Guide 1 - INTRODUCTION

In this introduction we present REPCoder - a unicode reporting library for programmers and database developers. Here you will find general information about the product. The main guides that teach how to use REPCoder are: "Graphics" and "Reports" (with examples). For programmers additionally - "DLL".

IMPORTANT:

The current version of REPCoder is using 4 independent mechanisms for communication with databases: BDE (Borland Database Engine), ODBC (Open Database Connectivity), Firebird client direct, Interbase client direct.

Find the "Next page" item in the "Form" menu and go to the next page.

1. What is "REPCoder" ?

REPCoder *is an advanced software tool product created to design and execute* ***database reports.*** *On the design level we call them* ***forms.*** *After they are executed they are* ***reports.*** *This is a* ***UNICODE*** *database application, that displays UNICODE and UTF8 characters from databases. The product's destination is to fully manage the* ***reporting process.*** *But its capabilities are much wider. It can also write data to databases. For this it is especially useful when you want to transfer data between different databases. The program can efficiently read and write BLOBs (binary large objects). It fully supports NUMERIC data type based on 64-bit integers. One of the most important features is the built-in script language (similar to C), with more than 100 predefined utility functions. It can be used to write algorithms that make your reports do what you exactly need. The program is not complicated. It is very intuitive, easy to learn and use. But it requires good practical knowledge of SQL language to design advanced reports. REPCoder is a WYSIWYG tool (what you see is what you get). It can be very easily used (if you are a programmer) in your database application, because it is a single DLL (32-bit: 1.6 MB, 64-bit: 2 MB) that does not need any additional libraries.*

It allows you to create very advanced presentations of data (reports). Its hidden power and reach capabilities will surely satisfy even the most demanding end user and database programmer. It is completely different from other reporting tools on the market. It was especially designed to be used by database programmers. Hence its name is ***REPCoder.***

2. Is there any new method that differs this program from other reporting tools ?

Yes. A well known idea of physicists of old times, that ***"the nature is affraid of the vacuum"*** *became the main idea of the design process. Here it is new, original and not used before. It translates to the main rule, that* ***"the form pages are affraid of holes".*** *This rule is the main architectural concept of the tool. Together with other mechanisms and solutions applied, they are thought to obtain the maximum compromise between the two difficult to combine properties of a good software:* ***reach capabilities*** *and* ***user friendly.***

3. Who can be the user of the product ?

You can work with ***Repcoder.exe (Repcoder64.exe)*** *as a seperate application. It can be used by any database user and report developer. You can also work with* ***Repcoder.dll (Repcoder64.dll)*** *if you are a programmer and want to call reports from your own applications. For that you will also need the header file "Repcoder.h" and the import library "Repcoder_ms.lib", "Repcoder64_ms.lib" or "Repcoder_bor.lib". You can alternatively use "Repcoder.h" with "Repcoder.c" in your project. This is the recommended method, because it does not need compiler-dependent import libraries. If you are a C# programmer, just use only "Repcoder.cs" file. There is also programming interface for Delphi and Java (Windows only). The files are "Repcoder.pas" and "Repcoder.java" respectively. The DLL functions can call the reports with different options, parameters and user privileges. Thus we have 3 groups of users:*

- designer	*Creates report forms (.SFM file) - graphics and SQL queries.*
- programmer	*Uses functions of "Repcoder.dll" to call the reports from his own application. This way he enables (all or limited) capabilities of the tool to end users of his application.*
- end user	*Executes the reports and uses its results. He can also make changes to report forms, becoming also a designer this way.*

4. What special knowlege and skills should the users have ?

The end user who only executes reports and uses its results, does not need to have any specialized knowledge. The report designer instead, must be a specialist in database problems. He has to know SQL very well, because REPCoder is SQL-based. There is a possibility to produce almost any kind of reports, no matter how complicated. Moreover the programmer must additionally know how to call DLL functions. He will also usually be the report designer and database programmer in one person. The author of the product belongs to this group. Because of that, it was created and designed especially for the ***use of programmers.*** *Using REPCODER.DLL API is very simple and does not require any additional knowledge or learning. This is just in principle only one C-language function. This way you can have completely solved the problem of reporting in your application. It will be never needed to support a report by additional code in your application. You only need to pass a set of text* ***parameters*** *from your EXE to the* ***repc_open_report*** *function of REPCODER.DLL.*

*The underlying idea is to maximally **seperate the report project from the application code.** Everything must be enclosed in the report project. Thus any changes in the report **will not require** changes in the application code.*

5. In what programming language "REPCoder" was written ?

*The program was written as a classical Windows application in **C/C++** language. It uses only pure WIN32/WIN64 API, together witch C runtime library (linked statically). The author is a theoretical physicist and a database programmer. It was compiled by the compiler: **Microsoft Visual C++ Express 2008.** In the current version, the communication with databases is provided by 4 independent mechanisms: **BDE, ODBC, Firebird, Interbase.** The product also uses the excelent compression library **ZLIB** (Jean-loup Gailly, Mark Adler), which is statically linked into REPCoder (you don't need ZLIB.DLL). To display graphics the program uses Windows GDI32.DLL and GDIPLUS.DLL. You can display pictures and blobs of various types: BMP, ICO, JPG, TIF, GIF, PNG.*

6. What files are shipped with the product ?

The working files of REPCoder are only:

- Repcoder.exe, Repcoder64.exe *(EXE version of the product)* — *- works as an independent application*

*The programmers will also need the following files (**DevTools** subdirectory):*

- Repcoder.dll, Repcoder64.dll *(DLL version of the product)* — *- to be used in database applications*

Repcoder.h, Repcoder.c,
Repcoder.cs, Repcoder.java, Repcoder.pas — *- for database applications developers*

Repcoder_ms.lib, Repcoder64_ms.lib, Repcoder_bor.lib
(Microsoft and Borland-specific import libraries) — *- for database applications developers*

7. What files are created by the program ?

*The project files of the designed reports have the **SFM** extension. The saved results of executed reports (visual archives) have the **SFR** extension. The program also creates at the first time, its configuration file: **Repcoder.cfg.** The configuration can be changed using the "Options" item in the "File" menu.*

Thus the files created by the program are:

Repcoder.cfg *(configuration file)*

***.SFM** *(project files of the reports)*

***.SFR** *(the results of executed reports - "visual archives")*

8. What is in the configuration file (REPCODER.CFG) ?

*There are graphics and display defaults (described in the guides) and 2 additional parameters that influence the program interaction with **databases:***

*- **BDE Path.** The program is using BDE by calling the functions of "IDAPI32.DLL" library. Its location is usually found in the registry. However, when the standard entry was removed or the path was changed, you can enter here the valid path of "IDAPI32.DLL".*

*- **FBCLIENT.DLL (GDS32.DLL / IBCLIENT64.DLL) is Thread-Safe.** By default, while working directly with Firebird/Interbase databases the program uses Windows critical sections to synchronize different threads, assuming that Firebird/Interbase client DLL **is not thread-safe.** This however slows down the performance. Therefore if you are using newer, thread-safe client libraries, just check this option. This will increase the performance, especially if you run many reports simultanously.*

The program can work (with some limitations) without communication to databases. *You can design graphics for the reports or view SFR files (visual archives).*

IMPORTANT: *While reading data from the results of* ***SQL queries,*** *the program dynamically allocates (malloc) a memory block of only 3/4 MB. The read data is compressed (by ZLIB) and swapped locally in *.rwp files. So the program does not need to much memory and space to work properly on a client computer.*

9. You can also configure the user interface language of the program.

There are two built in user interface languages: English (default) and Polish.

10. What files can be opened by REPCoder ?

Each program ussually opens files that it creates itself. Therefore the only files that can be opened by our program have extensions: ***SFM*** *(report projects) and* ***SFR*** *(report results).*

11. What can you do about reporting with REPCoder ?

The program supports the following areas of the ***reporting process:***

designing	***execution***	***printing***	***archivisation***	***modification of data (*)***

(*) - additional function, enables modification of data from the executed report

13. In what sequence should the REPCoder guides be studied ?

There are 4 guides listed below. They should be read in this sequence to learn REPCoder. They were written using just REPCoder and thus they are SFM files. This way the user can directly observe and test the possibilities of the program while reading. You can make step by step exercises with the tool and the guide files. The integral part of the guides are ***sample reports,*** *listed in the "Reports" guide. They are located in the* ***"Samples"*** *directory and work with a Firebird 2.1.4 sample database: "repcoder_test.fdb".*

Guide 1 - Introduction.sfm	*This guide. Contains general information and characteristics of the product.*
Guide 2 - Graphics.sfm	*Teaches how to design the graphics of your reports. It allows you to make step by step exercises directly on that guide file.*
Guide 3 - Reports.sfm	*Teaches how to complete a report after you have learned "Graphics". Then the other stages of the reporting process are reviewed. This guide works together with the sample reports, which explain and directly present how the described mechanisms work in practice.*
Guide 4 - DLL.sfm	*For programmers only. Teaches how to call and use the functions of REPCODER.DLL API in your own applications.*

These guides have clear and dydactic form. Studying them you not only teach yourself REPCoder, but also have the opportunity to extend your database skills. There are some interesting SQL queries used in the sample reports. The leading idea of the guides is ***learning by example.*** *Theoretical considerations are compressed to the necessary minimum inside the guides. It is the practical knowledge that has the highest priority here. Just for that reason the REPCoder guides were written using REPCoder.*

Guide 2 - GRAPHICS

This guide will teach you how to put texts and other graphics objects into the pages of your report form. This basic knowledge will be necessary to learn how to finally complete a report in the next guide - "Reports".

Find the "Next page" item in the "Form" menu and go to the next page ...

Welcome to "REPCoder"

GUIDE - GRAPHICS

We are starting just now ...

1. First click your mouse left button on this caption

What happened ? Let me guess ? You can see the red border that appeared around the whole page.
If you are observant, you will also notice that the program menu is now extended.
Two additional items are added: "Edit" and "Selected". What is "Selected" ? This is this huge rectangle with the red border. Exactly in the center, you can find its dimensions: width 190 mm, length 278 mm.

Do you want to go back to the initial state ? Just click somewere in the page area outside the red border.

What happened ? The red border dissapeared. These two additional positions in the menu also no longer exist.
Now none of the rectangles is "Selected". Thus there is no appropriate position in the menu.

2. Now double-click inside this huge page rectangle

Look at the caption of the dialog window that was opened: "Texts".
Here we have just the texts of the "Selected" rectangle. If you want to open this window without using double-click, you can use the item "Texts" in the menu "Selected".

In the lower part of the dialog you can find the "List of Texts". There is plenty of them here.
This is just the contents of this huge primary rectangle. You set all the properties of the texts here: contents, font, orientation, alignment in the rectangle, offsets (distances from the borders).
The "List of Texts" displays always only the first line of each text.

3. Maybe you don't like the size of the page ?

Use standard "CTRL +" keys to increase and "CTRL -" to decrease the size respectively.
There is a 8 - degree scale of the page sizes. You can also find it in the menu "Form".
The smallest size is set in this way, that the whole page is visible on the screen.

4. Have you already noticed that little snake in the upper-left corner ?

Maybe it is attracting your attention since the beginning of the session. In the menu "Selected" you can find the item "Images", where you can insert pictures into the selected rectangle.
They are BMP, ICO, JPG, TIF, GIF, PNG files. You can also find our snake here.
Image files must be located in the same directory that the current SFM project file.
Let me remind you, that SFM is the extension of our files.

5. Have you noticed that the bottom line of our (so called primary) rectangle is much more pale than the others ?

This is an example of a hidden line. It exists but is not visible. It can be still seen on the computer screen, to remind you that it exists. It will be however completely not visible on the printout. Do you want to make it appear ? Move the mouse pointer very close to this line and click the right button. Click it once more to make it disappear again.
You set the thickness, colour and style of the lines of the selected rectangle using the item "Lines" in the menu "Selected".

Now find the "Next page" item in the "Form" menu and go to the next page (or use the numeric '+' key).

6. And now some biology lesson

Our primary rectangle has reproduced. Rectangles just like bacteria reproduce by splitting into two.
Do you want to help them to do it ?
There are 3 items in the menu "Selected": "Vertical Divide", "Horizontal Divide", "Divide".
Remember the key combinations: CTL+N, CTRL+H.

The descendant rectangles are completely identical to their parent. The only difference is the size.
They are smaller and occupy the previous parent area. After the reproduction,
the parent vanishes and no longer exists.

Here on the right-hand side there was one yellow rectangle before. But someone had selected the item "Divide" in the menu "Selected" and divided it into 3 equal parts. Now we have its 3 yellow children instead. By the way pay your attention that this text is justified with 3 mm distances from the left and right borders and 2 mm from the upper border.

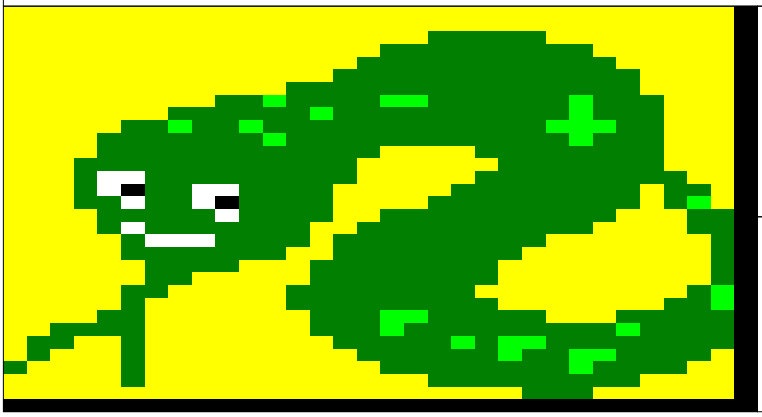
Use the "Background" item in the menu "Selected" to set the rectangle's background colour.

This rectangle on the right, was divided horizontally into 7 parts. The parent was blue, so the descendants have the blue background as well. Moreover the parent had two texts: green and red. The kids inherited them just like the colour. There is no need to explain that the parent also had a small snake in the center. Every child inherited it too.

green		**red**
green		**red**
green		**red**
green		**red**
green		**red**
green		**red**
green		**red**

All the lines of this rectangle are hidden. If you will make it the selected one (left click on it), its border lines will have a pale pink colour instead of the red (which is the normal colour of the selected rectangle). This is how the hidden lines of the selected rectangle look like on the screen. Here I remind you, that they will not be visible on the printout.

Use the keys CTRL+L or the "Lines" item in the menu "Selected" to check the settings of this rectangle's border lines. Notice that the bottom line has the DOT style. The left line is hidden. The upper and right lines are thicker and have colours.



The rectangle on the left-hand side has the "snake.bmp" image inside. The image is deformed and expanded, to fit in the whole rectangle. Open the "Images" dialog in the menu "Selected" to check how it is configured. Notice that the "Stretch" option is selected.

Since we have so many pictures on this page, try F4 function key or find the "Show/Hide" item in the menu "Form". It works as a switch showing or hiding all the images on the page.

Go to the next page ...

REPCoder

7. Rectangles - how do they move and vanish ?

We have a yellow rectangle below. Its size is 50 x 50 mm. Left click to make it the selected one. The red border will appear around it. Place the mouse pointer on any of its lines. The mouse pointer changed and now you have the possibility to move the line. Try to move each of the 4 lines. Observe that while moving the upper or the right line, some other rectangles are also moved. Only when moving left or bottom lines we don't make a panic among the rectangles around. It only influences the nearest neighbour, who may be eaten by accident when we are moving the line to quickly. The texts alligned in the rectangle's corners are also moving with them. The central text has always the appropriate position.

left upper right upper

Try to move my borders

I am very afraid of being swallowed by my right neighbour when its left border is moving to fast in my direction

left bottom right bottom

8. How to move rectangle borders using the keybord arrow keys ?

How it works ? Do you want to move the left border with arrow keys for example ? Click the mouse in the left half of the yellow rectangle. Now use the left and right arrow keys. Then click in the upper half of the rectangle. Use up and down arrows to move the upper line. It's very simple.

IMPORTANT !

9. How to move all the rectangles collectively (horizontally or vertically) ?

To try this behaviour click just above the bottom line of our yellow sample rectangle. You already know, that the up and down arrow keys can move this line now. Press the CTRL key as well. Observe, that now all the rectangles lying above this line are moving up or down without changing their sizes. Only the rectangle that is at the same top of the page will be changing its size. The collective movement of these rectangles occurs at the cost of that top one. You can obtain the same behaviour moving the border using mouse and the CTRL key.

Now click just below the upper line. Press the CTRL key again together with the up or down arrow. All the rectangles that are lying below this line are moving now. Their sizes stay unchanged. Only the one at the same bottom of the page suffers and changes its size. You can also make the horizontal collective movement selecting left or right border of some rectangle. Teach yourself this technique carefully. It will be very usefull for you in the future (especially the vertical collective movement). Remember the important properties of the CTRL key. You can also find this behaviour in the program "Selected" menu.

10. Be carefull while moving the lines. You can make some rectangles to dissapear this way.

The moving border of a rectangle swallows without any warning its neighbouring area. If you don't stop it, it will eat one by one other rectangles and will be stopped only on the same border of the page. And this border can also be moved on the distance of a few milimeters. This movement is only limited by the page margin minimum value, which cannot be smaller than 0 mm. If you loose some rectangles by accident, there is a possibility to reconstruct the previous state. Use the "Undo" item in the "Edit" menu (CTRL+Z). The program can remember up to 20 previous states of the current page.

REPCoder

11. Why the size of a rectangle is a multiple of 1 mm ? Can we work with higher precision ?

A good question. Some reports will require more accuracy than 1 mm. You have another alternative: 1/10 mm precision. I hope this will fully satisfy yours and your clients needs. Use the "Properties" item in the "Form" menu to switch to the higher (1/10 mm) resolution.

It will however require more patience from you when moving the rectangles or their borders by arrow keys. They will move very slowly with 1/10 mm steps.

12. Can we make them move faster ?

Yes. Just press also the SHIFT key together with arrow keys. If you are working with 1 mm precision, it will double the speed to 2 mm/step. If your form has the 1/10 mm precision, the SHIFT key will increase the speed 10 times, so you will move with 1 mm/step.

13. Can we always switch the resolution from 1 mm to 1/10 mm ?

The answer is YES. You can always do it. But the come back to 1 mm precision is not always so easy to do.

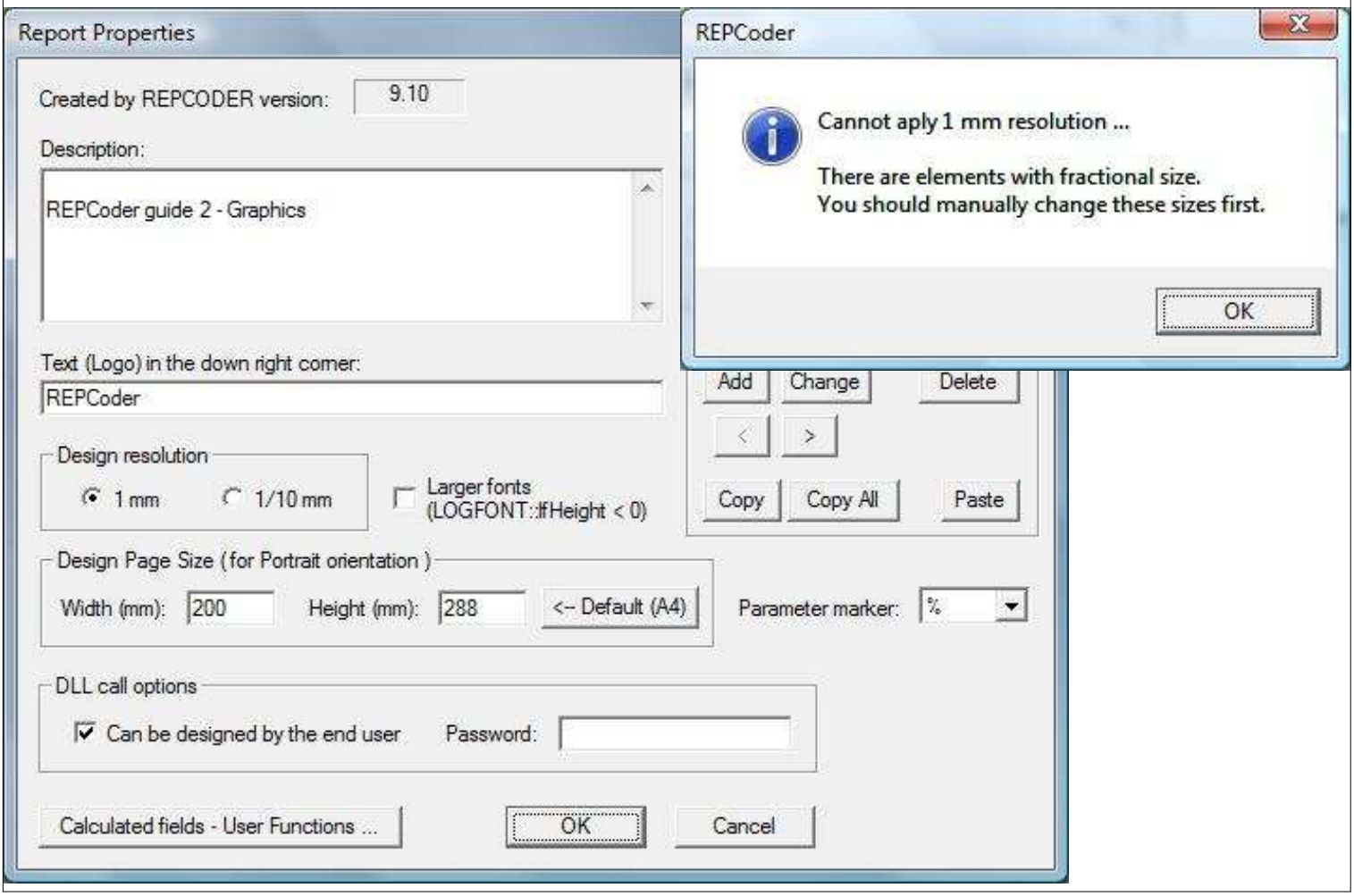
14. Can we always switch the resolution from 1/10 mm to the default 1 mm ?

