

First Steps with Rave Reports for Delphi, C++Builder and Kylix



© 2003 Thomas Pfister, Germany

This article will show you, how easy RAVE works and all that you need for your first report with Nevrona RAVE Reports 5.

Since Delphi 7, many developers know Nevrona RAVE as the new report engine for Delphi, but RAVE isn't a new product. This product has existed since 1995 and in these eight years it has developed into the ultimate Reporting Technology.

The history of RAVE:

- April 1995: ReportPrinter 1
- September 1996: ReportPrinter Pro 2
- December 1998: RAVE 3.0
- October 2000: RAVE 4.0
- August 2002: RAVE 5.0 ("BE" in Delphi 7)

In ReportPrinter Pro 1 and 2, you were able to create code-based reports with countless options. In Version 3, Nevrona added the visual designer RAVE (Report Authoring Visual Environment) into the Print-Suite. With RAVE 5, you now get an extensive and stable report engine as well as the powerful code-based components. After Installation of the RAVE BEX-version you'll get a new register with 19 components and more then 500 properties and events for printing. Nevrona offer a trial-version of RAVE BEX on their site.

What are the differences between the RAVE BE (Borland Edition) and the RAVE BEX (Borland Edition eXtended)? Many rumors are on the different newsgroups about this. In fact, there are differences, but with BE bundled with Delphi 7 you have a full version of RAVE and can create reports (and deploy, of course) in your application. The greatest differences are:

- Source code – the BEX version comes with full component source code (not the IDE source).
- "Archived" Code Based Components (including the powerful Shell-components) are included with the BEX version for backwards compatibility.
- More frequent updates.
- Support for TeeChart Pro – the BE version only supports the bundled TeeChart.
- Improved Scripting in 5.1x BEX

So we are getting a "reasonable" version bundled with Delphi and if any of the above matter to you, then you should consider getting the BEX version.

Before we go into the RAVE-System I'll list some distinguishing features of RAVE Reporting:

- Enabled for multiplatform development (like Windows, Linux) with VCL and CLX-Technology.
- RAVE for .Net is announced in the Nevrona-newsletter from July 2003
- Works with Delphi 4-7, C++ Builder 4-6, Kylix 3
- Connection with TDataset-descendent or directly with a database (BDE, ADO, dbExpress, Oracle, Interbase)

Now we can start the first Report with RAVE.



Many developers will change from QuickReport to RAVE. The first major difference is the RAVE IDE, meaning that RAVE doesn't use a Delphi-form for the report. This means that all reports are off the application and will be stored in (at least) one external file with the format "*.rav". It is possible to link this file into the exe-file and then you are able to deploy only one exe-file with all necessary information. No DLLs or other files are required for printing your reports.

With version 5.0.8 and higher you can use both dfm-formats, with older RAVE-versions (the bundled RAVE5-version are 5.0.4 but a free update to v5.0.8 is available!) you must define the binary-dfm-format for the Delphi-form that contains the RAVE Project.

But I advise that you deploy an external file with your applications. The advantages in my opinion are a quick update for the reports (The reports generally change more often than the actual application) and with the optional End User Designer License (EUDL) the user can modify the reports (for the EUDL you need the BEX-version). In addition you can store the report-file into a database or in a DLL file.

While RAVE can be used with any database engine, the BDE will be used in all the examples because every delphi developer has it installed. Once you understand how to build reports using the BDE examples, it should be relatively simple to change to any other database.

