |
| 7     |   |   |   |   |   |   | 7 |   |   |    | 7     |
| 8     |   |   |   |   |   |   |   | 8 |   |    | 8     |
| 9     |   |   |   |   |   |   |   |   | 9 |    | 9     |
| 10    |   |   |   |   |   |   |   |   |   | 10 | 10    |
| Total | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 55    |

Let us demonstrate how to add a value "21" to the matrix, at the intersection of column 7 and row 3. For that, change a code in the following way:

```
private void Matrix1_ManualBuild(object sender, EventArgs e)
{
    object[] columnValues = new object[1];
    object[] rowValues = new object[1];
    object[] cellValues = new object[1];

    for (int i = 1; i <= 10; i++)
    {
        columnValues[0] = i;
        rowValues[0] = i;
        cellValues[0] = i;

        Matrix1.AddValue(columnValues, rowValues, cellValues);
    }
    columnValues[0] = 7;
    rowValues[0] = 3;
    cellValues[0] = 21;
    Matrix1.AddValue(columnValues, rowValues, cellValues);
}
```

As a result, we have the following:

|       | 1 | 2 | 3 | 4 | 5 | 6 | 7  | 8 | 9 | 10 | Total |
|-------|---|---|---|---|---|---|----|---|---|----|-------|
| 1     | 1 |   |   |   |   |   |    |   |   |    | 1     |
| 2     |   | 2 |   |   |   |   |    |   |   |    | 2     |
| 3     |   |   | 3 |   |   |   | 21 |   |   |    | 24    |
| 4     |   |   |   | 4 |   |   |    |   |   |    | 4     |
| 5     |   |   |   |   | 5 |   |    |   |   |    | 5     |
| 6     |   |   |   |   |   | 6 |    |   |   |    | 6     |
| 7     |   |   |   |   |   |   | 7  |   |   |    | 7     |
| 8     |   |   |   |   |   |   |    | 8 |   |    | 8     |
| 9     |   |   |   |   |   |   |    |   | 9 |    | 9     |
| 10    |   |   |   |   |   |   |    |   |   | 10 | 10    |
| Total | 1 | 2 | 3 | 4 | 5 | 6 | 28 | 8 | 9 | 10 | 76    |

As seen, the matrix automatically calculates the totals.

You can use the `ManualBuild` event handler for the matrix which is connected to data. In this case, event handler is called first, then the matrix is filled with data from the data source.

# Advanced Matrix Object

This object, like the "Matrix" object, allows you to build summary reports.

|                            | Top 5 customers     |                     | Total                 |
|----------------------------|---------------------|---------------------|-----------------------|
|                            | Total               | Others (84)         |                       |
| ► Beverages (12) ⇄         | \$144 007,88        | \$192 684,30        | <b>\$336 692,18</b>   |
| ► Condiments (12) ⇄        | \$123 407,20        | \$105 665,89        | <b>\$229 073,09</b>   |
| ► Confections (13) ⇄       | \$54 823,13         | \$112 615,09        | <b>\$167 438,23</b>   |
| ► Dairy Products (10) ⇄    | \$103 074,28        | \$166 433,01        | <b>\$269 507,29</b>   |
| ► Grains/Cereals (7) ⇄     | \$31 368,01         | \$64 376,58         | <b>\$95 744,59</b>    |
| ▼ Meat/Poultry (6) ⇄       | \$269 495,38        | \$113 526,98        | <b>\$383 022,36</b>   |
| 1. Alice Mutton            | \$13 428,48         | \$19 269,90         | <b>\$32 698,38</b>    |
| 2. Mishi Kobe Niku         | \$1 319,20          | \$5 907,30          | <b>\$7 226,50</b>     |
| 3. Pâté chinois            | \$7 137,60          | \$10 288,80         | <b>\$17 426,40</b>    |
| 4. Perth Pasties           | \$6 916,56          | \$13 657,61         | <b>\$20 574,17</b>    |
| 5. Thüringer Rostbratwurst | \$239 795,40        | \$60 573,28         | <b>\$300 368,67</b>   |
| 6. Tourtière               | \$898,14            | \$3 830,10          | <b>\$4 728,24</b>     |
| ► Produce (5) ⇄            | \$67 154,22         | \$71 955,36         | <b>\$139 109,58</b>   |
| ► Seafood (12) ⇄           | \$31 732,55         | \$99 591,19         | <b>\$131 323,74</b>   |
| <b>Total</b>               | <b>\$825 062,63</b> | <b>\$926 848,41</b> | <b>\$1 751 911,04</b> |

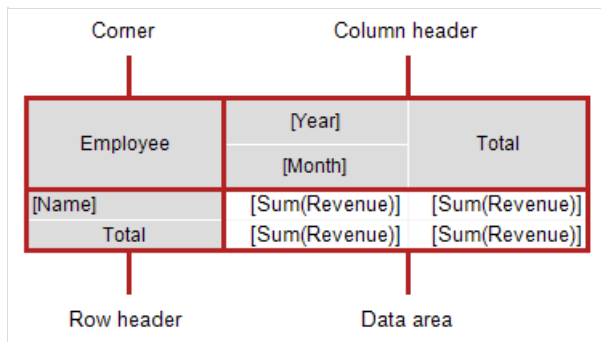
Here is a list of its key features:

- row and column headers can contain groups and simple elements in any order. This allows you to build asymmetric reports;
- collapse buttons allow you to interactively manage the visibility of individual elements;
- sorting buttons allow you to interactively sort the matrix by the selected values, including the total values;
- Top N grouping allows you to display N values in the header, and group the remaining values into a separate element with the ability to expand;
- output of matrix headers in a stepped form;
- sorting headers by total values;
- a wide range of aggregate functions;
- support of custom aggregate functions;
- a wide range of special functions that allow you to get the values of totals, adjacent cells, as well as functions for calculating percentages;
- support for "Sparkline" and "Gauge" objects in data cells.



# Matrix structure

The "Advanced Matrix" object consists of the following elements:



## Corner

The cells located in the corner of the matrix can contain arbitrary information. You can also split/combine them as you like.

## Header

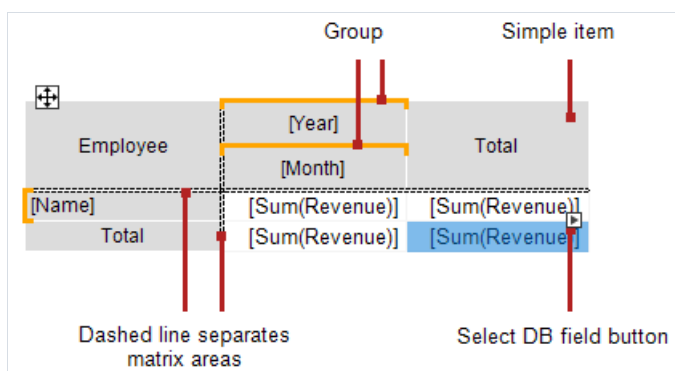
Matrix header can contain two types of elements:

- simple element: displays static information such as the text "Total".
- group: displays a list of values grouped by a specific criterion.

The header has a tree structure. The root element is invisible, it contains visible first-level elements.

Any arrangement of elements is allowed; for example, a header may not have a group, or it may have several adjacent groups. The totals can also be arranged in an arbitrary way.

In design mode, the matrix displays visual cues in the header area:



In this case, the header structure is as follows:

```
Row header
- "Name" group
- "Total" element

Column header
- "Year" group
- "Month" group
- "Total" element
```

See section [Header Setup](#) for more information.

## Data area

Cells in a data area usually contain an aggregate function. See section [Data Area Setup](#).

# Matrix setup

## Structure setup

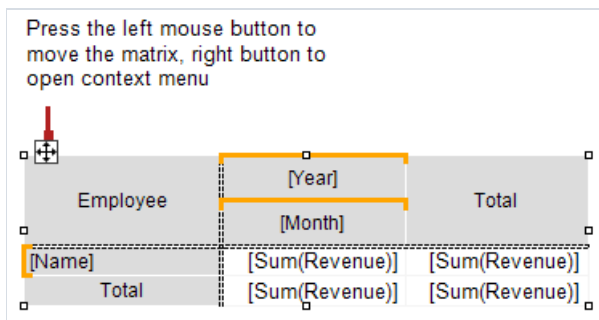
Follow these steps to set up the matrix:

1. Set up the headers (see section [Header Setup](#)).
2. Set up the data cells (see section [Data area Setup](#)).
3. Add totals (see section [Header Setup](#)). This step is best done last to save time setting up new data cells.

The matrix must be connected to the data source — the `DataSource` property is responsible for this. Typically, this property is set up automatically during the header and cell setup.

## Context menu

To open the context menu, select any element of the matrix, then right-click on the area in the upper left corner of the matrix:



The following commands are available in the menu:

- "Style" — select one of the available styles;
- "Swap Columns and Rows" — allows you to quickly swap columns and rows in the matrix;
- "Repeat headers" — column and row headers will be printed on each page if the matrix takes several pages.

## Settings available in the "Properties" window

The following properties are available in the "Properties" window that are specific to the "Advanced Matrix" object:

| Property                 | Value          | Description  |
|--------------------------|----------------|--|
| <b>DataRowPriority</b>   | Rows           | The priority of headers when accessing database fields from data cells. See section <a href="#">Properties available from data cells</a> . |
| <b>DataSource</b>        |                | Data source.   |
| <b>EvenStylePriority</b> | Rows           | The priority of the number of rows or columns to enable <code>EvenStyle</code> property.   |
| <b>Filter</b>            |                | Data filtering expression. See section <a href="#">Filtering Data</a> .  |
| <b>Layout</b>            | AcrossThenDown | See section <a href="#">Table Layout</a> .   |

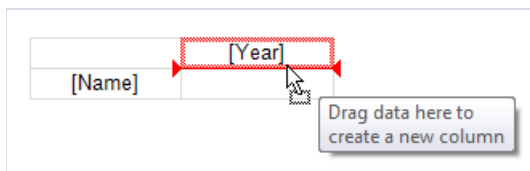
| Property              | Value | Description   |
|-----------------------|-------|---|
| <b>PrintIfEmpty</b>   | True  | Print matrix if empty.  |
| <b>RepeatHeaders</b>  | True  | Repeat headers on a new page.   |
| <b>ResetDataOnRun</b> | False | Reset data every time you run a report. By default, the matrix is not rebuilt during interactive operations (see section <a href="#">Interactive Options</a> ). |
| <b>Style</b>          |       | Matrix style.   |
| <b>WrappedGap</b>     | 0     | The gap between the parts of the matrix in the mode <code>Layout = Wrapped</code> .   |

# Header setup

## Adding an item

There are two ways to add an element to the header:

- by dragging the field from the "Data" window. As you drag, it will show in which part of the header a new element will be added:



- using the header context menu. Select the element next to which you want to add a new element and select one of the "Add a new element" commands from the context menu. An empty new element will be added;
- you can also add a "Total" element (before or after the selected element) using the context menu. A new element will be added with the text "Total" (the text depends on the current localization).

Adding a total is equivalent to adding an empty item and editing its text.

When you add a database field from the "Data" window to an empty matrix, its `DataSource` property is automatically configured.

## Removing an element

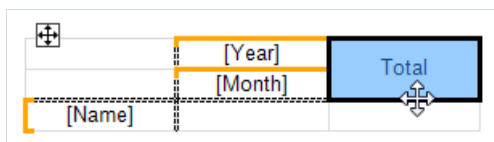
You can delete an element by selecting the "Delete" item in the context menu. You can delete only the selected element, or the element tree (the selected element and all its children).

You can also delete an element by pressing the `Delete` key. In this case, only the selected element is deleted.

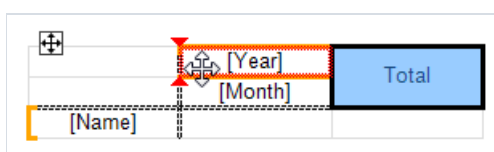
An element with a lock icon cannot be deleted in the described way. See section [TopN Grouping](#).

## Moving an element

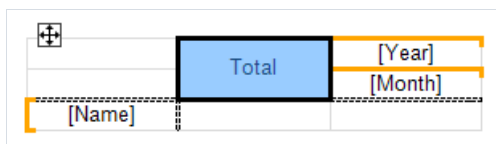
To move an element to a new location, select it with the left mouse button. The element will be marked with a thick black frame:



Grab the element by the frame and move it to a new location. As you drag, it will show in which part of the header a new element will be added:



Release the mouse button and the element will be moved to a new location:



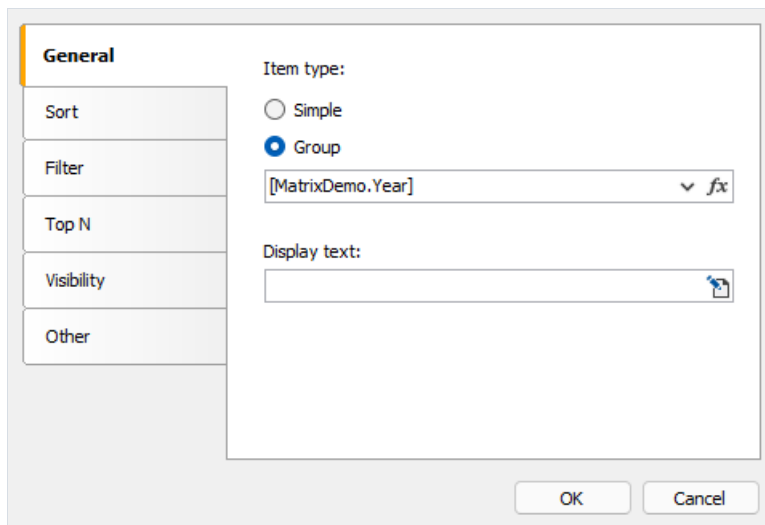
If the element has children, they will be moved with the main element, if it is possible.

## Editing elements

To call the element editor, double-click on it with the left mouse button, or select the "Edit ..." in the context menu. You can also call the editor by pressing Enter.

# Grouping

As described above, the matrix header can contain two types of elements — a group and a simple element. The element type is set in the header editor:



The screenshot shows a dialog box titled 'General' with a sidebar on the left containing buttons for 'Sort', 'Filter', 'Top N', 'Visibility', and 'Other'. The main area has two sections: 'Item type:' with radio buttons for 'Simple' and 'Group' (the latter is selected), and a dropdown menu showing '[MatrixDemo.Year]' with a 'fx' icon. Below this is a 'Display text:' section with an empty text box and a 'fx' icon. At the bottom are 'OK' and 'Cancel' buttons.

You can turn a simple element into a group and vice versa. Some of the settings in the editor window are not available for a simple element.

The group allows you to display a list of values grouped by condition. In the example above, the condition `[MatrixDemo.Year]` is specified — this means that the element will display a list of years. The list will contain non-duplicate values; identical values will be grouped.

By default, the group displays the values specified in the grouping condition. For example, if the condition `[MatrixDemo.Month]` is specified, the month numbers will be displayed. The "Display Text" property allows you to display a different value, for example:

```
[MonthName([MatrixDemo.Month])]
```

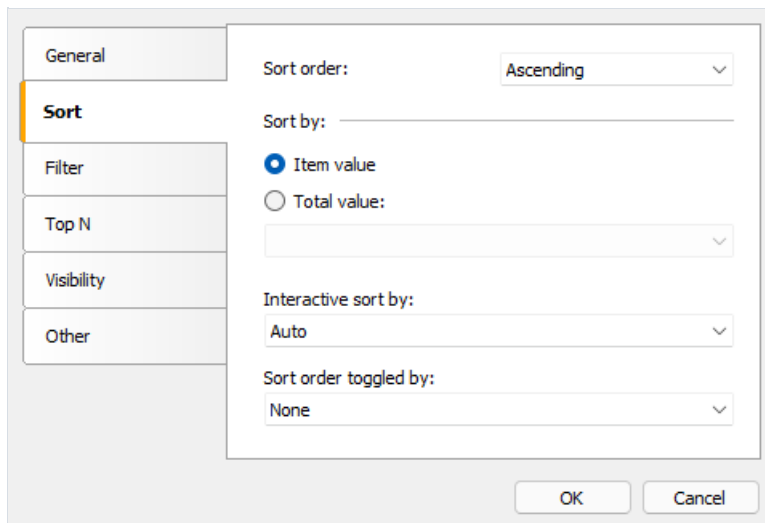
will display the name of the month instead of its number.

In this property, you can call special functions of the matrix (see the section [Properties accessible from header cells](#)), for example:

```
[Matrix1.RowNo].[MatrixDemo.Name]
```

# Sort

Sort settings for output values are presented on the tab of the same name in the header editor. The settings are active if the "Group" element type is selected:



General

**Sort**

Filter

Top N

Visibility

Other

Sort order: Ascending

Sort by:

☒ Item value

☐ Total value:

Interactive sort by: Auto

Sort order toggled by: None

OK Cancel

Description of settings:

"Sort order" — sets the sort order — ascending, descending or no sort.

"Sort by" — select one of two sort options: by displayed values, or by total value. The pictures below show how the "Year" element is sorted:

By element value:

|                 | 2011        | 2012       | 2013       | 2014       | 2015       |
|-----------------|-------------|------------|------------|------------|------------|
| Andrew Fuller   | \$3,900.00  | \$2,100.00 |            |            | \$1,800.00 |
| Janet Leverling | \$6,100.00  | \$3,200.00 |            |            |            |
| Nancy Davolio   | \$3,300.00  | \$2,700.00 | \$3,100.00 |            | \$1,700.00 |
| Steven Buchanan |             |            | \$3,999.00 | \$8,100.00 |            |
| Total           | \$13,300.00 | \$8,000.00 | \$7,099.00 | \$8,100.00 | \$3,500.00 |

By total value:

|                 | 2015       | 2013       | 2012       | 2014       | 2011        |
|-----------------|------------|------------|------------|------------|-------------|
| Andrew Fuller   | \$1,800.00 |            | \$2,100.00 |            | \$3,900.00  |
| Janet Leverling |            |            | \$3,200.00 |            | \$6,100.00  |
| Nancy Davolio   | \$1,700.00 | \$3,100.00 | \$2,700.00 |            | \$3,300.00  |
| Steven Buchanan |            | \$3,999.00 |            | \$8,100.00 |             |
| Total           | \$3,500.00 | \$7,099.00 | \$8,000.00 | \$8,100.00 | \$13,300.00 |

"Interactive sort by total" - defines how to sort the header values if interactive sort is active (see [Interactive sort](#)). The options are None, Auto, and the name of the total.

"Sort order toggled by button" — here the name of the button of type `MatrixSortButton` is specified, which is located in the corner of the matrix or outside of it. When you click the button in the report viewer, the sort order is changed and the matrix is updated.



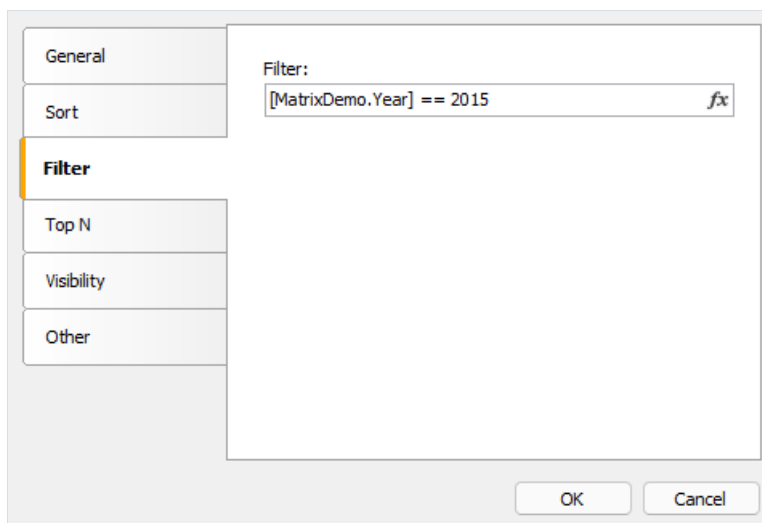
# Data filtering

There are two ways to filter the data displayed in the matrix.

Method 1: filtering using the `Filter` matrix property. To do this, select the "Matrix" object and fill the Property `Filter` in the "Properties" window:

```
[MatrixDemo.Year] == 2015
```

Method 2: filtering in the header editor using a similar expression:



In both cases, the filter expression returns the type `bool`.

The difference between the two described methods is how much data gets into the matrix. In the first case, all values are discarded, except for those where the year is 2015. The result may look like this:

|               | 2015       |
|---------------|------------|
| Andrew Fuller | \$1,800.00 |
| Nancy Davolio | \$1,700.00 |

In the second case, the values of a specific element in the header are filtered. Unlike the previous option, the result can contain empty values:

|                 | 2015       |
|-----------------|------------|
| Andrew Fuller   | \$1,800.00 |
| Janet Leverling |            |
| Nancy Davolio   | \$1,700.00 |
| Steven Buchanan |            |

# TopN grouping

If the number of values in the header group is large, it will generate an excessive number of report pages. TopN grouping allows you to display the first N values, and show the remaining values in a collapsed form:

|                         | Top 5 customers          |                            |                    |              |              |              | Total                 |
|-------------------------|--------------------------|----------------------------|--------------------|--------------|--------------|--------------|-----------------------|
|                         | Lonesome Pine Restaurant | Rattlesnake Canyon Grocery | Save-a-lot Markets | QUICK-Stop   | Ernst Handel | Others (84)  |                       |
| ► Beverages (12) ⇅      | \$2 708,00               | \$26 876,15                | \$65 498,00        | \$36 216,43  | \$12 709,30  | \$192 684,30 | <b>\$336 692,18</b>   |
| ► Condiments (12) ⇅     | \$82 480,00              | \$6 075,20                 | \$11 567,00        | \$9 214,94   | \$14 070,06  | \$105 665,89 | <b>\$229 073,09</b>   |
| ► Confections (13) ⇅    | \$549,00                 | \$10 947,21                | \$11 981,07        | \$18 530,09  | \$12 815,76  | \$112 615,09 | <b>\$167 438,23</b>   |
| ► Dairy Products (10) ⇅ | \$815,00                 | \$42 854,87                | \$21 107,10        | \$13 800,85  | \$24 496,46  | \$166 433,01 | <b>\$269 507,29</b>   |
| ► Grains/Cereals (7) ⇅  | \$190,00                 | \$4 831,31                 | \$8 298,10         | \$5 310,90   | \$12 737,70  | \$64 376,58  | <b>\$95 744,59</b>    |
| ► Meat/Poultry (6) ⇅    | \$140 098,40             | \$83 657,28                | \$27 659,18        | \$9 754,96   | \$8 325,56   | \$113 526,98 | <b>\$383 022,36</b>   |
| ► Produce (5) ⇅         | \$16 379,20              | \$13 836,05                | \$16 387,90        | \$8 081,40   | \$12 469,67  | \$71 955,36  | <b>\$139 109,58</b>   |
| ► Seafood (12) ⇅        | \$625,00                 | \$884,73                   | \$13 604,60        | \$9 367,74   | \$7 250,48   | \$99 591,19  | <b>\$131 323,74</b>   |
| Total                   | \$243 844,60             | \$189 962,80               | \$176 102,95       | \$110 277,31 | \$104 874,98 | \$926 848,41 | <b>\$1 751 911,04</b> |

## How it works

TopN function uses four elements to display data:

1. TopN group is a source group containing a large number of values.
2. TopN total, which displays a total of TopN values.
3. Group "Others", which displays values that are not included in TopN.
4. The result of the group "Others".

If the source group has fewer values than specified in the `TopN.Count` property, it is displayed as usual, without TopN grouping. Otherwise, the following happens:

- N values are left in the main group;
- the rest of the values are transferred to the "Others" group;
- data in the main group and in the "Others" group are aggregated;
- the obtained values are displayed as a total of TopN and as a total of the group "Others".

## Setup

TopN is set up for the main group. To do this, double-click on the element or select "Edit ..." from the context menu.

General

Sort

Filter

**Top N**

Visibility

Other

Display Top N values: 3

Additional items:

☒ Top N total Total

☐ Others

☒ Others total Others

☐ Create additional items in the matrix template for better customization

OK Cancel

There are two options for working with additional elements:

- the elements "TopN total", "Others", "Others total" are created automatically when building the matrix. Their visual design is copied from the main element. You can manage the visibility of the elements, as well as specify text for the total elements. There are no other options to customize the appearance;
- the above elements are added to the matrix template. This allows you to fully customize the appearance, as well as change the order of elements. You can add collapse buttons to interactively manage the visibility of individual elements.

Below is what the matrix template looks like when you add additional elements to it:

|                | Top 5 customers |               |              |              |              | Total        |
|----------------|-----------------|---------------|--------------|--------------|--------------|--------------|
|                | Total           | [CompanyName] | Others       |              |              |              |
| [CategoryName] | [Sum(Price)]    | [Sum(Price)]  | [Sum(Price)] | [Sum(Price)] | [Sum(Price)] | [Sum(Price)] |
| [ProductName]  | [Sum(Price)]    | [Sum(Price)]  | [Sum(Price)] | [Sum(Price)] | [Sum(Price)] | [Sum(Price)] |
| Total          | [Sum(Price)]    | [Sum(Price)]  | [Sum(Price)] | [Sum(Price)] | [Sum(Price)] | [Sum(Price)] |

In this case, the main group is highlighted in red, and additional elements are marked with a lock icon. You can delete them in the main element editor window by unticking "Create additional items" checkbox.

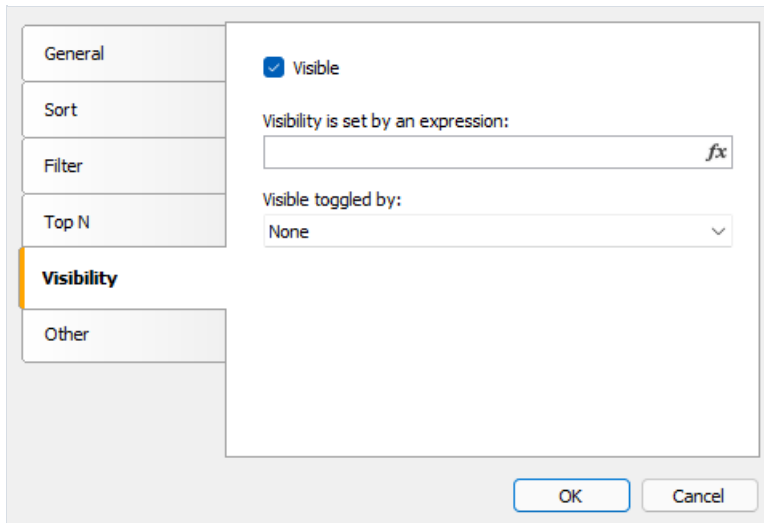
## TopN, BottomN, FirstN, LastN

The TopN engine uses the first N values in the original group. The meaning of the resulting values depends on how the original group was sorted:

- the group is sorted by the value of the header: the first N (ascending sort) or the last N values (descending sort) are displayed;
- the group is sorted by total value: the largest N (descending sort) or the N smallest values (ascending sort) are displayed.

# Element visibility

You can manage the visibility of an element on the "Visibility" tab:

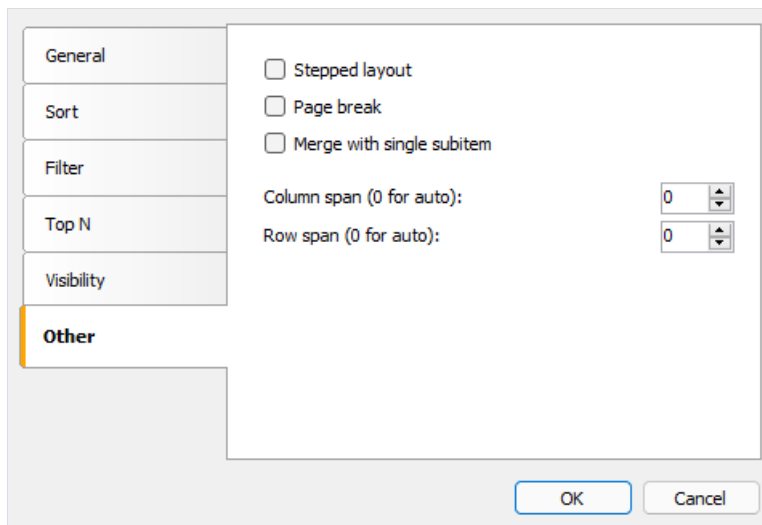


The screenshot shows a configuration dialog with a sidebar on the left containing tabs: General, Sort, Filter, Top N, **Visibility** (highlighted with an orange bar), and Other. The main area of the dialog is for the 'Visibility' tab. It contains a checked checkbox labeled 'Visible'. Below it, the text 'Visibility is set by an expression:' is followed by a text input field with a small 'fx' icon on the right. Further down, the text 'Visible toggled by:' is followed by a dropdown menu currently showing 'None'. At the bottom right of the dialog are 'OK' and 'Cancel' buttons.

- "Visible" - sets the initial visibility;
- "Visibility set by an expression" is an expression that returns type `bool` that sets visibility. If the expression is not empty, its value is used instead of the "Visible" option;
- "Visible toggled by button" - the name of the button of type `MatrixCollapseButton` that manages the visibility of this element. See the section [Collapsing/Expanding elements](#).

## Other settings

Other settings can be found on the "Other" tab of the header editor:



The "Stepped layout" option toggles the layout of nested elements between block (default) and stepped layout. Let's look at the example of a matrix with a nested row header (Year, Month). The block layout looks like this:

|        |         | [Name]         |
|--------|---------|----------------|
| [Year] | [Month] | [Sum(Revenue)] |
| Total  |         | [Sum(Revenue)] |

The finished report looks like this:

|       |    | Andrew Fuller | Janet Leverling |
|-------|----|---------------|-----------------|
| 2011  | 2  |               |                 |
|       | 10 | \$1,900.00    | \$3,000.00      |
|       | 11 | \$2,000.00    | \$3,100.00      |
|       | 12 |               |                 |
| 2012  | 1  |               |                 |
|       | 2  | \$2,100.00    |                 |
|       | 3  |               | \$3,200.00      |
| 2013  | 1  |               |                 |
|       | 2  |               |                 |
|       | 3  |               |                 |
| 2014  | 1  |               |                 |
|       | 2  |               |                 |
| 2015  | 1  | \$1,800.00    |                 |
| Total |    | \$7,800.00    | \$9,300.00      |

If you enable the "Stepped layout" option for the "Year" element, the arrangement of the elements will change as follows:

|         | [Name]         |
|---------|----------------|
| [Year]  | [Sum(Revenue)] |
| [Month] | [Sum(Revenue)] |
| Total   | [Sum(Revenue)] |

The finished report looks like this:

|       | Andrew Fuller | Janet Leverling |
|-------|---------------|-----------------|
| 2011  | \$3,900.00    | \$6,100.00      |
| 2     |               |                 |
| 10    | \$1,900.00    | \$3,000.00      |
| 11    | \$2,000.00    | \$3,100.00      |
| 12    |               |                 |
| 2012  | \$2,100.00    | \$3,200.00      |
| 1     |               |                 |
| 2     | \$2,100.00    |                 |
| 3     |               | \$3,200.00      |
| 2013  |               |                 |
| 1     |               |                 |
| 2     |               |                 |
| 3     |               |                 |
| 2014  |               |                 |
| 1     |               |                 |
| 2     |               |                 |
| 2015  | \$1,800.00    |                 |
| 1     | \$1,800.00    |                 |
| Total | \$7,800.00    | \$9,300.00      |

When you enable the "Stepped layout" option, the nested element changes the Property `Padding.Left`, which is responsible for text indentation. You can adjust this value in the Properties window.

This option can be used for any element that has nested elements.

The "Page Break" option inserts a new page before printing the element. No new page is inserted before the first element.

The option "Merge with a single subitem" is used in the case of dynamically collapsed headers (see section [Collapsing/expanding elements](#)). The option allows you to hide the "Total" element. Below is the view of the finished report with the disabled option (by default).

|       |       | Andrew Fuller |
|-------|-------|---------------|
| 2011  | Total | \$3,900.00    |
| 2012  | Total | \$2,100.00    |
| 2013  | 1     |               |
|       | 2     |               |
|       | 3     |               |
|       | Total |               |
| 2014  | Total |               |
| 2015  | Total | \$1,800.00    |
| Total |       | \$7,800.00    |

and with enabled, pay attention to the output of the "Year" element:

|       |       | Andrew Fuller |
|-------|-------|---------------|
| 2011  |       | \$3,900.00    |
| 2012  |       | \$2,100.00    |
| 2013  | 1     |               |
|       | 2     |               |
|       | 3     |               |
|       | Total |               |
| 2014  |       |               |
| 2015  |       | \$1,800.00    |
| Total |       | \$7,800.00    |

"Column span" and "Row span" options allow you to merge cells in columns and rows when element is printed. By default, the element controls these parameters automatically.

# Properties accessible from header cells

The matrix has a set of properties that you can use when printing header cells. The name of the matrix is used to access the property:

```
[Matrix1.RowNo]
```

| Property         | Return value | Description   |
|------------------|--------------|---|
| <b>RowNo</b>     | int          | Header element ordinal.                               |
| <b>ItemCount</b> | int          | The number of children in the current header element. |

These properties can be used in the "Displayed Text" field of a header cell, for example:

```
[Matrix1.RowNo].[MatrixDemo.Name]
```

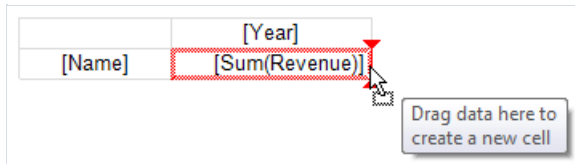
will display the text

```
1. Andrew Fuller
```

# Data area setup

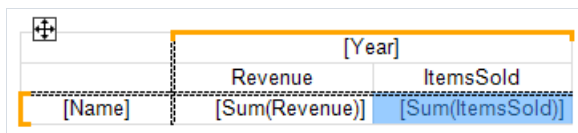
## Adding an item

You can add an element by dragging the DB field from the "Data" window. As you drag, it will be shown in which part of the data area the new element will be added:



When you add an element to an empty matrix, its Property `DataSource` is automatically configured.

When adding a new item, its title will also be added next to the existing one:



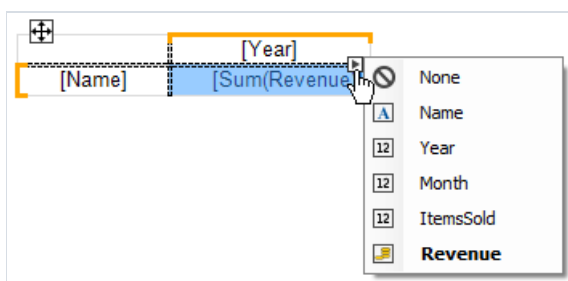
## Removing an element

You can clear the text of an element by choosing Delete or "Clear" from the item menu. The data element itself cannot be deleted; for this, you should delete the corresponding element in the matrix header. In the example above, you must remove the element "Revenue" from the header in order to delete `[Sum(Revenue)]`.

## Editing an item

To edit the text of an element, double click to call the text editor window. You can also call the editor by pressing the Enter key.

You can quickly select a DB field displayed in an element using the (smarttag) speed button, which is displayed when the mouse cursor is inside the element:



You can also perform the following operations using the context menu of an element:

- call the element editor;
- customize data formatting;
- change type of aggregate function (see section [Aggregate functions](#));
- add interest calculation;
- insert a progress indicator or sparkline into the cell.

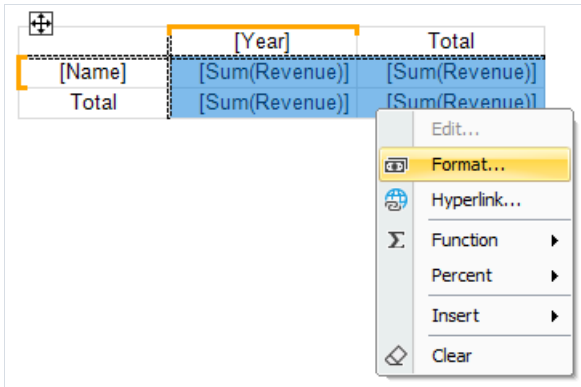


## Editing multiple elements

For bulk editing of cells, you can select the entire data area or multiple cells. This can be done using the **Shift** key or the mouse:

- select the starting cell;
- click the left mouse button and hold it while moving the mouse to select a group of cells;

For the selected cells, you can call the context menu or use the quick selection button for the database field:



# Aggregate functions

Aggregate functions are used in data cells to aggregate cell values and to calculate totals. An aggregate function call looks like this:

```
[Function(Expression)]
```

Square brackets are used to specify expressions in cell text. You can use multiple aggregate functions in one cell along with regular text.

Expression is usually a data source field. An example of using an aggregate function:

```
[Sum([MatrixDemo.Revenue])]
```

Below is a list of aggregate functions:

| Function             | Description   |
|----------------------|---|
| <b>Sum</b>           | Returns the sum of values.  |
| <b>Min</b>           | Returns the minimum value.  |
| <b>Max</b>           | Returns the maximum value.  |
| <b>Avg</b>           | Returns the average value.  |
| <b>Count</b>         | Returns the number of values.   |
| <b>CountDistinct</b> | Returns the number of different (unique) values.  |
| <b>StDev</b>         | Returns the standard deviation of a sample.   |
| <b>StDevP</b>        | Returns the standard deviation of a population.   |
| <b>Var</b>           | Returns the variance for a sample.  |
| <b>VarP</b>          | Returns the variance for a population.  |
| <b>First</b>         | Returns the first value.  |
| <b>Last</b>          | Returns the last value.   |
| <b>ValuesList</b>    | Returns a list of all values found in a cell. This aggregate is used to work together with the "Diagram" and "Sparkline" objects. |
| <b>_name</b>         | Custom aggregate function defined in the report code.   |

A custom function has a name that begins with an underscore. Its code should be placed in the body of the main report class, `ReportScript`. The function is defined as follows:

```
object _FuncName(List<dynamic> l)
```

Example of a custom function `_Sum`:

```
public class ReportScript
{
    public object _Sum(List<dynamic> l)
    {
        dynamic value = 0;
        foreach (dynamic v in l)
            value += v;
        return value;
    }
}
```

# Special functions

Special functions can be used in the data cell of the matrix. They allow you to get the value of another cell in the same row or column.

## GrandColumnTotal

Returns the value of the grand total for the column.

| Parameter | Description   |
|-----------|---|
| aggregate | (optional) Aggregate function. Parameter can be omitted if the aggregate has already been accessed in the cell. |

Examples:

```
GrandColumnTotal(Sum([MatrixDemo.Revenue]))
Sum([MatrixDemo.Revenue]) / GrandColumnTotal()
```

## GrandRowTotal

Returns the value of the grand total for a row.

| Parameter | Description   |
|-----------|---|
| aggregate | (optional) Aggregate function. Parameter can be omitted if the aggregate has already been accessed in the cell. |

Examples:

```
GrandRowTotal(Sum([MatrixDemo.Revenue]))
Sum([MatrixDemo.Revenue]) / GrandRowTotal()
```

## GrandTotal

Returns the value of the grand total.

| Parameter | Description   |
|-----------|---|
| aggregate | (optional) Aggregate function. Parameter can be omitted if the aggregate has already been accessed in the cell. |

Examples:

```
GrandTotal(Sum([MatrixDemo.Revenue]))
Sum([MatrixDemo.Revenue]) / GrandTotal()
```

## ColumnTotal

Returns the value of the column total for the current group.

| Parameter              | Description   |
|------------------------|---|
| <code>aggregate</code> | (optional) Aggregate function. Parameter can be omitted if the aggregate has already been accessed in the cell. |

Examples:

```
ColumnTotal(Sum([MatrixDemo.Revenue]))  
Sum([MatrixDemo.Revenue]) / ColumnTotal()
```

## RowTotal

Returns the value of the row total for the current group.

| Parameter              | Description   |
|------------------------|---|
| <code>aggregate</code> | (optional) Aggregate function. Parameter can be omitted if the aggregate has already been accessed in the cell. |

Examples:

```
RowTotal(Sum([MatrixDemo.Revenue]))  
Sum([MatrixDemo.Revenue]) / RowTotal()
```

## ColumnMaxValue

Returns the maximum value of the column total for the current group.

| Parameter              | Description   |
|------------------------|---|
| <code>aggregate</code> | (optional) Aggregate function. Parameter can be omitted if the aggregate has already been accessed in the cell. |

Examples:

```
ColumnMaxValue(Sum([MatrixDemo.Revenue]))  
Sum([MatrixDemo.Revenue]) / ColumnMaxValue()
```

## ColumnMinValue

Returns the minimum value of the column total for the current group.

| Parameter | Description   |
|-----------|---|
| aggregate | (optional) Aggregate function. Parameter can be omitted if the aggregate has already been accessed in the cell. |

Examples:

```
ColumnMinValue(Sum([MatrixDemo.Revenue]))
Sum([MatrixDemo.Revenue]) / ColumnMinValue()
```

## RowMaxValue

Returns the maximum value of the row total for the current group.

| Parameter | Description   |
|-----------|---|
| aggregate | (optional) Aggregate function. Parameter can be omitted if the aggregate has already been accessed in the cell. |

Examples:

```
RowMaxValue(Sum([MatrixDemo.Revenue]))
Sum([MatrixDemo.Revenue]) / RowMaxValue()
```

## RowMinValue

Returns the minimum value of the row total for the current group.

| Parameter | Description   |
|-----------|---|
| aggregate | (optional) Aggregate function. Parameter can be omitted if the aggregate has already been accessed in the cell. |

Examples:

```
RowMinValue(Sum([MatrixDemo.Revenue]))
Sum([MatrixDemo.Revenue]) / RowMinValue()
```

## FirstColumn

Returns the value of the first cell in the column.

| Parameter | Description   |
|-----------|---|
| aggregate | (optional) Aggregate function. Parameter can be omitted if the aggregate has already been accessed in the cell. |

| Parameter                             | Description   |
|---------------------------------------|---|
| <code>useInteractiveSort=false</code> | (optional) Include the results of interactive sort, in which the order of the elements can be changed . |
| <code>useThisGroup=true</code>        | (optional) Use a group at the same level to find a value.   |

The returned value depends on where the function is used:

- total cell: returns the value of the first cell in the group if parameter `useThisGroup` is `true` , otherwise it returns the value of the first total;
- group cell: returns the value of the first cell in the group.

Examples:

```
FirstColumn(Sum([MatrixDemo.Revenue]))
FirstColumn(Sum([MatrixDemo.Revenue]), true)
Sum([MatrixDemo.Revenue]) / FirstColumn()
```

## FirstRow

Returns the value of the first cell in a row.

| Parameter                             | Description   |
|---------------------------------------|---|
| <code>aggregate</code>                | (optional) Aggregate function. Parameter can be omitted if the aggregate has already been accessed in the cell. |
| <code>useInteractiveSort=false</code> | (optional) Include the results of interactive sort, in which the order of the elements can be changed.          |
| <code>useThisGroup=true</code>        | (optional) Use a group at the same level to find a value.   |

The returned value depends on where the function is used:

- total cell: returns the value of the first cell in the group if parameter `useThisGroup` is `true` , otherwise it returns the value of the first total;
- group cell: returns the value of the first cell in the group.

Examples:

```
FirstRow(Sum([MatrixDemo.Revenue]))
FirstRow(Sum([MatrixDemo.Revenue]), true)
Sum([MatrixDemo.Revenue]) / FirstRow()
```

## LastColumn

Returns the value of the last cell in the column.

| Parameter                             | Description   |
|---------------------------------------|---|
| <code>aggregate</code>                | (optional) Aggregate function. Parameter can be omitted if the aggregate has already been accessed in the cell. |
| <code>useInteractiveSort=false</code> | (optional) Include the results of interactive sort, in which the order of the elements can be changed.          |
| <code>useThisGroup=true</code>        | (optional) Use a group at the same level to find a value.   |

The returned value depends on where the function is used:

- total cell: returns the value of the last cell in the group if parameter `useThisGroup` is `true`, otherwise it returns the value of the last total;
- group cell: returns the value of the last cell in the group.

Examples:

```
LastColumn(Sum([MatrixDemo.Revenue]))
LastColumn(Sum([MatrixDemo.Revenue]), true)
Sum([MatrixDemo.Revenue]) / LastColumn()
```

## LastRow

Returns the value of the last cell in a row.

| Parameter                             | Description   |
|---------------------------------------|---|
| <code>aggregate</code>                | (optional) Aggregate function. Parameter can be omitted if the aggregate has already been accessed in the cell. |
| <code>useInteractiveSort=false</code> | (optional) Include the results of interactive sort, in which the order of the elements can be changed.          |
| <code>useThisGroup=true</code>        | (optional) Use a group at the same level to find a value.   |

The returned value depends on where the function is used:

- total cell: returns the value of the last cell in the group if parameter `useThisGroup` is `true`, otherwise it returns the value of the last total;
- group cell: returns the value of the last cell in the group.

Examples:

```
LastRow(Sum([MatrixDemo.Revenue]))
LastRow(Sum([MatrixDemo.Revenue]), true)
Sum([MatrixDemo.Revenue]) / LastRow()
```

## PreviousColumn

Returns the value of the previous cell in the column.



| Parameter                             | Description   |
|---------------------------------------|---|
| <code>aggregate</code>                | (optional) Aggregate function. Parameter can be omitted if the aggregate has already been accessed in the cell. |
| <code>useInteractiveSort=false</code> | (optional) Include the results of interactive sort, in which the order of the elements can be changed.          |
| <code>useThisGroup=true</code>        | (optional) Use a group at the same level to find a value.   |

The returned value depends on where the function is used:

- total cell: returns the value of the previous cell in the group if parameter `useThisGroup` is `true`, otherwise it returns the value of the previous total;
- group cell: returns the value of the previous cell in the group.

Examples:

```
PreviousColumn(Sum([MatrixDemo.Revenue]))
PreviousColumn(Sum([MatrixDemo.Revenue]), true)
Sum([MatrixDemo.Revenue]) / PreviousColumn()
```

## PreviousRow

Returns the value of the previous cell in a row.

| Parameter                             | Description   |
|---------------------------------------|---|
| <code>aggregate</code>                | (optional) Aggregate function. Parameter can be omitted if the aggregate has already been accessed in the cell. |
| <code>useInteractiveSort=false</code> | (optional) Include the results of interactive sort, in which the order of the elements can be changed.          |
| <code>useThisGroup=true</code>        | (optional) Use a group at the same level to find a value.   |

The returned value depends on where the function is used:

- total cell: returns the value of the previous cell in the group if parameter `useThisGroup` is `true`, otherwise it returns the value of the previous total;
- group cell: returns the value of the previous cell in the group.

Examples:

```
PreviousRow(Sum([MatrixDemo.Revenue]))
PreviousRow(Sum([MatrixDemo.Revenue]), true)
Sum([MatrixDemo.Revenue]) / PreviousRow()
```

## NextColumn

Returns the value of the next cell in the column.

| Parameter                             | Description   |
|---------------------------------------|---|
| <code>aggregate</code>                | (optional) Aggregate function. Parameter can be omitted if the aggregate has already been accessed in the cell. |
| <code>useInteractiveSort=false</code> | (optional) Include the results of interactive sort, in which the order of the elements can be changed.          |
| <code>useThisGroup=true</code>        | (optional) Use a group at the same level to find a value.   |

The returned value depends on where the function is used:

- total cell: returns the value of the next cell in the group if parameter `useThisGroup` is `true`, otherwise it returns the value of the next total;
- group cell: returns the value of the next cell in the group.

Examples:

```
NextColumn(Sum([MatrixDemo.Revenue]))
NextColumn(Sum([MatrixDemo.Revenue]), true)
Sum([MatrixDemo.Revenue]) / NextColumn()
```

## NextRow

Returns the value of the next cell in a row.

| Parameter                             | Description   |
|---------------------------------------|---|
| <code>aggregate</code>                | (optional) Aggregate function. Parameter can be omitted if the aggregate has already been accessed in the cell. |
| <code>useInteractiveSort=false</code> | (optional) Include the results of interactive sort, in which the order of the elements can be changed.          |
| <code>useThisGroup=true</code>        | (optional) Use a group at the same level to find a value.   |

The returned value depends on where the function is used:

- total cell: returns the value of the next cell in the group if parameter `useThisGroup` is `true`, otherwise it returns the value of the next total;
- group cell: returns the value of the next cell in the group.

Examples:

```
NextRow(Sum([MatrixDemo.Revenue]))
NextRow(Sum([MatrixDemo.Revenue]), true)
Sum([MatrixDemo.Revenue]) / NextRow()
```

## SpecificColumn

Returns the value of the cell with the specified column index.

| Parameter              | Description   |
|------------------------|---|
| <code>index</code>     | Index value   |
| <code>aggregate</code> | (optional) Aggregate function. Parameter can be omitted if the aggregate has already been accessed in the cell. |

The returned value depends on where the function is used:

- total cell: returns the value of the cell in the group of the total;
- otherwise, it returns the value of the cell in the column of the current group.

Examples:

```
SpecificColumn("Andrew Fuller", Sum([MatrixDemo.Revenue]))
SpecificColumn(2011, Sum([MatrixDemo.Revenue]))
Sum([MatrixDemo.Revenue]) / SpecificColumn(2011)
```

## SpecificRow

Returns the value of the cell with the specified index in a row.

| Parameter              | Description   |
|------------------------|---|
| <code>index</code>     | Index value   |
| <code>aggregate</code> | (optional) Aggregate function. Parameter can be omitted if the aggregate has already been accessed in the cell. |

The returned value depends on where the function is used:

- total cell: returns the value of the cell in the group of the total;
- otherwise, it returns the value of the cell in the row of the current group.

Examples:

```
SpecificRow("Andrew Fuller", Sum([MatrixDemo.Revenue]))
SpecificRow(2011, Sum([MatrixDemo.Revenue]))
Sum([MatrixDemo.Revenue]) / SpecificRow(2011)
```

## PercentOfColumnTotal

Returns the value of the current cell divided by the value of the column total.

| Parameter              | Description   |
|------------------------|---|
| <code>aggregate</code> | (optional) Aggregate function. Parameter can be omitted if the aggregate has already been accessed in the cell. |

Function call

```
PercentOfColumnTotal(Sum([MatrixDemo.Revenue]))
```

is equivalent to the following code:

```
Sum([MatrixDemo.Revenue]) / ColumnTotal()
```

Examples:

```
PercentOfColumnTotal(Sum([MatrixDemo.Revenue]))  
[Sum([MatrixDemo.Revenue])] [PercentOfColumnTotal()]
```

## PercentOfRowTotal

Returns the value of the current cell divided by the value of the row total.

| Parameter              | Description   |
|------------------------|---|
| <code>aggregate</code> | (optional) Aggregate function. Parameter can be omitted if the aggregate has already been accessed in the cell. |

Examples:

```
PercentOfRowTotal(Sum([MatrixDemo.Revenue]))  
[Sum([MatrixDemo.Revenue])] [PercentOfRowTotal()]
```

## PercentOfGrandTotal

Returns the value of the current cell divided by the value of the grand total.

| Parameter              | Description   |
|------------------------|---|
| <code>aggregate</code> | (optional) Aggregate function. Parameter can be omitted if the aggregate has already been accessed in the cell. |

Examples:

```
PercentOfGrandTotal(Sum([MatrixDemo.Revenue]))  
[Sum([MatrixDemo.Revenue])] [PercentOfGrandTotal()]
```

## PercentOfPreviousColumn

Returns the value of the current cell divided by the value of the previous cell in the column.

| Parameter                             | Description   |
|---------------------------------------|---|
| <code>aggregate</code>                | (optional) Aggregate function. Parameter can be omitted if the aggregate has already been accessed in the cell. |
| <code>useInteractiveSort=false</code> | (optional) Include the results of interactive sort, in which the order of the elements can be changed .         |
| <code>useThisGroup=true</code>        | (optional) Use a group at the same level to find a value.   |

The returned value depends on where the function is used:

- total cell: the value of the previous cell in the group is used if parameter `useThisGroup` is `true` , otherwise the value of the previous total is used;
- group cell: the value of the previous cell in the group is used.