The answer in NO. You cannot do it if there exists any rectangle with fractional dimensions in your form. Try to verify it. You will receive a warning message, that you should first manually change these fractional dimensions to integer values. It could in principle be done automatically, but it is more safe that you decide yourself where to cut or add these fractions of a millimeter. If there are no longer fractional-size rectangles in your project, you can switch to the default 1 mm accuracy.

15. What is: Design Page Size ?

It is the size of (all !!!) pages in the report project form. The default value is 200mm x 288mm (A4 format in the "Portrait" orientation). You can however define different values here for your specific reports. For example to work with A3 format, you should enter: 288mm x 400mm. But remember to choose "A3 paper" in the printer settings when you print the report. You can also use other values, which correspond to the paper size in your printer (for example: 92mm x 92 mm). The values you enter here are for the "Portrait" orientation, so it must always be: Width <= Height. The most important here is the ratio of these 2 values. It should be exactly the same as the ratio of the paper width and height in your printer. Then you will fully obtain the ***WYSIWYG*** *effect (what you see is what you get) - the screen and the printer will show exactly the same.*

Do you already recognize these pictures ? If not, practice some more with this resolution.

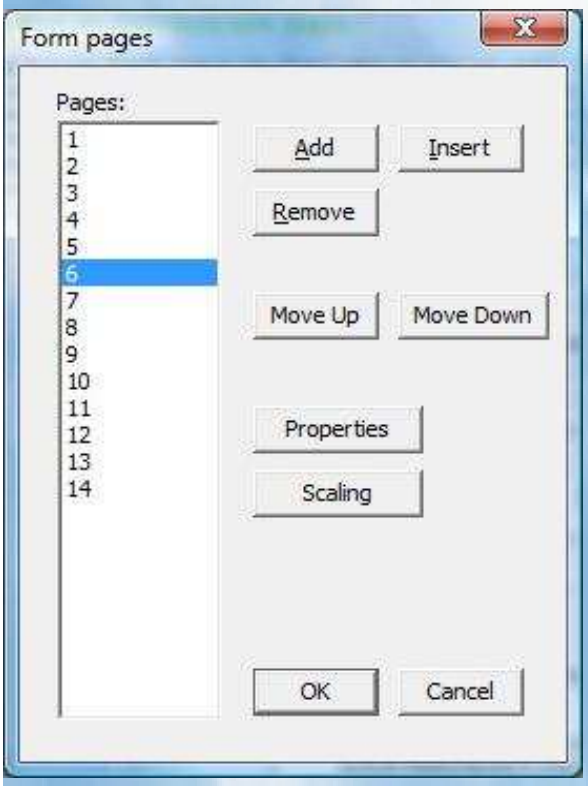


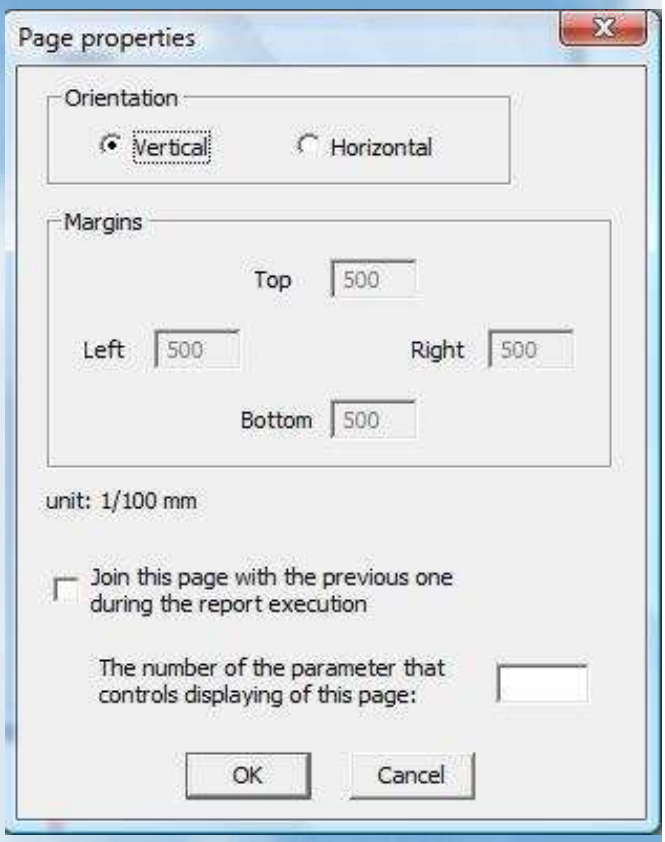
16. Can we scroll the window wile using the arrow keys instead of moving the rectangle's borders ?

Yes. But it is possible only in the state without the selected rectangle. In other words, if none of them has this characteristic red border around. You can go to this state only one way - just click the mouse on the area outside the page borders. This way the two menus: "Edit", "Selected" will also dissapear. We have already mentioned it at the beginning of this guide. Now try to scroll the entire page with the arrow keys. By the way, have you noticed that the program's reaction to any mouse click is much retarded on this page ? It will be explained soon on this page.

17. How do we create new pages ?

We manage the pages in the "Form pages" dialog. Use the "Pages" item in the "Form" menu to open this dialog. To add a new page at the end use the "Add" button. To remove a page use the "Remove" button. You can also insert a new page before the one selected in the listbox ("Insert" button). It is also possible to change the order of pages ("Move Up", "Move Down"). You also set the orientation and margins for each page ("Properties" button). But the margins can be changed here only at the same beginning of the page design work. It is possible when the page contains only one rectangle (called the primary one). In all other nontrivial cases, you cannot modify margines in this dialog. There is a danger to swallow some boundary rectangles this way. Thus it was blocked. **You can also make the page to be joined with the previous one during the report execution.**





We set the margins in 1/100 mm units. The minimum value is equal to 0 mm. The default values are 5 mm (500 units).

18. So, how do we change the margins of a page containing more than one rectangle ?

The solution is very simple. You only need to move the border of one boundary rectangle using mouse or arrow keys. By making this rectangle the selected one, you can change the left or right margins of the current page.

19. This page has "Clouds.bmp" as a background bitmap. What about the page images ?

You have to distinguish between the images of a rectangle and the images of a page. We have already mentioned the former. The latter we configure using the same dialog, but from a different position in the program menu. As you know, we manage the selected rectangle's images from the "Selected" menu. The images of the page are configured from the "Form" menu. There is also "Images" item here.

IMPORTANT: The images of a page do not depend on its division into rectangles.

You can add many images to a form page, setting alignment and offsets for each of them. You can also stretch the image to the page. You should keep in mind, that the page images are displayed (in the sequence they were added) before the images of the rectangles. Thus they really work as background pictures of a page. Then the images of the rectangles are displayed on this background (also in the right sequence). Then at the same end the rectangle texts are displayed. Notice, that the small "snake.bmp" in the center of the page is also the page image. Because it was added after "clouds.bmp" it is visible on this background. Otherwise the snake would be "obscured by clouds". As was already mentioned at the top of this page, all the operations on this page are strongly retarded (for example the movement of the rectangles). The reason is that now it takes much more time to redisplay the page that contains images. But you can use F4 function key and temporarily hide all pictures.

This page has the horizontal orientation in contrast to all previous pages.

Check the appropriate settings in the "Page properties" dialog (Form->Pages->Properties). The left margin is also different than the default. It has 1 mm size.

The REPCoder design rules:

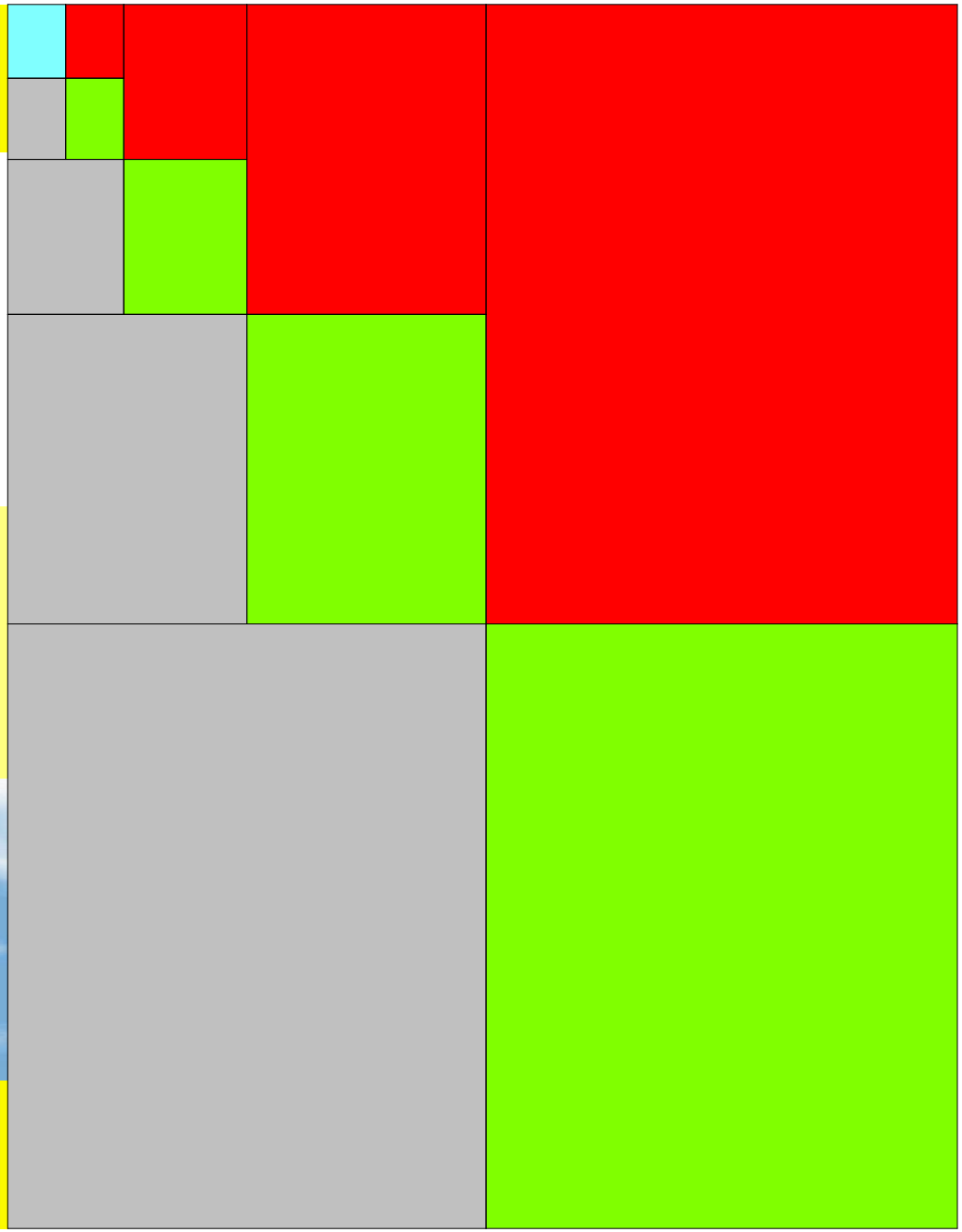
1. *The rectangles cover the page **entirely** and **separately**.*
2. *Thus there are no **holes** on the pages.*
3. *The rectangles never **overlap**.*
4. *The motion of rectangles never violates these **rules**.*

The division of a page into seperate rectangles resimbles the division of the european continent into countries. There is no area in Europe that doesn't belong to any country, or belongs simultaneously to more than one. The only difference is, that Europe and its countries do not have rectangular shapes.

Your pages, just like the Nature ...

"are afraid of the vacuum"

This concept is original and specific to "REPCoder" when you compare it with other reporting tools.



More about texts ...

20. How to inline database texts between the words of a "normal" text ?

My friend ***John Smith*** was born ***12 JUL 1967*** in ***New York***.

The blue (itallic font) texts in the above sentence represent database data. They will originate from the result of some SQL query and can have different lengths. There is a smart mechanism in our program that allows to expand these data between the words of normal text. Check the contents of the above rectangle to see how to obtain this effect. The concept is very simple. If you want a few texts to be displayed one after another, no matter how long they are, you need to set the same alignment and offsets for each of them. It means that they should be displayed exactly at the same origin and thus overlap. But designing such a situation would make no sense. Thus the program interprets it, that such texts should be displayed one after another. The texts can have different fonts and colours. This is a very usefull technique and you will be using it very often. Remember about the spaces at the beginnig or end of some texts. The mechanism itself does not add space characters between the texts.

Here we have another example. The inlined texts all have 4 mm offsets from the left rectangle's border:

The total salary in the **ZOLTAR** company in the month **July** of the year **2000** was equal **$ 45 000**.

21. What effect do we obtain when we check "Format" together with the text alignment ?

To see the difference, compare the way the same text is displayed with and without formatting:

	formatted:	**not formatted:**
Aligned to the left	The formatted text is expanded inside the rectangle regardless of the number of lines and their lengths. The automat that breaks it into lines takes into account the offsets. These offsets always apply to both opposite borders of the rectangle. In this case 5 mm is the exact distance from the left border and the minimum from the right border. A not formatted text has always the number of lines equal the entered in the editor. There is a danger that they will not fit inside the rectangle's area.	The formatted text is expanded inside the rectangle regardless of th their lengths. The automat that breaks it into lines takes into accou These offsets always apply to both opposite borders of the rectangl is the exact distance from the left border and the minimum from th A not formatted text has always the number of lines equal the ente There is a danger that they will not fit inside the rectangle's area.
Centered	The formatted text is expanded inside the rectangle regardless of the number of lines and their lengths. The automat that breaks it into lines takes into account the offsets. These offsets always apply to both opposite borders of the rectangle. In this case 5 mm is the minimum distance from the left and right borders. A not formatted text has always the number of lines equal the entered in the editor. There is a danger that they will not fit inside the rectangle's area.	atted text is expanded inside the rectangle regardless of the number of lengths. The automat that breaks it into lines takes into account the off ffsets always apply to both opposite borders of the rectangle. In this cas is the minimum distance from the left and right borders. formatted text has always the number of lines equal the entered in the There is a danger that they will not fit inside the rectangle's area.
Aligned to the right	The formatted text is expanded inside the rectangle regardless of the number of lines and their lengths. The automat that breaks it into lines takes into account the offsets. These offsets always apply to both opposite borders of the rectangle. In this case 5 mm is the exact distance from the right border and the minimum from the left border. A not formatted text has always the number of lines equal the entered in the editor. There is a danger that they will not fit inside the rectangle's area.	xpanded inside the rectangle regardless of the number of lines and The automat that breaks it into lines takes into account the offsets. s apply to both opposite borders of the rectangle. In this case 5 mm stance from the right border and the minimum from the left border. text has always the number of lines equal the entered in the editor. There is a danger that they will not fit inside the rectangle's area.

22. We also have the "Justify" option for our texts. Justified texts appear very often in our guides.

23. Database texts can be inserted and mixed with normal texts. They can be single or multiline. All can be formatted or justified. Each text can have its own font and colour.

My friend **John Smith** was born ***22 JUL 1950*** in *New York*. He attended school between 1957 and 1965. Then he went to study in **England** and continued it for the period of 5 years.

My friend **John Smith** was born ***22 JUL 1950*** in *New York*. He attended school between 1957 and 1965. Then he went to study in **England** and continued it for the period of 5 years.

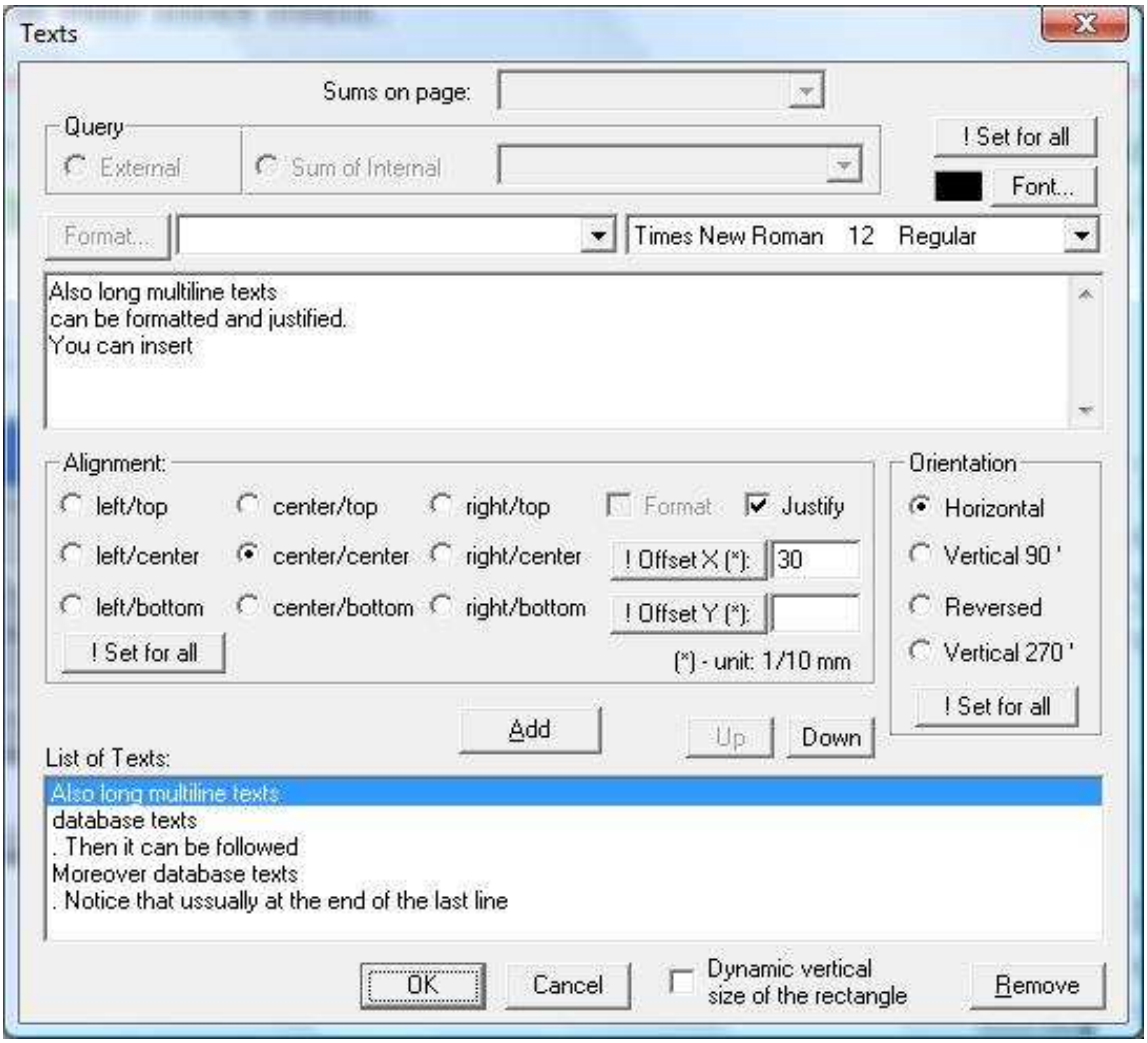
When you analyse the texts of these rectangles, pay your attention to the identical alignment with the "Justify" option set for each text

The sum of salaries in the company **"Very long name"** in the month august of the year 2000 was equal **$ 58 234** (in words: ***fifty eight thousand two hundred thirty four dollars***).

Also long multiline texts can be formatted and justified. You can insert **database texts**. Then it can be followed by normal texts again. **Moreover database texts can also be multiline**. Notice that ussually at the end of the last line we add a space character to separate the texts after formatting. The mechanism doesn't recognize the end of line characters. It treats the entire text as one and splits it into lines itself.

24. To set the properties of all texts of the Selected rectangle use the 5 buttons beginning with "!" character in the "Texts" dialog.

You can quickly change font, alignment, orientation and offsets.



The "Texts" dialog will be the one most often used during your design work. So lets watch it more carrefully.

In the upper and middle parts we set the properties of a text. Then we use the "Add" button to add it to the "List of Texts" at the bottom of the dialog. Only first lines of texts are displayed in the list. The "Up" and "Down" buttons allow to change the order of texts inside the list.

The "Format" button (disabled here) in the upper-left corner will be used to set the display format of a dabase text. There is a combobox next to it, with result field names of the SQL query. The queries will be defined in some other place. But it will be discussed in the "Reports" guide.

Let me also remind you, that the offsets X, Y from the rectangle borders, we set in 1/10 mm units. You can use the two buttons to set the given X or Y value for all texts in the "List of Texts". They are the 2 of these 5 buttons beginning with "!". The other 3 buttons have the same name "! Set for all" and allow you to set the same value of font, alignment and orientation for all texts.