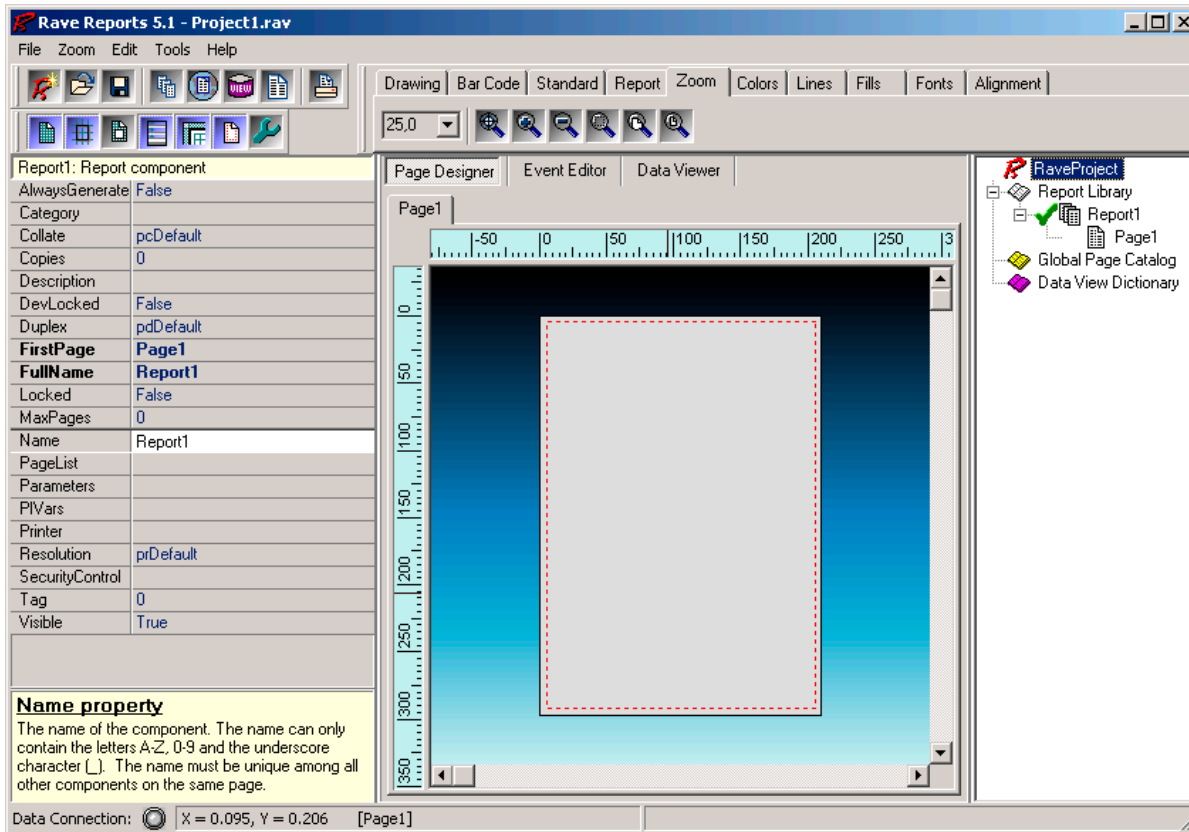
For controlling we insert a TDataSource and a Datagrid and connect this to Table1.

After this Delphi-work we must insert a RAVE-component "RvProject" on the form. This is the link between your application and the report engine. Here you can define some properties (If you know Nevrona then you're surprised about the less properties and events ☺).

The property DLLFile is for the EUDL and we'll skip this in this document. The next property is more interesting. The Engine property will connect to one engine. The correct engine should be the RvSystem-component. Insert the RvSystem-component from the RAVE-register and now you can connect to this component (I'll describe this component later). The next property is the ProjectFile. Here you must define the RAVE-project-file (this file doesn't exist yet). With the StoreRav-property you can store the project-file into the exe-file (look above).

Now we will start the RAVE IDE with a double-click on or right click and select "RAVE Visual Designer" the RAVEProject-component to bring up the visual designer (in Kylix you must start the RAVE IDE under the Pulldown Menu Tools). With RAVE 5.1.0 or earlier you must start the RAVE IDE from inside the Borland-Environments. With Version 5.1.1 Nevrona changed the developer version of the RAVE report designer (RAVE.exe) so that Delphi does not need to be running to execute (BEX version).