Examples:

```
PercentOfPreviousColumn(Sum([MatrixDemo.Revenue]))
PercentOfPreviousColumn(Sum([MatrixDemo.Revenue]), true)
[Sum([MatrixDemo.Revenue])] [PercentOfPreviousColumn()]
```

## PercentOfPreviousRow

Returns the value of the current cell divided by the value of the previous cell in a row.

| Parameter                             | Description   |
|---------------------------------------|---|
| <code>aggregate</code>                | (optional) Aggregate function. Parameter can be omitted if the aggregate has already been accessed in the cell. |
| <code>useInteractiveSort=false</code> | (optional) Include the results of interactive sort, in which the order of the elements can be changed .         |
| <code>useThisGroup=true</code>        | (optional) Use a group at the same level to find a value.   |

The returned value depends on where the function is used:

- total cell: the value of the previous cell in the group is used if parameter `useThisGroup` is `true` , otherwise the value of the previous total is used;
- group cell: uses the value of the previous cell in the group.

Examples:

```
PercentOfPreviousRow(Sum([MatrixDemo.Revenue]))
PercentOfPreviousRow(Sum([MatrixDemo.Revenue]), true)
[Sum([MatrixDemo.Revenue])] [PercentOfPreviousRow()]
```

# Properties accessible from data cells

The matrix has a set of properties that you can use when printing data cells. The name of the matrix is used to access the property:

```
[Matrix1.RowIndex]
```

| Property            | Return value | Description                             |
|---------------------|--------------|---|
| <b>ColumnIndex</b>  | int          | Index of the current column.            |
| <b>RowIndex</b>     | int          | Index of the current line.              |
| <b>ColumnValues</b> | object[]     | Array of values from the column header. |
| <b>RowValues</b>    | object[]     | Array of values from the row header.    |

These properties can be useful for highlighting cells with color based on a condition.

You can also access data source fields from a cell. As a rule, this is required to enable conditional highlighting (see section [Conditional highlighting](#)). Thus, for a data cell, you can specify the following condition to highlight values related to 2012:

```
[MatrixDemo.Year] == 2012
```

The value of the DB field that was used to print the cell is taken from the matrix header. Since there are two headings (row and column), you must specify which heading values will have priority. Matrix property `DataRowPriority` is responsible for this. The property is set to `Rows` by default.



# Interactive options

This section discusses interactive options of the "Advanced Matrix" object:






- [collapsing/expanding header elements](#);
- [interactive sort](#).

# Collapsing/Expanding elements

You can interactively manage the visibility of individual header elements using a special button of type `MatrixCollapseButton`. The button is inserted into the header element and manages the visibility of other elements. The picture below shows the button and its managed element (marked with a red frame when the button is selected):

|        |   |                |       |
|--------|---|----------------|-------|
|        |  | [Year]         |       |
|        |  | [Month]        | Total |
| [Name] | [Sum(Revenue)]  | [Sum(Revenue)] |       |

When you click on a button in the preview window, related elements are hidden or shown. In this case, the report is rebuilt:

|                 |  |  |  |  |            |  |             |
|-----------------|--|--|--|--|------------|--|-------------|
|                 |  2011 |  2012 |  2013 |  2014 |            |  2015 |             |
|                 | Total  | Total  | Total  | 1  | 2          | Total  | Total       |
| Andrew Fuller   | \$3,900.00   | \$2,100.00   |  |  |            | \$1,800.00   | \$7,800.00  |
| Janet Leverling | \$6,100.00   | \$3,200.00   |  |  |            |  | \$9,300.00  |
| Nancy Davolio   | \$3,300.00   | \$2,700.00   | \$3,100.00   |  |            | \$1,700.00   | \$10,800.00 |
| Steven Buchanan |  |  | \$3,999.00   | \$4,000.00   | \$4,100.00 | \$8,100.00   | \$12,099.00 |

## Adding a button

You can add a button to a header element using the context menu. Select the item, right-click and select Collapse button. The button is added to the left side of the element.



When a button is added, the element's Property `Padding.Left` changes so that the button does not overlap the text.

## Button customization


When a button is added, FastReport automatically configures the link between the button and its managed element. In some cases, you may need to configure the link manually. To do this, open the editor of the element that should be dependent on the button, and specify the name of the button on the "Visibility/Visibility is toggled by" tab.

The button can manage several elements at the same time.

The button can be placed above the managed element:

|        |   |                |       |
|--------|---|----------------|-------|
|        |  | [Year]         |       |
|        |  | [Month]        | Total |
| [Name] | [Sum(Revenue)]  | [Sum(Revenue)] |       |

or at the same level with it:

|        |                |   |       |
|--------|----------------|---|-------|
|        | [Year]         |  | Total |
| [Name] | [Sum(Revenue)] | [Sum(Revenue)]  |       |

The initial state of the controlled element — its visibility — is set in the element editor on the "Visibility/Visible" tab.

## Deleting a button



You can delete a button in two ways:

- select the button and press the `Delete` key;
- uncheck the "Collapse button" in the context menu of the element.

## Moving a button

By default, the button has Property `Dock = Left`. This means that it is docked to the left edge of the element. Set Property `Dock = None` in the "Properties" window to move the button to a new place.

You can also use the Property `Anchor` of a button to anchor it to a specific location of an element.

## Customizing the appearance of the button

Using the "Border" toolbar, you can customize the button icon: frame color and style, background color. In addition, you can set the following properties of the button in the Properties window:

| Property                      | Default value | Description   |
|-------------------------------|---------------|---|
| <b>Cursor</b>                 | Hand          | Mouse cursor shape.   |
| <b>Exclusive</b>              | False         | If <code>true</code> , then only one element can be expanded.   |
| <b>Exportable</b>             | False         | If <code>true</code> , the button will be displayed when exporting the report.  |
| <b>Printable</b>              | False         | If <code>true</code> , the button will be displayed when printing a report.   |
| <b>ShowCollapseExpandMenu</b> | False         | Determines whether a menu with "Collapse/Expand All" items should be shown when the right mouse button is pressed on this button. |
| <b>Symbol</b>                 | PlusMinus     | The symbol displayed inside the button.   |
| <b>SymbolSize</b>             | 5             | Button Symbol Size.   |

# Interactive sort

The sort button `MatrixSortButton` allows you to interactively sort the rows or columns of the matrix. The button should be inserted into a lower-level header element:

|        | [Year]         | Total          |
|--------|----------------|----------------|
| [Name] | [Sum(Revenue)] | [Sum(Revenue)] |

When you click on the button in the preview window, the opposite header is sorted. The example below sorts rows by value in the selected column:

|                 | 2011       | 2012       | 2013       | 2014       | 2015       | Total       |
|-----------------|------------|------------|------------|------------|------------|-------------|
| Andrew Fuller   | \$3,900.00 | \$2,100.00 |            |            | \$1,800.00 | \$7,800.00  |
| Janet Leverling | \$6,100.00 | \$3,200.00 |            |            |            | \$9,300.00  |
| Nancy Davolio   | \$3,300.00 | \$2,700.00 | \$3,100.00 |            | \$1,700.00 | \$10,800.00 |
| Steven Buchanan |            |            | \$3,999.00 | \$8,100.00 |            | \$12,099.00 |

Each press of the button switches the sort mode: ascending/descending/no sort.

## Adding a button

You can add a button to a header element using the context menu. Select the element, right-click and select "Sort Button". The button will be added to the right part of the element.

When you add a button, the element's Property `Padding.Right` changes so that the button does not overlap the text.

## Button customization

The header sort mode is set in its editor on the "Sort/Interactive Sort" by Total tab. The following values are possible:

- "No" - this header is not sorted.
- "Auto" is the default mode. Sort is performed by the value of the first total (aggregate).
- Total (aggregate) name: if the header has several output values, you can select one of them to sort. In the example below, to sort the row header by the `ItemsSold` value, select the `Sum ([MatrixDemo.ItemsSold])` aggregate:

|                 |           | Total       |
|-----------------|-----------|-------------|
| Nancy Davolio   | Revenue   | \$10,800.00 |
|                 | ItemsSold | 12          |
| Steven Buchanan | Revenue   | \$12,099.00 |
|                 | ItemsSold | 11          |
| Janet Leverling | Revenue   | \$9,300.00  |
|                 | ItemsSold | 9           |
| Andrew Fuller   | Revenue   | \$7,800.00  |
|                 | ItemsSold | 8           |

## Removing a button

There are two ways to remove a button:

- select the button and press the `Delete` key;
- uncheck the "Sort button" item in the context menu of the element.

## Moving a button

By default, the button has Property `Dock = Right` . This means that it is docked to the right edge of the element. To move the button to a new location, set Property `Dock = None` in the Properties window.

You can also use the Property `Anchor` of a button to anchor it to a specific location on an element.

## Customizing button appearance

Using the "Border" toolbar, you can customize the button icon: frame color and style, background color. You can also set the following properties of the button in the Properties window:

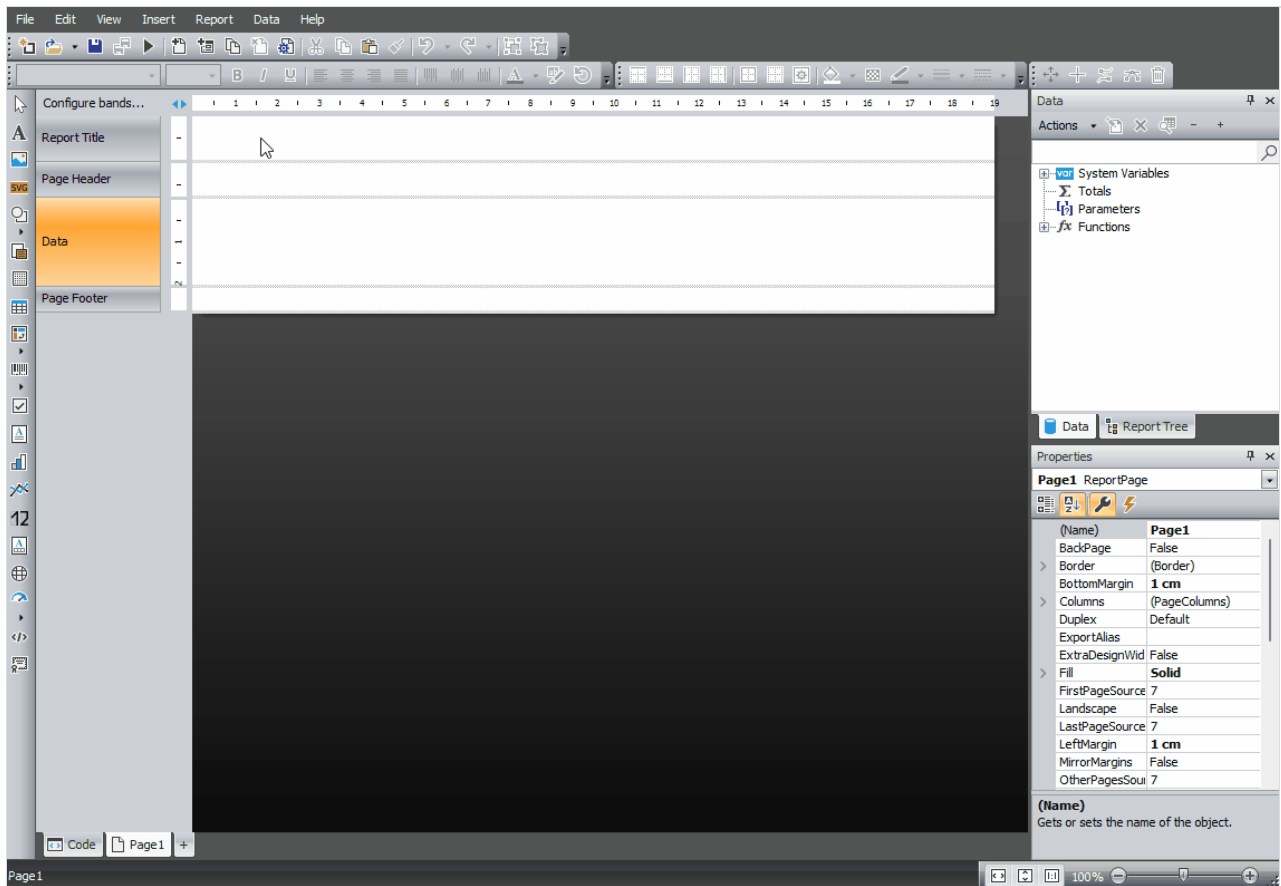
| Property                 | Default value | Description  |
|--------------------------|---------------|--|
| <b>AllowInactiveSort</b> | True          | Determines whether the button can be in an inactive state ("no sort" mode).    |
| <b>Cursor</b>            | Hand          | Mouse cursor shape.  |
| <b>Exportable</b>        | False         | If <code>true</code> , the button will be displayed when exporting the report. |
| <b>InactiveSortColor</b> | Gray          | Button color in inactive state.  |
| <b>Printable</b>         | False         | If <code>true</code> , the button will be displayed when printing a report.    |
| <b>Symbol</b>            | Arrow         | Button symbol.   |
| <b>SymbolSize</b>        | 7             | Button symbol size.  |

# Examples

Below are examples of using the "Advanced Matrix" object in the form of animated gifs.

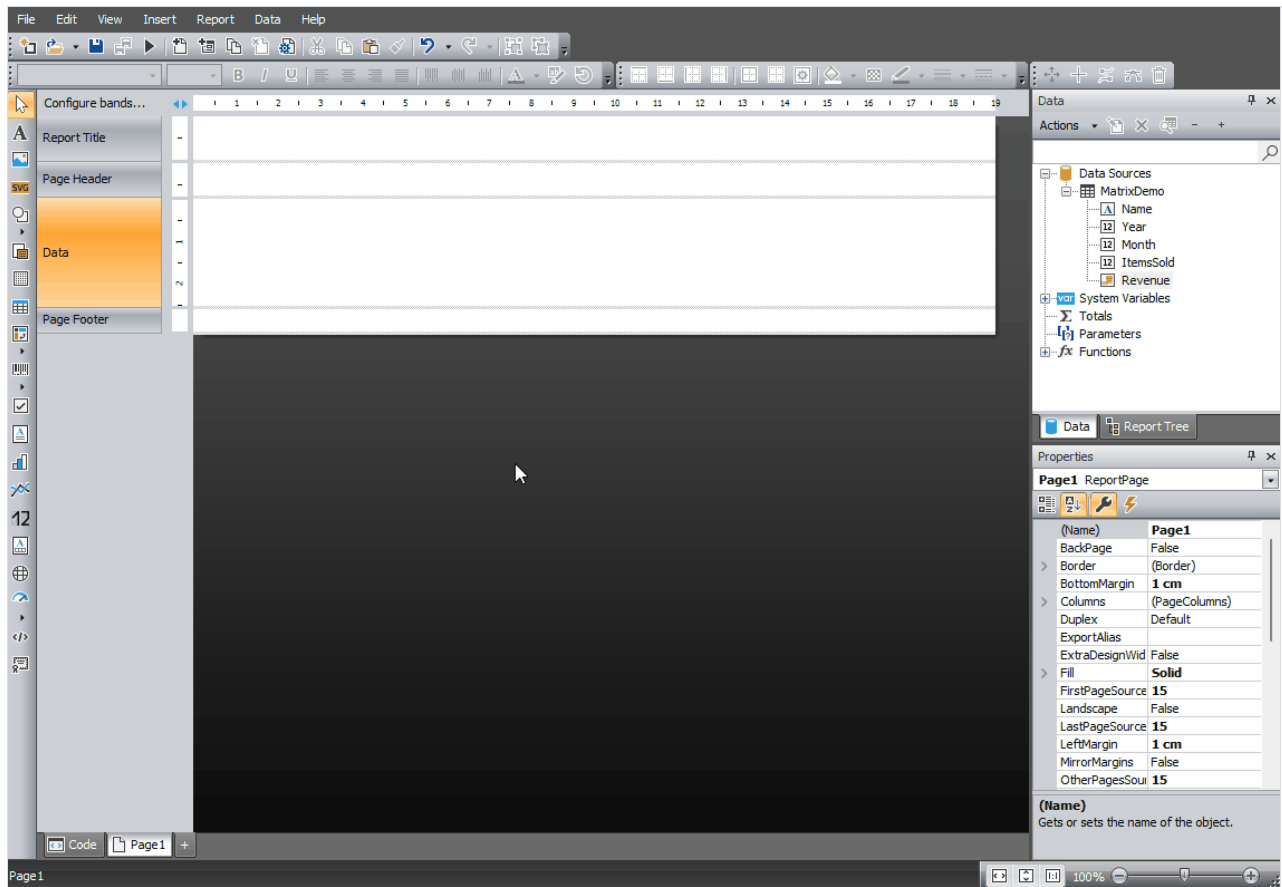
# Example 1. Simple matrix

Creating a simple matrix, choosing a style, adding totals:



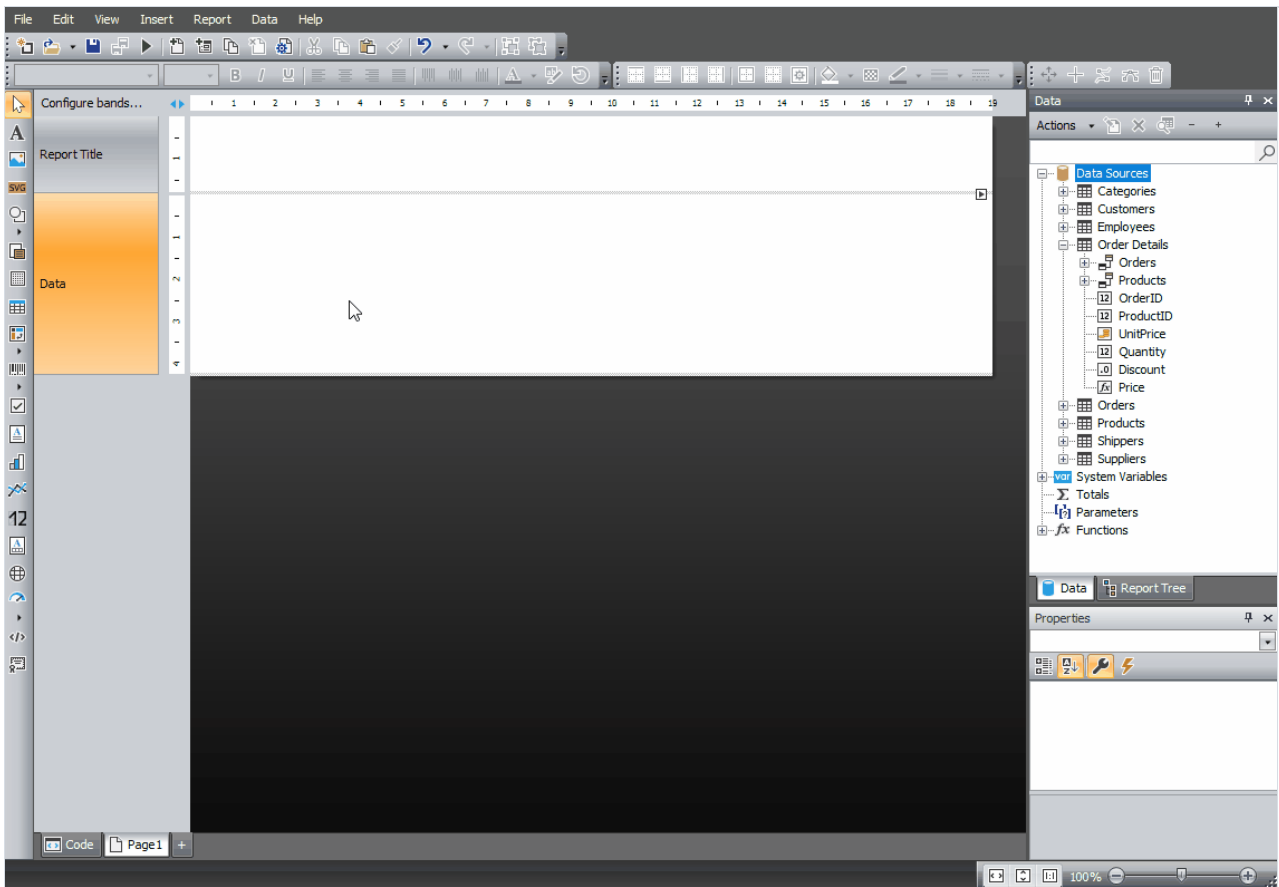
## Example 2. Matrix with Composite Headers

Creating a matrix with composite headers, adding totals, using the `DisplayText` property, hierarchical display of headers, collapse/expand buttons:



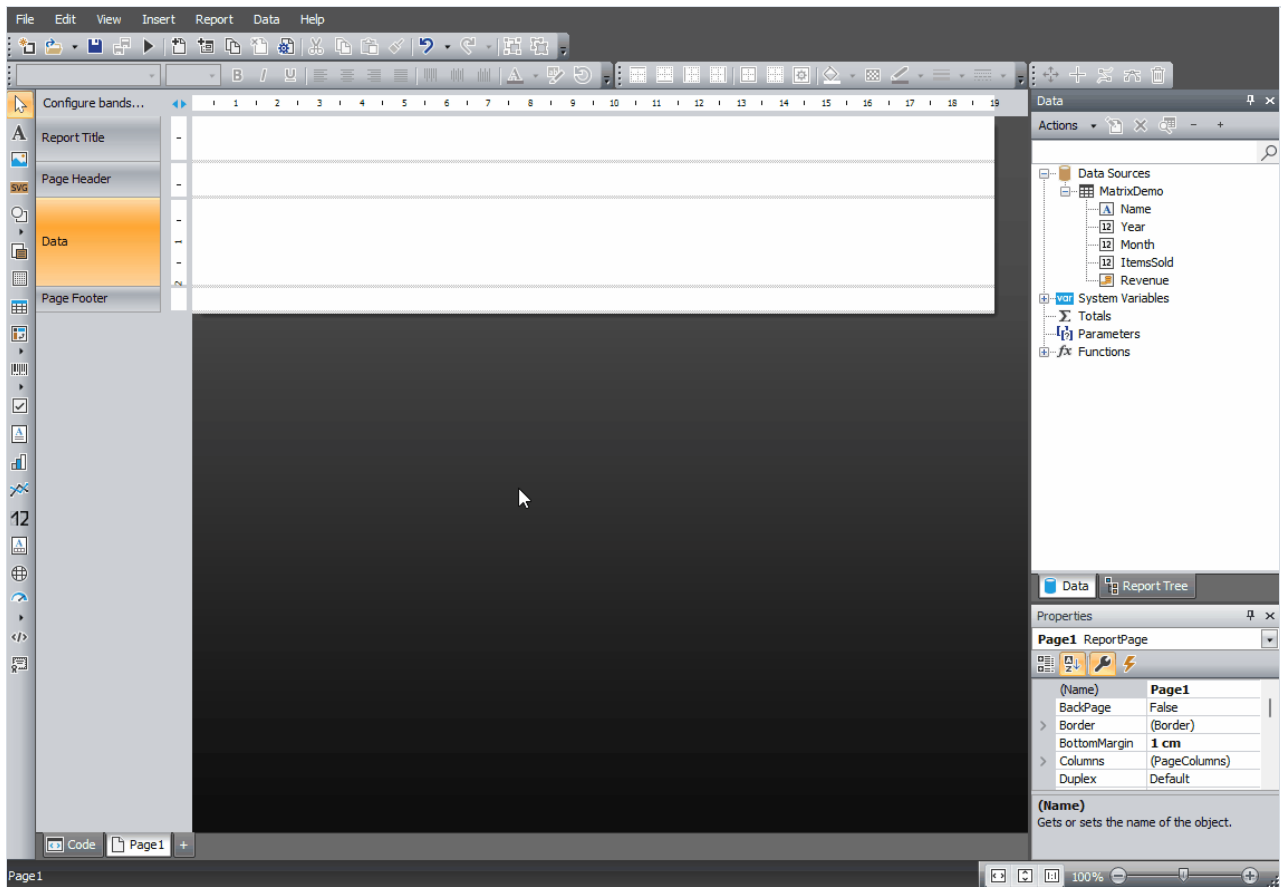
## Example 3. TopN Grouping

Creating and configuring a TopN group (two variants - simple and customizable), expanding items, using the matrix property `ItemCount` :



## Example 4. Interactive Sorting

Adding interactive sorting buttons, adding a second database field to the data area ( `ItemsSold` ), configuring interactive sorting by this field.





# Interactive reports

A FastReport's prepared report can be made interactive. This means that, it will react to the user's actions in the preview window. You can use the following interaction:

- when clicking on the report object, some kind of operation is performed. For example, you can run detailed report and show it in a separate window;
- preview window can show the report outline, which can be used for navigating on the report.

# Hyperlink

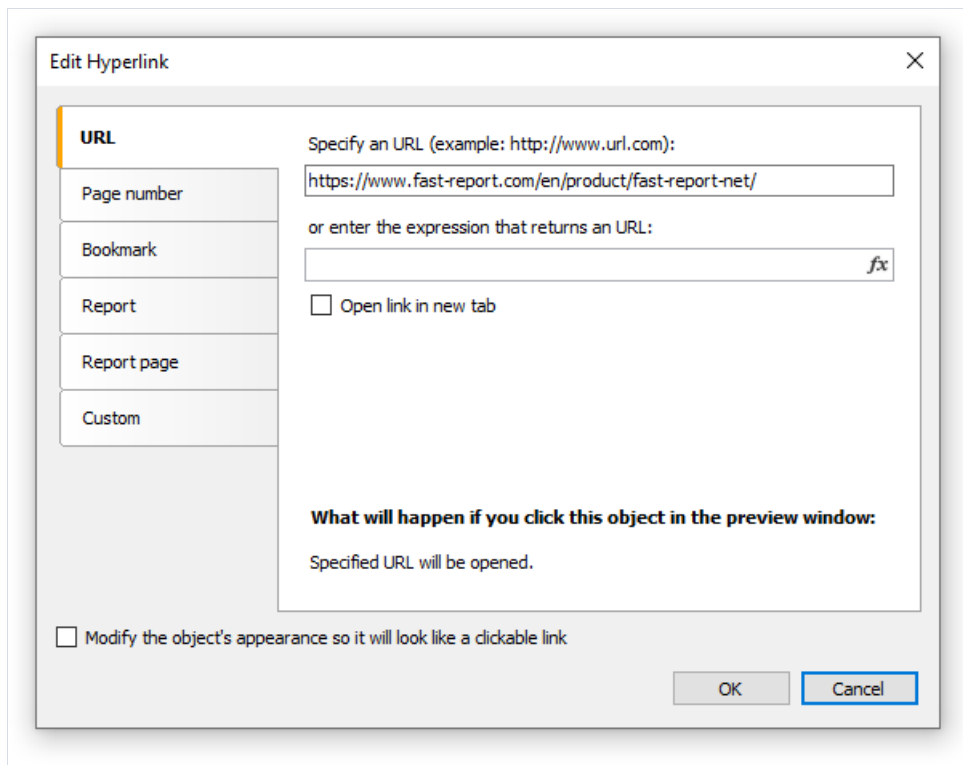
Almost all report objects have the `Hyperlink` property. Using this property, you can define an object's reaction to the mouse click in the preview window.

When clicking such an object, one of the following can occur:

- navigate to the URL address;
- send e-mail;
- execute any kind of system command;
- navigate to the report page with the indicated number;
- navigate to the bookmark, defined in another report object;
- run detailed report in a separate preview window;
- custom action, defined in a script.

# Hyperlink configuration

To configure a hyperlink, select the object which you want to make interactive, and right click on it. In the context menu, select the "Hyperlink..." item. The hyperlink editor window will open:



Choose the type of hyperlink by selecting the tab in the left side of the window. After you have done, you may click the "Modify the object's appearance..." checkbox at the bottom of the window. The appearance of the object will change in the following way:

- blue color will be set for the text and it will underlined;
- a hand cursor form will be set.

In some cases hyperlink needs to be shown in the preview window, but there is no need to print it. This is easy to do, if you disable the `Printable` object property. This can be done in the "Properties" window.

# Link to the URL

Using this type of link, you can:

- navigate to the given internet address;
- execute some kind of system commands, for example, `mailto:` for sending an email.

When clicking on the link of this type, the `System.Diagnostics.Process.Start` method is executed with the link's value as a parameter.

You can indicate the value of the link by using two methods:

- indicate the value directly, for example, "http://www.fast-report.com";
- indicate an expression, which returns the value of the link. This expression will be calculated when you run the report.

# Link to the page number

By using this link type, you can organize navigation on the pages of a prepared report. Most often, navigation to the first page is used. For this, indicate the page number (1 in the given case) as a link value.

You can indicate the page number by using two methods:

- indicate the number directly, for example, 1;
- indicate an expression, which returns the page number. This expression will be calculated when you run the report.

# Link to a bookmark

By using this type of link, you can navigate to a bookmark, defined in another report object.

For those who know the HTML language, it is enough to say that a bookmark acts like an anchor. A bookmark has got a name and a definite position in a prepared report (page number and position on the page). When moving to a bookmark by its name, you navigate onto the indicated position.

In order to use this link type, you first need to define the bookmark. In order to do this, select the object, where you want to move when you click on the link. Almost all report objects have the `Bookmark` property. Changing this property can be done with the help of the "Properties" window.

The `Bookmark` contains an expression, which you can use in the following way:

- indicate the bookmark name as a string:

```
"MyBookmark"
```

- indicate an expression, which returns the name of the bookmark. For example, you can use a data column as an expression. The value of the expression will be calculated when the report is run.

After the bookmark has been defined, you can indicate its name in the hyperlink configurations window. This can be done by using two methods:

- indicate the name of the bookmark directly;
- indicate an expression which returns the name of the bookmark. For example, this can be a data column. This expression will be calculated when the report is run.

# Link to a detailed report

Using this link type, you can execute another report and show it in a separate preview window.

You must set the following parameters for this type of hyperlink:

- detailed report's name;
- name of the report's parameter, which will take the hyperlink's value;
- hyperlink value.

**Edit Hyperlink**

URL

Page number

Bookmark

**Report**

Report page

Custom

Report name:  
D:\FR.NET\Demos\Reports\Interactive Report - Details.frx

Report parameter:  
CategoryName

Specify a parameter value:

or enter the expression that returns a parameter value:  
[Categories.CategoryName]

**What will happen if you click this object in the preview window:**  
Specified report will be opened in separate preview tab.

☐ Modify the object's appearance so it will look like a clickable link

OK Cancel

When the link is clicked, the following will take place:

- the indicated report will be loaded;
- the report's parameter will be set to the hyperlink's value;
- the report will be built and run in a separate preview window.

Report parameter's value can be indicated by using two methods:

- indicate the value directly;
- indicate an expression, which returns the value. This expression will be calculated when the report is run.

## Link to a detailed page

This link type works in the same way, except that, another page in the current report is used as a detailed report. For this, your report must contain at least two pages: one with the main report, another with detailed one.

You must set the following parameters for this type of hyperlink:

- page name in that report;
- name of the report's parameter, which will take the hyperlink's value;
- hyperlink value.

The screenshot shows the 'Edit Hyperlink' dialog box. On the left, a sidebar contains tabs: 'URL', 'Page number', 'Bookmark', 'Report', 'Report page' (selected), and 'Custom'. The main area is divided into two sections. The top section has three fields: 'Report page:' with a dropdown menu showing 'Page2', 'Report parameter:' with a dropdown menu showing 'CategoryName', and 'Specify a parameter value:' with an empty text input field. Below these is a field for 'or enter the expression that returns a parameter value:' containing the text '[Categories.CategoryName]'. The bottom section contains a preview text: 'What will happen if you click this object in the preview window: Specified page will be generated and opened in separate preview tab.' At the bottom left, there is a checkbox labeled 'Modify the object's appearance so it will look like a clickable link' which is currently unchecked. At the bottom right, there are 'OK' and 'Cancel' buttons.

When the link is clicked, the following will take place:

- the report's parameter will be set to the hyperlink's value;
- the indicated report page will be built and shown in a separate preview window.

Report parameter's value can be indicated by using two methods:


- indicate the value directly;
- indicate an expression, which returns the value. This expression will be calculated when the report is run.

When you choose a report page, its **Visible** property resets to **false**. This means that, when the main report will be built, this page will be skipped.



# Custom link

Using this type of link you can define own reaction to the clicking of the mouse. For this, use the `Click` event handler of the object. To do this:

- select the object and open the "Properties" window;
- Click the  button to show the object's events;
- double click on the `Click` event. FastReport switches to the "Code" window and creates an empty event handler.

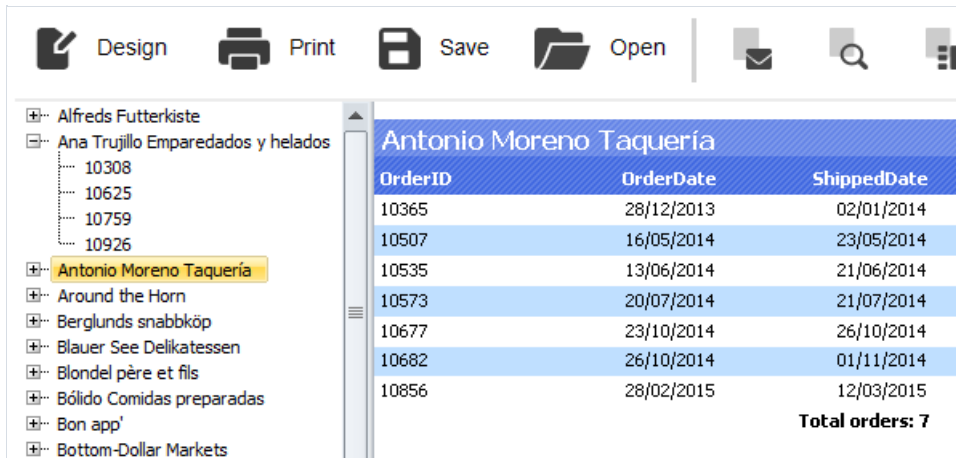
In the handler's code, do everything what you need. You, most likely, will need a link to the object, which the handler calls, and a hyperlink's value. Use the handler's parameter `sender` :

```
private void Text2_Click(object sender, EventArgs e)
{
    // sender - this is the object which was clicked.
    // In order to receive the value of the hyperlink, you need
    // to cast the sender to ReportComponentBase type.
    object hyperlinkValue = (sender as ReportComponentBase).Hyperlink.Value;


    MessageBox.Show("Hyperlink value = " + hyperlinkValue.ToString());
}
```

# Report outline

Report outline (also known as the "document map") is a TreeView control displayed in the preview window:



This control shows a tree structure, which was formed during the report building. If you click the tree element, you will navigate to the corresponding report element.

If the report has got an outline, it will be shown automatically. You can show or hide the outline by clicking the  button on the toolbar. The report does not create an outline automatically - you should take care about this.

The report page and all its bands have the `OutlineExpression` property. To fill the outline, indicate an expression which returns the element's text in this property. This expression will be calculated when printing a band, and its value will be added to the outline. If your report is of master-detail or group type, the structure of the outline will be similar to the report's structure.

The `OutlineExpression` property can be set in the "Properties" window.

Here are the recommendations on how to configure the outline for different types of reports:

- if you want to show the sheets of a prepared report in the outline, set the `OutlineExpression` property of the report page. The expression will return the number of the page:

[PageN]

- in the "Simple list" report type with one "Data" band, set the `OutlineExpression` property of that band. As an expression, use any data column which is printed in the band;
- in the master-detail report type with two "Data" bands, set the `OutlineExpression` property of the corresponding bands. For example, in the "Category/Product" report type, the `OutlineExpression` for the first band will contain the name of the category, for the second - product's name;
- in the group report, configure the `OutlineExpression` property of the group header and a "Data" band. As an expression for the group header, use the grouping condition. For the "Data" band, use any data column which is printed in the band.

# Examples

# Example 1. Link to a web page

In this example, we will create a simple report with one "Text" object. When clicking on the object in the preview window we will move to the FastReport web-page.

Create a new report, and add the "Text" object. Write the following text into it:

Go FastReport home page

Right click on the object and select the "Hyperlink..." item in the context menu. Configure the link in the following way:

**Edit Hyperlink**

**URL**

Specify an URL (example: <http://www.url.com>):

<https://www.fast-report.com/en/product/fast-report-net/>

or enter the expression that returns an URL:

*fx*

☐ Open link in new tab

**What will happen if you click this object in the preview window:**

Specified URL will be opened.

☐ Modify the object's appearance so it will look like a clickable link

OK Cancel

After this, enable the "Modify the object's appearance..." checkbox, in order to apply some link attributes (blue text color, underlining and a hand-like cursor) to the object.

Run the report and click on the object. The web-browser window opens, and you will move to the FastReport home page.

## Example 2. Building a detailed report

In this example we will build a report that displays the category list. When clicking on the category name, a detailed report that contains the list of products' in the given category will be shown.

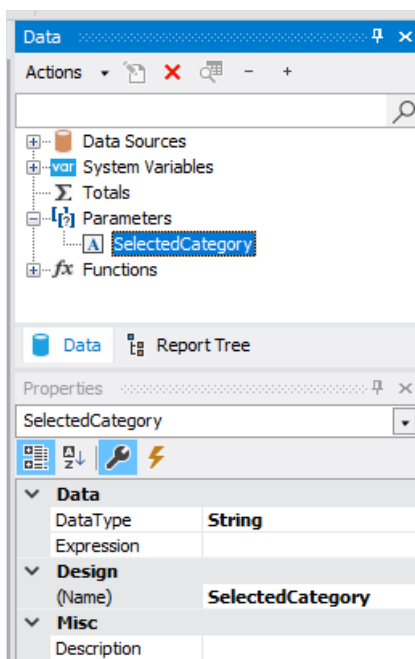
You need to do the following:

- first create a detailed report;
- define the report parameter which identifies a category;
- set up data filtration on this parameter;
- create the main report;
- in the main report, configure the hyperlink such that, the detailed report can be run with the parameter set to the chosen category.

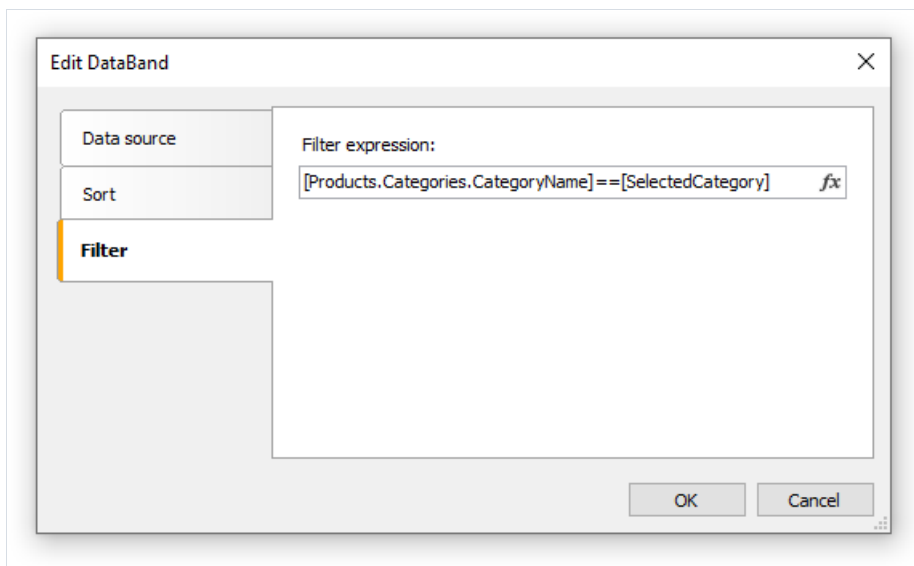
Firstly, we will create a detailed report which prints a list of products. For this, create a new report and choose the "Products" table as a data source. Place the objects in the following way:

|                |   |                        |                            |                      |
|----------------|---|------------------------|----------------------------|----------------------|
| Report Title   | - | Products               |                            |                      |
| Page Header    | - | Product Name           | Quantity Per Unit          | UnitPrice            |
| Data: Products | - | [Products.ProductName] | [Products.QuantityPerUnit] | [Products.UnitPrice] |

Create the parameter which will be used to pass a selected category from the main report to the detailed one. For category identification, we will use the `CategoryID` column which is contained in both "Categories" and "Products" tables. Configure the parameter in the following way:



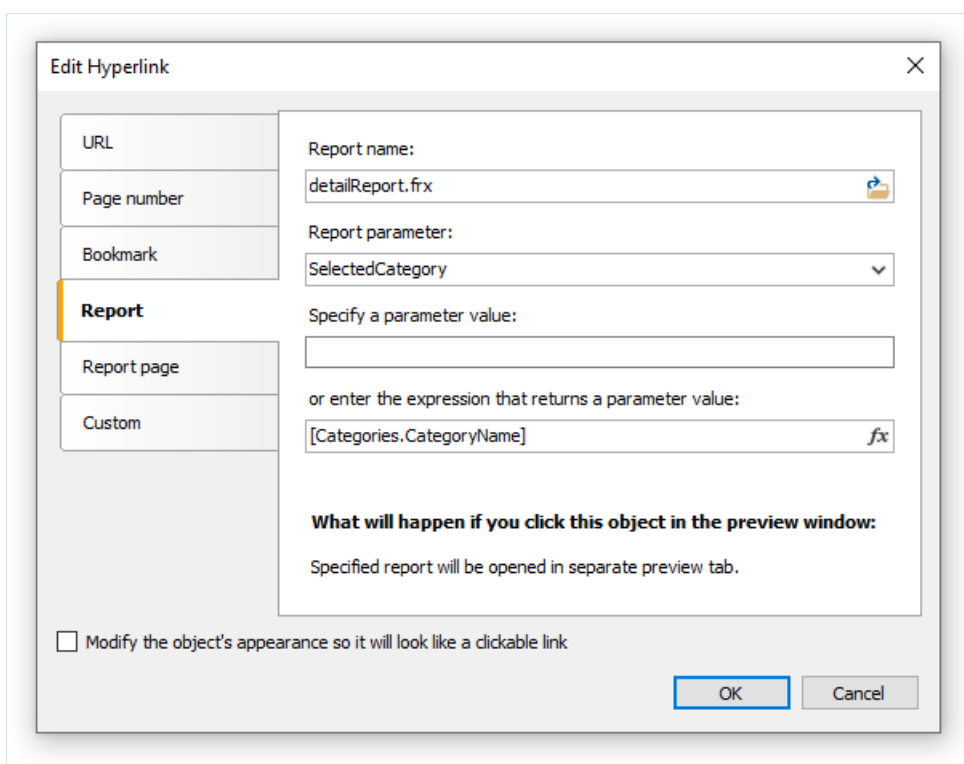
Now we need to set data filtering to filter all products that belong to the specified category. To do this, double click the "Data" band. Switch to the "Filter" tab and indicate the following condition:



Now create the main report. Create a new report and choose the "Categories" table as a data source. Place the objects in the following way:



Right click on the "Text" object and select the "Hyperlink..." menu item. Set up the link in the following way:



As a report name, choose the name of the detailed report file. Report parameter can be chosen from the drop-down list, by pressing the button on the right side of the list. As a parameter value, indicate the `[Categories.CategoryID]` expression.

Run the report, and you will see the categories list:

[Beverages](#)

[Condiments](#)

[Confections](#)

[Dairy Products](#)

[Grains/Cereals](#)

[Meat/Poultry](#)

[Produce](#)

[Seafood](#)

If you click on one of the categories, a detailed report will be built. It will be shown on a separate tab of the preview window:

Close detailed report      Detailed report tab

| Product Name           | Quantity Per Unit    | UnitPrice |
|------------------------|----------------------|-----------|
| Pavlova                | 32 - 500 g boxes     | 17.45     |
| Teatime Chocolate      | 10 boxes x 12 pieces | 9.2       |
| Sir Rodney's Marmalade | 30 gift boxes        | 81        |
| Sir Rodney's Scones    | 24 pkgs. x 4 pieces  | 10        |
| NuNuCa Nuß-Nougat-     | 20 - 450 g glasses   | 14        |
| Gumbär Gummibärchen    | 100 - 250 g bags     | 31.23     |
| Schoggi Schokolade     | 100 - 100 g pieces   | 43.9      |
| Zaanse koeken          | 10 - 4 oz boxes      | 9.5       |
| Chocolade              | 10 pkgs.             | 12.75     |
| Maxilaku               | 24 - 50 g pkgs.      | 20        |
| Valkoinen suklaa       | 12 - 100 g bars      | 16.25     |
| Tarte au sucre         | 48 pies              | 49.3      |
| Scottish Longbreads    | 10 boxes x 8 pieces  | 12.5      |

As seen on the picture, the title of the tab is set to the hyperlink's value. In our case, this is the numeric value contained in the `CategoryID` data column. This appears not informative and not beautiful. Let's change our report to use the category name instead of its number. For this, do the following:

In the detail report:

- change the parameter's `DataType` property to the `String` ;
- add the "Categories" data source into the report. It will be used for referring to the `CategoryName` column when filtering data;
- change the filtering expression of the "Data" band:

```
[Products.Categories.CategoryName] == [SelectedCategory]
```

In the main report:

- change the hyperlink settings. Now we will pass the `[Categories.CategoryName]` value into the report parameter.

If we run the report now, we will see that the title of the tab is set to category name. We can improve the detailed report a little. Add the "Text" object, which will print the name of the chosen category in the report title:

| Products               |                      |           |
|------------------------|----------------------|-----------|
| Product Name           | Quantity Per Unit    | UnitPrice |
| Pavlova                | 32 - 500 g boxes     | 17.45     |
| Teatime Chocolate      | 10 boxes x 12 pieces | 9.2       |
| Sir Rodney's Marmalade | 30 gift boxes        | 81        |
| Sir Rodney's Scones    | 24 pkgs. x 4 pieces  | 10        |
| NuNuCa Nuß-Nougat-     | 20 - 450 g glasses   | 14        |
| Gumbär Gummibärchen    | 100 - 250 g bags     | 31.23     |
| Schoggi Schokolade     | 100 - 100 g pieces   | 43.9      |
| Zaanse koeken          | 10 - 4 oz boxes      | 9.5       |
| Chocolade              | 10 pkgs.             | 12.75     |
| Maxilaku               | 24 - 50 g pkgs.      | 20        |
| Valkoinen suklaa       | 12 - 100 g bars      | 16.25     |
| Tarte au sucre         | 48 pies              | 49.3      |
| Scottish Longbreads    | 10 boxes x 8 pieces  | 12.5      |

While we are working with this example, we have created two reports and swap between them several times. This is not very comfortable. In order to make the task easier, two reports can be placed into one: the main report will be on the first page, the detail one on the second page. In this case the hyperlink needs to be set in the following way:

Edit Hyperlink

URL

Page number

Bookmark

Report

**Report page**

Custom

Report page:

Page2

Report parameter:

SelectedCategory

Specify a parameter value:

or enter the expression that returns a parameter value:

[Categories.CategoryName] *fx*

**What will happen if you click this object in the preview window:**

Specified page will be generated and opened in separate preview tab.

☒ Modify the object's appearance so it will look like a clickable link

OK

Cancel

In the given case, we need to choose Page2 as the detail report page.



## Example 3. Interactive "Matrix" object

In this example we will see how to build a detailed report if we click on the cell of the "Matrix" object. As an example, we will use a matrix which displays the sales of employees grouped by year.

As a data source for the matrix, the "MatrixDemo" table is used. It presents the sales of the employees, grouped by year and month:

| Name            | Year | Month | ItemsSold | Revenue |
|-----------------|------|-------|-----------|---------|
| Nancy Davolio   | 1999 | 2     | 1         | 1000    |
| Nancy Davolio   | 1999 | 11    | 1         | 1100    |
| Nancy Davolio   | 1999 | 12    | 1         | 1200    |
| Nancy Davolio   | 2000 | 1     | 1         | 1300    |
| Nancy Davolio   | 2000 | 2     | 2         | 1400    |
| Nancy Davolio   | 2001 | 2     | 2         | 1500    |
| Nancy Davolio   | 2001 | 3     | 2         | 1600    |
| Nancy Davolio   | 2002 | 1     | 2         | 1700    |
| Andrew Fuller   | 2002 | 1     | 2         | 1800    |
| Andrew Fuller   | 1999 | 10    | 2         | 1900    |
| Andrew Fuller   | 1999 | 11    | 2         | 2000    |
| Andrew Fuller   | 2000 | 2     | 2         | 2100    |
| Janet Leverling | 1999 | 10    | 3         | 3000    |
| Janet Leverling | 1999 | 11    | 3         | 3100    |
| Janet Leverling | 2000 | 3     | 3         | 3200    |
| Steven Buchanan | 2001 | 1     | 3         | 4000    |
| Steven Buchanan | 2001 | 2     | 4         | 4100    |
| Steven Buchanan | 2000 | 1     | 4         | 3999    |

Configure the matrix in the following way:

- put the `MatrixDemo.Year` data column in the column header;
- put the `MatrixDemo.Name` data column in the row header;

- put the `MatrixDemo.Revenue` data column in the cell.

A prepared matrix will be as follows:

| Employee        | Year      |           |           |          | Total     |
|-----------------|-----------|-----------|-----------|----------|-----------|
|                 | 1999      | 2000      | 2001      | 2002     |           |
| Andrew Fuller   | 3 900,00  | 2 100,00  |           | 1 800,00 | 7 800,00  |
| Janet Leverling | 6 100,00  | 3 200,00  |           |          | 9 300,00  |
| Nancy Davolio   | 3 300,00  | 2 700,00  | 3 100,00  | 1 700,00 | 10 800,00 |
| Steven Buchanan |           | 3 999,00  | 8 100,00  |          | 12 099,00 |
| Total           | 13 300,00 | 11 999,00 | 11 200,00 | 3 500,00 | 39 999,00 |

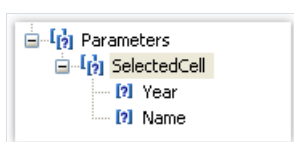
As seen, the value of a cell is the sum of employee's sales for the whole year. Let us create a detailed report which will be displayed when we click the cell. In our case the detailed report can contain the sales of a selected employee for every month of a selected year.

How to connect a cell with data, on which basis it was printed? Each cell of the matrix has got its own address. This is a combination of the values from the column and row headers. In our example, the address of the cell is a combination of the year and name of the employee. Exactly this data can be passed to the detailed report. How can this be done? Very simple: set the hyperlink, showing only the report name and name of the parameter. Parameter values do not need to be indicated: for a matrix cell, FastReport itself forms the value and passes it into the parameter.

Assuming that, we have clicked on the top left cell, containing the number 3900. This is the sum of the sales of the employee named "Andrew Fuller" for the year 1999. What form is used to pass this value into the parameter? FastReport combines column and row values, by using a separator:

```
1999;Andrew Fuller
```

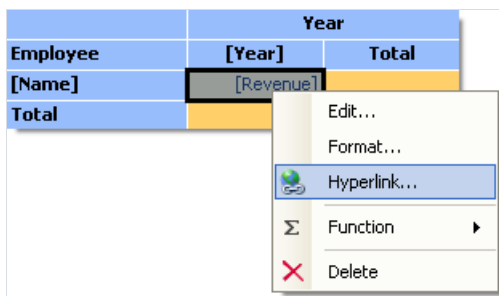
Does it mean that we must extract the value of the year and the name of the employee from this string, convert the year into the int, and use these values for data filtering? No, it's much simpler. All that we need to do is to create a parameter that has nested parameters. You can learn about this in the ["Data"](#) chapter. In the given case, parent parameter can be like this:



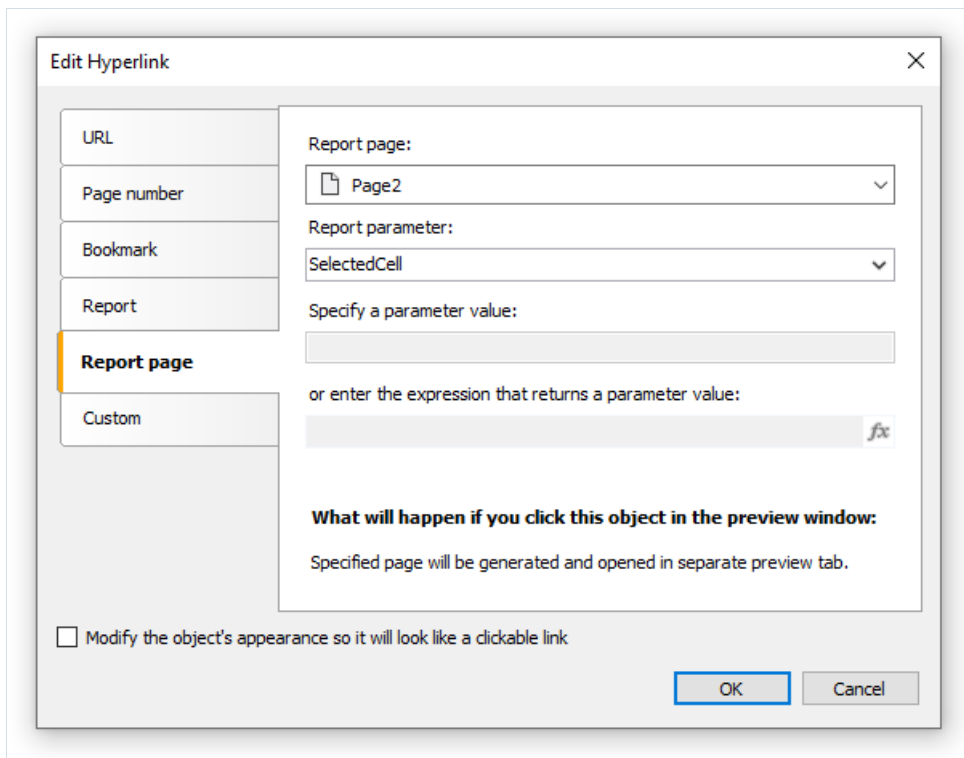
When creating the parameter, consider the following moments:

- you don't need to set up the parent parameter. Just give it the name;
- a parent parameter must have as many nested parameters, as there are values passed from the matrix. In the given case, there are two values;
- order of the nested parameters must correspond with the order of the values passed from the matrix. In our case, the year will be passed in the first parameter, and the employee name will be passed in the second parameter;
- nested parameters can be named as you wish, but it is better to give them names which correspond with the names of the matrix elements;
- it is very important to set the data type for every nested parameter correctly. Data type must correspond with the value, which is passed into the parameter. In our case, the first parameter (Year) must be an integer type ( `Int32` ), and the second (employee name) - `String` .