25. As you have probably already noticed, there is also a possibility for vertical texts in our reports. There are 4 possible text orientations:

Check the contents of the yellow rectangle and see how to set the orientation of texts.

Horizontal orientation

Vertical 270' orientation

Vertical 90' orientation

Reversed orientation

26. You can format and justify texts of all orientations.

Something about clipboard ...

27. We can access the clipboard only if there exists the Selected rectangle on the page. Only in such a situation there exists the "Edit" menu where you access the clipboard functions.

The clipboard will be very usefull during your design work. It will be used in three different ways:

To copy and paste	To copy and paste	To copy and paste
rectangles	**groups**	**pages**

Important: Clipboard can be used to copy and paste the above structures within the same application and also between different applications of our program.

How to copy a rectangle:

Select this rectangle. Then copy it to clipboard. You can use CTRL+C or the "Copy selected" in the "Edit" menu. Then select the blue rectangle, and use CTRL+V or "Paste selected".

Use "Clear selected" in the "Edit" menu or Delete key to clear the contents of this rectangle. Then restore its blue background (in the "Selected" menu).

Important: *The entire contents of the rectangle is coppied (texts and images), background colour and border lines. On "Paste" only the size remains unchanged. You can make the program to skip the lines and background on "Paste" operation. There is a checkbox for that in the "Options" dialog. When we "Clear" the rectangle, all is removed except the lines.*

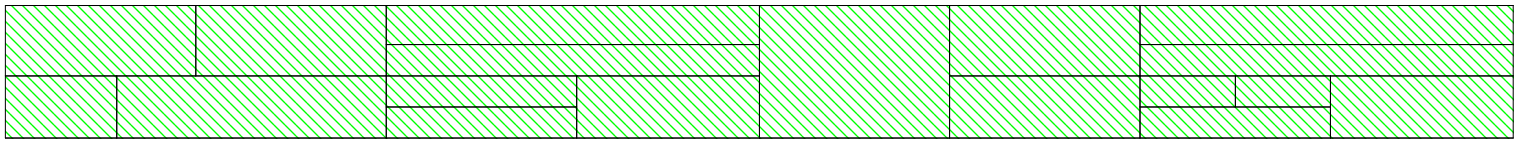
28. What else can we find in the "Edit" menu ?

*There are also 2 additional functions often used during design work. They are **"Undo"** and **"Redo"**. You can alternatively use "CTRL+Z" and "CTRL+Y" to access them. There is a possibility to reconstruct the state of the **current page** up to **20** steps back. You can use the CTRL+Z and CTRL+Y shortcuts only in the state with the Selected rectangle (when the "Edit" menu exists). After going to a new page the memory is cleared and the "Undo"/"Redo" functions do not work.*

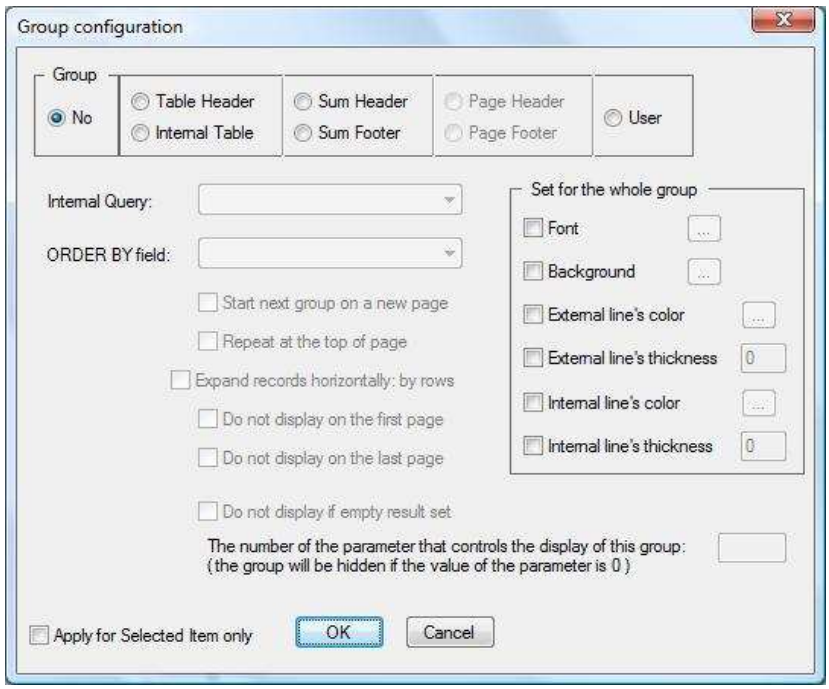
Important: *When we copy and paste **the groups** (a collection of rentangles that also form a rectangle), the group is pasted into the interior of the Selected rectangle. The rectangle becomes divided into smaller ones. The proportion of their sizes in the original group is conserved. We talk about groups on the next page.*

29. What are the groups ?

A group is a collection of rectangles that also together form a rectangle.



We have the example of a group above. They can be easily recognized on the pages of our forms because of their specific background. It is always the square or diagonal net. Its kind and colour is specific to the type of the group. There are 7 types of groups. We access the group's configuration dialog by the "Group" item in the "Selected" menu. This way we set the properties of the whole group that the Selected rectangle belongs to.



"Apply for Selected Item only" (bottom-left corner) means that the current configuration will be applied for the Selected rectangle only. You can add or remove it from a group this way, or transfer to a different one. You cannot however violate the basic rules of the group existence. For that reason, this checkbox is not always enabled or program will not allow to apply the settings. There is a high level of the program security and control during the group configuration. Notice, that you can also change the font and colour of all texts in the whole group. The display of a group can be controlled by one of the report's parameters. It will be presented later in one of the samples.

30. Why do we need groups ?

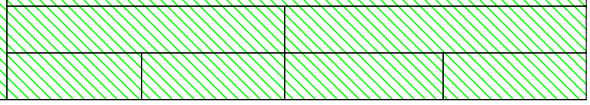
The groups represent **dynamic elements** of the report on the project form. They exist to make it possible to place **dynamic tables** inside our report. Without groups our form would be a **static** one. In general, the form can have both static and dynamic elements. A standard rectangle that doesn't belong to any group (a static element) has the group configuration set to **"No".** This is just the group configuration of all our previous rectangles in this guide. Here we will review the properties of groups. But we cannot show here these most important types: **"Internal Table"**, **"Sum Header"**, **"Sum Footer"**. The reason is, that they can be placed in the form project only after **database connections** and **SQL queries** are defined. But it is not the goal of this graphics guide.

Some more examples of groups:

This is a group of "Table Header" type. If there will be one of the working groups placed directly below it, then this group will be duplicated after the execution of the report. It will be placed always at the top of a page, playing the role of a header for the lying below records of SQL query or its partial sums.

This is a group of the "User" type. Its main destination is that we can put here a collection of rectangles that we want to keep together on the same page after the report's execution. Belonging to the same group always protects the rectangles from being divided on the page boundaries. You will ussually use it for making complicated sums of "Internal Queries" to have them on a single page. You can also use "User" groups during the design work. It is a common practice to put some rectangles temporarilly into a group, copy the group to clipboard and then paste it into the contents of some other rectangle.

These rectangles will always be kept together on the same page after the execution of the report, because they belong to a group. They are not afraid of page boundaries. All types of groups guarantee this behaviour.



There is no need for a group to be as wide as the page. But it is forbidden to place 2 groups horizontally one by one. The groups must always lie **vertically "one below another"**.

You cannot create any group here because there is already one on the left. But you can always add this rectangle to that group if you want.

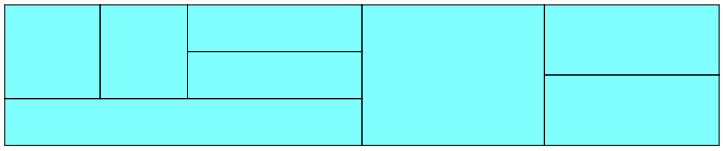
This group is of the "Page Footer" type. It can be created only at the same bottom of a page. This group guarantees that it will always have its reserved place here on the executed report. This place will not be used for result records of SQL queries. Its other important destination is that you can place only here (and also in the "Page Header" group) the sums of records on page and transfered from previous pages.

This group is of the "Page Header" type. It can be created on the same top of a page only. It will have its reserved place here after the report execution.

31. Groups can aslo be created using the mouse.

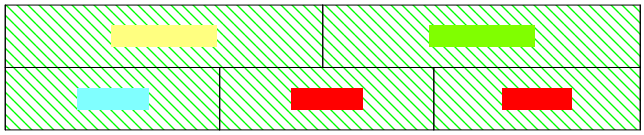
To do this, you need to select the rectangles using the left mouse button. If they together form a rectangle, the group configuration dialog will be opened. Otherwise a warning message is displayed.

As the exercise, select with mouse the blue rectangles and configure the group as a "User". Then configure it as "No" to restore the original state.



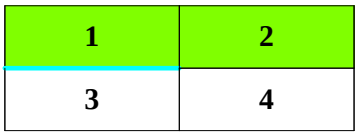
32. The background colour of rectangles in a group.

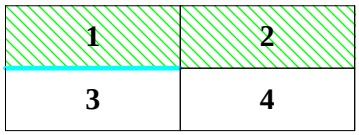
We have a sample group below, with rectangles that have some background colours. But the characteristic net identifying the group type must aslo be displayed (only in the "Design Mode"). Thus the real background colour (that will be displayed on the executed report) of the rectangles is shown only in some limited area in the center. The specific group net is displayed on the edges:



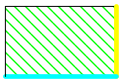
33. The motion of rectangles in the neighbourhood of groups.

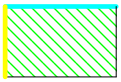
The examples below show how the presence of a group influences the motion of surrounding rectangles. In the case without groups (left example), the motion occurs according to already described rules. In the case with groups (right example) these rules are slightly modified, because of the requirement that the group must always have a rectangular shape. To see the difference, try to move that blue border line between the rectangles 1 and 3 in both examples. In the left example, the motion is free and has no effects on the other rectangles (2 and 4). In the right example, the border of the whole group is moved together with the blue line. The rectangular shape of a group has higher priority and cannot be violated. But you can move the border between 1 and 2 rectangles without problems.





IMPORTANT: *The program security also controls, that the groups alway lie vertically one below another. In the previous left example, try to join 1 and 2 rectangles into a group. The program will not allow this, because there already exists a group on the right hand side. Here we have another example to show how this requirement is controlled during the motion. Try to move the blue lines here. Try also to move to the right yellow line in the left rectangle. Its motion is possible only to the border of the right rectangle. Otherwise the groups would join and form a non rectangular shape.*



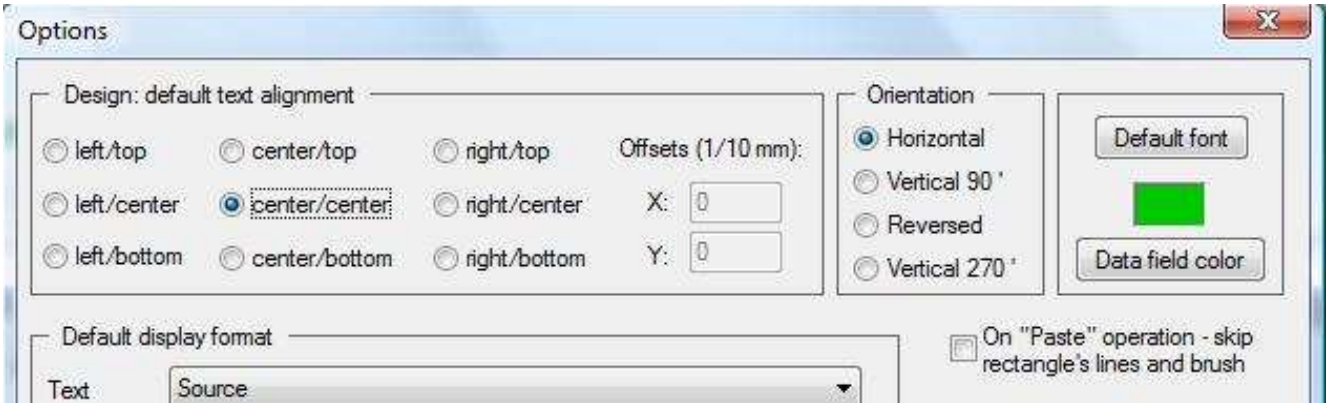


The principles of group existence and interaction:

1. *A group as a whole always has a **rectangular** shape.*
2. *Groups must be lying vertically **one below another**, they cannot be aligned horizontally **one next to another**.*
3. *The above rules are always strongly controlled during the **motion** of rectangles.*
4. *The **Page Header** must always be **at the top** and the **Page Footer** must always be **at the bottom** of a page.*
5. *The rectangles of the same group will always be lying **on the same page** in the executed report. A group will never be divided on the page boundary.*

34. The program configuration - "Options" dialog.

REPCoder has some default settings - the configuration. You find the "Options" dialog in the "File" menu. But it is only its upper part that we will describe in this graphics guide - "Default Text Options". Here we configure the default properties of every new text added to rectangles in the "Texts" dialog.

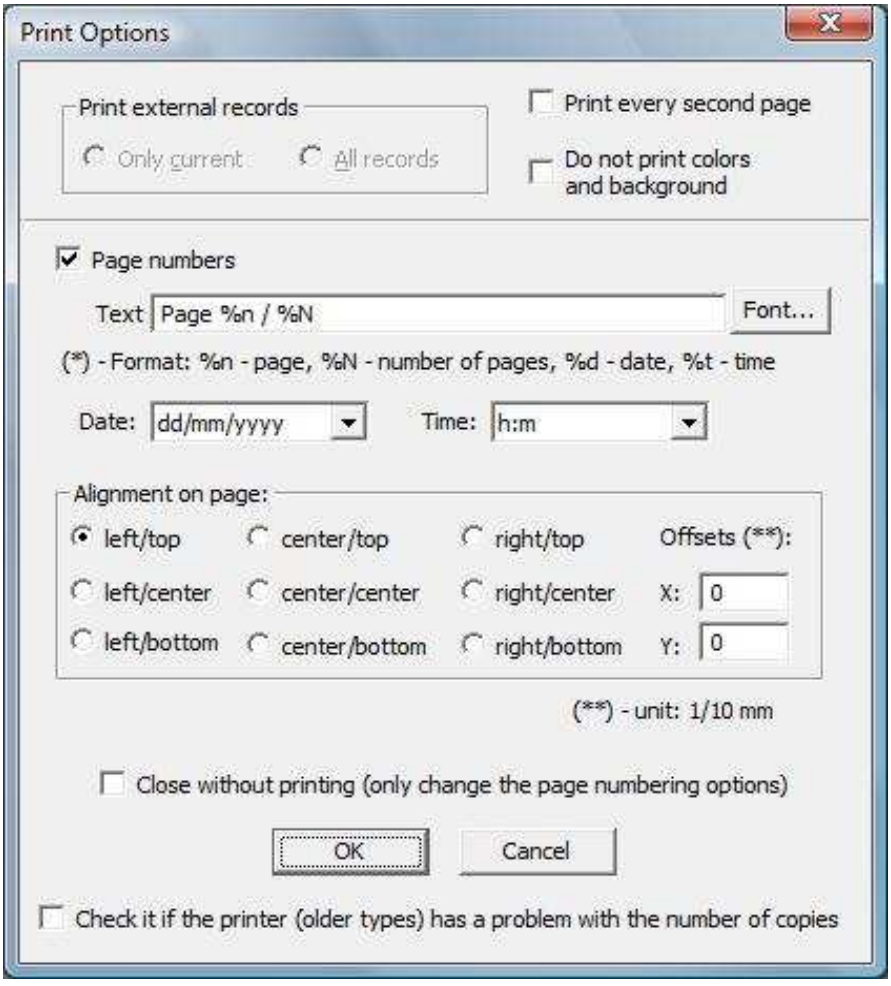


35. WYSIWYG technology - *"what you see is what you get"*

*There is no **Print Preview** option in our program. It is completely unnecessary because everything what you see on the screen will be exactly on any printer. The most important here is the line breaking problem of multiline texts that are formatted or justified. It is very important for users and programmers. Remember to set correct **Design Page Size** in the "Form-->Properties" dialog. The default values (200mm x 288mm) correspond to the standard A4 paper format. But you can define different values for your specific reports if your printer is using a different paper size.*

***IMPORTANT:** Practical realization of the WYSIWYG technology in graphical applications **is not a trivial** problem. The standard Windows graphics does not offer such a possibility. So most of the standard applications simply cannot do it. It is reserved for professional programs only and is a precious feature of our tool. Another important mechanism - text justification is also implemented very carrefully and professionally although Windows does not directly offer this option.*

36. Printing and page numbering.



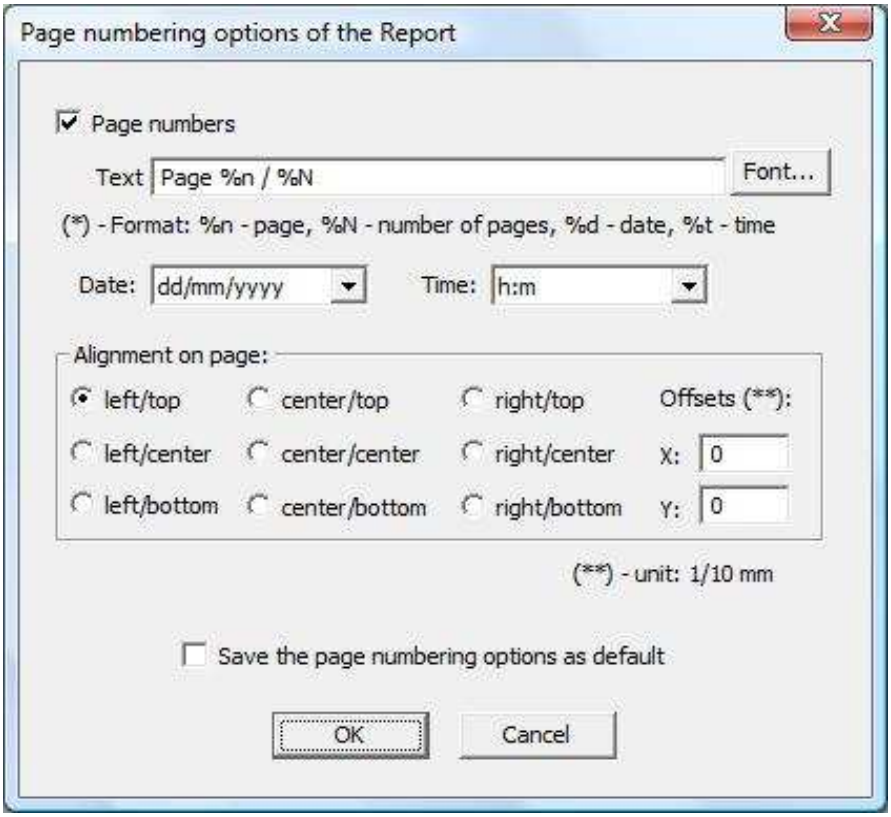
*We open the "Print Options" dialog with "Print" item in the "File" menu. Here we switch on or off the page numbering. When this option is on, we also choose font and format of the text to be displayed. You can use some symbols which will then be translated to: current page number - **%n**, the total number of pages in the printout - **%N**, date - **%d**, time - **%t**. We also set the **display format** for the date and time and the alignment of the numbering text on pages.*

This dialog is displayed only in the Working mode (after the report is executed with F5 key). You can test it with sample reports, where database connections and SQL queries are defined.

After you close the dialog with the OK button, the program goes to the printing mode. If you don't want to print, but only save the changed options, mark the checkbox "Close without printing ...". The bottom checkbox is provided for printer drivers of older types, that have problems to correctly print the required number of copies of the document. In such cases the program does not rely on driver, but prints all the copies itself. It is however slower than in the case of new drivers.

37. Can we save page numbering options in the report project file ?

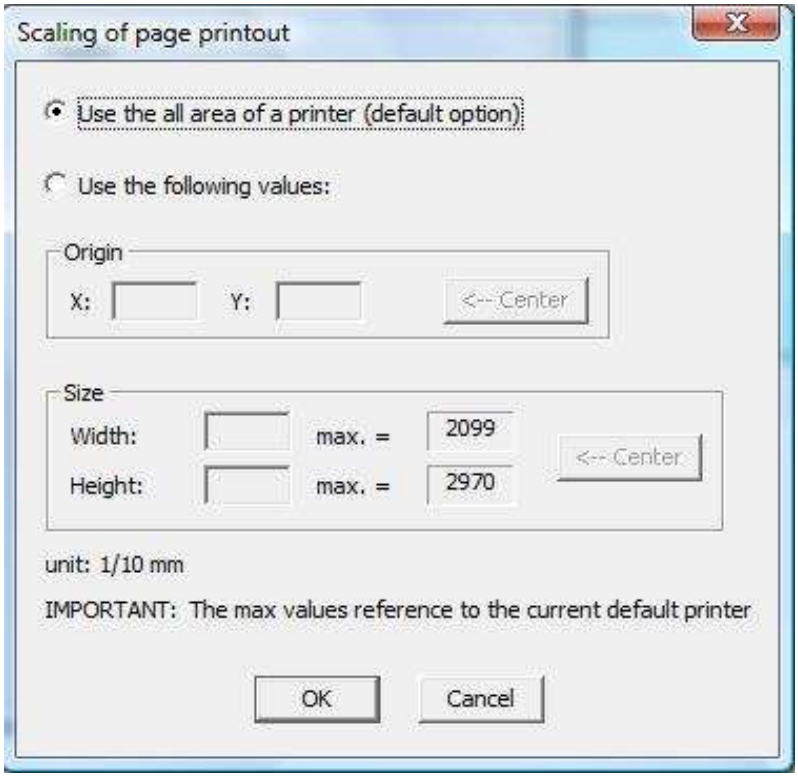
Yes. We configure it in the "Page numbering" item in the "Form" menu (picture on the right).



