
# OI BRW 10.0 Reference Guide


REVELATION
SOFTWARE
**A Division of Revelation Technologies, Inc.**

# Introduction

The OI BRWDesigner application is a tool used for creating and editing OI BRW report definition files.

## Create New Reports

Use the OI BRW Wizard to quickly and easily create a new report.

1. Select the data source for the new report
2. Select the fields you want to include in the report
3. Set the layout, style and title for the new report

## Design and Modify Reports

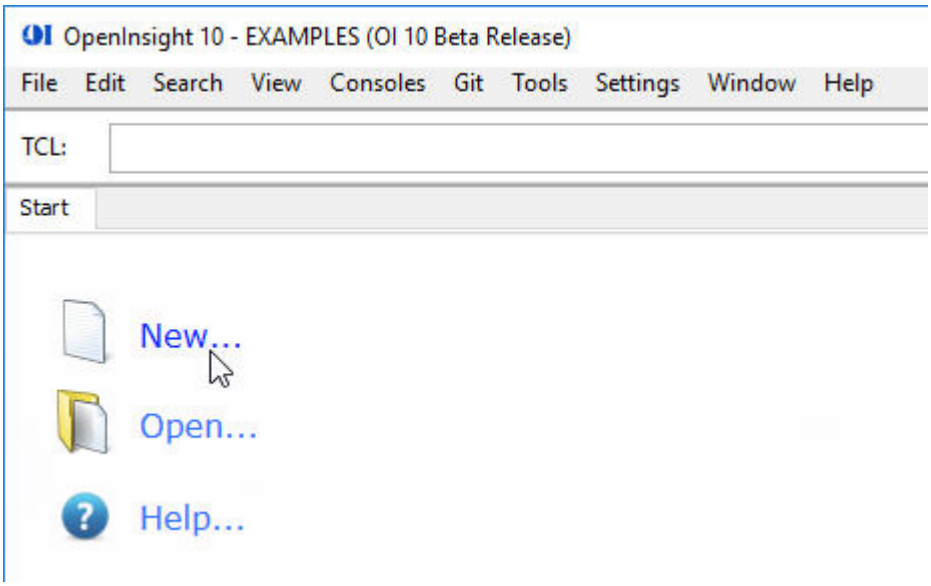
The Access-style WYSIWYG design surface makes designing reports easy and intuitive. Drag and drop report fields from the toolbar onto the report's design surface. Banded regions mark each area of the report such as header, body and footer. Set all field related properties directly in the application itself. You can even write custom VBScript code from the Properties window.

## Export Reports

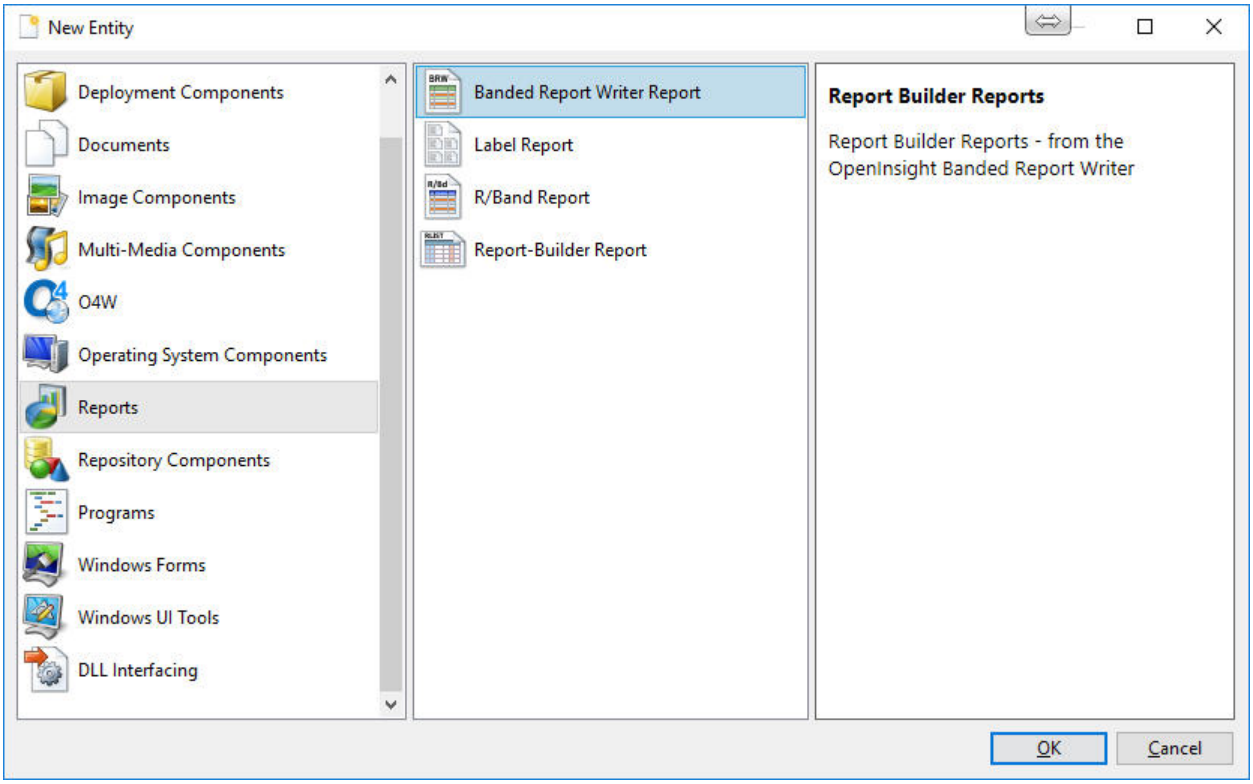
Directly export a report to any of the supported file formats: HTML, PDF, RTF, XLS, XLSX, TIF, TXT or ZIP.

# Creating a Quick Report using the OI BRW

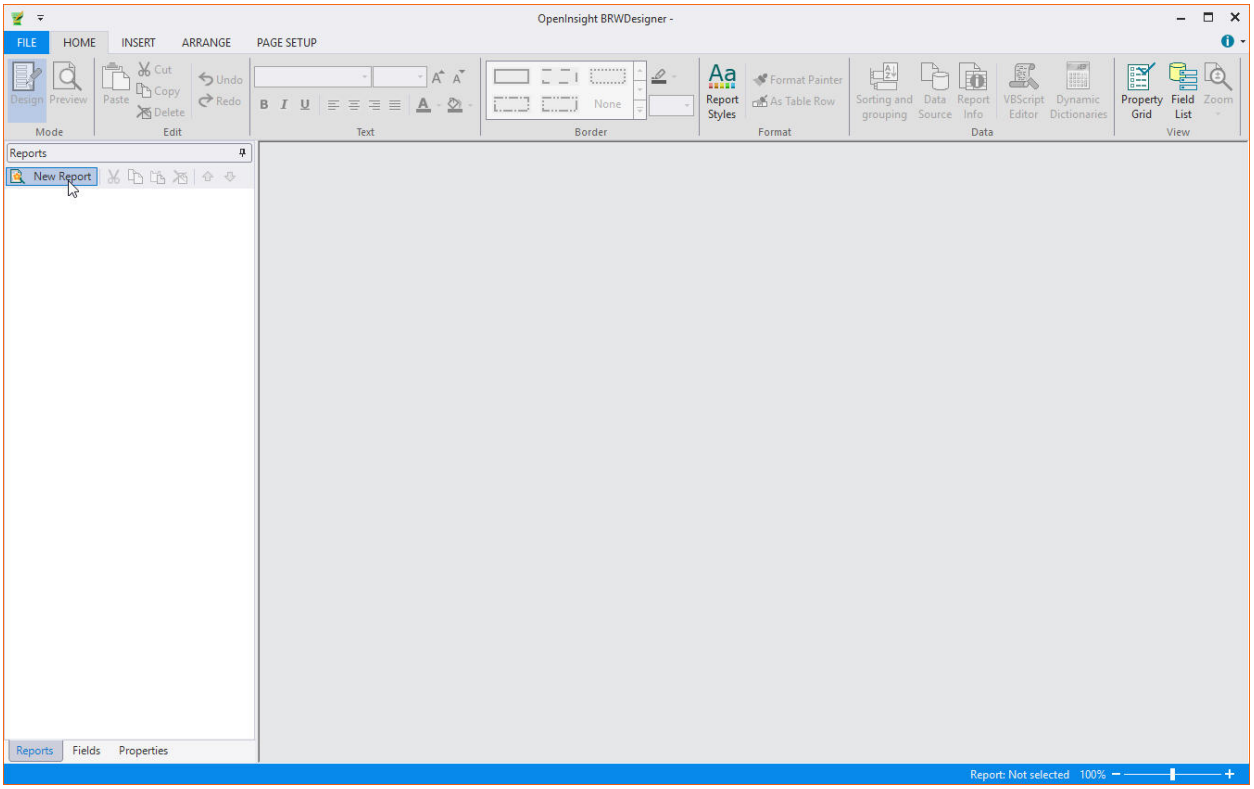
From the OpenInsight IDE, choose New…



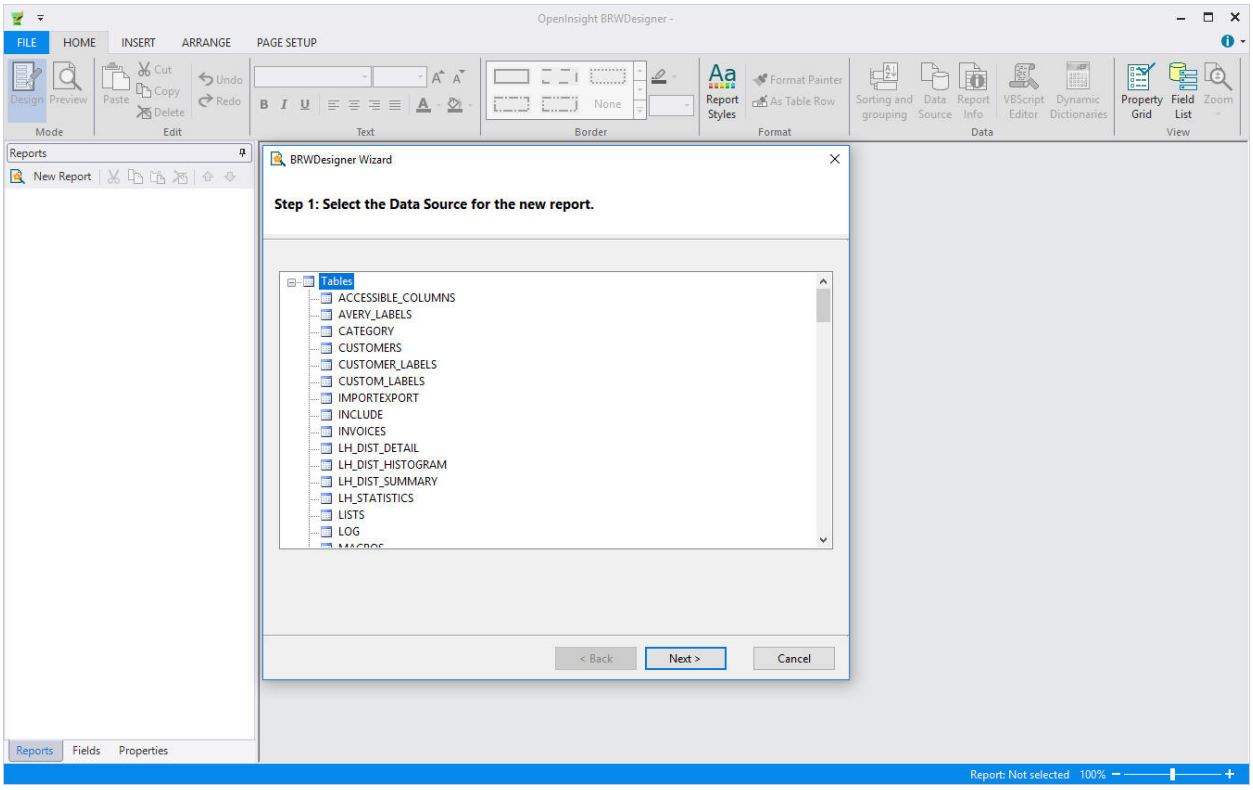

From the New Entity dialog choose Reports, Banded Report Writer Report and click OK.



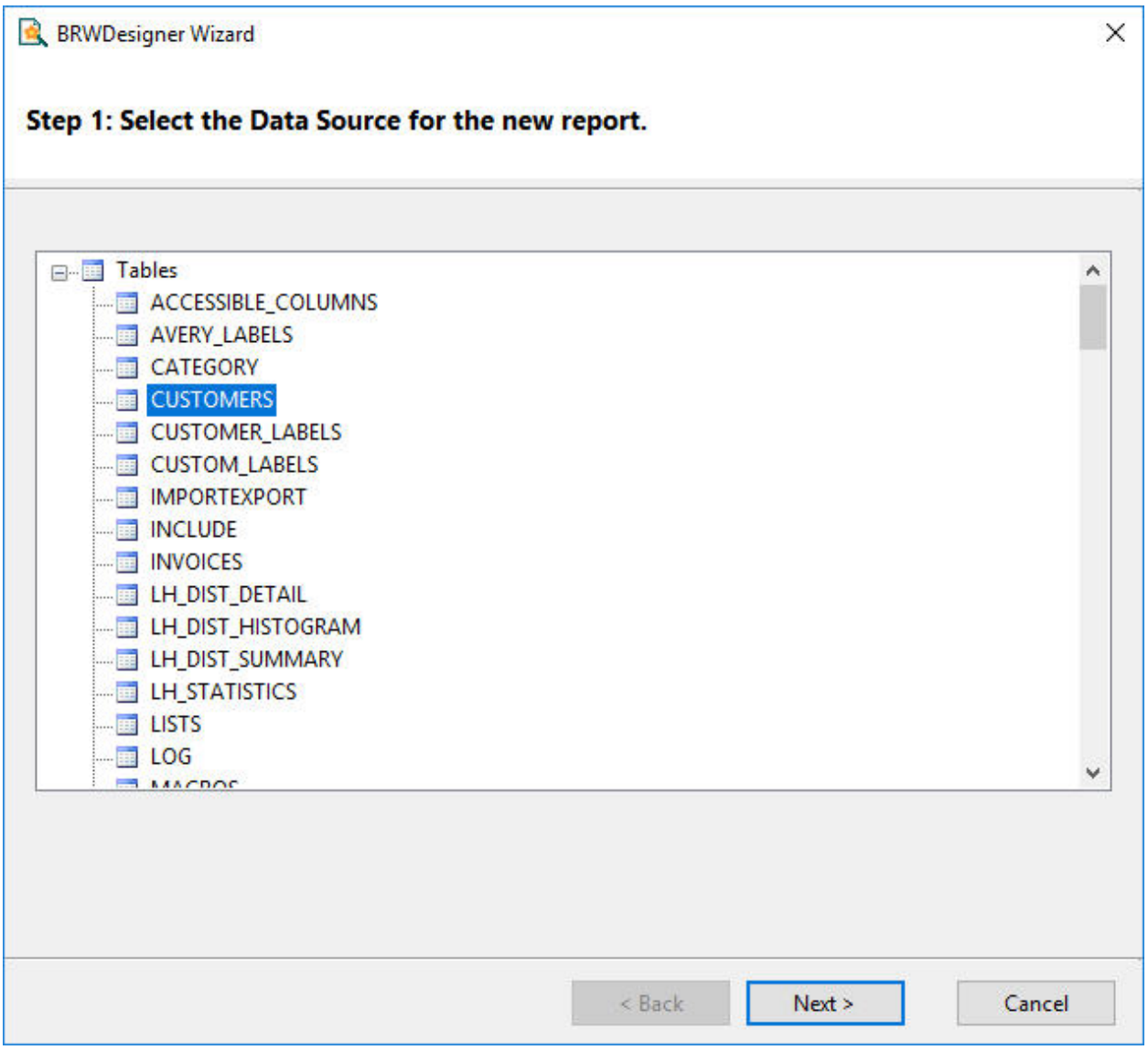
Once the OI BRW launches make sure that you choose the Report View from the bottom left panel. This will allow you to choose an existing report or create a new report.



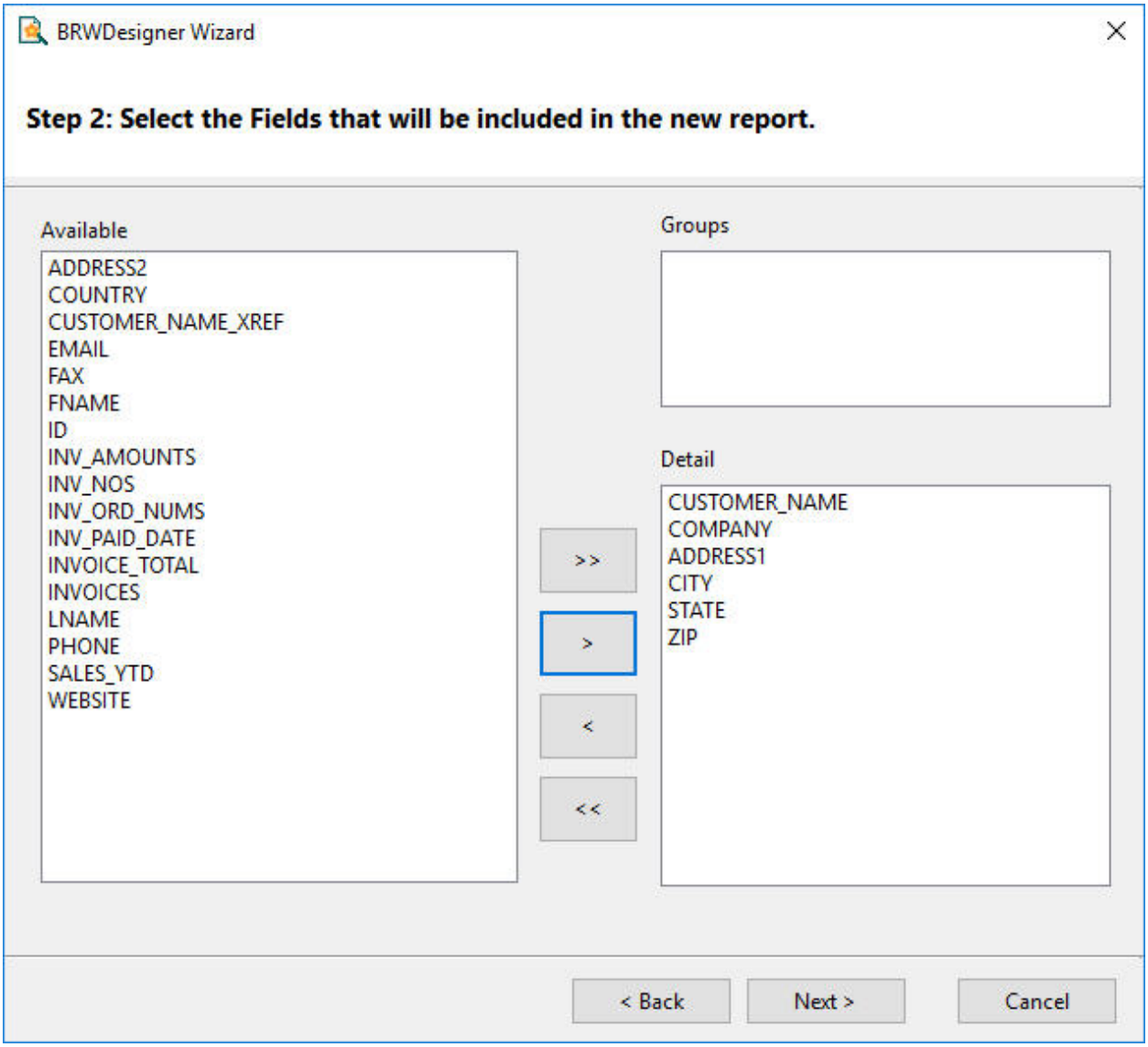
Click the New Report tab to create a new report. This will launch the BRWDesigner Wizard.



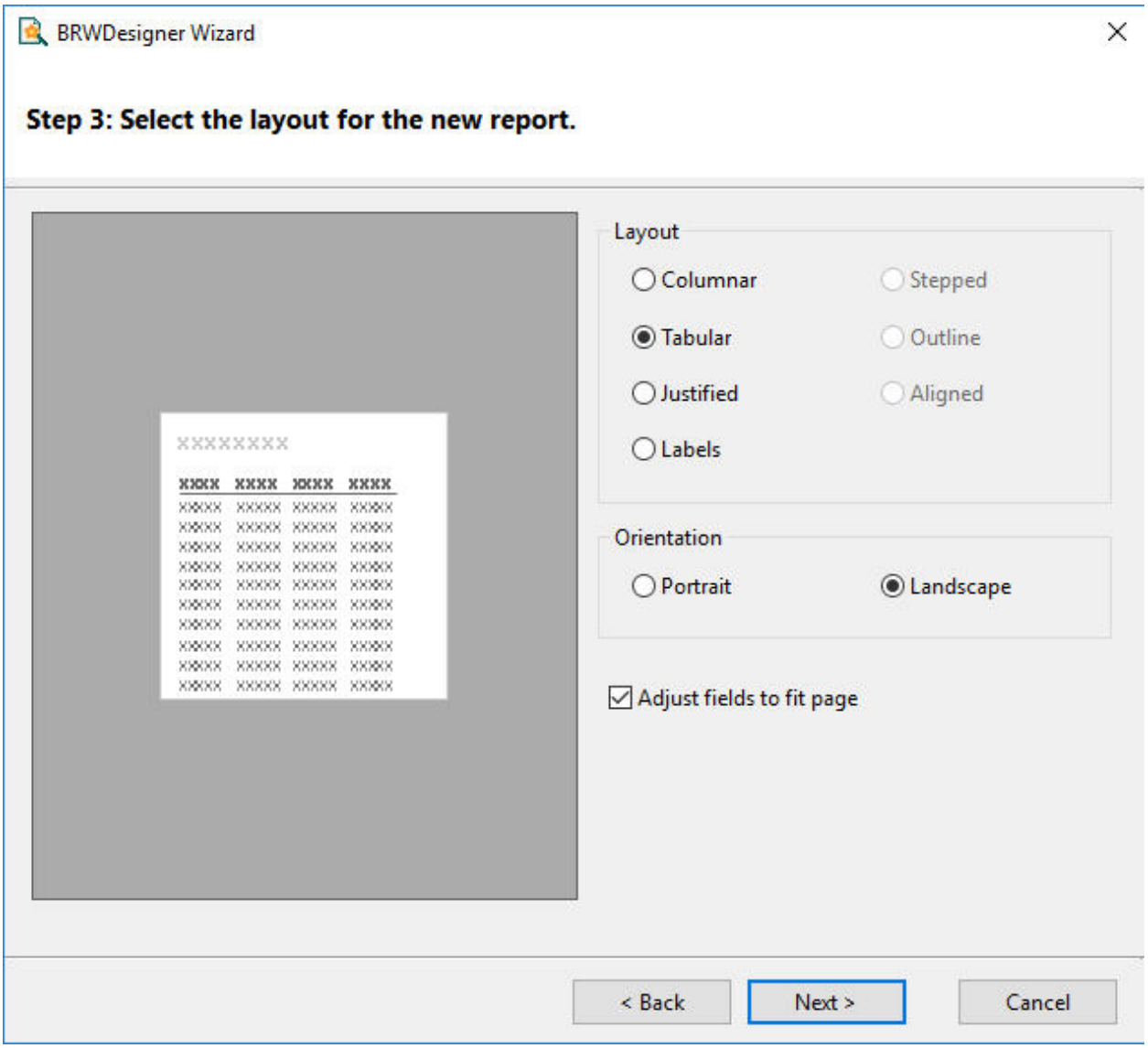
Your OpenInsight tables attached to your application will be displayed. Select the table for use with your new report and click Next.



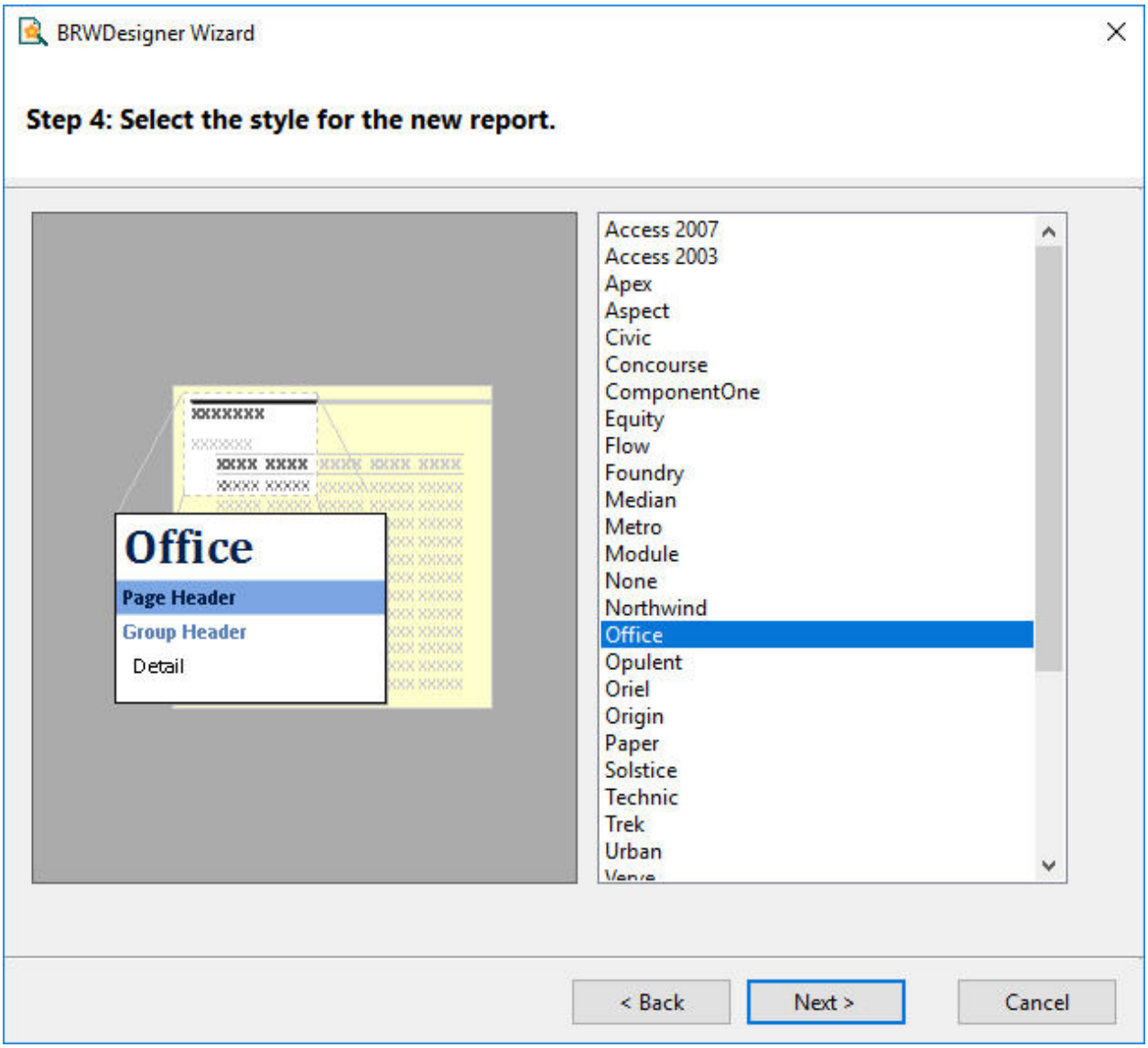
The OpenInsight dictionary fields for your selected table will be displayed. Select the fields that you want to be included in your report and click Next.



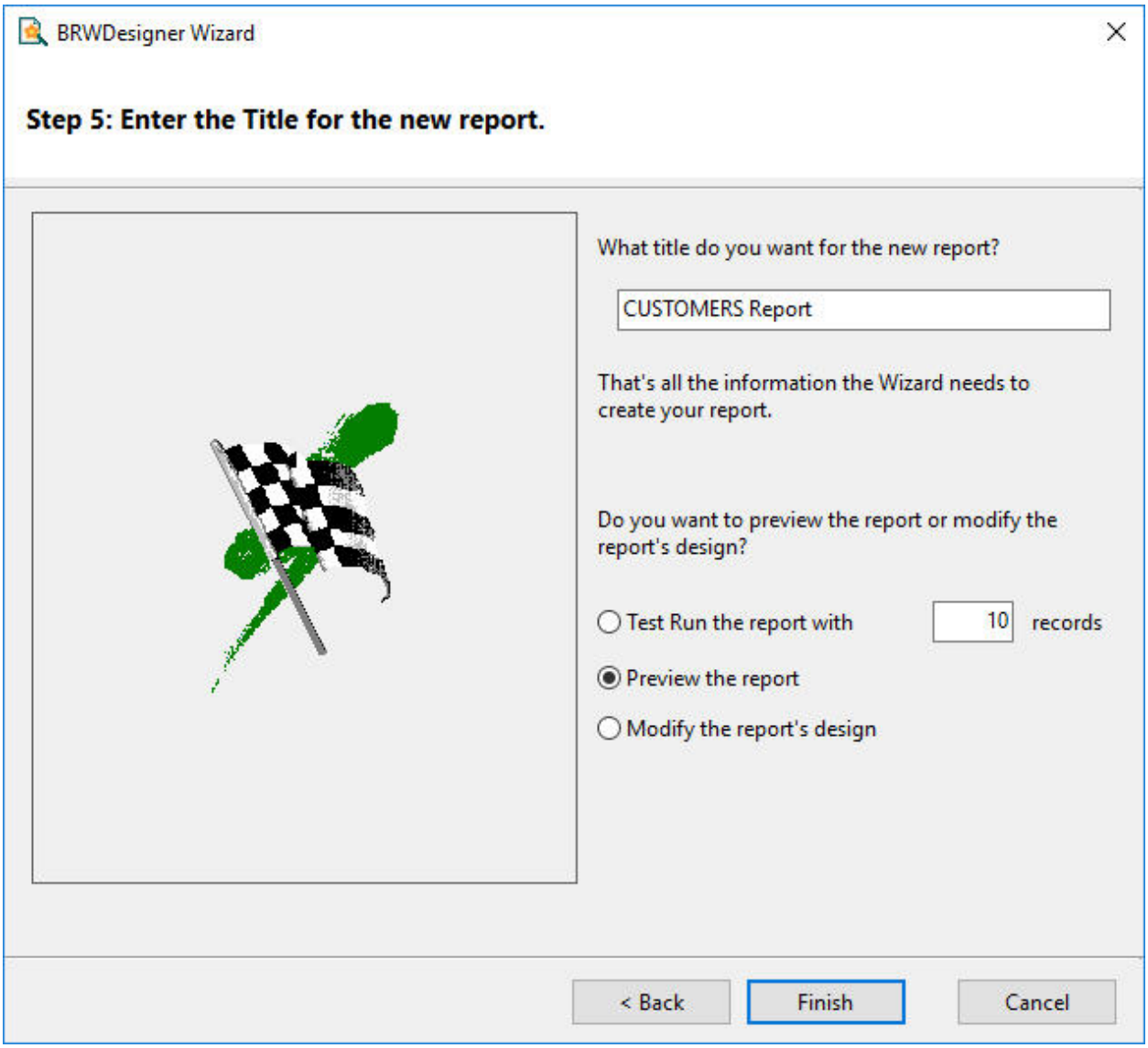
Select your report layout and click Next.



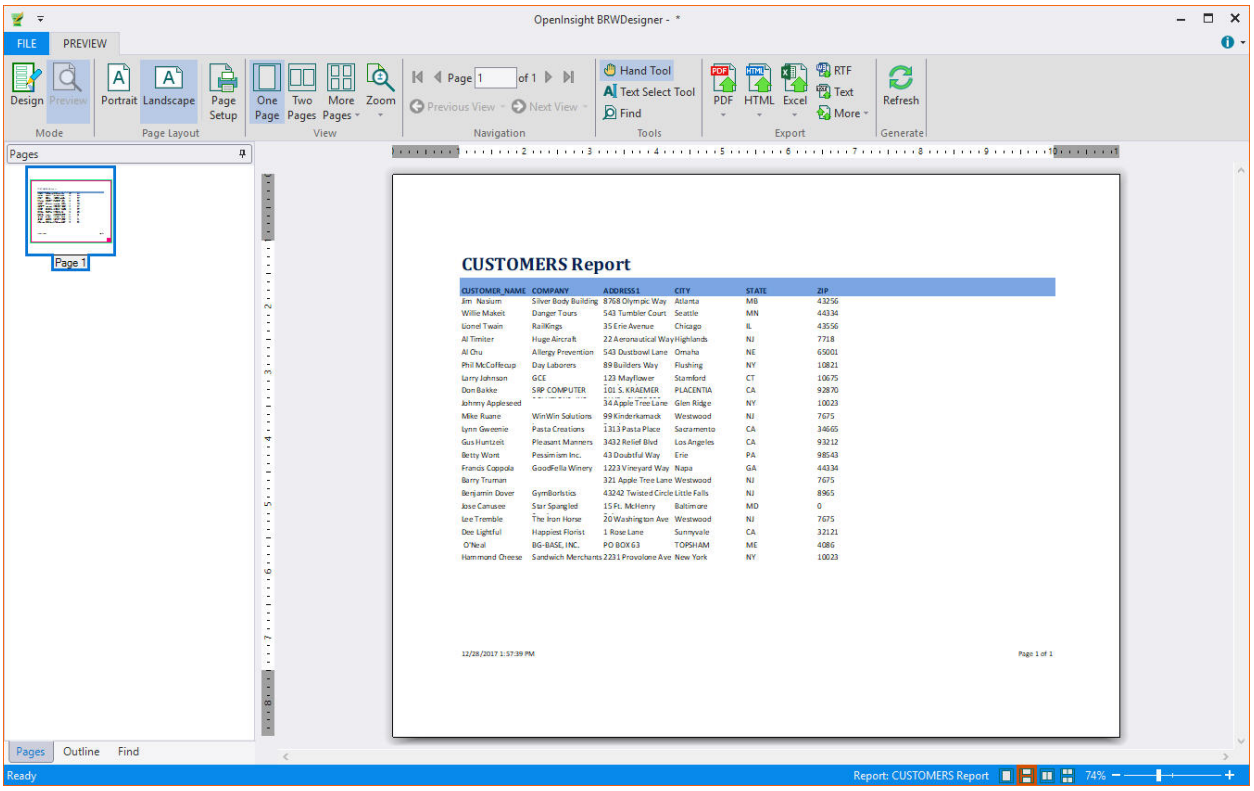
Select your report style and click Next.