After we have clarified all the needed things, we will create the report. Select the cell of the matrix and call the hyperlink editor:

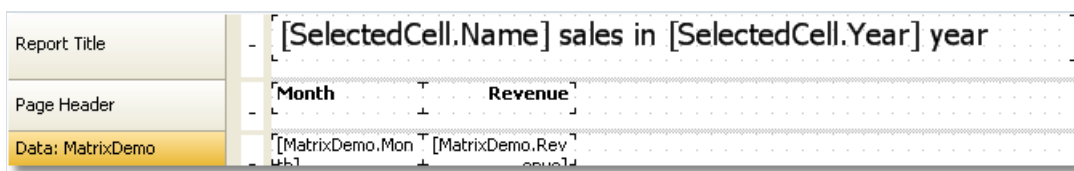


In the hyperlink configuration, indicate the parent parameter as a report parameter (in our example - `SelectedCell`):



FastReport passes the values into the `SelectedCell.Year` and `SelectedCell.Name` nested parameters. These values will be converted into data types, indicated in the parameter configuration - this is why it is important to configure parameter data types correctly.

Detailed report is placed on a separate page of the main report and uses the same data source:



In order to show the sales of a chosen employee for a chosen year, set up the filtering. For this, open the "Data" band editor and indicate the following filtering condition:

```
[MatrixDemo.Year] == [SelectedCell.Year] && [MatrixDemo.Name] == [SelectedCell.Name]
```

The report is ready. Run it for execution and click on the top left cell. A detailed report will be opened, having the following data:

### Andrew Fuller sales in 1999 year

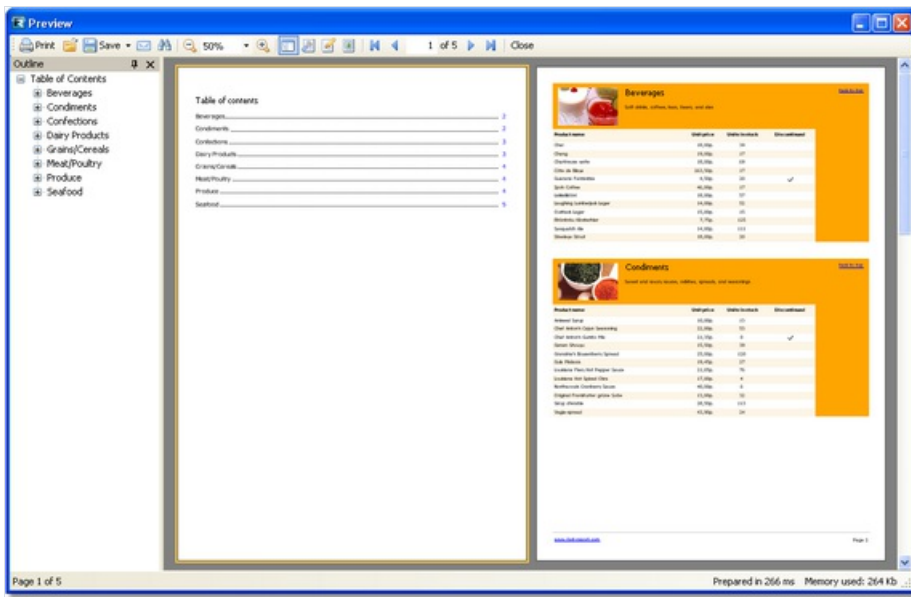
| Month | Revenue  |
|-------|----------|
| 10    | 1 900,00 |
| 11    | 2 000,00 |

As seen, the sum of the values (1900+2000) corresponds with the cell of the matrix, on which we clicked.

## Example 4. Report with table of contents, navigation and outline

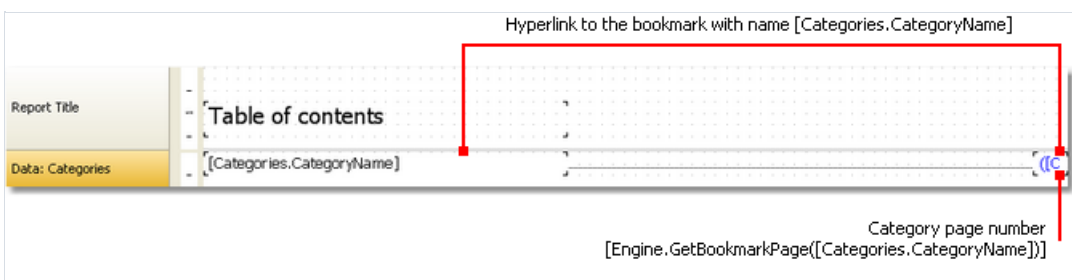
In this example, we will look at creating a report, which has the following features:

- on the first page it will print the "Table of Contents" (TOC) which is interactive, i.e. you can click its elements to navigate to the corresponding page;
- in the preview window, it will display the outline, which is interactive as well.

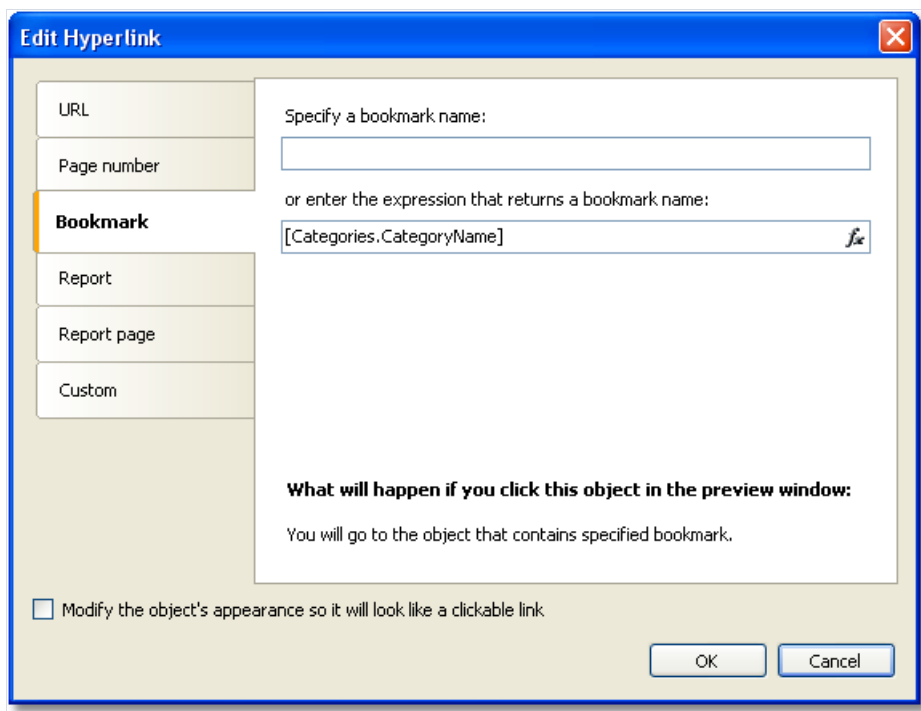


The report will use the "Categories" and "Products" tables. In the TOC, we will print the categories list. The rest of report will print the categorized list of products. Report template will be made up of two pages: the first page will be used to print the TOC; the second one is the main part of the report.

We will discuss the TOC firstly. Create a new report and add "Categories" and "Products" data sources into it. Connect the "Data" band to the "Categories" table and place the objects in the following way:



In order to make the TOC objects interactive, configure its **Hyperlink** property:



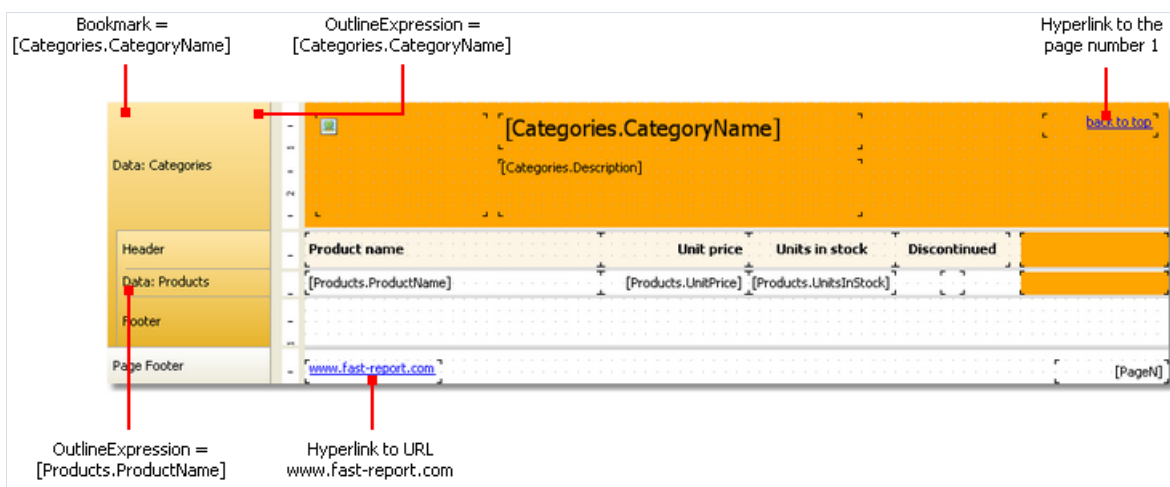
Indicate the category name as a bookmark. We will define the object's bookmark later.

In order to print the page number in the TOC, you need to do the following:

- enable the "double pass" setting of the report. This can be done in the "Report|Options..." menu. This needs to be done, because the TOC page is printed before other pages. At this moment FastReport does not know where the categories will be printed;
- use the `Engine.GetBookmarkPage` function, which returns page number for the specified bookmark. In our case, we use the `[Categories.CategoryName]` expression as a bookmark name, so the function call will be as follows:

```
[Engine.GetBookmarkPage([Categories.CategoryName])]
```

On the second page of the report, we will create a master-detail report as shown in the figure below:



Set up the bookmark we navigate to when clicking on an element in the TOC. For this, select the first "Data" band and indicate the following expression in its **Bookmark** property:

```
[Categories.CategoryName]
```

To set up the report outline, do the following:

- select the first report page. This can be done by switching to the page;
- in the "Properties" window, set the following value to the `OutlineExpression` property:

```
"Contents"
```

- switch to the second report page;
- select the first "Data" band and set its `OutlineExpression` property:

```
[Categories.CategoryName]
```

- select the second "Data" and set its `OutlineExpression` property:

```
[Products.ProductName]
```

# Report inheritance

Often we have many reports with the same data in it - for example, same header/footer with company logo and some data - email, address etc. Now imagine the situation, that you need to change some company data - for example, email. You have to do this in each report! To avoid this, you can use report inheritance. What is it?

For example, you have some common elements in each report (logo, company name, email etc). These elements are typically placed on the report title and/or page header. You can create a base report that contains only common elements. All other reports will use base report and thus will contain such common elements plus own elements defined in a report.

In case you need to change something (logo, email) you just open the base report and make necessary changes. All other reports that inherit from a base, will be changed automatically. In fact, when you open a report that is inherited one, the base report is opened first, then the inherited one.

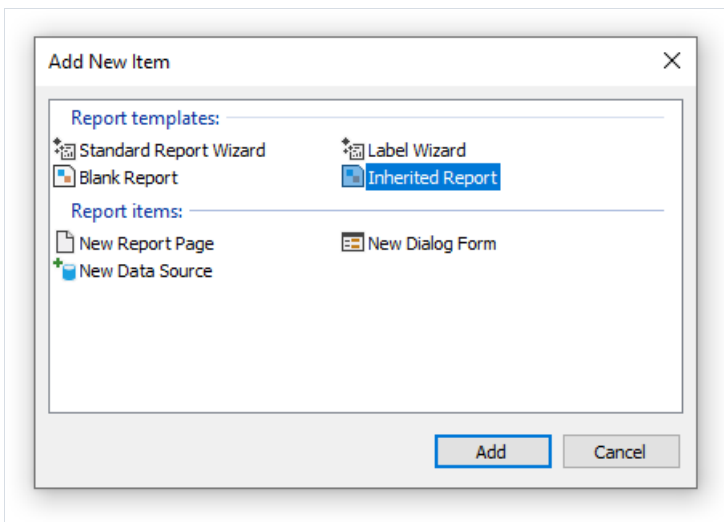


# Creating a report

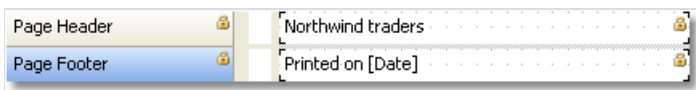
To use the inheritance, you need to do the following:

- create a base report and save it to a file;
- create a new report that inherits from base.

To create an inherited report, choose the "File|New..." menu item, then choose the "Inherited report" item in the window:



You will be asked to select a base report file. That file must be created at this moment. After that, the base report will be loaded into the designer. You can change it as you want. You see that objects from the base report are marked with the "lock" sign:



That means, you cannot delete such objects, rename it or move to another band.

You can add new objects or bands, change the object's appearance, size and location. When you have done, save the report.

# Changing the base report

Let us look what happens if we change the base report. We can:

- delete the object from the base report. This object will be deleted from the inherited report as well;
- add the object into the base report. This object will appear automatically in the inherited report;
- change the size, location, text, appearance of the object. All changes will be reflected in the inherited report, in case where this object was not changed in the inherited report.

The last point requires some explanations. Let us look at two examples of using inheritance. In the first example, we will do the following:

- create a base report which contains the Text1 object;
- create an inherited report and just save it, without changing anything;
- open the base report and move the Text1 object;
- open the inherited report and we will see that the Text1 object is moved as well.

In the second example, we will do the following:

- create a base report which contains the Text1 object;
- create an inherited report;
- in the inherited report, move the Text1 object to a new position and save the report;
- open the base report and move the Text1 object to different position;
- open the inherited report and we will see that the Text1 object is not moved.

It happens because we have changed the object in the inherited report. This change was saved in the inherited report file. Now, if we change the original object in the base report, it will be ignored in the inherited report. In this case, the new object's location will be ignored. All other changes (such as text color, for example) will still be reflected in the inherited report.

This behavior will become clear if we look at the contents of inherited report file. For example, this is how the original object is saved in the inherited report, in case this object was not changed:

```
<inherited Name="Text1"/>
```

If we change the object's location in the inherited report, it will be saved like this:

```
<inherited Name="Text1" Left="255.15" Top="28.35"/>
```

When opening the inherited report, FastReport will load all object's properties, defined in the base report, plus properties, saved in the inherited report.

# Limitations

The report inheritance was designed to meet the following goal: save common report elements such as headers and footers in separate files, and reuse them in inherited reports. Do not try to use the inheritance to perform more complex tasks. In particular, avoid to do the following:

- do not inherit the report from the inherited one (i.e. do not inherit twice);
- do not use complex objects such as Table and Matrix, in the base report;
- do not use script in the base report;
- do not use parameters in the base report.

# Reports with charts

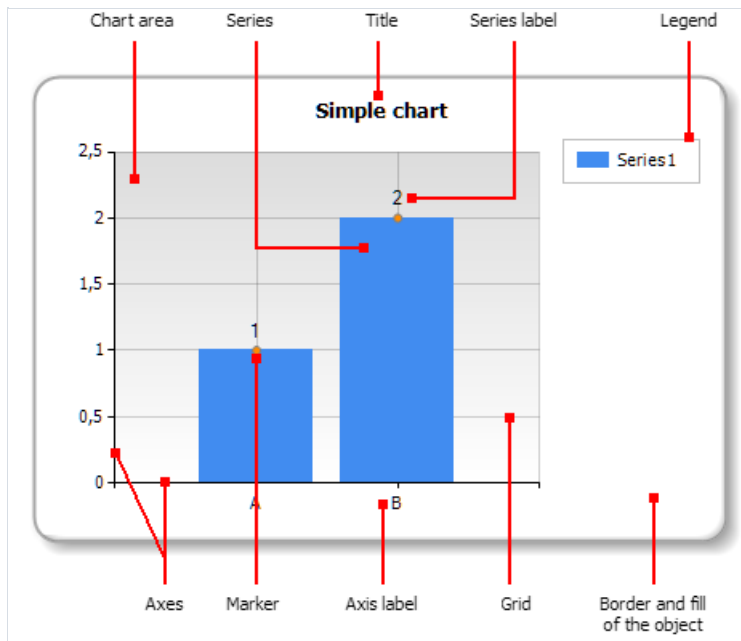
FastReport uses the `System.Windows.Forms.DataVisualization` library to build diagrams.

Let us highlight some features of Microsoft Chart:

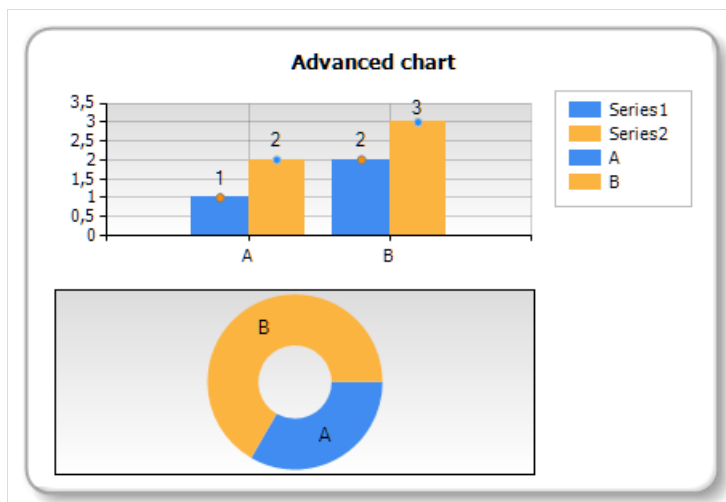
- more than 30 types of series (bars, columns, areas, lines, bubbles, pie, circular, financial, pyramidal, ranges);
- 3D support;
- supports several series of different types in one chart;
- full control over appearance and behavior of each chart element.

# Chart elements

Microsoft Chart consists of the following elements:



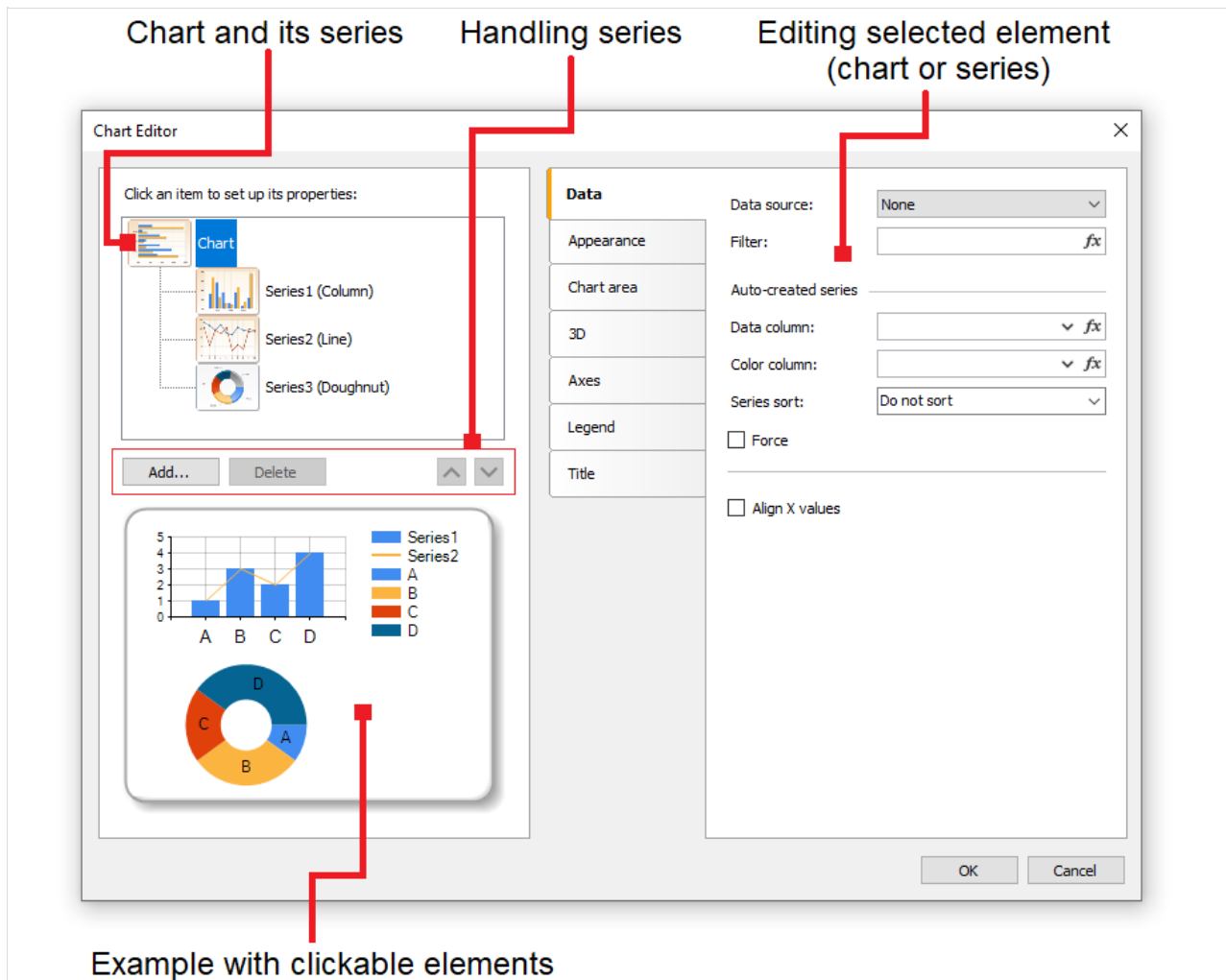
One chart may have one or several chart areas. One chart area may contain one or several series. Below you can see the chart which contains two chart areas (the first area contains two series, the second area contains one series):



Some series (for example, pie series) require exclusive chart area.

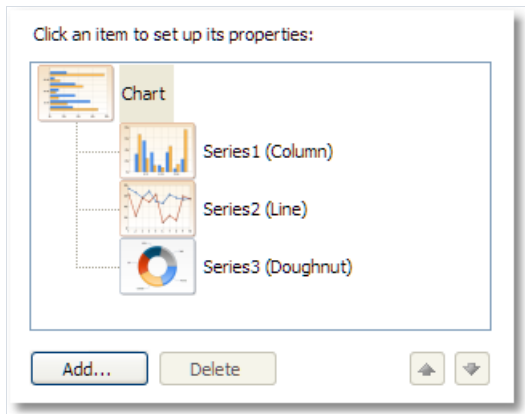
# Chart editor

The "Chart" object contains numerous settings which can be handled in the chart editor. To invoke the editor, double click the "Chart" object:

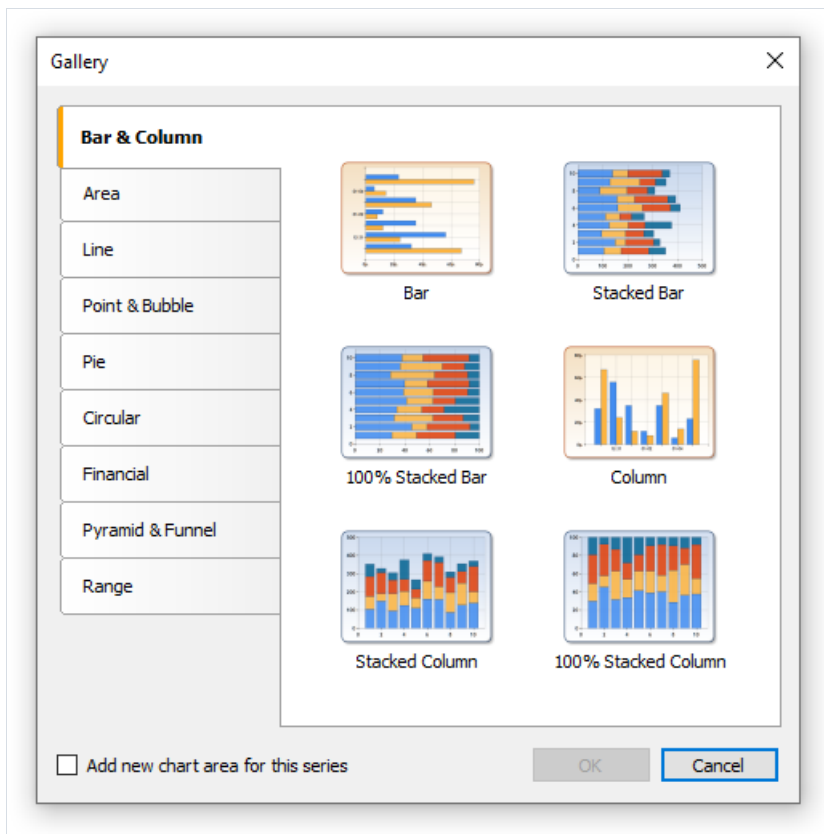


# Handling series

The "Chart" object can contain one or several series. Series list is displayed in the editor:



To add a new series, press the "Add..." button. You will see the "Gallery" dialog:



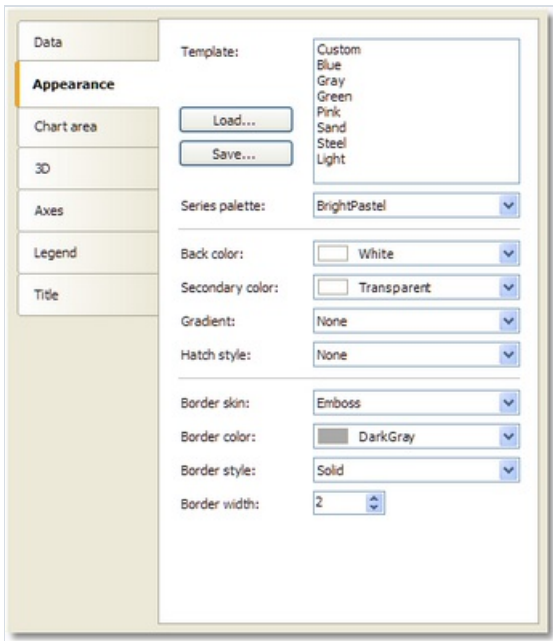
Select the needed category, then - needed series type. If you want to place the series in its own chart area, check the "Add new chart area for this series" checkbox. For some series types (such as pie, circular, financial, pyramidal) the new chart area is added automatically regardless of this checkbox state.

To delete the series, press the "Delete" button. To change series order, use "Up" and "Down" buttons.

# Setting up the appearance

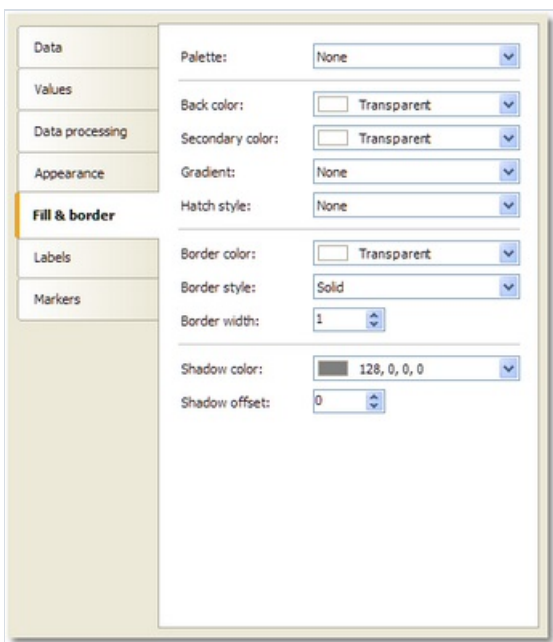
Using the chart editor, you can set up appearance of each chart element. All properties (more than 100) are splitted in several categories. Some of them are specific to "Chart" object, while others are part of series.

If you choose the "Chart" object from the series list, you will see the following property pages:



- "Appearance" - border and fill of the chart;
- "Chart area" - border, fill, shadow;
- "3D" - 3D settings;
- "Axes" - setup appearance of axis, its title, labels, grid, markers, custom labels and strips;
- "Legend" - style of legend, docking, border, fill, shadow and font;
- "Title" - style of title, docking, border, fill, shadow, font.

If you choose the series object from the series list, you will see the following property pages:



- "Appearance" - some settings specific to the selected series type;



- "Fill & border" - fill and border of the series values;
- "Labels" - series labels. You can choose label type, font, color and fill;
- "Markers" - series markers. You can choose marker type, its color and border.

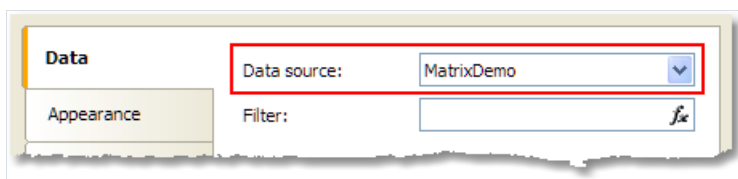
# Connecting chart to data

You can fill the chart with data in several ways:

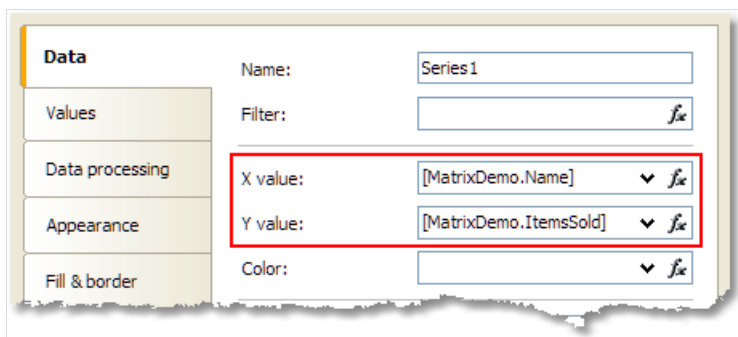
1. Use data source. To do this, you need to indicate the data source for the "Chart" object and connect each series to data columns.
2. Use fixed values for each series.
3. Fill the object with data using the script.

To connect the chart to a data source, follow these steps:

- select the "Chart" object in the series list;
- switch to the "Data" tab;
- choose the data source:

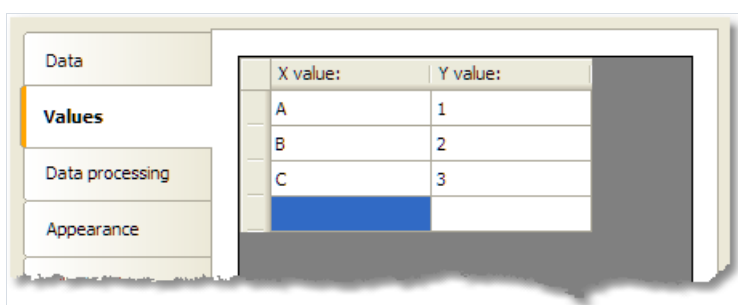


- if necessary, set the data filter expression. This filter will be applied to all chart series;
- select the series in the series list;
- switch to the "Data" tab;
- choose data columns for each series value. Depending on series type, it may have two or more values. Most series types have two values - X value and Y value:



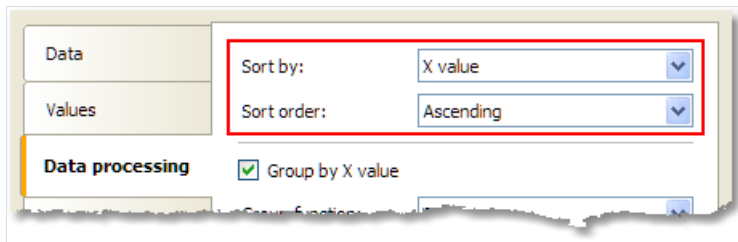
- if necessary, set the data filter expression. This filter will be applied to current series only;
- in the "Color" control, you may indicate a data column which returns a color value.

You may also provide list of values for the series. In this case, the data connection is not needed. To do this, select a series in the series list and switch to the "Values" tab. Fill the table with values:



# Sorting the data

By default, the chart object displays data in natural order. You can change the sort order; to do this, select the series from the series list and switch to the "Data processing" tab:



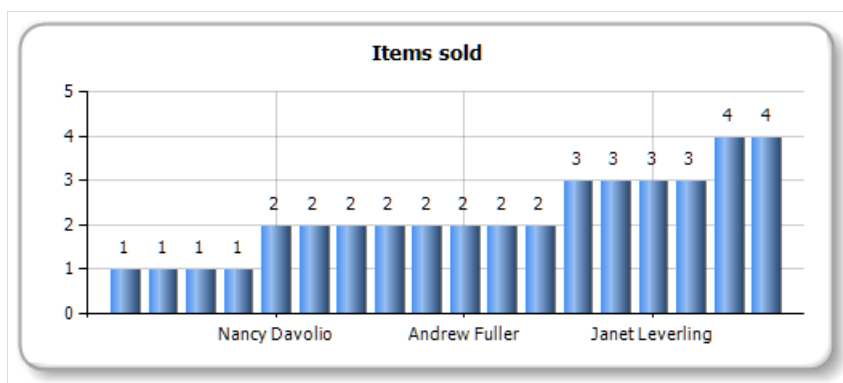
You can choose one of the sort modes - do not sort, sort by X value, and sort by Y value.

# Grouping the data

Sometimes we face a problem when series is filled with data with several identical X values. For example, the MatrixDemo table, which is used to demonstrate charts, has the following data:

| Name            | Year | Month | ItemsSold | Revenue |
|-----------------|------|-------|-----------|---------|
| Andrew Fuller   | 2002 | 1     | 2         | 1800    |
| Andrew Fuller   | 1999 | 10    | 2         | 1900    |
| Andrew Fuller   | 1999 | 11    | 2         | 2000    |
| Andrew Fuller   | 2000 | 2     | 2         | 2100    |
| Janet Leverling | 1999 | 10    | 3         | 3000    |
| Janet Leverling | 1999 | 11    | 3         | 3100    |
| Janet Leverling | 2000 | 3     | 3         | 3200    |
| ...             |      |       |           |         |

If we try to build a chart based on this data (for example, employee's sales - set X value to Name column, Y value to ItemsSold column), we will get the following wrong result:

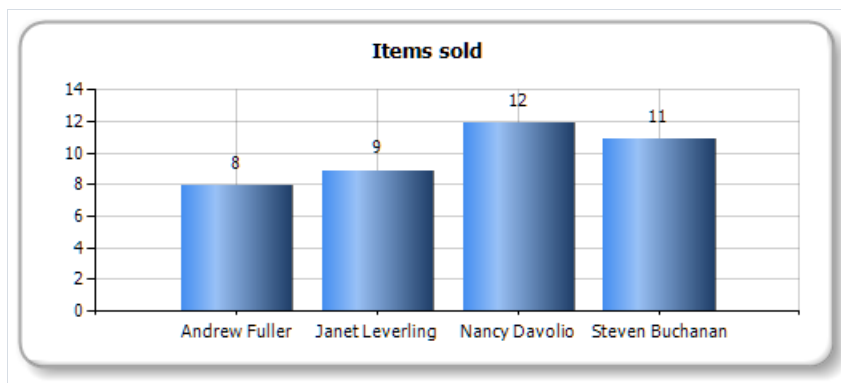


In this situation, we need to group the same employees into one value. To do this, select the series in the series list, and switch to the "Data processing" tab. Select the group type - "X value" and choose "Sum" as group function:

The screenshot shows the 'Data processing' tab with the following settings:

- Group by: X value
- Group interval: 1,00
- Group function: Sum

As a result, all identical employees will be grouped into one value, their sales will be summarized. You will see the following result:



# Collecting the data

This instrument for data processing allows to collect several series value into one value. You can choose one of the following algorithms:

| Algorithm                   | Description  |
|-----------------------------|--|
| <b>TopN</b>                 | Only top N values are displayed. All other values are collected and displayed as "others" value (you can choose the label for this value). |
| <b>BottomN</b>              | Bottom N values are displayed. If the text for the collected value is not set, this value is not displayed.                                |
| <b>Less than value</b>      | Series values less than specified value, are collected and displayed as "others" value.  |
| <b>Less than percent</b>    | Series values less than specified percent, are collected and displayed as "others" value.  |
| <b>Greater than value</b>   | Series values greater than specified value, are collected and displayed as "others" value.   |
| <b>Greater than percent</b> | Series values greater than specified percent, are collected and displayed as "others" value.   |

For example, to display top 5 values, set up the series in the following way:

Collect data:

TopN

Value:

5

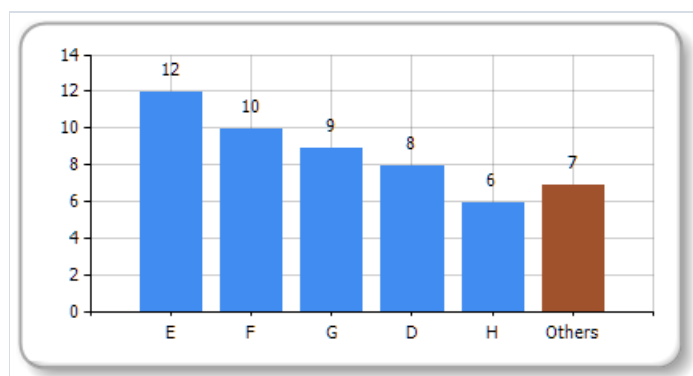
Collected text:

Others

Collected color:

Sienna

You will get the following result:



# Exploding the values

For pie-type series, you can explode some values. To do this, select the series in the series list and switch to the "Data processing" tab:

Explode:

Biggest value

▼

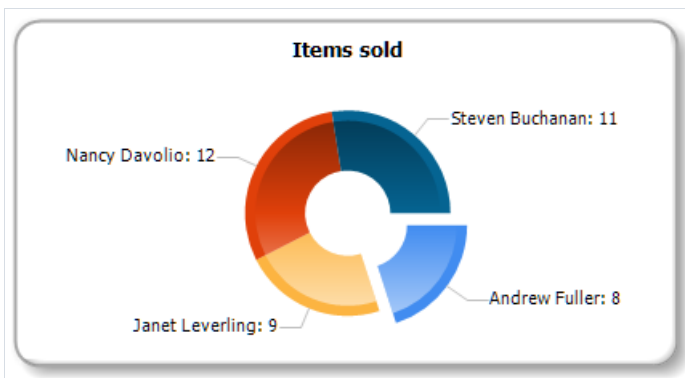
Value:

*fx*

You can use one of the following explode types: biggest value, lowest value and specific value. If you choose the latter mode, you have to specify a value which you want to explode. It may be any expression (see the ["Expressions"](#) chapter for details). For example, if you need to explode Andrew Fuller's value, use the following expression:

"Andrew Fuller"

You will get the following result:



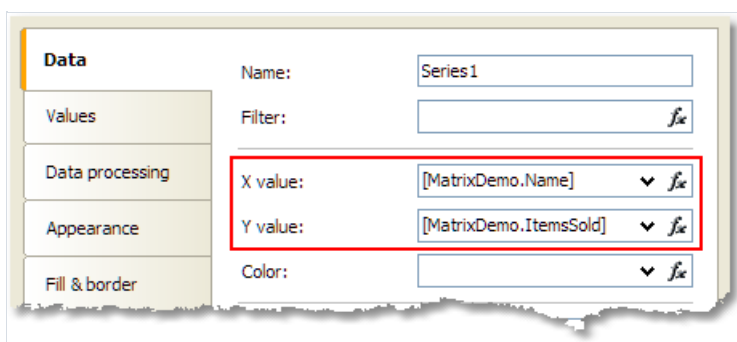
# Setting up auto-series

You can set up the chart so it will create new series automatically, depending on data in a data source. To set up auto-series, do the following:

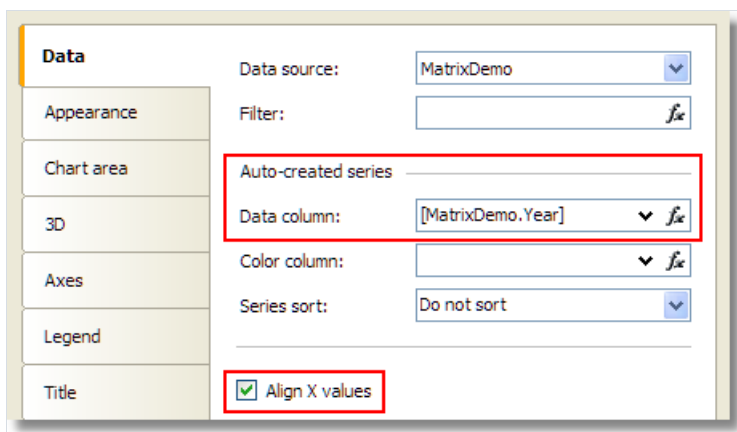
- create one series and set up its properties. This series will be used as a template for all new series;
- select "Chart" object and set up the auto-series data column. The value of this column will be a name of new series. If there is no series with such name yet, the new series will be added.

Let us demonstrate how to create auto-series. We will use the MatrixDemo data table to get a chart of employee's sales per year. One series will represent one year. To do this:

- connect the chart to the MatrixDemo data source;
- create one series and set up its data:



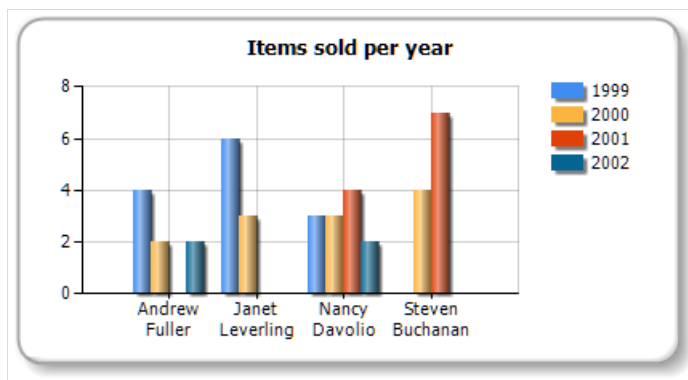
- on the "Data processing" tab, check the "Group by X value" checkbox. It is necessary because our data source has several employees with the same name;
- select the chart in the series list and set up its auto-series column on the "Data" tab:



- our series may have different number of values (because some employees do not have sales in this particular year). To align series values, check the "Align X values" checkbox.

We will get the following result:





# Interactive charts

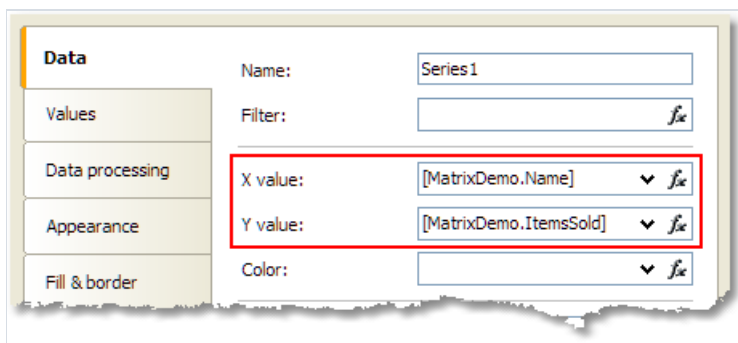
The chart, as well as other report objects, can be interactive. You can set up chart in the way, that when you click its value, another (detailed) report will be executed and shown. To do this, you need to set up the "Hyperlink" property as described in the "[Interactive reports](#)" chapter. The chart will pass the value to a hyperlink by itself, when you click on its element.

Let us observe the "Charts/Interactive Chart" report from the FastReport demo program.

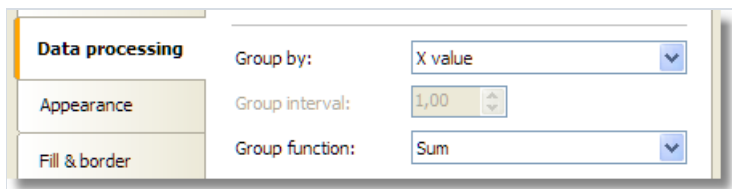
Create a report with two pages. The first page will contain a chart, the second one will contain a detailed report that will be displayed when you click a chart value.

Place the "Chart" object on the first report page and set up its properties in the chart editor:

- select the "Chart" element from the series list and choose the MatrixDemo data source;
- select the series from the series list and set up X and Y values: `X = [MatrixDemo.Name]` ,  
`Y = [MatrixDemo.ItemsSold]` :



- switch to the "Data processing" tab and select the group type - "X value":



On the second report page, create the list-type report:

- in the "Data" window, create a new report parameter called `SelectedEmployee` ;
- create the following report layout:

|                  |                           |                   |                    |                        |
|------------------|---------------------------|-------------------|--------------------|------------------------|
| ReportTitle      | [SelectedEmployee] orders |                   |                    |                        |
| Page Header      | Name                      | Year              | Month              | ItemsSold              |
| Data: MatrixDemo | [MatrixDemo.Name]         | [MatrixDemo.Year] | [MatrixDemo.Month] | [MatrixDemo.ItemsSold] |
| Report Summary   | Total:                    |                   |                    | [TotalItems]           |

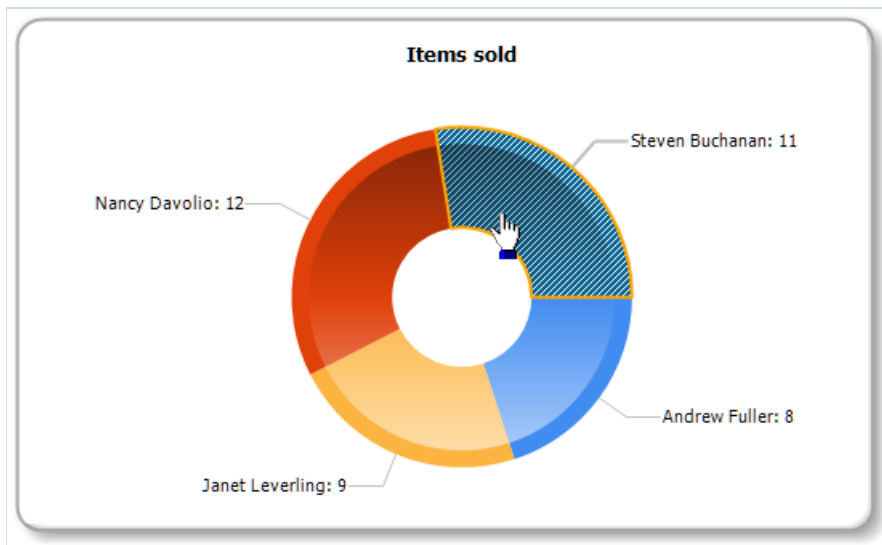
- open the data band editor and indicate the following filter condition:

```
[MatrixDemo.Name] == [SelectedEmployee]
```

Now set up the hyperlink of the "Chart" object:

- in the context menu of the "Chart" object, select "Hyperlink...";
- choose the hyperlink type - "Report page";
- choose the second report page and indicate the parameter name - `SelectedEmployee` .

The report is finished. Run it and move the mouse to any chart value. This value will be visually selected, and the mouse cursor will change its shape:



If you click the value, you will see the following detailed report:

| Steven Buchanan orders |      |       |           |
|------------------------|------|-------|-----------|
| Name                   | Year | Month | ItemsSold |
| Steven Buchanan        | 2001 | 1     | 3         |
| Steven Buchanan        | 2001 | 2     | 4         |
| Steven Buchanan        | 2000 | 1     | 4         |
| Total:                 |      |       | 11        |

# Reports with maps

The MapObject component is intended for displaying 2D maps in "ESRI shapefile" format. You can find information on this file format at:

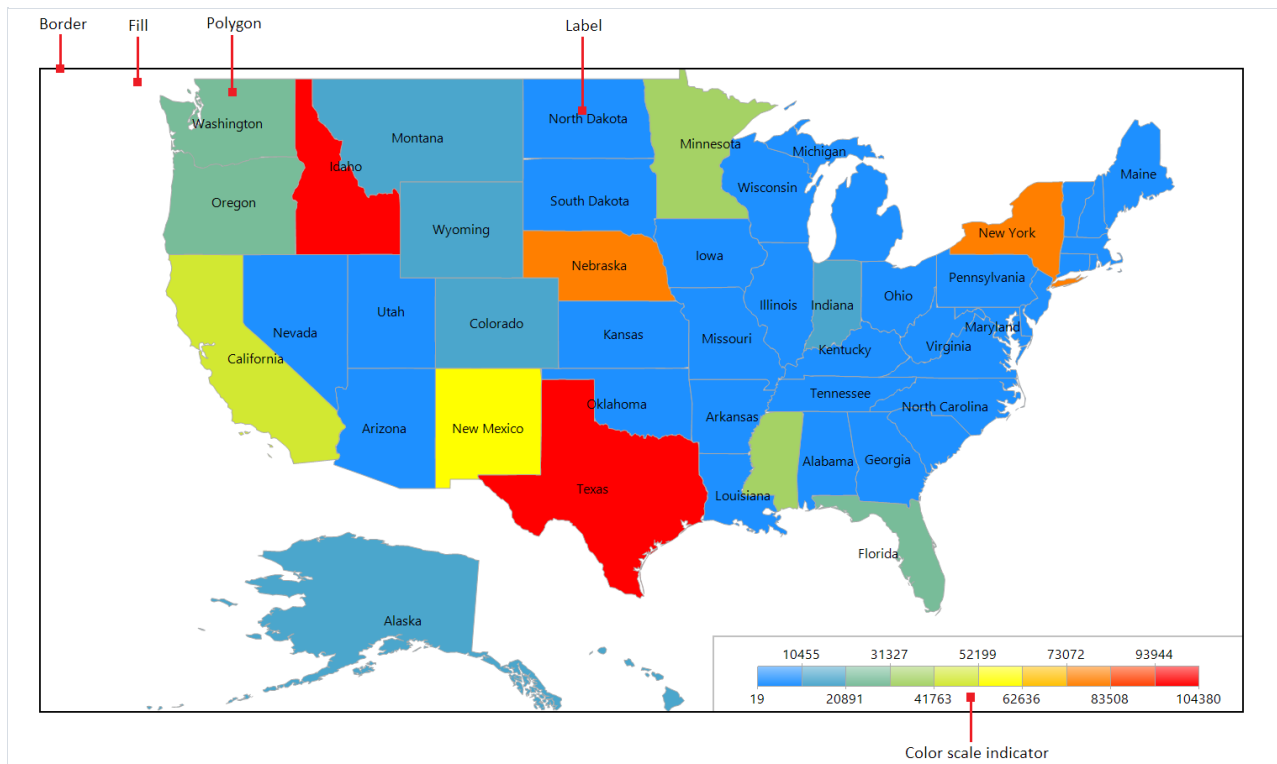
<http://en.wikipedia.org/wiki/Shapefile>

Two files are required:

- .shp (shape format - the feature geometry itself);
- .dbf (attribute format - columnar attributes for each shape).