38. Scaling of the page printout.

There is a built-in scaling and guiding mechanism that works when we print pages. It is mainly used in situations when we want to insert database texts exactly into correct places on special ready paper forms in the printer. We open the appropriate dialog with the "Scaling" button in the "Pages" dialog in the "Form" menu.

By default pages are printed using the whole area of a printer. So they can be slightly stretched or squeezed as needed (depending on the given printer). If we want to obtain exact dimensions of the printout, we must define our own scaling. We must provide the origin and extension of the printout (SetVieportOrgEx and SetViewportExtEx system functions). Additionally if we want the image to be centered, we can do it in two different ways. First we can set the coordinates of the printout origin relative to the upper-left corner of the page. The default origin is (0, 0). Then using the lower "Center" button, program calculates the printout's size for the given origin, so that all is centered on page. The other way works the opposite way. First you introduce the size of the printout and then you can use the upper "Center" button to calculate the origin coordinates (to center the printout on page). All numbers are in 1/10 mm units. The displayed here maximum size of the printout is for the current default printer. Of course the printout does not need to be centered on the page if you don't want it. You can set the printout origin and size manually. This way it is possible to find by the trial and error method, the optimal values and obtain the best position of the page on the paper.



Guide 3 - REPORTS

After the "Graphics" guide we can go to the final step. You already know how to put texts and other graphics objects into our report project form. This guide will teach you how to connect the form with databases and to complete the report. We will also review the other aspects of the reporting process - execution, printing, archivisation and also modification of data.

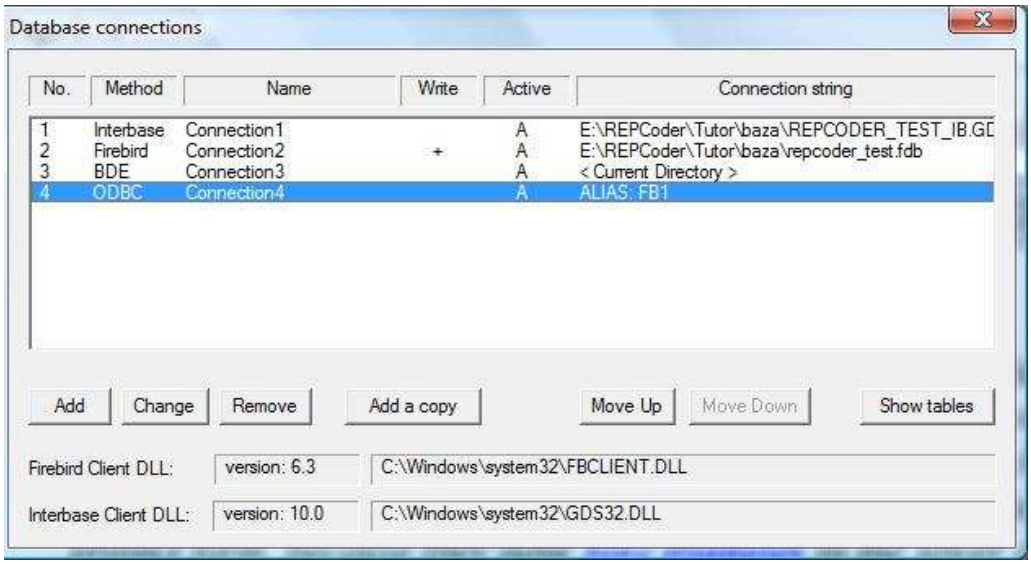
IMPORTANT:

We assume that the user is familiar with databases and SQL language.

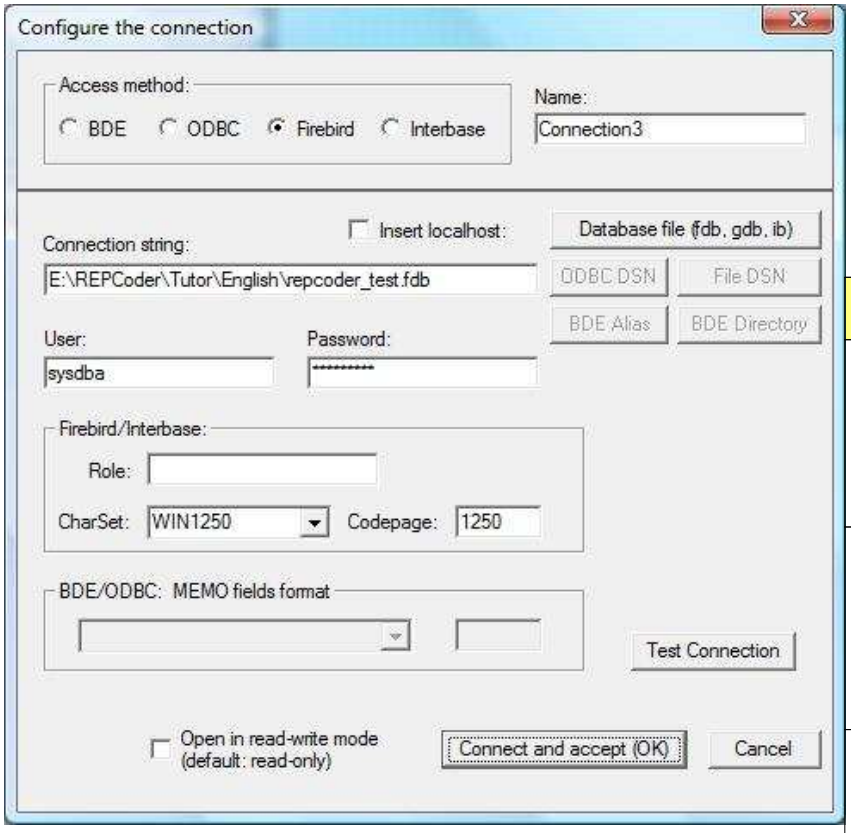
1. What are the database access methods ?

One report can use many databases. Each of them can have its own data access method. Currently there are 4 independent methods in our program: BDE, ODBC, Firebird, Interbase. The Firebird and Interbase methods use directly the client libraries: FBCLIENT.DLL and GDS32.DLL / IBCLIENT64.DLL respectively. To load a client library, the program is using the system function: LoadLibrary().

2. How we define database connections for a report ?



You have to use "Database connections" dialog (menu "Form"). Here you can add, change and remove database connections. You can also change their order on the list. It is important, because the connections are established in the sequence defined here. For each database you can also see the list of tables. At the same bottom there is information about Firebird and Interbase client libraries (version and path), that the program has loaded (with LoadLibrary()).



*Here you define a connection. At first you have to choose the **access method (BDE, ODBC, Firebird, Interbase).** For each method you enter the **Connection string, User, Password.** The name of the connection is only for you. It is shown in this dialog and in some other dialogs, where you define SQL queries for a given connection. It has no other meaning, but is required.*

Connection string:

For Firebird/Interbase methods you enter full path name of some FDB/GDB database file, together with the server name. The Role (optional) and CharSet (required) are also for this method only.

For ODBC method there are 3 possibilities for the connection string: 1 - ODBC alias (DSN defined in ODBCAD32.EXE application), 2 - the name of a .DSN file, where the connection string is defined, 3 - the ODBC connection string directly (with semicolons ;)

*For BDE method the connection string can be a **BDE alias.** It can also be a **directory name** of a standard database that represents paradox (db) and dbase (dbf) files.*

3. What is the "MEMO fields format" ?

MEMO fields in a database can have different formats. Here you can choose what this format exactly is. The program will then try to translate it to UNICODE and display on the screen. This option is available only for ODBC and BDE methods. The Firebird (Interbase) method receives MEMO data from the client library in the CharSet defined as the connection option.

4. What is the checkbox "Open in read-write mode" ?

We ussually create reports only to view, not change the data. However the program allows the user to append, update or delete data after the report is executed. To be able to do it you must check this option here while designing the report. If this option is not checked, REPCoder will be trying to open a read-only connection to the database. This is the default behaviour.

5. What is the reason to choose more than one database for our report ? Is a single connection not good enough ?

*In most cases our report will be using only a single database. It can be the name entered (or browsed) manually, or some alias (BDE, ODBC). However we will be able to define many **SQL queries** in our report. **Each of them can have its own database.** This way we can create multi database reports.*

6. Can a single SQL query use more than one database ?

*It is important to notice that usually a single SQL query cannot join data from different databases. Most servers simply do not support this. However **BDE method** allows us to define such queries. It is in agreement with the concept of **"multi-database queries"** supported locally by Borland Database Engine. To be able to do this (in BDE method) you need to choose a database representing some working directory. They are so-called **BDE standard databases.** You can find more information about such queries in the BDE documentation. And here it is important to notice that our program fully supports this concept. **But today it is not so important, because Borland has stopped to develop BDE many years ago. Use it only to access dBase or Paradox files.** For other databases use ODBC method. For Firebird (Interbase) databases REPCoder is a natural solution, because it calls FBCLIENT.DLL (GDS32.DLL/IBCLIENT64.DLL) directly.*

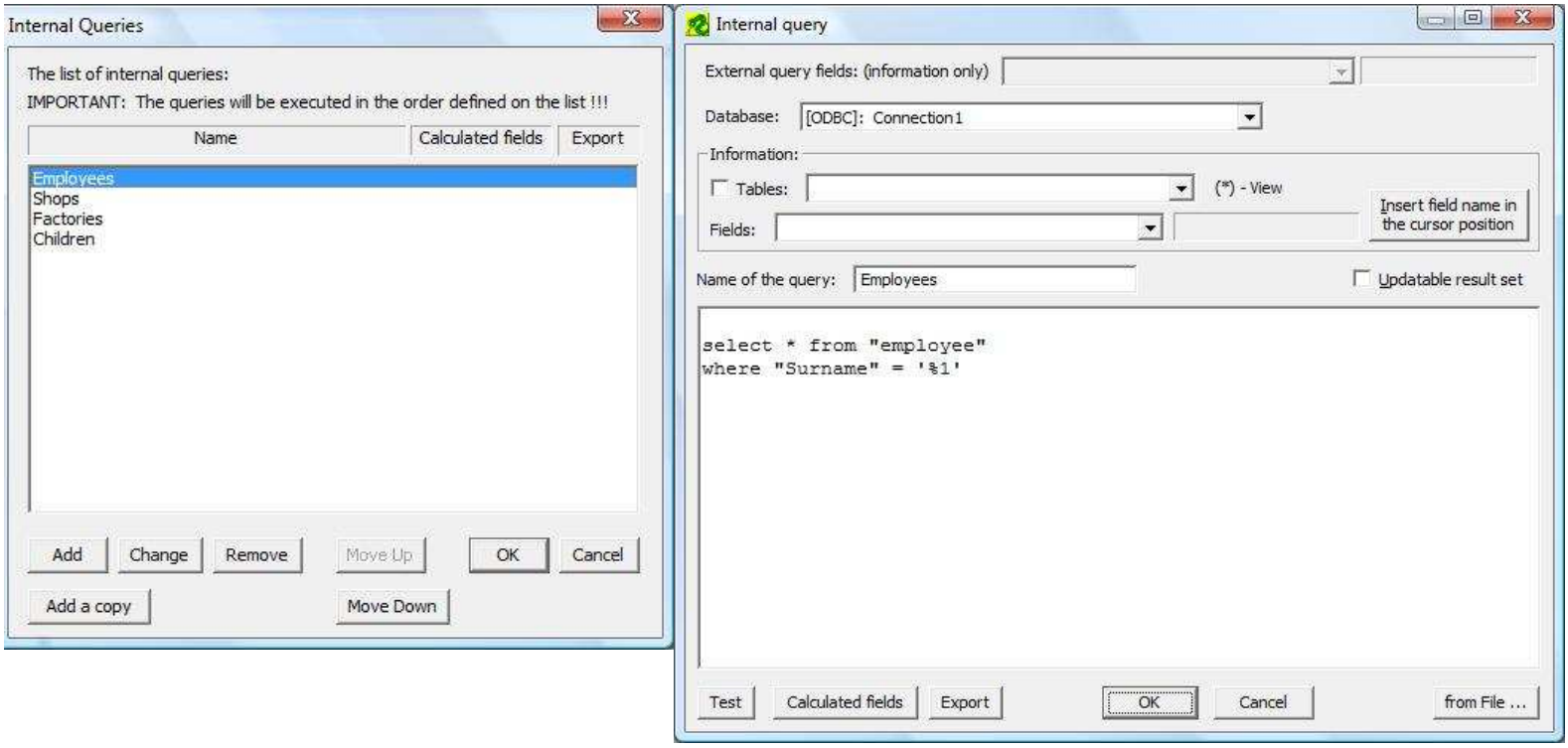
7. We have already defined database (one or more) for our report. What next ?

*Now we must write **SQL qeries.** The concept of this program is, that there are **two kinds of queries.** They are: **external queries** and **internal queries.***

8. What is external query ? How it differs from internal queries ?

*It should be emphasized first, that our report can have **only one** external query. We enter its text in the "External Query" dialog in the "Form" menu. In contrast, it can have **many** internal queries. The concept of these two kinds of queries will be explained in the following examples. At first we will emphasize, that our report is defined for **a single record** of the external query's result set. When we go to another record, the report is rebuilt. It means that all database texts are filled with new values. All internal queries are reexecuted, because they are ussually parametrized with the fields of the "external record".*

The "Internal Queries" dialog allows us to define many queries. For each of them we choose a **database** from the list of connections designed earlier in the "Database connections" dialog. The list of database tables and fields suplied here is only for information. The same for the list of fields of the "External Query" (in the upper left corner). We add, change or delete the queries. For each query we edit its SQL text. We also enter its name that will identify the query in our form. Before adding you can test the query ("Test" button). If the query is not correct it cannot be added to the list of internal queries.



Here you have 3 possible types of reports with explicitely shown external and internal queries:

Describtion of a sample report	External query *Example*	Internal query *Example*
A combination of simple personal data of an employee: name, surname, date of birth and so on. The report can have one or more pages. ***simple static report***	A query returning one record for each employee. The report is rebuilt every time you go to a different record. *select * from employees*	***No ...***
A table with the list (name and surname) of employees. The number of pages in the executed report depends on the number of records (employees). ***simple dynamic report***	***No ...***	A query returning the list of employees. It will be expanded in the table form. *select * from employees*
A report with simple personal data of employees (name, surname, date of birth, ...) and the list of children of each employee. ***complex report (always dynamic)***	A query returning one record for each employee. The report is rebuilt every time you go to a different record. The internal query is also reexecuted but with a new value of the parameter - the parent identifier. *select **id**, name, surname, birth_date, ... from employees*	A query returning the list of children of the given employee. It will be expanded in the table form. This query is related to the external query in master - detail form. The joined fields are: id, parent_id. The query has a **parameter**. This is the parent's identifier. This is a field of the record of the external query (we call it the "external record"). We edit its name in the **square bracket,** to make it the parameter of a query. *select name, birth_date from children where parent_id = [**id**]*

9. So there are 3 possible types of reports:

simple static	simple dynamic	complex
external query	***internal queries***	***external query + internal queries***

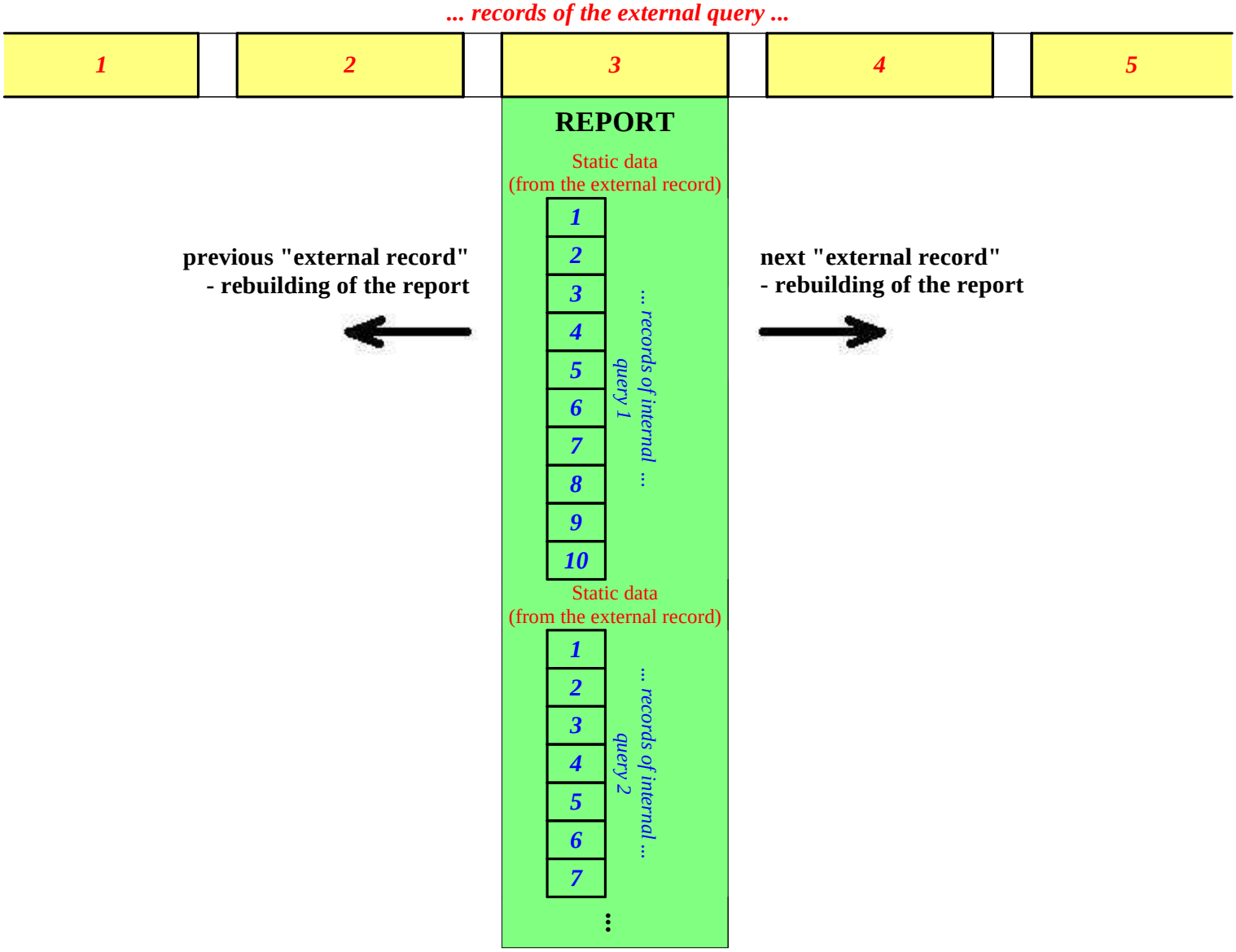
*The origin of the name **"external query"** is, that the report is rebuilt every time we go to a different record of the result set. A new report is created using the same template (form). So we always see only a single, **current** record of the query's result set in our report. So the entire result set is lying **"outside"** the report. Thus we call it "external".*

*The **"internal query"** has its name because its entire result set (all records) are displayed **inside** the report.*

*If the report does not have internal queries, it is a **static** report. There are no dynamic elements and thus no expandable tables there. The data in the report is displayed always in the same places, exactly as it was designed on the project form. The data here comes only from the result set of the external query.*

*If the report has internal queries, it is a **dynamic** report. It has dynamic elements (expandable tables). They are filled with data coming from the results of internal queries.*

*If our report has both internal queries and (exactly one!!!) external query, it is a **complex** report. It has static and dynamic elements (tables) on its pages. The data for static elements comes from the external query. The data for dynamic elements comes from internal queries.*



The above diagram illustrates the described idea. The report is always defined for a single external record. These records are lying outside the report. The records of internal queries are always displayed inside the report. There can be many internal queries in the report. When we move to a different external record, the report is rebuilt. It means that all internal queries are reexecuted with new values of parameters. These parameter values are just the values of fields of the current external record.

10. Does it make sense to define a complex report, where internal queries are not parametrized with some field values of the external record ?

There is one trivial but very often situation when such a complex report makes sense. It happens when the external query returns ***a single record,*** *which is not in any master-detail relation with internal queries. In such situations, the external record plays rather the role of a description in the report. Its static data is usually used in the report's title. For example we can have a report displaying the employee list of some company. The title can have the company's name, address and so on. The internal query returns many records (the list of employees), the external query returns a single record with data describing the company. There is only one such a company in our problem. There are no more of them or they are not interesting for us. So there will be no navigation over the external query's result set. We should emphasize at the end, that the discussed situation is not any error. You can have internal queries in the report that are not related to the external query. The resulting dynamic tables will not be changed during the walk over the external query's result set. There are situations that such reports will be created and used.*

11. Can we expand the records of an internal query horizontally on page ?

There is a possibility to configure "Internal Table" group in this way, that the records will be aligned also horizontally (if there is some free space on page). After this space is filled, next records start in the next line.

from theory to practice ...