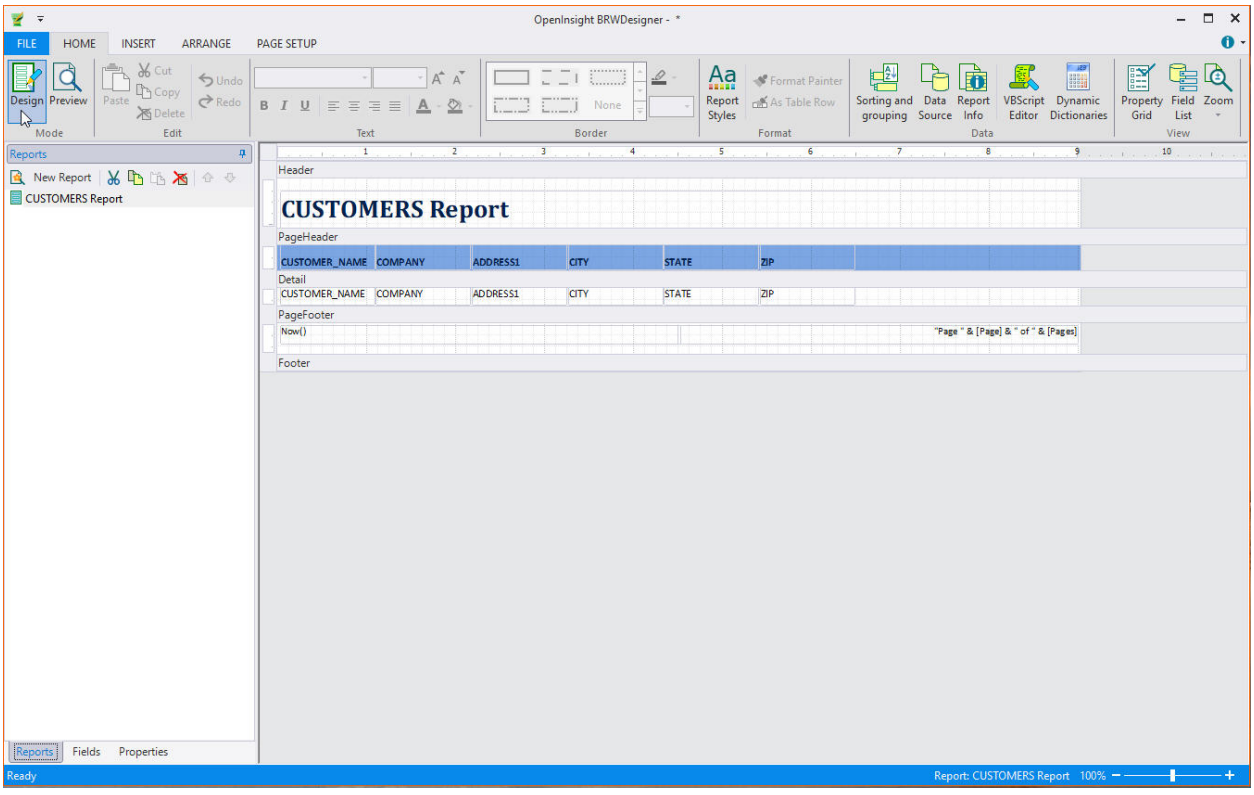
Enter your report title, select Preview the Report and click Finish.



Your completed report will display in the Print Preview window.




Click the Design button to return to Design mode.



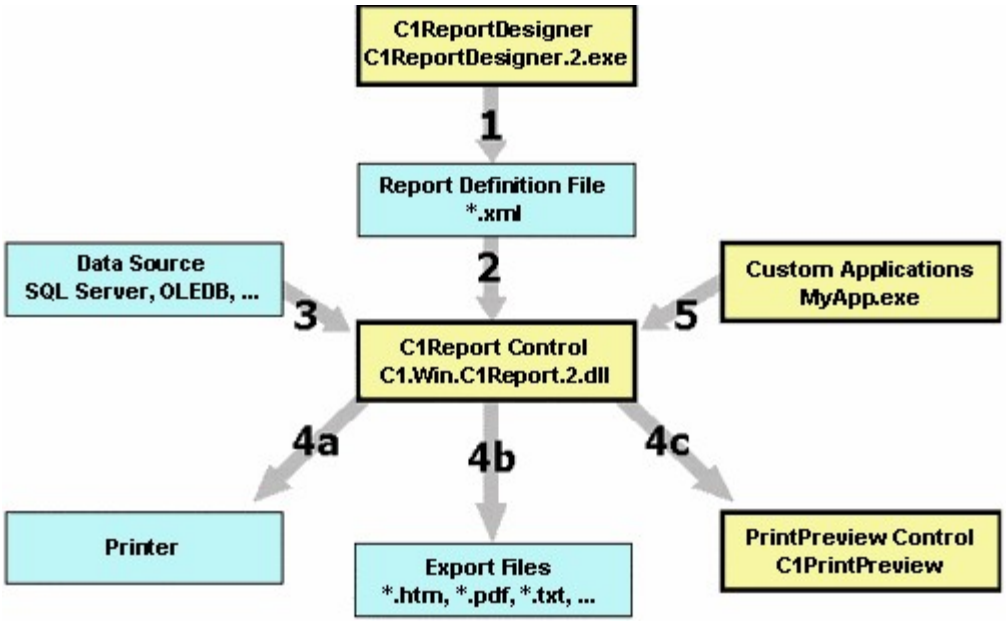
# Working with OI BRW

You can use OIBRW in many different scenarios, on the desktop and on the Web. The main sequence of steps is always the same:

1. You start by creating a report using the **OIBRWDesigner** application to create report definitions; report definitions are saved in XML files, and can be designed from scratch or imported from existing Microsoft Access reports. You can then modify the basic report using **OIBRWDesigner**.
2. The OIBRW component reads the report definitions and renders the reports using data from any standard .NET data source.
3. The report definitions can be loaded at design time, and embedded in your application, or they can be read and modified at run time. (You can even create report definitions from scratch, using the OIBRW object model.)
4. Reports can be rendered directly to a printer, into a **C1PrintPreview** control, or into HTML and PDF files that can be published on the Web.

The following diagram shows the relationship between the components in the **Reports for WinForms** package:

**Note:** Boxes with a bold border represent code components (controls and applications). Boxes with a thin border represent files finished reports).




The following numbers refer to the numbered arrows in the image, indicating relationships between the components:

1. Use the **OIBRWDesigner** application to create, edit, and save XML report definition files.
2. The OIBRW component loads report definitions from the XML files created with the Designer. This can be done at design time (in this case the XML file is persisted with the control and not needed at run time) or at run time using the Load method.
3. The OIBRW component loads data from the data source specified in the report definition file. Alternatively, you can provide your own custom data source.
4. The OIBRW component formats the data according to the report definition and renders reports to a (a) printer, (b) to one of several file formats, or (c) to a print preview control.
5. Custom applications can communicate with the OIBRW component using a rich object model, so you can easily customize your reports or generate entirely new ones. **OIBRWDesigner** is a good example of such an application.

## See Also

Object Model Summary
Sections of a Report
Developing Reports for Desktop Scenarios
Developing Reports for Web Scenarios
Creating, Loading, and Rendering the Report

## *Object Model Summary*

Working with OIBRW > Object Model Summary

The object model for the OIBRW component is largely based on the Microsoft Access model, except that Access has different types of controls (label control, textbox control, line control, and so on), while OIBRW has a single Field object with properties that can make it look like a label, textbox, line, picture, subreport, and so on.

The following table lists all objects, along with their main properties and methods (note that OIBRW uses *twips* for all measurements. One *twip* is 1/20 point, so 72 points = 1440 *twips* = 1 inch):

| |
|---|
| **OIBRW Object:** the main component |
| ReportName, GetReportInfo, Load, Save, Clear, Render, RenderToFile, RenderToStream, PageImages, Document, DoEvents, IsBusy, Cancel, Page, MaxPages, Font, OnOpen, OnClose, OnNoData, OnPage, OnError, Evaluate, Execute |
| **Layout Object:** determines how the report will be rendered on the page |
| Width, MarginLeft, MarginTop, MarginRight, MarginBottom, PaperSize, Orientation, Columns, ColumnLayout, PageHeader, PageFooter, Picture, PictureAlign, PictureShow |
| **DataSource Object:** manages the data source |
| ConnectionString, RecordSource, Filter, MaxRecords, Recordset |
| **Groups Collection:** a report may have many groups |
| **Group Object:** controls data sorting and grouping |
| Name GroupBy, Sort, KeepTogether, SectionHeader, SectionFooter, Move |
| **Sections Collection:** all reports have at least 5 sections |
| **Section Object:** contains Field objects (also known as "report band") |
| Name, Type, Visible, BackColor, OnFormat, OnPrint, Height, CanGrow, CanShrink, Repeat, KeepTogether, ForcePageBreak |
| **Fields Collection:** a report usually has many Fields |
| **Field Object:** a rectangular area within a section where information is displayed |
| Name, Section, Text, TextDirection, Calculated, Value, Format, Align, WordWrap, Visible, Left, Top, Width, Height, CanGrow, CanShrink, Font, BackColor, ForeColor, BorderColor, BorderStyle, LineSlant, LineWidth, MarginLeft, MarginRight, MarginTop, MarginBottom, LineSpacing, ForcePageBreak, HideDuplicates, RunningSum, Picture, PictureAlign, Subreport, CheckBox, RTF |

## *Sections of a Report*



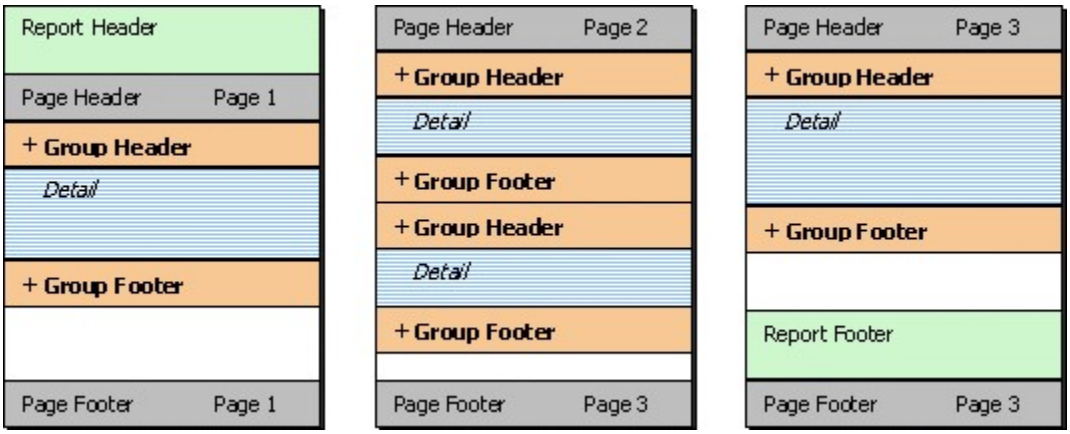
Every report has at least the following five sections:

| Section | Description |
|---|---|
| Detail | The Detail section contains fields that are rendered once for each record in the source recordset. |
| Header | The Report Header section is rendered at the beginning of the report. |
| Footer | The Report Footer section is rendered at the end of the report. |
| Page Header | The Page Header section is rendered at the top of every page (except optionally for pages that contain the Report Header). |
| Page Footer | The Page Footer section is rendered at the bottom of every page (except optionally for pages that contain the Report Footer). |

In addition to these five sections, there are two additional sections for each group: a Group Header and a Group Footer Section. For example, a report with 3 grouping levels will have 11 sections.

Note that sections can be made invisible, but they cannot be added or removed, except by adding or removing groups.

The following diagram shows how each section is rendered on a typical report:




## Report Header

The first section rendered is the Report Header. This section usually contains information that identifies the report.

## Page Header

After the Report Header comes the Page Header. If the report has no groups, this section usually contains labels that describe the fields in the Detail Section.

## Group Headers and Group Footers

The next sections are the Group Headers, Detail, and Group Footers. These are the sections that contain the actual report data. Group Headers and Footers often contain aggregate functions such as group totals, percentages, maximum and minimum values, and so on. Group Headers and Footers are inserted whenever the value of the expression specified by the GroupBy property changes from one record to the next.

## Detail

The Detail section contains data for each record. It is possible to hide this section by setting its Visible property to **False**, and display only Group Headers and Footers. This is a good way to create summary reports.

## Page Footer

At the bottom of each page is the Page Footer Section. This section usually contains information such as the page number, total number of pages in the report, and/or the date the report was printed.

## Report Footer

Finally, the Report Footer section is printed before the last page footer. This section is often used to display summary information about the entire report.

## Customized sections

You can determine whether or not a section is visible by setting its Visible property to **True** or **False**. Group Headers can be repeated at the top of every page (whether or not it is the beginning of a group) by setting their Repeat property to **True**. Page Headers and Footers can be removed from pages that contain the Report Header and Footer sections by setting the PageHeader and PageFooter properties on the Layout object.

## *Grouping and Sorting Data*

Working with OIBRW > Grouping and Sorting Data

Show All

This section shows how you can organize the data in your report by grouping and sorting data, using running sums, and creating aggregate expressions.

### Grouping Data:

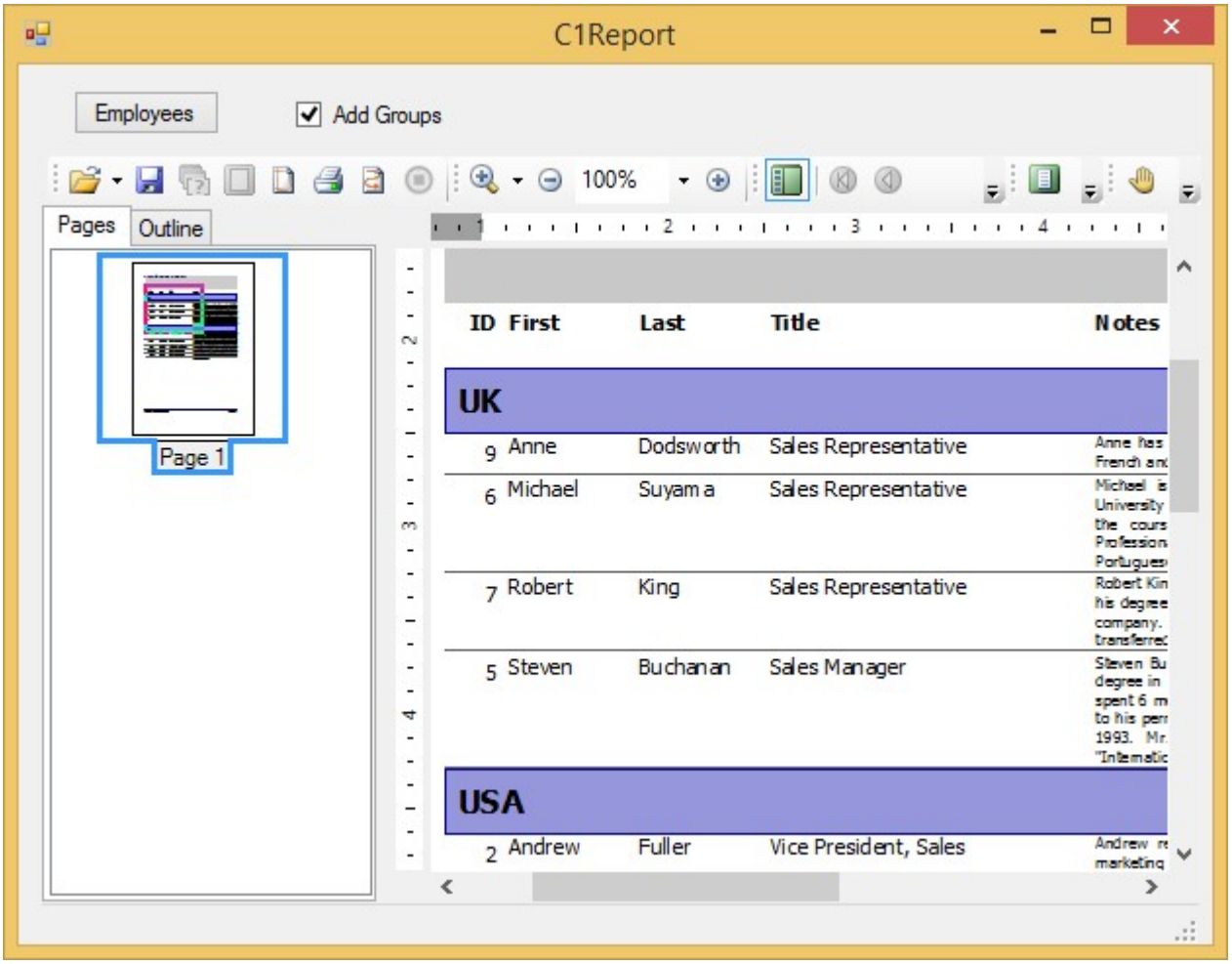
After designing the basic layout, you may decide that grouping the records by certain fields or other criteria would make the report easier to read. Grouping allows you to separate groups of records visually and display introductory and summary data for each group. The group break is based on a grouping expression. This expression is usually based on one or more recordset fields but it can be as complex as you like.

You can group the data in your reports using the **OIBRWDesigner** application or using code:

Adding grouping and sorting using OIBRWDesigner

Adding grouping and sorting using code

Here's an example of a report using a group object:



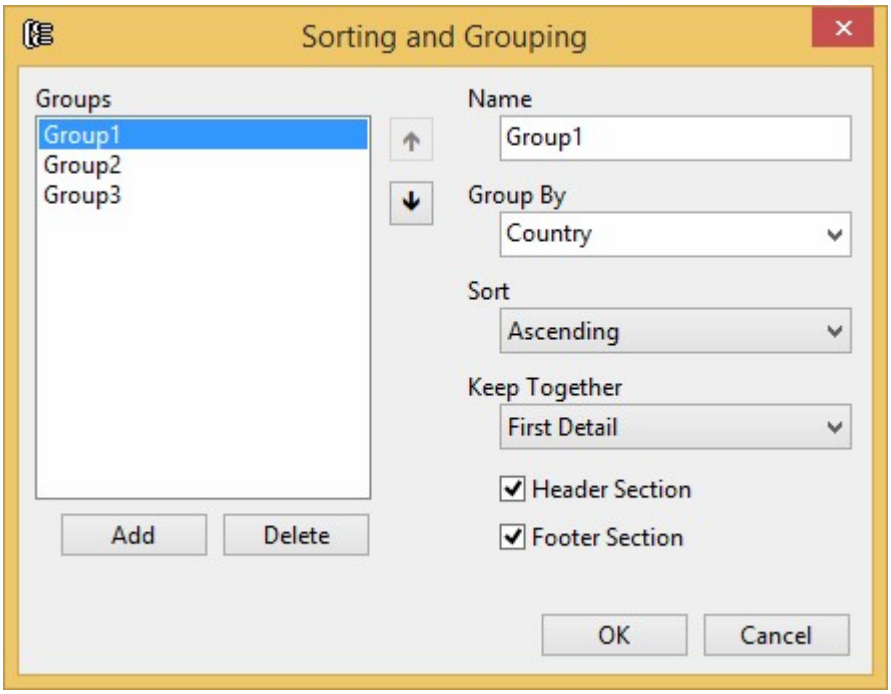
## Sorting Data:

You can sort data in reports the following two ways:

- Sort the data source object itself (for example, using a SQL statement with an ORDER BY clause).
- Add groups to the report and specify how each group should be sorted using the group's GroupBy and Sort properties.

Group sorting is done using the **DataView.Sort** property, which takes a list of column names only (not expressions on column names). So if your grouping expression is DatePart("yyyy", dateColumn), the control will actually sort on the dates in the *dateColumn* field, not on the years of those dates as most would expect.
To sort based on the dates, add a calculated column to the data table (by changing the SQL statement), and then group/sort on the calculated column instead. See the Sort property for an XML discussion of this, including a sample.

This is what the **Sorting and Grouping** editor looks like in the **OIBRWDesigner** application. Note the fields where you can specify group sorting:




If you use both approaches, the sorting set in the report groups will prevail (it is applied after the data has been retrieved from the database).

# Adding Running Sums



OIBRW field objects have a RunningSum property that makes it easy to maintain running sums over groups or over the entire report.

## Adding Running Sums over a Group

To keep running sums over groups, complete the following tasks:

1. Open the **OIBRWDesigner** application. For more information on how to access the **OIBRWDesigner**, see Accessing OIBRWDesigner from Visual Studio.
2. Create a new report or open an existing report. Once you have the report in the **OIBRWDesigner** application, you can modify the report properties.
3. Click the **Design** button to begin editing the report.
4. In Design mode, select the report from the drop-down list above the Properties window. The available properties for the report appear.
5. Add a calculated field to the report:
    1. Click the **Add Calculated Field** button from the Designer toolbar.
    2. In the **VBScript Editor**, enter the following script:
       Sum(ProductSalesCtl)
    3. Drag the mouse over the GroupHeader section of the report and the cursor changes into a cross-hair  . Click and drag to define the rectangle that the new field will occupy, and then release the button to create the new field.
6. Set the RunningSum property to **SumOverGroup**. (Note that for this property to appear, the properties filter must be turned off. It's the funnel icon above the Properties window.)

## Adding Running Sums over the Entire Report

To keep running sums over pages, you need to use script. For example, you could add a **pageSum** field to the report and update it with script. To do this, complete the following tasks:

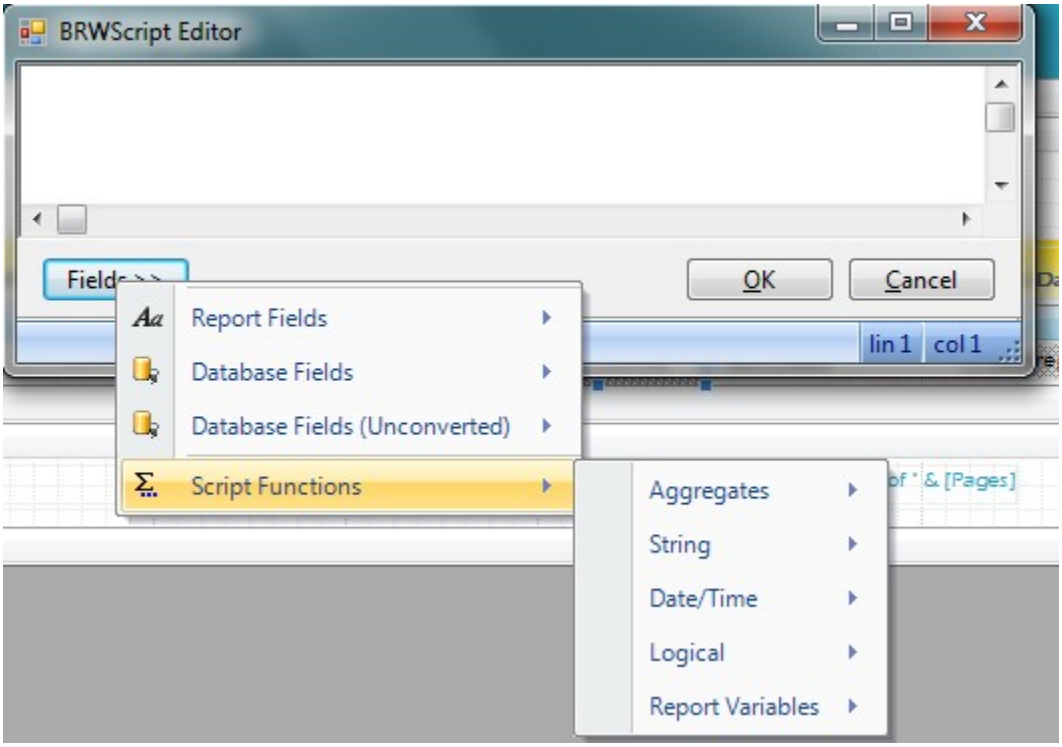
1. Open the **OIBRWDesigner** application. For more information on how to access the **OIBRWDesigner**, see Accessing OIBRWDesigner from Visual Studio.
2. Create a new report or open an existing report. Once you have the report in the **OIBRWDesigner** application, you can modify the report properties.
3. Click the **Design** button to begin editing the report.
4. In Design mode, select the report from the drop-down list above the Properties window. The available properties for the report appear.
5. Locate the OnPage property and click the empty field next to it, and then click the **ellipsis** button.
6. The **VBScript Editor** appears. Enter the following VBScript expression in the code editor: ' VBScript:
   Report.OnPage
   pageSum = 0
7. Then select **Detail** from the drop-down list above the Properties window. The available properties for the Detail section appear.
8. Locate the OnPrint property and click the empty field next to it, and then click the **ellipsis** button.
9. The **VBScript Editor** appears. Enter the following VBScript expression in the code editor: ' VBScript:
   Detail.OnPrint
   pageSum = pageSum + UnitsInStock

## Adding Subtotals and Other Aggregates

OI BRW supports aggregate expressions in all its calculated fields. The aggregate expressions include all the usual **Sum**, **Avg**, **Min**, **Max**, **Count**, **Range**, **StDev**, and so on.

A Calculated field can be added to a report by clicking on the Calculated Column icon in the **Fields** tab, as seen below:


Once the icon is clicked, the script editor window appears. Clicking on the **Fields** button causes menus to appear.




All aggregate functions take an expression as an argument and evaluate it within a scope that is determined by their position in the report. For example, aggregates in group headers or footers have the scope of the group. Aggregates in the report header or footer have the scope of the entire report.

For example, the following aggregate expression would return the sum of all values in the *Invoice_Total* field for the scope of the aggregate (group or report):

```
Sum(Invoice_Total)
```

The following aggregate expression would return the total amount of sales taxes paid for all values in the report (assuming an 8.5% sales tax):

```
Sum(Invoice_Total * 0.085)
```

You can reduce the scope of any aggregate using a second argument called *domain*. The *domain* argument is an expression that determines whether each value in the current scope should be included in the aggregate calculation.

For example, the following aggregate expression would return the sum of all values in the *Invoice_Total* field for products in category 1:

```
Sum(Invoice_Total, Category = 1)
```

The following aggregate expression would return the number of sales over $10,000:

```
Count(*, Invoice_Total > 10000)
```

## Counting items in a Group or Report

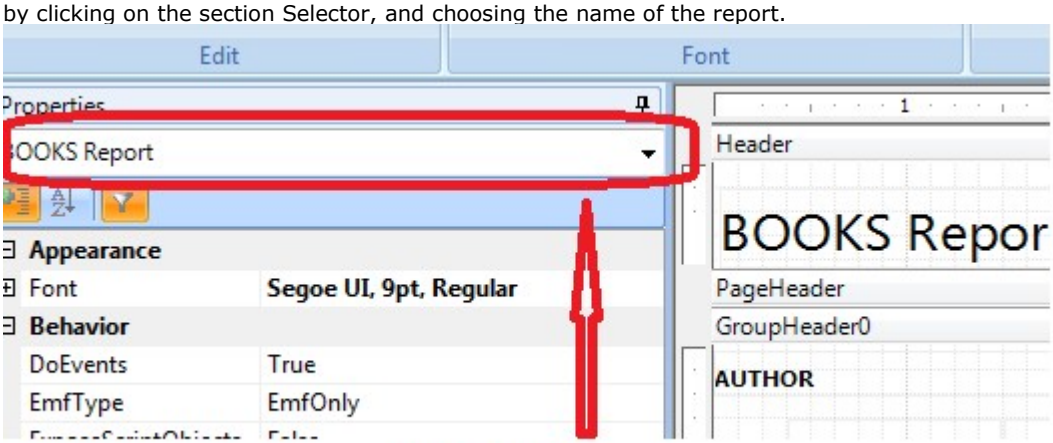
In certain cases you will want to get a running number of items in a group, or over an entire report

1. In the BRW designer, create a calculated field, and in the script editor just put in "1" as the expression (not in quotes - just the number 1);
2. Draw in the calculated field wherever you want the incrementing counter to go on the report;
3. In the properties for that calculated field, set the "RunningSum" property to "SumOverGroup"

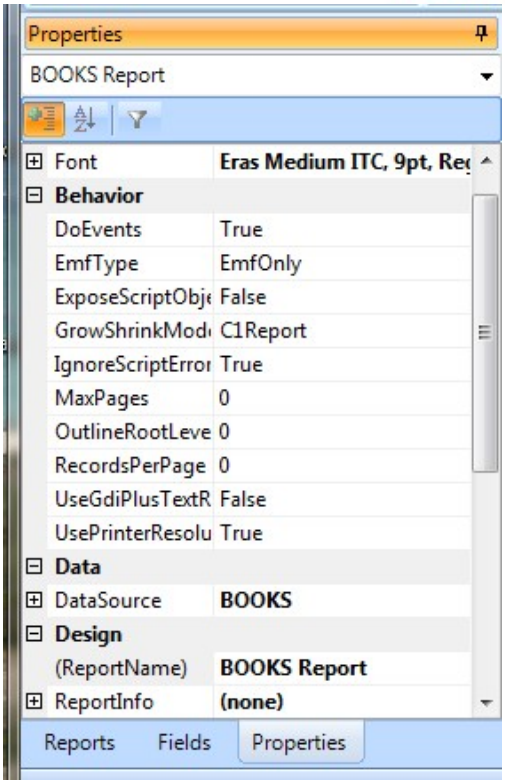
## Selecting Rows

A report rarely ever wants to display all the records in a table. Usually, users require a subset of the data in the report, based upon some selection criteria. The OI BRW allows such functionality.

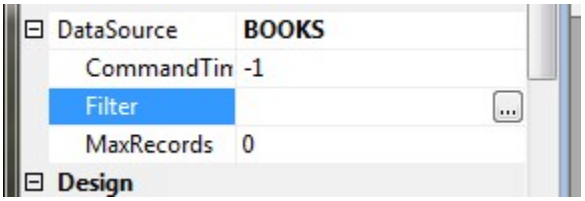
With the report in design mode, click on the property pane for the whole report, as seen below. This can be done by clicking on the section Selector, and choosing the name of the report.