# Map elements

The "Map" object consists of the following elements:



One "Map" object can display one or more layers. Each layer contains its own map.

# Controlling the map with the mouse

In the report designer and in the preview window the map can be controlled by the mouse:

- zoom in and out using the mouse wheel;
- pan the map by dragging the left mouse button;
- change a polygon's properties in the "Properties" window by clicking the left mouse button inside a map polygon.

The minimum and maximum zoom values can be set in the `MinZoom` and `MaxZoom` properties. These properties are changed in the "Properties" window.

The map cannot be controlled in the asp.net report viewer.

## The "Map" object editor

The "Map" object has many properties, which can be set in the object editor. The editor is opened by double-clicking on the object, or from its context menu:

×

Edit Map

Click an item to set up its properties:

Map

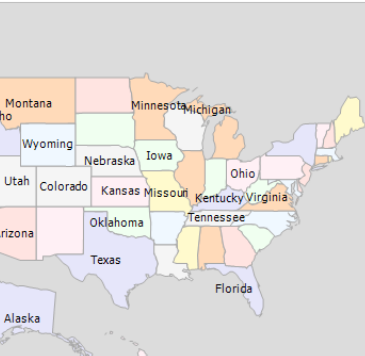
Layer 1

Add...

Delete

⬆

⬇



Data

Appearance

Color scale

Size ranges

Labels

Data source:None

Filter:fx

Spatial data:

Column:NAME

Value:fx

Analytical data:

Value:fx

Function:Sum

Zoom the polygon:

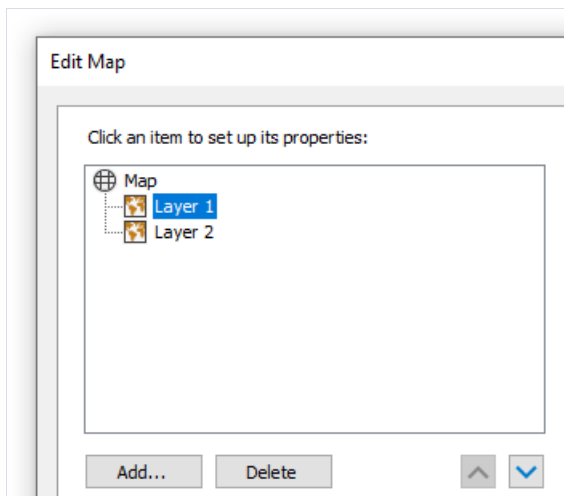
Value:fx

OK

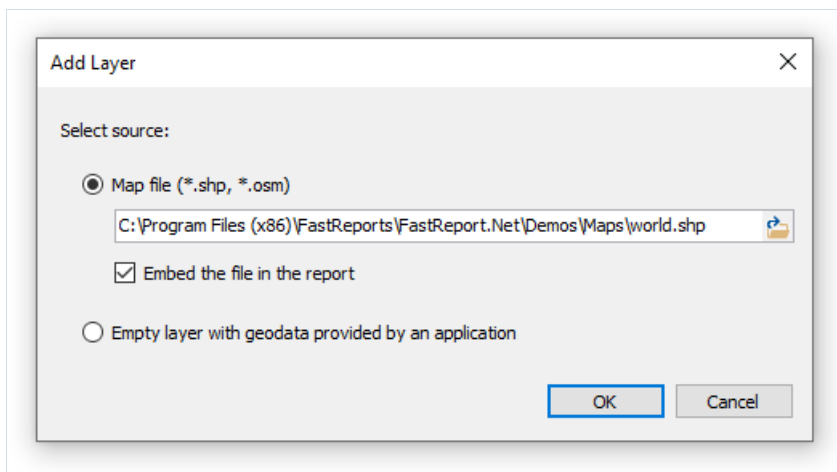
Cancel

# Adding map layers

The "Map" object can contain one or more layers. The layer structure is displayed in the upper left part of the editor window:



To add a new layer, click on the "Add..." button, which opens the following dialog:



At this point, choose the type of the layer, from:

- the map from an ESRI shapefile. This is the most commonly used type for maps. For example, you can display the world map and highlight some countries in color, depending on the sales level for these countries;
- the geodata from your application. Your application must provide the following data: latitude, longitude, name and value. This data is displayed as a small point on the layer. The point can have a caption and can also have a variable size/color, dependent on the value provided. In practice, this type of layer is used together with a base layer of "ESRI shapefile" type. The base layer (the first one) is configured to display a country map (for example) and the second layer (geodata from an application) displays points - city names where sales occurred. The color and size of a point can be configured to reflect sales level.

If the "ESRI shapefile" layer type is chosen, select how the map data is stored, from:

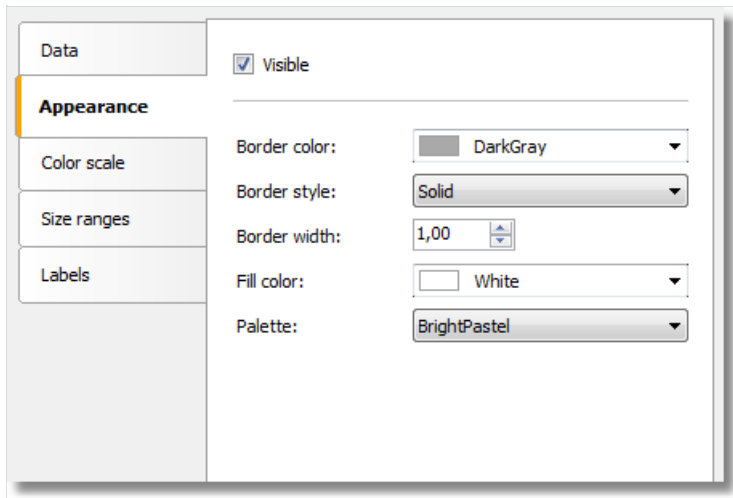
- the whole map data is embedded in the report file. In this case the report file (.frx) may be large;
- the report file holds a reference to the map files (.shp/.dbf). This mode is useful if several reports use the same map files.



Large map files (more than 30Mb) or map files containing a lot of polygons (more than 20,000) slow down report generation.

# Setting up the layer appearance

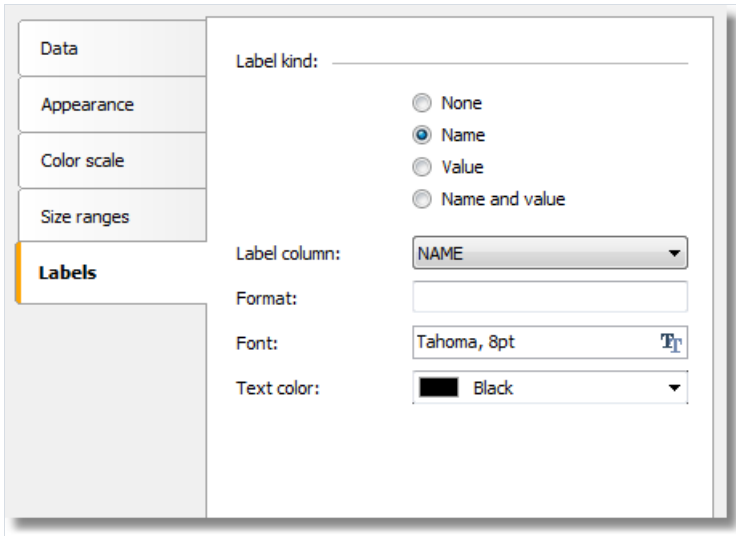
To set the layer appearance, choose the layer and switch to the "Appearance" tab:



Set the border color and style of the map polygons and choose the color palette. Note that the palette is ignored if you configure the color scale (more about this later).

# Setting up label display

The map can display labels, such as country names. Set the label type and appearance on the "Labels" tab:



If the "ESRI shapefile" layer type is chosen set the field name which contains the displayed information. As a rule, it's a "NAME" field. The world map included in the FastReport demo program contains the following fields:

- NAME (eg: Germany)
- ABBREV (eg: Ger.)
- ISO\_A2 (eg: DE)
- ISO\_A3 (eg: DEU)

Other maps will have a different set of fields.

If the "application geodata" layer type is chosen set the minimum zoom value for displaying the labels. The default value is 1, meaning the labels are always displayed.

# Connecting the map layer to data

Most reports use the "Map" object to display analytical data, for example sales levels in various countries. So a map layer must be connected to a data source using the map object editor window. Select the layer from the layers tree and switch to the "Data" tab. Data appropriate for the map layer type must be provided:

- for map layer type "ESRI shapefile" the "Data" tab looks like:

The screenshot shows the 'Data' tab of a map object editor. On the left, there's a sidebar with tabs: 'Data' (highlighted), 'Appearance', 'Color scale', 'Size ranges', and 'Labels'. The main panel is divided into sections for configuring data. 'Data source' is set to 'Sales'. 'Filter' is empty. 'Spatial data' is empty. 'Column' is set to 'NAME'. 'Value' is set to '[Sales.Country]'. 'Analytical data' is empty. 'Value' is set to '[Sales.SalesTotal]'. 'Function' is set to 'Sum'. 'Zoom the polygon' is empty. 'Value' is empty. Each input field has a small 'fx' icon to its right, indicating a formula or selection tool.

The data required is:

- name (eg: the country name);
- numeric value (eg: sales volume in this country).

For example, a "Sales" data source may have this data:

| Country | SalesTotal |
|---------|------------|
| USA     | 500000     |
| Germany | 1200000    |
| Russia  | 300000     |

Set up the "Data" tab in the following way:

- data source - Sales;
- spatial data, column name - select the column containing country names; usually this is the `NAME` column;
- spatial data, value - `[Sales.Country]` ;
- analytical data, value - `[Sales.SalesTotal]` ;
- analytical data, function - `Sum` ; this function is used if you have several values per country.

The "Zoom the polygon" edit box allows zooming of the polygon with a given name, so it occupies the whole "Map" object workspace. For example, to zoom Germany on the world map, type "Germany" (with quotes) in this edit box.

- for map layer type "geodata from an application" the "Data" tab looks like:

And the data required is:

- spatial data - latitude and longitude;
- label (eg: a city name);
- numeric value (eg: sales volume in this city).

For example, a "Sales" data source may have this data:

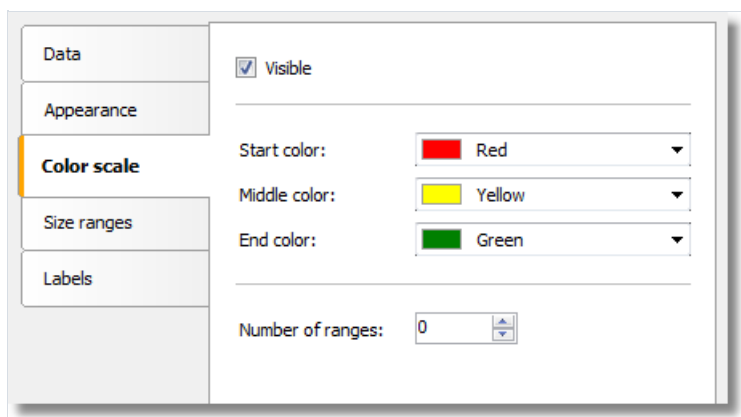
| Latitude | Longitude | CityName | SalesTotal |
|----------|-----------|----------|------------|
| 48.13641 | 11.57753  | München  | 50000      |
| 50.94165 | 6.95505   | Köln     | 36000      |

Set up the "Data" tab in the following way:

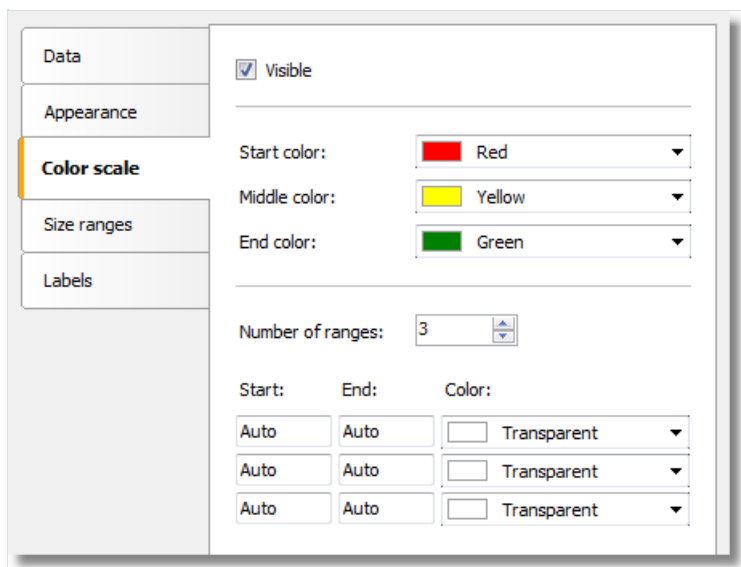
- the data source - Sales;
- spatial data, latitude - `[Sales.Latitude]` ;
- spatial data, longitude - `[Sales.Longitude]` ;
- spatial data, label - `[Sales.CityName]` ;
- analytical data, value - `[Sales.SalesTotal]` ;
- analytical data, function - `Sum` ; this function is used if you have several values per city.

# Highlighting data using colors

After you have finished connecting a layer to its data, the next question is - how to display analytical data (for example the sales volumes in various countries)? The easiest way is to set up labels displaying a country name and its sales (see "Setting up label display"). More visual impact can be achieved by painting each country with a color, dependent on its sales volume. For example, low sales as red, mid sales as yellow and high sales as green. This is done by setting up the color scale on the "Color scale" tab:



The color scale consists of several ranges. Each range has the following properties: min value, max value and color. You can use as many ranges as you need. To set up the color scale set the number of ranges first and then the properties for each range.

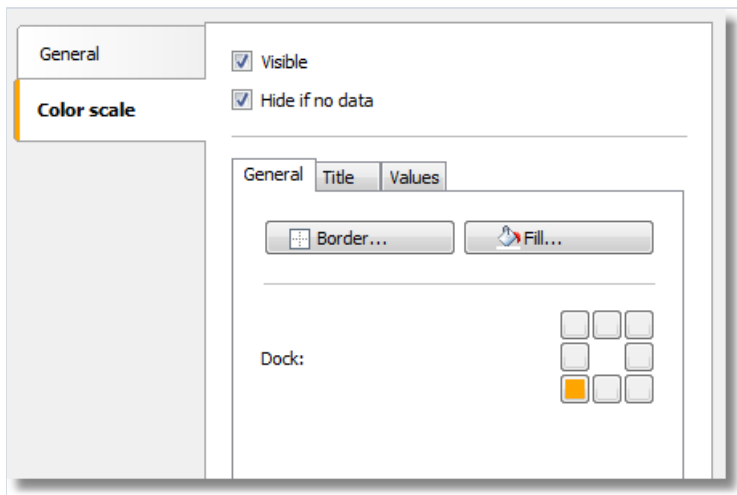


By default all range properties are set to **Auto**, meaning FastReport calculates the minimum and maximum values for each range automatically. The auto color is chosen from three presets ("Start color", "Middle color", "End color"). The **Auto** mode may be suitable in most cases.

When a color scale is set up, an indicator control is displayed in the bottom part of the "Map" object:



To set the appearance and position of the indicator, select the "Map" element in the layers tree control and switch to the "Color scale" tab:



# Highlighting data using size

When using layer type "geodata from an application", the data is displayed as a small circle with a caption. The size of the circle can be bound to the data in the same way as highlighting data using colors. To do this set up the size ranges on the "Size ranges" tab:

|                    |      |       |
|--------------------|------|-------|
| Data               |      |       |
| Appearance         |      |       |
| Color scale        |      |       |
| <b>Size ranges</b> |      |       |
| Labels             |      |       |
| Start size:        | 4    |       |
| End size:          | 20   |       |
| Number of ranges:  | 3    |       |
| Start:             | End: | Size: |
| Auto               | Auto | Auto  |
| Auto               | Auto | Auto  |
| Auto               | Auto | Auto  |

The size ranges have the following properties: min value, max value and size (in pixels). You can use as many ranges as you need. Set the number of ranges first and then set the properties for each range.

By default all range properties are set to "Auto", meaning FastReport calculates the minimum and maximum values for each range automatically. The auto size is chosen from two presets ("Start size", "End size"). The "Auto" mode may be suitable in most cases.



# Data

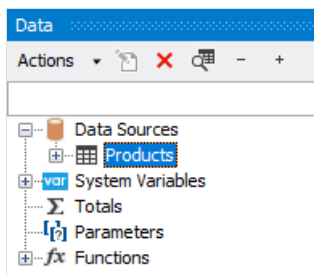
Any report prints some data. In FastReport, you can operate with the following data:

- data sources;
- system variables;
- total values;
- report parameters;
- expressions, containing any of the above mentioned data.

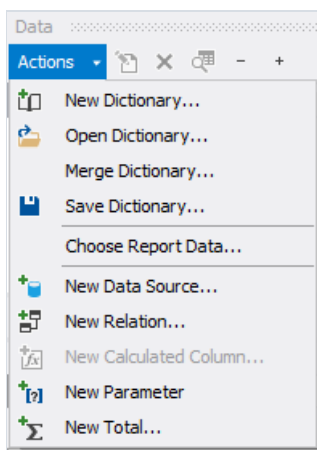
In this chapter, we will look at how to work with these data.

# The "Data" window

All data is accessible from the "Data" window. This window can be shown by choosing the "Data>Show Data Window" menu.



The "Data" window allows to operate with all data elements and also to drag them into the report page. All operations can be done with the help of the toolbar and "Action" menu:



A part of these operations is duplicated in a context menu of the "Data" window. For example, if you select a data source, you can use its context menu to create a calculated column, delete a data source, or view its data.

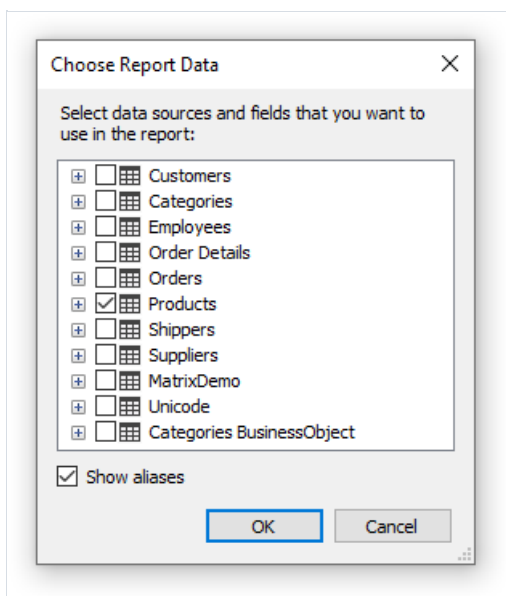
# Data sources

Ordinarily, the data source represents a DB table or SQL query. There can be several data sources in a report. For most of reports, only one data source is needed. A report like master-detail needs two data sources which are connected to each other using a relation (we will learn about it later in this chapter).

Data source has one or several data columns. Each column has a definite data type. To look at column type, select it and open the "Properties" window. Column type is indicated in the `DataType` property. The icon near the column name also helps to determine its type.

There are two ways to define a data source for the report.

The first method - data source is defined in the application and registered in a report. It's up to the programmer who created this application (see details in the "Programmer's manual"). A user should only choose the needed data source in order to use it in a report. It can be done in the "Data|Choose Report Data..." menu.



All data registered in a report is listed in this window. Just tick off those data which are needed in your report. It can be done at any moment while working with a report.

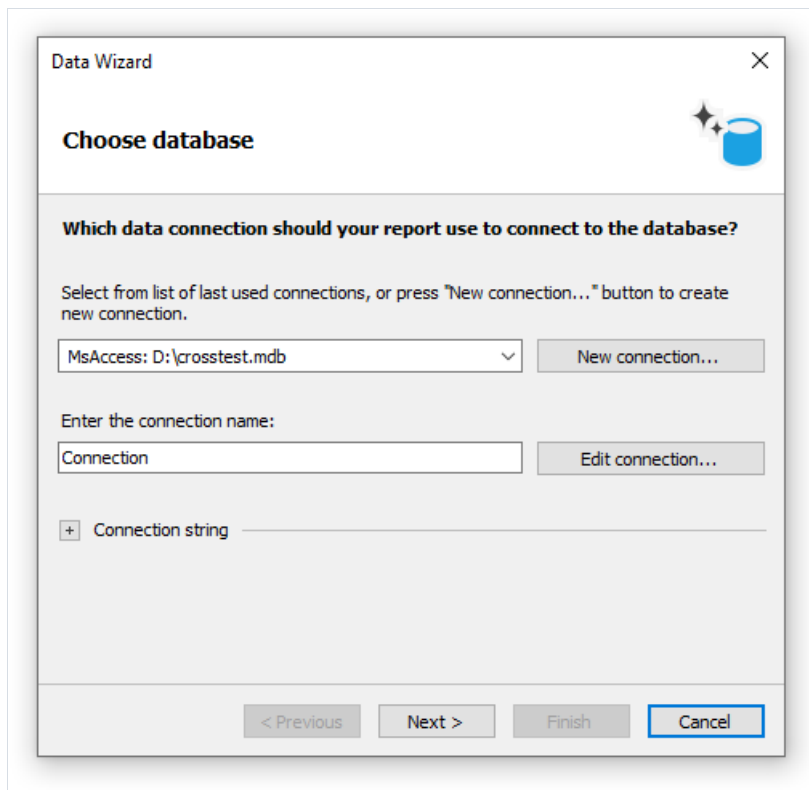
The second method - you create a new data source yourself. It can be a DB table or a SQL query. In such a case, data source definition is saved in a report file.

FastReport allows connecting to many popular DBMS (data base management systems) such as MS SQL, Oracle, Interbase, Access. Besides this, you can use data files which are saved in xml/xsd format.

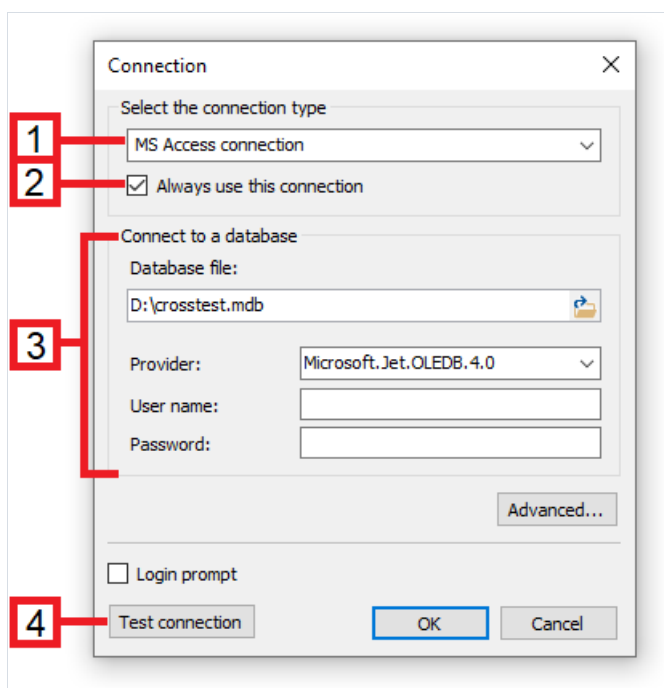
DB table content is not saved in a report file. Instead, the connection string and the data source schema are stored. A connection string can contain such data as login and password, that is why it is kept ciphered in a report file. When needed, you may increase the safety by using own key for data ciphering. In this case a report file can be opened correctly only in your program.

# Creating a data source

To create a new data source, choose the "Data|Add Data Source..." menu item or press the "Actions" button in "Data" window and choose the "New Data Source..." item. You will see the "Data Wizard" window:



First of all, you are offered to create the connection. For that, press the "New connection..." button. You will see a window with connection settings:

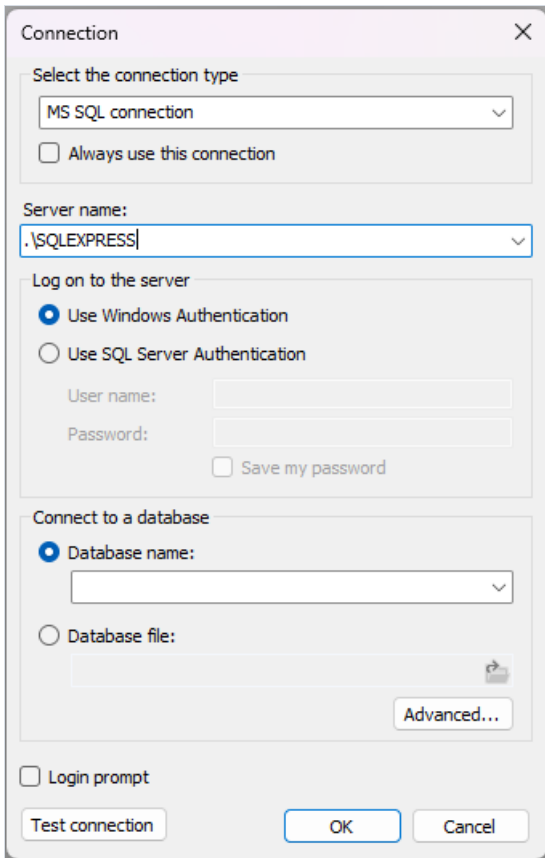


The following elements are shown in the figure:

1. Connection type;
2. If enabled, the chosen connection type will be used by default;

3. Connection settings;
4. Test connection button.

Connection with MS Access data base is shown in the picture. If another type of connection is chosen, then connection settings area (3) will be changed. For example, connection to MS SQL data base has the following settings:

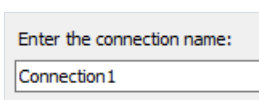


The screenshot shows a 'Connection' dialog box with the following settings:

- Select the connection type:** MS SQL connection (selected in the dropdown).
- Always use this connection:** ☐
- Server name:** .\SQLEXPRESS (selected in the dropdown).
- Log on to the server:**
  - ☒ Use Windows Authentication
  - ☐ Use SQL Server Authentication
  - User name:** (empty text box)
  - Password:** (empty text box)
  - ☐ Save my password
- Connect to a database:**
  - ☒ Database name: (empty dropdown)
  - ☐ Database file: (empty text box with a folder icon)
- Advanced...** button
- ☐ Login prompt
- Test connection** button
- OK** and **Cancel** buttons

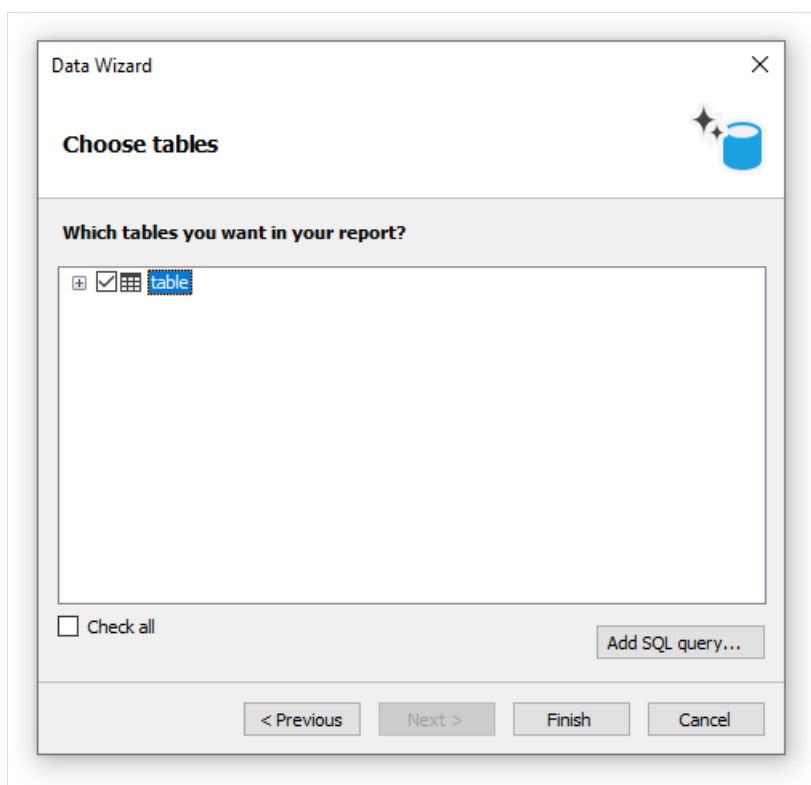
Choose the needed connection type and set up its parameters. After pressing the OK button, the window will be closed and you will return to the data wizard window.

Next, you need to set a connection name. This name will appear in the "Data" window.

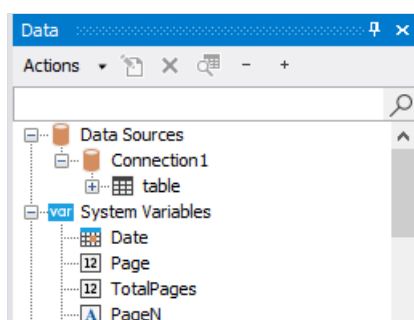


The screenshot shows a small dialog box titled 'Enter the connection name:' with a text box containing 'Connection1'.

Press the "Next" button to continue. Here you will be offered to choose tables which are accessible in the data base:



Tick off the needed tables and close the wizard by pressing the "Finish" button. Now you can see in the "Data" window a connection created by you which contains the chosen data sources:



# Connecting to JSON (JavaScript Object Notation)

This connection allows you to transfer a static JSON file or a URL address, at which it will be received before building a report, as a data source.

## JSON connector

When creating a new connection in the FastReport designer, it looks like this:

| Setting     | Description   |
|-------------|---|
| Encoding    | Sets the encoding in which the request for JSON will be processed, if a link to receive it is provided. |
| JSON or URL | Sets static JSON or a link to receive JSON via API.   |
| JSON Schema | Sets the JSON schema.   |
| Headers     | Sets the required HTTP headers for connecting to JSON via the API (optional).                           |

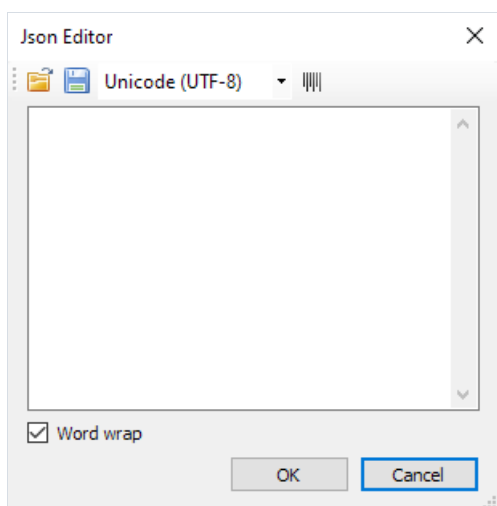
On the right side there are buttons for editing the field in a separate window.

If the JSON Schema value is empty, then when you click OK or Test connection, a schema will be automatically built using JSON.

If you change JSON or URL when the schema is already specified, FastReport will offer to update the schema.

## JSON Editor Window

This window allows you to edit JSON:



Description of the editor window from left to right:

1. Open file - allows you to open JSON and paste the contents of the file into the editor.
2. Save file - allows you to save the contents of the editor to a file.
3. Encoding - sets the encoding in which the JSON file will be opened.
4. Formatting - enables JSON formatting, and also validates JSON against specifications.

Then there is the editor field where you can change the JSON text.

The "Word Wrap" checkbox will enable or disable word wrapping in the editor.

## How the connection works

JSON is not a table, therefore FastReport does not treat a JSON connection as a data table.

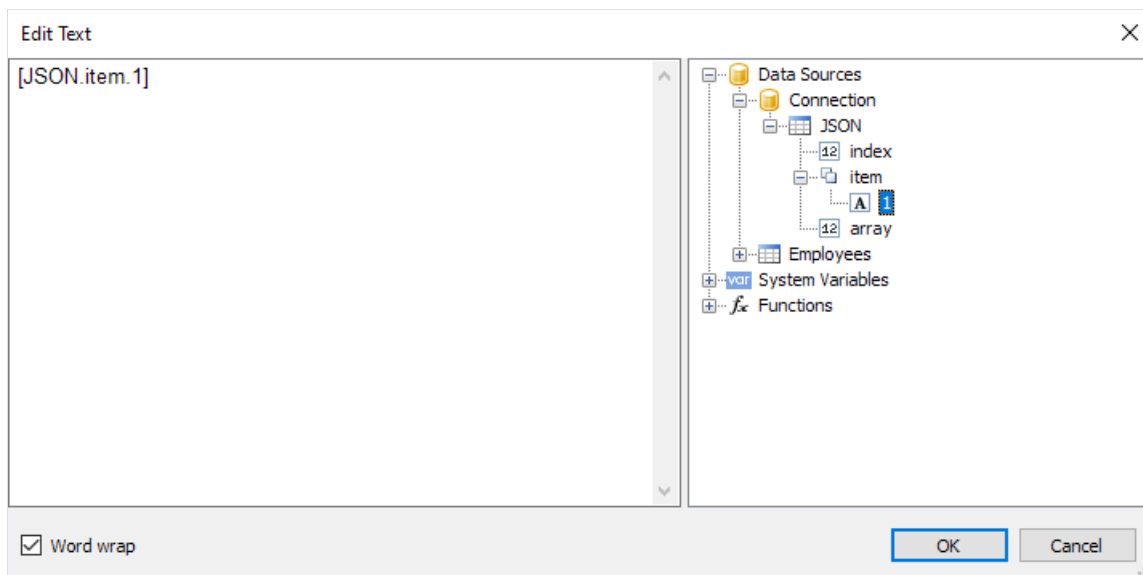
Instead, FastReport treats each JSON array as a hierarchical data source with three fields:

| Field        | Description            |
|--------------|------------------------|
| <b>index</b> | Number of the item     |
| <b>item</b>  | Item                   |
| <b>array</b> | URL to the items array |

Any JSON array can be connected to the DataBand.

Then you can use fields from JSON in report objects.





# Parameters in connections

When using parameters in a connection and transmitting them using the connection wizard, the parameters have a string data type. Not all connections can correctly convert it to a type that the database can accept. Therefore, for some connections, functions have been added that convert string values to other data types supported by connection libraries. The following are the types the string is converted to based on the type of parameter. If it is necessary to assign a type to a parameter value that is not contained in the tables below, or is not implemented, then this can only be done programmatically.

## PostgresDataConnection

| NpgsqlDbType | Type | C# Type         | Example/Format                             |
|--------------|------|-----------------|--|
| Bigint       |      | long            | `123456`                                   |
| Money        |      | decimal         | `123456`                                   |
| Numeric      |      | decimal         | `123456`                                   |
| Integer      |      | int             | `123456`                                   |
| Oid          |      | uint            | `123456`                                   |
| Xid          |      | uint            | `123456`                                   |
| Cid          |      | uint            | `123456`                                   |
| Smallint     |      | short           | `123456`                                   |
| InternalChar |      | byte            | `123456`                                   |
| Real         |      | float           | `123456.12`                                |
| Double       |      | double          | `123456.12`                                |
| Boolean      |      | bool            | `True` or `1`                              |
| Bit          |      | string          | `1`  |
| Timestamp    |      | DateTime        | `12:15:12`                                 |
| TimestampTZ  |      | DateTime        | `12:15:12`                                 |
| Date         |      | DateTime        | `16/02/2008`                               |
| Time         |      | TimeSpan        | `6:12:14`                                  |
| Interval     |      | TimeSpan        | `6:12:14`                                  |
| TimeTZ       |      | DateTimeOffset  | `05/01/2008`                               |
| Uuid         |      | Guid            | `81a130d2-502f-4cf1-a376-63edeb000e9f`     |
| Box          |      | NpgsqlBox       | `((x1,y1),(x2,y2))`                        |
| Circle       |      | NpgsqlCircle    | <(x,y),r> (center point and radius)        |
| Line         |      | NpgsqlLine      | {A,B,C}                                    |
| Polygon      |      | NpgsqlPolygon   | ((x1,y1),...)                              |
| Path         |      | NpgsqlPath      | ((x1,y1),...)                              |
| LSeg         |      | NpgsqlLSeg      | ((x1,y1),(x2,y2))                          |
| Point        |      | NpgsqlPoint     | (x,y)                                      |
| Cidr         |      | NpgsqlCidr      | `192.168.100.128/25` IP address with mask  |
| Inet         |      | NpgsqlInet      | `192.168.100.128/25` IP address            |
| MacAddr      |      | PhysicalAddress | `08:00:2b:01:02:03`                        |
| TsQuery      |      | NpgsqlTsQuery   | `fat & rat`                                |
| TsVector     |      | NpgsqlTsVector  | `a fat cat sat on a mat and ate a fat rat` |
| Char         |      | string          | `a` single char                            |
| Text         |      | string          | `string`                                   |
| Varchar      |      | string          | `string`                                   |
| Name         |      | string          | `string`                                   |
| Citext       |      | string          | `string`                                   |
| Bytea        |      | string          | `12AA` hex string                          |
| Varbit       |      | string          | `01101` bit string                         |
| Tid          |      | NpgsqlTid       | `12345, 123` uint number and ushort number |
| Array        |      | ---             | Not implemented                            |
| Range        |      | ---             | Not implemented                            |
| Hstore       |      | ---             | Not implemented                            |
| Oidvector    |      | ---             | Not implemented                            |
| MacAddr8     |      | ---             | Not implemented                            |
| Int2Vector   |      | ---             | Not implemented                            |

## MySqlDataConnection

| MySqlDbType Type | C# Type  | Example                                |
|------------------|----------|--|
| Int64            | long     | `123456`                               |
| UInt64           | ulong    | `123456`                               |
| Int32            | int      | `123456`                               |
| Int24            | int      | `123456`                               |
| UInt24           | uint     | `123456`                               |
| UInt32           | uint     | `123456`                               |
| Int16            | short    | `123456`                               |
| UInt16           | ushort   | `123456`                               |
| Decimal          | decimal  | `123456`                               |
| NewDecimal       | decimal  | `123456`                               |
| Byte             | sbyte    | `12345`                                |
| UByte            | byte     | `12345`                                |
| Year             | byte     | `1901`                                 |
| Float            | float    | `123456.12`                            |
| Double           | double   | `123456.12`                            |
| Bit              | bool     | `True` or '1' bool                     |
| Bool             | bool     | `True` or '1' bool                     |
| DateTime         | DateTime | '16/02/2008 12:15:12'                  |
| Date             | DateTime | '16/02/2008'                           |
| Newdate          | DateTime | '16/02/2008'                           |
| Time             | DateTime | `6:12:14`                              |
| Timestamp        | DateTime | '16/02/2008 12:15:12'                  |
| Timestamp        | DateTime | '16/02/2008 12:15:12'                  |
| Guid             | Guid     | `81a130d2-502f-4cf1-a376-63edeb000e9f` |
| VarChar          | string   | 'string'                               |
| String           | string   | 'string'                               |
| TinyText         | string   | 'string'                               |
| MediumText       | string   | 'string'                               |
| LongText         | string   | 'string'                               |
| Text             | string   | 'string'                               |
| VarString        | string   | 'string'                               |
| JSON             | string   | 'string'                               |
| Enum             | string   | 'string'                               |
| Binary           | byte[]   | `12AA` hex string                      |
| VarBinary        | byte[]   | `12AA` hex string                      |
| Blob             | byte[]   | `12AA` hex string                      |
| TinyBlob         | byte[]   | `12AA` hex string                      |
| MediumBlob       | byte[]   | `12AA` hex string                      |
| LongBlob         | byte[]   | `12AA` hex string                      |
| Null             | DBNull   |  |
| Set              | ---      | Not implemented                        |
| Geometry         | ---      | Not implemented                        |

## ClickHouseDataConnection

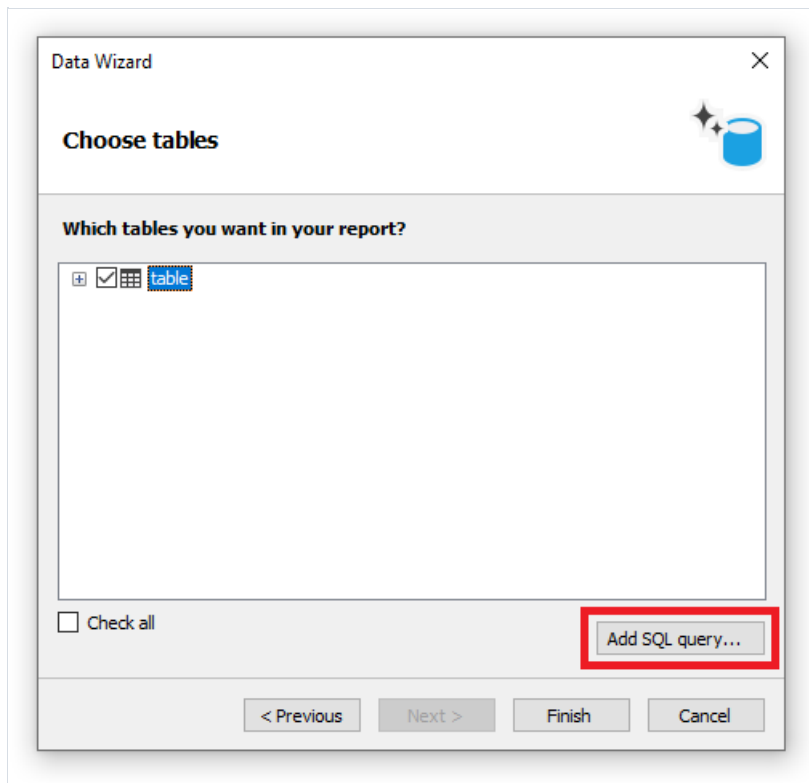
| ClickHouse Type | C# Type        | Example  |
|-----------------|----------------|--|
| Enum8           | sbyte          | `123456`   |
| Int8            | sbyte          | `123456`   |
| Enum16          | short          | `123456`   |
| Int16           | short          | `123456`   |
| Int32           | int            | `123456`   |
| Int64           | long           | `123456`   |
| Decimal         | decimal        | `123456`   |
| Float32         | float          | `123456.12`  |
| Float64         | double         | `123456.12`  |
| UInt8           | byte           | `123456`   |
| UInt16          | ushort         | `123456`   |
| UInt32          | uint           | `123456`   |
| UInt64          | ulong          | `123456`   |
| Date            | DateTime       | '16/02/2008 12:15:12'                                |
| DateTime        | DateTimeOffset | `05/01/2008`   |
| DateTime64      | DateTimeOffset | `05/01/2008`   |
| UUID            | Guid           | `81a130d2-502f-4cf1-a376-63edeb000e9f`               |
| IPv6            | IPAddress      | '2001:0db8:85a3:08d3:1319:8a2e:0370:7344' IP address |
| IPv4            | IPAddress      | '127.0.0.1' IP address                               |
| String          | string         | 'string'   |
| FixedString     | string         | 'string'   |
| Nothing         | DBNull         |  |
| Array           | ---            | Not implemented                                      |
| Nested          | ---            | Not implemented                                      |
| Tuple           | ---            | Not implemented                                      |
| Nullable        | ---            | Not implemented                                      |
| LowCardinality  | ---            | Not implemented                                      |

## MsSqlDataConnection

| SqlDbType Type   | C# Type        | Example                                |
|------------------|----------------|--|
| BigInt           | long           | `123456`                               |
| Bit              | bool           | `True` or `1`                          |
| DateTime         | DateTime       | `16/02/2008 12:15:12`                  |
| SmallDateTime    | DateTime       | `16/02/2008 12:15:12`                  |
| Date             | DateTime       | `16/02/2008 12:15:12`                  |
| Time             | DateTime       | `16/02/2008 12:15:12`                  |
| DateTime2        | DateTime       | `16/02/2008 12:15:12`                  |
| Char             | string         | `a` single char                        |
| NChar            | string         | `string`                               |
| NText            | string         | `string`                               |
| NVarChar         | string         | `string`                               |
| Text             | string         | `string`                               |
| VarChar          | string         | `string`                               |
| Xml              | string         | `<xml/>` xml string                    |
| Decimal          | decimal        | `string`                               |
| Money            | decimal        | `string`                               |
| SmallMoney       | decimal        | `string`                               |
| Real             | float          | `123456.12`                            |
| Float            | float          | `123456.12`                            |
| Int              | int            | `123456`                               |
| UniqueIdentifier | Guid           | `81a130d2-502f-4cf1-a376-63edeb000e9f` |
| SmallInt         | short          | `123456`                               |
| TinyInt          | byte           | `123456`                               |
| DateTimeOffset   | DateTimeOffset | `05/01/2008`                           |
| Variant          | object         |  |
| Udt              | object         |  |
| Binary           | byte[]         | `12AA` hex string                      |
| Image            | byte[]         | `12AA` hex string                      |
| Timestamp        | byte[]         | `12AA` hex string                      |
| VarBinary        | byte[]         | `12AA` hex string                      |
| Structured       | ---            | Not implemented                        |

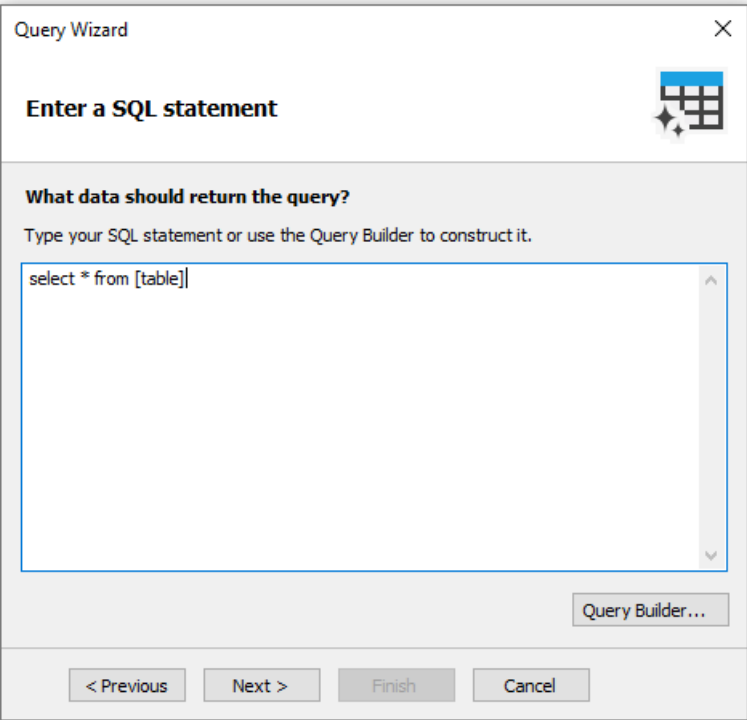
# Creating a SQL query

Data wizard allows quickly choosing tables, contained in a data base. Creation of SQL query requires additional work. For this, press the "Add SQL query..." button in the second step of the wizard.



You will see the query wizard window. Query wizard has four pages. Use the "Next" and "Back" buttons to switch between the pages.

In the first step, you need to set the name of a query. This name will appear in the "Data" window. Enter any unique name and press the "Next" button.



**Query Wizard**

**Enter a SQL statement**

What data should return the query?

Type your SQL statement or use the Query Builder to construct it.

`select * from [table]`

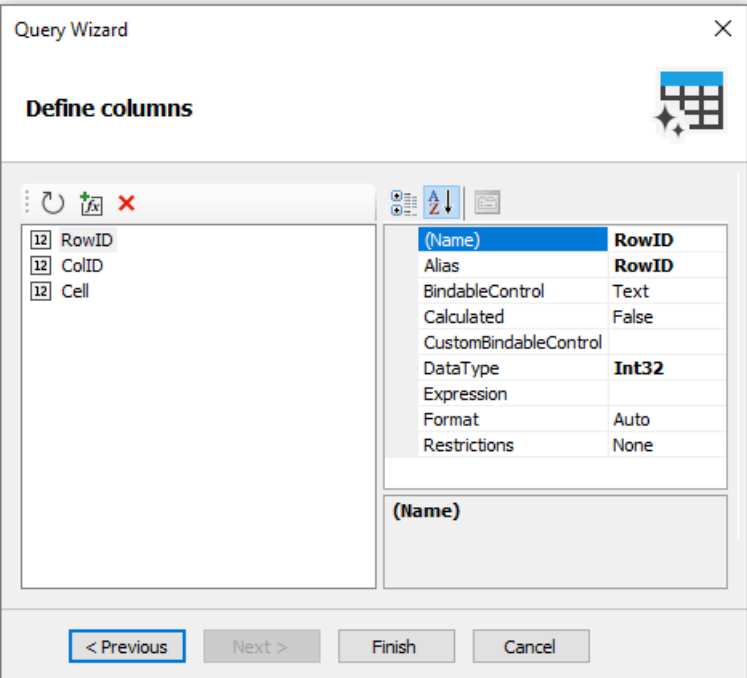
Query Builder...

< Previous   Next >   Finish   Cancel

In the second step, you have to enter a query in SQL language. Use the language dialect which is supported by your MSDB. You can use the query builder for visual query creation. To do this, press the "Query Builder" button. The query builder will be described in details later.

After you have entered the query text, press the "Next" button. In the third step, you can define the query parameters. It is required if your query has parameters. We will consider parameters later in this chapter.

On the last step of the wizard, you can set up the columns which were returned by the query:



**Query Wizard**

**Define columns**

RowID  
ColID  
Cell

|                       |       |
|-----------------------|-------|
| (Name)                | RowID |
| Alias                 | RowID |
| BindableControl       | Text  |
| Calculated            | False |
| CustomBindableControl |       |
| DataType              | Int32 |
| Expression            |       |
| Format                | Auto  |
| Restrictions          | None  |

(Name)

< Previous   Next >   Finish   Cancel

If you made a mistake in the query text or in the parameter definition, you will see error message when turning to the last page of the wizard.

As a rule, it is enough to be assured that the query has returned all the needed columns. On this step, you can do

the following:

- delete unnecessary columns using the "Delete" button;
- reset the columns by pressing the "Refresh" button;
- add a calculated column by pressing the "Add calculated column" button. For a new column, it is necessary to set the `Name`, `DataType` and `Expression` properties.

After closing the wizard by pressing the "Finish" button, you will return to the "Data Wizard" window.



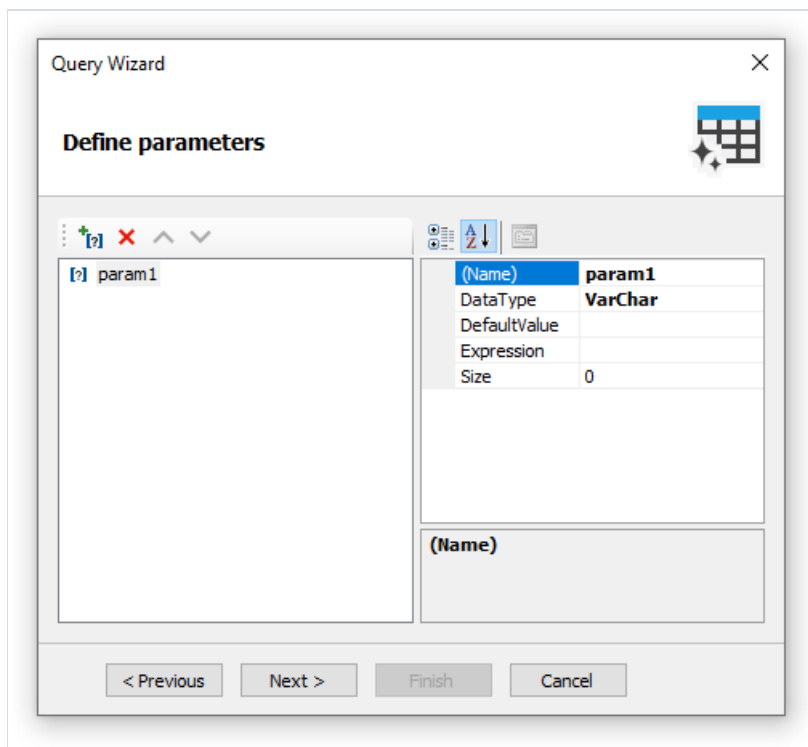
# Query parameters

There can be parameters in a query text. Let us see the following query:

```
select * from DVDs
where Title = @param1
```

This is the query to MS SQL demonstration database. The parameter with `param1` name is defined in a query. Here it should be noted: method of describing parameters in a query differs for different DBMS. For MS SQL a parameter is marked by a `@` symbol, MS Access parameters do not have names and are marked by the `?` symbol.