*This guide works together with the **sample reports** (SFM files) that are located in the "Samples" subdirectory. These reports are integral part of the guide. They directly present how the described ideas and mechanisms work in practice. You should study them carefully one by one. Here we have examples of the 3 discussed basic types of reports:*

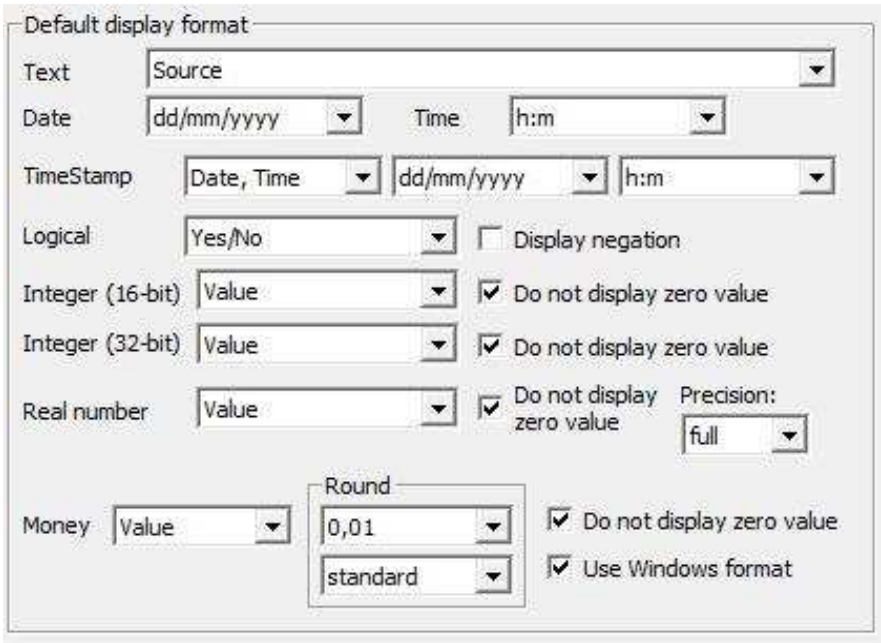
Sample report SFM file	Description
Report_01 - simple static.sfm	A simple report with the external query, without internal queries.
Report_02 - simple dynamic.sfm	A simple report with one internal query, without external - we are using Groups !!! There is also presented the **"Built-in Export"** mechanism of SQL results to text files (or to other database).
Report_03 - complex.sfm	A complex report with the external query and one related to it internal query. It is also shown how to inline database texts between the words of a "normal" text.
Report_04 - records horizontally.sfm	A simple dynamic report, that teaches how to expand records horizontally on page.
Report_05 - substrings.sfm	How to display substrings of database texts ?

*The above examples show how and where we add the texts of SQL queries (external and internal) in the report form. You can also find out how to realize the master-detail relation in a complex report. It is also presented how to add database texts, display row numbers and total number of records of the internal query's result set. Moreover, you can learn how **to execute** the report, or more generally - switch between **"Design Mode"** and **"Working Mode".** It is also shown how to navigate over the external records in the executed report. You will also see how to use the groups "Table Header" and "Internal Table" in order to create dynamic elements in the form. There is also shown an intelligent mechanism that enables to inline database texts into "normal" text.*

12. Database field types supported by the program.

- ***Text***
- ***Date***
- ***Time***
- ***TimeStamp***
- ***Logical***
- ***Integer (16-bit)***
- ***Integer (32-bit)***
- ***Real number***
- ***Money***

The default display format of data we set in the program configuration ("Options" item in the "File" menu). Here you see the corresponding fragment of the configuration dialog with the default display settings.



13. Example - display format of data

Sample report SFM file	Description
Report_06 - display format.sfm	It is shown how to set different display formats of data

14. List of display formats

Type of data	Possible display formats					
Text	*Source*	*Upper case*	*Lower case*	*Format of the rectangle*	*Image file (5 options)*	*Barcode (5 options)*
Date	*dd/mm/yyyy*	*dd.mm.yyyy*	*dd-mm-yyyy*	*yyyy/mm/dd*	*yyyy.mm.dd*	
	yyyy-mm-dd	*dd/mm/yy*	*mm/dd/yy*	*dd miesiąc yyyy*	*mm/yyyy*	
	miesiąc yyyy	*miesiąc*	*dd*	*mm*	*yyyy*	

REPCoder

Time	h:m	h:m:s	h:m:s:ms	h	m	s	ms
TimeStamp	**Date, Time**		**Date**		**Time**		
Logical	Yes / No		Y / N	1 / 0		X / Blank	
Integer	Value	In words	month name	month Name	**Logical**	Minutes as time	
Real number	Value			**Money**			
Money	Value		$ Value		In words		
BLOB: Memo	Source						
BLOB	Stretch to the size of rectangle saving original ratio	Stretch saving original ratio, but do not enlarge	Stretch exactly to the size of rectangle	Original size limited by the size of rectangle	Original size without limits		

In the diagram above, in the places where we have the names of different data types, means that all the display formats of this type are accessible. So for example, "TimeStamp" data can be displayed seperately as "Date" or as "Time". In the same way, "Integer" types can be displayed as "Logical" type. "Real numbers" can be displayed in the money formats. For "Money" type we also set ***the type of round*** *(0.01, 0.1, 1, Fraction only) and* ***the rounding method*** *(standard, to up, to down). You also decide if the money value should be displayed using the system format ("Use Windows format"), or in the ordinary way (without any formatting). For real numbers we also set the display precision.*

The formats shown on the yellow background need some comment ...

15. What are "Format of the rectangle" and "Image file" formats for the "Text" data type ?

"Format of the rectangle" works for text fields, if they have a specific value. It allows you to dynamically control the view of the rectangle where the given text field belongs to. This way you can set: background color, line colors, line widths, the rectangle's height in milimeters and fonts of the texts. How to use this option is exactly described inside the program in the dialog where you set this option.

"Format of the rectangle" format allows for example to fill the background of the rectangle where the field belongs to with a given color. The text must have exactly 11 characters in the following format: ***rrr ggg bbb****. We simply define here the red, green, blue components of the desired color. For example the rectangle will obtain the red color for the field value equal to:* ***255 000 000****.*

Suppose we want to display the photos of employees in our report. But the scanned pictures are not stored in a database, but only as ordinary BMP or JPG files on disk. Now you can put these file names into a text field in some database table. Then you can use the "Image file" display formats for this text field. It will make the program to display the right graphics file inside a rectangle. The alignment of the displayed picture will be the same as set in the "Images" dialog of the rectangle. Choosing one of the 5 available options will result in the picture fitted to the rectangle's size as you need. The supported file formats are the same as those that can be placed on the pages of the forms (BMP, JPG, ICO, TIF, GIF, PNG).

IMPORTANT: *This mechanism allows you to display pictures in the report, without the need to store them in a database. You only need to store their file names in some text field in some database table.* ***The program also allows to display graphical data types (BMP, JPG, ICO, TIF, GIF, PNG) stored in the database as blob fields. Memo blobs are displayed as normal text.***

Sample report SFM file	Description
Report_07 - image files.sfm	It is shown how to use "Image file" and "Stretched image file" format for text data, to be interpreted and displayed as a name of a graphics file.

16. What is the "X / Blank" display format for the "Logical" type ?

It is presented in the previous ***"Report - display format.sfm"****example. The idea is, that for the "true" logical value the rectangle will be marked with a cross. It will of course not happen for the "false" value.*

17. What is the "Minutes as time" display format of integers ?

You can make the program to interpret an integer value, as the number of minutes of some time period. It will be displayed in the "h:m" format. For example the number ***"80"*** *(minutes) will be displayed as* ***"1h:20m"****.*

Sample report SFM file	Description
Report_08 - integers.sfm	Different display formats of integers are shown. The report also uses the technique of **script queries!**

18. What are the script queries ?

*This is a collection of a few queries (more strictly - SQL commands), so called **SQL script.** There can be more complicated situations, when the desired result set cannot be obtained as the result of a single SQL query. So you will need the SQL script. Only one of these queries will be the **working query,** with the result returning "select" statement. The remaining ones are the commands that help to prepare and then remove **temporary tables or views** necessary to execute the working query. These auxiliary commands must be ended here with **single semicolons.** In contrast, the working query must be ended with **double semicolon.** Here is some example of a script query:*

```
drop table "HELP" ;

create table "HELP" ( Id integer, Surname char(30), Name char(30),
January money, February money, March money, April money, May money, June money,
July money, August money, September money, October money, November money, December money,
Together money ) ;

insert into "HELP" select Id, Surname, Name, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 from "employee.db" ;

update "HELP" h set
h.January = ( select w.payment from "payment.db" w where w."month" = 1 and w.emp_id = h.Id ),
h.February = ( select w.payment from "payment.db" w where w."month" = 2 and w.emp_id = h.Id ),
h.March = ( select w.payment from "payment.db" w where w."month" = 3 and w.emp_id = h.Id ),
h.April = ( select w.payment from "payment.db" w where w."month" = 4 and w.emp_id = h.Id ),
h.May = ( select w.payment from "payment.db" w where w."month" = 5 and w.emp_id = h.Id ),
h.June = ( select w.payment from "payment.db" w where w."month" = 6 and w.emp_id = h.Id ),
h.July = ( select w.payment from "payment.db" w where w."month" = 7 and w.emp_id = h.Id ),
h.August = ( select w.payment from "payment.db" w where w."month" = 11 and w.emp_id = h.Id ),
h.September = ( select w.payment from "payment.db" w where w."month" = 8 and w.emp_id = h.Id ),
h.October = ( select w.payment from "payment.db" w where w."month" = 9 and w.emp_id = h.Id ),
h.November = ( select w.payment from "payment.db" w where w."month" = 10 and w.emp_id = h.Id ),
h.December = ( select w.payment from "payment.db" w where w."month" = 12 and w.emp_id = h.Id ) ;

update "HELP" set
Together = January + February + March + April + May + June
+ July + August + September + October + November + December ;

select * from "HELP" order by Surname, Name ;;

drop table "HELP" ;
```

There are 6 auxiliary queries here. First 5 of them precede the working query and prepare the temporary table "HELP". Then we have the working query ended with double semicolon (;;). It contains the "select" statement, that returns the desired result set. At the same end, there is the 6-th auxiliary query that removes the temporary table. ***IMPORTANT:*** *Have you noticed that the first auxiliary query is **the same as the last** one ? It removes just in case any "HELP" table, which will be created in the next query. It could happen that because of some reason it was not removed before. In such situation, the **"create table ..."** statement would not be executed and SQL error would be generated. **It is recommended to use the above scheme in the script queries.** It makes you to put the clearing instructions both at the beginning and at the end of a script.*

Sample report SFM file	Description
Report_09 - script query.sfm	It is shown how the above script query works in practice. Moreover, it is demonstrated how to display sums of an internal query in the report.

Because REPCoder has now the built-in script language (similar to C), it is possible to create (and remove) temporary data in this way (so-called "Calculated fields"). This script language provides a much stronger mechanism and is used more often than script queries described here.

19. In what sequence are the commands of a script query executed ?

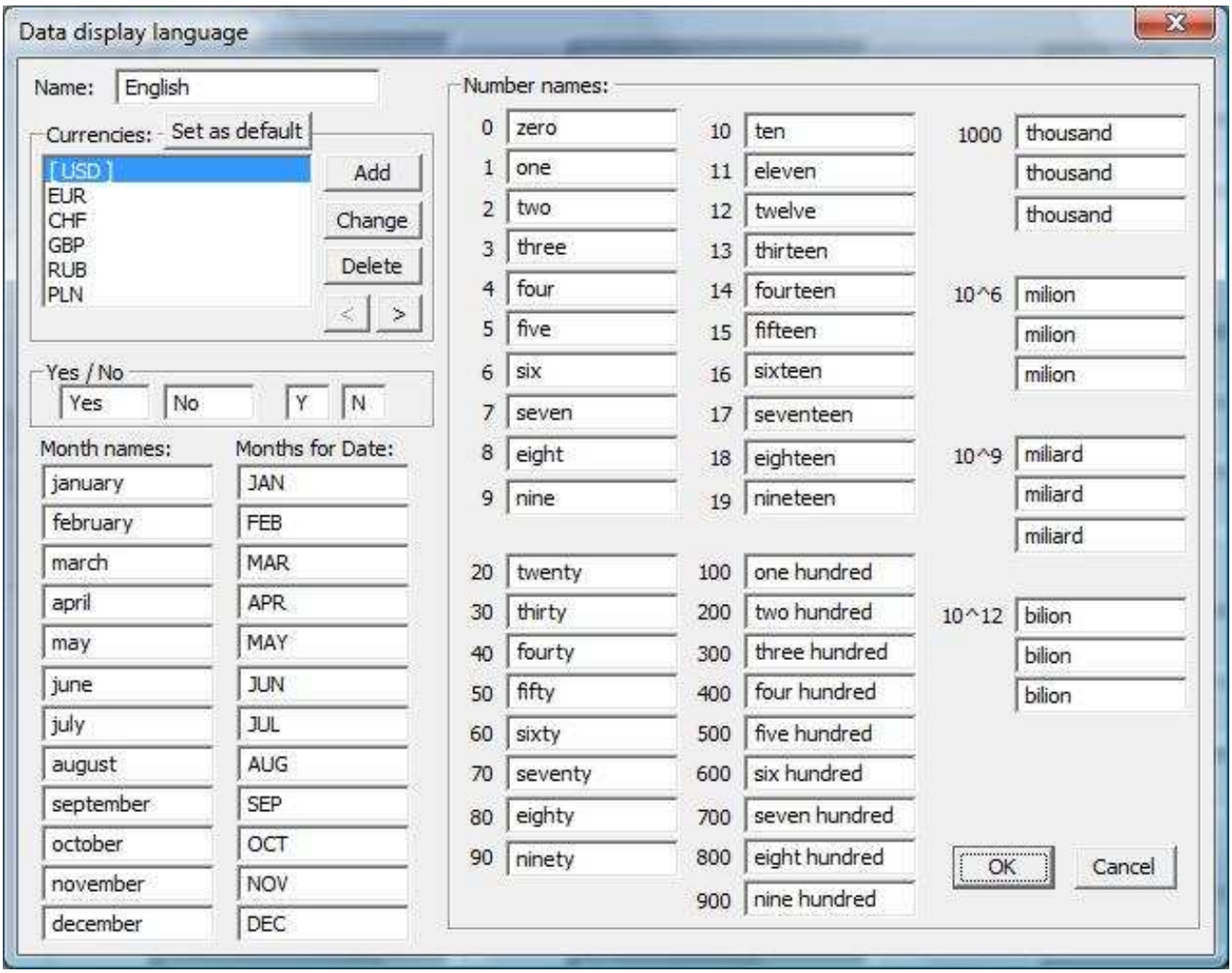
Script queries *are executed in a specific way while the report is executed. After you switch to the "Working Mode", all the commands preceding the working one are immediately executed. In contrast, the commands that follow it, are executed only* ***at the same end,*** *after we leave the "Working Mode" (this is the case of the external query), or eventually after we go to a different external record (this is the case of internal queries). It implies, that in the case of a complex report, we can prepare auxiliary tables (or views) in the external query and then use them in texts of internal queries. You don't need to be afraid, that they will be removed to early. So the internal queries can use temporary data created in the external query. Moreover they can of course also create their own temporary data. Remember to remove temporary data at the beginning and at the end of a script.*

20. What is a display language of data ?

All types of data except those of the "Text" type, are displayed according to some ***display language.*** *There are 2 built-in display languages in our program. They are* ***english*** *and* ***polish.*** *Moreover, users can define their own display languages and use them in reports. It can be emphasized here, that display language (together with the display format) can be configured individually for each database text in the "Texts" dialog. So it is for example possible to display money using different currencies. Similarly, some dates can be displayed in english, polish or german languages. The default display language of a report we set in its "Properties" in the "Form" menu.*

21. How to define a new display language ?

We do it also in the report properties ("Properties" in the "Form" menu). To define a new display language, we have to fill the data that will enable us to display "in words" numbers, dates and logical values. We also choose the ***default currency.*** *Moreover we can define* ***foreign currencies*** *in which we want to display money.*



Sample report SFM file	Description
Report-10 - display languages.sfm	How to display data using different display languages. How to translate money to foreign currencies.

IMPORTANT: ***Display languages*** *are defined for each report (in report Properties). You can use "Copy" and "Paste" buttons to transfer them between different reports.*

Now we go to important problem ...

Sums of Internal Queries

Internal queries defined in our report can return result sets, with some numeric columns that can be ***summed.*** *They represent numeric data types. The program can sum them up and then display the sums as ordinary data.* ***These sums are calculated by the program independently, without calling SQL sum aggregate function.***

22. Couldn't the problem be solved using additional SQL queries with aggregate expressions to return the needed sums ?

It is of course one of possible solutions. And it will be good enough in some simple situations. But you cannot obtain this way some important effects like displaying ***subtotals*** *between the sorted records of a result set. The standard SQL language does not give you such possibilities. Similarly you will never obtain with SQL, the* ***sums on page*** *statistics, with the sums on all preceding pages. Our program* ***"helps"*** *the SQL server and makes it possible to obtain various important sums in our report. Moreover they are calculated without additional SQL queries. So it does not require additional server job to execute a report with a complicated summing scheme. The sums are accumulated by REPCoder itself while reading data from SQL results.*

23. We have 3 types of record sums:

sum of all records - in groups: "No", "User", "Table Header", "Page Header", "Page Footer".	*We want to display sums of some columns below all displayed records of internal query's result.*
subtotals of a sorted result set - in groups: "Sum Header", "Group Sum".	*We want to display subtotals between sorted records, after the ones, where the values of some columns (used in "order by" clause) are changed. We can have* ***many orders of sums*** *- independently for each such column.*
sum of records on page - in groups: "Page Header", "Page Footer".	*It is a well known problem in accounting reports, that such sums on pages are required. You can also display the sums of all preceding pages and together with the current page. These sums can only be placed in "Page Header" and "Page Footer" groups.*

Sample report SFM file	Description
Report_11 - subtotals.sfm	It is shown how to use "Sum Header" and "Group Sum" groups to display subtotals of a sorted query's result set. You will also see how it works, when the records of an internal query are expanded on page horizontally.
Report_12 - sums on pages.sfm	It is shown how to display sums of records on individual pages of a report - we use the "Page Footer" group. You can display numeric types and also the number of records statistics.

IMPORTANT: *In the program configuration ("Options" in the "File" menu) you can find the option* ***"Case sensitive group sums".*** *This problem is treated in different ways by different SQL servers. Because our program calculates the sums independently itself, you can decide how to proceed. For example if to treet the surnames "SMITH" and "Smith" as equal or different.*

24. What is the <SUM OF RECTANGLE> ?

While we are already talking about sums, let's add that there is a simple mechanism in the program that allows to sum up data of the same type displayed in ***the same rectangle.*** *So we do not sum records here as in previous cases. We can sum independently 4 types of data:* ***short integers, long integers, real numbers, money.*** *This mechanism is not however very important. This effect can also easily be obtained by extending the text of the SQL query. But it is worth to know that such a simple mechanism exists in the program.*

Sample report SFM file	Description
Report_13 - sum of rectangle.sfm	It is shown how to display sum of integers, real numbers and money, that are displayed in the same rectangle, without the need to extend the text of SQL query.

another important and useful mechanism ...
Parameters of the report

- without these parameters good reports couldn't exist

25. IMPORTANT: You have to distinguish previously described parameters of SQL queries in complex reports, from parameters of the report !!!