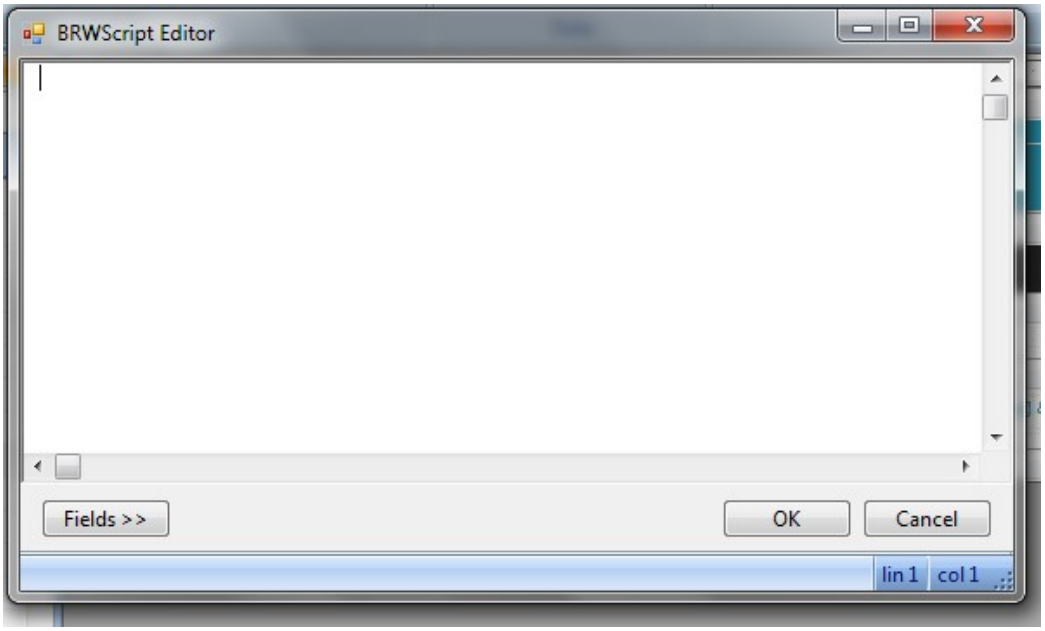

Section Selector



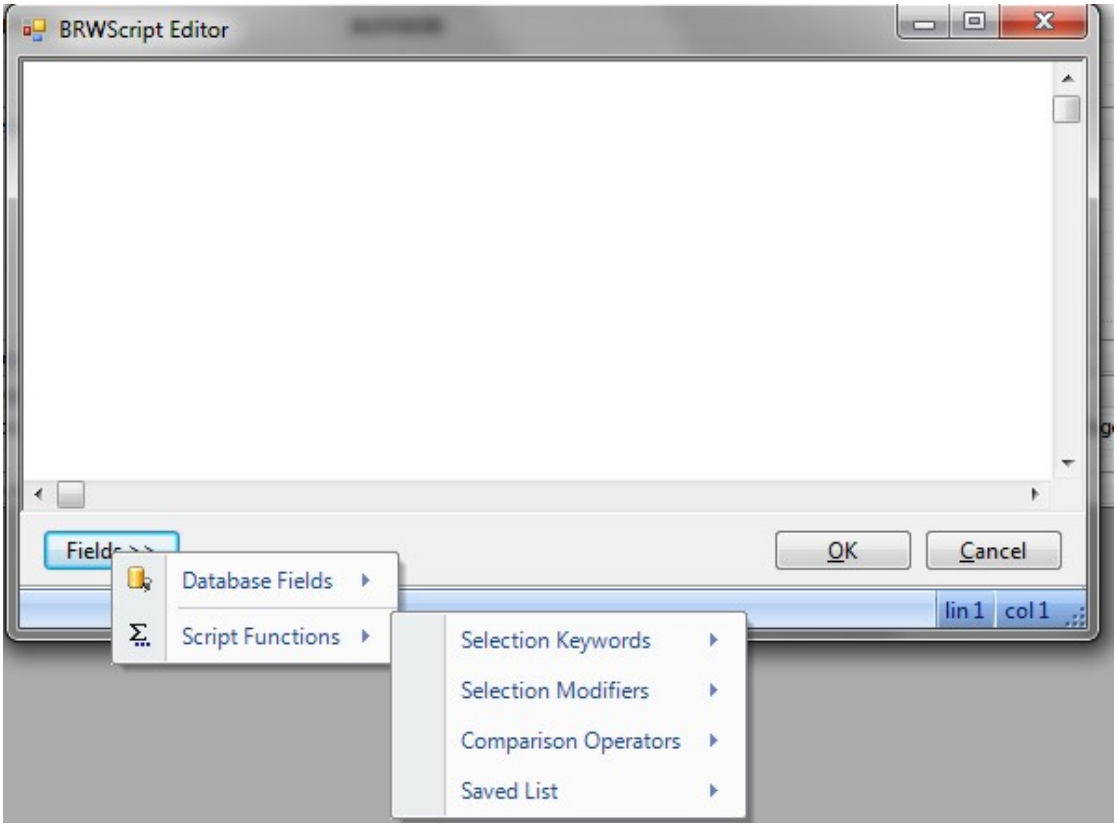

Expand the 'DataSource' section by clicking the '+' and see the 'Filter' option, with the button with an ellipsis on it.



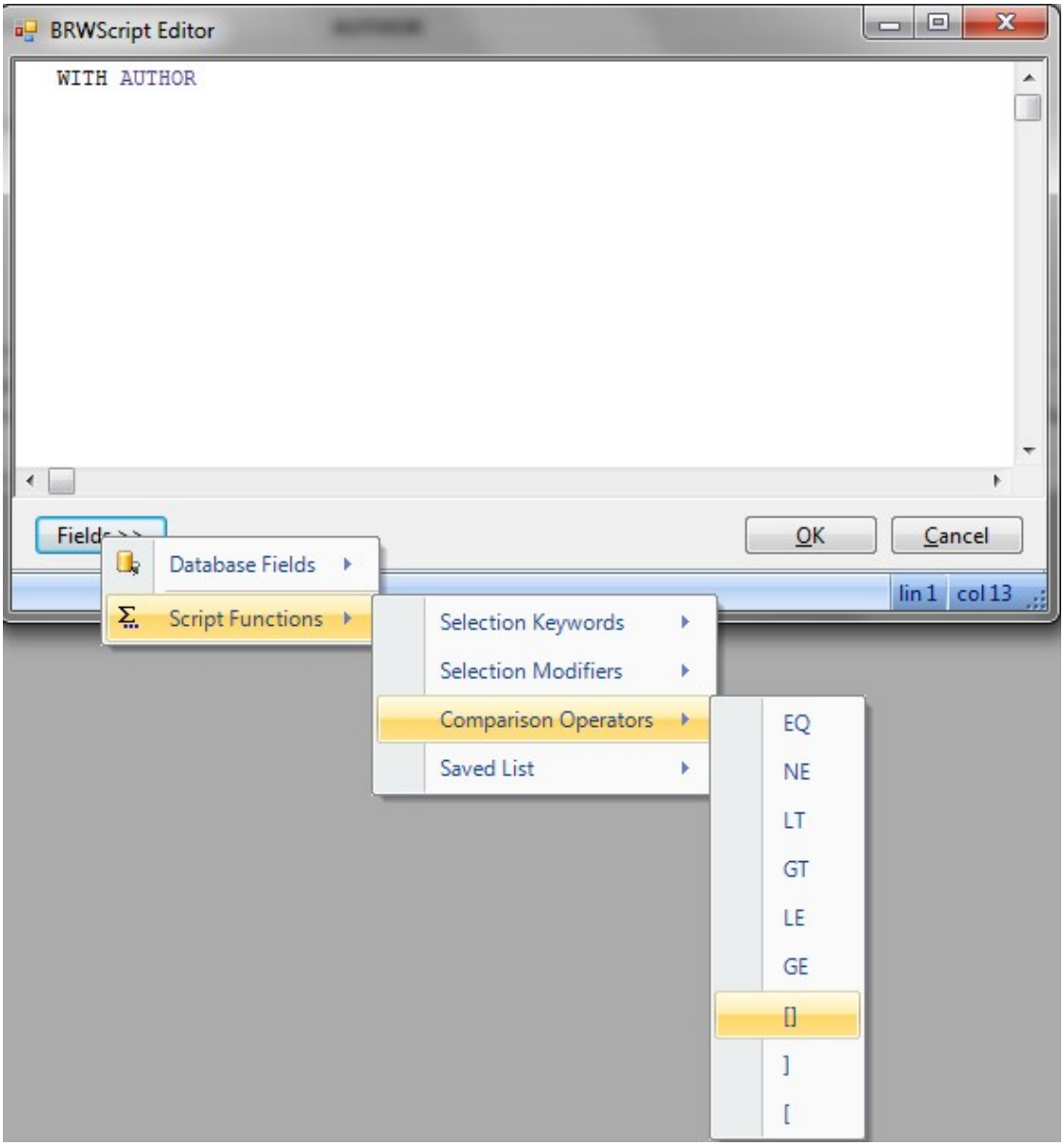

Clicking on that button will bring up the BRW Script editor window.



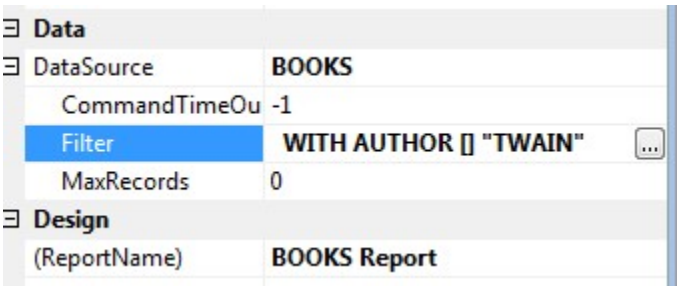

In this window, by clicking on the "Fields >>" button, the user can choose from either list of database fields and operators and build the select statement for the report.




In the image below, we've already chosen the selection keyword "WITH', and then chosen the database field 'AUTHOR', and are about to select the operator for 'Containing'




We'll complete our selection criteria by searching on Author containing the word "TWAIN" and press the OK button. At this point, the property pane for the report shows the filter we have specified.




Running the report at this point gives us a list of all books whose author contains the work "Twain".

The select statement may also be entered in manually, rather than building it using the "Fields>>" button.

Specifying MaxRecords with a value greater than zero will limit the number of records returned to the value entered.

OpenInsight 9.3.1 and above includes the following enhancements:

- When specifying a filter with a prompt (WITH AUTHOR [] "?Enter Author Name?"), if the same prompt is specified more than one time in the filter, the user is not re-prompted – the already-entered value is repeated.
- When specifying a filter with a prompt, it is now possible to specify the default response – use a pipe, and then the suggested default (WITH AUTHOR [] "?Enter Author Name|Twain?").
- A user-specified function can now be called to generate the list of matching IDs – specify CALL <functionname> (and then any other optional selection filter, including prompted values), and <functioname> will be called – we expect a list of IDs (@FM delimited) as the response to the call. For example:

  CALL SELECTME WITH AUTHOR [] "?Enter Author Name|Twain?"

  Will show the prompt "Enter Author Name" with the default value of Twain, and then we will senf to the function SELECTME, as parameter #1, the string WITH AUTHOR [] "Dickens" (if the user entered Dickens). If this was a test run of 10 records, then parameter#1 will be 10 WITH AUTHOR [] "Dickens"

## *Working with VBScript*

Working with OIBRW > Working with VBScript

VBScript expressions are widely used throughout a report definition to retrieve, calculate, display, group, sort, filter, parameterize, and format the contents of a report. Some expressions are created for you automatically (for example, when you drag a field from the Toolbox onto a section of your report, an expression that retrieves the value of that field is displayed in the text box). However, in most cases, you create your own expressions to provide more functionality to your report.

Note the following differences between VBScript expressions and statements:

- **Expressions** return values, you can assign them to things like **Field.Text**, for example:
  Field1.Calculated = true
  Field1.Text = "iif( 1=1, 1+2, 1+3 )"
- **Statements** don't return values. You can assign them to event properties like **OnFormat**. For example:
  c1r.OnOpen = "if 1=1 then msgbox("OK!!!") else msgbox("ooops")"

OIBRW relies on VBScript to evaluate expressions in calculated fields and to handle report events.

VBScript is a full-featured language, and you have access to all its methods and functions when writing OIBRW expressions. For the intrinsic features of the VBScript language, refer to the Microsoft Developer's Network (MSDN).

Global Scripts can be written in the new VBScript Editor. This editor allows users to define VBScript functions and subroutines that are accessible throughout the report. To directly access the VBScript Editor, press **F7** and to close the editor and save the changes, use the shortcut key **Ctrl+W**. Users can switch between scripts and also change options such as fonts or colors within the editor. The editor also makes the scripting experience intuitive and easy for developers with advanced features such as syntax check, pre-defined VBScript functions, and rearranged scripting functions.

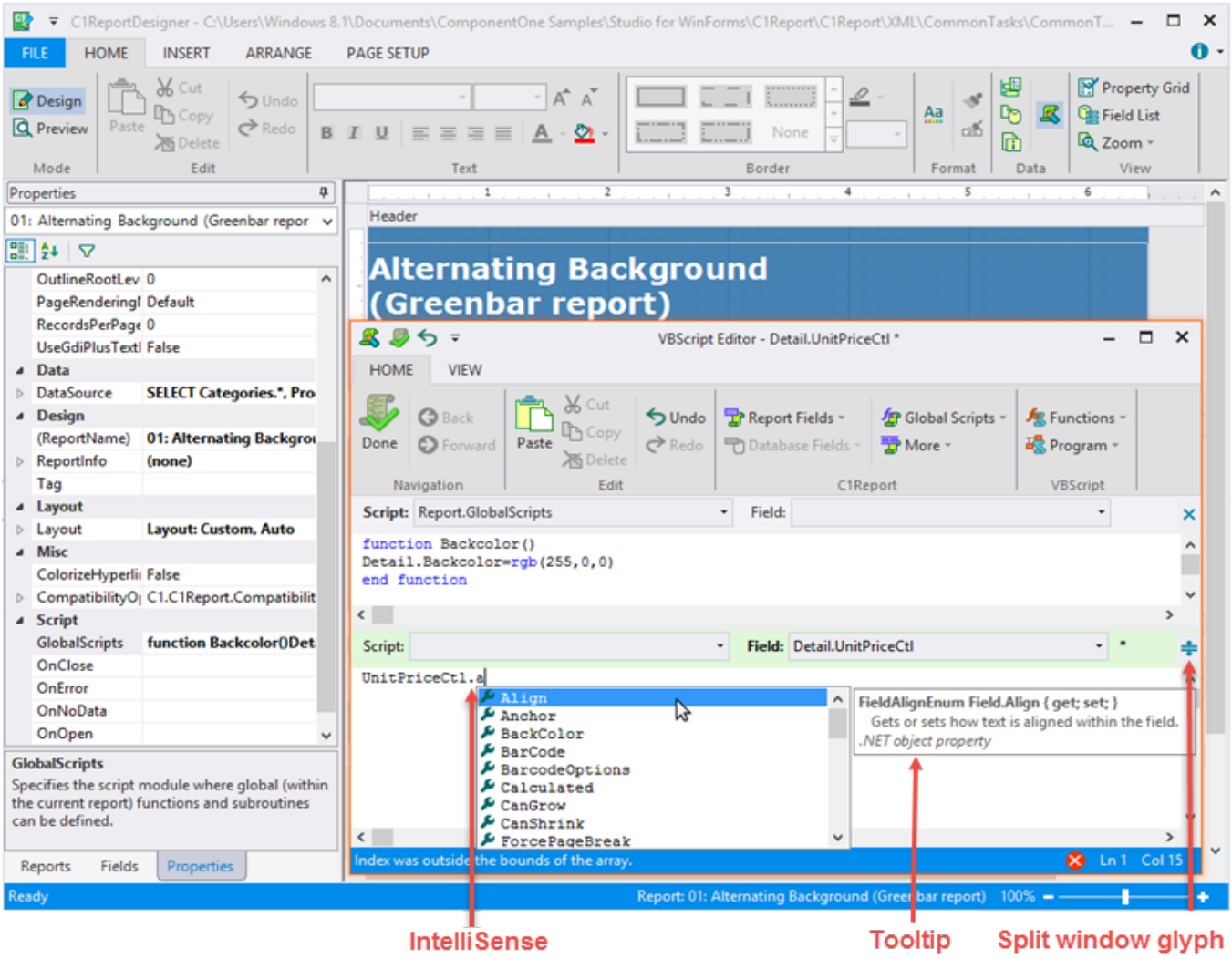
To write global scripts using **VBScript Editor** option,

1. Go to Home Tab of OIBRWDesigner.
2. Click **VBScript Editor** and write desired global script; for example,

```
function Backcolor()
Detail.Backcolor=rgb(255,0,0)
end function
```

You can also write global scripts using **GlobalScripts** property of **OIBRWDesigner** as follows:

1. Select the report in which you want to write global script.
2. Go to the GlobalScripts property of the report and then click ellipses. This opens **VBScript Editor** dialog box.
3. Write the global script as above, in the **VBScript Editor**.
4.




So, you have defined a global function 'Backcolor()', which can be used throughout the report.
The VBScript Editor has the following additional features:

- **IntelliSense**: Provides auto code completion prompts for the scripts supported by VBScript Editor. IntelliSense in VBScript Editor has following features:
  - The IntelliSense window that displays a context-dependent list of available words also displays a detailed help on VBScript functions and keywords in a small tooltip or help window. The italic font on the detailed help basically shows the category to which the current item belongs (such as 'VBScript function', 'OIBRW aggregate script function', .NET object property, and so on).
  - On editing DataSource.Filter, the editor opens as **Expression Editor - DataSource.Filter** and IntelliSense shows keywords or functions available in that with corresponding help.
  - Icons associated with IntelliSense entries indicate the type of the entry. The icons' color palette is different for VBScript, report built-in stuff, and DataSource.Filter.
  - When a user types and Intellisense window is opened, the list is filtered according to the letters being typed for example, typing 't' will only show words that contain the letter 't', typing 'te' will narrow the list to words that contain 'te', and so on.
  - Backspace in the IntelliSense window undoes the last filter.
  - Pressing square bracket '**[**' shows the list of available db fields.
  - Pressing dot '**.**' after the name of an object such as a report, field, or section shows the .NET properties available for that object
  - Pressing Ctrl+J, Ctrl+Space, or a letter after a non-letter character shows the list of available VBScript functions, keywords, etc. depending on the context.
- **Split Window**: Lets you view or write two same or different scripts in single VBScriptEditor. By default, the VBScript editor opens as a single window.

**To switch to Split Window**

Switch to the split window mode by clicking the split window glyph and dragging it down to open another editor at

the top. The windows can be resized by dragging the divider between the windows.

**To switch back to the single window**

Click the 'x' glyph on the top right corner of the window to close the top window, turn the split mode off, and zoom out the bottom window. The enabled or disabled state of ribbon buttons depends on the current window, which is shown with a light green caption bar. The split window mode has following additional functionalities:

- Switch between the two windows by pressing **F6**.
- Hide the top window in a split window mode by dragging the split window glyph or the divider line high enough across the top window.

 Note that **Global Scripts** dropdown in **VBScript Editor** is enabled only if you have previously defined global script(s) in your r

**OIBRW** extends VBScript by exposing additional objects, variables, and functions. These extensions are described in the sections listed in See Also.

**See Also**

VBScript Elements, Objects, and Variables
Using Compatibility Functions: Iif and Format
Using Aggregate Functions

# VBScript Elements, Objects, and Variables

The following tables detail VBScript elements, objects, and variables.

## Operators

The following table contains the VBScript operators:

| Operator | Description |
|---|---|
| And | Performs a logical conjunction on two expressions. |
| Or | Performs a logical disjunction on two expressions. |
| Not | Performs a logical disjunction on two expressions. |
| Mod | Divides two numbers and returns only the remainder. |

## Reserved symbols

The following table contains the VBScript reserved symbols and how to use them:

| Keyword | Description |
|---|---|
| True | The **True** keyword has a value equal to -1. |
| False | The **False** keyword has a value equal to 0. |
| Nothing | Used to disassociate an object variable from any actual object. To assign **Nothing** to an object variable, use the **Set** statement, for example:<br>`Set MyObject = Nothing`<br>Several object variables can refer to the same actual object. When **Nothing** is assigned to an object variable, that variable no longer refers to any actual object. When several object variables refer to the same object, memory and system resources associated with the object to which the variables refer are released only after all of them have been set to **Nothing**, either explicitly using **Set**, or implicitly after the last object variable set to **Nothing**. |
| Null | The **Null** keyword is used to indicate that a variable contains no valid data. |
| vbCr | When you call print and display functions, you can use the following constants in your code in place of the actual values. |
| vbCrLf | When you call print and display functions, you can use the following constants in your code in place of the actual values. |
| vbLf | When you call print and display functions, you can use the following constants in your code in place of the actual values. |
| vbFormFeed | When you call print and display functions, you can use the following constants in your code in place of the actual values. |
| vbNewLine | When you call print and display functions, you can use the following constants in your code in place of the actual values. |
| vbNullChar | When you call print and display functions, you can use the following constants in your code in place of the actual values. |
| vbTab | When you call print and display functions, you can use the following constants in your code in place of the actual values. |
| vbVerticalTab | When you call print and display functions, you can use the following constants in your code in place of the actual values. |
| vbBlack | Black. Value = 0x0. |
| vbRed | Red. Value = 0xFF. |
| vbGreen | Green. Value = 0xFF00. |

| | |
|---|---|
| vbYellow | Yellow. Value = 0xFFFF. |
| vbBlue | Blue. Value = 0xFF0000. |
| vbMagenta | Magenta. Value = 0xFF00FF. |
| vbCyan | Cyan. Value = 0xFFFF00. |
| vbWhite | White. Value = 0xFFFFFF. |

### Built-in functions

The VBScript built-in functions are listed below:

| | | | | |
|---|---|---|---|---|
| Abs | Date | Iif | Minute | Sign |
| Acos | DateAdd | InputBox | Month | Space |
| Asc | DateDiff | InStr | MonthName | Sqr |
| Asin | DatePart | InStrRev | MsgBox | StrComp |
| Atn | DateSerial | Int | Now | String |
| CBool | DateValue | IsDate | Oct | Tan |
| CByte | Day | IsEmpty | Pi | Time |
| CCur | Exp | IsNull | Replace | Timer |
| CDate | Fix | IsNumeric | RGB | TimeSerial |
| CDbl | Format | IsObject | Right | TimeValue |
| Chr | FormatCurrency | LCase | Rnd | Trim |
| CInt | FormatDateTime | Left | Round | TypeName |
| CLng | FormatNumber | Len | RTrim | UCase |
| Cos | FormatPercent | Log | Second | WeekDay |
| CSng | Hex | LTrim | Sgn | WeekDayName |
| CStr | Hour | Mid | Sin | Year |

For more information on the VBScript functions, see the MSDN documentation. Note that the following VBScript features are **not** supported in OI BRW:

- Arrays
- Functions/Subs
- Select/Case statements

Also note that the following OI BRW features are **not** part of VBScript:

- Aggregate functions (Sum, Average, StDev, Var, Count, and so on)
- Report and Database field names
- Page/Pages variables
- Report object

### Statement keywords

The VBScript statement keywords are listed below:

| | | | | |
|---|---|---|---|---|
| If | ElseIf | To | While | Dim |
| Then | EndIf | Next | Wend | Redim |
| Else | For | Step | Const | |

### Report Field Names

Names of Field objects are evaluated and return a reference to the object, so you can access the field's properties. The default property for the Field object is Value, so by itself the field name returns the field's current value. For example:

- Visual Basic
- C#

**Note:** If you give a report field the same name as a database field, you won't be able to access the report field.

### Report Section Names

Names of Section objects are evaluated and return a reference to the object, so you can access the section's properties. The default property for the Section object is Name. For example:

- Visual Basic
- C#

### Database Field Names

Names of fields in the report's dataset source are evaluated and return the current field value. If a field name contains spaces or periods, it must be enclosed in square brackets. For example:

```
OrderID
UnitsInStock
[Customer.FirstName]
[Name With Spaces]
```

### Report Variables

#### Page

The page variable returns or sets the value of the Page property. This property is initialized by the control when it starts rendering a report, and is incremented at each page break. You may reset it using code. For example:

- Visual Basic
- C#

#### Pages

The pages variable returns a token that gets replaced with the total page count when the report finishes rendering. This is a read-only property that is typically used in page header or footer fields. For example:

- Visual Basic
- C#

### Report Object

The report object returns a reference to the control object, so you can access the full OI BRW object model from your scripts and expressions. For example:

- Visual Basic
- C#

### Cancel

Set **Cancel** to **True** to cancel the report rendering process. For example:

- Visual Basic
- C#

## Using Compatibility Functions: Iif and Format

To increase compatibility with code written in Visual Basic and Microsoft Access (VBA), OI BRW exposes two functions that are not available in VBScript: **Iif** and **Format**.

**Iif** evaluates a Boolean expression and returns one of two values depending on the result. For example:

```
Iif(Invoice_TotalAmount > 1000, "Yes", "No")
```

**Format** converts a value into a string formatted according to instructions contained in a format expression. The value may be a number, Boolean, date, or string. The format is a string built using syntax similar to the format string used in Visual Basic or VBA.

The following table describes the syntax used for the format string:

| Value Type | Format String | Description |
|---|---|---|
| **Number** | Percent, % | Formats a number as a percentage, with zero or two decimal places.<br>For example:<br>`Format(0.33, "Percent") = "33%"`<br>`Format(0.3333333, "Percent") = "33.33%"` |

| | | |
|---|---|---|
| | #,###.##0 | Formats a number using a mask. The following symbols are recognized:<br><br># digit placeholder<br>0 digit placeholder, force display<br>, use thousand separators<br>( enclose negative values in parenthesis<br>% format as percentage<br><br>For example:<br>`Format(1234.1234, "#,###.##") = "1,234.12"`<br>`Format(-1234, "#.00") = "(1234.12)"`<br>`Format(.1234, "#.##") = ".12"`<br>`Format(.1234, "0.##") = "0.12"`<br>`Format(.3, "#.##%") = "30.00%"` |
| **Currency** | Currency, $ | Formats a number as a currency value. Displays number with thousand separator, if appropriate; displays two digits to the right of the decimal separator.<br>For example:<br>`Format(1234, "$") = "$1,234.00"` |
| **Boolean** | Yes/No | Returns "Yes" or "No". |
| **Date** | Long Date | `Format(#12/5/1#, "long date") = "December 5, 2001"` |
| | Short Date | `Format(#12/5/1#, "short date") = "12/5/2001"` |
| | Medium Date | `Format(#12/5/1#, "medium date") = "05-Dec-01"` |
| | q,m,d,w,yyyy | Returns a date part (quarter, month, day of the month, week of the year, year).<br>For example:<br>`Format(#12/5/1#, "q") = "4"` |
| **String** | @@-@@/@@ | Formats a string using a mask. The "@" character is a placeholder for a single character (or for the whole value string if there is a single "@"). Other characters are interpreted as literals.<br>For example:<br>`Format("AC55512", "@@-@@/@@") = "AC-555/12"`<br>`Format("AC55512", "@") = "AC55512"` |
| | @;Missing | Uses the format string on the left of the semi-colon if the value is not null or an empty string, otherwise returns the part on the right of the semi-colon.<br>For example:<br>`Format("@;Missing", "UK") = "UK"`<br>`Format("@;Missing", "") = "Missing"` |

Note that VBScript has its own built-in formatting functions (**FormatNumber**, **FormatCurrency**, **FormatPercent**, **FormatDateTime**, and so on). You may use them instead of the VBA-style **Format** function described here.

## Using Aggregate Functions

Aggregate functions are used to summarize data over the group being rendered. When used in a report header field, these expressions return aggregates over the entire dataset. When used in group headers or footers, they return the aggregate for the group.

All OI BRW aggregate functions take two arguments:

- A string containing a VBScript expression to be aggregated over the group.
- An optional string containing a VBScript expression used as a filter (domain aggregate). The filter expression is evaluated before each value is aggregated. If the filter returns **False**, the value is skipped and is not included in the aggregate result.

OI BRW defines the following aggregate functions:

| Function | Description |
|---|---|
| Avg | Average value of the expression within the current group. For example, the |

| | |
|---|---|
| | following expressions calculate the average sales for the whole group and the average sales for a certain type of product:<br>`Avg(SalesAmount)`<br>`Avg(SalesAmount, ProductType = 3)` |
| Sum | Sum of all values in the group. |
| Count | Count of records in the group with non-null values. Use an asterisk for the expression to include all records. For example, the following expressions count the number of employees with valid (non-null) addresses and the total number of employees:<br>`Count(Employees.Address)`<br>`Count(*)` |
| CountDistinct | Count of records in the group with distinct non-null values. |
| Min, Max | Minimum and maximum values for the expression.<br>For example:<br>`"Min Sale = " & Max(SaleAmount)` |
| Range | Range between minimum and maximum values for the expression. |
| StDev, Var | Standard deviation and variance of the expression in the current group. The values are calculated using the sample (n-1) formulas, as in SQL and Microsoft Excel. |
| StDevP, VarP | Standard deviation and variance of the expression in the current group. These values are calculated using the population (n) formulas, as in SQL and Microsoft Excel. |

To use the aggregate functions, add a calculated field to a Header or Footer section, and assign the expression to the field's Text property.

## *Using Event Properties*

You are not restricted to using VBScript to evaluate expressions in calculated fields. You can also specify scripts that are triggered when the report is rendered, and you can use those to change the formatting of the report.These scripts are contained in *event properties*. An event property is similar to a Visual Basic event handler, except that the scripts are executed in the scope of the report rather than in the scope of the application that is displaying the report. For example, you could use an event property to set a field's Font and ForeColor properties depending on its value. This behavior would then be a part of the report itself, and would be preserved regardless of the application used to render it.

Of course, traditional events are also available, and you should use them to implement behavior that affects the application rather than the report. For example, you could write a handler for the StartPage event to update a page count in your application, regardless of which particular report is being rendered.

The following table lists the event properties that are available and typical uses for them:

| Object | Property | Description |
|---|---|---|
| Report | OnOpen | Fired when the report starts rendering. Can be used to modify the ConnectionString or RecordSource properties, or to initialize VBScript variables. |
| | OnClose | Fired when the report finishes rendering. Can be used to perform clean-up tasks. |
| | OnNoData | Fired when a report starts rendering but the source recordset is empty. You can set the **Cancel** property to **True** to prevent the report from being generated. You could also show a dialog box to alert the user as to the reason why no report is being displayed. |
| | OnPage | Fired when a new page starts. Can be used to set the **Visible** property of sections of fields depending on a set of conditions. The control maintains a Page variable that is incremented automatically when a new page starts. |
| | OnError | Fired when an error occurs. |
| Section | OnFormat | Fired before the fields in a section are evaluated. At this point, the fields in the source recordset reflect the values that will be rendered, but the report fields do not. |

| | | |
|---|---|---|
| | OnPrint | Fired before the fields in a section are printed. At this point, the fields have already been evaluated and you can do conditional formatting. |

The following topics illustrate typical uses for these properties.

---

- Formatting a Field According to Its Value
- Hiding a Section if There is No Data for It
- Showing or Hiding a Field Depending on a Value
- Prompting Users for Parameters
- Resetting the Page Counter
- Changing a Field's Dimensions to Create a Bar Chart

## Formatting a Field According to Its Value

Formatting a field according to its value is probably the most common use for the OnPrint property. Take for example a report that lists order values grouped by product. Instead of using an extra field to display the quantity in stock, the report highlights products that are below the reorder level by displaying their name in bold red characters.

### To highlight products that are below the reorder level using the OI BRWDesigner:

Alternatively, instead of writing the code, you can use the **OI BRWDesigner** application to type the following script code directly into the VBScript Editor of the Detail section's OnPrint property. Complete the following steps:

1. Select **Detail** from the Properties window drop-down list in the Designer. This reveals the section's available properties.
2. Click the empty box next to the OnPrint property, then click the drop-down arrow, and select **Script Editor** from the list.
3. In the **VBScript Editor**, simply type the following script in the window:

```
If UnitsInStock < ReorderLevel Then
  ProductNameCtl.ForeColor = RGB(255,0,0)
  ProductNameCtl.Font.Bold = True
Else
  ProductNameCtl.ForeColor = RGB(0,0,0)
  ProductNameCtl.Font.Bold = False
End If
```

The control executes the VBScript code whenever the section is about to be printed. The script gets the value of the "ReorderLevel" database field and sets the "ProductName" report field's Font.**Bold** and ForeColor properties according to the value. If the product is below reorder level, its name becomes bold and red.

The following screen capture shows a section of the report with the special effects:

| ProductID | ProductName | QuantityPerUnit | ReorderLevel | UnitsInStock |
|---|---|---|---|---|
| 10 | Ikura | 12 - 200 ml jars | 0 | 31 |
| 13 | Konbu | 2 kg box | 5 | 24 |
| 18 | Carnarvon Tigers | 16 kg pkg. | 0 | 42 |
| 30 | Nord-Ost Matjeshering | 10 - 200 g glasses | 15 | 10 |
| 36 | Inlagd Sill | 24 - 250 g jars | 20 | 112 |
| 37 | Gravad lax | 12 - 500 g pkgs. | 25 | 11 |
| 40 | Boston Crab Meat | 24 - 4 oz tins | 30 | 123 |
| 41 | Jack's New England Clam Chowder | 12 - 12 oz cans | 10 | 85 |
| 45 | Røgede sild | 1k pkg. | 15 | 5 |
| 46 | Spegesild | 4 - 450 g glasses | 0 | 95 |

## Hiding a Section if There is No Data for It

You can change a report field's format based on its data by specifying an expression for the Detail section's OnFormat property.