If your SQL query contains parameters, you have to declare them. It can be done in the third step of the "Query Wizard" which we have looked at above. To create a parameter, press the "Add parameter" button. A new parameter will be created:



The following parameter's properties should be set in the properties window:

| Property            | Description  |
|---------------------|--|
| <b>Name</b>         | Parameter name. Here you need to indicate the same name which you use in the query text. Some DBMS (for example, MS Access) do not support named parameters. In this case do not change this property. |
| <b>DataType</b>     | Parameter data type.   |
| <b>DefaultValue</b> | Value which will be used if the <code>Expression</code> property is not specified, or if it is impossible to calculate it (for example, when operating with the query in the report design mode).      |
| <b>Expression</b>   | Expression which returns parameter's value. This expression will be processed when you run the report. You can indicate any expression in this property (see details in the "Expressions" chapter).    |

**Property****Description****Size**

Parameter data size. This property must be specified if the parameter has a `string` specified in the `DataType` property.

If you set the parameter properties incorrectly, you will get an error when turning to the last page of the wizard.

# Passing a value to the parameter

Parameters are often used to ask a value from the user. Let us look at two ways to pass a value to the query parameter.

In the first way, you pass a value programmatically. Since there is no easy way to pass a value directly to the query parameter, you need to use the report parameter, which can be easily set via code. You should do the following:

- Create the report parameter (for more details about parameters, refer to the section ["Report parameters"](#)). Set the same `DataType` for the report parameter, as it is used in the query parameter.
- In the `Expression` property of the query parameter, refer to a report parameter, for example:

```
[MyReportParameter]
```

- Pass a value to the report parameter:

```
report1.SetParameterValue("MyReportParameter", 10);
```

In the second way, you use the dialogue forms to do this (for more details about dialogs, refer to the chapter ["Dialogue forms"](#)). For example, if you need to ask a string-type value, do the following:

- add a dialog into your report;
- put the "TextBoxControl" on it. This control will be used to enter the string value;
- set up the parameter as follows:

```
Name=param1  
DataType=VarChar  
DefaultValue= (empty string)  
Expression=TextBox1.Text  
Size=255
```

Where TextBox1 is a control which contains a value entered by the user.

## Editing a connection

Data connection which was created with the help of the "Data Wizard", can be edited. In order to do this, choose the data connection in the "Data" window and press the "Edit" button on the toolbar. You will see a data wizard window which we have looked at earlier. In this window, you can change the connection settings, by pressing the "Edit connection..." button. The type of connection cannot be changed.

On the second page of the wizard, you can select the tables which you want in your report. When you have done, press the "Finish" button.

## Editing a data source

Data source which was created with the help of the "Data Wizard", can be edited. In order to do this, choose the data source in the "Data" window and press the "Edit" button on the toolbar. You will see a "Query Wizard" window which we have looked at earlier. In this window, you can change the SQL query text, set up the query parameters and data columns.

In order to delete the data source, select it and press the "Delete" button on the toolbar. During this physical deletion of the source does not occur, it just changes to inaccessible. You can enter into the "Data|Choose Report Data..." menu and enable such a data source. However, this should never trouble you, because deleted data sources are never saved in the report file, and accordingly, do not get restored when the report is being read the next time.

# Adding a data source to an existing connection

There are two ways to add a data source (table or query) to an existing connection:

- open the connection editor as described in "[Edit Connection](#)". The connection wizard window will be shown, on the second page of which you can select or create a new data source;
- click the right mouse button on the "Connection" item and select the menu item "New data source...". As in the previous method, this will open the window of the Connection Wizard.

# Aliases

Every data element (data sources and columns) has got its own name. By default, this is the name defined in the database. In some cases, it can be difficult to understand what is hidden behind such name, for example, `ProdID`.

Data elements have got a second name - alias. By using an alias, you can rename an element. For example, if we have got a data source `CATEGORY_TABLE` with a column called `PROD_ID`, you can give the following alias:

```
CATEGORY_TABLE --> Categories  
PROD_ID --> Product ID
```

You can refer to such a data column in the following way:

```
[Categories.Product ID]
```

When referring to the data element, you must use the alias, if it has been defined. Never refer to an element by using its original name in this case.

In order to rename a data element in this case, choose it in the "Data" window, and press F2. Also, you can select a "Rename" item in the object's context menu. After this, enter the required name and press Enter.

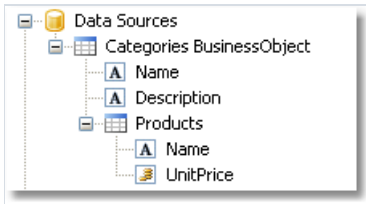
You can also rename an element by using the "Properties" window. Select an element in the "Data" window, switch to the "Properties" window and change the value of the `Alias` property.

In order to delete an alias (reset to the original name), select an element and choose the "Delete alias" item in its context menu.

# Hierarchical data sources

The data sources we have looked at, are relational, that is, they come from a relational DBMS (often called RDBMS). FastReport also supports other kinds of data - the hierarchical data sources. Such data sources come from so-called business objects, which are often used in applications to represent a relational data sources as a .NET classes.

The only way to add a hierarchical data source in your report is to register it programmatically. It will be discussed in the "Programmer's manual". Now we will look at some differences between ordinal and hierarchical data sources. In the figure below you can see two data sources, "Categories BusinessObject" and "Products". As you can see, the "Products" data source is contained within its parent, "Categories BusinessObject":



This means that, these two data sources are related to each other and can be used in the "master-detail" report type. You can also use each of these data sources separately in a "simple list" report type.



# Relations

Between two data sources, a relation can be set. The relation is used to define the "master-detail" relationship. For example, one record in the "Categories" table can have multiple entries in the "Products" table:

| Categories |              |
|------------|--------------|
| CategoryID | CategoryName |
| 1          | Beverages    |

| Products  |            |                  |
|-----------|------------|------------------|
| ProductID | CategoryID | Product name     |
| 1         | 1          | Chai             |
| 2         | 1          | Chang            |
| 39        | 1          | Chartreuse verte |
| 38        | 1          | Côte de Blaye    |
| 24        | 1          | Fantástica       |

In order to create a relation, you need to indicate the following:

1. parent table;
2. child table;
3. set of key columns in the parent table;
4. set of key columns in the child table.

As an example, we will look at "Categories" and "Products" tables from the demo database. They have the following structure:

| Categories             | Products             |
|------------------------|----------------------|
| CategoryID             | ProductID            |
| CategoryName           | ProductName          |
| Description            | SupplierID           |
| Picture                | CategoryID           |
| CategoriesTableAdapter | QuantityPerUnit      |
|                        | UnitPrice            |
|                        | UnitsInStock         |
|                        | UnitsOnOrder         |
|                        | ReorderLevel         |
|                        | Discontinued         |
|                        | EAN13                |
|                        | ProductsTableAdapter |

Both tables have got the `CategoryID` field, on which the relationship can be set. So, one category may contain several products.

How can related data sources be used in FastReport? There are two methods of doing this.

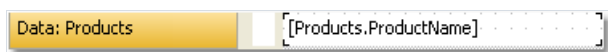
First method makes it possible to build reports of "master-detail" type. To do this, two "Data" bands are used. The master band is connected to the master data source, the detail band - to detail data source. Our example will be like this:

|                  |                           |
|------------------|---------------------------|
| Data: Categories | [Categories.CategoryName] |
| Data: Products   | [Products.ProductName]    |

Such a report, if you run it, will print a list of products in every category:

|                              |
|------------------------------|
| <b>Beverages</b>             |
| Chai                         |
| Chang                        |
| Chartreuse verte             |
| Côte de Blaye                |
| Guaraná Fantástica           |
| Ipoh Coffee                  |
| Lakkalikööri                 |
| Laughing Lumberjack Lager    |
| Outback Lager                |
| Rhönbräu Klosterbier         |
| Sasquatch Ale                |
| Steeleye Stout               |
| <b>Condiments</b>            |
| Aniseed Syrup                |
| Chef Anton's Cajun Seasoning |
| Chef Anton's Gumbo Mix       |
| Genen Shouyu                 |
| Grandma's Boysenberry Spread |
| Gula Malacca                 |

The second method allows referring to the master from the detail data source. We will show this by an example. Let us say, we want to print a list of all the products. For this, we need one "Data" band, which is connected to the "Products" table:



Such a report will print all the products from all the categories. Let us say, beside each product, we want to print a category name to which it belongs. Without using relation, this would have been harder. All we know about the product's category is its id (represented by `CategoryID` column in the "Products" table). Category name, which we would like to print, is stored in the `CategoryName` column of the "Categories" table. With the help of relation, we can refer to the name of a category in the following way:

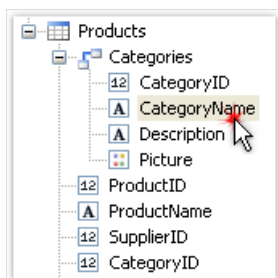
```
[Products.Categories.CategoryName]
```

For the current row of the "Products" table, FastReport will find the corresponding parent row in the "Categories" table, and return a value of the `CategoryName` column.

In a general case, way of referring to a parent table field allows an unlimited number of table ancestors:

```
[Child_table.Its_parent.Parent_of_a_parent.And _so_on.Column_name]
```

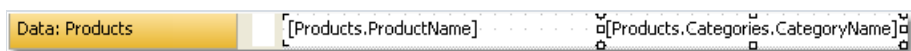
To add such a data column into a report, open the "Products" table in the "Data" window. You will see that among its columns, there is a link to the "Categories" table:



If we drag the column shown above into the report, then we will get a "Text" object with a text:

```
[Products.Categories.CategoryName]
```

Our report will be as follows:

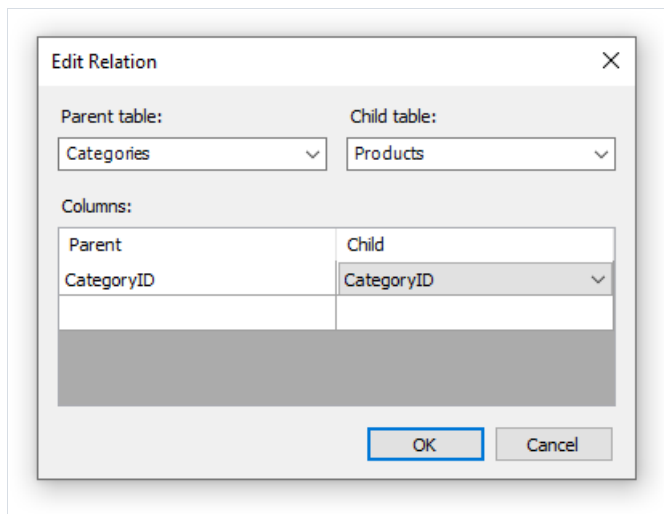


If we run it, we will see the following:

|                   |                |
|-------------------|----------------|
| Alice Mutton      | Meat/Poultry   |
| Aniseed Syrup     | Condiments     |
| Boston Crab Meat  | Seafood        |
| Camembert Pierrot | Dairy Products |
| Carnarvon Tigers  | Seafood        |
| Chai              | Beverages      |
| Chang             | Beverages      |

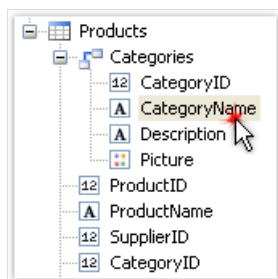
# Creating a relation

In order to create a relation, click the "Actions" button in the "Data" window, and select the "New relation..." item. You will see the relation editor:



In the first place, you need to choose the parent and the child tables. After this, in the lower part of the window, you need to choose related data columns. The tables can be related with the help of one or several data columns. After the columns have been set, close the relation editor by pressing the Ok button.

The relation that you've created can be seen in the "Data" window, if you choose the child data source and open a list of its columns. Among the columns, you will see the relationship with the parent source:



Parent source's data column can be inserted onto the report by using the drag&drop method. So, if we choose the columns shown in the figure, and drag it onto the report page, we will get a "Text" object with the following contents:

```
[Products.Categories.CategoryName]
```

## Editing a relation

In order to edit a relation, open the list of columns of the child data source, find the needed relation and click the "Edit..." button located on the toolbar. This will invoke the relation editor we have looked at earlier.

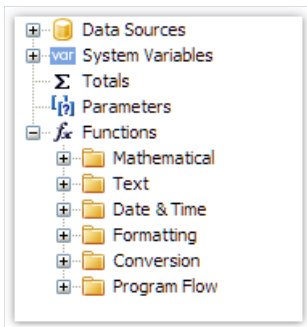
# System variables

There is a list of system variables that can be used in a report:

| Variable              | Description  |
|-----------------------|--|
| <b>Date</b>           | Date and time of the report's start.   |
| <b>Page</b>           | Current page number.   |
| <b>TotalPages</b>     | Total number of pages in the report. To use this variable, you need to enable the report's double pass. You can do this in "Report Properties..." menu.  |
| <b>PageN</b>          | Page number in the form: "Page N".   |
| <b>PageNofM</b>       | Page number in the form: "Page N of M".  |
| <b>Row#</b>           | Data row number inside the group. This value is reset at the start of a new group.   |
| <b>AbsRow#</b>        | Absolute number of data row. This value is never reset at the start of a new group.  |
| <b>Page#</b>          | Current page number. If you join several prepared reports into one package, this variable will return current page number in a package. This variable is actually a macro. Its value is substituted when the component is viewed in the preview window. That means you cannot use it in an expression.   |
| <b>TotalPages#</b>    | Total number of pages in the report. If you join several prepared reports into one package, this variable will return the number of pages in a package. You don't need to use double pass to get the correct value. This variable is actually a macro. Its value is substituted when the component is viewed in the preview window. That means you cannot use it in an expression. |
| <b>HierarchyLevel</b> | Current level of hierarchy in a hierarchical report (see <a href="#">"Printing hierarchy"</a> ). The top level is equal to 1.  |
| <b>HierarchyRow#</b>  | Full row number like "1.2.1" in a hierarchical report.   |

# Functions

FastReport.NET contains a lot of built-in functions (over 60). All functions are splitted to several categories and are accessible through the "Data" window:



You may use a function in any expression, in the script, or print its value in the "Text" object. For example, the following text in the "Text" object:

```
[Sqrt(4)]
```

Will be printed as "2" (square root of 4).

The following expression will return 4:

```
Sqrt(4) + 2
```

Let us look at the ways to insert a function in a report:

- you may drag&drop a function from the "Data" window to a report page. The "Text" object will be created, which contains a function call. You have to edit the text to add parameters to the function call;
- you can drag&drop a function to the script code;
- in the expression editor, you can see a copy of the "Data" window which acts the same way - you may drag items from it and drop them in the expression text.

Below we will describe each function in details.

# Mathematical

## Abs

| Function | Parameters    | Return value |
|----------|---------------|--------------|
| Abs      | sbyte value   | sbyte        |
| Abs      | short value   | short        |
| Abs      | int value     | int          |
| Abs      | long value    | long         |
| Abs      | float value   | float        |
| Abs      | double value  | double       |
| Abs      | decimal value | decimal      |

Returns the absolute value.

Example:

```
Abs(-2.2) = 2.2
```

## Acos

| Function | Parameters | Return value |
|----------|------------|--------------|
| Acos     | double d   | double       |

Returns the angle (in radians) whose cosine is `d`. `d` must be in range between -1 and 1.

Multiply the return value by `180 / Math.PI` to convert from radians to degrees.

Example:

```
Acos(0) * 180 / Math.PI = 90
```

## Asin



| Function          | Parameters            | Return value        |
|-------------------|-----------------------|---------------------|
| <code>Asin</code> | <code>double d</code> | <code>double</code> |

Returns the angle (in radians) whose sine is `d`. `d` must be in range between -1 and 1.

Multiply the return value by `180 / Math.PI` to convert from radians to degrees.

Example:

```
Asin(0) = 0
```

## Atan

| Function          | Parameters            | Return value        |
|-------------------|-----------------------|---------------------|
| <code>Atan</code> | <code>double d</code> | <code>double</code> |

Returns the angle (in radians) whose tangent is `d`.

Multiply the return value by `180 / Math.PI` to convert from radians to degrees.

Example:

```
Atan(1) * 180 / Math.PI = 45
```

## Ceiling

| Function             | Parameters             | Return value         |
|----------------------|------------------------|----------------------|
| <code>Ceiling</code> | <code>double d</code>  | <code>double</code>  |
| <code>Ceiling</code> | <code>decimal d</code> | <code>decimal</code> |

Returns the smallest integer greater than or equal to `d`.

Example:

```
Ceiling(1.7) = 2
```

## Cos

| Function         | Parameters            | Return value        |
|------------------|-----------------------|---------------------|
| <code>Cos</code> | <code>double d</code> | <code>double</code> |

Returns the cosine of the specified angle `d`. The angle must be in radians.

Multiply by `Math.PI / 180` to convert degrees to radians.

Example:

```
Cos(90 * Math.PI / 180) = 0
```

## Exp

| Function         | Parameters            | Return value        |
|------------------|-----------------------|---------------------|
| <code>Exp</code> | <code>double d</code> | <code>double</code> |

Returns e (2.71828), raised to the specified power `d`.

Example:

```
Exp(1) = 2.71828
```

## Floor

| Function           | Parameters             | Return value         |
|--------------------|------------------------|----------------------|
| <code>Floor</code> | <code>double d</code>  | <code>double</code>  |
| <code>Floor</code> | <code>decimal d</code> | <code>decimal</code> |

Returns the largest integer less than or equal to `d`.

Example:

```
Floor(1.7) = 1
```

## Log

| Function         | Parameters            | Return value        |
|------------------|-----------------------|---------------------|
| <code>Log</code> | <code>double d</code> | <code>double</code> |

Returns the logarithm of a specified number `d`.

Example:

```
Log(2.71828) = 1
```

## Maximum

| Function | Parameters                 | Return value |
|----------|----------------------------|--------------|
| Maximum  | int val1, int val2         | int          |
| Maximum  | long val1, long val2       | long         |
| Maximum  | float val1, float val2     | float        |
| Maximum  | double val1, double val2   | double       |
| Maximum  | decimal val1, decimal val2 | decimal      |

Returns the larger of `val1` , `val2` .

Example:

```
Maximum(1,2) = 2
```

## Minimum

| Function | Parameters                 | Return value |
|----------|----------------------------|--------------|
| Minimum  | int val1, int val2         | int          |
| Minimum  | long val1, long val2       | long         |
| Minimum  | float val1, float val2     | float        |
| Minimum  | double val1, double val2   | double       |
| Minimum  | decimal val1, decimal val2 | decimal      |

Returns the smaller of `val1` , `val2` .

Example:

```
Minimum(1,2) = 1
```

## Round

| Function | Parameters | Return value |
|----------|------------|--------------|
| Round    | double d   | double       |

| Function | Parameters | Return value |
|----------|------------|--------------|
| Round    | decimal d  | decimal      |

Rounds `d` to the nearest integer.

Example:

```
Round(1.47) = 1
```

| Function | Parameters            | Return value |
|----------|-----------------------|--------------|
| Round    | double d, int digits  | double       |
| Round    | decimal d, int digits | decimal      |

Rounds `d` to a precision specified in the `digits` parameter.

Example:

```
Round(1.478, 2) = 1.48
```

## Sin

| Function | Parameters | Return value |
|----------|------------|--------------|
| Sin      | double d   | double       |

Returns the sine of the specified angle `d`. The angle must be in radians.

Multiply by `Math.PI / 180` to convert degrees to radians.

Example:

```
Sin(90 * Math.PI / 180) = 1
```

## Sqrt

| Function | Parameters | Return value |
|----------|------------|--------------|
| Sqrt     | double d   | double       |

Returns the square root of `d`.

Example:

```
Sqrt(4) = 2
```

## Tan

| Function | Parameters | Return value |
|----------|------------|--------------|
| Tan      | double d   | double       |

Returns the tangent of the specified angle `d`. The angle must be in radians.

Multiply by `Math.PI / 180` to convert degrees to radians.

Example:

```
Tan(45 * Math.PI / 180) = 1
```

## Truncate

| Function | Parameters | Return value |
|----------|------------|--------------|
| Truncate | double d   | double       |
| Truncate | decimal d  | decimal      |

Calculates the integral part of `d`.

Example:

```
Truncate(1.7) = 1
```

# Text

Note:

- these functions do not modify the string value passed in. Instead, they return a new modified string;
- the first character in a string has 0 index. Keep it in mind when working with functions that take the character index, such as `Insert`

## Asc

| Function         | Parameters          | Return value     |
|------------------|---------------------|------------------|
| <code>Asc</code> | <code>char c</code> | <code>int</code> |

Returns an `Integer` value representing the character code corresponding to a character.

Example:

```
Asc('A') = 65
```

## Chr

| Function         | Parameters         | Return value      |
|------------------|--------------------|-------------------|
| <code>Chr</code> | <code>int i</code> | <code>char</code> |

Returns the character associated with the specified character code.

Example:

```
Chr(65) = 'A'
```

## Insert

| Function            | Parameters  | Return value        |
|---------------------|---|---------------------|
| <code>Insert</code> | <code>string s, int startIndex, string value</code> | <code>string</code> |

Inserts a `value` substring into the `s` string at a specified index position `startIndex` and returns a new string.

Example:

```
Insert("ABC", 1, "12") = "A12BC"
```

# Length

| Function            | Parameters            | Return value     |
|---------------------|-----------------------|------------------|
| <code>Length</code> | <code>string s</code> | <code>int</code> |

Returns the length of `s`.

Example:

```
Length("ABC") = 3
```

# LowerCase

| Function               | Parameters            | Return value        |
|------------------------|-----------------------|---------------------|
| <code>LowerCase</code> | <code>string s</code> | <code>string</code> |

Converts all characters of `s` to lower case and returns a result.

Example:

```
LowerCase("ABC") = "abc"
```

# PadLeft

| Function             | Parameters                            | Return value        |
|----------------------|---------------------------------------|---------------------|
| <code>PadLeft</code> | <code>string s, int totalWidth</code> | <code>string</code> |

Right-aligns the characters in the `s` string, padding with spaces on the left for a total width specified in the `totalWidth` parameter.

Example:

```
PadLeft("ABC", 5) = "  ABC"
```

| Function             | Parameters  | Return value        |
|----------------------|---|---------------------|
| <code>PadLeft</code> | <code>string s, int totalWidth, char paddingChar</code> | <code>string</code> |

Right-aligns the characters in the `s` string, padding with `paddingChar` characters on the left for a total width specified in the `totalWidth` parameter.

Example:

```
PadLeft("ABC", 5, '0') = "00ABC"
```

## PadRight

| Function              | Parameters                            | Return value        |
|-----------------------|---------------------------------------|---------------------|
| <code>PadRight</code> | <code>string s, int totalWidth</code> | <code>string</code> |

Left-aligns the characters in the `s` string, padding with spaces on the right for a total width specified in the `totalWidth` parameter.

Example:

```
PadRight("ABC", 5) = "ABC  "
```

| Function              | Parameters  | Return value        |
|-----------------------|---|---------------------|
| <code>PadRight</code> | <code>string s, int totalWidth, char paddingChar</code> | <code>string</code> |

Left-aligns the characters in the `s` string, padding with `paddingChar` characters on the right for a total width specified in the `totalWidth` parameter.

Example:

```
PadRight("ABC", 5, '0') = "ABC00"
```

## Remove

| Function            | Parameters                            | Return value        |
|---------------------|---------------------------------------|---------------------|
| <code>Remove</code> | <code>string s, int startIndex</code> | <code>string</code> |

Deletes all the characters from the `s` string beginning at `startIndex` position and continuing through the last position.

Example:

```
Remove("ABCD", 3) = "ABC"
```

| Function            | Parameters                                       | Return value        |
|---------------------|--|---------------------|
| <code>Remove</code> | <code>string s, int startIndex, int count</code> | <code>string</code> |

Deletes a number of characters specified in the `count` parameter from the `s` string, beginning at a `startIndex`



position.

Example:

```
Remove("A00BC", 1, 2) = "ABC"
```

## Replace

| Function | Parameters                                 | Return value |
|----------|--|--------------|
| Replace  | string s, string oldValue, string newValue | string       |

Returns a string `s` in which a specified substring `oldValue` has been replaced with another substring `newValue`.

Example:

```
Replace("A00", "00", "BC") = "ABC"
```

## Substring

| Function  | Parameters               | Return value |
|-----------|--------------------------|--------------|
| Substring | string s, int startIndex | string       |

Retrieves a substring from the `s` string. The substring starts at a character position specified in the `startIndex` parameter.

Example:

```
Substring("ABCDEF", 4) = "EF"
```

| Function  | Parameters                           | Return value |
|-----------|--------------------------------------|--------------|
| Substring | string s, int startIndex, int length | string       |

Retrieves a substring from the `s` string. The substring starts at a character position specified in the `startIndex` parameter and has a length specified in the `length` parameter.

Example:

```
Substring("ABCDEF", 1, 3) = "BCD"
```

## TitleCase

| Function  | Parameters | Return value |
|-----------|------------|--------------|
| TitleCase | string s   | string       |

Converts the specified string to titlecase.

Example:

```
TitleCase("john smith") = "John Smith"
```

## Trim

| Function | Parameters | Return value |
|----------|------------|--------------|
| Trim     | string s   | string       |

Removes all occurrences of white space characters from the beginning and end of the `s` string.

Example:

```
Trim(" ABC ") = "ABC"
```

## UpperCase

| Function  | Parameters | Return value |
|-----------|------------|--------------|
| UpperCase | string s   | string       |

Converts all characters of `s` to upper case and returns a result.

Example:

```
UpperCase("abc") = "ABC"
```

# Date & Time

To set a specific date, you can use the constructors of the `DateTime` structure or the `ToDateTime` functions (see more details in the "Conversion") or the `DateSerial` function (discussed below).

For example, to create a date and time using constructors, you can use the following code:

```
new DateTime(year, month, day, hours, minutes, seconds);
```

When using the date conversion functions from this section, the time information is by default preserved. To remove the time from the date, you can use the `Format` or `FormatDateTime` functions. More detailed information about these functions is presented in the "Formatting" section.

As an example, the `[Employees.BirthDate]` field from the demonstration database will be used in the functions.

```
[Employees.BirthDate] = 27.01.1986
```

## AddDays

| Function             | Parameters                               | Return value          |
|----------------------|--|-----------------------|
| <code>AddDays</code> | <code>DateTime date, double value</code> | <code>DateTime</code> |

Adds the specified number of days ( `value` ) to the `date` date and returns a new date.

Example:

```
AddDays(new DateTime(2024,4,2), 2) = 4/4/2024 12:00:00 AM
AddDays.ToDateTime("4/2/2024"), 2) = 4/4/2024 12:00:00 AM
AddDays([Employees.BirthDate], 2) = 1/29/1986 12:00:00 AM
```

## AddHours

| Function              | Parameters                               | Return value          |
|-----------------------|--|-----------------------|
| <code>AddHours</code> | <code>DateTime date, double value</code> | <code>DateTime</code> |

Adds the specified number of hours ( `value` ) to the `date` date and returns a new date.

Example:

```
AddHours(new DateTime(2024,4,2,5,30,5), 1) = 4/2/2024 6:30:05 AM
AddHours.ToDateTime("4/2/2024 5:30:05"), 1) = 4/2/2024 6:30:05 AM
AddHours([Employees.BirthDate], 1) = 1/27/1986 1:00:00 AM
```

## AddMinutes

| Function   | Parameters                  | Return value |
|------------|-----------------------------|--------------|
| AddMinutes | DateTime date, double value | DateTime     |

Adds the specified number of minutes ( `value` ) to the `date` date and returns a new date.

Example:

```
AddMinutes(new DateTime(2024,4,2,5,30,5), 10) = 4/2/2024 5:40:05 AM
AddMinutes(ToDateTime("4/2/2024 5:30:05"), 10) = 4/2/2024 5:40:05 AM
AddMinutes([Employees.BirthDate], 10) = 1/27/1986 12:10:00 AM
```

## AddMonths

| Function  | Parameters               | Return value |
|-----------|--------------------------|--------------|
| AddMonths | DateTime date, int value | DateTime     |

Adds the specified number of months ( `value` ) to the `date` date and returns a new date.

Example:

```
AddMonths(new DateTime(2024,4,2,5,30,5), 2) = 6/2/2024 5:30:05 AM
AddMonths(ToDateTime("4/2/2024 5:30:05"), 2) = 6/2/2024 5:30:05 AM
AddMonths([Employees.BirthDate], 2) = 3/27/1986 12:00:00 AM
```

## AddSeconds

| Function   | Parameters                  | Return value |
|------------|-----------------------------|--------------|
| AddSeconds | DateTime date, double value | DateTime     |

Adds the specified number of seconds ( `value` ) to the `date` date and returns a new date.

Example:

```
AddSeconds(new DateTime(2024,4,2,5,30,5), 10) = 4/2/2024 5:30:15 AM
AddSeconds(ToDateTime("4/2/2024 5:30:05"), 10) = 4/2/2024 5:30:15 AM
AddSeconds([Employees.BirthDate], 10) = 1/27/1986 12:00:10 AM
```

## AddYears

| Function | Parameters               | Return value |
|----------|--------------------------|--------------|
| AddYears | DateTime date, int value | DateTime     |

Adds the specified number of years ( `value` ) to the `date` date and returns a new date.

Example:

```
AddYears(new DateTime(2024,4,2,5,30,5), 3) = 4/2/2027 5:30:05 AM
AddYears(ToDateTime("4/2/2024 5:30:05"), 3) = 4/2/2027 5:30:05 AM
AddYears([Employees.BirthDate], 3) = 1/27/1989 12:00:00 AM
```

## DateDiff

| Function              | Parameters                                  | Return value          |
|-----------------------|---|-----------------------|
| <code>DateDiff</code> | <code>DateTime date1, DateTime date2</code> | <code>TimeSpan</code> |

Returns the interval (number of days, hours, minutes, seconds) between two dates.

Example:

```
DateDiff(new DateTime(2024,4,2,5,0,0), new DateTime(2025,1,2,5,30,5)) = -275.00:30:05
DateDiff(ToDateTime("1/2/2025 5:30:05"), ToDateTime("4/2/2024 5:00:00")) = 275.00:30:05
DateDiff(ToDateTime("4/2/2024 5:00:00"), [Employees.BirthDate]) = 13945.05:00:00
```

## DateSerial

| Function                | Parameters                                | Return value          |
|-------------------------|---|-----------------------|
| <code>DateSerial</code> | <code>int year, int month, int day</code> | <code>DateTime</code> |

Creates a new `DateTime` value from the specified `year` , `month` and `day` .

Another available method to set a specific date.

Example:

```
DateSerial(2024,4,2) = 4/2/2024 12:00:00 AM
```

## Day

| Function         | Parameters                 | Return value     |
|------------------|----------------------------|------------------|
| <code>Day</code> | <code>DateTime date</code> | <code>int</code> |

Gets the day of the month (1-31) represented by the specified `date` .

Example:

```
Day(new DateTime(2024,4,2)) = 2
Day(DateTime("4/2/2024")) = 2
Day([Employees.BirthDate]) = 27
```

## DayOfWeek

| Function  | Parameters    | Return value |
|-----------|---------------|--------------|
| DayOfWeek | DateTime date | string       |

Gets the localized name of the day of the week represented by the specified `date` .

Example:

```
DayOfWeek(new DateTime(2024,4,2)) = "Tuesday"
DayOfWeek(DateTime("4/2/2024")) = "Tuesday"
DayOfWeek([Employees.BirthDate]) = "Monday"
```

## DayOfYear

| Function  | Parameters    | Return value |
|-----------|---------------|--------------|
| DayOfYear | DateTime date | int          |

Gets the day of the year (1-365) represented by the specified `date` .

Example:

```
DayOfWeek(new DateTime(2024,4,2)) = 93
DayOfWeek(DateTime("4/2/2024")) = 93
DayOfWeek([Employees.BirthDate]) = 27
```

## DaysInMonth

| Function    | Parameters          | Return value |
|-------------|---------------------|--------------|
| DaysInMonth | int year, int month | int          |

Returns the number of days in the specified `month` and `year` .

Example:

```
DaysInMonth(2024, 4) = 30
```

## Hour

| Function | Parameters    | Return value |
|----------|---------------|--------------|
| Hour     | DateTime date | int          |

Gets the hour component (0-23) represented by the specified `date`.

Example:

```
Hour(new DateTime(2024,4,2,5,30,5)) = 5
Hour(ToDateTime("4/2/2024 5:30:05")) = 5
Hour([Employees.BirthDate]) = 0
```

## Minute

| Function | Parameters    | Return value |
|----------|---------------|--------------|
| Minute   | DateTime date | int          |

Gets the minute component (0-59) represented by the specified `date`.

Example:

```
Minute(new DateTime(2024,4,2,5,30,5)) = 30
Minute(ToDateTime("4/2/2024 5:30:05")) = 30
Minute([Employees.BirthDate]) = 0
```

## Month

| Function | Parameters    | Return value |
|----------|---------------|--------------|
| Month    | DateTime date | int          |

Gets the month component (1-12) represented by the specified `date`.

Example:

```
Month(new DateTime(2024,4,2)) = 4
Month(ToDateTime("4/2/2024")) = 4
Month([Employees.BirthDate]) = 1
```

## MonthName

| Function  | Parameters | Return value |
|-----------|------------|--------------|
| MonthName | int month  | string       |

Gets the localized name of the specified `month` (1-12).

Example:

```
MonthName(1) = "January"
```

## Second

| Function            | Parameters                 | Return value     |
|---------------------|----------------------------|------------------|
| <code>Second</code> | <code>DateTime date</code> | <code>int</code> |

Gets the seconds component (0-59) represented by the specified `date`.

Example:

```
Second(new DateTime(2024,4,2,5,30,5)) = 5
Second(ToDateTime("4/2/2024 5:30:05")) = 5
Second([Employees.BirthDate]) = 0
```

## Year

| Function          | Parameters                 | Return value     |
|-------------------|----------------------------|------------------|
| <code>Year</code> | <code>DateTime date</code> | <code>int</code> |

Gets the year component represented by the specified `date`.

Example:

```
Year(new DateTime(2024,4,2)) = 2024
Year(ToDateTime("4/2/2024")) = 2024
Year([Employees.BirthDate]) = 1986
```



# Formatting

## Format

| Function            | Parameters                                       | Return value        |
|---------------------|--|---------------------|
| <code>Format</code> | <code>string format, params object[] args</code> | <code>string</code> |

Replaces the format item in a specified `format` string with the value of a corresponding Object instance in a specified `args` array.

For example, the following function call:

```
Format("Name = {0}, hours = {1:hh}", myName, DateTime.Now)
```

contains the following format items: `{0}` and `{1:hh}`. They will be replaced with values of `myName` and `DateTime.Now`. The result may look as follows:

```
Name = Alex, hours = 12
```

Each format item takes the following form:

```
{index[,alignment][:formatString]}
```

- `index` - a zero-based integer that indicates which element in a list of objects to format;
- `alignment` - an optional integer indicating the minimum width of the region to contain the formatted value. If the length of the formatted value is less than alignment, then the region is padded with spaces. If alignment is negative, the formatted value is left justified in the region; if alignment is positive, the formatted value is right justified;
- `formatString` - an optional string of format specifiers.

The following table describes the standard numeric format strings.

| Format Specifier | Name     | Description   |
|------------------|----------|---|
| <b>C or c</b>    | Currency | The number is converted to a string that represents a currency amount.<br><code>Format("{0:C}", 10) = "\$10.00"</code>                                    |
| <b>D or d</b>    | Decimal  | This format is supported for integral types only. The number is converted to a string of decimal digits (0-9).<br><code>Format("{0:D}", 10) = "10"</code> |

| Format Specifier | Name        | Description   |
|------------------|-------------|---|
| <b>E or e</b>    | Scientific  | The number is converted to a string of the form <code>-d.ddd...E+ddd</code> or <code>-d.ddd...e+ddd</code> , where each <code>d</code> indicates a digit (0-9).<br><code>Format("{0:E}", 10) = "1.000000E+001"</code>   |
| <b>F or f</b>    | Fixed-point | The number is converted to a string of the form <code>-ddd.ddd...</code> where each <code>d</code> indicates a digit (0-9).<br><code>Format("{0:F}", 10) = "10.00"</code>   |
| <b>G or g</b>    | General     | The number is converted to the most compact notation.<br><code>Format("{0:G}", 10) = "10"</code>  |
| <b>N or n</b>    | Number      | The number is converted to a string of the form <code>-d,ddd,ddd.ddd...</code> , where each <code>d</code> indicates a digit (0-9).<br><code>Format("{0:N}", 1234.56) = "1,234.56"</code>   |
| <b>P or p</b>    | Percent     | The number is converted to a string that represents a percent. The converted number is multiplied by 100 in order to be presented as a percentage.<br><code>Format("{0:P}", 0.15) = "15.00%"</code>   |
| <b>X or x</b>    | Hexadecimal | The number is converted to a string of hexadecimal digits. The case of the format specifier indicates whether to use uppercase or lowercase characters for the hexadecimal digits greater than 9. For example, use <code>x</code> to produce <code>ABCDEF</code> , and <code>x</code> to produce <code>abcdef</code> .<br><code>Format("{0:X}", 26) = "1A"</code> |

If you format the floating-point values, you may indicate a number of decimal places after the format string:

```
Format("{0:C1}", 12.23) = "$12.2"
```

If the standard numeric format specifiers do not provide the type of formatting you require, you can use custom format strings:

| Format character | Description  |
|------------------|--|
| <b>0</b>         | Zero placeholder. If the value being formatted has a digit in the position where the <code>0</code> appears in the format string, then that digit is copied to the result string. The position of the leftmost <code>0</code> before the decimal point and the rightmost <code>0</code> after the decimal point determines the range of digits that are always present in the result string. |
| <b>#</b>         | Digit placeholder. If the value being formatted has a digit in the position where the <code>#</code> appears in the format string, then that digit is copied to the result string. Otherwise, nothing is stored in that position in the result string.   |
| <b>.</b>         | Decimal point. The first <code>.</code> character in the format string determines the location of the decimal separator in the formatted value.  |
| <b>,</b>         | Thousand separator. If the format string contains a <code>,</code> character, then the output will have thousand separators inserted between each group of three digits to the left of the decimal separator.  |

| Format character | Description |
|------------------|-------------|
|------------------|-------------|

|   |  |
|---|--|
| % | Percentage placeholder. The presence of a % character in a format string causes a number to be multiplied by 100 before it is formatted. |
| ; | Section separator. The ; character is used to separate sections for positive, negative, and zero numbers in the format string.           |

Examples of use:

```
Format("{0:$#,##0.00}", 1024.25) = "$1,024.25"
Format("{0:00%}", 0.25) = "25%"
Format("{0:$#,##0.00;($#,##0.00);Zero}", 1024.25) = "$1,024.25"
Format("{0:$#,##0.00;($#,##0.00);Zero}", -1024.25) = "($1,024.25)"
Format("{0:$#,##0.00;($#,##0.00);Zero}", 0) = "Zero"
```

The following table describes the standard format specifiers for formatting the DateTime values:

| Format Specifier | Name                                   | Example                             |
|------------------|--|-------------------------------------|
| <b>d</b>         | Short date pattern                     | "4/2/2024"                          |
| <b>D</b>         | Long date pattern                      | "Tuesday, April 2, 2024"            |
| <b>f</b>         | Full date/time pattern (short time)    | "Tuesday, April 2, 2024 8:41 PM"    |
| <b>F</b>         | Full date/time pattern (long time)     | "Tuesday, April 2, 2024 8:41:01 PM" |
| <b>g</b>         | General date/time pattern (short time) | "4/2/2024 8:41 PM"                  |
| <b>G</b>         | General date/time pattern (long time)  | "4/2/2024 8:41:01 PM"               |
| <b>t</b>         | Short time pattern                     | "8:41 PM"                           |
| <b>T</b>         | Long time pattern                      | "8:41:01 PM"                        |

The following table describes the custom date/time format specifiers and the results they produce.

| Format Specifier | Description |
|------------------|-------------|
|------------------|-------------|

|             |   |
|-------------|---|
| <b>d</b>    | Displays the current day of the month, measured as a number between 1 and 31, inclusive. If the day is a single digit only (1-9), then it is displayed as a single digit.     |
| <b>dd</b>   | Displays the current day of the month, measured as a number between 1 and 31, inclusive. If the day is a single digit only (1-9), it is formatted with a preceding 0 (01-09). |
| <b>ddd</b>  | Displays the abbreviated name of the day.   |
| <b>dddd</b> | Displays the full name of the day.  |

| Format Specifier | Description |
|------------------|-------------|
|------------------|-------------|

|               |   |
|---------------|---|
| <b>f or F</b> | Displays the most significant digit of the seconds fraction.  |
| <b>h</b>      | Displays the hour in the range 1-12. If the hour is a single digit (1-9), it is displayed as a single digit.  |
| <b>hh</b>     | Displays the hour in the range 1-12. If the hour is a single digit (1-9), it is formatted with a preceding 0 (01-09).   |
| <b>H</b>      | Displays the hour in the range 0-23. If the hour is a single digit (1-9), it is displayed as a single digit.  |
| <b>HH</b>     | Displays the hour in the range 0-23. If the hour is a single digit (1-9), it is formatted with a preceding 0 (01-09).   |
| <b>m</b>      | Displays the minute in the range 0-59. If the minute is a single digit (0-9), it is displayed as a single digit.  |
| <b>mm</b>     | Displays the minute in the range 0-59. If the minute is a single digit (0-9), it is formatted with a preceding 0 (01-09).   |
| <b>M</b>      | Displays the month, measured as a number between 1 and 12, inclusive. If the month is a single digit (1-9), it is displayed as a single digit.  |
| <b>MM</b>     | Displays the month, measured as a number between 1 and 12, inclusive. If the month is a single digit (1-9), it is formatted with a preceding 0 (01-09).   |
| <b>MMM</b>    | Displays the abbreviated name of the month.   |
| <b>MMMM</b>   | Displays the full name of the month.  |
| <b>s</b>      | Displays the seconds in the range 0-59. If the second is a single digit (0-9), it is displayed as a single digit only.  |
| <b>ss</b>     | Displays the seconds in the range 0-59. If the second is a single digit (0-9), it is formatted with a preceding 0 (01-09).  |
| <b>t</b>      | Displays the first character of the A.M./P.M. designator.   |
| <b>tt</b>     | Displays the A.M./P.M. designator.  |
| <b>y</b>      | Displays the year as a maximum two-digit number. The first two digits of the year are omitted. If the year is a single digit (1-9), it is displayed as a single digit.  |
| <b>yy</b>     | Displays the year as a maximum two-digit number. The first two digits of the year are omitted. If the year is a single digit (1-9), it is formatted with a preceding 0 (01-09).   |
| <b>yyyy</b>   | Displays the year, including the century. If the year is less than four digits in length, then preceding zeros are appended as necessary to make the displayed year four digits long.   |
| <b>z</b>      | Displays the time zone offset for the system's current time zone in whole hours only. The offset is always displayed with a leading sign (zero is displayed as "+0"), indicating hours ahead of Greenwich mean time (+) or hours behind Greenwich mean time (-). The range of values is -12 to +13. If the offset is a single digit (0-9), it is displayed as a single digit with the appropriate leading sign. |

| Format Specifier | Description |
|------------------|-------------|
|------------------|-------------|

|            |   |
|------------|---|
| <b>zz</b>  | Displays the time zone offset for the system's current time zone in whole hours only. The offset is always displayed with a leading or trailing sign (zero is displayed as "+00"), indicating hours ahead of Greenwich mean time (+) or hours behind Greenwich mean time (-). The range of values is -12 to +13. If the offset is a single digit (0-9), it is formatted with a preceding 0 (01-09) with the appropriate leading sign.           |
| <b>zzz</b> | Displays the time zone offset for the system's current time zone in hours and minutes. The offset is always displayed with a leading or trailing sign (zero is displayed as "+00:00"), indicating hours ahead of Greenwich mean time (+) or hours behind Greenwich mean time (-). The range of values is -12:00 to +13:00. If the offset is a single digit (0-9), it is formatted with a preceding 0 (01-09) with the appropriate leading sign. |
| :          | Time separator.   |
| /          | Date separator.   |

Examples of use:

```
Format("{0:d MMM yyyy}", DateTime.Now) = "2 Apr 2024"
Format("{0:MM/dd/yyyy}", DateTime.Now) = "04/02/2024"
Format("{0:MMMM, d}", DateTime.Now) = "April, 2"
Format("{0:HH:mm}", DateTime.Now) = "20:41"
Format("{0:MM/dd/yyyy hh:mm tt}", DateTime.Now) = "04/02/2024 08:41 PM"
```

## FormatCurrency

| Function                    | Parameters                | Return value        |
|-----------------------------|---------------------------|---------------------|
| <code>FormatCurrency</code> | <code>object value</code> | <code>string</code> |

Formats the specified `value` as a currency, using the Windows regional settings.

Example:

```
FormatCurrency(1.25) = "$1.25"
```

| Function                    | Parameters                                   | Return value        |
|-----------------------------|--|---------------------|
| <code>FormatCurrency</code> | <code>object value, int decimalDigits</code> | <code>string</code> |

Formats the specified `value` as a currency. The `decimalDigits` parameter indicates how many places are displayed to the right of the decimal.

Example:

```
FormatCurrency(1.25, 1) = "$1.3"
```

## FormatDateTime

| Function       | Parameters     | Return value |
|----------------|----------------|--------------|
| FormatDateTime | DateTime value | string       |

Formats the specified `value` as a date/time, using the Windows regional settings. This function does not include neutral values in the resulting string.

Example:

```
FormatDateTime(new DateTime(2024,4,2)) = "4/2/2024"  
FormatDateTime(new DateTime(2024,4,2,1,30,0)) = "4/2/2024 1:30:00 AM"  
FormatDateTime(new DateTime(1,1,1,1,30,1)) = "1:30:01 AM"
```

| Function       | Parameters                    | Return value |
|----------------|-------------------------------|--------------|
| FormatDateTime | DateTime value, string format | string       |

Formats the specified `value` as a date/time, using the named format specified in the `format` parameter. The valid values for this parameter are:

- Long Date;
- Short Date;
- Long Time;
- Short Time.

Example:

```
FormatDateTime(new DateTime(2024,4,2,1,30,0), "Long Date") = "Tuesday, April 2, 2024"  
FormatDateTime(new DateTime(2024,4,2), "Short Date") = "4/2/2024"  
FormatDateTime(new DateTime(1,1,1,1,30,1), "Long Time") = "1:30:01 AM"  
FormatDateTime(new DateTime(1,1,1,1,30,1), "Short Time") = "01:30"
```

## FormatNumber

| Function     | Parameters   | Return value |
|--------------|--------------|--------------|
| FormatNumber | object value | string       |

Formats the specified `value` as a number, using the Windows regional settings.

Example:

```
FormatNumber(1234.56) = "1,234.56"
```

| Function                  | Parameters                                   | Return value        |
|---------------------------|--|---------------------|
| <code>FormatNumber</code> | <code>object value, int decimalDigits</code> | <code>string</code> |

Formats the specified `value` as a number. The `decimalDigits` parameter indicates how many places are displayed to the right of the decimal.

Example:

```
FormatNumber(1234.56, 1) = "1,234.6"
```

## FormatPercent

| Function                   | Parameters                | Return value        |
|----------------------------|---------------------------|---------------------|
| <code>FormatPercent</code> | <code>object value</code> | <code>string</code> |

Formats the specified `value` as a percent, using the Windows regional settings.

Example:

```
FormatPercent(0.15) = "15.00%"
```

| Function                   | Parameters                                   | Return value        |
|----------------------------|--|---------------------|
| <code>FormatPercent</code> | <code>object value, int decimalDigits</code> | <code>string</code> |

Formats the specified `value` as a percent. The `decimalDigits` parameter indicates how many places are displayed to the right of the decimal.

Example:

```
FormatPercent(0.15, 0) = "15%"
```

# Conversion

## ToBoolean

| Function               | Parameters                | Return value      |
|------------------------|---------------------------|-------------------|
| <code>ToBoolean</code> | <code>object value</code> | <code>bool</code> |

Converts the specified `value` to `boolean` .

Example:

```
ToBoolean(1) = true  
ToBoolean(0) = false
```

## ToByte

| Function            | Parameters                | Return value      |
|---------------------|---------------------------|-------------------|
| <code>ToByte</code> | <code>object value</code> | <code>byte</code> |

Converts the specified `value` to `byte` .

Example:

```
ToByte("55") = 55
```

## ToChar

| Function            | Parameters                | Return value      |
|---------------------|---------------------------|-------------------|
| <code>ToChar</code> | <code>object value</code> | <code>char</code> |

Converts the specified `value` to `char` .

Example:

```
ToChar(65) = 'A'
```

## ToDateTime



| Function                | Parameters                | Return value          |
|-------------------------|---------------------------|-----------------------|
| <code>ToDateTime</code> | <code>object value</code> | <code>DateTime</code> |

Converts the specified `value` to `DateTime` .

The separators `.` , `,` and `/` are allowed.

Example:

```

DateTime("4.2.2024") = 04/02/2024 12:00:00 AM
DateTime("4,2,2024") = 04/02/2024 12:00:00 AM
DateTime("4/2/2024") = 04/02/2024 12:00:00 AM

```

## ToDecimal

| Function               | Parameters                | Return value         |
|------------------------|---------------------------|----------------------|
| <code>ToDecimal</code> | <code>object value</code> | <code>decimal</code> |

Converts the specified `value` to `decimal` .

Example:

```

.ToDecimal(1) = 1m
.ToDecimal("1") = 1m

```

## ToDouble

| Function              | Parameters                | Return value        |
|-----------------------|---------------------------|---------------------|
| <code>ToDouble</code> | <code>object value</code> | <code>double</code> |

Converts the specified `value` to `double` .

Example:

```

ToDouble(1) = 1
ToDouble("1") = 1

```

## ToInt32

| Function             | Parameters                | Return value     |
|----------------------|---------------------------|------------------|
| <code>ToInt32</code> | <code>object value</code> | <code>int</code> |

Converts the specified `value` to `int` .

Example:

```
ToInt32(1f) = 1
ToInt32("1") = 1
```

## ToRoman

| Function             | Parameters                | Return value        |
|----------------------|---------------------------|---------------------|
| <code>ToRoman</code> | <code>object value</code> | <code>string</code> |

Converts the specified numeric `value` to its roman representation. The `value` must be in range 1-3998.

Example:

```
ToRoman(9) = "IX"
```

## ToSingle

| Function              | Parameters                | Return value       |
|-----------------------|---------------------------|--------------------|
| <code>ToSingle</code> | <code>object value</code> | <code>float</code> |

Converts the specified `value` to `float` .

Example:

```
ToSingle(1m) = 1f
ToSingle("1") = 1f
```

## ToString

| Function              | Parameters                | Return value        |
|-----------------------|---------------------------|---------------------|
| <code>ToString</code> | <code>object value</code> | <code>string</code> |

Converts the specified `value` to `string` .

Example:

```
ToString(false) = "False"
ToString(DateTime.Now) = "04/02/2024 8:41:00 PM"
```

## ToWords

| Function | Parameters   | Return value |
|----------|--------------|--------------|
| ToWords  | object value | string       |

Converts the specified currency `value` to words.

Example:

```
ToWords(1024.25) = "One thousand and twenty-four dollars and 25 cents"
```

| Function | Parameters                        | Return value |
|----------|-----------------------------------|--------------|
| ToWords  | object value, string currencyName | string       |

Converts the specified currency `value` to words. The `currencyName` parameter indicates the currency. Valid values for this parameter are:

- "USD";
- "EUR";
- "GBP".

Example:

```
ToWords(1024.25, "EUR") = "One thousand and twenty-four euros and 25 cents"
```

| Function | Parameters                            | Return value |
|----------|---------------------------------------|--------------|
| ToWords  | object value, string one, string many | string       |

Converts the specified integer `value` to words. The `one` parameter contains the name in singular form; the `many` parameter contains the name in plural form.