You should understand the difference between these two different concepts. Parametrized SQL queries are generally known. It was introduced in the definition of ***SQL standard.*** *In contrast,* ***parameters of the report*** *are another important mechanism in our program. Let us mention here, that they can also enter the texts of SQL queries, but the rules are completely different. They are replaced by their values* ***at the same beginning*** *of the report execution, before any other actions take place.*

26. What is the destination of the parameters ?

They have very important and reach applications. Generally, ***they allow to parametrize almost all texts entered into the report form.*** *It is difficult to list all applications of the parameters. This mechanism is available to both developers and end users of the reports. It depends on inventiveness of the report developer, how it will be used. To make the problem more clear, let us mention that* ***the values of parameters can be changed in 3 different ways:***

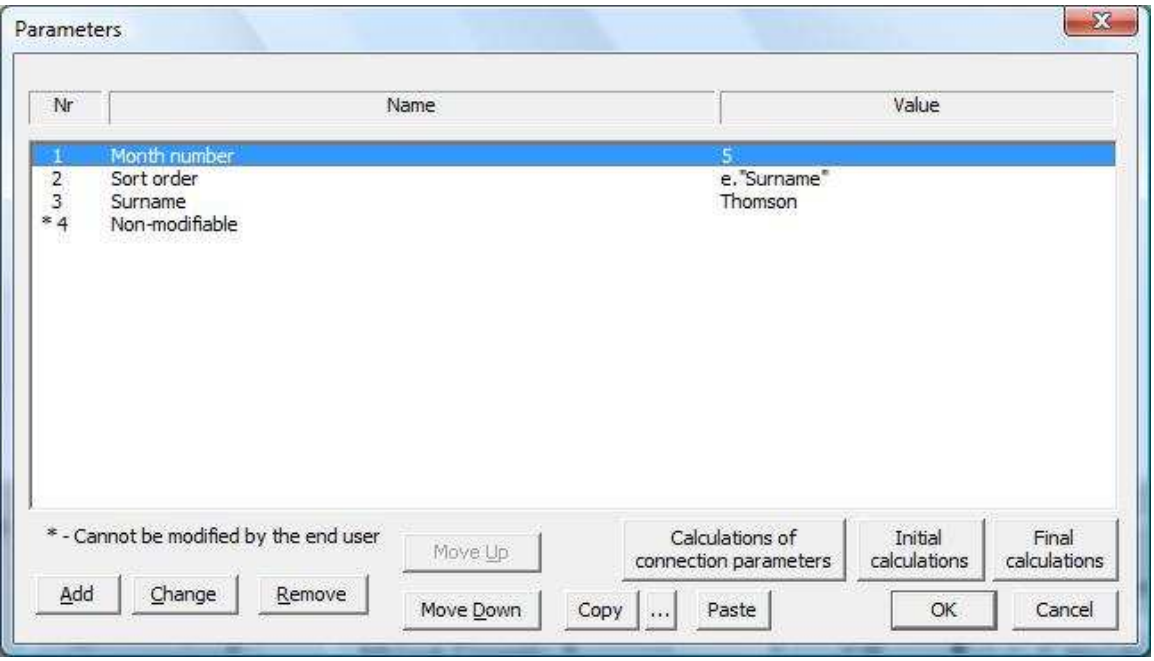
1. **By report developers during design work.**
2. **By the calling application** *(using "REPCODER.DLL" API functions).*
3. **By end users in the working mode of executed reports.**

27. How we design parameters and how we reference them in our form ?

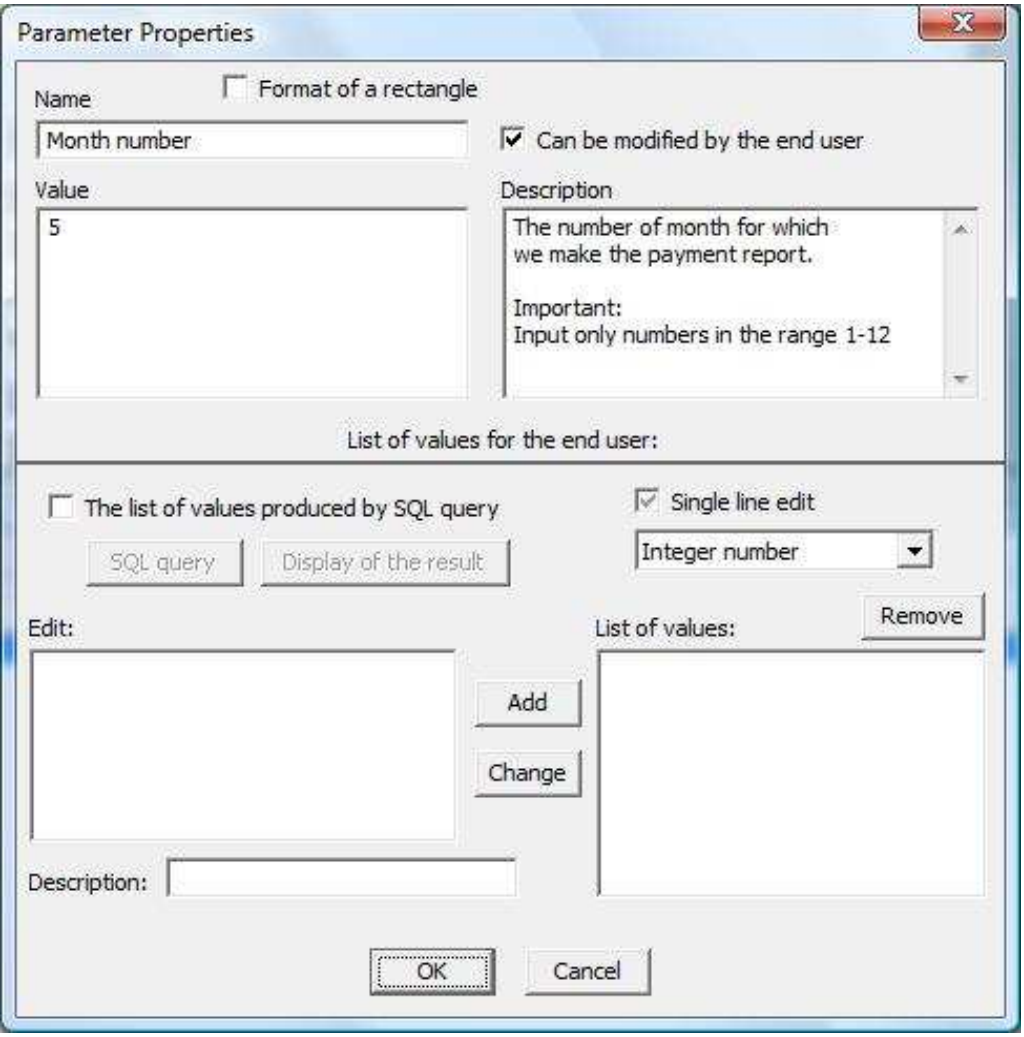
We must understand it exactly while working with parameters. It can be best explained using example. Suppose we want to create a parameter ***named*** *"Month number". Then we want to reference it in the text of SQL query. Together with the name, each parameter also has its unique* ***ordinal number*** *and the most important property - its* ***value.*** *We should emphasize here, that the parameter's number is much more important than its name. The role of the name is only informal (parameter also has its* ***description****). The parameter's number (together with the* ***%*** *character), work as the parameter's* ***identifier*** *in the program. Some SQL query can have for example the following form:*

select * from "PAYMENT" where "Month" = **%1**

You can see a reference to the parameter 1 here. We will always reference parameters in this way during our design work. To design the parameters itself, we call the appropriate dialog - "Parameters" item in the "Form" menu. You can also use the ***CTRL+E*** *shortcut keys.*



In this dialog we design and enter parameters for our report (as many as we need) - "Add", "Change" and "Remove" buttons. "Move Up", "Move Down" buttons allow to change the relative order of the parameters together with their important ordinal numbers. The "Initial calculations" allow us to calculate the initial values of some parameters using the powerfull mechanism of REPCoder scripts (the working function is SETPARAM). This mechanism is shortly described later in this guide. The * character before the parameter's number denotes a non-modifiable parameter, so it cannot be changed by the end user (see the description below). This dialog will also be used by the end user in the Working Mode, to modify the parameter values. But the non-modifiable parameters will no longer be listed here.



In this dialog we set the properties of the given parameter. So we design it just here. The basic properties are **name, value, description.** The description is optional. The value we enter must be a valid one. It means that it should agree with the parameter's destination. Otherwise there will be problems during the execution of our report. Another very important property is **"Can be modified by the end user".** It makes it possible for end users to change the parameter's value in the Working Mode. We may often not want to allow this. In the lower dialog part (below the black horizontal line) we can optionally define the **list of values,** that will be accessible to choose from by the end user. Moreover we can also (optionally) define description for each such a value in the list. If we want the list of values to be **built dynamically** in the working user mode, as a result of some **SQL query,** we check this option and build the query (SQL query button). We can also independently give the user possibility to edit the value manually - **"Can be edited by the end user".** The called edit window can be **multi-line** (default) or **single-line** with a format you need.

28. Where inside our project form, can we enter references to the parameters ?

*We have earlier mentioned, that the parameters can be referenced in any edit window in the program (everywhere where we edit some text). So we can place the reference **%n**, and the appropriate text can be defined as the value of the parameter n (this is its ordinal number). But there are situations where it makes no sense to do it. For example you cannot reference parameters in the program "Options" dialog, or where you define display languages and foreign currencies. Everyone can intuitively guess where it makes no sense. However to avoid problems, we will list below all the places (texts) in the program, where it is possible to enter references (%n) to the parameters of our report. These texts are:*

1	**"Normal" texts inside the rectangles.**	There are no limitations. You can use them for example in titles, subtitles, page header or footer.
2	**SQL queries texts - internal and external.**	No limitations. You can make the parameter even the whole query's text, or its some part. For example it can be the "where..." or "order by..." clause. But in most situations parameters used here represent more precise texts.
3	**SQL query text in the "Parameter Properties" dialog (see previous page).**	This is just the text of the SQL query, that returns the dynamic list of the parameter's values for the end user. There is one natural limitation here. You cannot use reference to the parameter that we just define here.
4	**The name of a database in the "Databases" dialog.**	Database names are very often parametrized this way.
5	**User name, password, role when we define database connections.**	This way you can place user name, password or role in the report's parameters.
6	**The text coupled to page numbering in the "Print Options" dialog.**	There are no limitations.

IMPORTANT:

- If the end user has changed some parameter's value, it means that the report has changed. Thus before closing it, program asks if you want to save the changes.

- The parameter values are text variables. When we enter some value (text, date, number, ...), we must keep in mind, that it will be treated by the program as an ordinary text. The references %n will be replaced with text constants exactly equal to that value. So it is up to the designer, to place the references in such a way and environment (apostrophes, brackets, ...), that they will be properly interpreted during the report's execution. Thus for example, you should be carefull how to place the apostrophes - in the text of a query or in the parameter's value.

Assume for example, that we want our SQL query to return exactly a single record of the "employee" table. We want the personal surname to be the report's parameter. We can realize it in two ways:

I: *query: select * from "employee.db" where Surname = %1* *parameter 1 value: 'Robinson'*	*the apostrophes are in the parameter's value, thus they are not in the query*
II: *query: select * from "employee.db" where Surname = '%1'* *parameter 1 value: Robinson*	*the apostrophes are in the query, thus they are not in the parameter's value*

Some other correct samples:

query: ***select * from "payment" where "Month" between %1 and %2***   *parameter 1 value:* ***7***, *parameter 2 value:* ***9***
query: ***select * from "employee" where Birth_date between '%1' and %2***   *parameter 1 value:* ***01.01.1970***, *parameter 2 value:* ***'31.12.1970'***

29. What we do first - design (define) the parameter, or enter references to it ?

Important quetion. We must keep it in mind. When we define SQL queries or database connections, the program tries to execute them before they are accepted. In the case of error the changes will not be applied. Thus if we try to reference a non existing parameter here, it will not be accepted by the program. So you should remember, that ***first we create a parameter, then we reference to it in our report.***

30. Important remark for programmers who will be using the program through DLL calls ...

If there are modifiable parameters in the report, and when you call it using "REPCODER.DLL", the "Form Parameters" dialog will be opened at the beginning. This way the user has the possibility to inspect the parameter values and to modify them if necessary, before the report is executed. He can of course modify these values also later, after the report's execution - in the Working Mode.

31. Time for example ...

Sample report SFM file	Description
Report_14 - parameters.sfm	It is demonstrated how to use parameters in a report.
Report_15 - parameters 2.sfm	It shows how to make use of the complex parameters, that have more than one field. It is also presented how to design a grid for such a parameter.
Report_16 - display control of groups.sfm	How to use a parameter that controls the display of a group.

Important: *We can also use in the program* ***%d*** *and* ***%t*** *symbols, that will be then after report execution expanded into the system* ***date*** *and* ***time*** *respectively. It is demonstrated on the page 2 of the 14'th example. Moreover the symbol* ***%D*** *will be expanded into the name of the* ***current directory.***

32. What about BLOBS ?

Blob fields (MEMO, pictures) can also be displayed in our reports.

Sample report SFM file	Description
Report_17 - blobs.sfm	It is demonstrated how to display BLOB fields. The pictures from a database can be exported to BMP files here. The "Calculated fields" mechanism (described later in this guide) is used to do it.
Report_18 - blobs BDE.sfm	It is demonstrated how to display BLOB fields in a paradox table.
Report_19 - blobs transfer to ODBC.sfm	Records with BLOB fields are coppied from a Firebird to some MS Access database using ODBC. The "Calculated fields" mechanism (described later in this guide) is also used.
Report_24 - long memo.sfm	How to display very long MEMO fields and divide it into many pages.

33. What about UTF-8 ?

Sample report SFM file	Description
Report_20 - utf8.sfm	A sample Firebird 2.5 database is used with UTF-8 default character set. You must have installed Firebird 2.5 server (or later) to execute this report.

after we have designed and executed report ...

Options accessible in the "Working Mode"

*So far we have discussed all the topics related to the designing of the reports. They are functions in the **"Design Mode".** Now we will focus our attention on the **"Working Mode",** that we switch to after the report's execution (F5 function key). Before that, let us mention one important technical remark about the process of execution itself:*

33. The report's execution is performed in the background.

Reading database data and building pages for some reports can be a long time operation. It depends on the complexity of SQL queries and amount of data processed. Thus this operation program performs in the background, using a seperate thread. So you can decrease the size or minimize the application window. The small dialog in the upper-left corner informs you about the progress of the work. It contains the "Cancel" button that allows you to break the process of data reading or pages building. This way you will obtain a partial, broken report. ***If there are no SQL queries in the report (for example just like in this guide), it is not possible to switch to the "Working Mode" (F5 key does not work).***

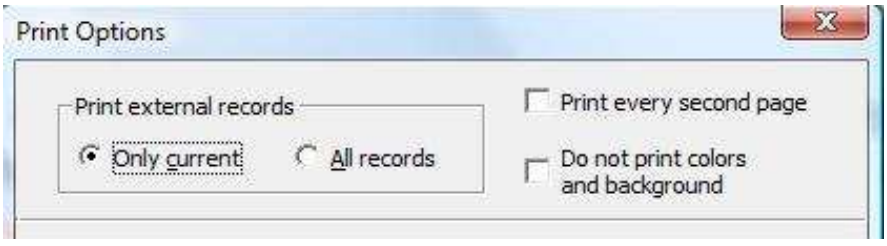
34. What the end user can do with executed report ?

The end user can perform the following actions in the "Working Mode":

1. **Print the report.**
2. **Save the result to disk - visual archives (SFR files).**
3. **Export to file -** saving results of SQL queries into text files (also in CSV format).
4. **Built-in Export -** designed earlier export to file or other database.
5. **Modification of data -** possibility to change database data from the executed report.
6. **Copy to clipboard -** as bitmaps or enhanced metafiles.

35. What is important to remember about printing ?

*Printing was already partially described in the "Graphics" guide. It was discussed the problem of **page numbering** there. We open the "Print Options" dialog using the "Print" item in the "File" menu. When we print a report we can moreover decide to print **"All records"** or **"Only current".** It of course only applies to **external records.** If our report does not have the external query, these options are disabled. When we decide to print "All records", the program performs a hidden "walk" on the entire external query's result set. It executes report for each external record and merges the results into a single virtual report. After that, the system printing dialog is called and you can print the entire merged virtual report. We can also independently choose the option to **"Print every second page"** only. It can be applied in situations, when you want to use both sides of the paper. First you can print odd pages, then reverse the papers in the printer and print the even pages. The report can be designed using colors for texts and background (it looks better on the screen). If you however want to print it in black and white without any background (to save ink in the printer) you can choose the option **"Do not print colors and background".** Here is the corresponding fragment of the "Print Options" dialog.*



IMPORTANT: *Merging the reports for all external records (bulding the virtual report) can be a time consuming process. Thus this operation is also performed **in the background.** The user has the possibility to break this process.*

36. What are the visual archives ?

An executed report, that contains database data, can be saved to disk. This result is strongly compressed (by ZLIB) during this operation. Before saving, program proposes you to ***describe the result.*** *It is intended to be a short text or comment. Moreover, in the case of external query, the user is also asked if all the external records should be saved, or only the current one. These so called* ***visual archives*** *can be then studied in the future and printed (all or selected pages). They have the* ***SFR*** *extension and can be very large. Instead of printing a huge number of report pages and storing them in paper form, you can use this possibility. It is faster, cheaper and more reliable. During our own tests, a sample report containing 15 000 pages was saved to SFR result of only 4MB size. If you later want to open such a big file, it all works perfectly, without any delay. The program buffers the pages in compressed packages as they are needed.*

IMPORTANT: *The option* ***"Save result"*** *(an item in the "File" menu)* ***is only accessible in the "Working Mode".*** *Thus we cannot find it in this guide, which contains no SQL queries and cannot be switched to the "Working Mode". You can however test creating (and then opening) visual archives, using any* ***sample report.***

IMPORTANT: *The program allows to open two types of files. They are:* ***"Projects (*.sfm),*** *and* ***"Results (*.sfr)".*** *You can try it using the "Open" item in the "File" menu. When you open an SFR archive file, the program works as an ordinary viewer. Thus the number of accessible options is smaller than in the case of SFM files. You have also no possibility to make any changes to SFR files.*

Sample report SFM file	Description
any sfm file ...	Open and execute any of previously described sample reports. Then save the result to disk (create a visual archive). Then open the newly created SFR file. While viewing it, find its description in the "Properties" item (in the "Form" menu).

37. How to design a "modifiable" report ?

The program offers a smart possibility to design some reports in such a way, that after the execution the user will be able to ***modify database data.*** *First you need to check, that your database connections should be opened in the* ***READ-WRITE mode.*** *Otherwise, they are opened in* ***READ-ONLY mode*** *by defalut. You check it in the "Database" dialog (an item in the "Form" menu).*

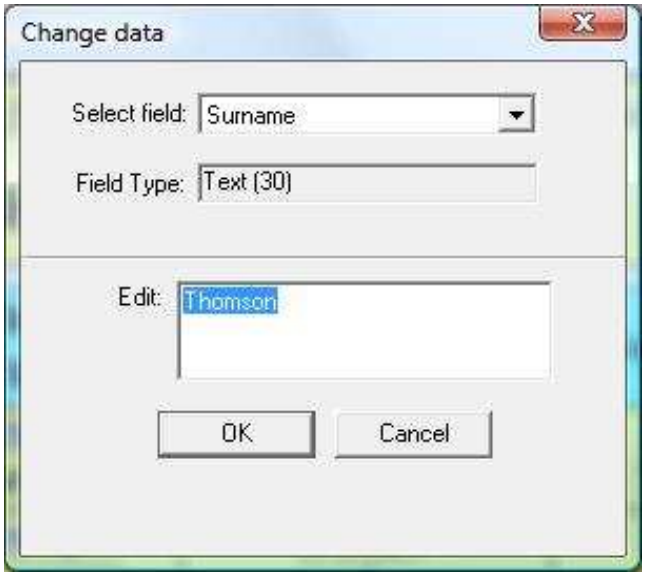
38. What tables in our databases can be updated this way ?

We can modify the table associated with the ***external query*** *(for static or complex reports only) and also tables returned as the results of* ***internal queries.***

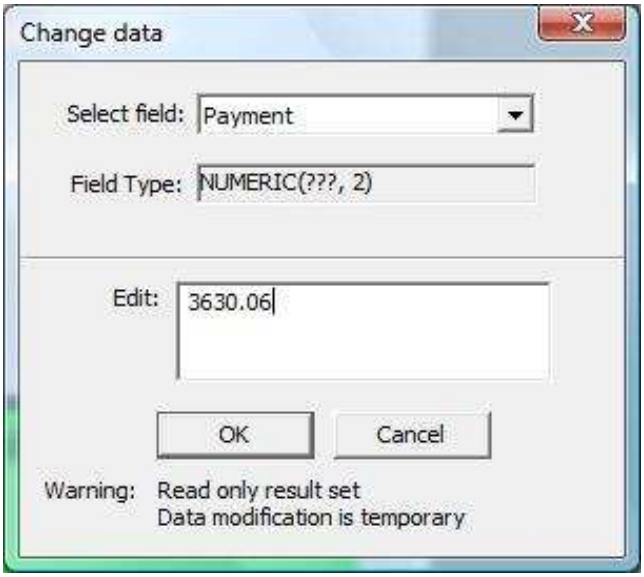
If you want to update external table, you should also check the option ***"Updatable result set",*** *just above the query's text in the dialog were you enter it. The same applies to updatable internal tables.*

39. Is it already enough ?

Absolutely not. It depends also on the type of the underlying SQL query and the ***database server.*** *The query itself must be* ***"updatable"*** *in terms of database theory and SQL language. So it has to return so called* ***"alive result set".*** *The program itself will however allow you to edit new data. It means that it will be possible to call the* ***"Change data"*** *dialog. We open it in the "Working Mode" only by* ***double-click*** *on the rectangle, where you want the data to be changed. This option is not present in the program menu, because there is no "Selected" rectangle in the "Working Mode". Thus we select it using only the double-click. In the example on the right hand side, the changes will be saved to a database. Apparently the query returns updatable result set.*



As it was just mentioned, the program allows you to open this dialog (using double-click), always when defining the SQL query, the "Updatable result set" option was checked. If however the query is so complicated, that the database server returns "read only" result set, the program will allow to open the "Change data" dialog. But a warning message at the bottom is displayed, informing that the modification is only ***temporary.*** *The entered changes will not be put to the database. Does it make any sense to do it ? It makes sense only if you want to print or save result (SFR), where these temporary modifications will be fixed. It can however lead to* ***fabricated*** *reports.* ***Thus the responsibility for enabling the end user this mechanism relies on the person who designed the report.***



IMPORTANT: *It can be forced, that the user will be able to change the data* ***only temporarily*** *(without saving to the database), even in the case when the SQL query returns the updatable result (so when it is allowed by the server). You only need* ***not to check*** *in the "Database" dialog, the option* ***"Open databases in read-write mode".***

40. Is it possible to modify (even temporarily) all data in the report ?

Not all of them. It is possible to modify only database fields that belong to ***updatable SQL query's result.*** *It means that it is not possible to modify all the* ***sums*** *- subtotals, on pages, totals, sums of rectangle. Thus the double-click will always have no effect for rectangles belonging to the groups "Sum Header" or "Group Sum". It will be working for "Internal Table" group if and only if the designer had checked the "Updatable result set" option while defining the appropriate internal query. In the case of the other groups ("No", "Page Header", "Page Footer", "User"), it depends on the same option checked or not, but for the external query.*

41. Are the appropriate sums also updated after we change the data in our report ?

Yes. The mechanism of the updates is ***smart*** *enough, to interfere all the sums that could be affected by the updated data. So the user does not have to worry about the* ***integrity of data with their sums.*** *The update of sums is* ***automatic and immediate.***

Sample report SFM file	Description
Report_21 - data modification.sfm	An example of a "modifiable" report, that allows the user to change database data after its execution. Independently, it is also demonstrated how to use "Page Header" and "Page Footer" groups.