For example, your Detail section has fields with an image control and when there is no data for that record's image you want to hide the record. To hide the Detail section when there is no data, in this case a record's image, add the following script to the Detail section's OnFormat property:

```
If isnull(PictureFieldName) Then
Detail.Visible = False
Else
Detail.Visible = True
End If
```

### To hide a section if there is no data for it using OI BRWDesigner:

You can use the **OI BRWDesigner** application to type the following script code directly into the VBScript Editor of the Detail section's OnFormat property. Complete the following steps:

1. Select **Detail** from the Properties window drop-down list in the Designer. This reveals the section's available properties.
2. Click the empty box next to the OnFormat property, then click the drop-down arrow, and select **Script Editor** from the list.
3. In the **VBScript Editor**:
   - Simply type the following script in the window:

     ```
     If isnull(PictureFieldName) Then
     Detail.Visible = False
     Else
     Detail.Visible = True
     End If
     ```

   - Or you could use the more concise version:

     ```
     Detail.Visible = not isnull(PictureFieldName)
     ```

## Showing or Hiding a Field Depending on a Value

Instead of changing the field format to highlight its contents, you could set another field's Visible property to **True** or **False** to create special effects. For example, if you created a new field called "BoxCtl" and formatted it to look like a bold rectangle around the product name, then you could change the script as follows:

```
If UnitsInStock < ReorderLevel Then
BoxCtl.Visible = True
Else
BoxCtl.Visible = False
End If
```

### To highlight products that are below the reorder level using OI BRWDesigner:

You can use the **OI BRWDesigner** application to type the following script code directly into the VBScript Editor of the Detail section's OnPrint property. Complete the following steps:

1. Select **Detail** from the Properties window drop-down list in the Designer. This reveals the section's available properties.
2. Click the empty box next to the OnPrint property, then click the drop-down arrow, and select **Script Editor** from the list.
3. In the **VBScript Editor**, simply type the following script in the window:

```
If UnitsInStock < ReorderLevel Then
BoxCtl.Visible = True
Else
BoxCtl.Visible = False
End If
```

The following screen capture shows a section of the report with the special effects:

| ProductID | ProductName | QuantityPerUnit | ReorderLevel | UnitsInStock |
|---|---|---|---|---|
| 10 | Ikura | 12 - 200 ml jars | 0 | 31 |
| 13 | Konbu | 2 kg box | 5 | 24 |
| 18 | Carnarvon Tigers | 16 kg pkg. | 0 | 42 |
| 30 | Nord-Ost Matjeshering | 10 - 200 g glasses | 15 | 10 |
| 36 | Inlagd Sill | 24 - 250 g jars | 20 | 112 |
| 37 | Gravad lax | 12 - 500 g pkgs. | 25 | 11 |
| 40 | Boston Crab Meat | 24 - 4 oz tins | 30 | 123 |
| 41 | Jack's New England Clam Chowder | 12 - 12 oz cans | 10 | 85 |
| 45 | Røgede sild | 1k pkg. | 15 | 5 |
| 46 | Spegesild | 4 - 450 g glasses | 0 | 95 |

## Resetting the Page Counter

The Page variable is created and automatically updated by the control. It is useful for adding page numbers to page headers or footers.

### To reset the page counter when a group starts:

In some cases, you may want to reset the page counter when a group starts. For example, in a report that groups records by country and has a calculated page footer field with the expression:

```
=[Country] & " - Page " & [Page]
```

#### Using OI BRWDesigner:

To reset the page counter when a group (for example, a new country) starts, set the PageFooter field's **Text** property by completing the following steps:

1. Select the PageFooter's **PageNumber** field from the Properties window drop-down list in the Designer. This reveals the field's available properties.
2. Click the empty box next to the **Text** property, then click the drop-down arrow, and select **Script Editor** from the list.
3. In the **VBScript Editor**, simply type the following script in the window:

```
=[Country] & " - Page " & [Page]
```

### To reset the Page variable for each new country:

It would be good to reset the Page variable for each new country. To do this, assign the following script to the Country Group Header section:

```
Page = 1
```

#### Using OI BRWDesigner:

To reset the Page variable for each new country, set the Country Group Header OnPrint property by completing the following steps:

1. Select the Country Group Header from the Properties window drop-down list in the Designer. This reveals the section's available properties.
2. Click the empty box next to the OnPrint property, then click the drop-down arrow, and select **Script Editor** from the list.

3. In the **VBScript Editor**, simply type the following script in the window:

```
Page = 1
```

## Changing a Field's Dimensions to Create a Bar Chart

This is the most sophisticated example. Instead of showing a field's value as text, you can change its dimensions to create a chart.

### Create the Chart

To create a chart, the first thing you need to do is find out the scale, that is, the measurements that will be used to the maximum and minimum values. The "Sales Chart" report has a field designed to do this. It is a report footer field called **SaleAmountMaxFld** that has the size of the longest bar you want to appear on the chart, and holds the following expression:

```
=Max([SaleAmount])
```

#### Using OI BRWDesigner:

To set the maximum value for the chart, set the **SaleAmountMaxFld.Text** property by completing the following steps:

1. Select the **SaleAmountMaxFld** from the Properties window drop-down list in the Designer. This reveals the field's available properties.
2. Click the empty box next to the **Text** property, then click the drop-down arrow, and select **Script Editor** from the list.
3. In the **VBScript Editor**, simply type the following script in the window:

```
=Max([SaleAmount])
```

### Draw the Chart's Bars

To draw the actual bars, the report has a detail field called **BarFld** that is formatted to look like a solid box. The Detail section has the following script assigned to its OnPrint property:

```
BarFld.Width = SaleAmountMaxFld.Width * (SaleAmountFld / SaleAmountMaxFld)
```

This expression calculates the width of the bar based on the width and value of the reference field and on the value of the **SaleAmountFld** for the current record.

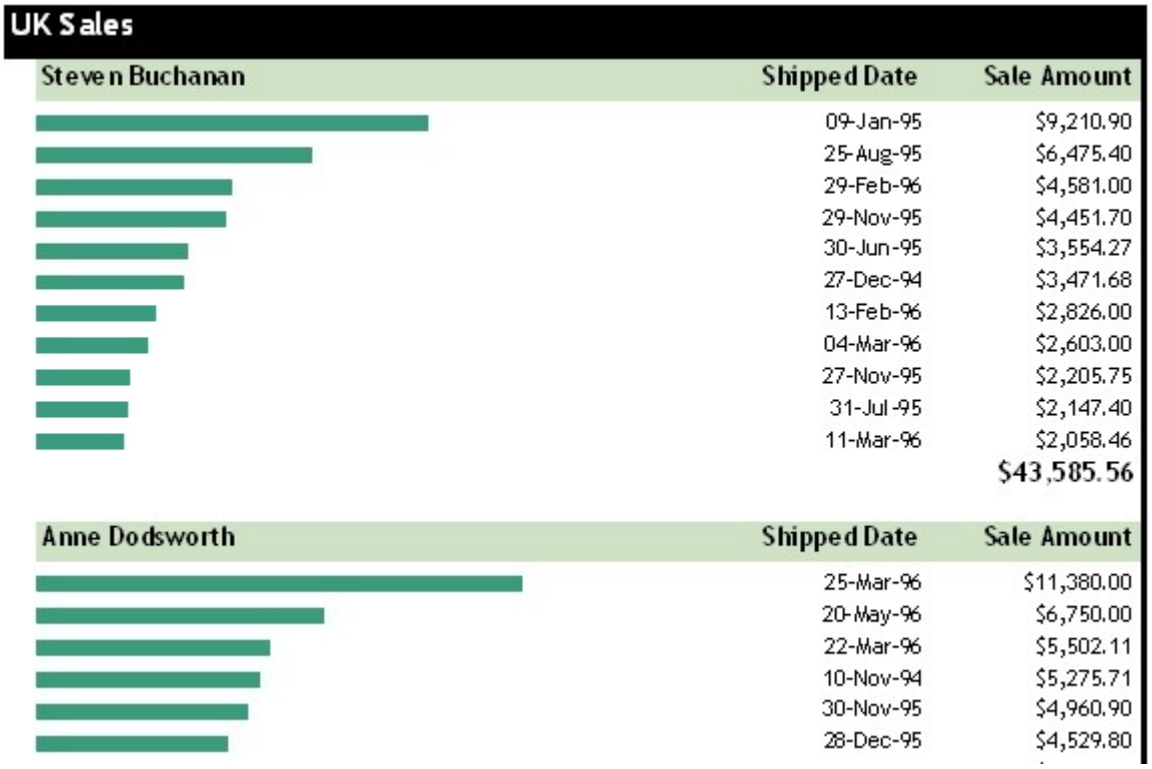
#### Using OI BRWDesigner:

To draw the actual bars for the chart, set the OnPrint property by completing the following steps:

1. Select **Detail** from the Properties window drop-down list in the Designer. This reveals the Detail section's available properties.
2. Click the empty box next to the OnPrint property, then click the drop-down arrow, and select **Script Editor** from the list.
3. In the **VBScript Editor**, simply type the following script in the window:

```
BarFld.Width = SaleAmountMaxFld.Width * (SaleAmountFld / SaleAmountMaxFld)
```

The following screen capture shows a section of the "Sales Chart" report with the special effects:



**UK Sales**

| Steven Buchanan | Shipped Date | Sale Amount |
|---|---|---|
| | 09-Jan-95 | $9,210.90 |
| | 25-Aug-95 | $6,475.40 |
| | 29-Feb-96 | $4,581.00 |
| | 29-Nov-95 | $4,451.70 |
| | 30-Jun-95 | $3,554.27 |
| | 27-Dec-94 | $3,471.68 |
| | 13-Feb-96 | $2,826.00 |
| | 04-Mar-96 | $2,603.00 |
| | 27-Nov-95 | $2,205.75 |
| | 31-Jul-95 | $2,147.40 |
| | 11-Mar-96 | $2,058.46 |
| | | **$43,585.56** |

| Anne Dodsworth | Shipped Date | Sale Amount |
|---|---|---|
| | 25-Mar-96 | $11,380.00 |
| | 20-May-96 | $6,750.00 |
| | 22-Mar-96 | $5,502.11 |
| | 10-Nov-94 | $5,275.71 |
| | 30-Nov-95 | $4,960.90 |
| | 28-Dec-95 | $4,529.80 |

# Working with OI BRWDesigner

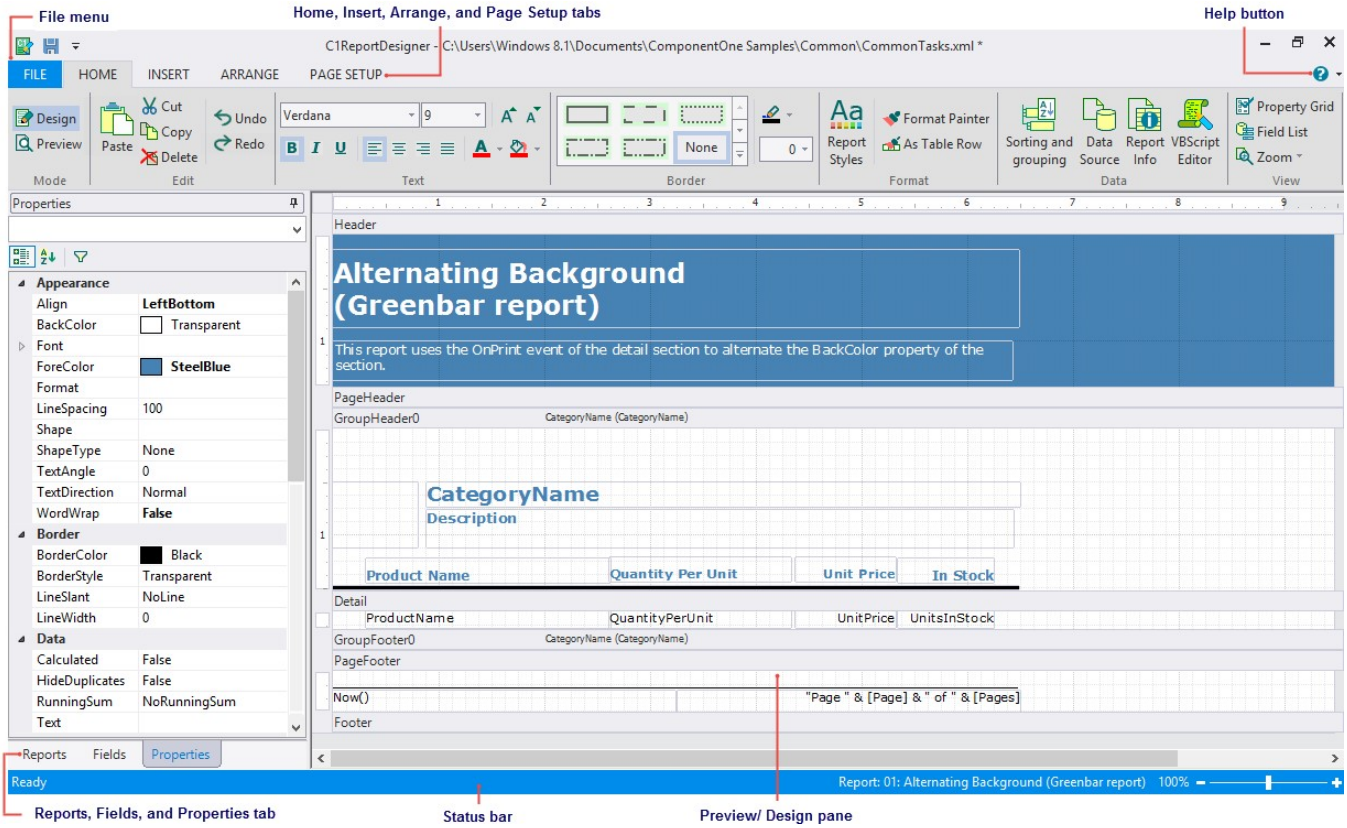
Working with OIBRWDesigner > About OIBRWDesigner

The **OIBRWDesigner** application is a tool used for creating and editing OIBRW report definition files. The Designer allows you to create, edit, load, and save files (XML) that can be read by the OIBRW component. It also allows you to import report definitions from Microsoft Access files (MDB) and VSReport 1.0 (VSR).
To run the Designer, double-click the **OIBRWDesigner.exe** file located by default in the following path for .NET 4.0:

- **C:\Program Files (x86)\ComponentOne\Apps\v4.0** for 64 bit platform
- **C:\Program Files\ComponentOne\Apps\v4.0** for 32 bit platform

Note that this directory reflects the default installation path and its path may be different if you made changes to the installation path. For steps on running the Designer from Visual Studio, see Accessing OIBRWDesigner from Visual Studio.

Here's what the Designer looks like with the CommonTasks.xml file opened:




The main Designer window has the following components:

- **File menu**: Click the **File** menu to load and save report definition files and to import and export report definitions. See File Menu for more information.
- **Design mode**: Provides shortcuts to the Edit, Text, Data, etc. menu functions. By default, Design Mode is selected which consists of Home, Insert, Arrange, Page Setup Tabs.
- **Preview mode**: Provides a preview of the report. See Preview Mode for more information.
- **Help button**: Provides options to open the online help file and view the **About** screen, which displays information about the application.
- **Reports tab**: Lists all reports contained in the current report definition file. You can double-click a report name to preview or edit the report. You can also use the list to rename, copy, and delete reports.
- **Fields tab**: Lists all the fields contained in the current report.
- **Properties tab**: Allows you to edit properties for the objects that are selected in the Designer.

- **Status bar**: The status bar displays information about what the Designer is working on (for example, loading, saving, printing, rendering, importing, and so on). You can zoom in and out of a selected report by dragging the zoom slider at the right of the status bar.

The topics that follow explain how you can use the **OIBRWDesigner** application to create, edit, use, and save report definition files.

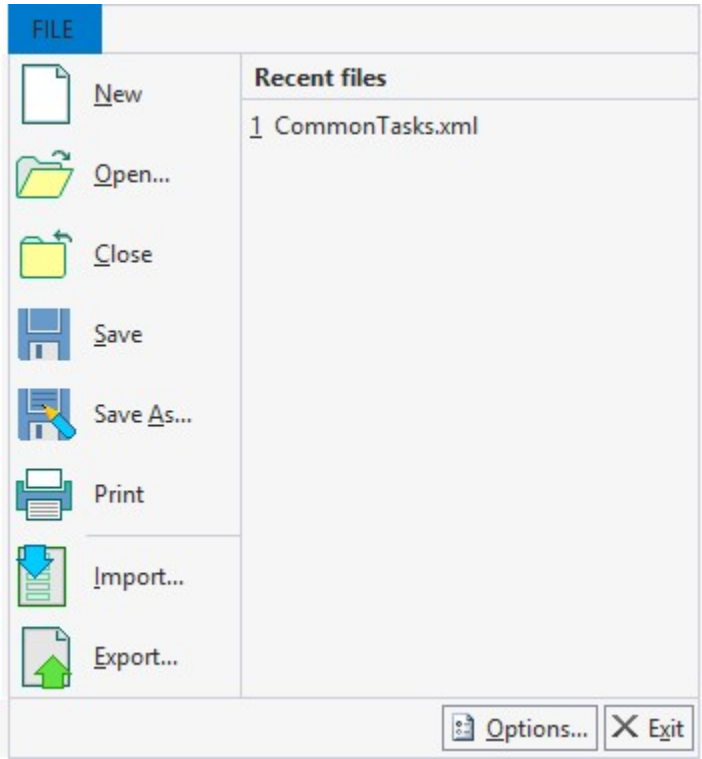
## See Also

Preview Mode
Design Mode
File Menu

## File Menu

Working with OIBRWDesigner > About OIBRWDesigner > File Menu

The **File** menu provides shortcut to load and save report definition files and to import and export report definitions. You can also access the **OIBRWDesigner** application's options through the **File** menu.
The following image displays the **File** menu:




The menu includes the following options:

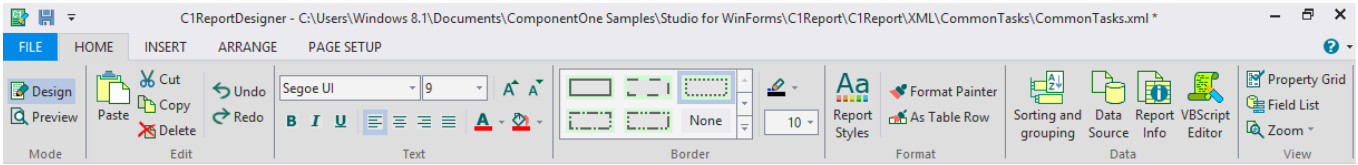
- **New**: Creates a new report definition file.
- **Open**: Brings up the **Open Report Definition File** dialog box, enabling you to select an existing file to open.
- **Close**: Closes the current report definition file.
- **Save**: Saves the report definition file, to the location previously saved.
- **Save As**: Opens the **Save Report Definition** dialog box allowing you to save your report definition as an XML file.
- **Print**: Prints the current report. Note that **Print** button is enabled only in preview mode of **OIBRWDesigner** application.

- **Import**: Opens the **Import Report Definition** dialog box enabling you to import Microsoft Access (.mdb and .adp) files and Crystal Reports (.rpt) files. See Importing Microsoft Access Reports and Importing Crystal Reports for more information.
- **Export**: Exports the current report file as an HTML, PDF, RTF, XLS, XLSX, TIF, TXT, ZIP, XPS, or C1DX file. Note that **Export** button is enabled only in preview mode of **OIBRWDesigner** application.
- **Recent files**: Lists recently opened report definition files. To reopen a file, select it from the list.
- **Options**: Opens the **OIBRWDesigner Options** dialog box which allows you to customize the default appearance and behavior of the **OIBRWDesigner** application. See Setting OIBRWDesigner Options for more information.
- **Exit**: Closes the **OIBRWDesigner** application.

## Design Mode

Working with OIBRWDesigner > About OIBRWDesigner > Design Mode

In **Design** mode, sections and fields of the selected report are displayed. This is the main working area of the designer where reports can be created or modified. The ribbon on the **Design** mode consists of the following tabs:




- **Home tab**: Provides shortcuts to the Edit, Text, Border, Format, Data, and View menu functions. See Home Tab for more information.
- **Insert tab**: Provides shortcuts to various fields such as Arrow, Calculated, and Chart. See Insert Tab for more information.
- **Arrange tab**: Provides shortcuts to Grid, Alignment, Position, and Size menu functions. See Arrange Tab for more information.
- **Page Setup tab**: Provides shortcuts to Page Layout menu functions. See Page Setup Tab for more information.

## See Also

Home Tab
Insert Tab
Arrange Tab
Page Setup Tab

## *Home Tab*

Working with OIBRWDesigner > About OIBRWDesigner > Design Mode > Home Tab

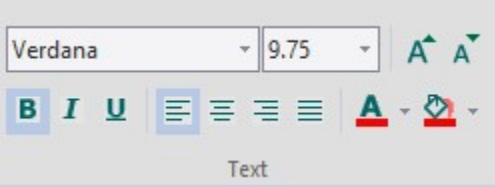
**Home tab** consists of several menu functions arranged in following groups:
**Edit group**: The following image displays **Edit** group:


It consists of the following options:

- **Paste**: Pastes the last copied item.
- **Cut**: Cuts the selected item, removing it from the report and allowing it to be pasted elsewhere.
- **Copy**: Copies the selected item so that it can be pasted elsewhere.

- **Delete**: Deletes the selected item.
- **Undo**: Undoes the last change that was made to the report definition.
- **Redo**: Redoes the last change that was made to the report definition.

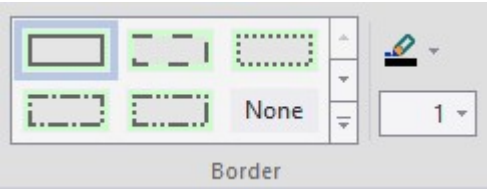
**Text group**: The following image displays **Text** group:




It consists of the following options:

- **Font**: Displays the current font of the selected text and allows you to choose another font for the selected item (to do so, click the drop-down arrow next to the font name).
- **Font Size**: Displays the current font size of the selected text and allows you to choose another font size. Type a number in the font size box or click the drop-down arrow to choose a font size.
- **Increase Font Size**: Increases the font size by one point.
- **Decrease Font Size:** Decreases the font size by one point.
- **Bold**: Makes the selected text bold (you can also press CTRL+B).
- **Italic**: Italicizes the selected text (you can also press CTRL+I).
- **Underline**: Underlines the selected text (you can also press CTRL+U).
- **Align Text Left:** Aligns text to the left.
- **Center Text**: Aligns text to the center.
- **Align Text Right**: Aligns text to the right.
- **Justify Text**: Justifies the selected text.
- **Font Color**: Allows you to select the color of the selected text.
- **Fill Color**: Allows you to select the background color of the selected text.

**Border group**: The following image displays **Border** group:




It consists of the following options:

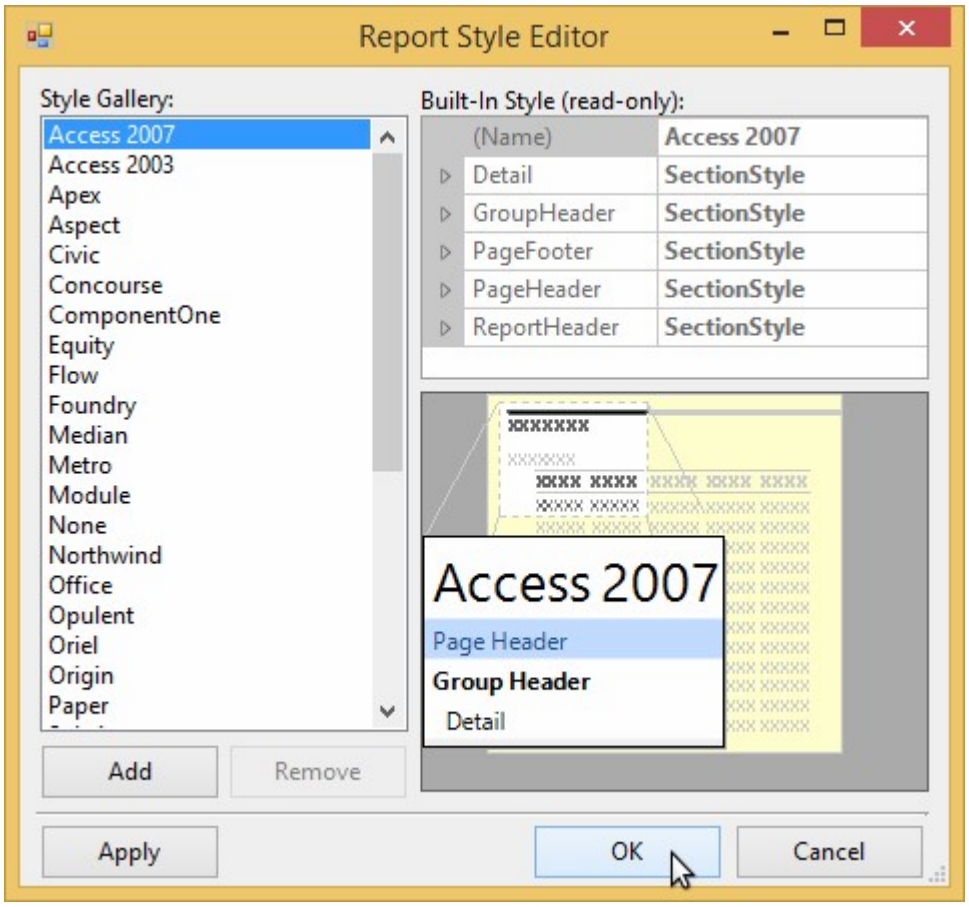
- **Border Line Style**: Defines the style of the border lines of the currently selected field(s). The styles available are: **Solid**, **Dash**, **Dot**, **Dash-Dot**, **Dash-Dot-Dot**, and **Transparent**.
- **Border Line Color**: Defines the color of the border lines of the currently selected field(s).
- **Border Line Width**: Defines the thickness of border line of the currently selected field(s) in *twips*.

**Format group**: The following image displays **Format** Group:


The **Format** group consists of the following options:

- **Report Styles**: Opens the **Report Style Editor** dialog box, where you can choose a built-in style or create and edit your own custom style.
- **Format Painter**: Applies style to the current selection.
- **As Table Row**: Formats the current selection as a table row.

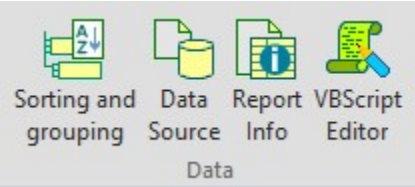
You can access the **Report Style Editor** dialog box by clicking **Report Styles** in the **Format** group.




It consists of following elements:

- **Style Gallery List**: Displays all the currently available built-in and custom styles. See Style Gallery for information about the available built-in styles.
- **Add button**: Adds a custom style to the Style Gallery list. The style that is added is based on the style selected in the Style Gallery list when the **Add** button was clicked.
- **Remove button**: Removes a selected custom style. The button is enabled only when a custom style is selected in the Style Gallery list.
- **Property grid**: Lets you change the properties and edit a custom style. The Property grid is only available and editable when a custom style is selected in the Style Gallery list.
- **Preview window**: Displays a preview of the style selected in the Style Gallery list.
- **Apply button**: Applies the style to your selection without closing the dialog box.
- **OK button**: Closes the dialog box, applies your changes, and sets the style as the current selected style.
- **Cancel button**: Cancels any changes you have made to styles.

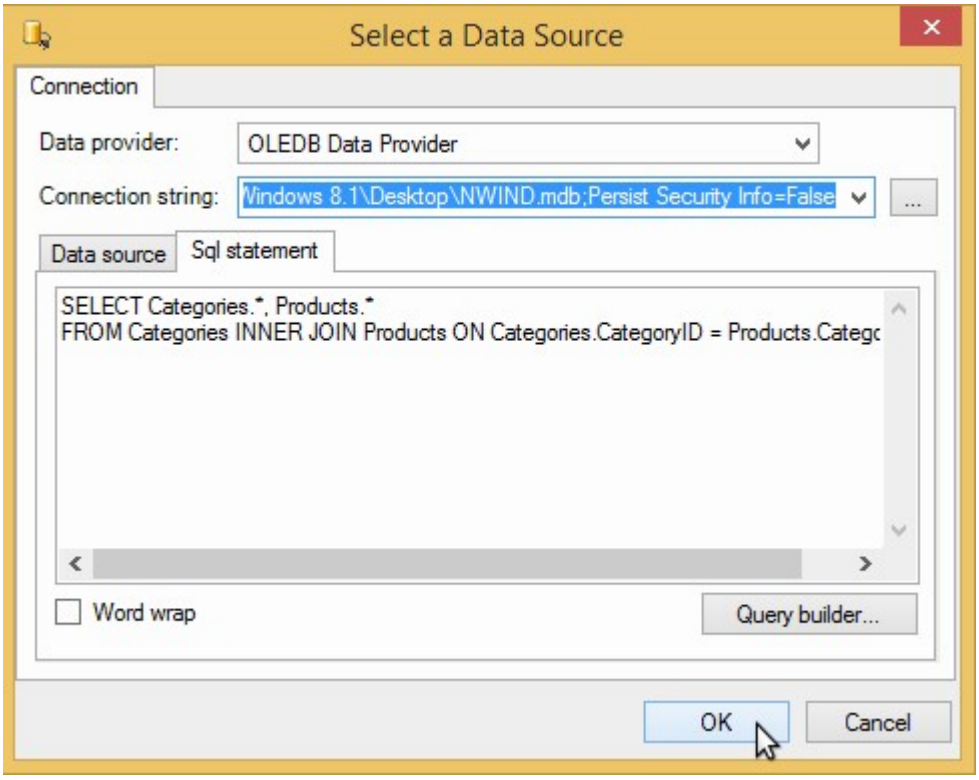
**Data group**: The following image displays **Data** group:




It consists of the following options:

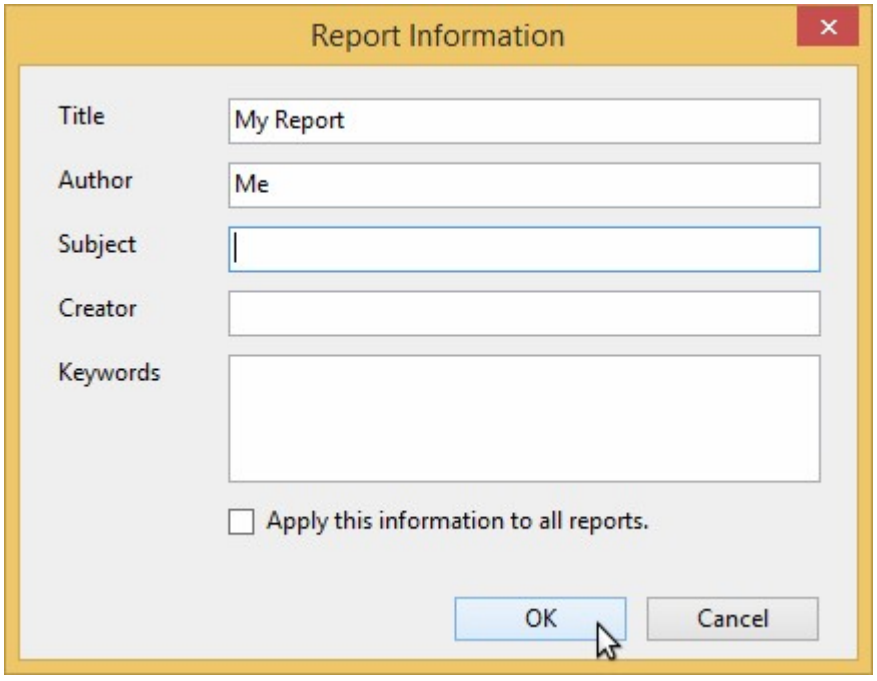
- **Sorting and grouping**: Clicking this button opens the **Sorting and Grouping** dialog box where you can add and delete sorting and grouping criteria. For more information see Grouping Data and Sorting Data.

- **Data Source**: Clicking this button opens the **Select a Data Source** dialog box. The **Select a Data Source** dialog box allows you to choose a new data source, change the connection string, and edit the SQL statement. Clicking the **Data source** tab displays the tables, views, and stored procedures in the current data source. Clicking the **Sql statement** tab allows you to view the current SQL statement:

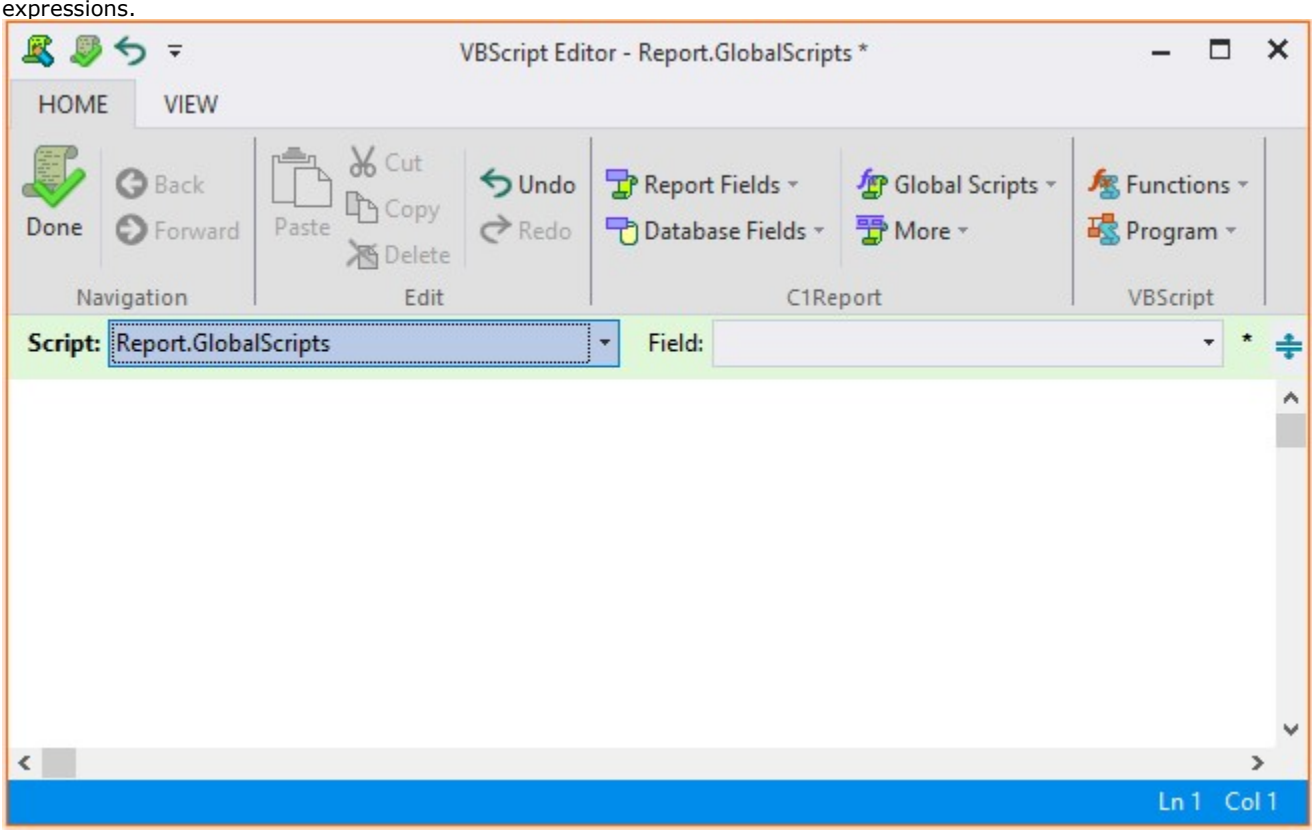



To change the connection string, click the ellipses button. This will open the **Data Link Properties** dialog box. To edit or change the SQL statement, click the **Query builder...** button which will open the **Sql Builder** dialog box.