You should see the RAVE-IDE like the picture:



This IDE looks like the Delphi-IDE and is in my opinion very intuitive. It is divided into

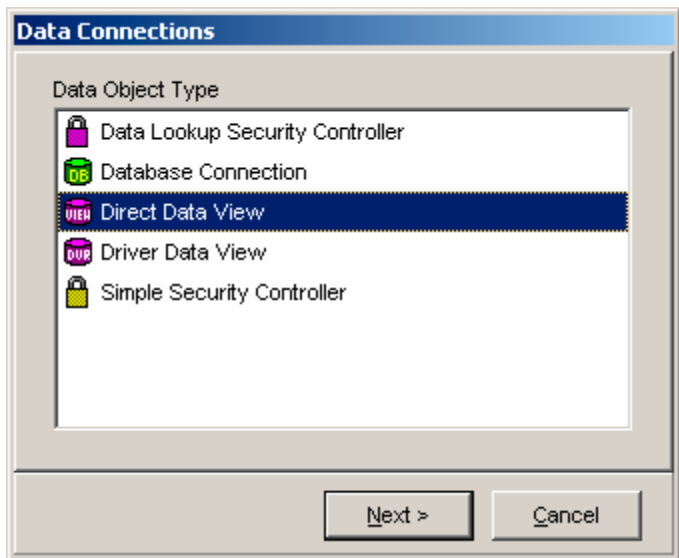
- Project Tree
- Property Panel
- Page Designer
- Status Bar
- IDE Toolbars

The Project Tree contains all the reports of you application, the global pages and all the database-components. The property-panel works like Delphi and the Page designer is for designing the report (what a surprise...). The IDE-Toolbar contains the RAVE-components and is divided into areas (for example Drawing, Standard, Report and so on).

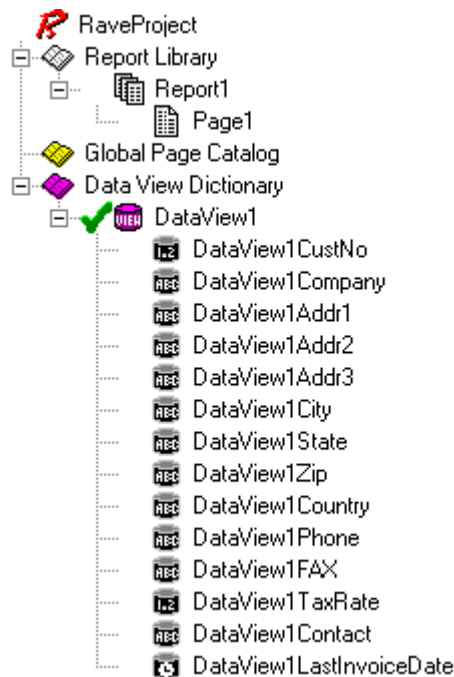
Although we've changed nothing we save the RAVE-file into a new folder and close the IDE. In Delphi we save the project files in the folder, too. Now we can define the Project file in the RvProject1-component. Attention: RAVE will store the complete file information including the folders. Delete this information and RAVE will use the RAVE-file inside the active directory.

After this we'll insert the link-component between our TTable and the report engine. There are four components on the RAVE-register from the type of DatasetConnections. The two components with the "T" and "Q" are for the BDE Table and BDE Query; since the BDE is out we should not use this component. The DS-connection componet works with all TDataset components including the TTable. Drop a RvDatasetConnection on the Delphi-form and connect the property Dataset with the TTable1. That's all.

Now we start the RAVE-IDE again and must here connect with the RvDatasetConnection inside our Delphi-Application. You'll find an icon called "View" or in the File menu the "New Dataobject". The next Dialogbox will be displayed:



Choose "Direct Data View", this means that you will connect to a TDataset-Connection from your Borland-Application. On "Next >" you can define the active Dataset connections. Now read RAVE the Meta information from the connection and display on the right side the Dataview1 like the following picture:



Beside the field you can see the type of the columns. And here you can define the format for every field. Simply mark the field and define the property display-format like App.A of the RAVE.pdf.