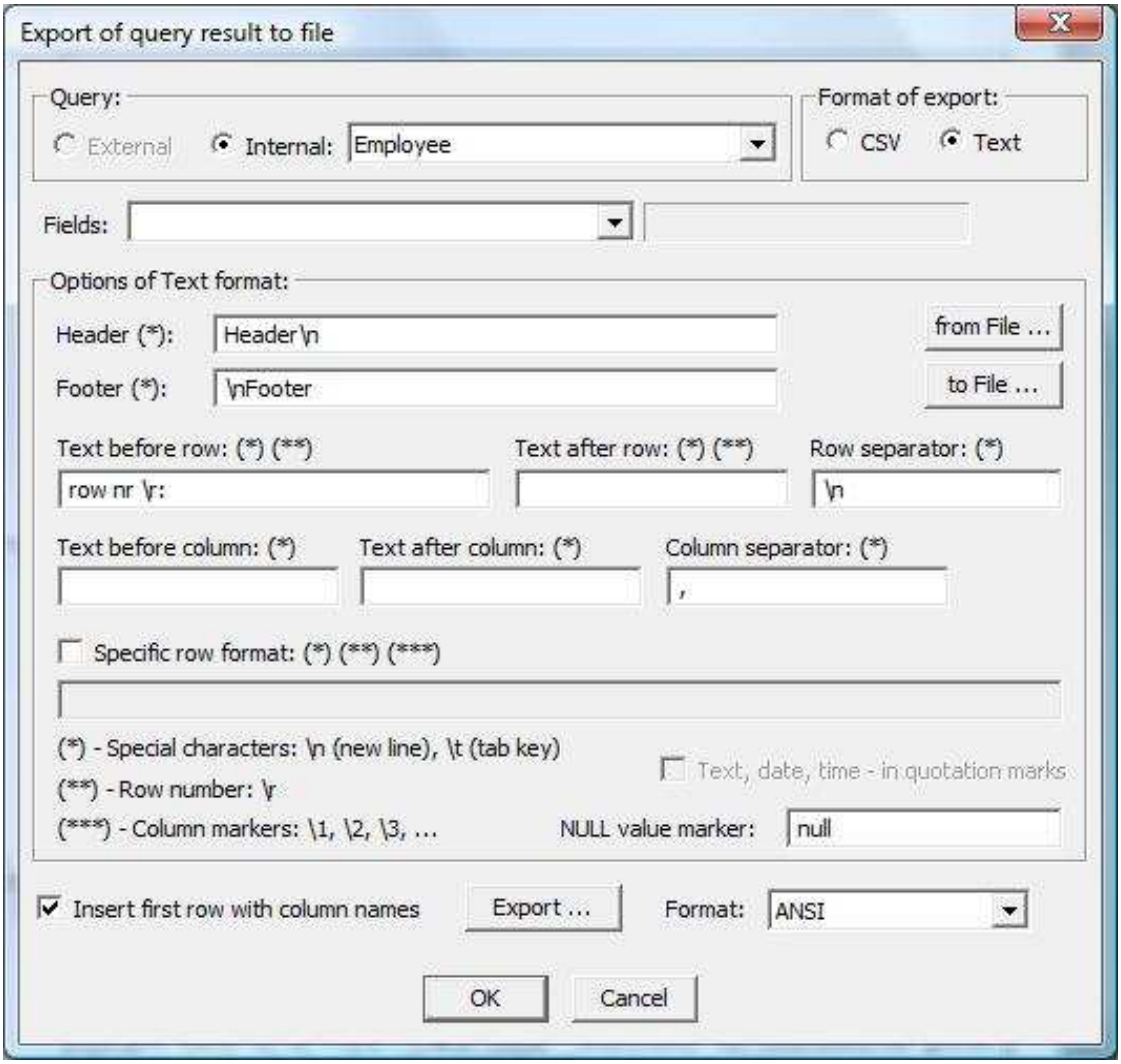
42. What are the calculated fields in REPCoder ?

Calculated fields extend our SQL queries. It is a powerful technique that allows us to define additional fields for each SQL query. REPCoder will calculate their values for each record of the result set as functions of fields defined in the query. Of course there exists in SQL language a possibility to write arithmetic formulas in texts of "select" statements, but it has very limited power in practical use. ***Moreover many SQL servers do not implement yet the case keyword in the "select" statement, although it is a part of SQL standard.*** *The mechanism of calculated fields in REPCoder is the solution and enables to define* ***very advanced algorithms*** *in the form of texts of* ***built-in script language.*** *It is also possible to define common* ***global variables*** *and* ***functions,*** *that will be accessible in scripts defining calculated fields for our report. The field definitions are not limited to single mathematical formulas. The script can also define* ***its local variables,*** *that can be used in the final formulas. The power of the built-in script language relies also on the fact that it allows to write* ***conditional expressions*** *and* ***loops*** *(keywords: IF, ELSEIF, ELSE, WHILE, BREAK, CONTINUE). You can find out more about REPCoder scripts in the sample below.* ***It is just a regular, programming language, similar to C.*** ***The dialog where we write the scripts offers its own, built-in "Help".***

Sample report SFM file	Description
Report_22 - calculated fields.sfm	How to define calculated fields for our SQL queries. You can also obtain this way the effect of "conditional display format" of data.
Report_23 - data transfer.sfm	How to use REPCoder script language (calculated fields) to transfer data between databases, making conversion if necessary.

43. How to export the results of SQL queries to text files ?

*Unlike visual archives, this option allows us only to **export SQL results** to text files. You can export results of our External Query or Internal Queries. The format is flexible enough and can be designed by the user. The export dialog is available only in the "Working Mode" and looks like this:*



*After you select a query, the list of its column names appears in the combobox together with their database types. It is provided only for information and helps us to design "Specific row format". The data can be exported in the built-in **CSV** format or arbitrary **text format.** For the text format we edit the options like header, footer, text preceding and following rows, row separator. The same for columns. You can use special characters here: **\n** - new line, **\t** - tab key, **\r** - row number. There is also a possibility to specify the row format more precisely. You only need to check **"Specific row format".** You define the exact format of the row in the edit window below the checkbox. You have to use the markers **\1, \2, \3, ...** representing consecutive columns. In both cases we also define the marker for the **NULL value. The display format of data is the one defined in the program "Options".** What about CSV format ? It is not too much flexible. The rows are always separated by new line characters, while the columns - by comma (,) characters. The columns of text type are placed in the quotation marks. Numbers and money are displayed simply without Windows formatting, using decimal dot. Only the date and time formats are the ones defined in program "Options". For both CSV and arbitrary text format we decide if the column names should be inserted at the beginning of the output file. For those we check **"Insert first row with column names".** We export data using the "Export" button. The format can be: ANSI, UTF-8, UTF-8 (+BOM), Unicode, Unicode reversed.*

Can we remember the defined export formats ?

*Yes. Any format designed in this dialog can be saved for future use also in a text file. You should use the button **"to File".** It can be later loaded using the **"from File"** button. By default REPCoder is searching for these format files in the current directory. After you close the dialog with the "OK" button, the defined format is automatically saved in the **"export.txt"** default file located in the current directory. This file is loaded every time you open the dialog. When you close with the "Cancel" button, the file is not changed.*

The possibility to export results of SQL queries to text files has very important practical applications. It enables us to use this tool to transfer data between different databases or computer systems. But now, a more advanced and precise method to do it is to use "Calculated fields".

IMPORTANT: *The described here export to file is used by the end user after the execution of the report. Therefore he must have some knowledge to configure the export. To avoid this problem you can define inside your report so called* ***built-in export.*** *It is done by the report developer, and the end user needs only to press a button. The built-in export can be defined for SQL queries (external and internal) similarly as calculated fields. This mechanism allows not only to export to files, but also to inject data directly to some other database. In this situation you need to write correct SQL queries (like "insert into ...") that will perform that job. The example of how to use built-in export is in the sample:* ***"Report - simple dynamic.sfm".*** *But now the "Calculated fields" (based on the REPCoder script language) provide much better mechanism to export data (to file or another database).*

44. Copying result pages to clipboard.

This option allows to copy the current page of the executed report or visual archive (sfr file) to clipboard for the use of other applications. Our program can do it in 2 formats: ***bitmap*** *or* ***enhanced metafile.*** *The size of the bitmap depends on the size of the page (you can change it with Ctrl+I, Ctrl+O). The quality of the bitmap is however limited. But the picture in the metafile format does not depend on the page size and its quality* ***is perfect.*** *So this is the recommended format. The report page coppied to clipboard (metafile or bitmap) can be easily pasted to other applications or text processors (for example "Microsoft Word").*

45. New advanced features.

REPCoder allows you to display very long texts from databases. This text data are usually placed in MEMO fields and cannot fit on a single page. So they must be divided into a few pages. This smart mechanism is shown in the "Report_24 - long memo.sfm" sample. Another useful new feature is the possibility to perform the script language calculations for aggregate records. This way you can for example calculate STDDEV statistics of your data in the report. It is shown in the "Report_25 - statistics.sfm" sample.

Sample report SFM file	Description
Report_24 - long memo.sfm	How to display very long MEMO texts from a database.
Report_25 - statistics.sfm	What is the ROW_TYPE predefined variable and how to use special input and output fields in the script language to calculate some statistics in aggregation records. What is the meaning of $SUM_, $MIN_, $MAX_ ? How to use $$$ special output fields ? How to calculate STDDEV statistics ?

Guide 4 - DLL

This guide is for programmers, who want to call reports in their own applications. The exported functions of REPCODER.DLL are listed and described here. The source files you will need are: REPCODER.H and REPCODER.C. You can alternatively use: REPCODER.H with REPCODER_MS.LIB (for MS VC++) or REPCODER_BOR.LIB (for Borland C++). These important files are located in the DevTools subdirectory. You can also find here the source codes of sample programs. When you create 64-bit database applications, just use REPCODER64.DLL. It was tested with 64-bit FBCLIENT.DLL, Interbase XE IBCLIENT64.DLL and some 64-bit ODBC drivers.

REPCoder has also API for C#, Java (for Windows only), Delphi.

REPCoder (EXE and DLL) is a UNICODE application. *So are all the functions of REPCODER.DLL. They use* ***wchar_t*** *type for string arguments. But these functions have also their* ***ANSI*** *versions, based on* ***char*** *type. So REPCoder can be used both by* ***UNICODE*** *and* ***ANSI*** *applications. The ANSI versions have the* ***_a*** *ending at their names.*

REPCoder API *is a very simple to learn and use programming interface. There is in principle only one working function there:* ***repc_open_report.*** *It allows you to call a report from your application. You will also need* ***repc_init*** *at the same beginning.The functions are all standard* ***C language*** *functions. Their names begin with* ***repc_*** *prefix. They are listed below:*

repc_open_report ***_a***	*The main function of the library. It opens a report with different options, parameters and user privileges.*
repc_call_report ***_a***	*The simplest way to call a report.*
repc_call_report_param ***_a***	*The simplest way to call a report. It allows you also to set the parameters of the report.*
repc_explorer ***_a*** ***repc_explorer_param*** ***_a***	*A dialog called by your application, that allows you to manage reports (open, change, remove). The second function allows to set report parameters.*
repc_get_report_desc_len ***_a***	*Returns the length of the report description in characters.*
repc_get_report_desc ***_a***	*Retrieves the description of a report into a text buffer.*
repc_set_bde_netdir ***_a***	*Allows to set the "NETDIR" BDE session property for paradox tables. Only in REPCoder 32-bit.*
repc_get_demo	*Returns 1 if REPCoder works in DEMO mode.*
repc_set_demo	*Switches on or off DEMO mode for REPCoder.*
repc_set_logo ***_a***	*Sets the logo text in the bottom-right corner of each report page.*
repc_get_open_count	*Returns the number of non-modal dialog windows of reports, opened with REPCoder by the current process.*
repc_can_exit	*Informs you if your application can be safely closed (if REPCoder perfoms no background operations).*
repc_set_fbclient_thread_safe	*These functions are now obsolete.*
repc_set_user, repc_set_pass ***_a***	*Sets username and password that REPCoder uses while connecting to databases.*
repc_set_role ***_a*** ***repc_set_charset*** ***_a***	*Sets role and character_set that REPCoder is using while connecting to Interbase/Firebird databases.*
repc_set_firebird_critical_section ***repc_set_interbase_critical_section***	*Allow to set the address of a critical section that your application is using when calling Interbase API functions. REPCoder will use it instead of its own default critical section.*
repc_init, repc_exit	*Called at the beginning and at the end of the work with REPCoder.*
repc_set_dialog_fonts	*Allows to set normal or large fonts in all dialog windows of the user interface (except Windows custom dialogs).*
repc_alloc_report_desc ***_a*** ***repc_free_memory***	*Returns the pointer to the allocated buffer with the report description. It must be later freed with repc_free_memory by the calling program.*
repc_create_result ***_a*** ***repc_create_result_param*** ***_a***	*Executes a report and creates its result (as SFR file) without opening any windows or messages. So it is a completely silent function.*
repc_admin_setup ***_a***	*Integrates REPCODER.DLL with the* ***REPCoder Admin*** *program (reporting server). It enables you to directly call the reports that are stored in a remote database (Firebird or InterBase) as blobs.*

1. The main function - *repc_open_report*

int WINAPI ***repc_open_report (***	Returns: ***1*** (OK), ***0*** (Error - invalid file name).
HWND ***hWndParent,***	*Parent window handle.*
BOOL ***bModal,***	*The dialog window type you want to open:* ***1*** *(modal),* ***0*** *(non-modal).*
wchar_t* ***FormName,***	*The report file name (SFM or SFR).*
int ***RecordSeqNr,***	*The initial external record number for the report (1,2,3,...).*
int ***nParams,***	*The number of report's parameters we want to set.*
int* ***pParamsNumbers,***	*The array of numbers of these parameters (1,2,3,...).*
wchar_t ***pParamsValues,***	*The array of texts with the values of these parameters.*
BOOL ***bParamsUpdate,***	*User privilage - possibility to change the report's parameters.*
BOOL ***bRecNavigate,***	*User privilage - possibility to change the external record.*
BOOL ***bDesignMode,***	*The openning mode for the report:* ***1*** *(Design Mode),* ***0*** *(Working Mode).*
BOOL ***bAllowDesign,***	*User privilage - possibility to swith to "Design Mode" (to make changes)*
BOOL ***bAllowUpdate);***	*User privilage - possibility to modify database data from executed report.*

Important: *The underlined parameters are* ***optional.*** *It means that you can use the* ***0*** *value instead. In such situations, default settings (designed inside the report's form) will be applied. The blue descriptions in the table above need some comments:*

- What is RecordSeqNr ?

If the report has the external query, we set the ***record number*** *for which we want initially to position our report - 1, 2, 3, ... If we do not set this argument (use the 0 value), the default* ***1*** *value will be applied. The report will be executed for the first record of the external query's result set.*

- How do we set the parameters of our report ?

We use the arguments ***nParams, pParamNumbers, pParamValues*** *of* ***repc_open_report*** *function. Here we have an sample code, that sets the parameters 1, 2, 3 for our sample "Report - parameters.sfm":*

```
int nParams = 3;  // the number of parameters we want to set
int* pParamNumbers[] = { 1, 2, 3 };  // the array of their numbers
wchar_t* pParamValues[] = { L"5", L"e.Surname", L"Thomson" };  // the array of values
```

Of course we don't need to set all the parameters of our report. If not set, they will keep their designed values. We set the values as C-language style strings (UNICODE or ANSI). They cannot be numeric types.

2. Function - repc_call_report

`int WINAPI repc_call_report (`	Returns: ***1*** (OK), ***0*** (Error - invalid file name).
`HWND hWndParent,`	*Parent window handle.*
`BOOL bModal,`	*The dialog window type you want to open:* ***1*** *(modal),* ***0*** *(non-modal).*
`wchar_t* FormName,`	*The report file name (SFM or SFR).*
`BOOL bDesignMode );`	*The openning mode for the report:* ***1*** *(Design Mode),* ***0*** *(Working Mode).*

It is the shortest way to call a report. This function is a simplified version of repc_open_report. It doesn't limit any user privilages. They are all set to 1 ***(bParamsUpdate, bRecNavigate, bAllowDesign, bUpdateMode).*** *The other (optional) arguments have their default (designed) values.*

3. Function - repc_call_report_param

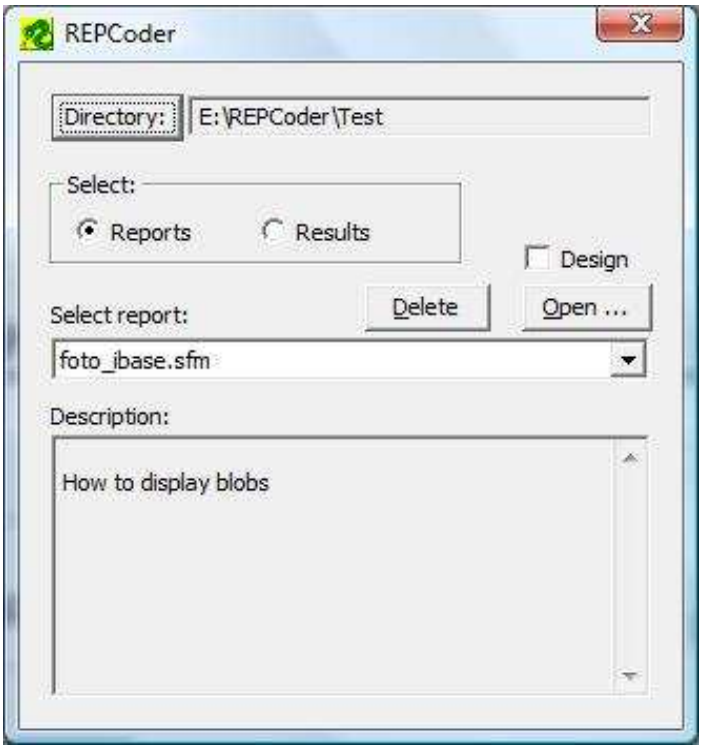
`int WINAPI repc_call_report_param (`	Returns: ***1*** (OK), ***0*** (Error - invalid file name).
`HWND hWndParent,`	*Parent window handle.*
`BOOL bModal,`	*The dialog window type you want to open:* ***1*** *(modal),* ***0*** *(non-modal).*
`wchar_t* FormName,`	*The report file name (SFM or SFR).*
`BOOL bDesignMode,`	*The openning mode for the report:* ***1*** *(Design Mode),* ***0*** *(Working Mode).*
`int nParams,`	*The number of report's parameters we want to set.*
`int* pParamsNumbers,`	*The array of numbers of these parameters (1,2,3,...).*
`wchar_t** pParamsValues,`	*The array of texts with the values of these parameters.*
`BOOL bParamsUpdate );`	*User privilage - possibility to change the report's parameters.*

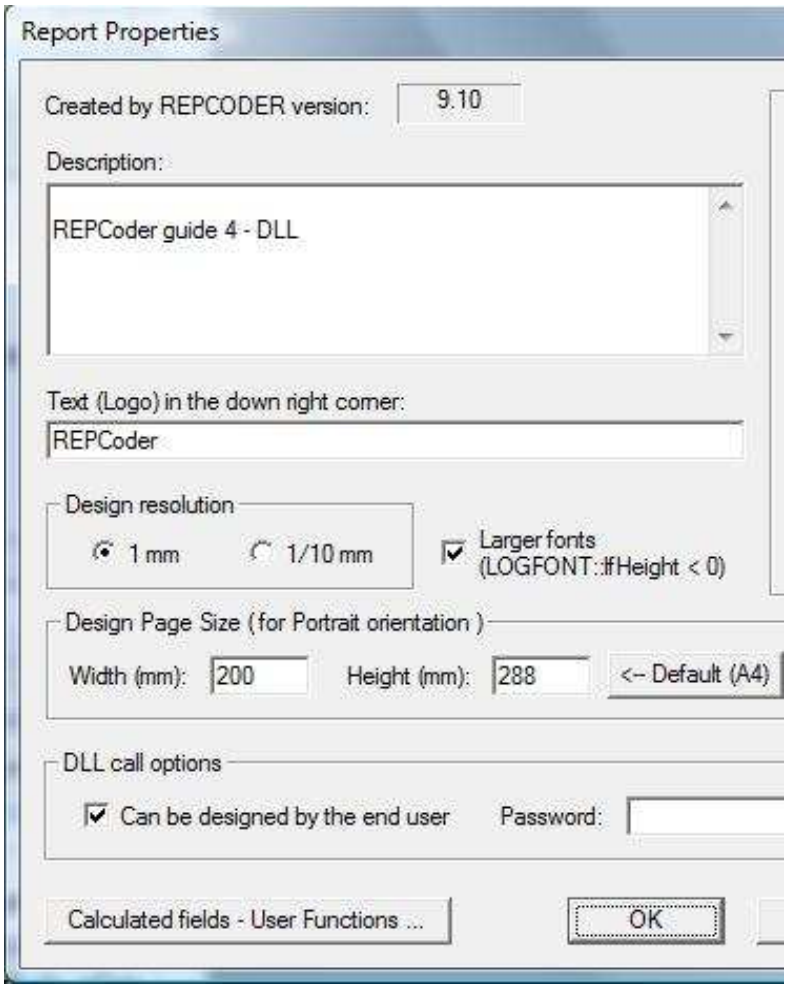
The only difference between this function and the previous one, is that it also allows to set parameters of the report.