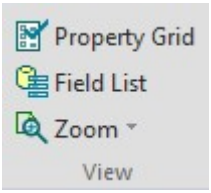
- **Report Info**: Opens the **Report Information** dialog box. This dialog box allows you to set the report's Title, Author, Subject, Creator, and Keywords. You can also choose to apply report information to all reports.



- **VBScript Editor**: Opens **VBScript Editor-Report.GlobalScripts** dialog box. Multiple scripts can be easily edited in the **VBScript Editor**, allowing switching between statements and expressions.




**View group**: The following image displays **View** group:




It consists of the following options:

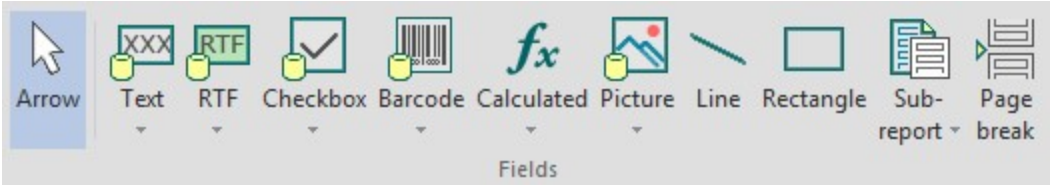
- **Property Grid**: Brings the **Properties** tab into view on the left pane. Note that you can also use F4 to view the **Properties** tab.
- **Field List**: Brings the **Fields** tab into view on the left pane.
- **Zoom**: Allows you to select a value to set the zoom level of the report.

## *Insert Tab*

Working with OIBRWDesigner > About OIBRWDesigner > Design Mode > Insert Tab

**Insert** tab consists of several fields which can be inserted while designing a report. Each field button creates a field and initializes its properties. The **Insert** tab consists of two groups:

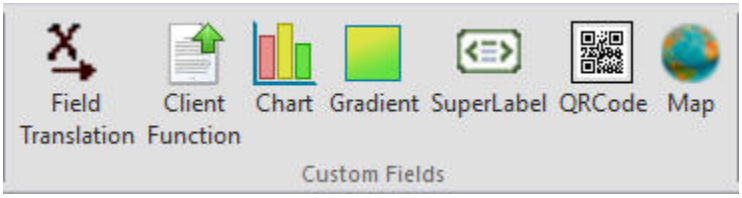
**Fields** group: The following image displays **Fields** group:



It consists of the following items:

- **Arrow**: Returns the mouse cursor to an arrow cursor.
- **Text**: Creates a field bound to the source recordset or an unbound (static) text label. When you click this button, a menu appears and you can select the recordset field. Bound fields are not limited to displaying raw data from the database. You can edit their Text property and use any VBScript expression.
- **RTF** : Creates an RTF field. When you click this button, a menu appears where you can select other fields that are contained in the same report definition file to be displayed in RTF format.
- **Checkbox**: Creates a bound field that displays a Boolean value as a check box. By default, the check box displays a regular check mark. You can change it into a radio button or cross mark by changing the value of the field's Checkbox property after it has been created.
- **Barcode** : Creates a field that displays a barcode. When you click this button, a menu appears where you can select other fields that are contained in the same report definition file to be displayed as a barcode. See Barcodes in Reports for more information.
- **Calculated**: Creates a calculated field. When you click this button, the code editor dialog box appears so you can enter the VBScript expression or an arbitrary formula whose value you want to evaluate. When you click the drop down, you can select commonly used expressions that render the date or time when the report was created or printed, the page number, page count, or "page n of m", or the report name.
- **Picture**: Creates a field for data bound stored in the recordset picture or static (unbound) picture. When you click this button, a drop down appears so you can select a picture field in the source recordset (if there is one; not all recordsets contain this type of field). If there are no data bound pictures, then this option creates a field that displays a static picture, such as a logo. A dialog box appears on clicking Picture to prompt you for a picture file to insert in the report. A copy is made of the picture you select and placed in the same directory as the report file. You must distribute this file with the application unless you embed the report file in the application. When you embed a report file in your application, any unbound picture files are embedded too.
- **Line**: Creates a line. Lines are often used as separators.
- **Rectangle**: Creates a rectangle. Rectangles are often used to highlight groups of fields or to create tables and grids.
- **Sub-Report** : Creates a field that displays another report. When you click this button, a menu appears and you can select other reports that are contained in the same report definition file. See Creating a Master-Detail Report Using Subreports for more information.
- **Page Break**: Creates a field that inserts a page break.

**Custom Fields** group: The following image displays **Custom Fields** group:




It consists of the following items:

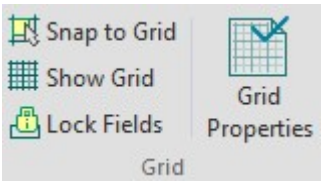
- **Field Translation**: Creates a field that "translates" from the current record to a different table and record, using OpenInsight's "XLATE" functionality (discussed below).
- **Client Function**: Creates a field that invokes a function in the OpenInsight database to return a value to display (discussed below).
- **Chart Field**: Creates a field that displays a chart. Unlike most bound fields, Chart fields display multiple values. To select the data you want to display, set the Chart field's **Chart.DataX** and **Chart.DataY** properties. To format the values along the X and Y axis, set the **Chart.FormatX** and **Chart.FormatY** properties. See Adding Chart Fields for more information.
- **Gradient Field**: Creates a gradient field. Gradients are often used as a background feature to make other fields stand out. See Adding Gradient Fields for more information.
- **SuperLabel Field**: Creates a field that renders HTML formatted text. The text property of the field is set to any HTML text that is required to be rendered.
- **QRCode Field**: Creates a Quick Response code field that renders 2D bar codes.
- **Map Field**: Creates a field that displays a region of earth, i.e., a map. See Maps in Reports for more information.

See Enhancing the Report with Fields for more information. For more information on adding fields to your report, see Creating Report Fields.

## *Arrange Tab*

Working with OIBRWDesigner > About OIBRWDesigner > Design Mode > Arrange Tab

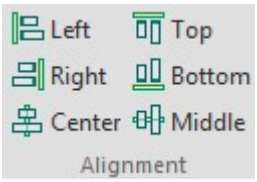
The **Arrange** tab provides shortcuts to grid, alignment, positioning , and sizing. It consists of the following groups.
**Grid group**: The following image displays **Grid** group:




The **Grid** group consists of the following items:

- **Snap to Grid**: Snaps fields to the grid. When this item is selected fields cannot be placed in between lines of the grid.
- **Show Grid**: Shows a grid in the background of the report in the preview. The grid can help you place and align fields. By default, this option is selected.
- **Lock Fields**: Locks and unlocks the fields in the report. After you've placed the fields in the desired positions, you can lock them to prevent inadvertent moving of fields with mouse or keyboard.
- **Grid Properties**: Opens the **OIBRWDesigner Options** dialog box. For details see Setting OIBRWDesigner Options.

**Alignment group**: The following image displays **Alignment** group:




The **Alignment** group consists of the following items:

- **Left**: Aligns the selected field horizontally to the left.
- **Right**: Aligns the selected field horizontally to the right.
- **Center**: Aligns the selected field horizontally to the center.
- **Top**: Aligns the selected field vertically to the top.
- **Bottom**: Aligns the selected field vertically to the bottom.
- **Middle**: Aligns the selected field vertically to the middle.

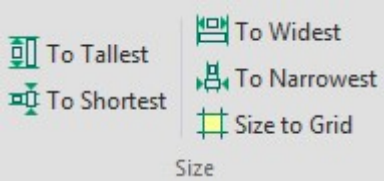
Note that the elements in a report can be both horizontally and vertically aligned - so, for example, an element can be both left and top aligned.
**Position group**: The following image displays **Position** group:


The **Position** group controls spacing between elements and how elements are layered. It consists of the following items:

- **Equalize Horizontal**: Equalizes horizontal spacing between selected fields.
- **Increase Horizontal**: Increases the horizontal spacing between selected fields.
- **Decrease Horizontal**: Decreases the horizontal spacing between selected fields.
- **Equalize Vertical**: Equalizes vertical spacing between selected fields.

- **Increase Vertical**: Increases the vertical spacing between selected fields.
- **Decrease Vertical**: Decreases the vertical spacing between selected fields.
- **Snap to Grid**: Snaps the currently selected field(s) to the nearest grid line(s).
- **Bring to Front**: Brings the selected field to the front of all layered fields.
- **Send to Back**: Sends the selected field behind all layered fields.

**Size group**: The following image displays the **Size** group:



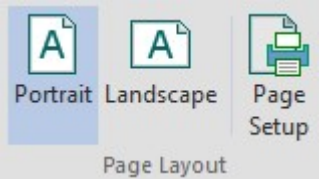

The **Size** group consists of the following items:

- **To Tallest**: Sets the height of all selected fields to the tallest field.
- **To Shortest**: Sets the height of all selected fields to the shortest field.
- **To Widest**: Sets the width of all selected fields to the width of the widest field.
- **To Narrowest**: Sets the width of all selected fields to the width of the narrowest field.
- **Size to Grid**: Snaps the bounds of the selected fields to the nearest grid lines.

## *Page Setup Tab*

Working with OIBRWDesigner > About OIBRWDesigner > Design Mode > Page Setup Tab

The **Page Setup** tab provides shortcuts to Page Layout menu functions.
The following image displays the **Page Layout** group on the **Page Setup** tab:



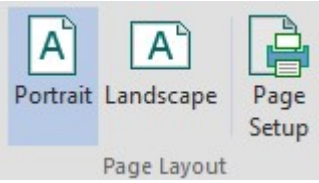

It consists of the following options:

- **Portrait**: Changes the layout of your report to **Portrait** view (where the height is longer than the width).
- **Landscape**: Changes the layout of your report to **Landscape** view (where the height is shorter than the width).
- **Page Setup**: Opens the printer's **Page Setup** dialog box.

## Preview Mode

Working with OIBRWDesigner > About OIBRWDesigner > Preview Mode

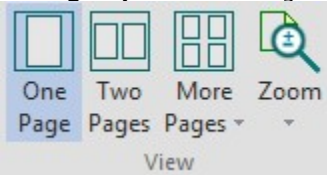
The **Preview** mode displays current report. The ribbon on the **Preview** mode consists of the following items:
**Page Layout group**: The following image displays the **Page Layout** group in Preview mode of the **OIBRWDesigner**:




It consists of the following options:

- **Portrait**: Changes the layout of your report to **Portrait** view (where the height is longer than the width).
- **Landscape**: Changes the layout of your report to **Landscape** view (where the height is shorter than the width).
- **Page Setup**: Opens the printer's **Page Setup** dialog box.

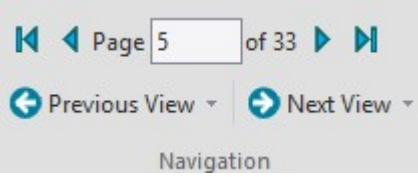
**View group**: The following image displays the **View** group:




It consists of the following options:

- **One page**: Allows you to preview one page at a time.
- **Two pages**: Allows you to preview two pages at a time.
- **More pages**: Clicking the drop-down arrow allows you to preview multiple pages at a time and includes the following options: **Four pages**, **Eight pages**, **Twelve pages**.
- **Zoom**: Zooms the page in to a specific percent or to fit in the window.

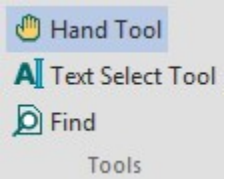
**Navigation group**: The following image displays the **Navigation** group:




It consists of the following options:

- **First Page**: Navigates to the first page of the preview.
- **Previous Page**: Navigates to the previous page of the preview.
- **Page**: Entering a number in this textbox navigates the preview to that page.
- **Next Page**: Navigates to the next page of the preview.
- **Last Page**: Navigates to the last page of the preview.
- **Previous View**: Returns to the last page viewed.
- **Next View**: Moves to the next page viewed. This is only visible after the **Previous View** button is clicked.

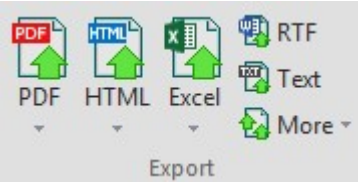
**Tools group**: The following image displays the **Tools** group:




It consists of the following options:

- **Hand Tool**: The hand tool allows you to move the preview through a drag-and-drop operation.
- **Text Select Tool**: The text select tool allows you to select text through a drag-and-drop operation. You can then copy and paste this text to another application.
- **Find**: Clicking the **Find** option opens the **Find** pane where you can search for text in the document. To find text enter the text to find, choose search options (if any), and click **Search.**

**Export Group**:The following image displays the **Export** group:

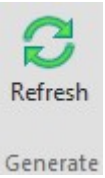



Each item in the Export group opens the **Export Report to File** dialog box where you can choose a location for your exported file. The **Export** group consists of the following options:

- **PDF**: Exports the document to a PDF file. The drop-down arrow includes options for **PDF (system fonts)** and **PDF (embedded fonts)** to choose if you want to use system fronts or embed your chosen fonts in the PDF file.
- **HTML**: Exports the document to an HTML file. You can then copy and paste this text to another application. The drop-down arrow includes options for **Plain HTML**, **Paged HTML**, and **Drilldown HTML**, and **Table-based HTML** allowing you to choose if you want to export to a plain HTML file, multiple HTML files that can be paged using included arrow links,or an HTML file that displays content that can be drilled down to, or a table-based HTML file that avoids use of absolute positioning.
- **Excel**: Exports the document to a Microsoft Excel file. The drop-down arrow includes options for **Microsoft Excel 97** and **Microsoft Excel 2007 - OpenXML** allowing you to choose if you want to save the document as an XLS or XLSX file.
- **RTF**: Exports the document to a Rich Text File (RTF).
- **Text**: Exports the document to a Text file (TXT).
- **More**: Clicking the **More** drop-down arrow includes additional options to export the report including: **Tagged Image File Format** (export as TIFF), **RTF (fixed positioning)**, **Single Page Text**, **Compressed Metafile** (export as ZIP), **XML Paper Specification**, **ComponentOne OpenXml** (C1DX).

For more information on exporting, see Exporting and Publishing a Report.

**Generate**:




It consists of **Refresh** button. Clicking **Refresh** regenerates the current report. This button changes to **Stop** while the document is regenerating; so you can also stop regenerating the report.

Other options available in the **Preview** mode of **OIBRWDesigner** are as follows:

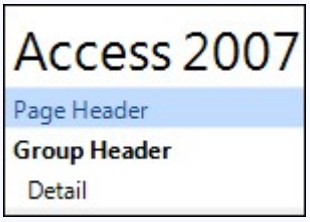
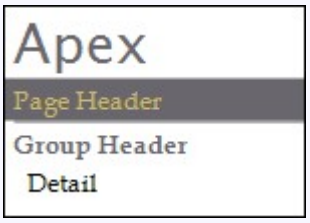
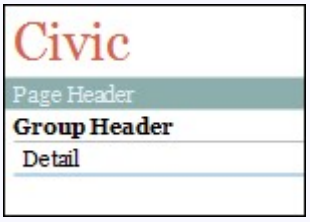
- **Pages tab**: Only available when in preview mode, this tab includes thumbnails of all the pages in the document.
- **Outline tab**: Only available when in preview mode, this tab displays a text outline of the document.
- **Find tab**: Only available when in preview mode, this tab displays find pane allowing you to search for text in the document.

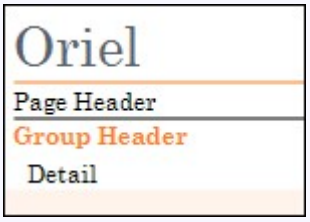
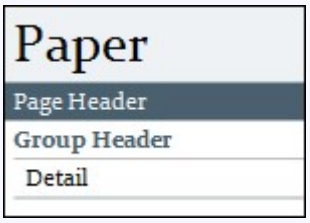
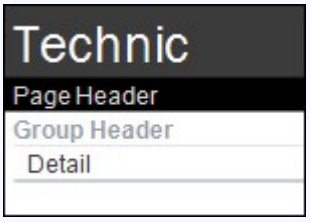
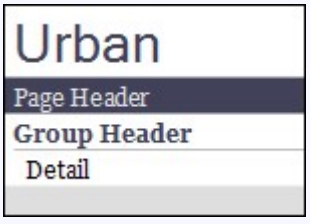
## Style Gallery

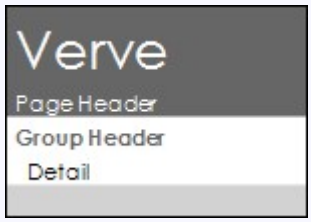
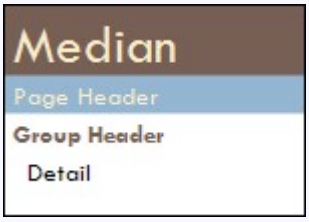
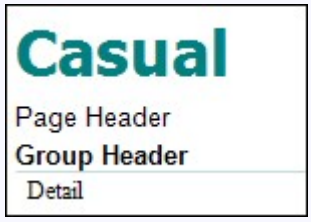
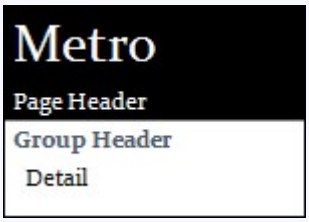
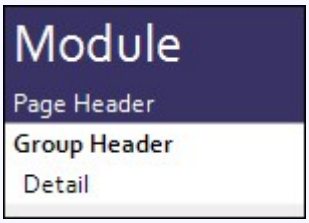
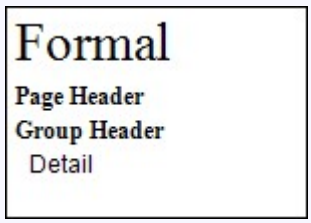
Working with OIBRWDesigner > About OIBRWDesigner > Style Gallery

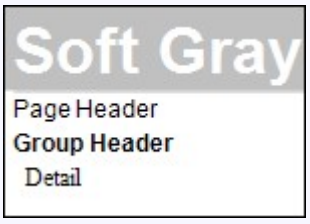
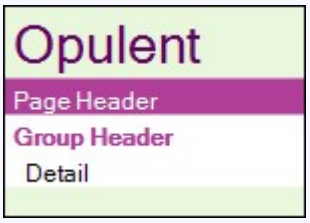
The **Style Gallery** dialog box details all the available built-in and custom styles that you can use to format your report. Built-in styles include standard Microsoft AutoFormat themes, including Vista and Office 2007 themes. You can access the **Style Gallery** from the **OIBRWDesigner** application by selecting the **Home** tab and clicking **Report Styles**.

The following built-in styles are included:

| Style Name | Preview |
|---|---|
| Access 2007 |   |
| Access 2003 |   |
| Apex |   |
| Aspect |   |
| Civic |   |
| Concourse |   |
| ComponentOne |   |

| Style Name | Preview |
|---|---|
| Oriel |   |
| Origin |   |
| Paper |   |
| Solstice |   |
| Technic |   |
| Trek |   |
| Urban |   |

| | | | |
|---|---|---|---|
| Equity |  | Verve |  |
| Flow |  | Windows Vista |  |
| Foundry |  | Bold |  |
| Median |  | Casual |  |
| Metro |  | Compact |  |
| Module |  | Corporate |  |
| None |  | Formal |  |

| | | | |
|---|---|---|---|
| Northwind |  | Soft Gray |  |
| Office |  | Verdana |  |
| Opulent |  | WebReport |  |

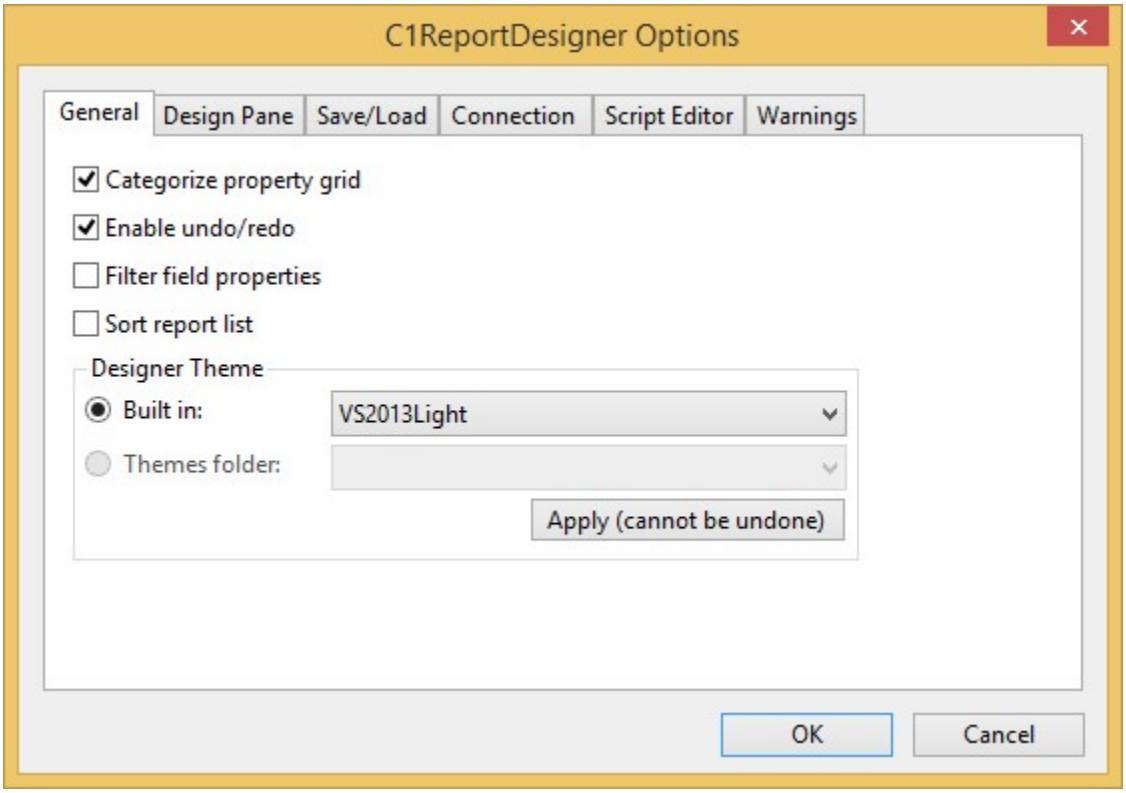
## *Setting OI BRWDesigner Options*

Working with OIBRWDesigner > Setting OIBRWDesigner Options

To access the **OIBRWDesigner Options** dialog box, click the **File** menu and then **Options**. For more information, see File Menu.

The **OIBRWDesigner Options** dialog box includes five tabs to control the appearance and behavior of the application. The tabs and the options available under each tab are:

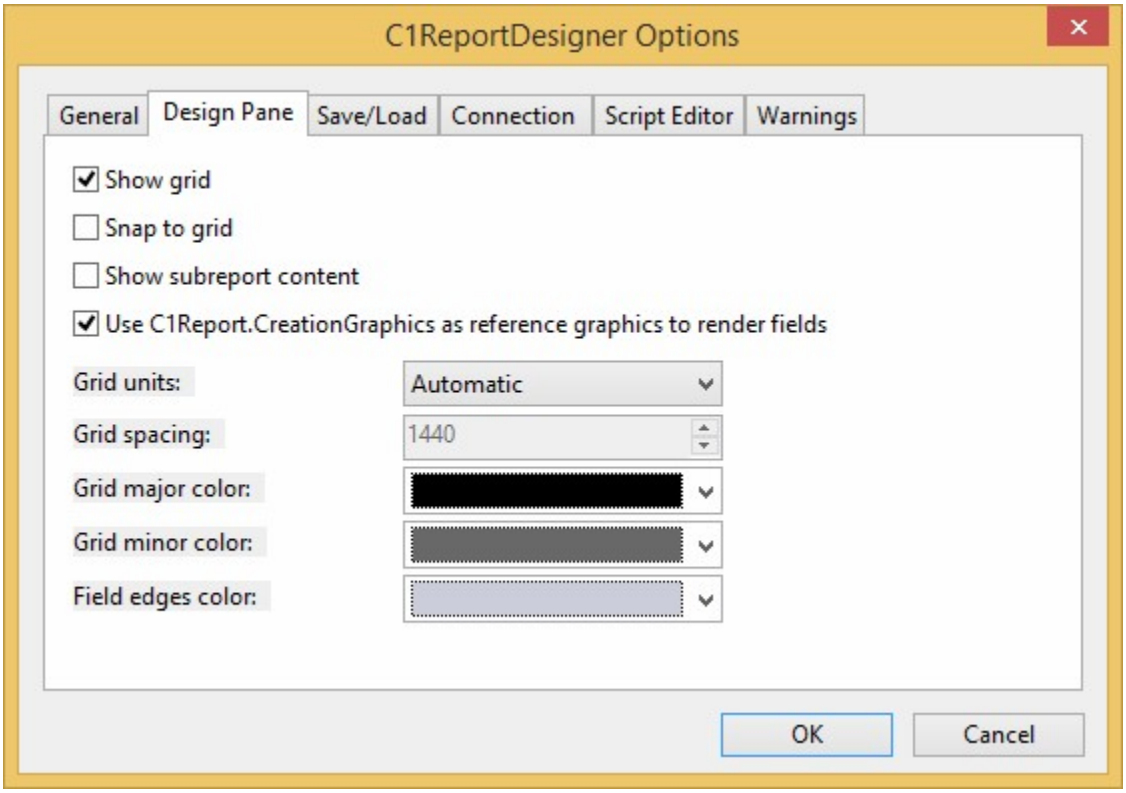
**General tab**:




It consists of the following options:

- **Categorize property grid**: Categorizes the Properties grid by property type. The Properties grid can be accessed by clicking the **Properties** tab located in the bottom of the left pane in Design view.
- **Enable undo/redo**: Enables undo and redo in the application.
- **Filter field properties**: Filters the Properties grid by properties that have been set. The Properties grid can be accessed by clicking the **Properties** tab located in the bottom of the left pane in Design view.
- **Sort report list**: Sorts the list of reports listed on the **Reports** tab. Reports can be accessed by clicking the **Reports** tab located in the bottom of the left pane in Design view.
- **Designer Theme**: Sets the theme from the options in **Built in** or in **Themes folder**.

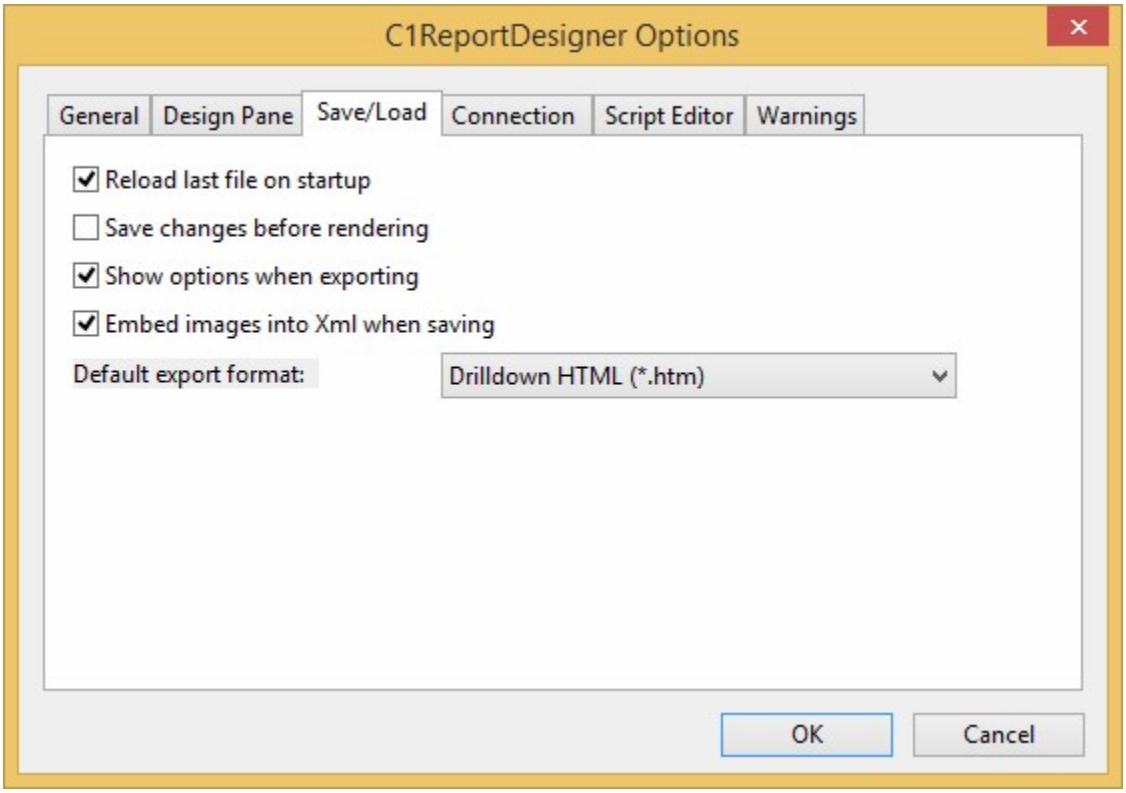
**Design Pane tab**:




It consists of the following options:

- **Show grid**: Shows the grid in the report preview window.
- **Snap to grid**: Snaps all objects the grid in the report. If this option is selected, you will not be able to place objects between grid lines.
- **Show subreport content**: Shows sub-report content in the report.
- **Use OIBRW.CreationGraphics as reference graphics to render fields**: Uses **OIBRW.CreationGraphics** as reference to render fields.
- **Grid units**: Indicates how the grid is spaced. Options include **Automatic**, **English (in)**, **Metric (cm)**, and **Custom**.
- **Grid spacing**: Sets the spacing of grid lines. This option is only available when the **Grid Units** option is set to **Custom**.
- **Grid major color**: Set the color of major grid lines.
- **Grid minor color**: Sets the color of minor grid lines.
- **Field edges color**: Sets the color of field edges in the report.