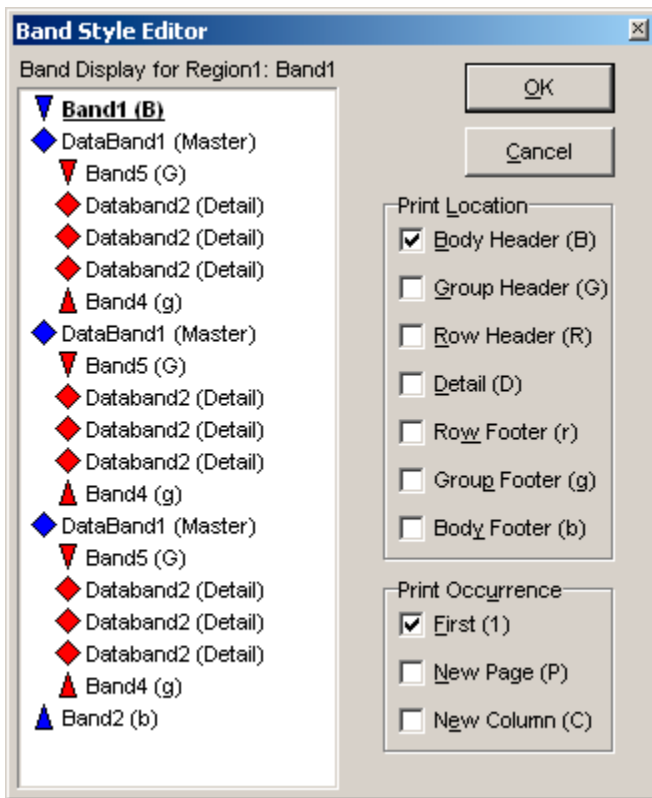
Now we can create our first report with a list of all customers.

RAVE allow both page-oriented and band-oriented approaches. For a list report, a band-oriented approach will be the better choice. You can see the page with a red line around. This is the printable area (inside the waste area) of the default printer on the developer workstation. Insert from the “Report” toolbar a Region-component and resize it inside the waste-area. That’s your band-area. Now you can insert a Databand from the “Report” toolbar into the Region. Connect the Databand1.Dataview with your Dataview1 and insert the CustNo and Company-field in this databand. With the CTRL-Key you can drag’ Drop these columns on the Databand. After this you can start the preview like Delphi with F9. Now you should see the OutputOptions-Dialog and after confirm the setting the correct report will displayed.



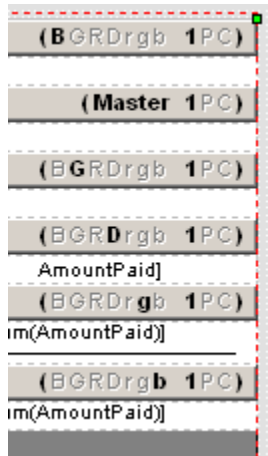
Tip: You can define in the Edit-Preferences -> “Printing” tab the OutputOptions and PrintDestinations (here only) for design time. Uncheck the “ShowOutputOptionsDialog” and set the Destination to Preview and you’ll run direct in the preview.

After this you can insert a normal band for header information. You must set the Controllerband to DataBand1 and then define the BandStyle like the following picture:



RAVE shows you an example of the runtime-result.

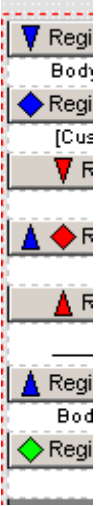
A little tricky is to re-order the bands after you've inserted the band; RAVE will insert the new bands always after the existing (data) bands. To change the design-view you must click on the band and with right-mouseclick "order" – MoveForward or MoveBehind you can re-order the bands and "synchronize" between runtime and designtime. Another way is in the alignment-toolbar and there you'll find "MoveForward" or "MoveBehind".