Example:

```
ToWords(124, "page", "pages") = "One hundred and twenty-four pages"
ToWords(1, "page", "pages") = "One page"
```

## ToWordsEnGb

| Function    | Parameters   | Return value |
|-------------|--------------|--------------|
| ToWordsEnGb | object value | string       |

Converts the specified currency `value` to words in Great Britain english. There are the following differences between this function and ToWords:

- the GBP currency is used by default;

- different words used when converting milliards and trilliards.

Example:

```
ToWordsEnGb(121) = "One hundred and twenty-one pounds and 00 pence"
```

| Function                 | Parameters                                     | Return value        |
|--------------------------|--|---------------------|
| <code>ToWordsEnGb</code> | <code>object value, string currencyName</code> | <code>string</code> |

Converts the specified currency `value` to words in Great Britain english. The `currencyName` parameter indicates the currency. Valid values for this parameter are:

- "USD";
- "EUR";
- "GBP".

Example:

```
ToWordsEnGb(1024.25, "EUR") = "One thousand and twenty-four euros and 25 cents"
```

| Function                 | Parameters   | Return value        |
|--------------------------|--|---------------------|
| <code>ToWordsEnGb</code> | <code>object value, string one, string many</code> | <code>string</code> |

Converts the specified integer `value` to words in Great Britain english. The `one` parameter contains the name in singular form; the `many` parameter contains the name in plural form.

Example:

```
ToWordsEnGb(124, "page", "pages") = "One hundred and twenty-four pages"
ToWordsEnGb(1, "page", "pages") = "One page"
```

## ToWordsRu

| Function               | Parameters                | Return value        |
|------------------------|---------------------------|---------------------|
| <code>ToWordsRu</code> | <code>object value</code> | <code>string</code> |

Converts the specified currency `value` to words in russian.

Example:

```
ToWordsRu(1024.25) = "Одна тысяча двадцать четыре рубля 25 копеек"
```

| Function  | Parameters                        | Return value |
|-----------|-----------------------------------|--------------|
| ToWordsRu | object value, string currencyName | string       |

Converts the specified currency `value` to words in russian. The `currencyName` parameter indicates the currency. Valid values for this parameter are:

- "RUR";
- "UAH";
- "USD";
- "EUR".

Example:

```
ToWordsRu(1024.25, "EUR") = "Одна тысяча двадцать четыре евро 25 евроцентов"
```

| Function  | Parameters   | Return value |
|-----------|--|--------------|
| ToWordsRu | object value, bool male, string one, string two, string many | string       |

Converts the specified integer `value` to words in russian. The `male` parameter indicates the gender of the name. The `one`, `two` and `five` parameters contain a form of the name used with "1", "2" and "5" numbers.

Example:

```
// the "страница" word is of female gender, male = false
ToWordsRu(122, false, "страница", "страницы", "страниц") =
"Сто двадцать две страницы"
// the "лист" word is of male gender, male = true
ToWordsRu(122, true, "лист", "листа", "листов") =
"Сто двадцать два листа"
```

# Program flow

## Choose

| Function            | Parameters  | Return value        |
|---------------------|---|---------------------|
| <code>Choose</code> | <code>double index, params object[] choice</code> | <code>object</code> |

Returns an element of the `choice` array with the index specified in the `index` parameter. The first array element has 1 index.

Example:

```
Choose(2, "one", "two", "three") = "two"
```

## IIf

| Function         | Parameters  | Return value        |
|------------------|---|---------------------|
| <code>IIf</code> | <code>bool expression, object truePart, object falsePart</code> | <code>object</code> |

Returns the `truePart` value, if the `expression` is `true`. Otherwise, returns the `falsePart` value.

Example:

```
IIf(2 > 5, "true", "false") = "false"
```

## Switch

| Function            | Parameters                               | Return value        |
|---------------------|--|---------------------|
| <code>Switch</code> | <code>params object[] expressions</code> | <code>object</code> |

The parameter `expressions` consists of pairs in the form "condition-value". It returns the first value for which the condition is `true`.

Example:

```
// returns one of the following values - "a greater than 0",  
// "a less than 0", "a equals to 0", depending on "a" value  
Switch(  
    a > 0, "a greater than 0",  
    a < 0, "a less than 0",  
    a == 0, "a equals to 0")
```

# IsNull

| Function            | Parameters               | Return value      |
|---------------------|--------------------------|-------------------|
| <code>IsNull</code> | <code>string name</code> | <code>bool</code> |

The parameter `name` can represent the name of a database column, a parameter, or a total, and must be enclosed in double quotes.

If the value of the object specified by the parameter `name` is `null`, the method returns `true`; otherwise, it returns `false`.

Example:

```
IsNull("Parameter")
```

# Totals

In many reports, we may need to show some total information: sum of the group, number of rows in the list, and many others. FastReport uses totals to perform this task. For the total, you need to indicate the following parameters:

- the total function type;
- the expression, which is supposed to be calculated. For the `Count` function, you do not need to indicate the expression;
- the condition. The function will be calculated if the condition is met. It's not obligatory to set up the condition.
- the data band, for which the function will be processed;
- the band, in which the total value will be printed.

The list of total functions is given below:

| Function             | Description                                     |
|----------------------|---|
| <b>Sum</b>           | Calculates the sum of the expression.           |
| <b>Min</b>           | Calculates the minimum value of the expression. |
| <b>Max</b>           | Calculates the maximum value of the expression. |
| <b>Average</b>       | Calculates the average value of the expression. |
| <b>Count</b>         | Returns the number of rows.                     |
| <b>CountDistinct</b> | Returns the number of distinct values.          |



# Creating a total

We will look at using the total function as an example. Let us create the "master-detail" report that uses two tables - "Categories" and "Products":



The prepared report will be as follows:

| Beverages                 |                | Condiments                       |                |
|---------------------------|----------------|----------------------------------|----------------|
| Product Name              | Units In Stock | Product Name                     | Units In Stock |
| Chai                      | 39             | Aniseed Syrup                    | 13             |
| Chang                     | 17             | Chef Anton's Cajun Seasoning     | 53             |
| Chartreuse verte          | 69             | Chef Anton's Gumbo Mix           | 0              |
| Côte de Blaye             | 17             | Genen Shouyu                     | 39             |
| Guaraná Fantástica        | 20             | Grandma's Boysenberry Spread     | 120            |
| Ippoh Coffee              | 17             | Gula Malacca                     | 27             |
| Lakkalikööri              | 57             | Louisiana Fiery Hot Pepper Sauce | 76             |
| Laughing Lumberjack Lager | 52             | Louisiana Hot Spiced Oliva       | 4              |
| Outback Lager             | 15             | Northwoods Cranberry Sauce       | 6              |
| Rhinbräu Klosterbier      | 125            | Original Frankfurter grüne Soße  | 32             |
| Sasquatch Ale             | 111            | Sirop d'érable                   | 113            |
| Steeleye Stout            | 20             | Veggie-spread                    | 24             |

Let us add total in this report which will be printing the total quantity of units in stock for each category - sum of `UnitsInStock` data column. Total will be printed in the "Data Footer" band.

To print total value, you need to create it first. For this, press "Action" button in the "Data" window, and choose the "New total" item. Another method - right click the "Totals" element in data tree and choose "New total" menu item. You will see the total editor window.

The screenshot shows the 'Edit Total' dialog box. It contains the following fields and options:

- Total name:** TotalUnits
- Function:** Sum
- Data column or expression:** [Products.UnitsInStock]
- Evaluate on each row of the band:** Data 1: Products
- Evaluate if the following condition is met:** fx
- Print on the band:** DataFooter 1: Products
- Options:**
  - ☒ Reset after print
  - ☐ Reset if band is repeated
  - ☐ Include invisible rows

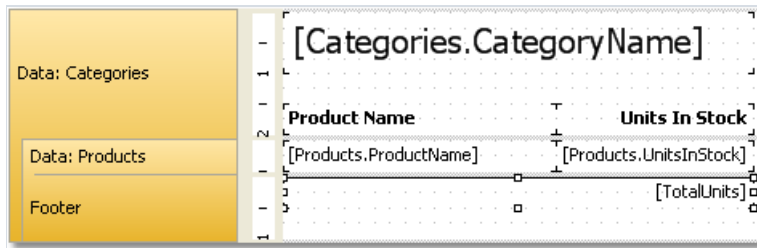
Buttons: OK, Cancel

First of all, you will be asked to indicate total's name. You will be referring to the total by its name, so name the total in such a way that it will be easy to understand as what it calculates. Let us call our total as **TotalUnits** .

Then we choose the **Sum** function for the total.

Now we need to indicate data range for which total will be calculated. For that in "Evaluate on each row of the band:" field, we choose the "Data" band in which a list of products is printed. In the "Print on the band:" field we choose a band in which total will be printed - that is, the "Data Footer" band.

Close editor by pressing OK button. You will see the new total appears in the "Data" window. Now you can drag it into the report:



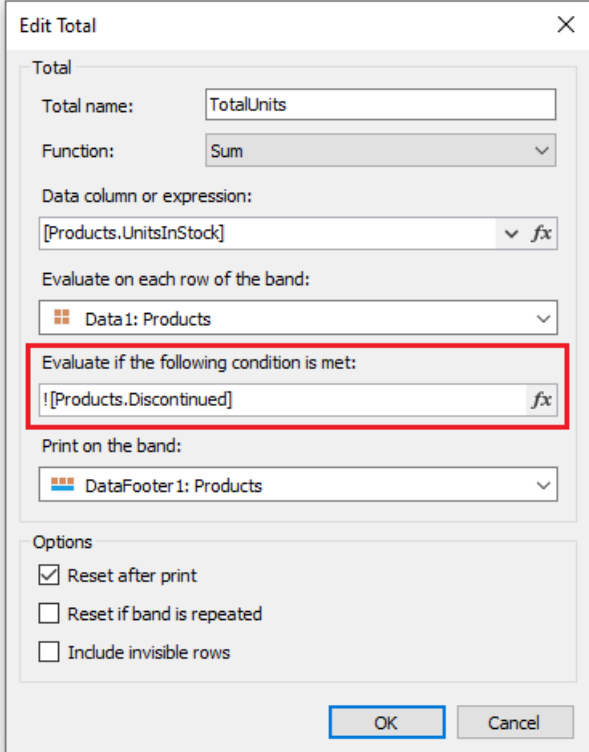
When we run the report, we will see the following:

| Beverages                 |                | Condiments                       |                |
|---------------------------|----------------|----------------------------------|----------------|
| Product Name              | Units In Stock | Product Name                     | Units In Stock |
| Chai                      | 39             | Aniseed Syrup                    | 13             |
| Chang                     | 17             | Chef Anton's Cajun Seasoning     | 53             |
| Chartreuse verte          | 69             | Chef Anton's Gumbo Mix           | 0              |
| Côte de Blaye             | 17             | Genen Shouyu                     | 39             |
| Guaraná Fantástica        | 20             | Grandma's Boysenberry Spread     | 120            |
| Ippoh Coffee              | 17             | Gula Malacca                     | 27             |
| Lakkalikööri              | 57             | Louisiana Fiery Hot Pepper Sauce | 76             |
| Laughing Lumberjack Lager | 52             | Louisiana Hot Spiced Okra        | 4              |
| Outback Lager             | 15             | Northwoods Cranberry Sauce       | 6              |
| Rhônebräu Klosterbier     | 125            | Original Frankfurter grüne Soße  | 32             |
| Sasquatch Ale             | 111            | Sirop d'érable                   | 113            |
| Steeleye Stout            | 20             | Vegie-spread                     | 24             |
|                           | 559            |                                  | 507            |

# Conditional totals

In the previous example, total was calculated for all data rows. We can limit this range by indicating the condition in the total editor. Total will be calculated for only those rows, whose condition returns `true` .

For example, we can set the following conditions:



The screenshot shows the 'Edit Total' dialog box with the following configuration:

- Total**
  - Total name: TotalUnits
  - Function: Sum
  - Data column or expression: [Products.UnitsInStock]
  - Evaluate on each row of the band: Data1: Products
  - Evaluate if the following condition is met: `![Products.Discontinued]` (This section is highlighted with a red border)
  - Print on the band: DataFooter 1: Products
- Options**
  - ☒ Reset after print
  - ☐ Reset if band is repeated
  - ☐ Include invisible rows

Buttons: OK, Cancel

This will mean that, total should be calculated for those products, whose `Discontinued` flag is not set.

# Running totals

In our example totals were reset after printing the "Data Footer" band. This occurred because we indicated in total editor that it is necessary to reset the total after printing it. As a result, each category printed its own total values.

If we uncheck the "Reset after print" checkbox, the total will not be reset after printing. This is what is called running total.

If you need to print two types of totals at the same time - ordinary totals and running totals - create one more total with similar settings and uncheck the "Reset after print" flag.

## Page totals

In order create a total which will be printed on the page footer, you will have the indicate page footer in the "Print on the band:" field.

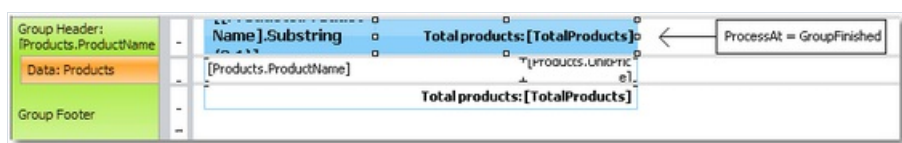
# Printing the total in the header

Usually you will print total values on the footer bands (such as data footer, group footer, etc). It's a natural printing order because when you print the total, its value is properly calculated and is ready to use. However, in some cases, you would need the total to be printed on the header (for example, on the group header). If you try to do this, you will see a zero value. At this moment, when you print a total, it is not calculated yet.

To solve this problem, FastReport has a feature called "delayed print". The "Text" object has a property called `ProcessAt` which can have one of the following values:

| Value              | Description   |
|--------------------|---|
| Default            | The default printing mode. This is the default value.   |
| ReportFinished     | The object's value will be calculated at the end of the report.                                 |
| ReportPageFinished | The object's value will be calculated when all bands in the page will be finished.              |
| PageFinished       | The object's value will be calculated at the end of the page.                                   |
| ColumnFinished     | The object's value will be calculated at the end of the column.                                 |
| DataFinished       | The object's value will be calculated at the end of the data band (when its footer is printed). |
| GroupFinished      | The object's value will be calculated at the end of the group (when its footer is printed).     |

Let's look at how it works. Put the "Text" object which prints the total, on the group header. Set the `ProcessAt` property of the "Text" object to `GroupFinished` :



When you run the report, FastReport will do the following:

- it will print the group header. The total value will be printed as 0 (wrong), but FastReport will remember this object to process it later;
- it will print all data rows;
- it will print the group footer. At this moment, FastReport will take the object that was printed in the group header, and process it again to print the correct total value.

The prepared report will be as follows:

| A             |  | Total products: 2 |
|---------------|--|-------------------|
| Alice Mutton  |  | \$39.00           |
| Aniseed Syrup |  | \$10.00           |
|               |  | Total products: 2 |

| B                |  | Total products: 1 |
|------------------|--|-------------------|
| Boston Crab Meat |  | \$18.40           |
|                  |  | Total products: 1 |

| C                 |  | Total products: 9 |
|-------------------|--|-------------------|
| Camembert Pierrot |  | \$34.00           |
| Carnarvon Tigers  |  | \$62.50           |
| Chai              |  | \$18.00           |
|                   |  | \$19.00           |

Using other values of the `ProcessAt` property, you may print the report total in the report title (set `ProcessAt = ReportFinished`), or print the page total in the page header (set `ProcessAt = PageFinished`).

The delayed print feature will not work if you turn on the report file cache ("Report|Options..." menu, "Use file cache" checkbox).

# Report parameters

You can define parameters in a report. Parameter is a variable, the value of which can be defined both in a report itself and outside of it (a program, calling a report can transfer parameters values into it. See details in "Programmer's manual"). A parameter can be used in expressions and be displayed in report objects like the "Text" object.

Most common methods of using parameters:

- data filtering by condition set in a parameter;
- printing parameter value in a report.

A parameter has the following properties:

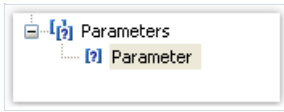
| Property          | Description  |
|-------------------|--|
| <b>Name</b>       | Parameter's name can have any symbols except dot <code>.</code> .  |
| <b>DataType</b>   | Parameter data type.   |
| <b>Expression</b> | Expression which returns parameter's value. More details about expressions can be found in the "Expression" chapter. This expression will be processed when calling a parameter. |
| <b>Value</b>      | Parameter value. This property is not available in the designer and can be filled programmatically.  |

You have to set up `Name` and `DataType` properties. The `Expression` property can be left empty. In this case parameter's value should be passed programmatically.



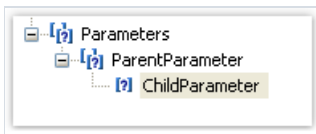
# Creating a parameter

To create a parameter, select the "Parameters" element in the "Data" window, right-click it and choose the "New parameter" item in a context menu:



Press F2 and give a parameter name, then go to the "Properties" window and set the parameter's **DataType** property.

Parameters can be nested. To create a nested parameter, select a parent parameter, right-click it and choose the "New parameter" item in a context menu:



You can refer to both, parent parameter and nested one. Nesting level is not limited.

# Using parameters in a report

You can refer to a parameter from an expression using square brackets:

```
[Parameter name]
```

To a nested parameter you need to refer using this method:

```
[Parent parameter.Child parameter]
```

Since a parameter has got a definite type (it is given in the `DataType` property), then with parameters, you can perform those actions which are allowed for data type. So, string type parameters can be used in an expression the following way:

```
[StringParameter].Substring(0, 2)
```

Let us see one example of using parameters. Assuming we have a report which prints "Employees" table. We want to modify the report to print information about an employee with an indicated number. To do this, we need to filter the data on the `EmployeeID` data column. Create a parameter with "EmployeeID" name. Indicate parameter's type - `Int32`, as exactly this type has the `EmployeeID` data column. To filter an employee with an indicated ID we need to enter "Data" band editor and indicate the following expression in "Filter" tab:

```
[Employees.EmployeeID] == [EmployeeID]
```

To pass parameter value from your program to the report, use the following code:

```
report1.SetParameterValue("EmployeeID", 2);
```

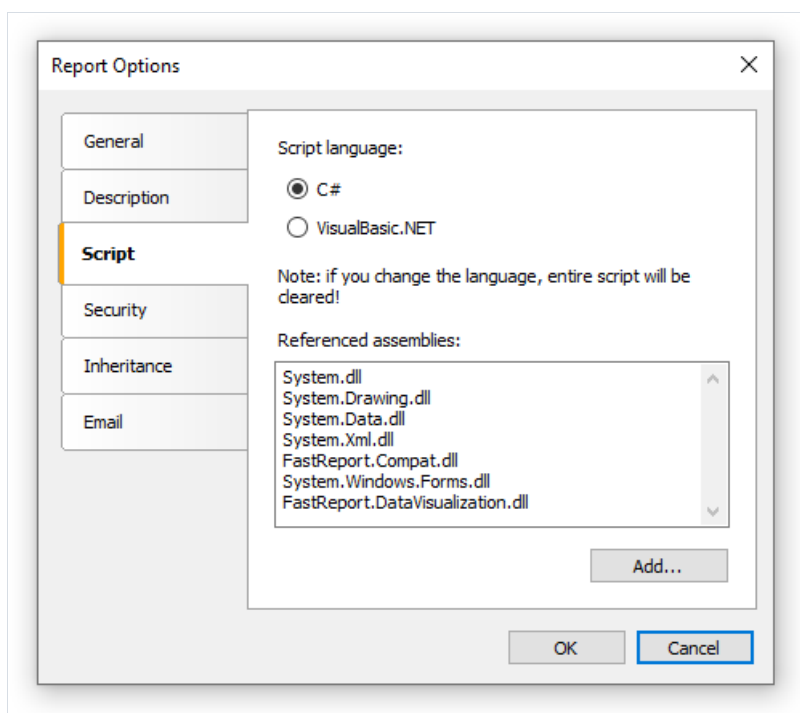
# Expressions

In many places in FastReport, expressions are used. For example, the "Text" object can contain expressions in square brackets.

An expression is a code in C# or VB.NET language, which returns any value. For example:

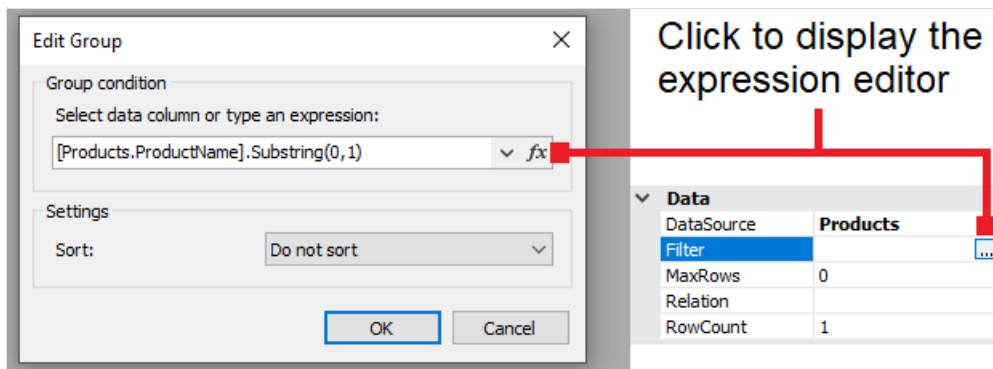
```
2 + 2
```

An expression should be written in a language chosen as a script in a report. By default, it is C#. You can change the language in the "Report|Options..." menu by choosing the "Script" element in a window.

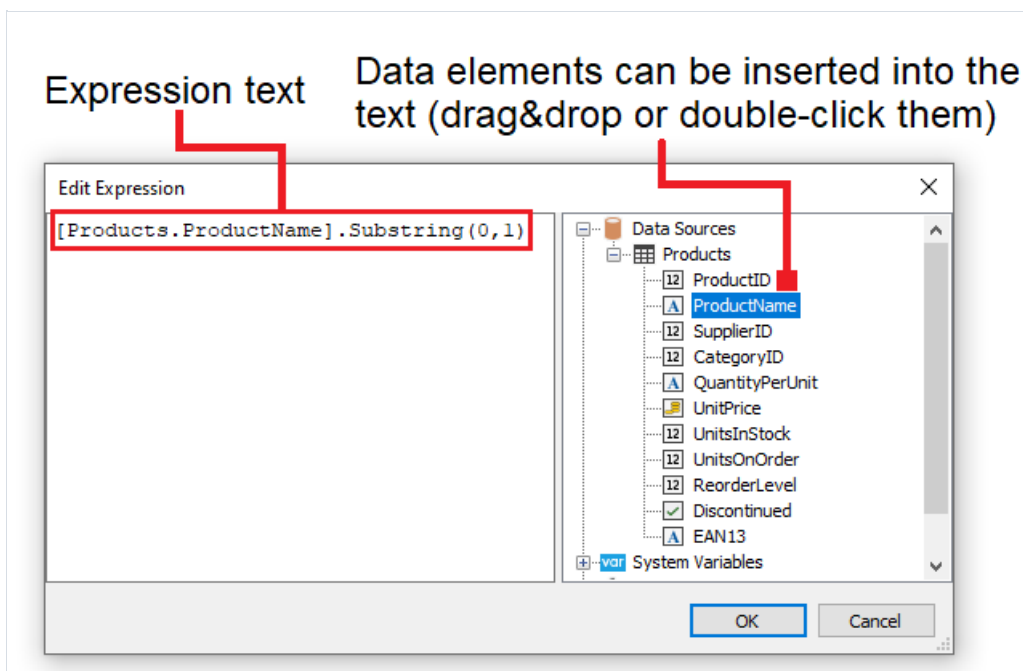


# Expression editor

To write an expression quickly, use the expression editor. It can be invoked in such places of FastReport UI, where you can type an expression:



The expression editor presents a window where you can write an expression and insert some data elements into it:



# Reference to report objects

For referring to report objects, use the object's name. The following example will return the height of the Text1 object:

```
Text1.Height
```

For referring to report properties, use `Report` variable. The following example returns file name from which a report was loaded.

```
Report.FileName
```

Besides, you can refer to nested object properties. The following example will return a report name:

```
Report.ReportInfo.Name
```

# Using .NET functions

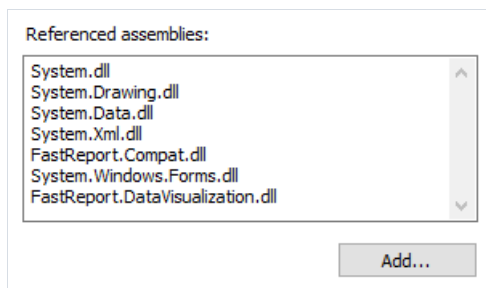
You can use any .NET objects in expressions. The following example demonstrates `Max` function use:

```
Math.Max(5, 10)
```

By default a report uses the following .NET assemblies:

```
System.dll
System.Drawing.dll
System.Windows.Forms.dll
System.Data.dll
System.Xml.dll
```

You have access to all .NET objects declared in these assemblies. If you need to have access to another assembly, add its name in report assemblies list. You can do it in the "Report[Options...]" menu, by choosing the "Script" element in a window:



For example, if you want to use a function in your report which was declared in your application, add application assembly (.exe or .dll) in a report assemblies list. After that you can call the function by using namespace of your application. For example, if the following function is defined in application:

```
namespace Demo
{
    public static class MyFunctions
    {
        public static string Func1()
        {
            return "Hello!";
        }
    }
}
```

You can use it in your report in the following way:

```
Demo.MyFunctions.Func1()
```

If you add the `using Demo` line at the top of the report's script, it will allow you to shorten the syntax:

```
MyFunctions.Func1()
```

To refer to the function or variable that was defined in a script, just use its name:

```
myPrivateVariableThatIHaveDeclaredInScript  
MyScriptFunction()
```

You can use in an expression only those functions which return a value.

# Reference to data elements

Apart from standard language elements, you can use the following report elements in expressions:

- data source columns;
- system variables;
- total values;
- report parameters.

All these elements are contained in the "Data" window. See more details in the "[Data](#)" chapter. Any of these elements can be used in an expression, by including it in square brackets. For example:

```
[Page] + 1
```

This expression returns next printed page number. A system variable `Page`, which returns current report page number, is used in the expression. It is enclosed in square brackets.



# Reference to data sources

For reference to data source columns, the following format is used:

```
[DataSource.Column]
```

Source name is separated from column name by a period, for example:

```
[Employees.FirstName]
```

Source name can be compound in case, if we refer to a data source by using a relation. See more details in the "Data" section. For example, this is how you can refer to a related data source column:

```
[Products.Categories.CategoryName]
```

Let us look at the following example of using columns in an expression:

```
[Employees.FirstName] + " " + [Employees.LastName]
```

Here it should be noted that: every column has a definite data type, which is set in the `DataType` property (you can see it in the "Properties" window if to choose data column in the "Data" window beforehand). How a column can be used in an expression depends on its type. For instance, in above mentioned example, both columns - first name and last name - have got a string type and that is why they can be used in such a way. In the following example, we will try to use `Employees.Age` column of numeric type, which will lead to an error:

```
[Employees.FirstName] + " " + [Employees.Age]
```

The error occurs because, you never mix strings with numbers. For this, you need to convert the number into a string:

```
[Employees.FirstName] + " " + [Employees.Age].ToString()
```

In this case, we refer to the `Employees.Age` column as if it is an integer variable. And it is so. We know that all expressions are compiled. All non-standard things (like referring to data columns) from a compiler's point of view are converted into another type, which is understandable to a compiler. So, the last expression will be turned into the following form:

```
(string)(Report.GetColumnValue("Employees.FirstName")) + " " +  
(int)(Report.GetColumnValue("Employees.Age")).ToString()
```

As seen, FastReport changes reference to data columns in the following way:

```
[Employees.FirstName] --> (string)(Report.GetColumnValue("Employees.FirstName"))
```

```
[Employees.Age] --> (int)(Report.GetColumnValue("Employees.Age"))
```

That is, we can use data column in an expressions as if it is a variable having a definite type. For example, the following expression will return the first symbol of an employee's name:

```
[Employees.FirstName].Substring(0, 1)
```

# Reference to system variables

You can use the following system variables in expressions (they are accessible in the "Data" window):

| Variable          | .Net data type | Description   |
|-------------------|----------------|---|
| <b>Date</b>       | DateTime       | Date and time of the report's start.  |
| <b>Page</b>       | int            | Current page number.  |
| <b>TotalPages</b> | int            | Total number of pages in the report. To use this variable, you need to enable the report's double pass. You can do this in "Report Properties..." menu. |
| <b>PageN</b>      | string         | Page number in the form: "Page N".  |
| <b>PageNofM</b>   | string         | Page number in the form: "Page N of M".   |
| <b>Row#</b>       | int            | Data row number inside the group. This value is reset at the start of a new group.  |
| <b>AbsRow#</b>    | int            | Absolute number of data row. This value is never reset at the start of a new group.   |

Every variable has a definite data type. And on this, depends how it will be used in the expression. Here is an example of an expression where date is being used:

```
[Date].Year
```

This expression returns the current year. As `Date` variable has `DateTime` type, we can refer to its `Year` property. We can get the current month similarly ( `[Date].Month` ).

FastReport converts reference to system variable into the following form (for example, the `Date` variable):

```
((DateTime)Report.GetVariableValue("Date"))
```

# Reference to total values

In order to refer to a total value, use its name:

```
[TotalSales]
```

FastReport converts reference to totals into the following form:

```
Report.GetTotalValue("TotalSales")
```

As you can see, the data type is not used here. It is so, because the total value is of `FastReport.Variant` type. It can be used directly in any expressions because it is automatically converted to any type. For example:

```
[TotalSales] * 0.2f
```

# Reference to report parameters

In order to refer to report parameters, you should use its name:

```
[Parameter1]
```

Parameters can be nested. In this case, you should use both parent and child parameter names in the following form:

```
[ParentParameter.ChildParameter]
```

Parameters have a definite data type. It is set in the `DataType` property of the parameter. The way it can be used in an expression depends on parameter's data type.

FastReport converts reference to a report parameter into the following way:

```
((string)Report.GetParameterValue("Parameter1"))
```

# Script

Script is a higher-level programming language, which is part of the report. Script can be written in one of the following .NET languages:

- C#
- VisualBasic.NET

A script is applicable in many places. Using the script, you can do the following:

- perform data handling, which cannot be done via regular means of the FastReport engine;
- control the printing of report pages and bands on the page;
- control the interaction between elements on dialogue forms;
- control the formation of dynamic "Table" objects;
- and many more.

In order to see the report's script, switch to the "Code" tab in the designer:

The screenshot displays the FastReport designer interface. The main window is titled 'FastReport - C:\Program Files (x86)\FastReports\FastReport.Net\Demos\Reports\Simple List.frx\*'. The 'Code' tab is selected, showing a C# script for a 'ReportScript' class. The script includes using statements for System, System.Collections, System.ComponentModel, System.Windows.Forms, System.Drawing, System.Data, and FastReport namespaces. It defines a 'ReportScript' class with a 'Picture1\_Click' event handler. The 'Properties' window on the right shows the 'Picture1' control selected, with the 'Click' event handler set to 'Picture1\_Click'. The 'Data' pane on the right shows a data source named 'Employees' with fields 'EmployeeID', 'LastName', and 'FirstName'. Red arrows point from text labels to specific elements in the interface: 'Main class' points to the 'ReportScript' class definition; 'Data elements can be dragged into the script' points to the 'Data' pane; 'Code' tab points to the 'Code' tab in the bottom toolbar; 'The event handler that was created in the Properties window' points to the 'Picture1\_Click' event handler in the 'Properties' window; and 'In the Properties window, you can create event handlers by double-clicking the event' points to the 'Click' event in the 'Properties' window.

**Main class**

**Data elements can be dragged into the script**

**"Code" tab**

**The event handler that was created in the "Properties" window**

**In the "Properties" window, you can create event handlers by double-clicking the event**

Script language can be set in the "Report|Options..." menu. This is supposed to be done just after you have created a new report, because when changing the language, the existing script gets deleted.

Report Options

General

Description

**Script**

Security

Inheritance

Email

Script language:

☒ C#  
☐ VisualBasic.NET

Note: if you change the language, entire script will be cleared!

Referenced assemblies:

System.dll  
System.Drawing.dll  
System.Data.dll  
System.Xml.dll  
FastReport.Compat.dll  
System.Windows.Forms.dll  
FastReport.DataVisualization.dll

^  
v

Add...

OK

Cancel

# General information

Contrary to other report generators, script in FastReport contains only what you have written. In the script, you can:

- add your variables, methods and properties to the main script class;
- create a report object's events handler;
- add new classes to the script, if needed. A class can be added either before the `ReportScript` main class or after it.

You cannot:


- delete, rename or change the visibility area of the `ReportScript` main class;
- rename a namespace in which the main class is located.

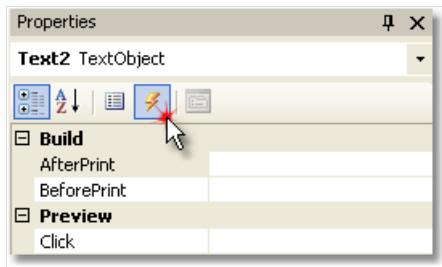
When the report is running, the following occurs:

- FastReport adds into the script a list of variables, whose names corresponds with the names of the report objects. This is done before compiling the script and allows you to refer to the report objects by their names;
- an expression handler is added to the script, which handles all expressions found in the report;
- a script is compiled, if it is not empty;
- the script class is initialized;
- the report is run.



# Event handlers

A script is mainly used for creating objects' event handlers. For creating event handler select the needed object. In the "Properties" window, press the  button, in order to switch on the list of events:



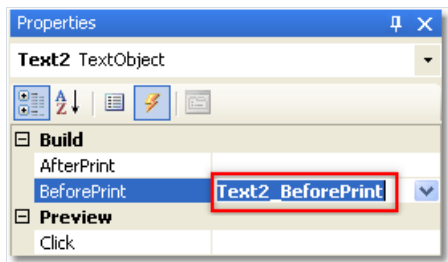
Select the event you want and double click it. FastReport adds an empty event handler into the report code:

```
private void Text2_BeforePrint(object sender, EventArgs e)
{
}
}
```

The "Report" object has got events as well. This object can be chosen by the following method:

- select "Report" in the "Report Tree" window;
- select "Report" in the drop-down list in the "Properties" window.

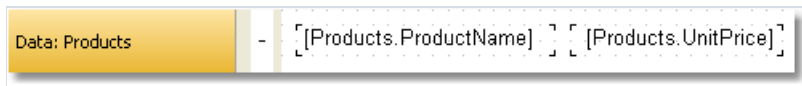
In order to delete the event handler, choose an event in the "Properties" window, select the text and press the Delete key:



# Report events

In order to control the report with maximum flexibility, every report object has got several events. For example, in a handler, connected to the "Data" band, you can filter records, that is, hide or show the band depending on certain conditions.

Let us consider the events which are fired during the report generation process. As an example, we will take a simple report, containing one page, one "Data" band and two "Text" objects on the band:



In the beginning of the report, the "Report" object fires the `StartReport` event. Before formation of the report page, the `StartPage` event is fired. This event is fired once for every template page (do not confuse with prepared report page!). In our case, regardless of how many pages were in the prepared report - event is fired once, since the template report has got one page. Then the `CreatePage` event is called, which occurs just at the moment when the page is created in the prepared report.

Further, printing of the "Data" band row starts. This happens in the following way:

1. the `BeforePrint` band event is fired;
2. the `BeforePrint` event of all objects lying on the band is fired;
3. all objects are filled with data;
4. the `AfterData` event of all objects lying on the band is fired;
5. the `BeforeLayout` band event is fired;
6. objects are placed on the band, the height of the band is calculated and band is stretched (if it can);
7. the `AfterLayout` band event is fired;
8. if the band cannot fit on a free space on the page, a new page is formed;
9. the band and all its objects are displayed on a prepared report page;
10. the `AfterPrint` band event is fired;
11. the `AfterPrint` event of all the band objects is fired.

Printing of the band row occurs as long as there is data in the source. After this, the formation of the report in our case ends. The `FinishPage` event of a page is fired and finally - the `FinishReport` event of the "Report" object.

So, by using events of different objects, you can control every step of report formation. The key to correct use of events - full understanding of the band printing process, expound in the eleven steps above.

So, a lot of operations can be done, by using only the `BeforePrint` band - any change, done to the object, will also be displayed. But in this event, it is not possible to analyze, on which page will the band be printed, if it stretches, because the height of the band will be calculated on step 6. This can be done with the help of the `AfterLayout` event in step 7 or `AfterPrint` in step 10, but in the latter case, the band is already printed and operations with objects do not give out anything.

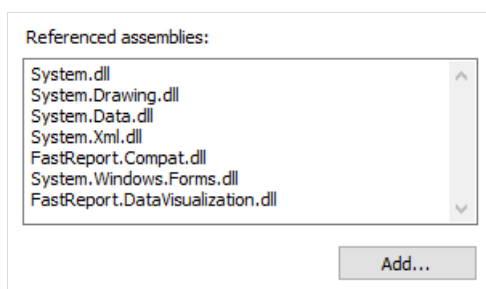
In one word, you must clearly state, at what moment each event is fired and use, those, which correspond with the given task.

# Using .NET objects

In a script, you can use any .NET objects, which are defined in the following assemblies:

```
System.dll
System.Drawing.dll
System.Windows.Forms.dll
System.Data.dll
System.Xml.dll
```

Apart from that, you can use any object, defined in the FastReport assembly. If you need access to another assembly, add it to the list of assemblies. This can be done in the "Report|Options..." menu, by choosing the "Script" tab:



For example, if you want to use a function in your report which was declared in your application, add application assembly (.exe or .dll) in a report assemblies list. After that you can call the function by using namespace of your application. For example, if the following function is defined in application:

```
namespace Demo
{
    public static class MyFunctions
    {
        public static string Func1()
        {
            return "Hello!";
        }
    }
}
```

Calling it in the script can be done in the following way:

```
string hello = Demo.MyFunctions.Func1();
```

If you add the `using Demo` line at the top of the report's script, it will allow you to shorten the syntax:

```
string hello = MyFunctions.Func1();
```

# Reference to report objects

For referring to report objects (for example, "Text" object) use the name of the object. The following example returns the height of Text1 object:

```
float height = Text1.Height;
```

Note that report's native unit of measurement is screen pixels. Keep it in mind when using such object's properties like `Left`, `Top`, `Width`, and `Height`. To convert pixels into centimeters and back, use the constants, defined in the `Units` class:

```
float heightInPixels = Text1.Height;  
float heightInCM = heightInPixels / Units.Centimeters;  
Text1.Height = Units.Centimeters * 5; // 5cm
```

# Report and Engine objects

Apart from objects, which are contained in the report, there are two variables defined in the script: `Report` and `Engine`.

The `Report` variable refers to the current report. In the list below, a list of the report object's methods is given:

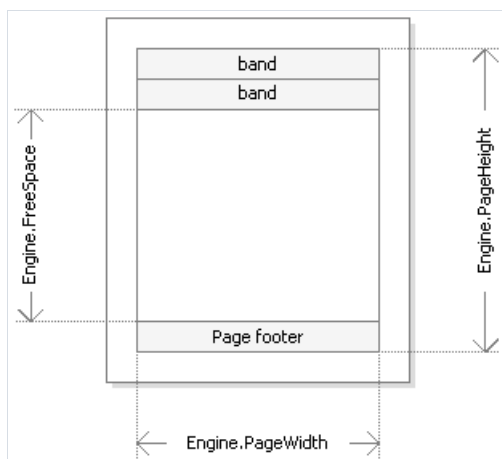
| Method  | Description   |
|---|---|
| <code>object Calc(string expression)</code>                           | Calculates an expression and returns the value. When calling this method the first time, an expression gets compiled, which needs some time.  |
| <code>object GetColumnValue(string complexName)</code>                | Returns the value of the data column. The name must be presented in the "DataSource.Column" form. If the column has got the null value, it is converted into a value by default (0, empty string, <code>false</code> ). |
| <code>object GetColumnValueNullable(string complexName)</code>        | Returns the value of the data column. Contrary to the previous method, it does not get converted into a default value and may be null.  |
| <code>Parameter GetParameter(string complexName)</code>               | Returns the reports parameter with the indicated name. Name can be compounded when referring to the nested parameter: <code>MainParam.NestedParam</code> .  |
| <code>object GetParameterValue(string complexName)</code>             | Returns the value of the report parameter with the indicated name.  |
| <code>void SetParameterValue(string complexName, object value)</code> | Sets the value of the report parameter with the indicated name.   |
| <code>object GetVariableValue(string complexName)</code>              | Returns the value of the system variable, for example, <code>Date</code> .  |
| <code>object GetTotalValue(string name)</code>                        | Returns the value of the total, defined in the "Data" window, by its name.  |
| <code>DataSourceBase GetDataSource(string alias)</code>               | Returns the data source, defined in the report, by its name.  |

The Engine object is an engine that controls the report creation. By using the methods and properties of the engine, you can manage the process of placing bands onto the page. You can use the following properties of the Engine object:

| Property                   | Description   |
|----------------------------|---|
| <code>float CurX</code>    | Current coordinates on the X-axis. This property can be assigned a value, so as to shift the printed object.          |
| <code>float CurY</code>    | Current printing position on the Y-axis. To this property, a value can be assigned so as to shift the printed object. |
| <code>int CurColumn</code> | Number of the current column in a multicolumn report. The first column has the number 0.                              |

| Property                              | Description  |
|---------------------------------------|--|
| <code>int CurPage</code>              | Number of the page being printed. This value can be received from the <code>Page</code> system variable.   |
| <code>float PageWidth</code>          | Width of the page minus the size of the left and right margins.  |
| <code>float PageHeight</code>         | Height of the page minus the size of the top and bottom margins.   |
| <code>float PageFooterHeight</code>   | Height of the page footer (and all its child bands).   |
| <code>float ColumnFooterHeight</code> | Height of the column footer (and all of its child bands).  |
| <code>float FreeSpace</code>          | Size of the free space on the page.  |
| <code>bool FirstPass</code>           | Returns <code>true</code> , if the first (or only) report pass is being executed. Number of passes can be obtained from the <code>Report.DoublePass</code> property. |
| <code>bool FinalPass</code>           | Returns <code>true</code> , if the last (or only) report pass is being executed.   |

On the figure below, you can see the meaning of some properties listed above.



`Engine.PageWidth` and `Engine.PageHeight` properties determine the size of the printing area, which is almost always less than the actual size of the page. Size of the printed area is determined by the page margins, which is given by the `LeftMargin` , `TopMargin` , `RightMargin` and `BottomMargin` page properties.

`Engine.FreeSpace` property determines the height of the free space on the page. If there is the "Page footer" band on the page, its height is considered when calculating the `FreeSpace` . Note that, after printing a band, free space is reduced.

How does the formation of a prepared report page take place? FastReport engine displays bands on the page until there is enough space for band output. When there is no free space, the "Report footer" band is printed and a new empty page is formed. Displaying a band starts from the current position, which is determined by the X and Y coordinates. This position is returned by the `Engine.CurX` and `Engine.CurY` properties. After printing a band, `CurY` automatically increases by the height of the printed band. After forming a new page, the position of the `CurY` is set to 0. The position of the `CurX` changes when printing a multicolumn report.

`Engine.CurX` and `Engine.CurY` properties are accessible not only for reading, but also for writing. This means that you can shift a band manually by using one of the suitable events. Examples of using these properties can be seen in the "Examples" section.

When working with properties, which return the size or position, remember that, these properties are measured in the screen pixels.

In the Engine object, the following methods are defined:

| Method  | Description  |
|---|--|
| <code>void AddOutline(string text)</code>     | Adds an element into the report outline (read the chapter " <a href="#">Interactive reports</a> ") and sets the current position to the added element. |
| <code>void OutlineRoot()</code>               | Sets the current position on the root of the outline.  |
| <code>void OutlineUp()</code>                 | Shifts the current position to a higher-level outline element.   |
| <code>void AddBookmark(string name)</code>    | Adds a bookmark (read the chapter " <a href="#">Interactive reports</a> ").  |
| <code>int GetBookmarkPage(string name)</code> | Returns the page number on which the bookmark with the indicated name is placed.   |
| <code>void StartNewPage()</code>              | Starts a new page. If the report is multicolumn, a new column is started.  |

By using the `AddOutline` , `OutlineRoot` , `OutlineUp` methods, you can form the report outline manually. Usually, this is done automatically with the help of the `OutlineExpression` property, which every band and report page have got.

The `AddOutline` method adds a child element to the current outline element, and makes it current. The current report page and the current position on the page are associated with the new element. If you call the `AddOutline` method several times, then you will have the following structure:

```
Item1
  Item2
    Item3
```

For controlling the current element, there are `OutlineUp` and `OutlineRoot` methods. The first method moves the pointer to the element, located on a higher level. So, the script

```
Engine.AddOutline("Item1");
Engine.AddOutline("Item2");
Engine.AddOutline("Item3");
Engine.OutlineUp();
Engine.AddOutline("Item4");
```

will create the following outline:

```
Item1
  Item2
    Item3
    Item4
```

The `OutlineRoot` method moves the current element to the root of the outline. For example, the following script:

```
Engine.AddOutline("Item1");  
Engine.AddOutline("Item2");  
Engine.AddOutline("Item3");  
Engine.OutlineRoot();  
Engine.AddOutline("Item4");
```

will create the following outline:

```
Item1  
  Item2  
    Item3  
Item4
```

For working with bookmarks, the `AddBookmark` and `GetBookmarkPage` methods of the Engine object are used. Usually bookmarks are added automatically when using the `Bookmark` property, which all objects of the report have got.

By using the `AddBookmark` method, you can add a bookmark programmatically. This method creates a bookmark on the current page at the current printing position.

The `GetBookmarkPage` method returns the page number on which the bookmark is placed. This method is often used when creating the table of contents, for displaying page numbers. In this case, the report must have a double pass.



# Reference to data sources

Contrary to the FastReport expressions (covered in the ["Expressions"](#) section), never use square brackets in script for referring to the data sources. Instead of this, use the `GetColumnValue` method of the Report object, which returns the value of the column:

```
string productName = (string)Report.GetColumnValue("Products.Name");
```

As seen, you need to indicate the name of the source and its column. The name of the source can be compound in case, if we are referring to the data source by using a relation. Details about relations can be found in the ["Data"](#) chapter. For example, you can refer to a column of the related data source in this way:

```
string categoryName = (string)Report.GetColumnValue("Products.Categories.CategoryName");
```

For making the work easier, use the "Data" window. From it you can drag data elements into the script, during this FastReport automatically creates a code for referring to the element.

For referring to the data source itself, use the `GetDataSource` method of the Report object:

```
DataSourceBase ds = Report.GetDataSource("Products");
```

Help on properties and methods of the `DataSourceBase` class can be received from the FastReport .NET Class Reference help system. As a rule, this object is used in the script in the following way:

```
// get a reference to the data source
DataSourceBase ds = Report.GetDataSource("Products");
// initialize it
ds.Init();
// enum all rows
while (ds.HasMoreRows)
{
    // get the data column value from the current row
    string productName = (string)Report.GetColumnValue("Products.Name");
    // do something with it...
    // ...
    // go next data row
    ds.Next();
}
```

# Rererence to system variables

For reference to system variables, use the `GetVariableValue` method of the Report object:

```
DateTime date = (DateTime)Report.GetVariableValue("Date");
```

A list of system variables can be seen in the "Data" window. From it, you can drag a variable into a script, during this FastReport automatically creates a code for referring to the variable.

# Reference to total values

For reference to the total value, use the `GetTotalValue` method of the Report object:

```
float sales = Report.GetTotalValue("TotalSales");
```

A list of totals can be seen in the "Data" window. From it, you can drag a total into the script, during this FastReport automatically creates a code for referring to the total.

Total value has got the `FastReport.Variant` type. It can be used directly in any expression, because the `FastReport.Variant` type is automatically converted to any type. For example:

```
float tax = Report.GetTotalValue("TotalSales") * 0.2f;
```

Reference to the total value can be done at that time when, it is being processed. Usually the total is "ready to use" at the moment of printing the band, on which it is located in the report.

# Reference to report parameters

For referring to report parameters, use the `GetParameterValue` method of the Report object:

```
int myParam = (int)Report.GetParameterValue("MyParameter");
```

Parameters can be nested. In this case, indicate the name of the parent parameter and after the period, the name of the child parameter:

```
Report.GetParameterValue("ParentParameter.ChildParameter")
```

Parameters have got a definite data type. It is given in the `DataType` property of the parameter. You must take this into account when referring to parameters. You can see a list of parameters in the "Data" window. From it, you can drag parameters into the script, during this FastReport automatically creates a code for referring to the parameters.

For changing the value of the parameter, use the `SetParameterValue` method of the report object:

```
Report.SetParameterValue("MyParameter", 10);
```

# Examples

## Example 1. Changing object's appearance

In this example we will show how to change the color of the text depending on the value printed in the object. We will be using:

- the `BeforePrint` event;
- reference to the data column from script.

Create a simple report having the following appearance:



Select the object, which prints the `UnitPrice` column, and create a `BeforePrint` event handler:

```
private void Text2_BeforePrint(object sender, EventArgs e)
{
    if (((Decimal)Report.GetColumnValue("Products.UnitPrice")) > 20)
        Text2.TextColor = Color.Red;
}
```

In order to insert the `Products.UnitPrice` data column into the script, drag it from the "Data" window. During this, the following string will be added in the script:

```
((Decimal)Report.GetColumnValue("Products.UnitPrice"))
```

If we run the report, we will see that all the products, having the price > 20, are highlighted in red:

|                              |       |
|------------------------------|-------|
| Chai                         | 18,00 |
| Chang                        | 19,00 |
| Aniseed Syrup                | 10,00 |
| Chef Anton's Cajun Seasoning | 22,00 |
| Chef Anton's Gumbo Mix       | 21,35 |
| Grandma's Boysenberry Spread | 25,00 |

The same effect can be achieved with the help of the conditional highlighting (you can read more about this in the ["Conditional highlighting"](#) section of the "Report creation" chapter).

## Example 2. Highlighting even rows of the band

In this example we will show how to change the fill color of the "Data" band's even rows. We will be using:

- `BeforePrint` band event;
- reference to the `Row#` system variable from the script.

Create a simple report having the following appearance:

|                |   |                        |                      |
|----------------|---|------------------------|----------------------|
| Data: Products | - | [Products.ProductName] | [Products.UnitPrice] |
|----------------|---|------------------------|----------------------|

Create a `BeforePrint` event handler for the band:

```
private void Data1_BeforePrint(object sender, EventArgs e)
{
    if (((Int32)Report.GetVariableValue("Row#")) % 2 == 0)
        Data1.FillColor = Color.Gainsboro;
}
```

The `Row#` system variable returns the number of the row of the printed band. In order to insert into the script a reference to the variable, drag it from the "Data" window. During this, in the script a string will be inserted:

```
((Int32)Report.GetVariableValue("Row#"))
```

If we run the report, we will see that even rows will be highlighted in light-gray color:

|                              |       |
|------------------------------|-------|
| Chai                         | 18,00 |
| Chang                        | 19,00 |
| Aniseed Syrup                | 10,00 |
| Chef Anton's Cajun Seasoning | 22,00 |
| Chef Anton's Gumbo Mix       | 21,35 |

The same effect can be achieved with the help of the `EvenStyle` property of the "Data" band. You can read more about this in the ["Highlight odd/even data rows"](#) section of the "Report creation" chapter.

## Example 3. Data filtering

In this example, we will show how to hide the "Data" band row depending on the given conditions. We will be using:

- `BeforePrint` band event;
- reference to the data source from a script.

Create a simple report having the following appearance:

|                |   |                        |                      |
|----------------|---|------------------------|----------------------|
| Data: Products | - | [Products.ProductName] | [Products.UnitPrice] |
|----------------|---|------------------------|----------------------|

Create a `BeforePrint` event handler for the band:

```
private void Data1_BeforePrint(object sender, EventArgs e)
{
    if (((Decimal)Report.GetColumnValue("Products.UnitPrice")) > 20)
        Data1.Visible = false;
}
```

In the given case, the band rows which have the unit price > 20 will be hidden:

|               |       |
|---------------|-------|
| Chai          | 18,00 |
| Chang         | 19,00 |
| Aniseed Syrup | 10,00 |
| Konbu         | 6,00  |
| Genen Shouyu  | 15,50 |
| Pavlova       | 17,45 |

The same effect can be achieved by using the data filter which can be set in the "Data" band editor.