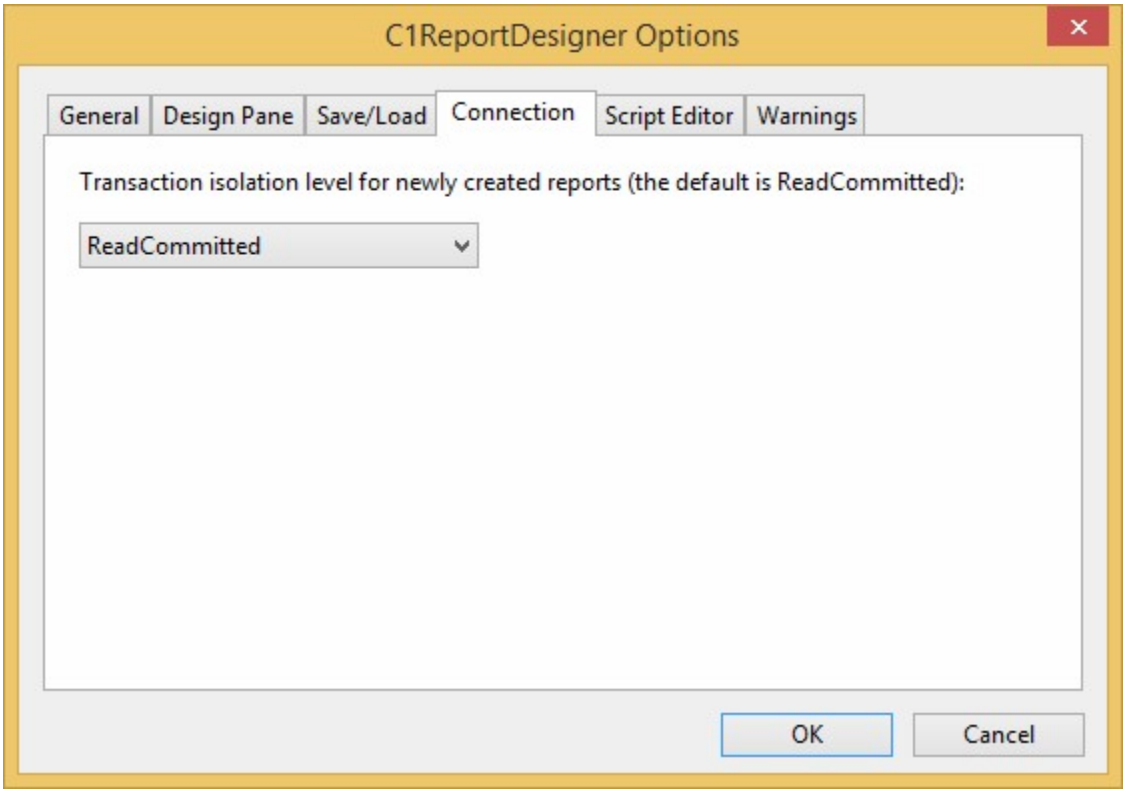
**Save/Load tab**:




It consists of the following options:

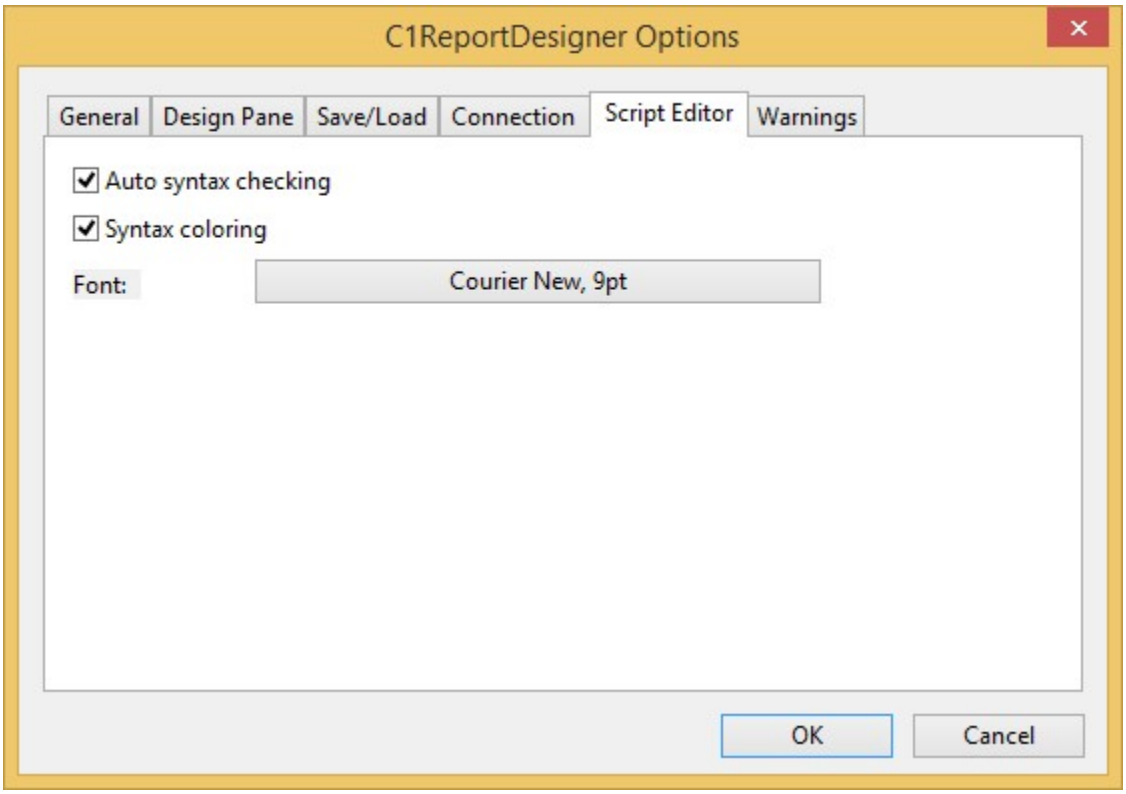
- **Reload last file on startup**: If this option is checked, the last opened file will appear whenever the **OIBRWDesigner** application is opened.
- **Save changes before rendering**: Checking this option saves the report before rendering.
- **Show options when exporting**: Checking this option saves the report's options when exporting.
- **Embed images into Xml when saving**: Checking this option embeds images into XML when the report is saved.
- **Default export format**: Sets the default export format. For more information about exporting see Export Group.

**Connection tab**:




It consists of the options for transaction isolation level.

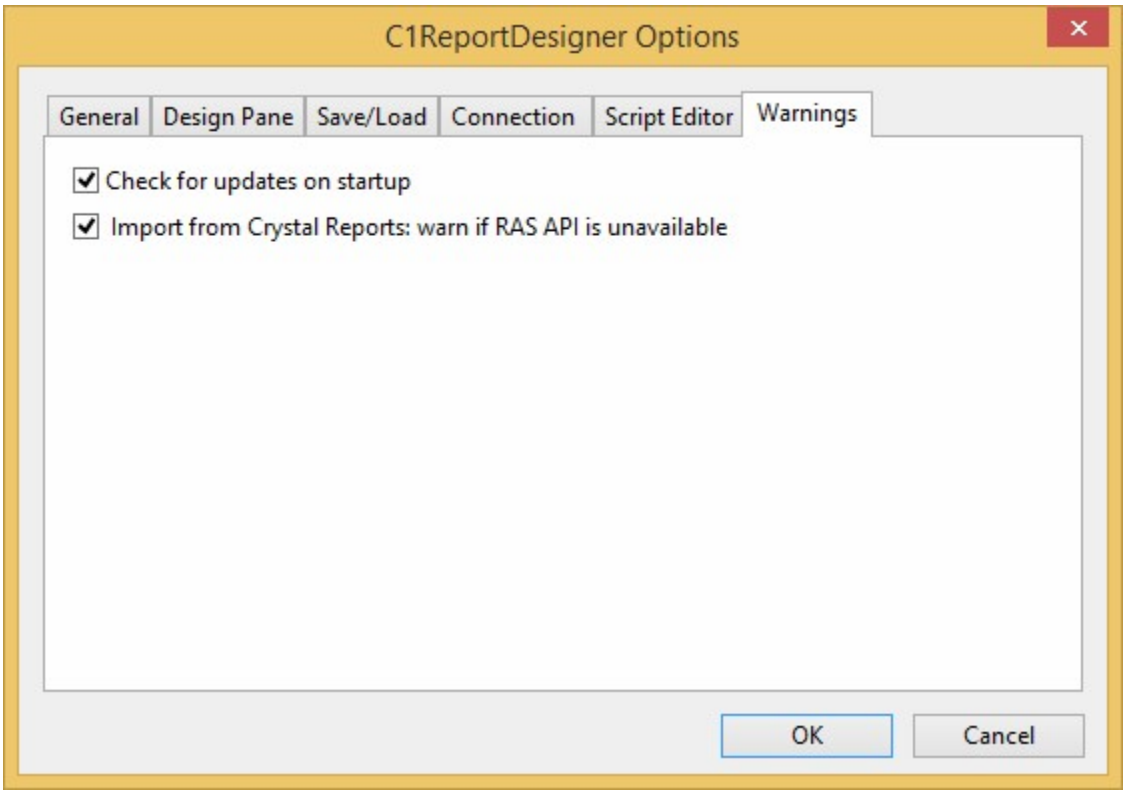
**Script Editor**:




It consists of the following options:

- **Auto syntax checking**: Determines if syntax is automatically checked in the **VBScript Editor** dialog box.
- **Syntax coloring**: Determines if syntax text is automatically colored in the **VBScript Editor** dialog box.
- **Font**: Defines the appearance of the text used in the **VBScript Editor** dialog box.

**Warnings tab**:




It consists of the following options:

- **Check for updates on startup**: Checking this option checks for any updates when OIBRWDesigner application is opened.
- **Import from Crystal Reports: warn if RAS API is unavailable**: Checking this option throws warning if RAS API is unavailable while importing Crystal Reports in **OIBRWDesigner**.

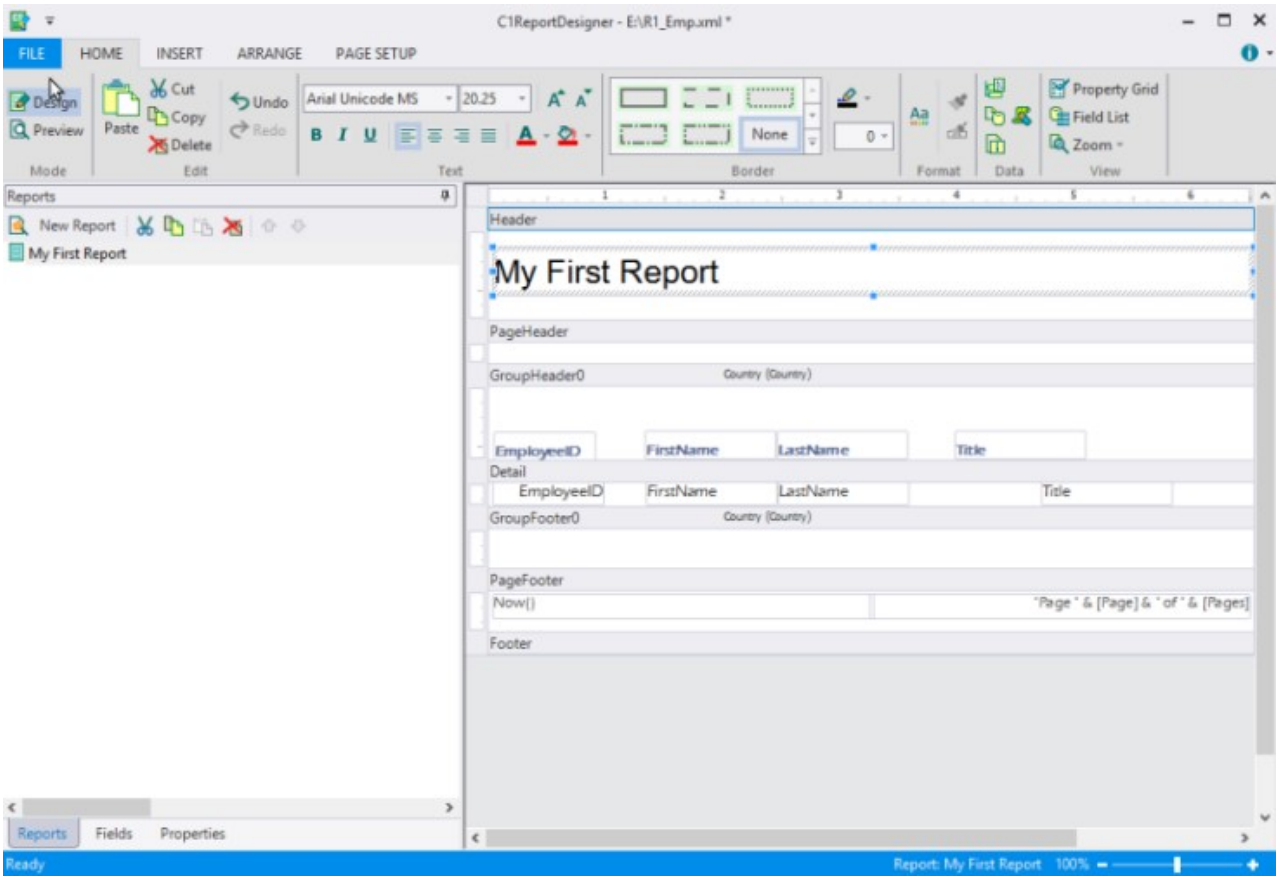
In each of the above tabs, clicking OK saves the changes and clicking Cancel cancels any changes that you have made in the **OIBRWDesigner Options** dialog box.

## *Modifying the Report Layout*

Working with OIBRWDesigner > Modifying the Report Layout

The report generated for you by the wizard is a good starting point, but you will usually need to adjust and enhance it to get exactly what you want. You can do this with the **OIBRWDesigner** application. See Step 1 of 4: Creating a Report Definition, to learn how to create a basic report definition.

To start using the Designer, click the **Design** mode. The right pane of the main window switches from Preview mode to Design mode, and it shows the controls and fields that make up the report:




The picture shows how the report is divided into sections (Header, Page Header, Detail, and so on). The sections contain fields that hold the labels, variables, and expressions that you want in the printed report. In this example, the Header section contains a label with the report title. The Page Header section contains labels that identify the fields in the Detail section, and the Page Footer section contains fields that show the current time, the page number and the total page count for the report.

The sections of the report determine what each page, group, and the beginning and end of the report look like. The following table describes where each section appears in the report and what it is typically used for:

| Section | Appears | Typically Contains |
|---|---|---|
| Report Header | Once per report | The report title and summary information for the whole report. |
| Page Header | Once per page | Labels that describe detail fields, and/or page numbers. |
| Group Header | Once per group | Fields that identify the current group, and possibly aggregate values for the group (for example, total, percentage of the grand total). |
| Detail | Once per record | Fields containing data from the source recordset. |
| Group Footer | Once per group | Aggregate values for the group. |
| Page Footer | Once per page | Page number, page count, date printed, report name. |
| Report Footer | Once per report | Summary information for the whole report. |

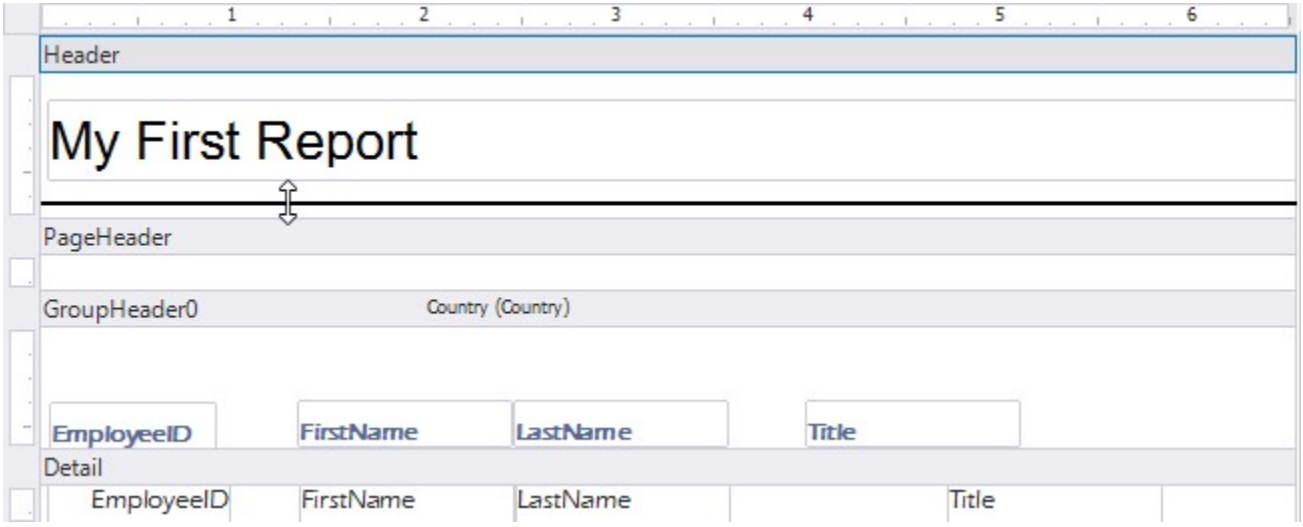
Note that you cannot add and delete sections directly. The number of groups determines the number of sections in a report. Every report has exactly five fixed sections (Report Header/Footer, Page Header/Footer, and Detail) plus two sections per group (a Header and Footer). You can hide sections that you don't want to display by setting their Visible property to **False**.

You can modify sections by changing their properties with the Properties window, or move and resize them with the mouse.

## Resizing a Section

To resize a section, select its border and with your mouse pointer drag to the position where you want it. The rulers on the left and on top of the design window show the size of each section (excluding the page margins). Note that you cannot make the section smaller than the height and width required to contain the fields in it. To reduce the size of a section beyond that, move or resize the fields in it first, then resize the Section.

To see how this works, move the mouse to the area between the bottom of the Page Header section and the gray bar on top of the Detail Section. The mouse cursor changes to show that you are over the resizing area. Click the mouse and drag the line down until the section is about twice its original height.




Release the mouse button and the section is resized.

# *Enhancing the Report with Fields*

To enhance your report, you can add fields (for example, lines, rectangles, labels, pictures, charts, and so on) to any Section. You can also modify the existing fields by changing their properties with the Properties window, or move and resize the fields with the mouse.

### Report Fields

The **Fields** group of the **Design** tab in the **OI BRWDesigner** application provides tools for creating report fields. This toolbar is enabled only in design mode. Each button creates a field and initializes its properties. For more information about the **Fields** group, see Fields Group. For more information on adding fields to your report, see Creating Report Fields.

---

- Adding Chart Fields
- Adding Gradient Fields
- Selecting, Moving, and Copying Fields
- Changing Field, Section, and Report Properties
- Changing the Data Source

## Adding Chart Fields

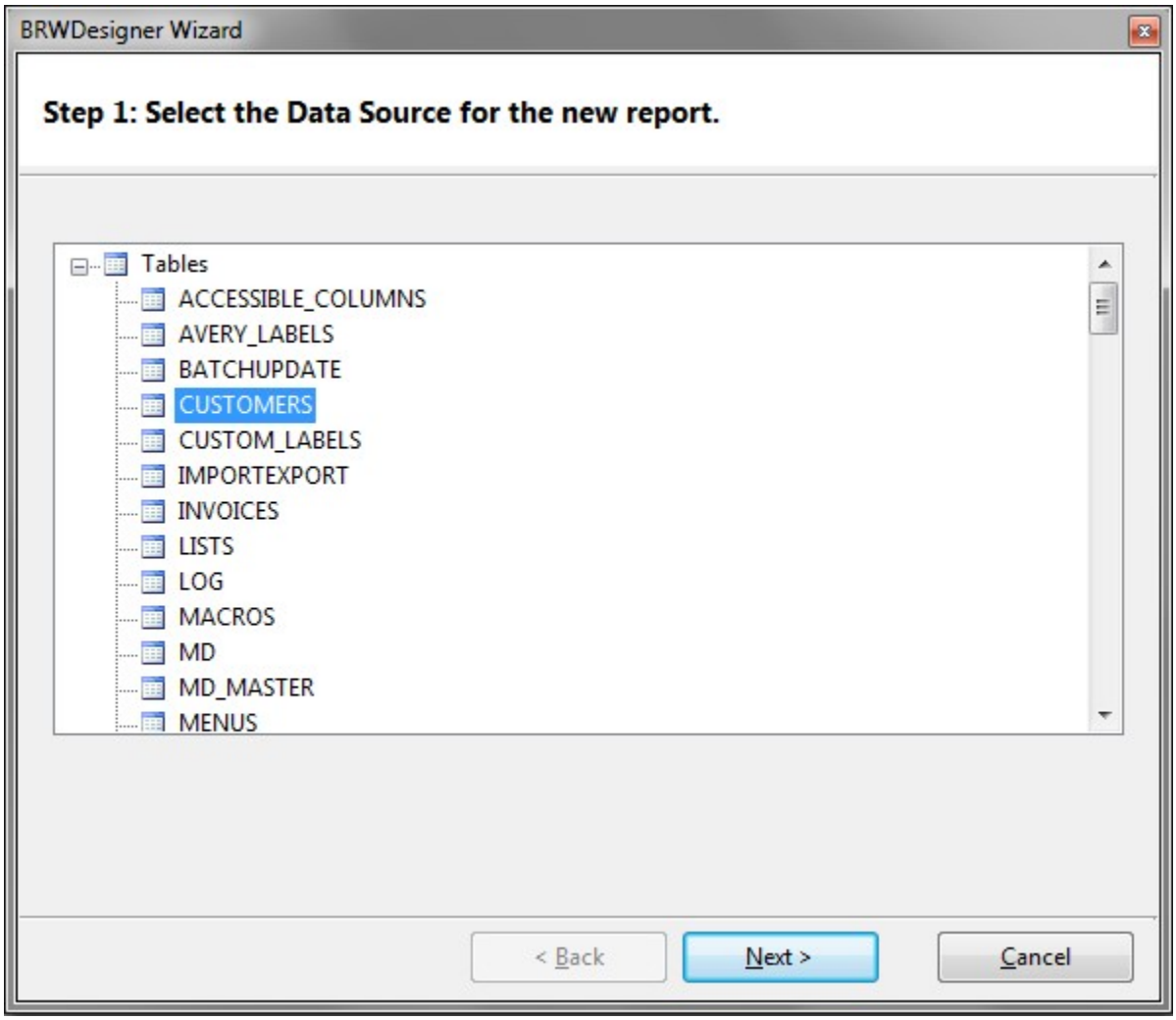
To add a Chart field to your report, complete the following steps:

Open the report in the **OIBRWDesigner** application.

Click the **Chart** button in the **Custom Fields** group of the **Insert** tab and mark the area in the report where the Chart should be displayed.
Set the field properties as usual.
The only unusual aspect of Chart fields is that unlike most bound fields, they display multiple values. To select the data you want to display, set the Chart field's **Chart.DataX** and **Chart.DataY** properties. To format the values along the X and Y axis, set the **Chart.FormatX** and **Chart.FormatY** properties. You can also customize the chart appearance by setting other properties such as **Chart.ChartType**, **Chart.Palette**, and so on.

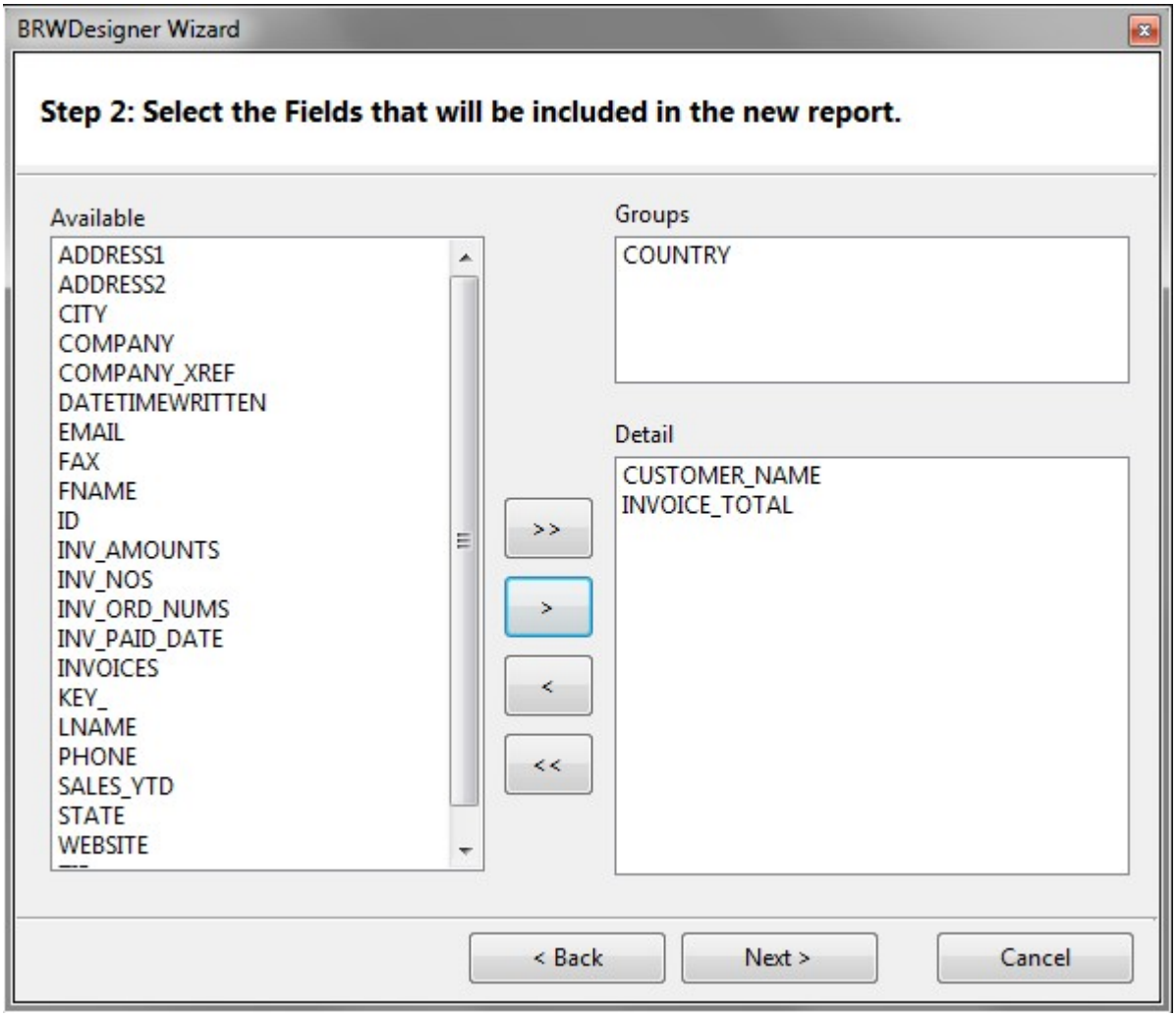
To create a new report with an embedded chart, use the **OI BRW Wizard**. Complete the following steps:

1. Select the Tables radio button, and then select the **Customers** table. The following image shows this step:




2. **Select the fields you want to display.**

   This example groups the data by Country and show Company_Name and Invoice in the Detail section of the report: To add groups and detail fields, with your mouse pointer drag them from the **Available** list on the left to the **Groups** or **Detail** lists on the right:



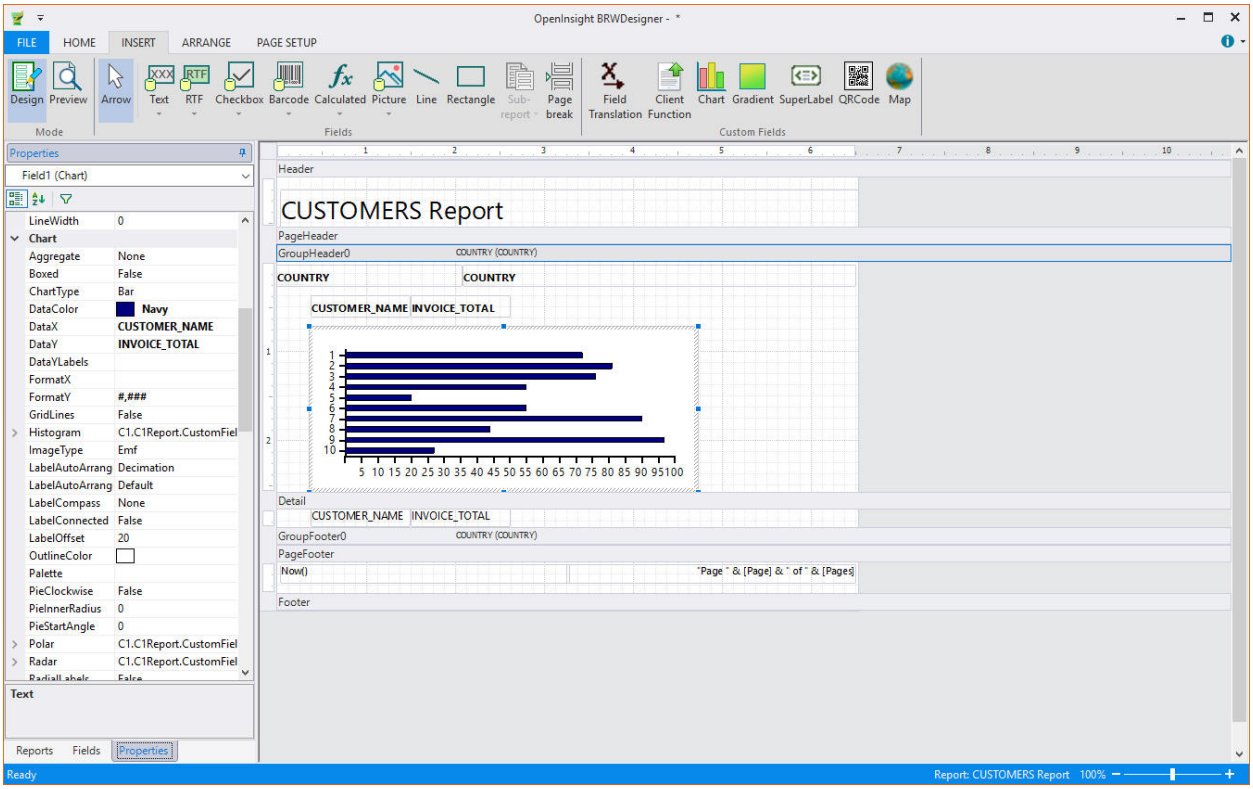

Continue clicking **Next** until the wizard is done. The wizard creates the initial version of the report.

**3. Add the Chart to the Group Header section of the report.**

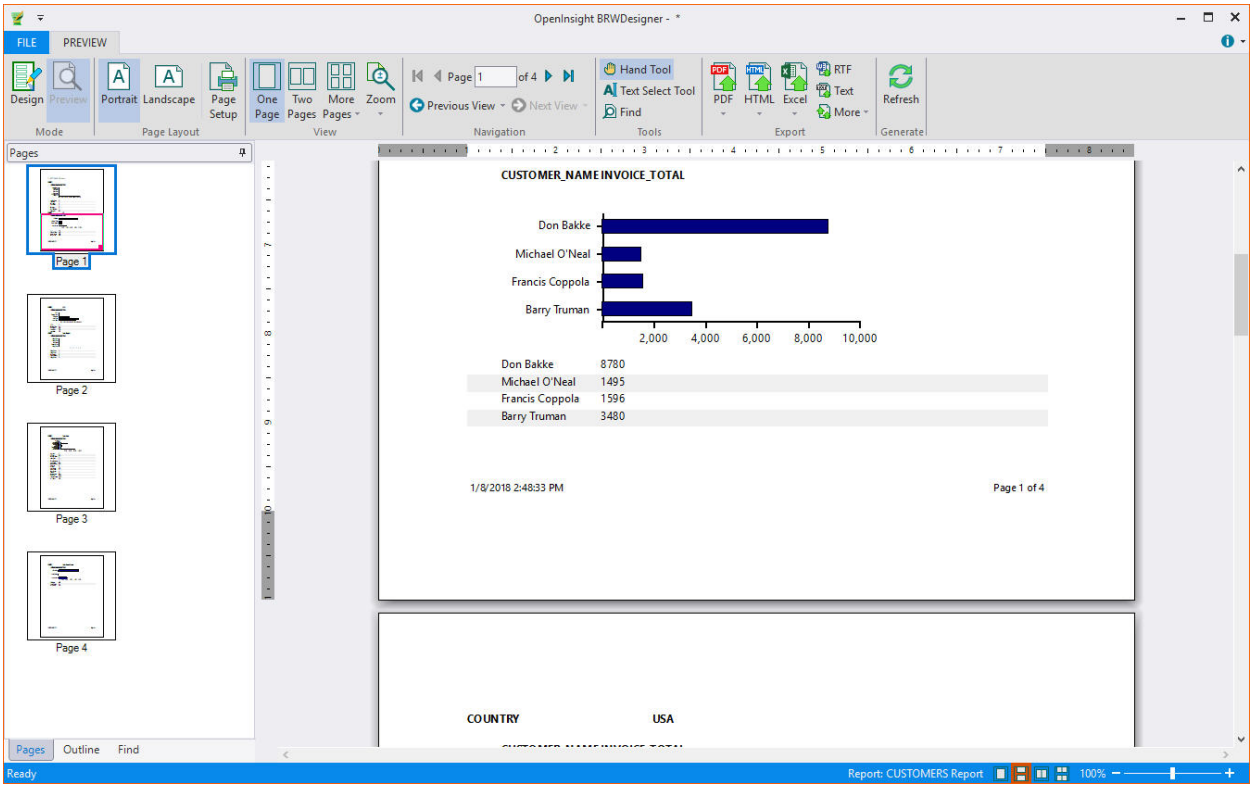
Charts usually make sense in the Group Header sections of a report, to summarize the information for the group. To add the chart to the Group Header section:

1. Click the **Design** button to switch to Design mode and begin editing the report.
2. Expand the Group Header Section by performing a drag-and-drop operation with the section's borders.
3. Click the **Chart** button in the **Custom Fields** group of the **Insert** tab and place the field in the report in the Group Header Section.
4. Resize the chart by clicking and dragging the chart field.
5. From the Properties window, set the **Chart.DataY** property to the name of the field that contains the values to be charted, in this case, **INVOICE_TOTAL**.
6. Note that the **Chart.DataY** property may specify more than one chart series. Just add as many fields or calculated expressions as you want, separating them with semicolons.
7. Set the **Chart.DataX** property to the name of the field that contains the labels for each data point, in this case, **CUSTOMER_NAME**.
8. From the Properties window, set the **Chart.FormatY** property to "#,###" to set the values along the axis to thousand-separated values.