4. Function - repc_explorer

`void WINAPI repc_explorer (`	*Calls the dialog, that allows you to manage reports (open, change, remove) from your application.*
`HWND hWndParent,`	*Parent window handle.*
`BOOL bModal,`	*The dialog window type you want to open:* ***1*** *(modal),* ***0*** *(non-modal).*
`wchar_t* Dir,`	*Directory for reports files (input and output parameter).*
`BOOL bDesign,`	*User privilage - possibility to design reports.*
`BOOL bResultOnly );`	*User limitation - possibility to open results (SFR files) only.*

*This function opens explorer-type dialog (modal or non-modal) where you can manage your reports. From this dialog you can then open **(always non-modal)** report windows.*





*The radio buttons at the top, allow you to select the file type - reports (SFM) or their results (SFR). If the argument **bResultOnly** of the **repc_explorer** function is equall 1, the user can open SFR files only. If the "Design" is checked, the report will be opened in the "Design Mode", otherwise - in the "Working Mode". But for that to happen, it is also required, that it was earlier checked the option: **"Can be designed by the end user"** in the **"Properties"** of the report (the dialog on the right). It could also be **password protected.** Notice that these options apply only when the reports are called by "REPCODER.DLL". Thus they are in the **"DLL call options"** section of the "Properties" dialog. They however do not affect the way reports are opened by "REPCODER.EXE" application. **After the directory is changed its new value is returned in the Dir** buffer (input - output function argument). So it must be global and sufficiently large.*

5. Function - *repc_get_report_desc, repc_get_report_desc_len*

`int WINAPI repc_get_report_desc (`	Returns: *1* (OK), *0* (Error - invalid file name).
`wchar_t* FormName,`	*The report file name (SFM or SFR).*
`wchar_t* Buffer,`	*Buffer that receives the report's description.*
`int buflen );`	*Buffer length (in wide characters including terminating null).*

*This function allows your application to read the **description** of a report that has the given name. The description length has no limits, so it should be first obtained by a call to **desc_length = repc_get_report_desc_len()**. You need a buffer length: (1 + desc_length)*sizeof(wchar_t). Then you pass a buffer length in wide characters: buflen = 1 + desc_length.*

6. Function - *repc_set_bde_netdir.* Only in REPCoder 32-bit.

`void WINAPI repc_set_bde_netdir (`	*It sets the "NETDIR" BDE session property for paradox tables.*
`wchar_t* Dir );`	*Full name of the network directory.*

*The "NETDIR" is the property of a **BDE session.** REPCODER always opens only one default BDE session. If you want to set "NETDIR" for that session, just use this function. You should call it always at the same beginning, before any report is called.*

7. Function - *repc_get_demo*

`BOOL WINAPI repc_get_demo ( );`	*Returns 1 or 0 depending on whether REPCODER is working in the DEMO mode.*

8. Function - *repc_set_demo*

`BOOL WINAPI repc_set_demo (`	*Returns the previous value of the bDemo variable.*
`BOOL bDemo );`	*Sets a new value of bDemo.*

These two functions can be used by programmers to force REPCODER to work in the DEMO mode. It means that "DEMO VERSION" caption will also appear on the printed pages. ***This function does not work in the TRIAL version of REPCoder!***

9. Function - *repc_get_open_count*

`int WINAPI repc_get_open_count ( );`	*Returns the number of non-modal form windows opened in the current process.*

10. Function - *repc_can_exit*

`BOOL WINAPI repc_can_exit ( );`	*Informs you if your application can be safely closed (REPCODER does not perform any background operation). It returns* ***1*** *(can exit), or* ***0*** *(cannot exit).*

These two functions help the programmer to ***close safely*** *his own application, without conflicts with "REPCODER.DLL" library. Such conflicts can happen, when the user wants to exit the application without closing all of the previously opened report windows. It of course only applies to the* ***non-modal*** *dialogs. The* ***repc_get_open_count*** *function, allows you to check how many such windows are opened in your application. If the opened reports do not perform any* ***background work,*** *there will be no problem to close the user application. All the opened REPCODER dialogs will automatically also be closed. However it belongs to a good end-user style, to close all opened non-modal windows before exiting the application. For example, some changes could be left unsaved. Thus the application can generate a warning message, informing the user about such situation. But the real dangerous conflict can happen, when some opened report is just being* ***executed*** *(in the background). If this is the case, the* ***repc_can_exit*** *function returns* ***0*** *value, which means that the application should not be exited.*

11. Function - *repc_explorer_param*

`void WINAPI repc_explorer_param (`	*Calls the modal dialog, that allows you to manage reports (open, change, remove) from your application, providing a given set of parameters.*
`HWND hWndParent,`	*Parent window handle.*
`BOOL bModal,`	*The dialog window type you want to open:* ***1*** *(modal),* ***0*** *(non-modal).*
`wchar_t* Dir,`	*Directory for reports files.*
`BOOL bDesign,`	*User privilage - possibility to design reports.*
`BOOL bResultOnly`	*User limitation - possibility to open results (SFR files) only.*
`int nParams,`	*The number of parameters for our reports.*
`int* pParamsNumbers,`	*The array of numbers of these parameters (1,2,3,...).*
`wchar_t** pParamsValues,`	*The array of texts with the values of these parameters.*
`BOOL bParamsUpdate );`	*User privilage - possibility to change the report's parameters.*

The destination of this function is very similar to that of ***repc_explorer.*** *It opens the explorer-type dialog* ***(modal or non-modal),*** *where you can manage your* ***(always non-modal)*** *report windows. Important extension here gives you the opportunity to provide a* ***set of parameters,*** *that will be later passed to each report called.*

12. Functions - *repc_set_user, repc_set_pass, repc_set_role, repc_set_charset*

`void WINAPI repc_set_user (`	*Sets username used by REPCODER while connecting with databases.*
`wchar_t* UserName );`	*User name.*

These functions allow to set username, password, role and charset that REPCODER will be using while connecting to databases. The values used here have higher priority than those built in the report during design work (there is one exception - see the red remark at the end). Application can ask user for name, password and role while connecting to database, and later REPCODER will be using these values to create reports. This way you can avoid asking the user again. Without using this function, the values built in the report (username, password, role) will be applied. ***However the highest priority have theese connection parameters (user, password, role, charset) which are defined by the report parameters.***

13. Obsolete Functions - *repc_set_fbclient_thread_safe, repc_set_ibclient_thread_safe*

void WINAPI *repc_set_fbclient_thread_safe (* **BOOL** *bSafe);*	*Call this function with bSafe=1 if your FBCLIENT.DLL or GDS32.DLL is thread-safe. This will increase performance.* *The value you want to set: **1** (thread-safe), **0** (non-thread-safe).*

*The logical variable **bSafe** corresponds to the checkbox in the program "Options" dialog: **"FBCLIENT.DLL (GDS32.DLL) is Thread-Safe".** By default, while working directly with Interbase/Firebird databases the program uses Windows critical sections to synchronize different threads, assuming that Interbase/Firebird client DLL **is not thread-safe.** This however slows down the performance. Therefore if you are using newer, thread-safe client libraries, just check this option. This will increase the performance, especially if you run many reports simultanously. **Now these functions have no effect. REPCoder checks the thread-safety of a Firebird or InterBase client library itself.***

14. Functions - *repc_set_firebird_critical_section, repc_set_interbase_critical_section*

void WINAPI *repc_set_firebird_critical_section (* **CRITICAL_SECTION*** *pCritSec);*	*Allows to set the address of a critical section that your application is using when calling Interbase/Firebird API functions.* *The address of the application-defined critical section.*

These 2 functions are for programmers writing multi-threaded applications based on InterBase (or Firebird) SQL server. They should know that the API functions of old versions of GDS32.DLL (or FBCLIENT.DLL) client library are not "thread safe". But newer versions are thread-safe. If these functions are not thread-safe the multi-threaded applications can meet serious problems. To avoid them they use Windows critical sections while calling the client functions. REPCODER.DLL is executing reports in the background as seperate threads using its own default critical section in this case. If the application is also using a critical section it is recommended that REPCODER is using the same. The repc_set_ibase_critical_section allows to pass the address of the application-defined critical section to REPCODER. Of course in such situation the application is responsible for the management of the section (functions: InitializeCriticalSection, DeleteCriticalSection).

IMPORTANT:** **If the application is not using any critical section it is recommended to call REPCODER reports as modal windows (bModal = 1). During the background execution of non-modal reports the user can have problems when trying to access the database from his application at the same time. This is not REPCODER who is guilty, but non-thread-safe client library. There is no problem if the client library is thread-safe. So start using Firebird 2.5 or its later versions. InterBase client DLL is thread-safe since version 9.0.

15. Function - *repc_init*

BOOL WINAPI *repc_init ( )*	*Initializes REPCoder. Returns: **1** (success), **0** (error).*

This function must be called by the user application at the very begining before any report was opened.

16. Function - *repc_exit*

void WINAPI *repc_exit ( );*	*Finishes work with REPCoder freeing all allocated resources. It also closes all opened non-modal report windows.*

*The application should call this function at the same end. Without doing it, some resources allocated internally by REPCODER.DLL will not be released. **Therefore it is strongly recommended to call this function.***

17. Function - *repc_set_dialog_fonts*

void WINAPI *repc_set_dialog_fonts (* **int** *Type);*	*Allows to set normal or large fonts in all dialog windows of the user interface (except Windows custom dialogs).* *Font type: **1** (Normal), **2** (Big), **3** (Extra Large)*

*The integer variable **Type** corresponds to the combobox in the program "Options" dialog: **"Font size in dialogs".** If your computer screen has large resolution you can switch to "Big" or "Extra Large" fonts in all dialog windows (except Windows custom dialogs). In the executed report the end-user can access this option in the "File" menu.*

18. Functions - *repc_alloc_report_desc, repc_free_memory*

`wchar_t* WINAPI repc_alloc_report_desc (`	*Returns the pointer to the allocated buffer with the report description. It must be later freed with repc_free_memory by the calling program.*
`wchar_t* FormName );`	*The report file name (SFM or SFR).*

Using this function you don't need to allocate a buffer in your application. But later do not forget to free the returned pointer using "repc_free_memory".

19. Functions - *repc_create_result, repc_create_result_param*

`BOOL WINAPI repc_create_result_param (`	Returns: *1* (OK), *0* (Error - invalid file name or could not execute).
`wchar_t* FormName,`	*The report file name (SFM only).*
`wchar_t* ResultName,`	*The name of the SFR file that must be created on output.*
`wchar_t* ResultDesc,`	*The optional description of the output result.*
`int nParams,`	*The number of report's parameters we want to set.*
`int* pParamsNumbers,`	*The array of numbers of these parameters (1,2,3,...).*
`wchar_t** pParamsValues,`	*The array of texts with the values of these parameters.*
`wchar_t* ErrorBuf,`	*Application-defined buffer for error text if the function returns 0.*
`int ErrorBufLen );`	*The size of the ErrorBuf in characters.*

This function executes a report and creates its result (as SFR file) without opening any windows or displaying message boxes. So it is a completely silent function. If an error occurs its text is placed in the optional ErrorBuf.

20. Function - *repc_admin_setup*

`BOOL WINAPI repc_admin_setup (`	Returns: *1* (OK), *0* (Error - invalid ConnString or other parameters).
`BOOL bInterBase,`	*The type of the reporting database: 1 (InterBase), 0 (Firebird).*
`wchar_t* ConnString,`	*Connection string to the reporting database.*
`wchar_t* UserName,`	*UserName for the reporting database.*
`wchar_t* Password,`	*Password for the reporting database.*
`wchar_t* ProjectName,`	*The name of the Project in the reporting database.*
`wchar_t* DbName,`	*The name of a target database in the reporting database.*
`wchar_t* DbLogin,`	*UserName for the target database (optional).*
`wchar_t* DbPassword,`	*Password for the target database.*
`wchar_t* DbRole,`	*RoleName for the target database (optional).*
`BOOL bVerify,`	*Should the above parameters be verified ? 1 (YES), 0 (NO).*
`BOOL bShowErrorMess );`	*Show the error messages of verification ? 1 (YES), 0 (NO).*

*This function integrates REPCODER.DLL with the **REPCoder Admin** program (reporting server). It enables you to directly call the reports that are stored in a remote database (Firebird or InterBase) as blobs. It is a more elegant solution when the reports are kept rather in a database than in simple disc files. After successful call of "repc_admin_setup", you can call all other functions of REPCODER.DLL. The only difference is that in the "FormName" parameter DO NOT specify the ".SFM" extension. Thats all. For details see the documentation of our "REPCoder Admin" new product. It can be downloaded together with REPCoder at **www.repcoder.com***

REPCoder for C#

*The programming interface for C# is provided in the **Repcoder class** (file: REPCODER.CS). All the 24 methods are static functions. They are listed below:*

```
public static bool init();
```

```
public static void exit();
```

```
public static bool open_report(int hWnd, bool bModal, string FormName, int RecordSeqNr,
int nParams, int[] pParamsNumbers, string[] pParamsValues, bool bParamsUpdate,
bool bRecNavigate, bool bDesignMode, bool bAllowDesign, bool bAllowUpdate);
```

```
public static bool call_report(int hWnd, bool bModal, string FormName, bool bDesignMode);
```

```
public static bool call_report_param(int hWnd, bool bModal, string FormName,
bool bDesignMode, int nParams, int[] pParamsNumbers, string[] pParamsValues,
bool bParamsUpdate);
```

```
public static void explorer(int hWnd, bool bModal, string Dir,
bool bDesign, bool bResultOnly);
```

```
public static void explorer_param(int hWnd, bool bModal, string Dir, bool bDesign,
bool bResultOnly, int nParams, int[] pParamsNumbers,
string[] pParamsValues, bool bParamsUpdate);
```

```
public static string get_report_desc(string FormName);
```

```
public static int get_open_count();
```

```
public static bool can_exit();
```

```
public static void set_user(string User);
```

```
public static void set_pass(string Pass);
```

```
public static void set_role(string Role);
```

```
public static void set_charset(string CharSet);
```

```
public static void set_logo(string Logo, bool bReplace);
```

```
public static bool get_demo();
```

```
public static bool set_demo(bool bDemo);
```

```
public static void set_bde_netdir(string dir);
```

```
public static void set_fbclient_thread_safe(bool bSafe);
```

```
public static void set_ibclient_thread_safe(bool bSafe);
```

```
public static void set_dialog_fonts(int Type);
```

```
public static void set_user_interface_lang(int Lang);
```

```
public static string create_result(string FormName, string ResultName, string ResultDesc);

public static string create_result_param(string FormName, string ResultName,
string ResultDesc, int nParams, int[] pParamsNumbers, string[] pParamsValues);
```

Returns empty string on success or string with error message on failure.

```
public static bool admin_setup(bool bInterBase, string ConnString, string User,
string Pass, string ProjectName, string DbName, string DbLogin, string DbPass,
string DbRole, bool bVerify, bool bShowErrorMess);
```

REPCoder for Java (only on Windows systems)

*The programming interface for Java is very similar to C#. It is also provided in the **Repcoder class** (file: REPCODER.JAVA). It is using JNA.JAR archive (Java Native Access) to access REPCODER.DLL. All the 24 methods are static functions. They are listed below:*

```
static boolean init();
```

```
static void exit();
```

```
static boolean open_report(int hParent, boolean bModal, String FormName, int RecordSeqNr,
int nParams, int[] pParamsNumbers, String[] pParamsValues, boolean bParamsUpdate,
boolean bRecNavigate, boolean bDesignMode, boolean bAllowDesign, boolean bAllowUpdate);
```

```
static boolean call_report(int hParent, boolean bModal, String FormName,
boolean bDesignMode);
```

```
static boolean call_report_param(int hParent, boolean bModal, String FormName,
boolean bDesignMode, int nParams, int[] pParamsNumbers, String[] pParamsValues,
boolean bParamsUpdate);
```

```
static void explorer(int hParent, boolean bModal, String Dir, boolean bDesign,
boolean bResultOnly);
```

```
static void explorer_param(int hParent, boolean bModal, String Dir, boolean bDesign,
boolean bResultOnly, int nParams, int[] pParamsNumbers, String[] pParamsValues,
boolean bParamsUpdate);
```

```
static String get_report_desc(String FormName);
```

```
static int get_open_count();
```

```
static boolean can_exit();
```

```
static void set_user(String User);
```

```
static void set_pass(String Pass);
```

```
static void set_role(String Role);
```

```
static void set_charset(String Charset);
```

```
static void set_logo(String Logo, boolean bReplace);
```

```
static boolean get_demo();
```

```
static boolean set_demo(boolean bDemo);
```

```
static void set_bde_netdir(String dir);
```

```
static void set_fbclient_thread_safe(boolean bSafe);
```

```
static void set_ibclient_thread_safe(boolean bSafe);
```

```
static void set_dialog_fonts(int Type);
```

```
static void set_user_interface_lang(int Lang);
```

```
static String create_result(String FormName, String ResultName, String ResultDesc);
```

```
static String create_result_param(String FormName, String ResultName, String ResultDesc,
int nParams, int[] pParamsNumbers, String[] pParamsValues);
```

Returns empty string on success or string with error message on failure.

```
static boolean admin_setup(boolean bInterBase, String ConnString, String User,
String Pass, String ProjectName, String DbName, String DbLogin, String DbPass,
String DbRole, boolean bVerify, boolean bShowErrorMess);
```

REPCoder for Delphi

The programming interface for Delphi is very similar to C# and Java. It is provided in the ***REPCODER unit*** *(file: REPCODER.PAS). The number of UNICODE functions in the unit is 25. Some of them have also their ANSI versions (blue color) as the C/C++ interface. They are listed below:*

```
repc_init: function: Bool;
```

```
repc_exit: procedure;
```

```
repc_open_report: function(hParent: HWND; bModal: Bool; FormName: WideString;
RecordSeqNr: Integer; nParams: Integer; pParamsNumbers: PInteger;
pParamsValues: PWideString; bParamsUpdate: Bool; bRecNavigate: Bool;
bDesignMode: Bool; bAllowDesign: Bool; bAllowUpdate: Bool): Integer;
```

```
repc_call_report: function(hParent: HWND; bModal: Bool; FormName: WideString;
bDesignMode: Bool): Integer;
```

```
repc_call_report_param: function(hParent: HWND; bModal: Bool; FormName: WideString;
bDesignMode: Bool; nParams: Integer; pParamsNumbers: PInteger; pParamsValues: PWideString;
bParamsUpdate: Bool): Integer;
```

```
repc_explorer: procedure(hParent: HWND; bModal: Bool; Dir: WideString; bDesign: Bool;
bResultOnly: Bool);
```

```
repc_explorer_param: procedure(hParent: HWND; bModal: Bool; Dir: WideString;
bDesign: Bool; bResultOnly: Bool; nParams: Integer; pParamsNumbers: PInteger;
pParamsValues: PWideString; bParamsUpdate: Bool);
```

```
repc_get_report_desc_len: function(FormName: WideString): Integer;
```

```
repc_get_report_desc: function(FormName: WideString; buf: PWideChar; len: Integer): Bool;
```

```
repc_get_open_count: function: Integer;
```

```
repc_can_exit: function: Bool;
```

```
repc_set_user: procedure(User: WideString);
```

```
repc_set_pass: procedure(Pass: WideString);
```

```
repc_set_role: procedure(Role: WideString);
```

```
repc_set_charset: procedure(CharSet: WideString);
```

```
repc_set_logo: procedure(szLogo: WideString; bReplace: Bool);
```

```
repc_get_demo: function: Bool;
```

```
repc_set_demo: function(bDemo: Bool): Bool;
```

```
repc_set_bde_netdir: procedure(dir: WideString);
```

```
repc_set_fbclient_thread_safe: procedure(bSafe: Bool);
```

```
repc_set_ibclient_thread_safe: procedure(bSafe: Bool);
```

```
repc_set_dialog_fonts: procedure(font_type: Integer);
```

```
repc_set_user_interface_lang: procedure(Lang: Integer);
```

```
repc_create_result: function(FormName: WideString; ResultName: WideString;
ResultDesc: WideString; ErrorBuf: PWideChar; ErrorBufLen: Integer): Bool;
```

```
repc_create_result_param: function(FormName: WideString; ResultName: WideString;
ResultDesc: WideString; nParams: Integer; pParamsNumbers: PInteger;
pParamsValues: PWideString; ErrorBuf: PWideChar; ErrorBufLen: Integer): Bool;
```

```
repc_admin_setup: function(bInterBase: Bool; ConnString: WideString; User: WideString;
Pass: WideString; ProjectName: WideString; DbName: WideString; DbLogin: WideString;
DbPass: WideString; DbRole: WideString; bVerify: Bool; bShowErrorMess: Bool): Bool;
```