## Example 4. Calculating total

In this example, we will show how to calculate the sum by using programming methods. We will use the following:

- `BeforePrint` band event;
- reference to the data column from a script;
- local variable, whose value will be printed in the report.

Create a report of the following form:

|                |   |                        |                      |
|----------------|---|------------------------|----------------------|
| Data: Products | - | [Products.ProductName] | [Products.UnitPrice] |
| Report Summary | - | Total: [sum]           |                      |

In the script, declare the `sum` variable and create a `BeforePrint` event handler belonging to the band:

```
public class ReportScript
{
    private decimal sum;

    private void Data1_BeforePrint(object sender, EventArgs e)
    {
        sum += (Decimal)Report.GetColumnValue("Products.UnitPrice");
    }
}
```

The `Products.UnitPrice` data column can be placed into the script, dragging it from the "Data" window.

If you run the report, you will see the following:

|                                 |       |
|---------------------------------|-------|
| Röd Kaviar                      | 15,00 |
| Longlife Tofu                   | 10,00 |
| Rhönbräu Klosterbier            | 7,75  |
| Lakkalikööri                    | 18,00 |
| Original Frankfurter grüne Soße | 13,00 |
| Total: 2222,71                  |       |

The same effect can be achieved by using totals. For more information, see the chapter ["Data"](#).

## Example 5. Shifting the print position

In this example, we will show how to shift the position of a band manually, by using the Engine object. We will be using:

- `BeforePrint` band event;
- Engine object.

Create a simple report of the following appearance:

|                |   |                        |                      |
|----------------|---|------------------------|----------------------|
| Data: Products | - | [Products.ProductName] | [Products.UnitPrice] |
|----------------|---|------------------------|----------------------|

Create a `BeforePrint` event handler for band:


```
private void Data1_BeforePrint(object sender, EventArgs e)
{
    Engine.CurX = ((Int32)Report.GetVariableValue("Row#")) * 10;
}
```

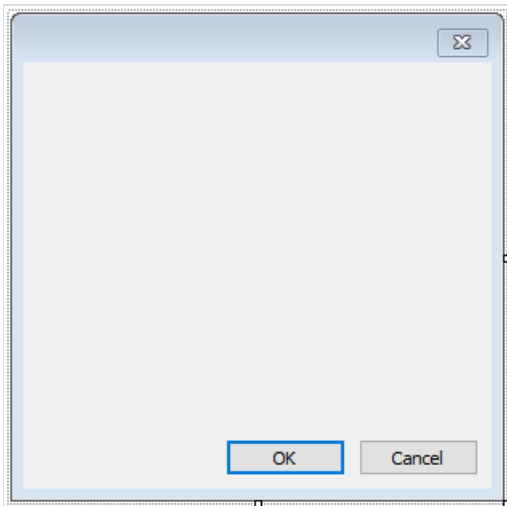
If you run the report, you will see the following:

|                              |       |
|------------------------------|-------|
| Chai                         | 18,00 |
| Chang                        | 19,00 |
| Aniseed Syrup                | 10,00 |
| Chef Anton's Cajun Seasoning | 22,00 |
| Chef Anton's Gumbo Mix       | 21,35 |

# Dialogue forms

Apart from an ordinary page, a report can contain one or several dialogue forms. Such dialogue form will be shown when the report is started. In the dialogue, you can enter any type of information, needed for creating a report. Also, a dialogue may be used to filter data, which is shown in the report.

For adding a dialogue into a report, press the  button on the designer's toolbar. A new dialogue looks as follows:























A report, having one or several dialogues, works like this:

- when started, the report shows the first dialogue;
- if the dialogue is closed with the help of the "OK" button, then the next dialogue is shown; otherwise, the report stops working;
- after all the dialogues have been shown, the report is built.

# Controls

On a dialogue from, the following controls can be used:

| Icon  | Name                  | Description   |
|---|-----------------------|---|
|    | ButtonControl         | Represents a Windows button control.  |
|    | CheckBoxControl       | Represents a Windows CheckBox.  |
|    | CheckedListBoxControl | Displays a ListBox in which a check box is displayed to the left of each item.  |
|    | ComboBoxControl       | Represents a Windows combo box control.   |
|    | DataGridViewControl   | Displays data in a customizable grid.   |
|    | DataSelectorControl   | Displays two lists and allows relocating an item from one list to the other.  |
|   | DateTimePickerControl | Represents a Windows control that allows the user to select a date and a time and to display the date and time with a specified format. |
|  | GroupBoxControl       | Represents a Windows control that displays a frame around a group of controls with an optional caption.                                 |
|  | LabelControl          | Represents a standard Windows label.  |
|  | ListBoxControl        | Represents a Windows control to display a list of items.  |
|  | ListViewControl       | Represents a Windows list view control, which displays a collection of items that can be displayed using one of four different views.   |
|  | MaskedTextBoxControl  | Uses a mask to distinguish between proper and improper user input.  |
|  | MonthCalendarControl  | Represents a Windows control that enables the user to select a date using a visual monthly calendar display.                            |
|  | NumericUpDownControl  | Represents a Windows spin box (also known as an up-down control) that displays numeric values.  |
|  | PanelControl          | Used to group collections of controls.  |
|  | PictureBoxControl     | Represents a Windows picture box control for displaying an image.   |
|  | RadioButtonControl    | Enables the user to select a single option from a group of choices when paired with other RadioButton controls.                         |

| Icon  | Name               | Description  |
|---|--------------------|--|
|  | RichTextBoxControl | Represents a Windows rich text box control.          |
|  | TextBoxControl     | Represents a Windows text box control.               |
|  | TreeViewControl    | Displays a hierarchical collection of labeled items. |

All controls, except for DataSelectorControl, are full analogs of standard controls available in the .NET Framework. Control names have the prefix Control to avoid name conflicts. Thus, the ButtonControl corresponds to the Button element in the .NET Framework.

# Referencing to a control from code

Referencing to a control can be done by using its name:

```
TextBoxControl1.Text = "my text";
```

In fact, the FastReport's control is just a wrapper for the standard .NET Framework control. It wraps many, but not all, properties of the standard control. If you need some property that is not implemented by the FastReport control, you may access a wrapped standard control in the following way:

- using the `Control` property, which is of `System.Windows.Forms.Control` type:

```
(TextBox1.Control as TextBox).ShortcutsEnabled = false;
```

- using the property which has the same name as the control itself, but without the "Control" suffix. For example, the `TextBoxControl` has got the `TextBox` property, which is of `System.Windows.Forms.TextBox` type and returns the wrapped `TextBox` control:

```
TextBox1.TextBox.ShortcutsEnabled = false;
```

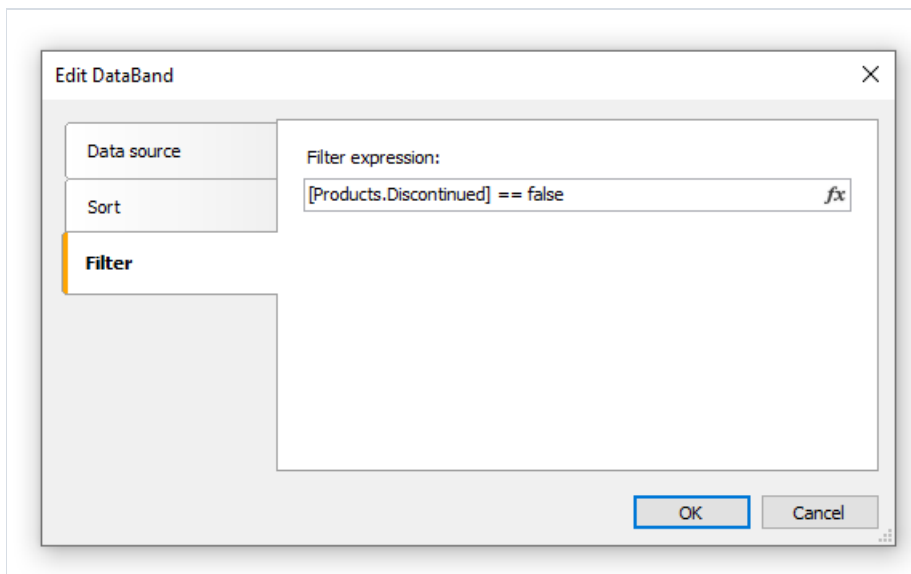
Help on properties and methods of the controls can be accessed from the MSDN.

# Data filtering

Dialogs can be used to filter the data which is printed in a report. For example, you have a report, which prints a list of all the employees. By using a dialog, you can choose one or several from it, and then when building the report, the data is filtered so that only the chosen employees can be shown.

For using the data filtering, it is necessary that the initial report contains all the data. The name "filtering" itself, assumes that, unnecessary data will not be printed when building the report.

The simplest method for organizing data filtering is to use the **Filter** property on the "Data" band. In the band editor, you can indicate the filter expression, for example:



By using the dialogue, you can ask a value from a user, and use it in the filtering expression. Look at the "Simple filter" example in the ["Examples"](#) section.

This method can be used, if a simple value is needed. If the task is to display a list of values and inquire one or several from it, implementing this becomes difficult. You may think that, it is a simple task - showing a list of employees in the `ListBoxControl` control element and choosing one or several values. For implementing this, you need to use the script, which does the following:

- get the data source by its name;
- initialize data;
- fill the `ListBoxControl` with the data from data source;
- after choosing the employee, build a filter expression that will be used in the "Data" band.












FastReport can do this automatically. For this, automatic filtering is used, which we will observe now.

# Automatic filtering - how it works

Control is connected to the data column using the `DataColumn` property. If the control can display a list of values (for example, `ListBoxControl`), it fills with values from the indicated data column. This happens automatically when the dialogue is shown. Further, the user works with the dialogue, selects one or several items in the control and closes the dialogue. At this time, the data source which was indicated in the `DataColumn` property is filtered automatically.

The advantage of this method is that, you can use it in any report without writing any code.

Automatic filtering is supported by the following controls:

| Icon  | Name                  |
|---|-----------------------|
|    | CheckBoxControl       |
|    | CheckedListBoxControl |
|    | ComboBoxControl       |
|  | DataSelectorControl   |
|  | DateTimePickerControl |
|  | ListBoxControl        |
|  | MaskedTextBoxControl  |
|  | MonthCalendarControl  |
|  | NumericUpDownControl  |
|  | RadioButtonControl    |
|  | TextBoxControl        |



# Filter operations

By default FastReport filters the data rows, which contain the value, equal to the value of the control. This behavior is defined in the `FilterOperation` property of the control. You can use the following operations:

| Operation                 | Equivalent | Effect  |
|---------------------------|------------|---|
| <b>Equal</b>              | =          | Filter the value if it is equal to the control's value.                 |
| <b>NotEqual</b>           | <>         | Filter the value if it is not equal to the control's value.             |
| <b>LessThan</b>           | <          | Filter the value if it is less than the control's value.                |
| <b>LessThanOrEqual</b>    | <=         | Filter the value if it is less than or equal to the control's value.    |
| <b>GreaterThan</b>        | >          | Filter the value if it is greater than the control's value.             |
| <b>GreaterThanOrEqual</b> | >=         | Filter the value if it is greater than or equal to the control's value. |

For example, if the `FilterOperation` property of the control is set to `LessThanOrEqual` and you enter the value `5` in the control, then all the data rows will be chosen, for which the corresponding column value is `less than or equal to 5`.

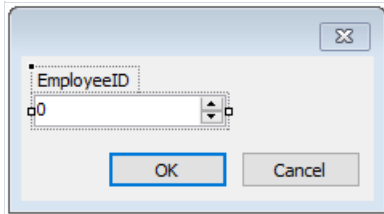
For the data of `string` type, you can use extra operations:

| Operation            | Effect  |
|----------------------|---|
| <b>Contains</b>      | Filter the value if it contains the control's value.            |
| <b>NotContains</b>   | Filter the value if it does not contain the control's value.    |
| <b>StartsWith</b>    | Filter the value if it starts with the control's value.         |
| <b>NotStartsWith</b> | Filter the value if it does not start with the control's value. |
| <b>EndsWith</b>      | Filter the value if it ends with the control's value.           |
| <b>NotEndsWith</b>   | Filter the value if it does not end with the control's value.   |

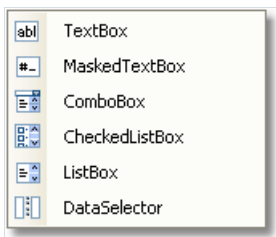
For example, if the `FilterOperation` property of the control is set to `StartsWith` and you enter the `A`, then, all data rows whose corresponding data column's value starts with `A`, will be chosen.

# Adding filter into a report

FastReport's dialogue designer has got very comfortable facilities for adding controls, which support the data filtering. For this, drag&drop the data column from the "Data" window onto the dialogue from. During this, FastReport creates the header (LabelControl control) and an actual control which will be used for data filtering:



The control type depends on the type of the data column. If the column is of string type, then after inserting it, you will be offered to choose the control type:



If you have inserted two similar controls, connected to the same data column, FastReport automatically configures the data range with the help of the `FilterOperation` property. The first control will have `FilterOperation = GreaterThanOrEqual`, the second - `LessThanOrEqual`. This will be done in case, if you insert a column which is not of string type.

As such, for adding data filtering into any report, you need to do the following:

- add a new dialogue into the report;
- drop onto the dialogue a data column, on which you want to filter the report.

# Filtering on data range

This method of filtering is comfortable for using when working with values, having a quantitative characteristic, for example, the cost. You can filter goods, having the cost less than or more than the given. In order to indicate, how to interpret the value entered in the element, use the `FilterOperation` property, looked at above.

Using two controls, which are connected to the same data column and have got different settings of the `FilterOperation` property, you can indicate the beginning and the ending of the data range. For the first control, you need to indicate the `FilterOperation = GreaterThanOrEqual` , for the second - `LessThanOrEqual` .

# Filtering on related data column

As we know, between two data sources, a relation can be set. See more details in the "[Data](#)" chapter. With the help of relation, it is possible to filter data in the source, by using a data column from a different source.

Assuming, you have placed on the dialogue a `ListBoxControl` and indicated the following data column in the `DataColumn` property:

```
Products.Categories.CategoryName
```

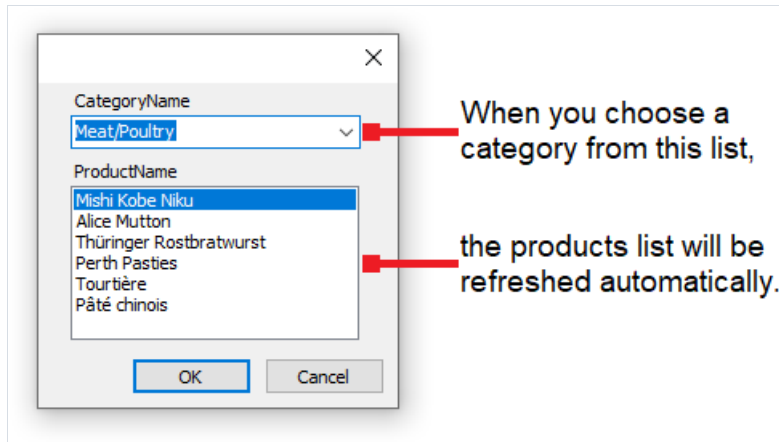
How will filtering work?

- when filling the control with values, the `CategoryName` column from the "Categories" data source will be used;
- the filter will be applied to the "Products" table. Those data rows will be filtered for which the following condition is correct:

```
the [Products.Categories.CategoryName] contains one of the values, selected by the user
```

# Filtering using cascading lists

A cascading list is a list with choices that change based on the value a user selects in another list. For example, you have two lists on a form - one with categories, another one with products. When you choose a category in the first list, you will see in the second list the products in a chosen category.



To create a cascading list, you need to use two data sources with master-detail relation between them (read more about data sources and relations in the ["Data"](#) chapter). Attach the master list to a column in the master data source; attach the detail list to a column in the detail data source. Also set the master list's `DetailControl` property to the detail list.

# Controlling the filtering from code

Even if automatic filtering is enough for many cases, you have the possibility of managing it manually. For this, the following methods and properties are used.

The `AutoFill` property controls the filling of controls with data. It is used by controls, which can show a list of values, for example, the `ListBoxControl`. Before showing a dialogue, FastReport fills such controls with data. By default, the property is set to `true`. If it is disabled, the control will not be filled, and you must do it yourself, by calling the `FillData` method:


```
ListBox1.FillData();
```

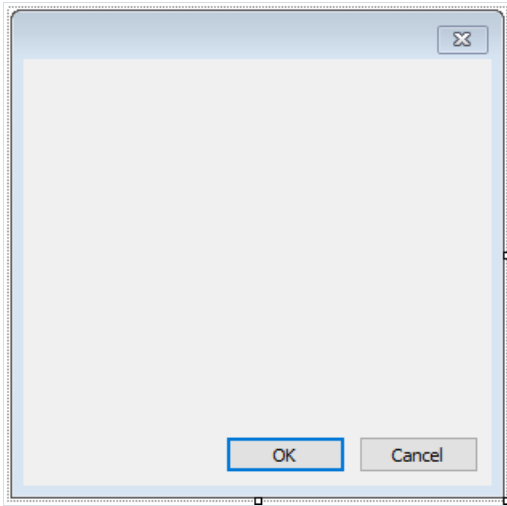
The `AutoFilter` property controls the data filtering. It is used by all controls. After the dialogue has been closed by the "OK" button, FastReport applies data filter automatically. By default, the property is set to `true`. If it is disabled, the filtering will not happen, and you must do it yourself, by calling the `FilterData` method:

```
ListBox1.FilterData();
```

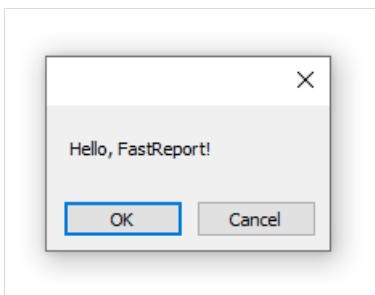
# Examples

## Example 1. "Hello, FastReport!"

In this example, all we will do - create a dialogue, which will be showing greetings. Create a new report and add a dialogue to into. For this, press the  button on the toolbar:



On the dialog, place the LabelControl and set its **Text** property in the "Properties" window:



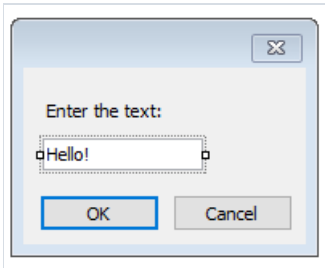
If you run the report, you will see the dialogue. Close it by the "OK" button, and the report will be built. If the dialogue is closed by the "Cancel" button or by the "X" button, the report will stop working, and you return to the designer.



## Example 2. Ask for a text from the user

In this example, we will create a dialogue, which will be asking for an arbitrary text from the user and later print the entered value in the report.

Create a new report and add a dialogue into it. On the dialogue, place LabelControl and TextBoxControl controls:



In the given case, the value we have entered is contained in the `Text` property of the `TextBoxControl`. In order to print this value in the report, add a new "Text" object on the "Report Title" band and write the following in it:

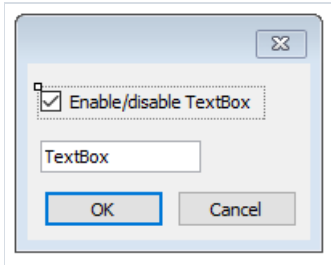
```
You have entered: [TextBox1.Text]
```


where `TextBox1` is a name of the `TextBoxControl` control.

## Example 3. Handling dialogue controls

By using the script and events of the controls, you can handle controls just like it is done in Visual Studio. We will show by example, how the CheckBoxControl can handle the TextBoxControl accessibility.

Create a new report and add a dialogue into it. On the dialogue, place the CheckBoxControl and TextBoxControl controls, as shown in the figure below:



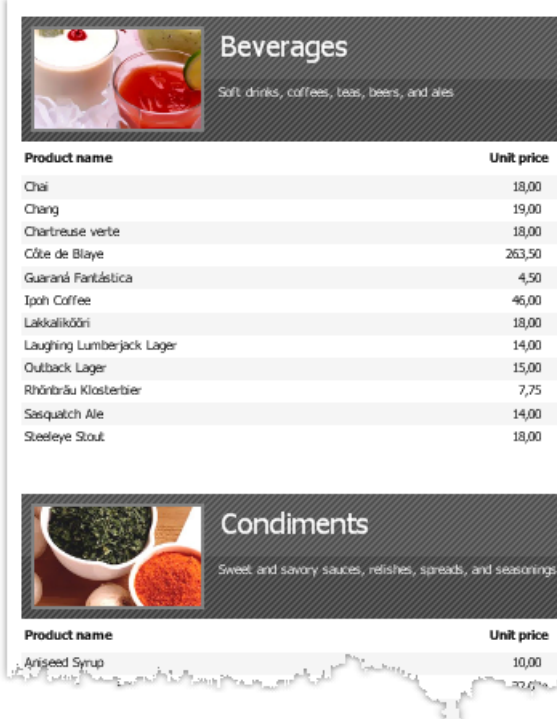
Now select the CheckBoxControl, open the "Properties" window and click the  button. Double click the `CheckedChanged` event, which is fired when changing the status of the checkbox. FastReport will create an empty handler for that event. Write the following code in it:

```
private void CheckBox1_CheckedChanged(object sender, EventArgs e)
{
    TextBox1.Enabled = CheckBox1.Checked;
}
```

If we run the report, we can enable or disable the TextBoxControl by the checkbox.

## Example 4. Handling report objects

Let us look at an example of a report, which prints a list of categories and products in each category:

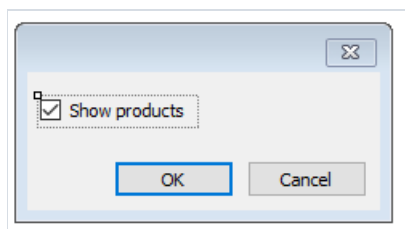


| Product name              | Unit price |
|---------------------------|------------|
| Chai                      | 18,00      |
| Chang                     | 19,00      |
| Chartreuse verte          | 18,00      |
| Côte de Blaye             | 263,50     |
| Guaraná Fantástica        | 4,50       |
| Ipioh Coffee              | 46,00      |
| Lakkalikööri              | 18,00      |
| Laughing Lumberjack Lager | 14,00      |
| Outback Lager             | 15,00      |
| Rhinbrau Klosterbier      | 7,75       |
| Sasquatch Ale             | 14,00      |
| Stoutley Stout            | 18,00      |

| Product name  | Unit price |
|---------------|------------|
| Aniseed Syrup | 10,00      |

We will show how to stop printing products and print just categories, with the help of the dialogue. For this, add a dialogue into the report:



Double click on the "OK" button. FastReport creates an empty event handler for the `Click` event. Write the following code in it:

```
private void btnOk_Click(object sender, EventArgs e)
{
    Data2.Visible = CheckBox1.Checked;
}
```

We will control the visibility of the band, which prints the product's list. In our example, this is a band with a name "Data2". If the report is run, and the checkbox is unchecked, we will have the following result:



## Beverages

Soft drinks, coffees, teas, beers, and ales



## Condiments

Sweet and savory sauces, relishes, spreads, and seasonings



## Confections

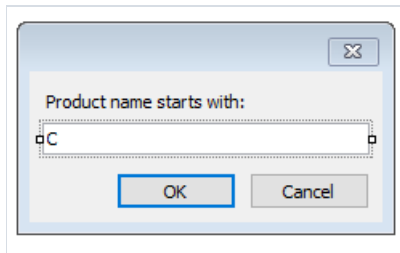
Desserts, candies, and sweet breads

## Example 5. Simple filter

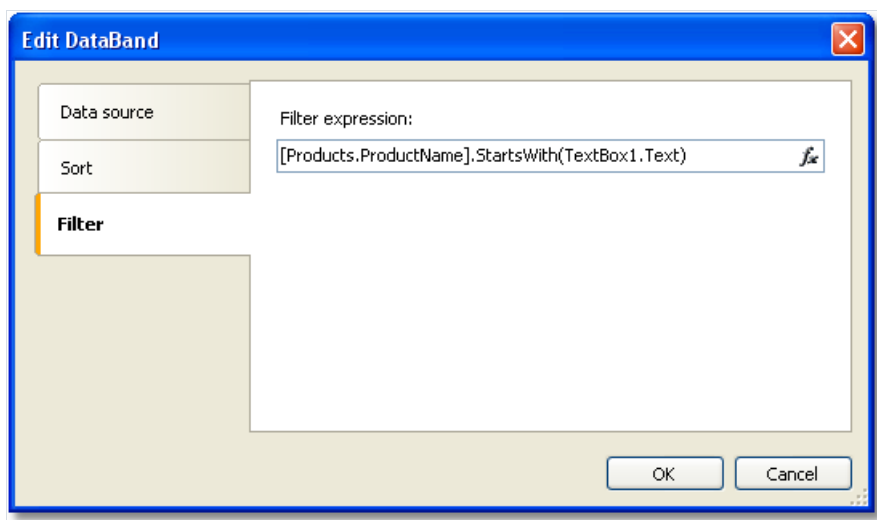
Let us look at the following report, which prints a products list:

|                |                        |
|----------------|------------------------|
| Page Header    | Product Name           |
| Data: Products | [Products.ProductName] |

We will show in this example how to filter a products list according to the first letter of the product's name. During this, we will not use automatic data filtering facilities. For this, add a new dialog into the report and place two controls onto it - LabelControl and TextBoxControl:



Now open the "Data" band editor and indicate the following filter expression:



Run the report and make sure that everything works:

×

Product name starts with:

C

OKCancel

**Product Name**


Chai  
Chang  
Chef Anton's Cajun  
Chef Anton's Gumbo Mix  
Camarvon Tigers  
Côte de Blaye  
Chartreuse verte  
Chocolade  
Camembert Pierrot

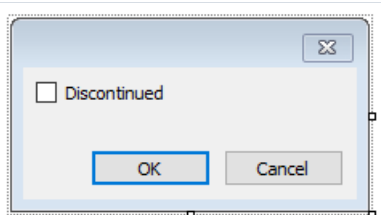
## Example 6. Automatic filtering

In this example we will show how to add a filter into the report which prints a product's list from the "Products" table. We will filter the data according to the `Products.Discontinued` column.

The report looks as follows:

| Header         | Product name           | Discontinued |
|----------------|------------------------|--------------|
| Data: Products | [Products.ProductName] | ✓            |

Add a new dialogue into the report by pressing the  button on the toolbar, and drag the `Products.Discontinued` column from the "Data" window onto the dialogue form:



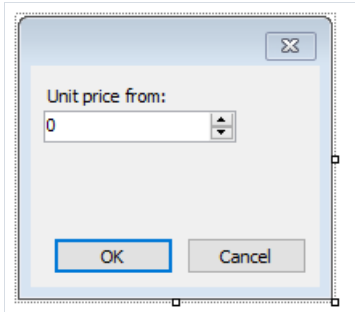
This is all we need to do - we did it just by two clicks. FastReport automatically connects the control to the data column.

Run the report and enable the `Discontinued` flag. After that, press the "OK" button, and you will see the report which contains only products with the `Discontinued` flag:

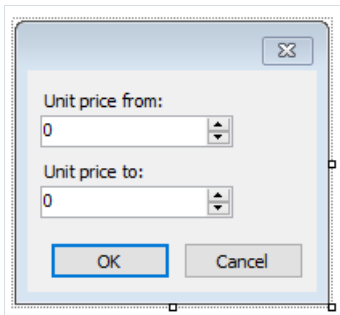
| Product name                  | Discontinued |
|-------------------------------|--------------|
| Alice Mutton                  | ✓            |
| Chef Anton's Gumbo Mix        | ✓            |
| Guaraná Fantástica            | ✓            |
| Mishi Kobe Niku               | ✓            |
| Perth Pasties                 | ✓            |
| Rössle Sauerkraut             | ✓            |
| Singaporean Hokkien Fried Mee | ✓            |
| Thüringer Rostbratwurst       | ✓            |

## Example 7. Automatic filtering by range

We will show with a report from the previous example, how to print products having the cost in the indicated range. For that we will add a dialogue into the report and drag `Products.UnitPrice` data column into it. After that we will correct the label text:



Now, in the same way, add one more `Products.UnitPrice` column and correct its header:



That is all we need to do, the rest of the work FastReport has done: connected the controls to the data column and set up their `FilterOperation` properties. The first control has got `FilterOperation = GreaterThanOrEqual`, the second - `LessThanOrEqual`.

Run the report and indicate the values, for example from 20 up to 30. When pressing the "OK" button, a report will be built. It contains products having values in the indicated range.

| Product name                     | UnitPrice |
|----------------------------------|-----------|
| Chef Anton's Cajun Seasoning     | 22,00     |
| Chef Anton's Gumbo Mix           | 21,35     |
| Fløtemysost                      | 21,50     |
| Grandma's Boysenberry Spread     | 25,00     |
| Gravad lax                       | 26,00     |
| Gustaf's Knäckebröd              | 21,00     |
| Louisiana Fiery Hot Pepper Sauce | 21,05     |
| Maxilaku                         | 20,00     |
| Nord-Ost Matjeshering            | 25,89     |
| Pâté chinois                     | 24,00     |
| Queso Cabrales                   | 21,00     |
| Sirop d'érable                   | 28,50     |
| Tofu                             | 23,25     |
| Uncle Bob's Organic Dried Pears  | 30,00     |



## Example 8. Filtering by related data column

In this example, we will use a column from a related data source to perform data filtering.

We will look at a "Simple list" report type, which prints products list. Category name is printed beside each product. This is done with the help of the relation:

```
[Products.Categories.CategoryName]
```

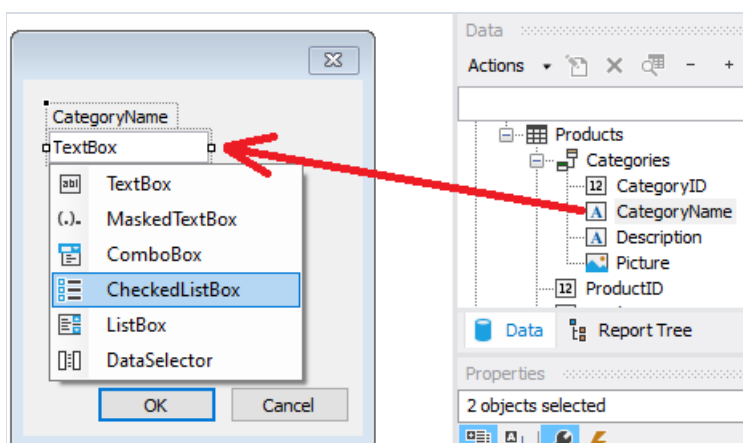
The report will be as follows:

| Report Title   | PRODUCT CATALOG        |                                    |                      |
|----------------|------------------------|------------------------------------|----------------------|
| Header         | Product name           | Category name                      | Unit price           |
| Data: Products | [Products.ProductName] | [Products.Categories.CategoryName] | [Products.UnitPrice] |

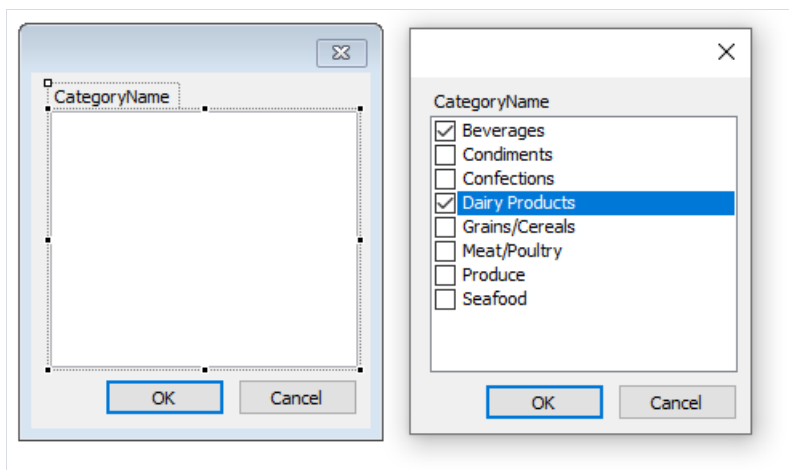
When we run the report, we will see the following:

| PRODUCT CATALOG              |                |            |
|------------------------------|----------------|------------|
| Product name                 | Category name  | Unit price |
| Alice Mutton                 | Meat/Poultry   | 39,00      |
| Aniseed Syrup                | Condiments     | 10,00      |
| Boston Crab Meat             | Seafood        | 18,40      |
| Camembert Pierrot            | Dairy Products | 34,00      |
| Carnarvon Tigers             | Seafood        | 62,50      |
| Chai                         | Beverages      | 18,00      |
| Chang                        | Beverages      | 19,00      |
| Chartreuse verte             | Beverages      | 18,00      |
| Chef Anton's Cajun Seasoning | Condiments     | 22,00      |
| Chef Anton's Gumbo Mix       | Condiments     | 21,35      |
| Chocolade                    | Confections    | 12,75      |

Let us add filtering by category name. For this, add a new dialogue and drag the `Products.Categories.CategoryName` column onto it:



When creating control, you will be required to select its type. Choose `CheckedListBoxControl`. If we run the report, we will see the following dialogue:



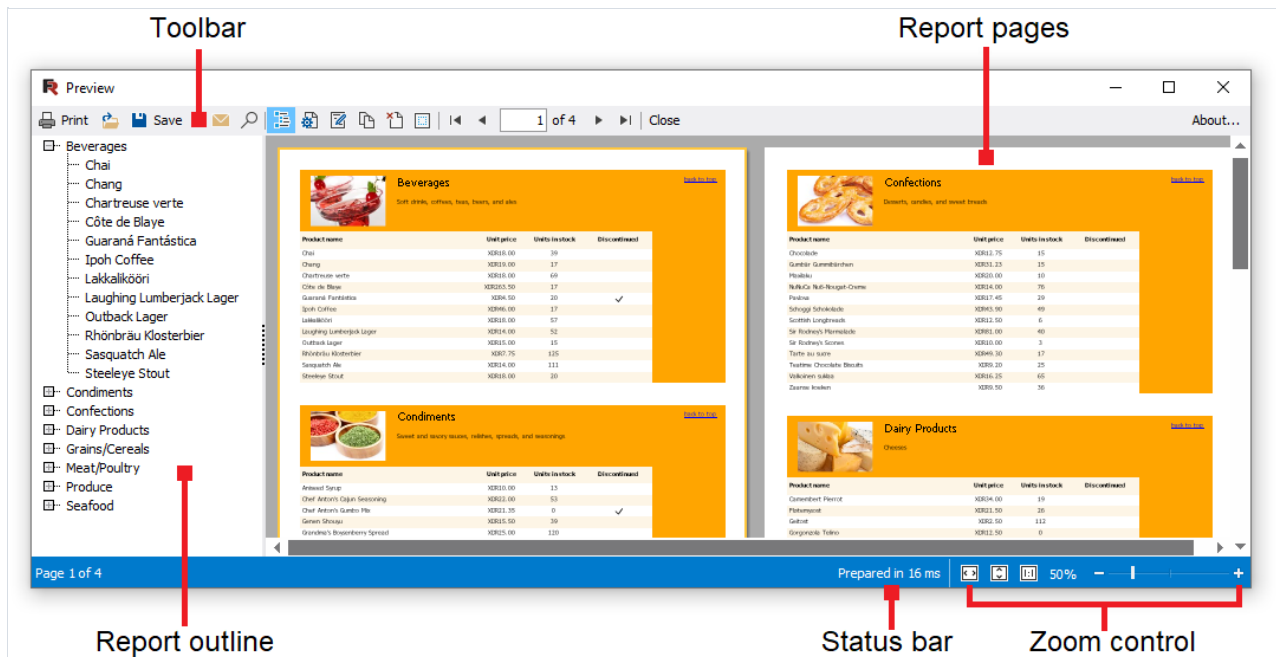
Choose several categories and click the "OK" button. After this, the data will be filtered and you will see the following report:

| PRODUCT CATALOG    |                |            |
|--------------------|----------------|------------|
| Product name       | Category name  | Unit price |
| Camembert Pierrot  | Dairy Products | 34,00      |
| Chai               | Beverages      | 18,00      |
| Chang              | Beverages      | 19,00      |
| Chartreuse verte   | Beverages      | 18,00      |
| Côte de Blaye      | Beverages      | 263,50     |
| Fløtemysost        | Dairy Products | 21,50      |
| Geitost            | Dairy Products | 2,50       |
| Gorgonzola Telino  | Dairy Products | 12,50      |
| Guaraná Fantástica | Beverages      | 4,50       |
| Gudbrandsdalsost   | Dairy Products | 36,00      |
| Ipoh Coffee        | Beverages      | 46,00      |
| Kalkkäse           | Beverages      | 18,00      |

As seen, only products have remained, which are in the chosen category.










# Preview, print, export



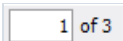


A built report can be shown on the screen, printed on the printer or exported into one of the supported formats. All these can be done in the preview window:



On the toolbar, you can find the following buttons:




| Button  | Description                                      |
|---|--|
|  | Print the report.                                |
|  | Open the prepared report file in FPX format.     |
|  | Save the report in one of the supported formats. |
|  | Send the report by email.                        |
|  | Text search in the report.                       |
|  | Shows or hides report outline.                   |
|  | Page settings.                                   |
|  | Edit current report page.                        |
|  | Watermark settings.                              |

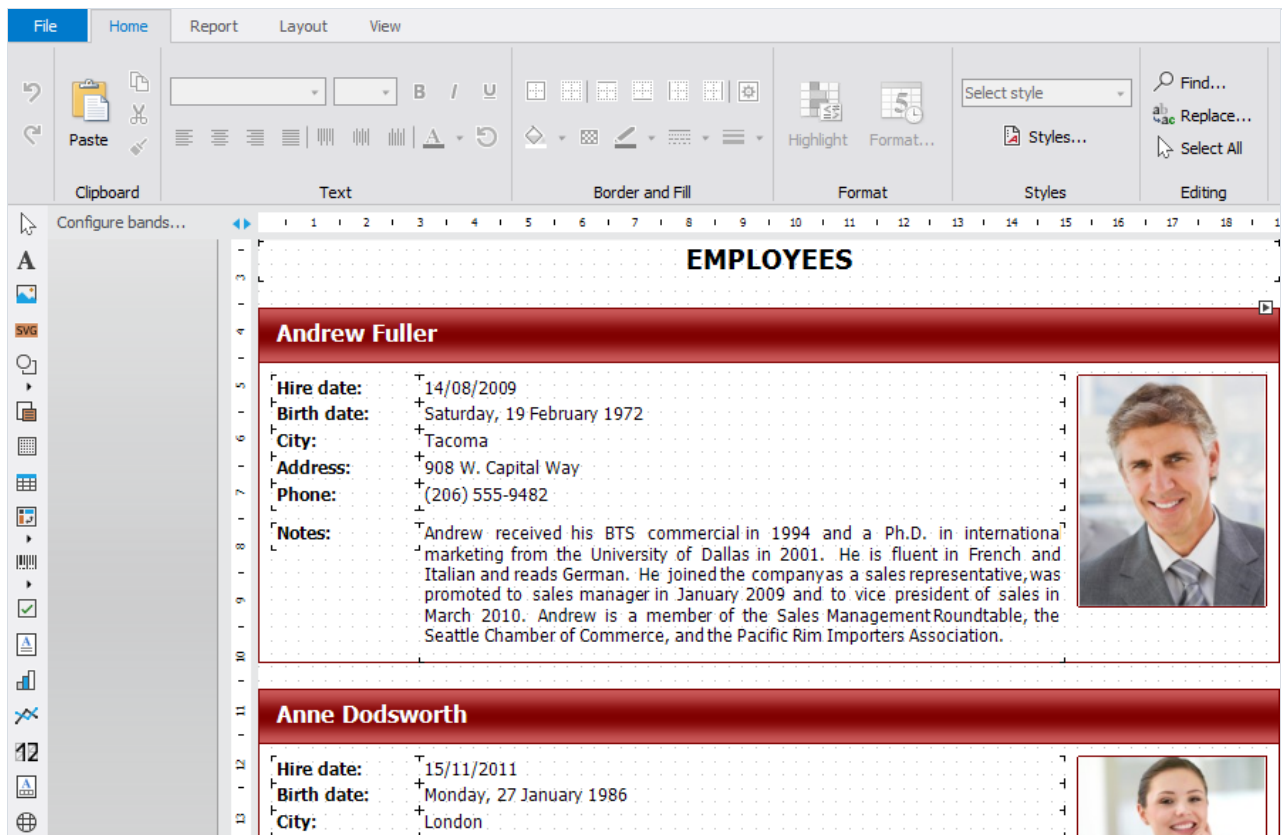
| Button  | Description  |
|---|--|
|  | Navigate to the first page.  |
|  | Navigate to the previous page.   |
|  | Navigate to the indicated page. Enter the page number and press Enter. |
|  | Navigate to the next page.   |
|  | Navigate to the last page.   |

You can use the following keyboard control:

| Key                     | Description                 |
|-------------------------|-----------------------------|
| <b>Ctrl+P</b>           | Print the report.           |
| <b>Ctrl+F</b>           | Text search.                |
| <b>Arrows</b>           | Scroll the preview.         |
| <b>PageUp, PageDown</b> | Page up/down.               |
| <b>Home</b>             | Navigate to the first page. |
| <b>End</b>              | Navigate to the last page.  |
| <b>Esc</b>              | Close the preview window.   |


# Editing the report

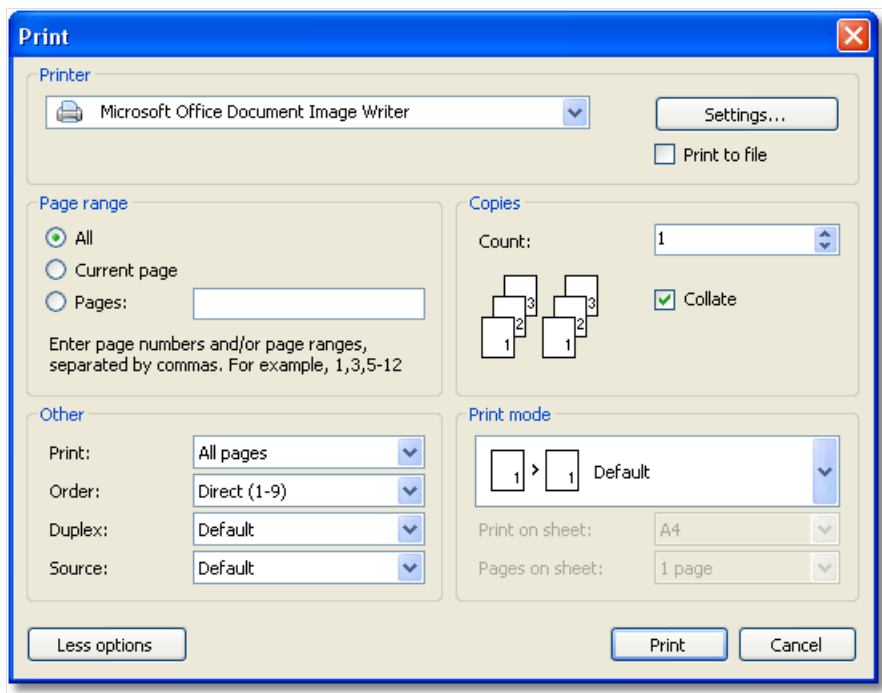
For editing a page of a prepared report, click the  button in the preview window. During this, the current page will be loaded in the report designer, where you can do whatever you want with it:



After editing, close the designer. When doing this, you will be asked to save the changes in the report page.

# Printing the report

In order for print a report, press the  button (or press a combination of Ctrl+P). You will see the print dialogue:



We will look at the settings accessible in this dialogue:

- the button "More/Less" allows showing the whole dialogue or just the basic settings. By default, a dialogue is shown in a simple form;
- "Printer" group: here, you can choose the printer, change its settings ("Settings..." button) and choose print to the file;
- "Page" group: here you can choose, which pages to print (all, current or the given page number);
- "Copy" group: here you can set the number of copies and choose the order of the pages in the copies ( **Collate** );
- "Other" group: here you can choose, which pages to print (all, even, odd), choose the order of printing (direct, reverse), set up the duplex printing (if your printer supports it) and choose the paper source;
- "Print mode" group allows choosing one of the printing modes:

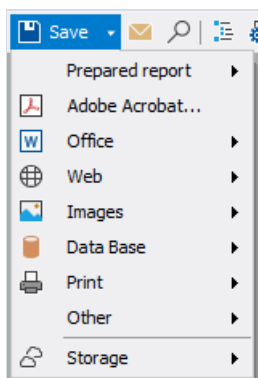
| Mode            | Description  |
|-----------------|--|
| Default         | The printer prints on a paper, indicated in the report. One report page corresponds with one printed sheet.  |
| Split big pages | Use this mode, if you need to print A3 report on a A4 format paper. One report page will produce two printed sheets. When using this mode, you have to choose the paper format from the "Print on sheet" list.   |
| Scale           | Use this mode, if you need to print A4 report having on a A3 format paper. On one printing sheet, you can print 1, 2, 4 or 8 report pages. When using this mode, you need to choose the format of the paper on which you want to print, from the "Printing on sheet" list, and also indicate the number of pages in the "Pages on sheet" list. |

After pressing the "Print" button, printing of the report will start. If the "Print to file" flag is chosen, then the name of the file will be requested and the report will be saved in that file (file with a PRN extension).

# Exporting the report

FastReport allows exporting the generated report to various formats for further editing, archiving, emailing, and other purposes.

For choosing export, press the "Save" button in the preview window and choose the export:



At the moment, export is supported in the following formats:

- PDF;
- Office:
  - Excel 2007;
  - Word 2007;
  - PowerPoint 2007;
  - OpenOffice Calc / OpenOffice Writer;
  - RTF;
  - Excel XML (Excel 2003+).
- Web:
  - HTML;
  - MHT.
- Images:
  - Bmp, Png, Jpeg, Gif, Tiff, Metafile;
  - SVG.
- Data Base:
  - CSV;
  - DBF;
  - Json.
- Print:
  - TXT;
  - ZPL;
  - PPML;
  - PostScript;
  - XPS.
- Other:
  - LaTeX;
  - DXF;
  - XAML.

Additionally, it is possible to save the report in cloud services such as Dropbox, Box, Google Drive, OneDrive, Simple Storage Service, as well as on an FTP server.



# Saving in FPX format

FPX format is FastReport's "native" format. The advantages of this format are as follows:

- saving the report without losing the quality. Opening an already saved file, you can do all operations with it, like print, export, editing;
- compact format based on XML, compressed with the help of ZIP;
- when needed, the report file can be unpacked by any archiver that supports the ZIP-format and corrected manually in any text editor.

The only thing lacking with the format is that, to view it, you need to have FastReport .NET.

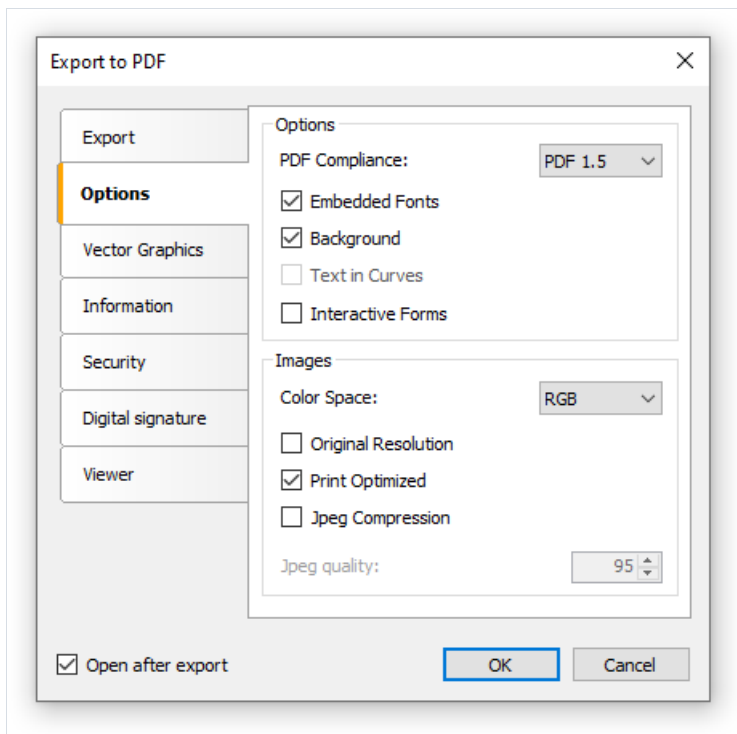
In order to save in the FPX format, press the "Save" button in the preview window and choose the "Prepared report" file type. For opening already saved files, press the "Open" button.

# Export to Adobe Acrobat (PDF)

PDF (Portable Document Format) is a platform-independent format of electronic documents created by Adobe Systems. The free Acrobat Reader package is used for viewing. This format is rather flexible – it allows the inclusion of necessary fonts, vector and bitmap images; it allows transferring and storage of documents intended for viewing and further printing.

Export method is a layered one.

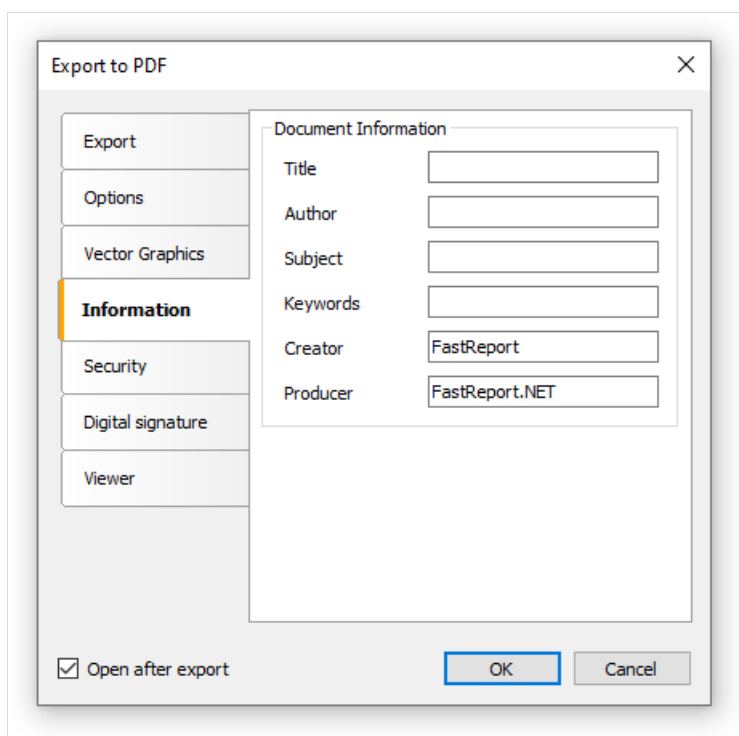
When exporting to Excel, there will be a dialogue window for setting parameters of the output file:



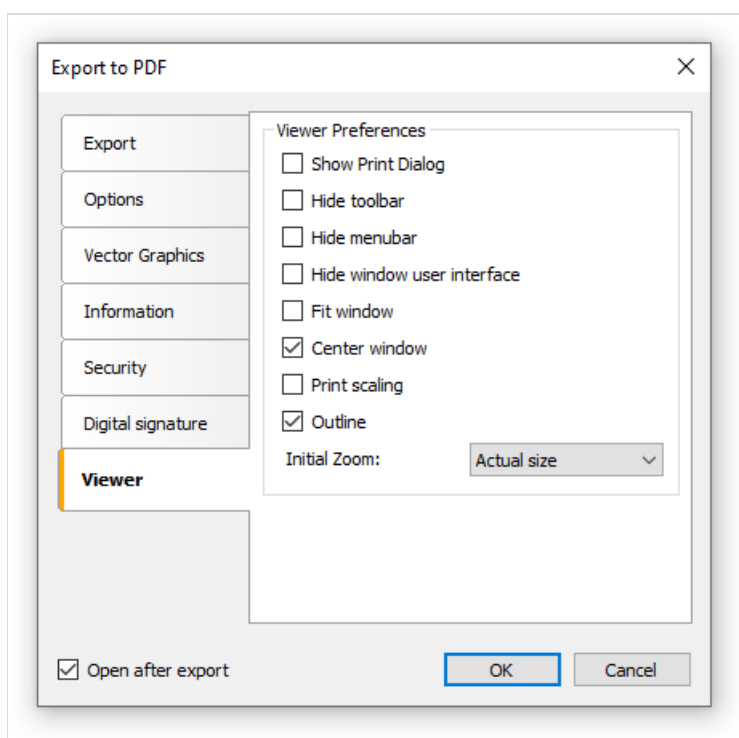
Export parameters:

- "Compressed" - output file is compressed. It reduces file size but increases export time;
- "Embedded fonts" - all fonts used in a report will be included into PDF file. This will significantly increase the file size;
- "Background" - the page watermark will be exported as an image. This will significantly increase the file size;
- "Print optimized" - output of all graphics objects (such as pictures, charts) in high resolution for further printing.