Your report should look similar to the following report:




The Chart control will now display some sample data so you can see the effect of the properties that are currently set (the actual data is not available at design time). You may want to experiment changing the values of some properties such as **Chart.ChartType**, **Chart.DataColor**, and **Chart.GridLines**. You can also use the regular field properties such as **Font** and **ForeColor**.
Your report should look similar to the following report:



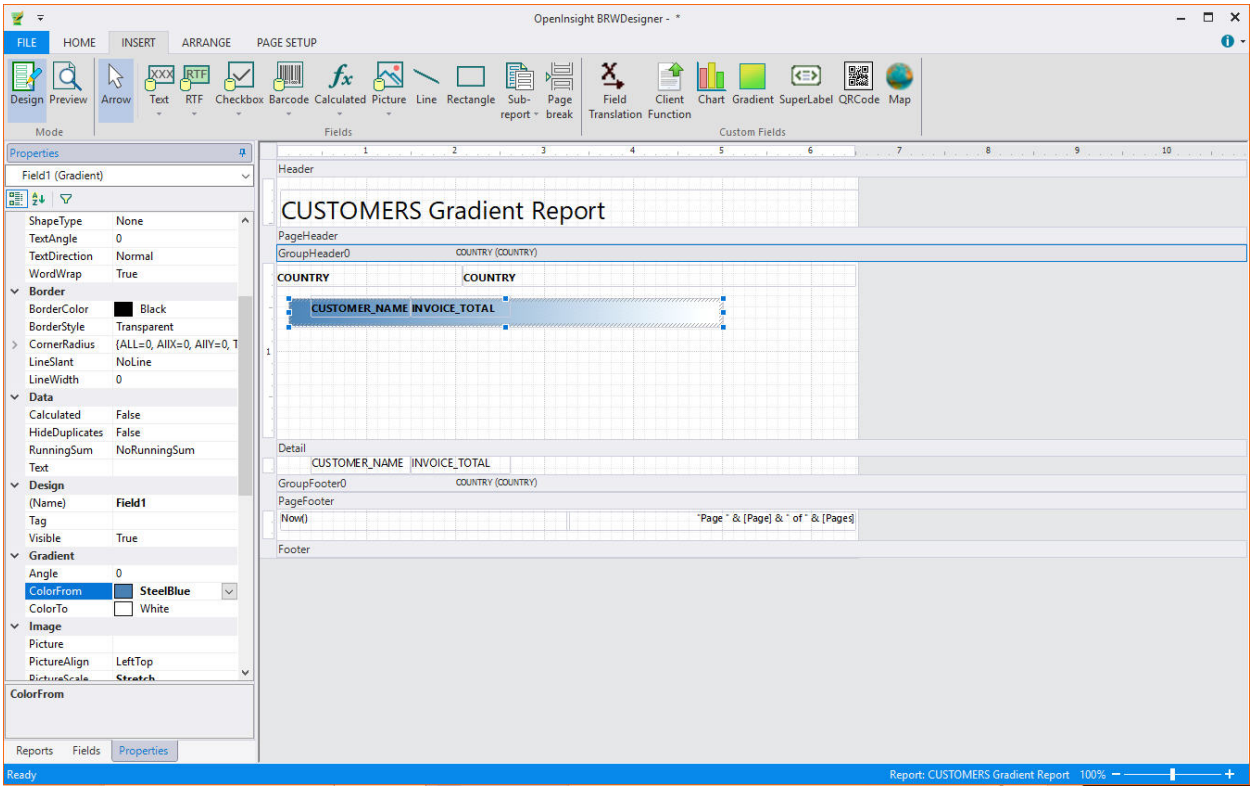

Note that the Report field is sensitive to its position in the report. Because it is in a Group Header section, it only includes the data within that group. If you place the Chart field in a Detail section, it will include all the data for the entire report. This is not useful because there will be one chart in each Detail section and they will all look the

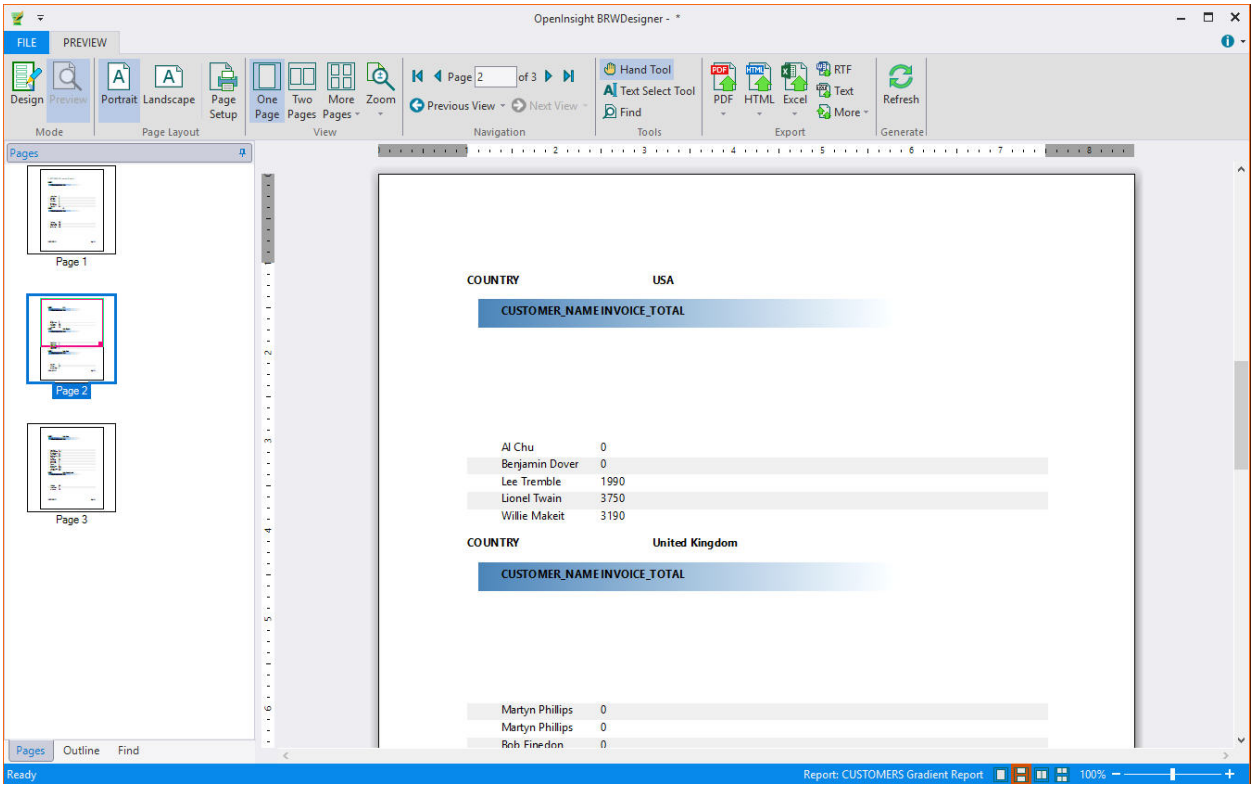
samIf you need more control over what data should be displayed in the chart, you can use the DataSource property in the chart field itself.

## Adding Gradient Fields

**Gradient** fields are much simpler than charts. They are mainly useful as a background feature to make other fields stand out.

The following image shows a report that uses gradient fields over the labels in the Group Header section:






To create a similar gradient field, complete the following steps:

1. In Design mode of the Designer, select the **Gradient** button from the **Custom Fields** group in the **Insert** tab.
2. Move your mouse cursor (which has changed to a cross-hair) over the labels in the Group Header section and drag the field to the desired size.
3. To ensure that the field is behind the labels, right-click the gradient field and select **Send To Back**.
4. Set the **Gradient.ColorFrom** and **Gradient.ColorTo** properties to **SteelBlue** and **White**, respectively.

Note that you may change the angle of the gradient field by setting the **Gradient.Angle** property another value (default value is 0).

To create a similar gradient, you need to set both the X and Y values of the corner radius. The **CornerRadius** property provides following options to set the X and Y values for the corners:

- All: Set X and Y values of all corners.
- AllX: Set X value of all corners.
- AllY: Set Y value of all corners.
- BottomLeftX: Set X value of bottom left corner.
- BottomLeftY: Set Y value of bottom left corner.
- BottomRightX: Set X value of bottom right corner.
- BottomRightY: Set Y value of bottom right corner.
- TopLeftX: Set X value of top left corner.
- TopLeftY: Set Y value of top left corner.
- TopRightX: Set X value of top right corner.
- TopRightY: Set Y value of top right corner.

You can also obtain different shapes of a gradient by setting **ShapeType** property from the Properties pane to the options available as:

- Line
- IsoscelesTriangle
- RightTriangle
- Rectangle
- Ellipse
- Arc
- Pie

There are some limitations for four corner round radii in a gradient field:

1. Range of Radius X is from 0 to field's width.
2. Range of Radius Y is from 0 to field's height.
3. For Radius X, TopLeft + TopRight/BottomLeft + BottomRight should not be greater than field's width.
4. For Radius Y, TopLeft + BottomLeft/TopRight + BottomRight should not be greater than field's height.
5. If customer's settings break rules 3 and 4, the TopRight or BottomLeft radius are reduced, keeping the TopLeft and BottomRight settings, as it is. For example, if the field's size is (200, 100) with TopLeft: (150, 80), BottomRight: (150, 80), TopRight: (100, 50), and BottomLeft: (100, 50), then TopRight will change to (50, 20) and BottomLeft will change to (50, 20).

## Selecting, Moving, and Copying Fields

Working with OIBRWDesigner > Enhancing the Report with Fields > Selecting, Moving, and Copying Fields

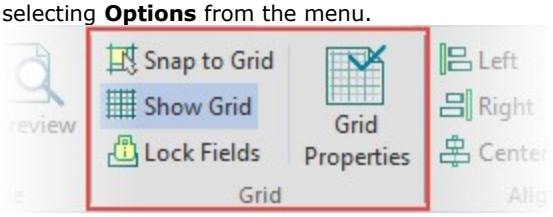
You can use the mouse to select fields in the **OIBRWDesigner** application as usual:

- Click a field to select it.
- Shift-click a field to toggle its selected state.
- Control-drag creates a copy of the selected fields.
- Click the empty area and drag your mouse pointer to select multiple fields.
- With your mouse pointer, drag field corners to resize fields.
- Double-click right or bottom field corners to auto size the field.

To select fields that intersect vertical or horizontal regions of the report, click and drag the mouse on the rulers along the edges of the Designer. If fields are small or close together, it may be easier to select them by name. You can select fields and sections by picking them from the drop-down list above the Properties window.

**Show a grid**

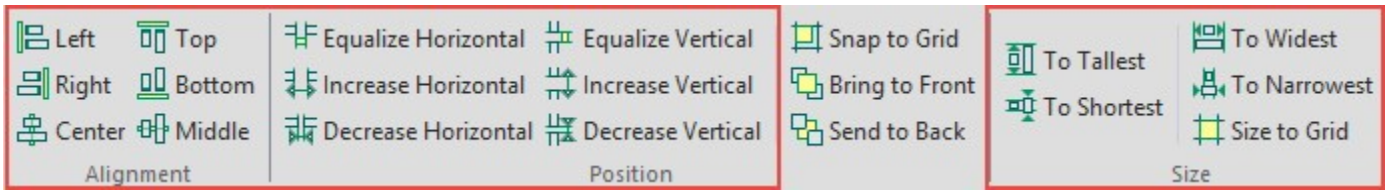
The **Snap to grid** and **Show grid** buttons located in the **Grid** group in the **Arrange** tab provide a grid that helps position controls at discrete positions. While the grid is on, the top left corner of the fields will snap to the grid when you create or move fields. You can change the grid units (English or metric) by clicking the **File** menu and selecting **Options** from the menu.




**Lock fields**

After you have placed fields in the desired positions, you can lock them to prevent inadvertently moving them with the mouse or keyboard. Use the **Lock Fields** button to lock and unlock the fields.

**Format fields**

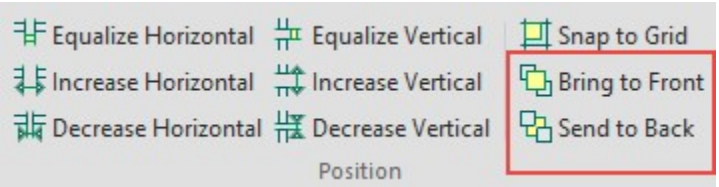



When multiple fields are selected, you can use the buttons on the **Alignment**, **Position**, and **Size** groups of the **Arrange** tab to align, resize, and space them. When you click any of these buttons, the last field in the selection is used as a reference and the settings are applied to the remaining fields in the selection.

**Apply styles**


The **Report Styles** button applies the style of the reference field to the entire selection. The style of a field includes all font, color, line, alignment, and margin properties. You can use the Properties window to set the value of individual properties to the entire selection.

**Determine order for overlapping fields**




If some fields overlap, you can control their z-order using the **Bring to Front/Send to Back** buttons in the **Position** group. This determines which fields are rendered before (behind) the others.

**Move fields using the keyboard**
The **OIBRWDesigner** application also allows you to select and move fields using the keyboard:

- Use the TAB key to select the next field.
- Use SHIFT-TAB to select the previous field.
- Use the arrow keys to move the selection one pixel at a time (or shift arrow to by 5 pixels).
- Use the DELETE key to delete the selected fields.
- When a single field is selected, you can type into it to set the **Text** property.

## Changing Field, Section, and Report Properties

Working with OIBRWDesigner > Enhancing the Report with Fields > Changing Field, Section, and Report Properties

Once an object is selected, you can use the Properties window to edit its properties.

When one or more fields are selected, the Properties window shows property values that all fields have in common, and leaves the other properties blank. If no fields are selected and you click on a section (or on the bar above a section), the **Section** properties are displayed. If you click the gray area in the background, the **Report** properties are displayed.

To see how this works, click the label in the Header section and change its Font and ForeColor properties. You can also change a field's position and dimensions by typing new values for the Left, Top, Width, and Height properties.

The Properties window expresses all measurements in *twips* (the native unit used by OIBRW), but you can type in values in other units (in, cm, mm, pix, pt) and they will be automatically converted into *twips*. For example, if you set the field's Height property to **0.5in**, the Properties window will convert it into 720 *twips*.

## *Creating a Master-Detail Report Using Subreports*

Working with OIBRWDesigner > Creating a Master-Detail Report Using Subreports

Subreports are regular reports contained in a field in another report (the main report). Subreports are usually designed to display detail information based on a current value in the main report, in a master-detail scenario.

In the following example, the main report contains categories and the subreport in the Detail section contains product details for the current category:

**Beverages**

*Soft drinks, coffees, teas, beers, and ales*

| Product Name | Quantity per Unit | Unit Price | Units in Stock | Units on Order |
|---|---|---|---|---|
| Chai | 10 boxes x 20 bags | $18.00 | 39 | 0 |
| Chang | 24 - 12 oz bottles | $19.00 | 17 | 40 |
| Guaraná Fantástica | 12 - 355 ml cans | $4.50 | 20 | 0 |
| Sasquatch Ale | 24 - 12 oz bottles | $14.00 | 111 | 0 |
| Steeleye Stout | 24 - 12 oz bottles | $18.00 | 20 | 0 |
| Côte de Blaye | 12 - 75 cl bottles | $263.50 | 17 | 0 |
| Chartreuse verte | 750 cc per bottle | $18.00 | 69 | 0 |
| Ipoh Coffee | 16 - 500 g tins | $46.00 | 17 | 10 |
| Laughing Lumberjack Lager | 24 - 12 oz bottles | $14.00 | 52 | 0 |
| Outback Lager | 24 - 355 ml bottles | $15.00 | 15 | 10 |
| Rhönbräu Klosterbier | 24 - 0.5 l bottles | $7.75 | 125 | 0 |
| Lakkalikööri | 500 ml | $18.00 | 57 | 0 |

**Condiments**

*Sweet and savory sauces, relishes, spreads, and seasonings*

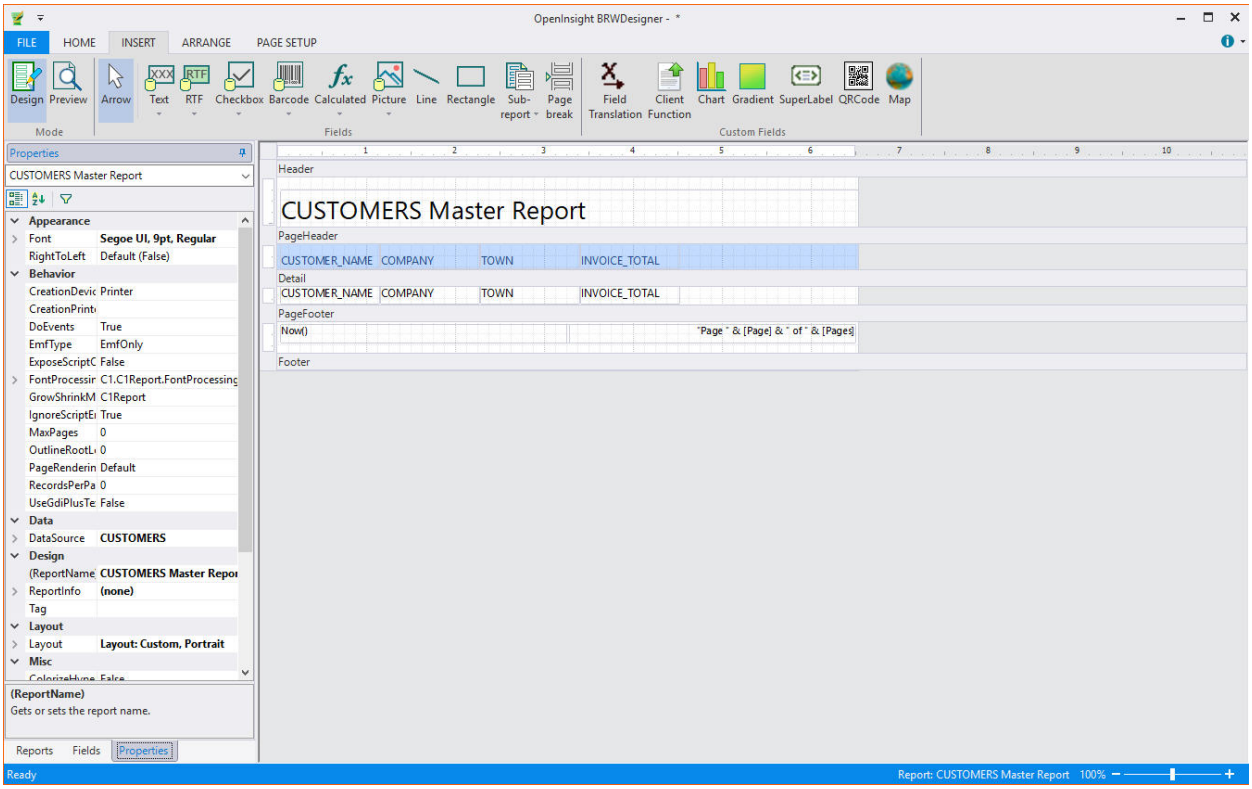
| Product Name | Quantity per Unit | Unit Price | Units in Stock | Units on Order |
|---|---|---|---|---|
| Aniseed Syrup | 12 - 550 ml bottles | $10.00 | 13 | 70 |
| Chef Anton's Cajun Seasoning | 48 - 6 oz jars | $22.00 | 53 | 0 |
| Chef Anton's Gumbo Mix | 36 boxes | $21.35 | 0 | 0 |
| Grandma's Boysenberry Spread | 12 - 8 oz jars | $25.00 | 120 | 0 |

Page 1 of 5

To create a master-detail report based on the **Categories** and **Products** tables, you need to create a Categories report (master view) and a Products report (details view).

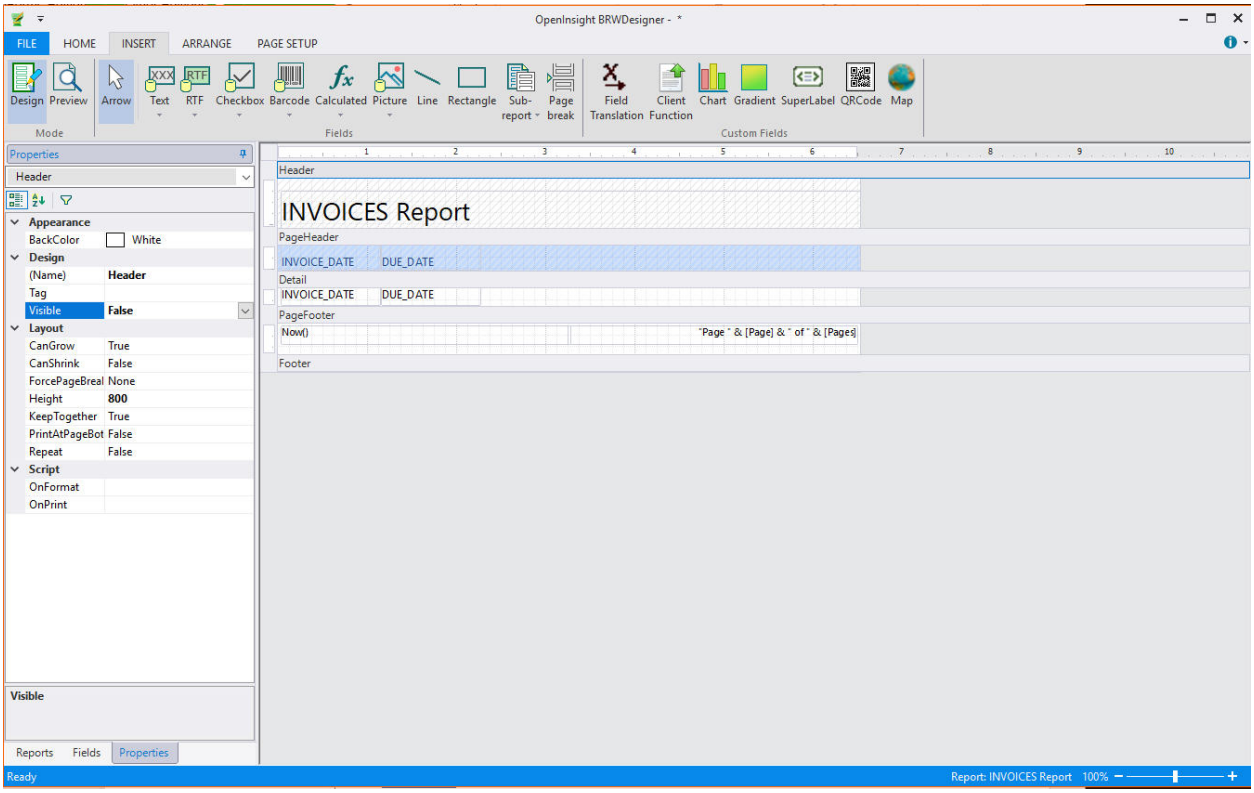
## *Step 1: Create the master report*

1. Create a basic report definition using the **OIBRW Wizard**.
    1. Select the **CUSTOMERS** table.
    2. Include the **CUSTOMER_NAME, COMPANY, TOWN** and **INVOICE_TOTAL** fields in the report.
2. In the **OIBRW** application, click the **Design** button to begin editing the report.
3. The Customers report should now look similar to the following image:




## *Step 2: Create the detail report*

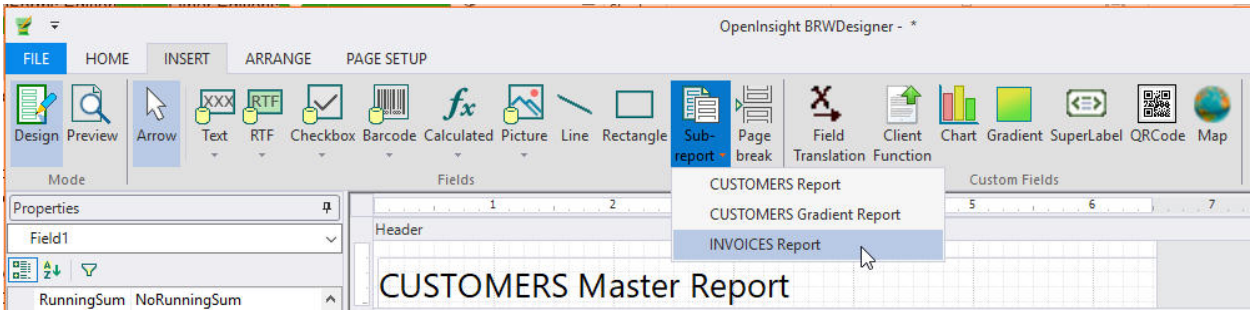
1. In the **OI BRWDesigner** application, click the **New Report** button to create a basic report definition using the **OI BRW Wizard**.
   a. Select the **Invoices** table.
   b. Include the following fields in the report: **Invoice_Date**, and **Due_Date**.
2. In the Report Designer, click the **Design** button to begin editing the report.
   a. Set the Page Header, Page Footer, and Header section's **Visible** property to **False**.
   b. In the Detail section, arrange the controls so that they are aligned with the heading labels. Use the Properties window to change the Appearance settings.



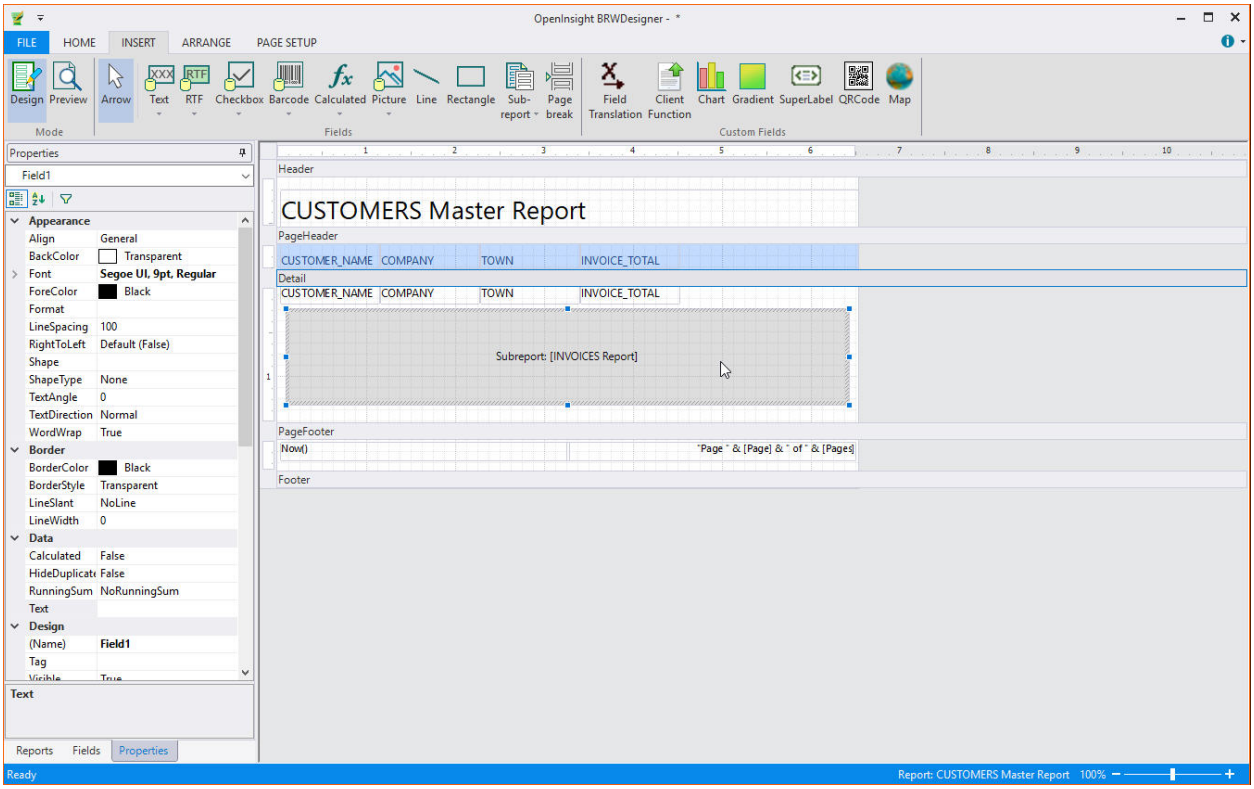

## *Step 3: Create the subreport field*

The **OI BRWDesigner** application now has two separate reports**, Customers Report** and **Invoices Report**. The next step is to create a subreport:

1. From the Reports list in the Designer, select **Customer Report** (master report).
2. In design mode, click the **Sub-report** in the **Fields** group of the **Insert** tab and select **Invoice**s **Report** from the drop-down menu.



3. In the Detail section of your report, click and drag the mouse pointer to make the field for the subreport:

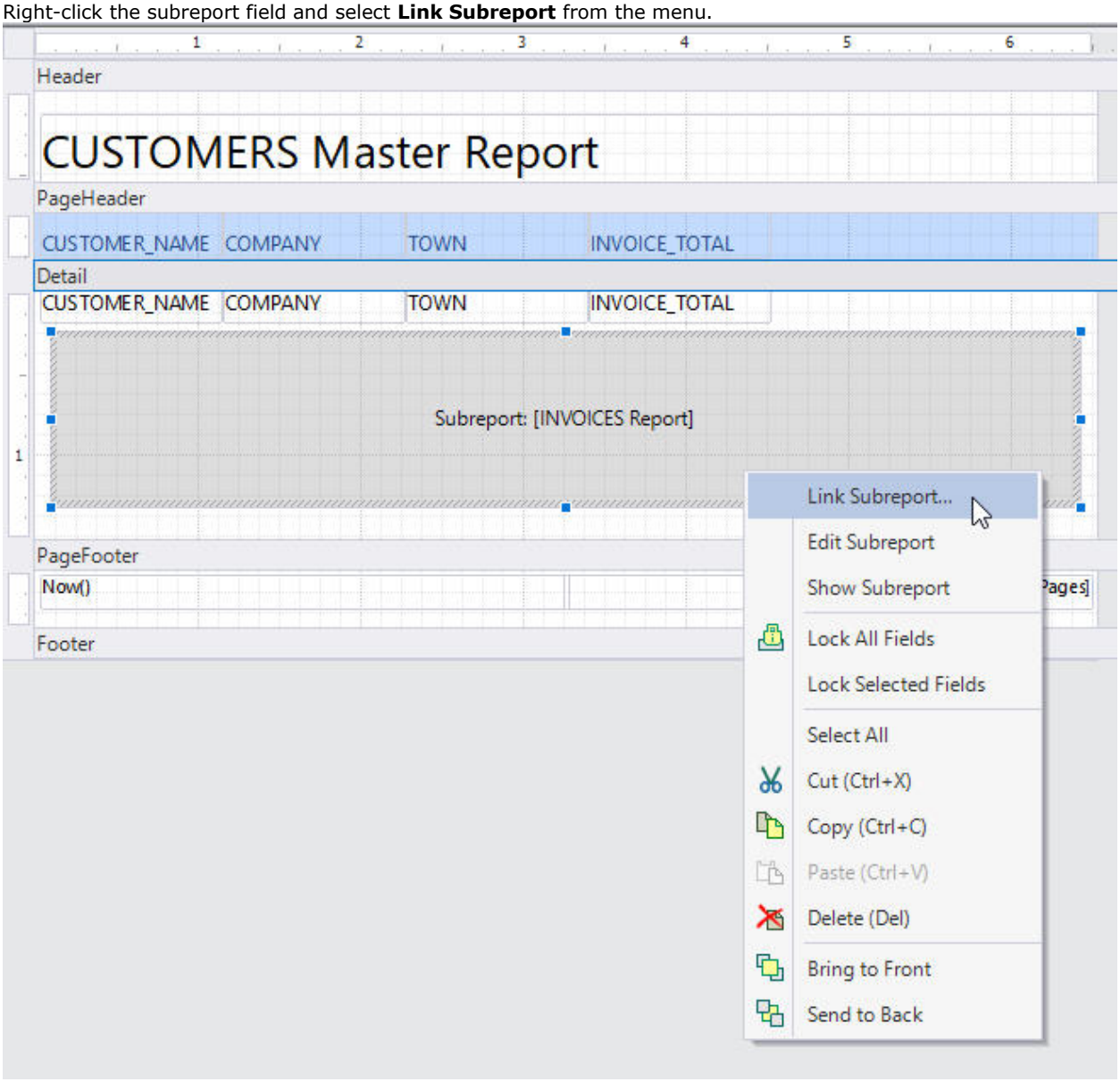



## *Step 4: Link the subreport to the master report*

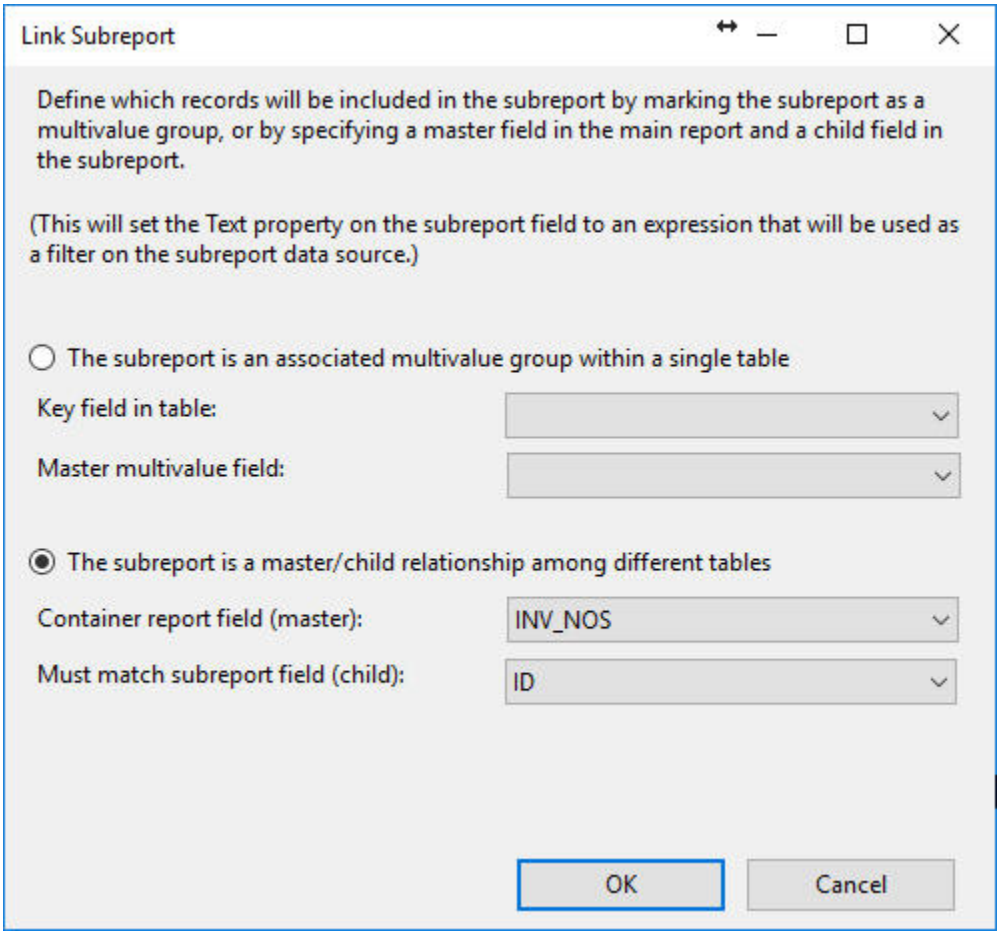
The master-detail relationship is controlled by the Text property of the subreport field. This property should contain an expression that evaluates into a filter condition that can be applied to the subreport data source.