In the Bandstyle-Editor you can define the Style (of course...) and you can see some characters after the print location, like "B" and on the Print Occurrence. These characters are on the designer on the BandHeaders of every band. Our Band1 are a Bodyheader and in the header of the band you see the character "B". Here a little example for a report with some bands more:

RAVE shows on the BandStyleEditor and on the designer different colors like the following picture:

The blue (databands) are connected and the read are connected (and with the blue for a master/detail-connection bundled). The green band is without any connection to a controllerband.



I think you'll see the great visual help on designing a report. And don't forget: We only go the first (of many) steps with RAVE!

Next we will insert in the report is the page-counter like "page 1 of 2".

First: "Don't limit RAVE by preconceptions". RAVE works band and page-oriented. We insert a DataText-component (from the report-register) direct on the page, not in the existing region. Now we open the Datatext-Editor with a click on the "..." in the DataField-Property. RAVE will display the powerful Datatext-Editor. Here you can "concat" for example Datafields into on DataText (addresses are a great area for this). With the +-sign you make a concat without spaces, with the &-sign RAVE insert between the fields a blank character.

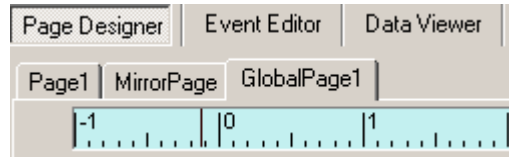
But we will go back to our page-counter. You can find in the editor on the left side "Report Variables". Here exist many static variables for reporting, e.g. Pages and Date/Time variables. On the Data text-Field at the bottom, write the following "page"&&"of"&&"pages". Now we can insert the Report variable "Current Page" between the first &&-characters and "TotalPages" between the second &&-characters. After this you can close the dialog with "ok" and must change the property "Always generate" in your Report (not the page!) into "true". This determines whether the report will generated before printing, without this RAVE can't calc the Total Pages.

You can use this for inserting date or time-information in your report as well.

After this information you should be able to create more reports like this. And don't forget: all your reports in you application will store in one rav-file. And on the right side of your RAVE-project you see after the report library the "global pages"-area. This feature is very powerful. You can insert in this global pages (just insert a global page with a click on the "report and the earth in back"-icon). RAVE inserts a "GlobalPage1" and show an empty page. Here you can store on one central place every RAVE component and can use this on every report.



You can switch very fast if you click the page-register on the top of the designer:

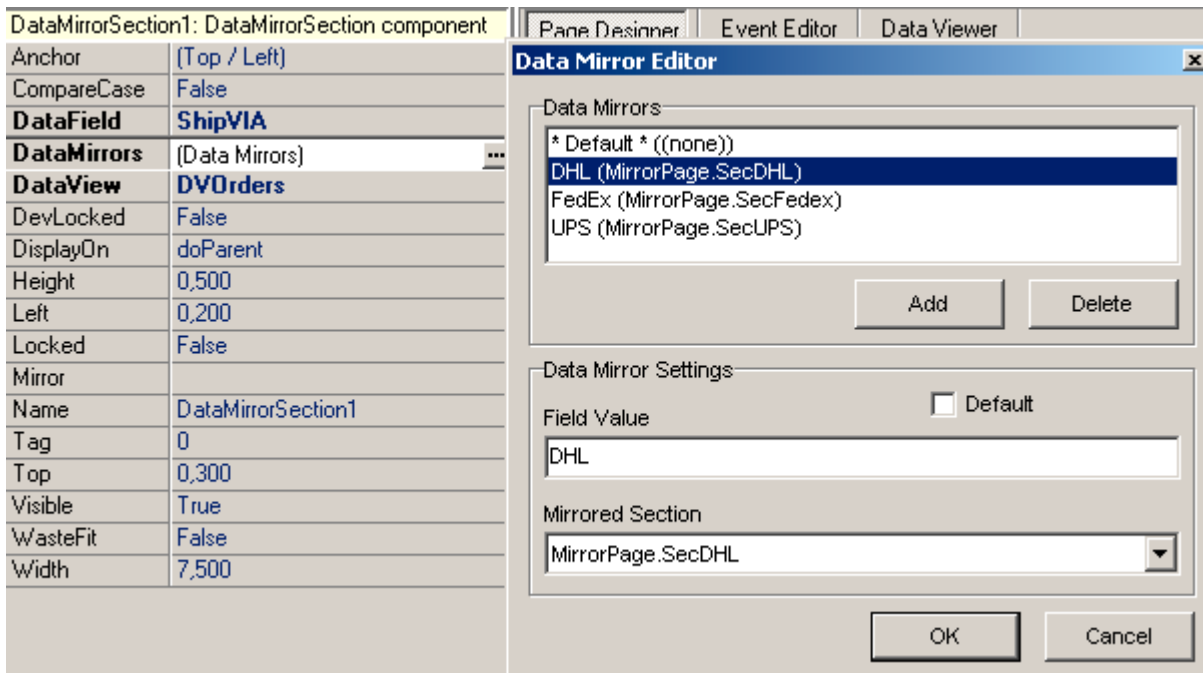


Like the corporate-design you can insert the logo of the company and use this jpeg (you'll find an add-on for using jpg's here: <http://www.nevrona.com/RAVE/addons.shtml>) on all your reports.

Insert on the global page a jpeg-component, defines the jpeg-file or image and on your report you must insert another jpeg-component. But don't load an image or file link for this component. You can define the property mirror with the jpeg-component on the global page. That's all. The component on your page one is not green marked, it is now with yellow marks. And with this know-how you're able to use the global page for a central storage and insert the mirrors at the real report. RAVE store only one (graphic-) component in the project and on executing RAVE insert the (graphic-) component into you report. If the company logo will change, no problem, you only change the component on the global page and all the inherited components are updated as well.

This is the first "mirror"-technique of RAVE, the other is the "datamirror". Normally this will take an own article. But I try to explain this in some lines and you will realize this and can use it for your reporting. Data Mirroring is another one of those areas in RAVE that take a bit of time to come to terms with but is really worth the effort. Basically, they allow different sections to be used depending on certain conditions. On the destination place you must insert an open space. The ""datamirror"-component for this you'll find in the register "Report". Now you need a place for the sections with the different components. You can use the global pages for this or you insert a new page in your report but don't use this page for printing, only for store components and use it with data mirror or "normal mirror" like the jpg-mirror above.

For example you can insert two sections (register standard) on your global page and insert a Text-component with the text "section1" and a Text-component with "section2" in the other section. On your "data mirror"-component you can define the mirrors in the data mirror editor. The editor defines combinations of field values and section components. There are used to determine which section will be mirrored when this component is printed. A default data mirror can be defined that will be uses if no other match is found. For example you can make a report with the DBDEMOS-Alias and in the orders.db you'll find the column "ShipVIA". Here you can define for every data, like UPS or FedEx, an own section with the special address-format. You can basically think of this like a Case Statement where different sections are used depending on the field value.



I've created three sections, called SecXXX, and for DHL, UPS and FedEx I've insert a special addresses-format.

Well, these are the first steps with RAVE. I hope you've enjoyed the first RAVE-party on these pages and can create your reports. And in the next time you'll see some articles more about RAVE reporting on the Nevrona website.

The next article will talk about the database-connections (meaning the different ways), group reporting, master/detail-reporting (which one is better), rendering (PDF- and html-export), RAVE reporting server, RAVE scripting and more.

Thanks for helping me to write this article:
Jim Gunkel, Nevrona Designs
Glenn Crouch, ESBconsult and TeamNevrona-Member
Trevor Keegan, TeamNevrona-Member

Stay tuned.