On the "Information" tab, you may fill in the document information fields:



On the "Viewer" tab, you may set up some options related to Adobe Acrobat document viewer:

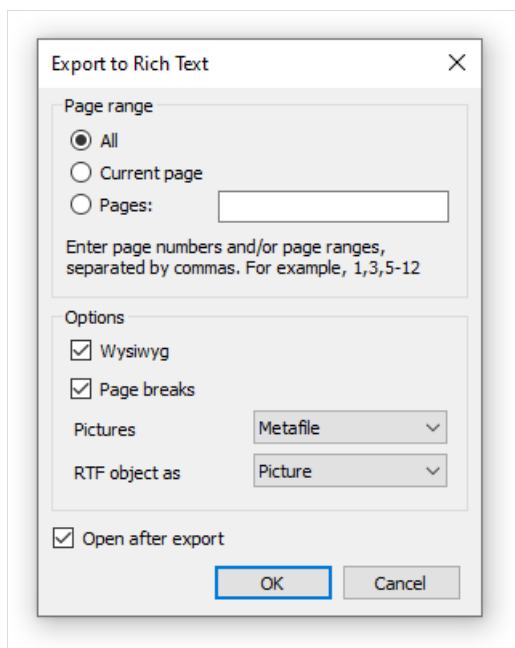


# Export to RTF

RTF (Rich Text Format) was developed by Microsoft as a standard format for exchanging text information. At the moment RTF-documents are compatible with many new text editors and operation systems.

Export method: tabular.

When exporting to RTF, there will be a dialogue window for setting parameters of the output file:



Export parameters:

- "Wysiwyg" - the result will be as close to the report as possible. If this option is disabled, FastReport will reduce the number of rows and columns in the resulting;
- "Page breaks" - enables page breaks in the RTF file;
- "Pictures" - select the format of pictures in the RTF file. Note that "Metafile" format is best for displaying of such report objects as MSChartObject and ShapeObject.

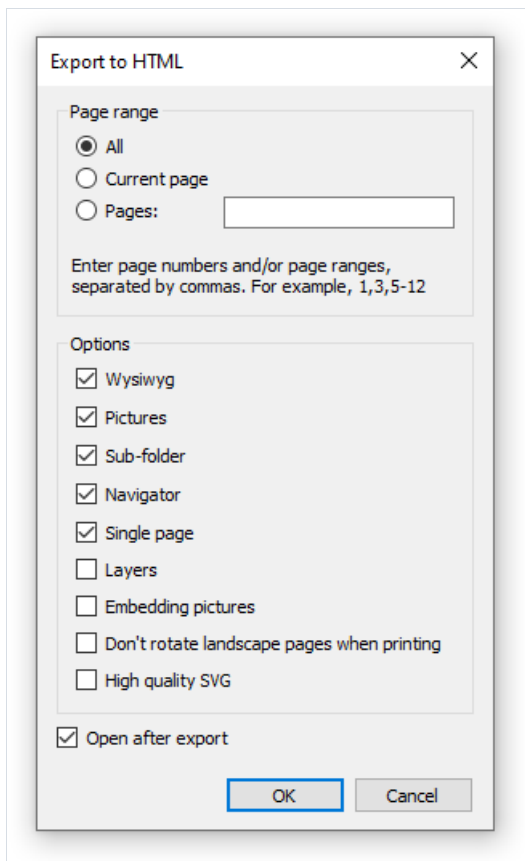
Appearance and size of the resulting file depends on the report template (see the ["Recommendations on report development"](#) section).

# Export to HTML

HTML (HyperText Markup Language) is the predominant markup language for Web pages. It provides a means of describing the structure of text-based information in a document - by denoting certain text as links, headings, paragraphs, lists, and so on - and to supplement that text with interactive forms, embedded images, and other objects.

Export method: tabular.

When exporting into HTML, a dialogue window will be offered for setting the parameters:



Export parameters:

- "Wysiwyg" - the export result will be as close to the report as possible;
- "Pictures" - enables to export pictures;
- "Sub-folder" - all extra files are saved in a separate folder called ".files";
- "Navigator" - creates a special navigator for navigating on pages;
- "Single page" - all pages will be saved in one file.

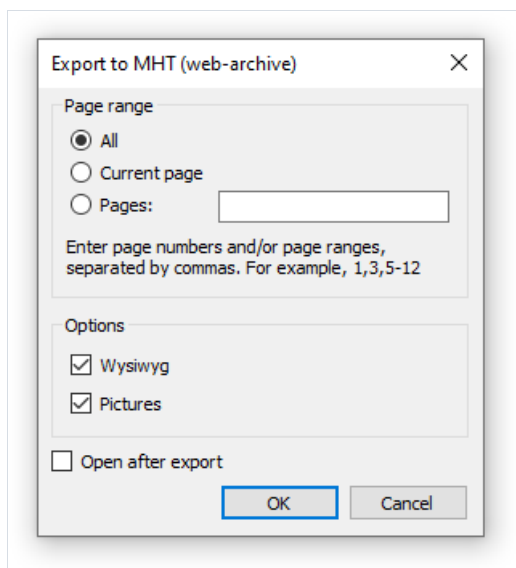
Appearance and size of the resulting file depends on the report template (see the ["Recommendations on report development"](#) section).

# Export to MHT (web archive)

MHT, short for MIME HTML, is a web page archive format used to bind resources which are typically represented by external links (such as images, Flash animations, Java applets, audio files) together with HTML code into a single file. The content of an MHT file is encoded as if it were an HTML e-mail message, using the MIME type multipart/related.

Export method: tabular.

When exporting into MHT, a dialogue window will be offered for setting the parameters:



Export parameters:

- "Wysiwyg" - the export result will be as close to the report as possible;
- "Pictures" - enables to export pictures.

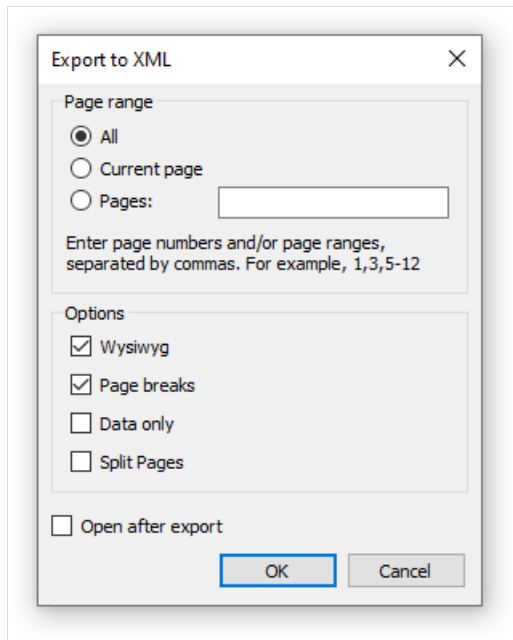
Appearance and size of the resulting file depends on the report template (see the ["Recommendations on report development"](#) section).

# Export to Excel (XML)

Excel is an application for working with electronic worksheets. It is included into Microsoft Office.

Export method: tabular.

When exporting to Excel, there will be a dialogue window for setting parameters of the output file:



Export parameters:

- "Wysiwyg" - the result will be as close to the report as possible. If this option is disabled, FastReport will reduce the number of rows and columns in the resulting file;
- "Page breaks" - enables page breaks in the resulting file.

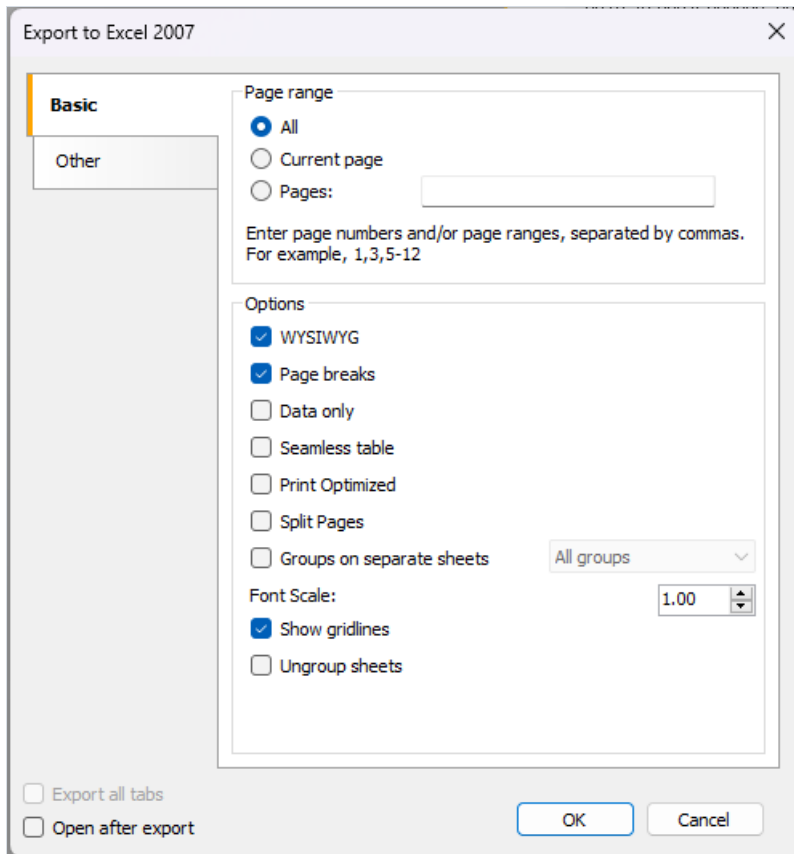
Appearance and size of the resulting file depends on the report template (see the ["Recommendations on report development"](#) section).

# Export to Excel 2007

Excel 2007 is an application for working with electronic worksheets. It is included into Microsoft Office 2007.

Export method: tabular.

When exporting to Excel, there will be a dialogue window for setting parameters of the output file:

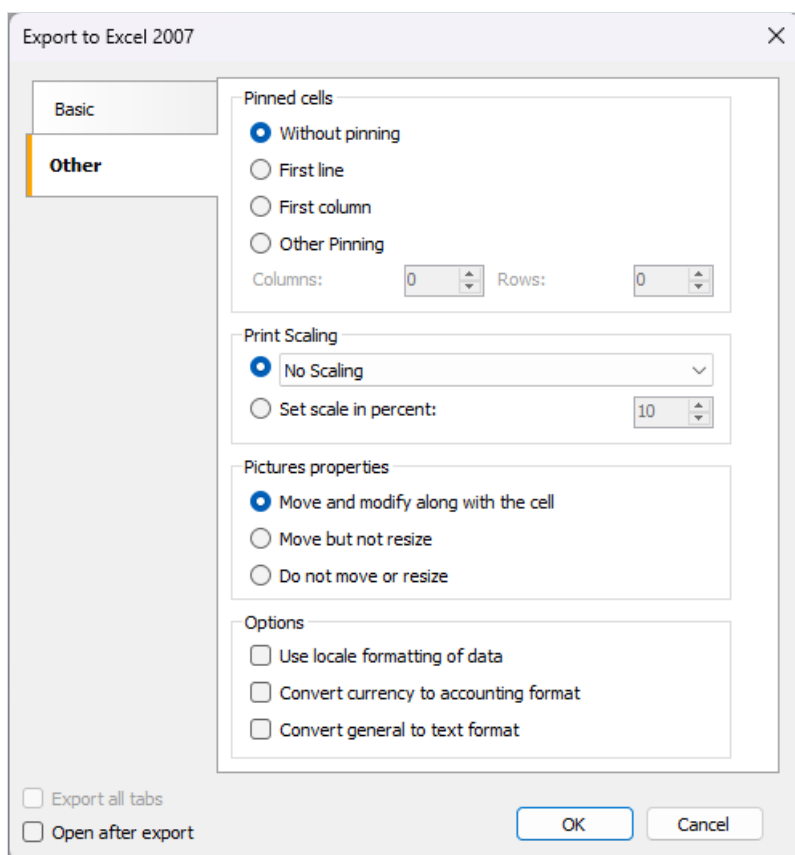


Export parameters:

- Wysiwyg – the result will be as close to the report as possible. If this option is disabled, FastReport will reduce the number of rows and columns in the resulting file;
- Page breaks – enables page breaks in the resulting file;
- Split pages – each page of the prepared report is exported to a separate sheet.
- Group on separate sheets – allows placing data from each group on a separate sheet. If there is only one group in the document or if there are no groups, the dropdown list will only have one option available – "All groups";
- Ungroup sheets – sheets grouped together are automatically synchronized, so any changes made to one of them will be applied to all selected sheets. By default, all sheets are grouped;
- Export all tabs – exports all tabs for interactive reports where detailed reports can be opened in new tabs;
- Open after export – the resulting file will be opened in Excel immediately after export.

The sections on the "Other" tab provide tools for customizing the appearance and behavior of data when exporting to Excel. Each section provides options to precisely configure the format and display of data in Excel:





"Pinned cells" section allows you to select the method of freezing specific areas of the Excel sheet during export. This can be useful for maintaining the visibility of certain rows or columns when scrolling or zooming the table in Excel.

"Print Scaling" section allows you to configure the way data is printed, determining how it will be distributed across pages when printing. This includes choosing the scale and placement of data on the page for optimal display.

"Pictures properties" section provides the ability to customize the behavior of images when placed in Excel cells. The selection of available options determines whether the images will move and resize along with the cells when resizing cells or moving data within the table.

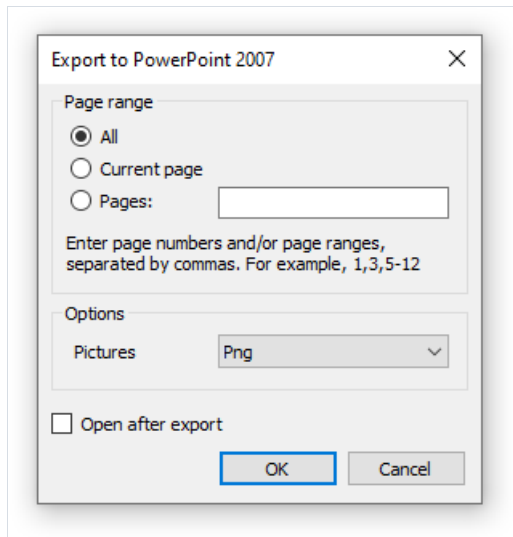
Appearance and size of the resulting file depends on the report template (see the ["Recommendations on report development"](#) section).

# Export to PowerPoint 2007

PowerPoint 2007 is an application for working with electronic presentations. It is included into Microsoft Office 2007.

Export method is a layered one.

When exporting to PowerPoint, there will be a dialogue window for setting parameters of the output file:



Export parameters:

- "Pictures" - select the format of pictures in the resulting file;
- "Open after export" – the resulting file will be opened in PowerPoint immediately after export.

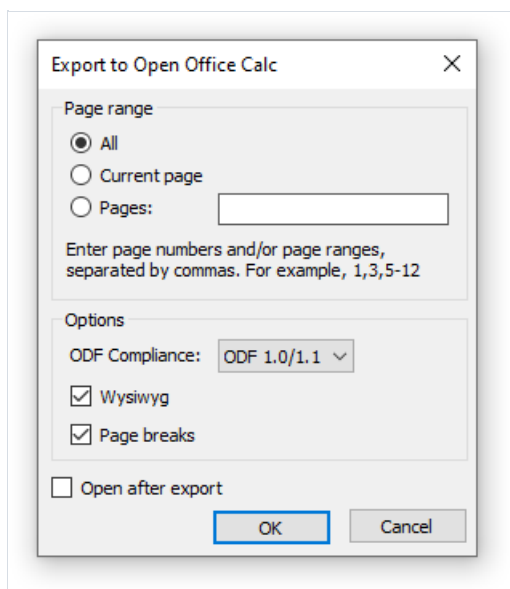
# Export to OpenOffice Calc

OpenDocument Format (ODF, OASIS Open Document Format for Office Application) was designed by OASIS and based on XML format used in OpenOffice.

FastReport supports export to table (.ods file). These files can be opened in OpenOffice.

Export method: tabular.

When exporting to OpenOffice Calc, there will be a dialogue window for setting parameters of the output file:



Export parameters:

- "Wysiwyg" - the result will be as close to the report as possible. If this option is disabled, FastReport will reduce the number of rows and columns in the resulting file;
- "Page breaks" - enables page breaks in the resulting file.

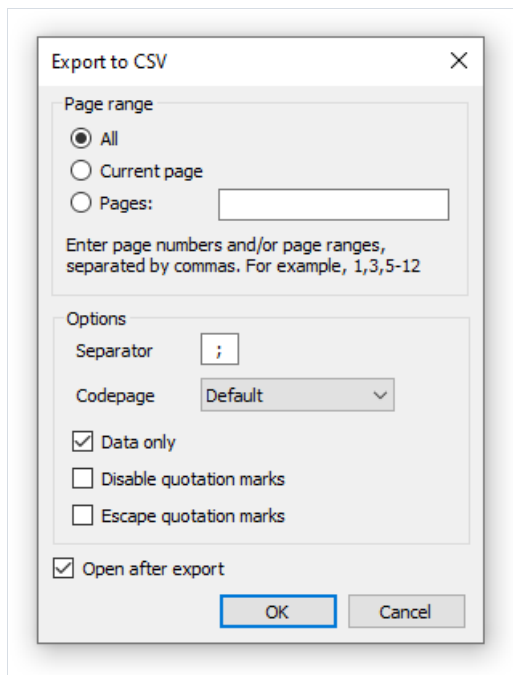
Appearance and size of the resulting file depends on the report template (see the ["Recommendations on report development"](#) section).

# Export to CSV

The CSV file is used for the digital storage of data structured in a table of lists form. Each line in the CSV file corresponds to a row in the table. Within a line, fields are separated by commas, each field belonging to one table column. CSV files are often used for moving tabular data between two different computer programs, for example between a database program and a spreadsheet program.

Export method: tabular.

When exporting to CSV, there will be a dialogue window for setting parameters of the output file:



Export parameters:

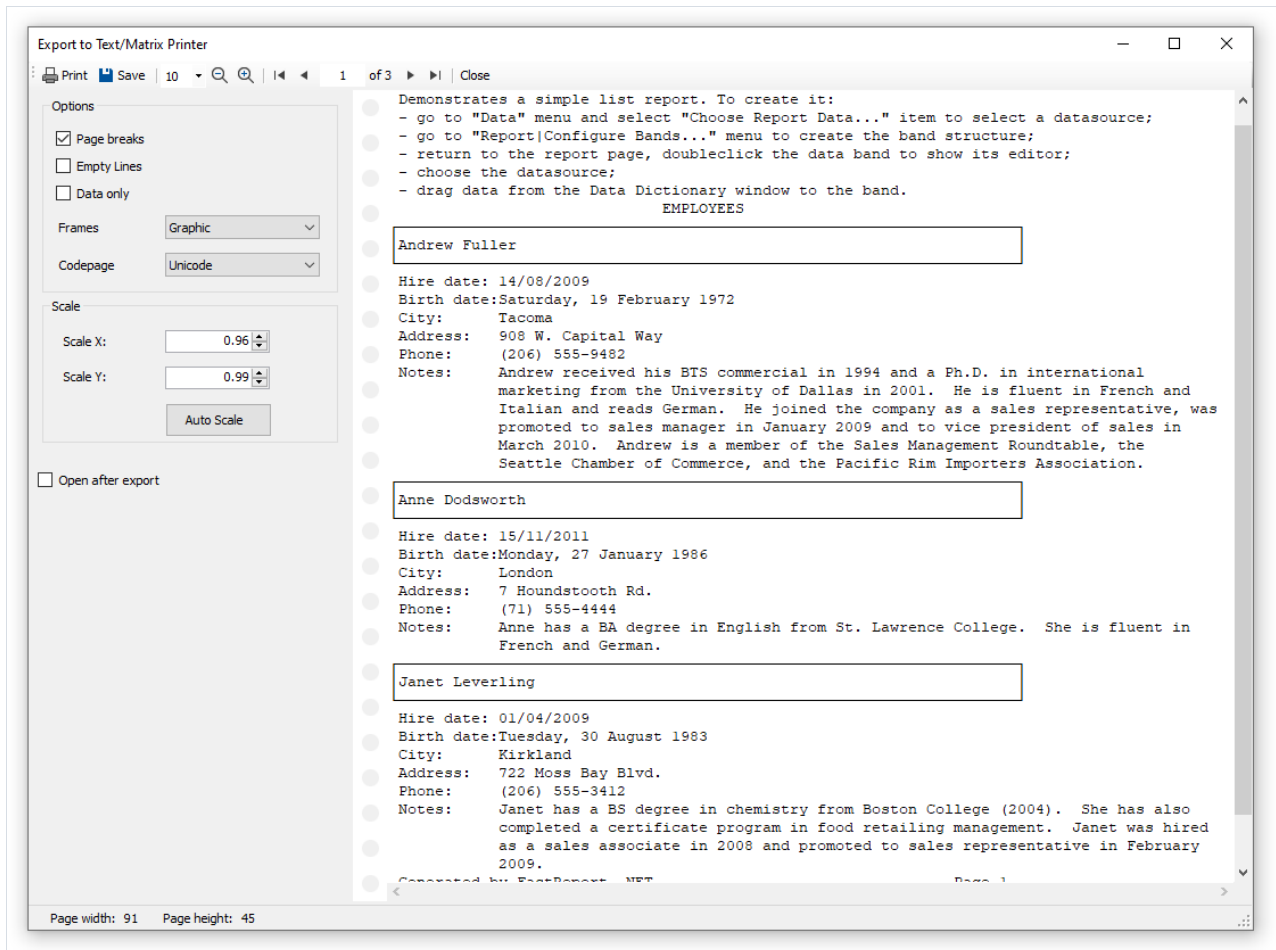
- "Separator" - the field separator character;
- "Codepage" - codepage used to encode the text in resulting file. The "Default" codepage refers to Windows default codepage. Note that Excel does not support unicode codepages;
- "Data only" - enable this checkbox to export objects laying on Data band only.

# Export to TXT

TXT is a regular text file that can be opened in any text editor, or printed to a dot-matrix printer.

Export method: tabular.

When exporting to TXT, there will be a dialogue window for setting parameters of the output file:



Export parameters:

- "Page breaks" - enables page breaks in the resulting file;
- "Empty Lines" - enables empty lines in the resulting file;
- "Data only" - enable this checkbox to export objects laying on Data band only;
- "Frames" - type of object's borders. Select "None" if you don't want to export borders;
- "Codepage" - codepage used to encode the text in resulting file;
- "Scale X" - horizontal scale;
- "Scale Y" - vertical scale;
- "Auto Scale" - calculates scale X and scale Y automatically to avoid data loss.

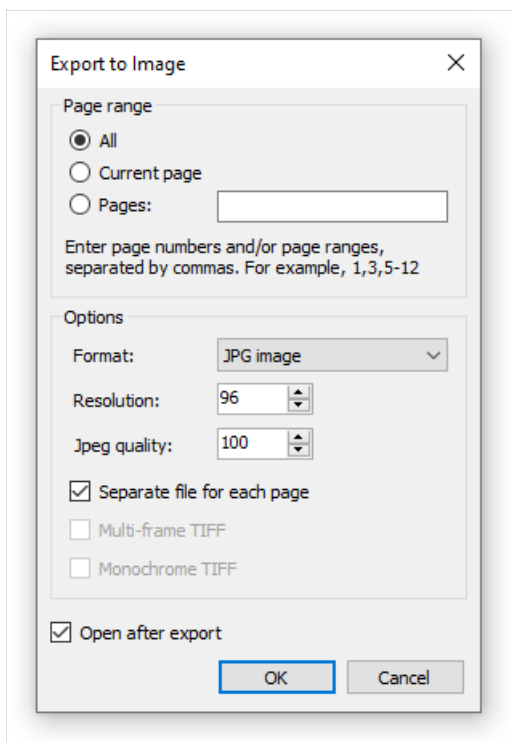
# Export to picture

FastReport allows exporting information into the following graphical formats:

- BMP;
- PNG;
- JPG;
- GIF;
- TIFF;
- Windows Metafile (EMF,WMF).

Exporting method: drawing.

When exporting into picture files, a dialogue window will be offered for setting the parameters:



Export parameters:

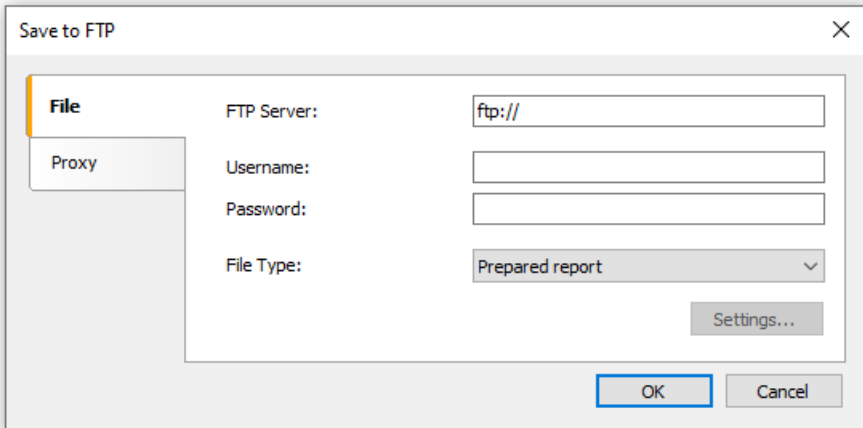
- "Resolution" - resolution of the graphical image. Use 96dpi for displaying, 300dpi for printing. When exporting into the TIFF format, you will be able to set separate values for horizontal and vertical resolution;
- "Jpeg quality" - JPG file compression level. This option is used only when exporting into the Jpeg format;
- "Separate file for each page" - if the option is enabled, then each report page will be exported into a separate file, the name of the file will be formed on the basis of the chosen page with the given number;
- "Multi-frame TIFF" - this option produces the multi-frame TIFF file. It is used only when exporting into the TIFF format;
- "Monochrome TIFF" - this option produces the monochrome TIFF file. It is used only when exporting into the TIFF format.

When exporting several pages into one file (when the "Separate file for each page" option is disabled), the export will use a lot of CPU/Memory resources.

# Export to FTP

A prepared report can be saved to an FTP server from the FastReport.NET preview. The report can be exported to one of the supported formats before being saved to FTP.

Press "Save" button and select "FTP...". This opens the "Save to FTP" window:



The screenshot shows the "Save to FTP" dialog box with the "File" tab selected. The dialog has a title bar with a close button (X). On the left, there are two tabs: "File" (selected) and "Proxy". The main area contains the following fields:

- FTP Server:** A text input field containing "ftp://".
- Username:** An empty text input field.
- Password:** An empty text input field.
- File Type:** A dropdown menu showing "Prepared report".
- Settings...** A button located below the File Type dropdown.

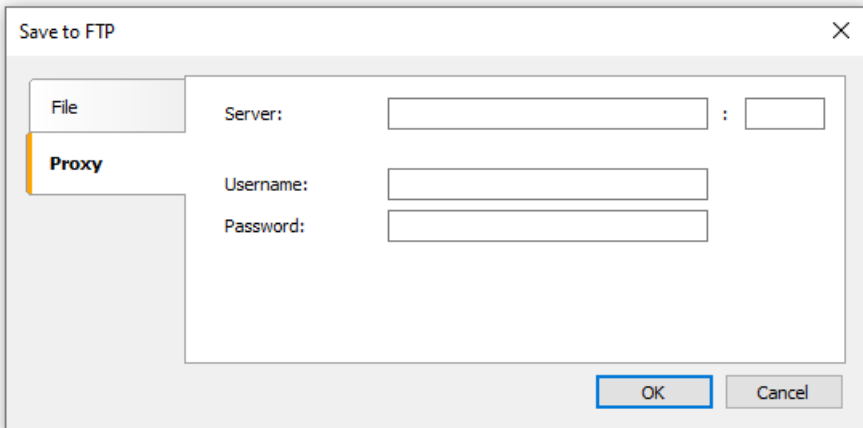
At the bottom right, there are two buttons: "OK" and "Cancel".

The File tab has the following fields:

- FTP Server: enter the URL-address of the FTP server
- Username and Password: enter your username and password
- File Type: select the file format (prepared report or one of the export formats) from the drop-down list.

If an export format is selected then the "Settings..." button becomes available, which opens the settings window for the chosen export format.

If a proxy server is used then the URL-address, port, username and password of the proxy server are set on the Proxy tab:



The screenshot shows the "Save to FTP" dialog box with the "Proxy" tab selected. The dialog has a title bar with a close button (X). On the left, there are two tabs: "File" and "Proxy" (selected). The main area contains the following fields:

- Server:** A text input field followed by a colon and a port input field.
- Username:** An empty text input field.
- Password:** An empty text input field.

At the bottom right, there are two buttons: "OK" and "Cancel".

When all settings have been made click the OK button to save the file to the FTP server.

# Export to Dropbox

A prepared report can be saved to a Dropbox from the FastReport.NET preview. The report can be exported to one of the supported formats before being saved to the Dropbox.

Before saving a report to a Dropbox an application must be created in your Dropbox account. Do this by logging in to your Dropbox account and taking the following steps:

- click the "More" button: it is located at the bottom of the Dropbox homepage
- choose "Developers" in the drop-down list: you will be directed to the page for developers
- go to "App Console": it directs you to the list of applications
- click the "Create App" button: the Dropbox will check your email: click Send Email. In your mail system inbox you will find an email having a "Confirm" button: click the button to confirm your email address.

Finally you will be directed to "Create a new Dropbox Platform app".

Select "Dropbox API app" and answer the question "What type of data does your app need to store on Dropbox?" by selecting "Files and datastores".

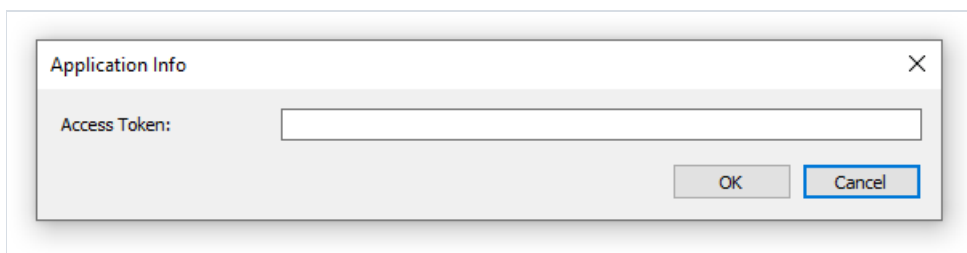
Answer the question "Can your app be limited to its own, private folder?" by choosing either of the two proposed answers.

This page also requires you to enter the name of the application.

After clicking the "Create app" button the system will create the application.

As a result the application settings page is opened. Here you can see the "App key" and the "App secret", which are required when exporting to a Dropbox.

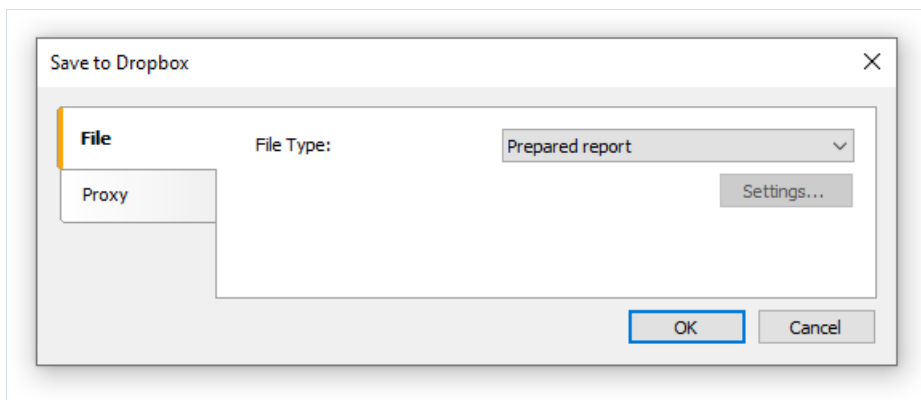
You can now go to the FastReport.NET preview and export the file to the Dropbox by pressing the "Save" button and selecting "Dropbox...". When exporting to a Dropbox for the first time the "Application Info" window will be displayed:



Enter the "Application Key" (App key) and "Application Secret" (App secret) obtained above. After clicking the "OK" button FastReport.NET saves these values and uses them again the next time.

The "Save to Dropbox" window has two tabs: File and Proxy:



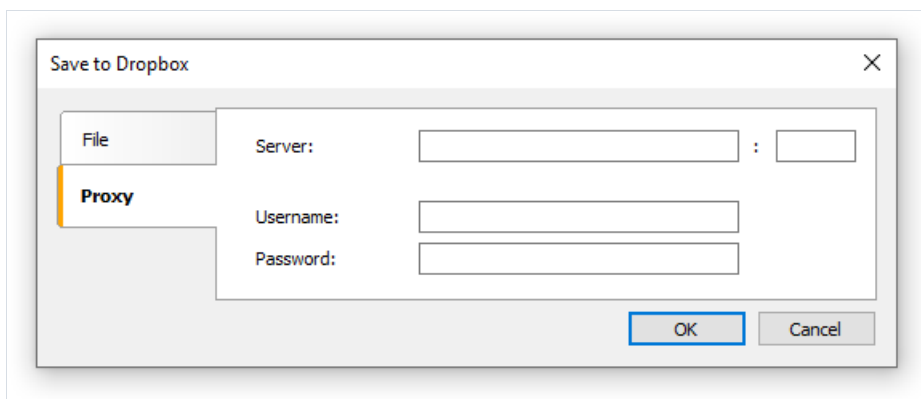


The File tab contains the following fields:

- File Type : select the file format (prepared report or one of the export formats) from the drop-down list.

If an export format is selected then the "Settings..." button becomes available, which opens the settings window for the chosen export format.

If a proxy server is used then the URL-address, port, username and password of the proxy server are set on the Proxy tab:



When all settings have been made click the "OK" button to save the file to the Dropbox.

# Export to Google Drive

A prepared report can be saved to Google Drive from the FastReport.NET preview. The report can be exported to one of the supported formats before being saved to Google Drive.

Before saving a report to Google Drive an application must be created in your Google Drive account.

Do this by going to <https://code.google.com/apis/console/>

- this page has the license agreement and leads to the Project Settings page.

Go to the "Services" tab and activate the "Drive API".

Go to the "API Access" tab and click "Create an OAuth 2.0 client ID".

In the "Branding Information" section enter the name of the application and click "Next".

In "Client ID Settings", select the following:

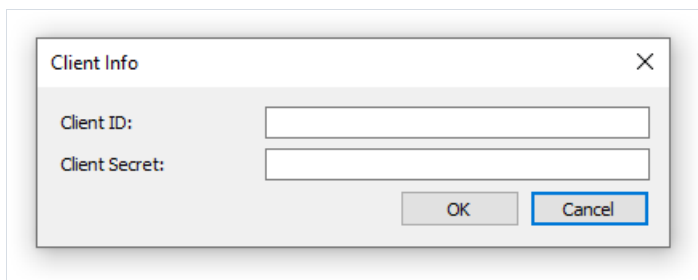
- "Installed application" for Application type
- "Other" for Installed application type.

Click "Create Client ID".

As a result the next page shows the "Client ID" and "Client secret".

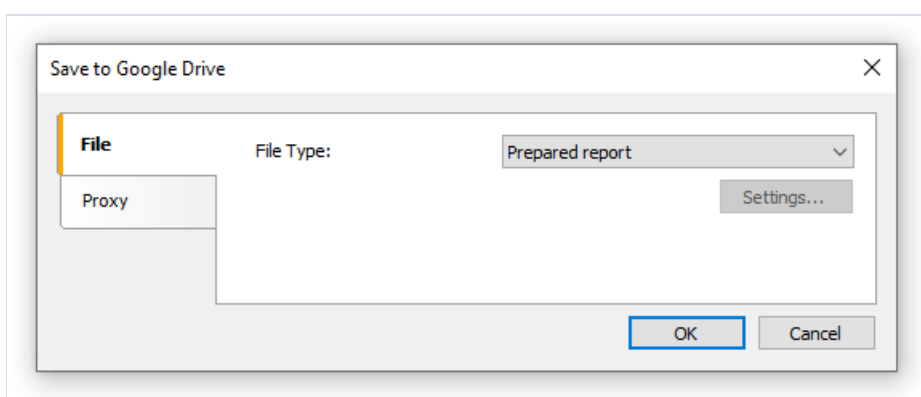
You can now go to the FastReport .NET preview and export the file to Google Drive by pressing the "Save" button and selecting "GoogleDrive..."

When exporting to a Dropbox for the first time the "Client Information" window will be displayed:



Enter the "Client ID" and "Client Secret" obtained above. After clicking the "OK" button FastReport.NET saves these values and uses them again the next time.

The "Save to Google Drive" window has two tabs : File and Proxy:

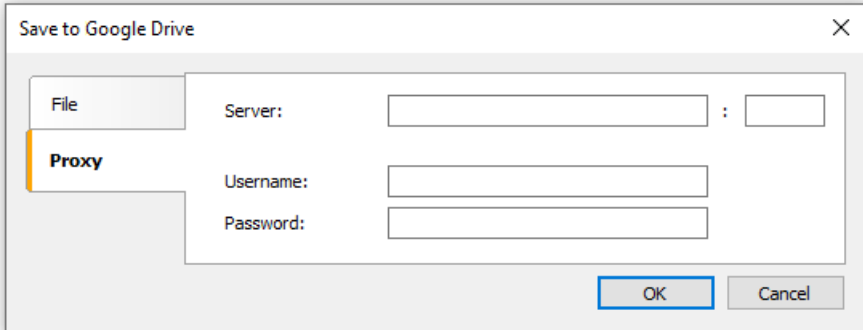


The File tab contains the following fields:

- File Type : select the file format (prepared report or one of the export formats) from the drop-down list.

If an export format is selected then the "Settings..." button becomes available, which opens the settings window for the chosen export format.

If a proxy server is used then the URL-address, port, username and password of the proxy server are set on the Proxy tab:



The screenshot shows a dialog box titled "Save to Google Drive" with a close button (X) in the top right corner. On the left side, there are two tabs: "File" and "Proxy". The "Proxy" tab is currently selected, indicated by a yellow highlight. The main area of the dialog contains three input fields: "Server:" followed by a text box and a port box separated by a colon, "Username:" followed by a text box, and "Password:" followed by a text box. At the bottom right of the dialog, there are two buttons: "OK" and "Cancel".

When all settings have been made click the "OK" button to save the file to Google Drive.

# Export to OneDrive

A prepared report can be saved to OneDrive from the FastReport.NET preview. The report can be exported to one of the supported formats before being saved to OneDrive.

Before saving a report to OneDrive an application must be created in your OneDrive account. Do this by going to the OneDrive home page and clicking "Developers".

In the Development Center click "My Apps" and then click "Create Application".

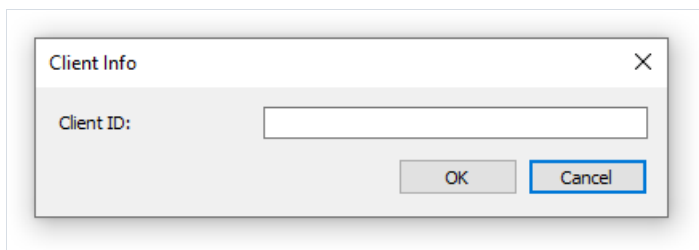
Enter the name of the application and select a language. Read the "Terms of Use" and "Privacy Statement" and click "I accept".

As a result the next page is the Settings page of the application which shows the "Client ID" and "Client Secret".

Enter the redirection domain and click "Save".

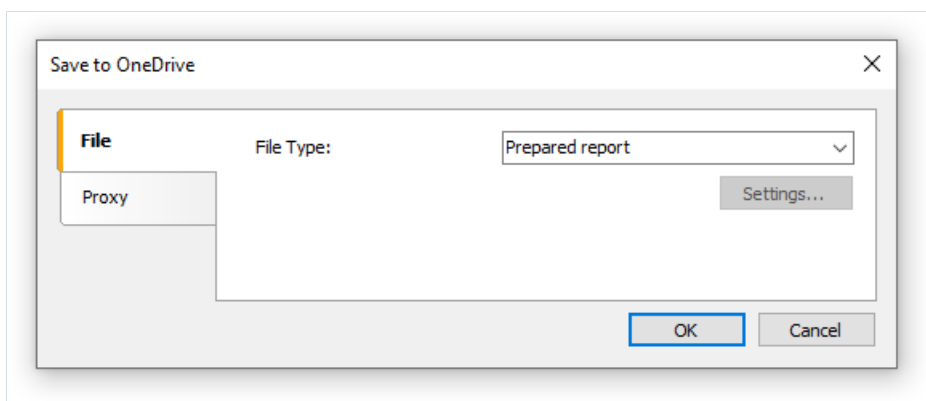
You can now go to the FastReport.NET preview and export the file to OneDrive by pressing the "Save" button and selecting "OneDrive...".

When exporting to OneDrive for the first time the "Client Information" window will be displayed:



Enter the "Client ID" and "Client Secret" obtained above. After clicking the "OK" button FastReport.NET saves these values and uses them again the next time.

The "Save to OneDrive" window has two tabs : File and Proxy:

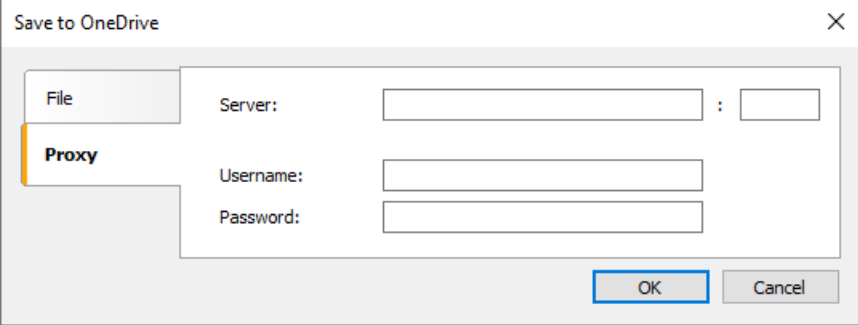


The File tab contains the following fields:

- File Type : select the file format (prepared report or one of the export formats) from the drop-down list.

If an export format is selected then the "Settings..." button becomes available, which opens the settings window for the chosen export format.

If a proxy server is used then the URL-address, port, username and password of the proxy server are set on the Proxy tab:



The image shows a 'Save to OneDrive' dialog box. It has a title bar with a close button (X). On the left, there are two tabs: 'File' and 'Proxy'. The 'Proxy' tab is selected, indicated by an orange vertical bar. The main area contains three input fields: 'Server:' followed by a text box and a port box (separated by a colon), 'Username:' followed by a text box, and 'Password:' followed by a text box. At the bottom right, there are two buttons: 'OK' and 'Cancel'. The 'OK' button is highlighted with a blue border.

When all settings have been made click the "OK" button to save the file to OneDrive.

# Report design recommendations

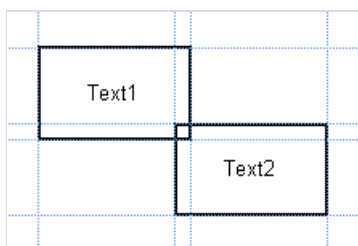
In this chapter, special design requirements of reports intended for export to other data formats will be discussed.

FastReport allows a great number of ways to manipulate objects during report creation. This gives the advantage of fast development of any reports and their further printing. Printed document will look just as on display. And this is the primary intent of FastReport report generator usage.

The downside of such development freedom is the complexity of exporting the FastReport document to different data formats, which have their own limits and requirements for information presentation, and are sometimes rather complex. Many formats, such as HTML, XLS or RTF, use table data presentation. These formats do not allow cell crossing or arranging in layers when table marking.

Export filters, as a rule, take into account these requirements. This is carried out by special algorithm which takes object crossings into account and places them optimally. At object crossing, there are new columns and rows in the resulting output table appear. That is necessary for getting maximum resemblance between the result and original report. A large number of crossed objects in report design, leads to an increased number of columns and rows in the resulting table, that affects the file size and its complexity.

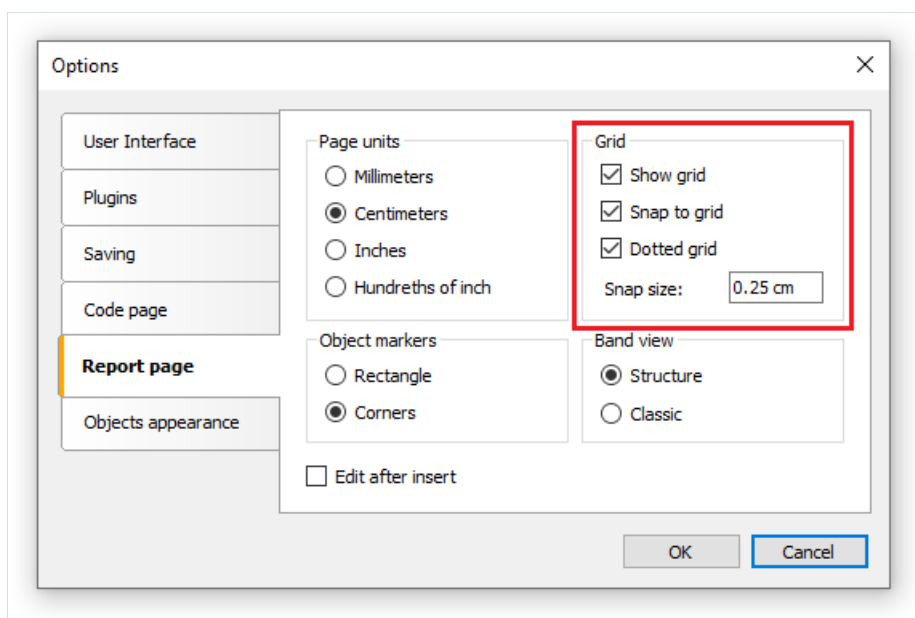
The quality of the export depends greatly on competent design of initial report. Let us look at the following example:



There is a slight crossing of two objects placed one under another on the same band. The number of records on report forming was 150. On export to RTF format 450 rows will be created (150 rows for each object and 150 ones for crossing). If we remove crossing, there will be only 300 rows in the resulting table. In large reports and on huge number of objects the difference will be really tremendous. That, of course, will affect output file size.

When creating tables in report, pay attention to neighboring cell's borders. It is important that cells do not cross and arrange in layers. Export filter algorithm will cut off cells but export result may be far from desirable (you will see not exactly what you wanted to). Arrange objects in such a way that they are placed in line vertically as well as horizontally. Guidelines can help to perform this.

The grid alignment can also be helpful in case of cells overlapping. Enable grid alignment in designer options. In order to simplify alignment you can extend grid pitch. Setting of grid pitch and alignment can be found in the "View|Options..." menu:



For text framing it is better to use text object's border instead of single graphic objects such as lines, rectangles, etc. Try not to use background objects under transparent text objects.

Applying these simple rules will help you to create a report which will look perfect after being exported to any table-based format.

# Sending the report by email

FastReport allows you to send a prepared report by email. It may work in two modes:

| Mode | Description   |
|------|---|
| SMTP | This is default mode. To send an email, you don't need any external programs.   |
| MAPI | You may turn on this mode programmatically. To do this, set <code>Config.EmailSettings.UseMAPI = true</code> , or, if you use EnvironmentSettings component, set its <code>EnvironmentSettings.EmailSettings.UseMAPI</code> property to true. To send an email, FastReport uses the default email client such as Outlook Express. This client must support the MAPI protocol. |

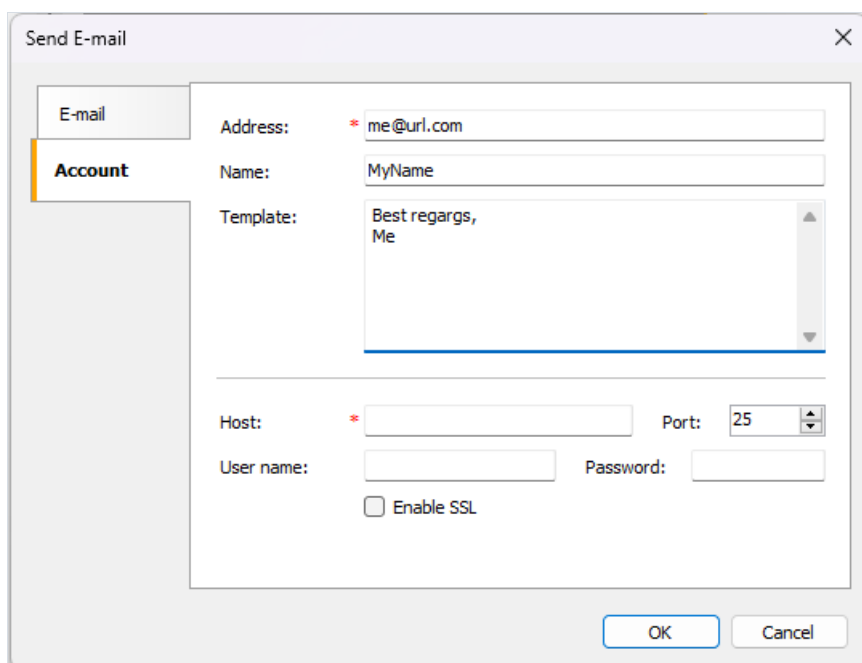
To send an email, you need to specify a recipient's email address. Also you need to specify the subject and email body, but this is not required. At the bottom of the dialog, select the format of your report - it will be attached to the message:

The screenshot shows a 'Send E-mail' dialog box. It has two tabs: 'E-mail' and 'Account'. The 'E-mail' tab is selected. The fields are as follows:

- Address:** \* john.smith@url.com (with a red asterisk indicating it's mandatory)
- CC:** (empty)
- Subject:** Test
- Message:** This is the test message.
- Attachment:** Prepared report (dropdown menu)
- Settings...** (button)
- Name attachment file:** Simple List
- Buttons:** OK, Cancel

If you use the SMTP mode, you need to set up an account. It is necessary to do only once. Once you have done it, FastReport will save the parameters in the configuration file. The parameters can be found on the "Account" tab. All obligatory fields are marked by red asterisk:





The image shows a 'Send E-mail' dialog box with a close button (X) in the top right corner. On the left, there is a sidebar with two tabs: 'E-mail' and 'Account'. The 'Account' tab is selected and highlighted with an orange bar. The main area of the dialog is divided into two sections. The top section contains three fields: 'Address:' with a red asterisk and the value 'me@url.com', 'Name:' with the value 'MyName', and 'Template:' with a text area containing 'Best regards, Me'. The bottom section contains four fields: 'Host:' with a red asterisk and an empty text box, 'Port:' with a value of '25' and a spinner control, 'User name:' with an empty text box, and 'Password:' with an empty text box. Below these fields is a checkbox labeled 'Enable SSL'. At the bottom right of the dialog are two buttons: 'OK' and 'Cancel'.

Send E-mail

E-mail

Account

Address: \* me@url.com

Name: MyName

Template: Best regards, Me

Host: \* Port: 25

User name: Password:

☐ Enable SSL

OK Cancel

If your host server requires an authentication, you need to fill in "User name" and "Password" fields as well.

# Minimum system requirements

Minimum system requirements for installing and using FastReport .NET:

- Operating system MS Windows 7-11, Windows Server 2012-2019;
- CPU: 1 GHz;
- RAM: 512 MB;
- You also need the installed [.NET Framework version 4.6.2 or higher](#).

# Contacts and technical support

You can always ask questions about using the product by [email](#), or by using [the form on the website](#).

We also welcome your suggestions on how to improve our product.