The Report Designer can build this expression automatically for you. Complete the following steps:

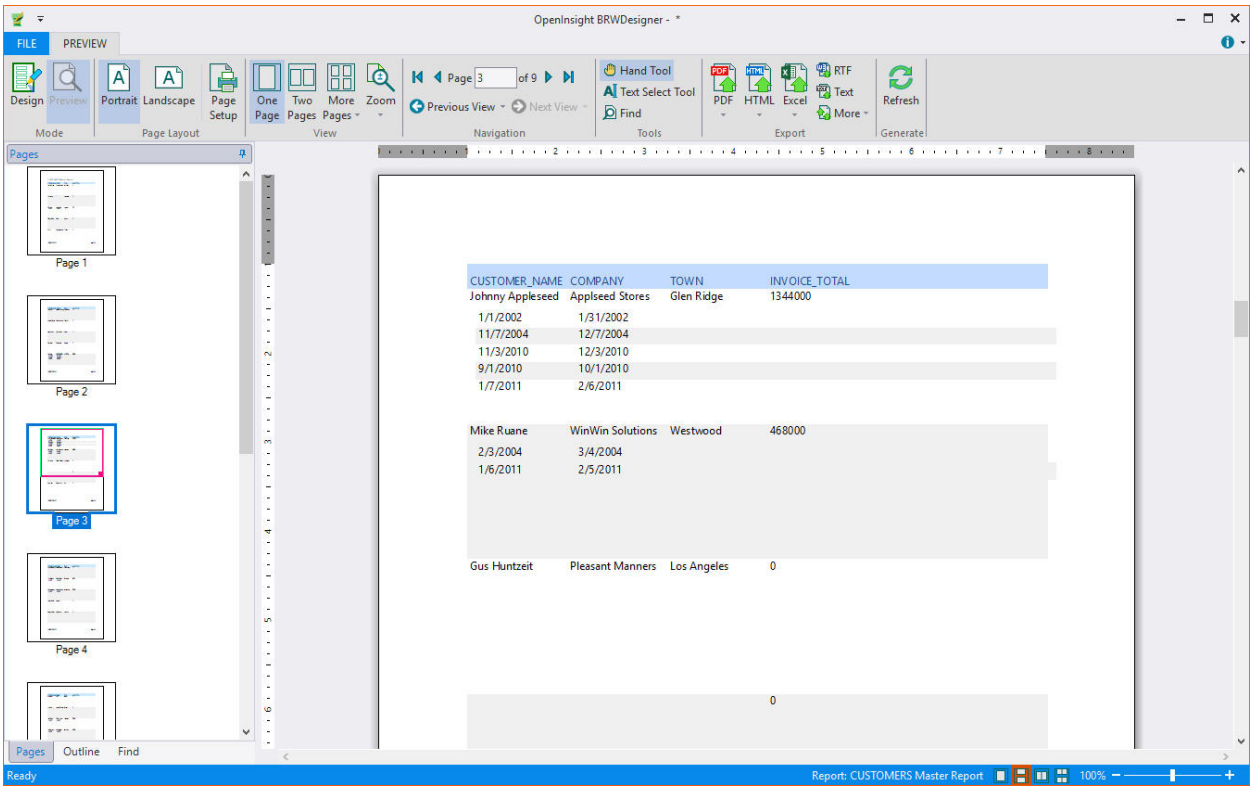
1. Right-click the subreport field and select **Link Subreport** from the menu.



2. A dialog box appears that allows you to select which fields should be linked.




3. The completed Customer Report with Sub-Report will output as follows:

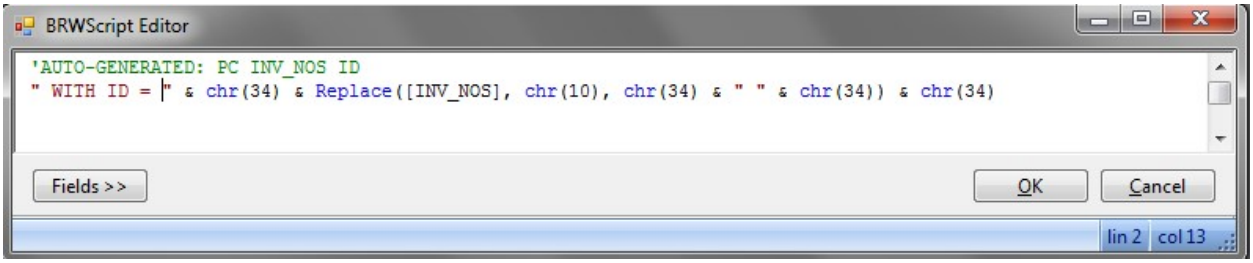


4. Once you make a selection and click **OK**, the Report Designer builds the link expression and assigns it to the Text property of the subreport field. In this case, the expression is:

```
"[CategoryID] = '" & [CategoryID] & "'"
```

Alternatively, you can also link the subreport to the master report by completing the following steps:

1. From the Properties window, click the Text property of the subreport field and select **Script Editor** from the drop-down list.



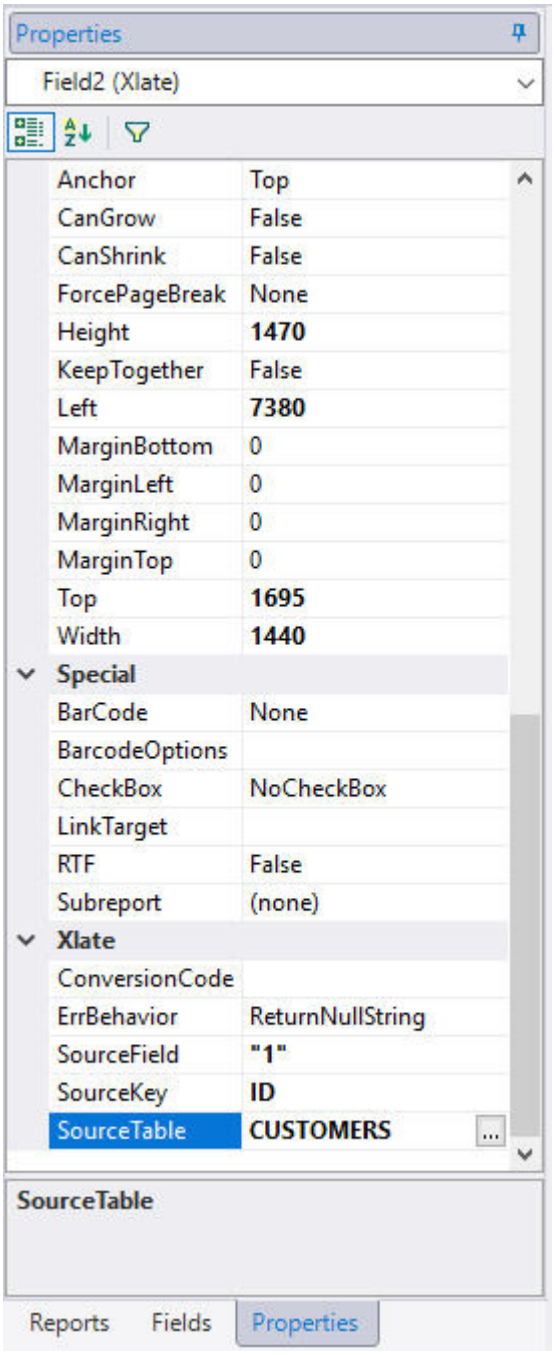

2 . The Auto Generated Comment is used to populate the Link Subreport screen. In general, it is recommended that you only use the Link Subreport screen.

3. Click **OK** to close the VBScript Editor and build the expression.

## *XLATE Support*

In the BRW Designer, a new "custom field" allows for the definition of an XLATE (file translate). The XLate custom field is found in the upper-right corner of the BRW Designer, and has an icon of an "X" over an arrow:


Once you select the Xlate custom field, your cursor should change to the crosshairs, and you should be able to draw the outline of the custom field on the form wherever you would like it to appear. Once the custom field has been applied to the form, you must set its properties (found on the "Properties" tab):




The SourceTable property specifies what "foreign table" should be used as the source for the Xlate. This can be a literal string indicating the name of the table, or a function that evaluates to the name of the table. You can use the BRW Report Designer script editor to define this script, and can use any of the available fields, variables, or script functions to build the table name.

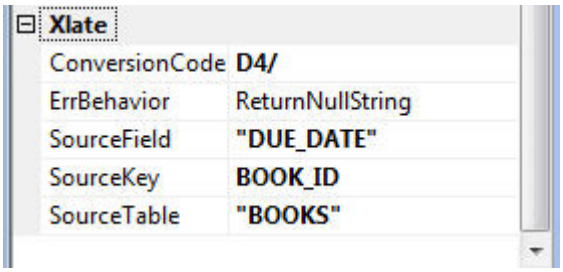
The SourceKey property specifies what key should be used to access the record in the SourceTable. This can be a literal string to use as the key, or a function that evaluates to a key in the table. You can use the BRW Report Designer script editor to define this script, and can use any of the available fields, variables, or script functions to build the record key.

The SourceField property defines which field should be returned from the record found in the SourceTable. This can be a literal string (either a field number, or the name of a dictionary field in the SourceTable), or a function that evaluates to a field number or name. You can use the BRW Report Designer script editor to define this script, and can use any of the available fields, variables, or script functions to build the field number or name. Note that if SourceField evaluates to the null string (""), the entire record is returned. If any multivalues (or multiple fields)

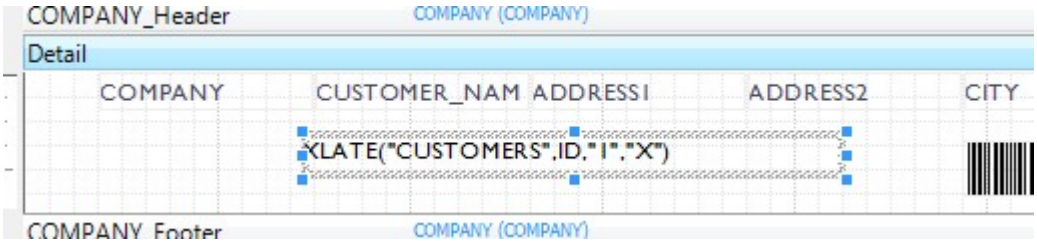
are returned by the Xlate, they will be converted to multiple lines of output on the form (you may need to set the CanGrow property to see these additional lines of output).

The ErrBehavior property defines what should be returned if the Xlate function is unable to retrieve the desired data. ReturnNullString (the default) will return a null string ("") if the Xlate fails. If ErrBehavior is set to ReturnKey, the value of the SourceKey will instead by returned if the Xlate fails.

In OpenInsight 10.0 and above there is a new property "ConversionCode" that is part of Xlate. Just set that value to whatever OpenInsight conversion you want on your data:



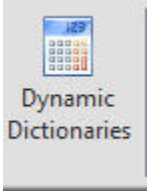

Once the custom field is fully defined, it is drawn as a "placeholder" on the form:



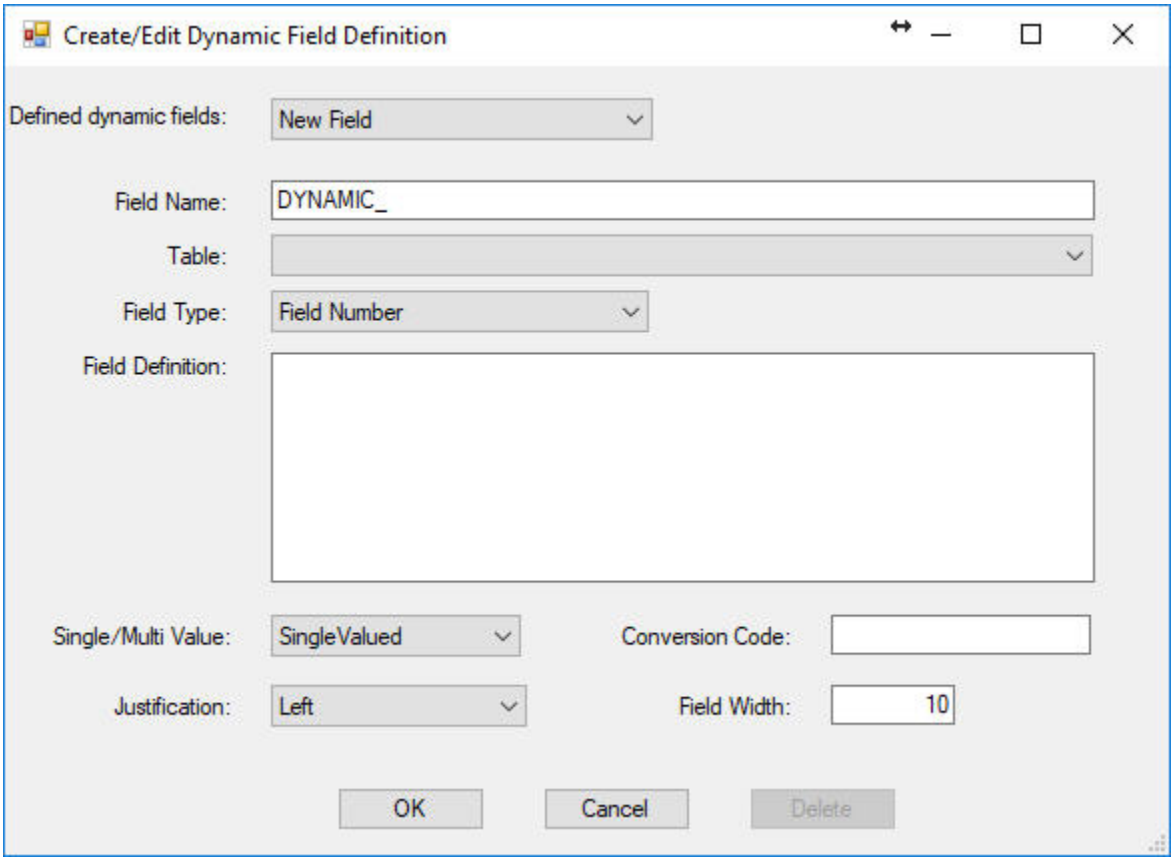

When the BRW report is run, the custom field properties are all evaluated (if they aren't literal strings), and are passed to the host, where the XLATE is performed. Your report will then include the output from the XLATE:

## *Dynamic Dictionaries*

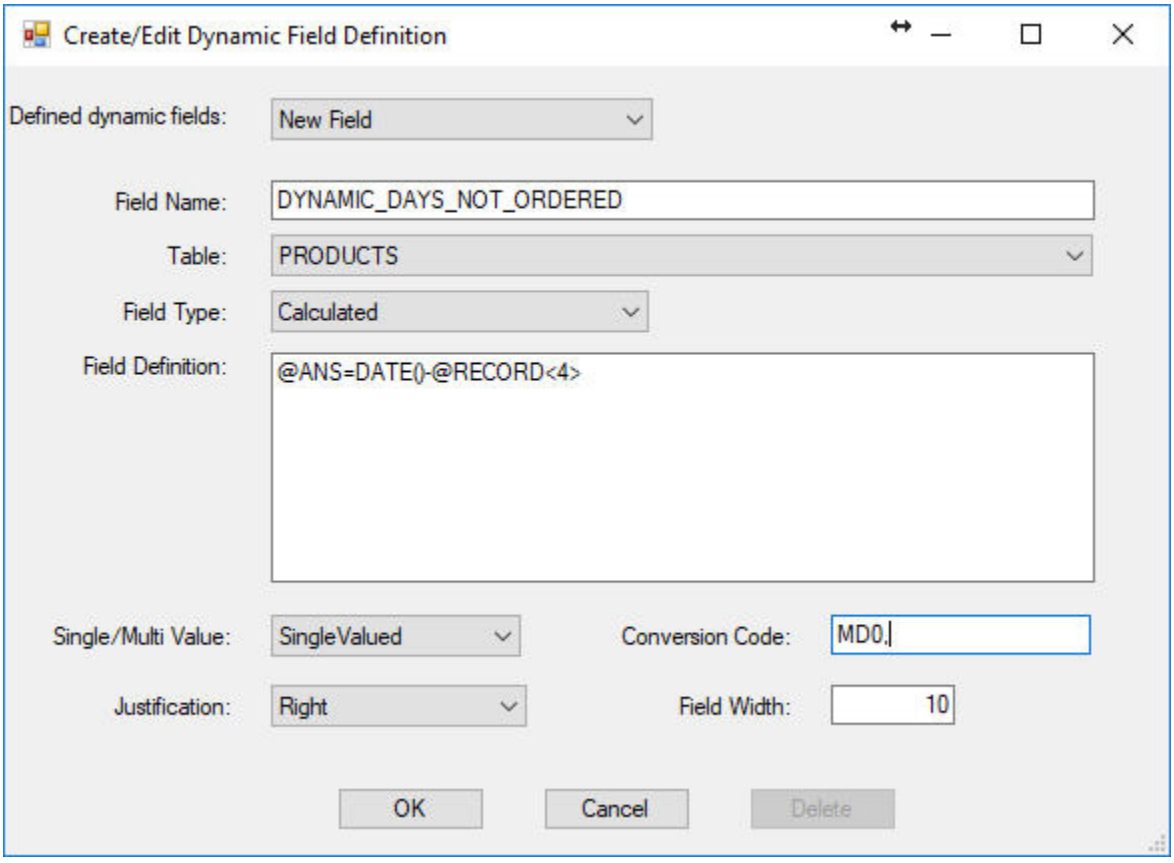
In the BRW Designer released with OpenInsight 10.0, "dynamic" dictionary creation is now supported. In the report definition, click on the Dynamic Dictionaries button:



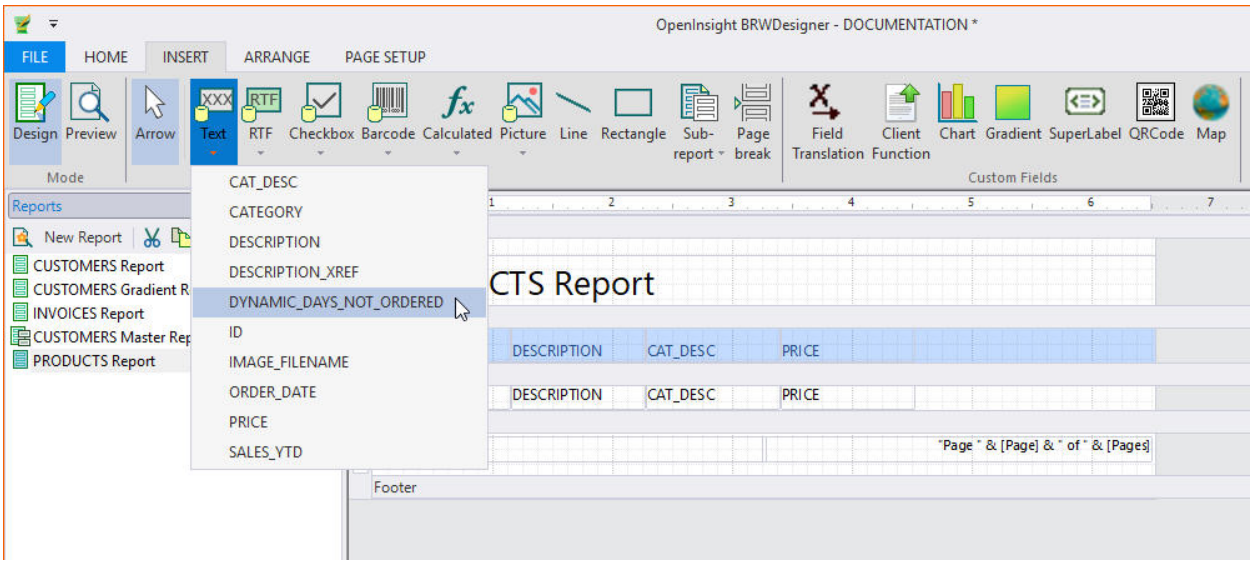

Select either an existing dynamic field to edit, or 'new field' to define a New Field, in the Create/Edit Dynamic Field Definition form:

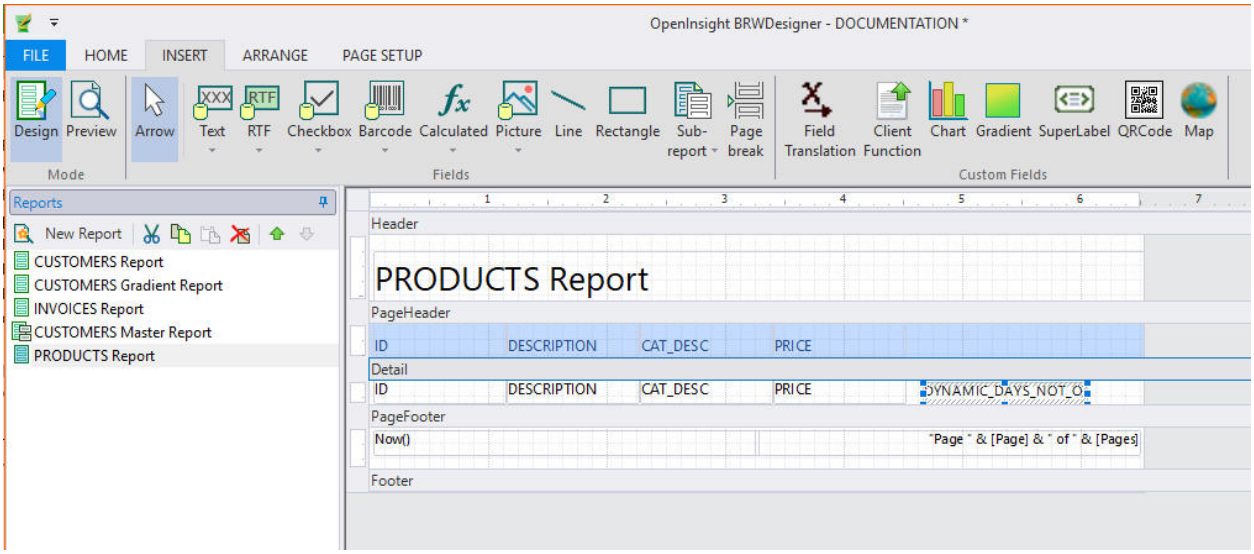



Specify all the relevant definition information (for a calculated field, build the calculation; for a regular field definition, specify the field number; etc.)

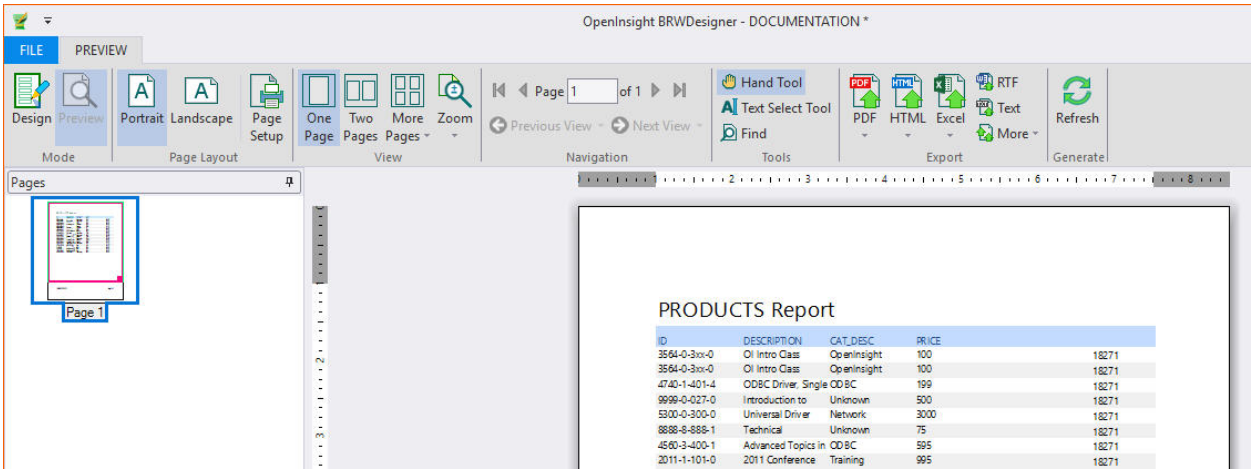



Once saved, the dictionary is available in the list of table fields, and can be used in the report, in scripts, in groups, in the filter, etc.






When run, the dynamic dictionary is created (using a unique temporary ID on the server) and runs just like a 'real' dictionary.

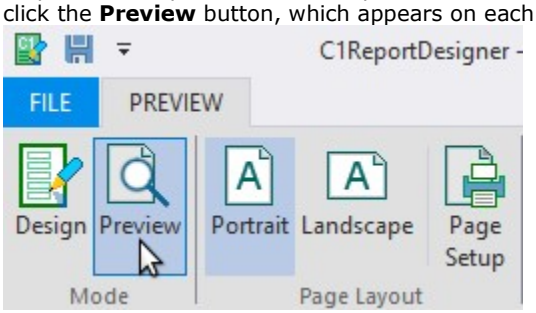



When the report is finished, the dynamic dictionary is cleaned up. The dynamic dictionary that is created has the name _OIBRW_<guid> (e.g. _OIBRW_AS0AS82KASDF8292). The dynamic dictionary information is stored in the report group. Report groups with these changes will now be written with "v2" in field 1, and the dynamic dictionary information stored in field 2. The same dynamic dictionary definition can be used for different reports in the same group, if they share the same table.

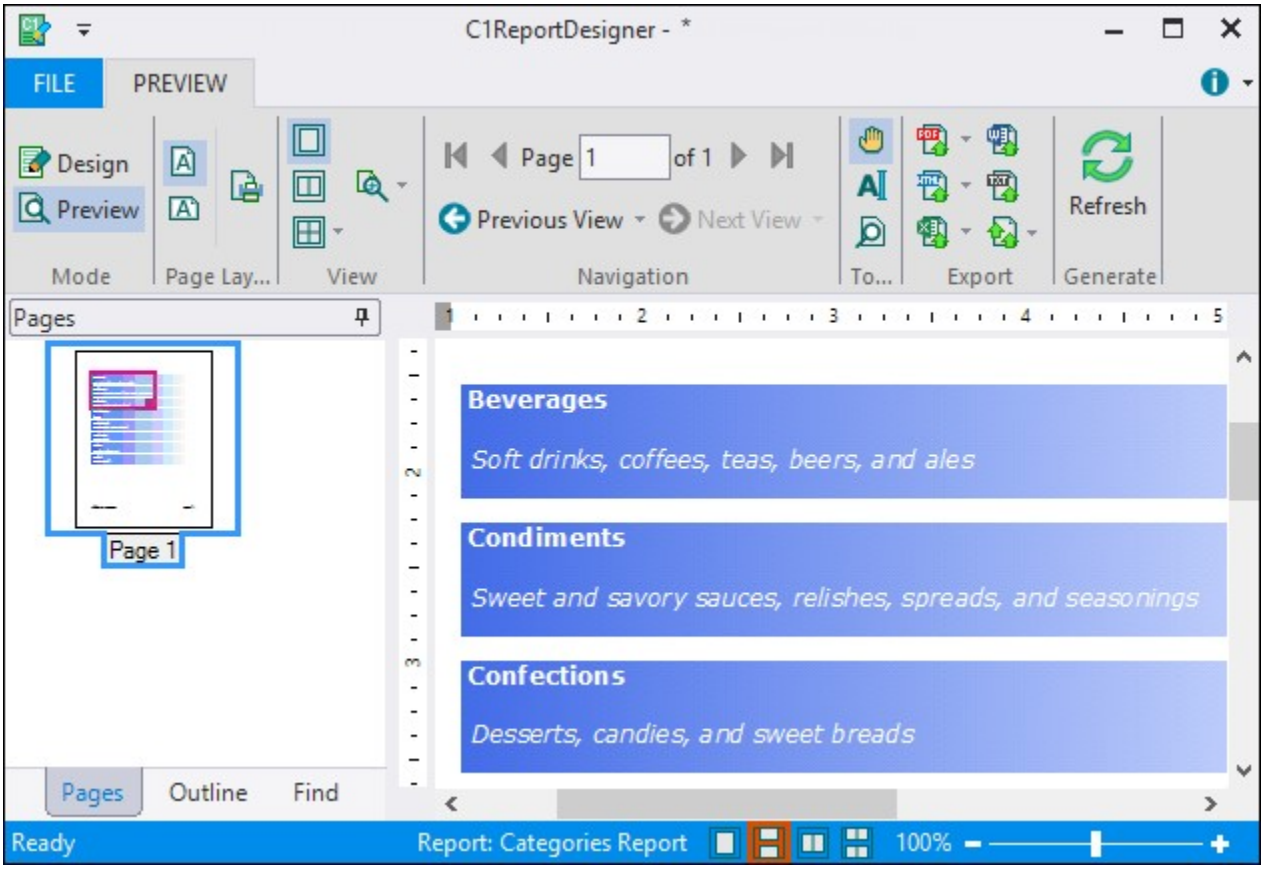
## *Previewing and Printing a Report*

Working with OIBRWDesigner > Previewing and Printing a Report

To preview a report, select the report to view from the Reports list on the left pane of the Designer window and click the **Preview** button, which appears on each Ribbon tab:

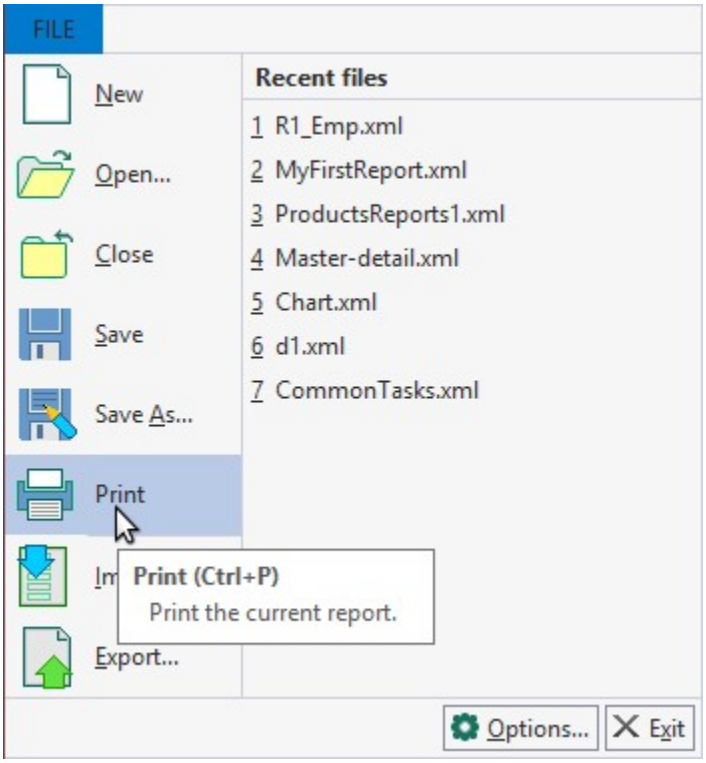



The report is displayed on the right pane, as shown in the following screen shot:




The main window has a preview navigation toolbar, with buttons that let you page through the document and select the zoom mode.

At this point, you can print the report by clicking **File** menu and then **Print**.



## *Exporting and Publishing a Report*

Working with OIBRWDesigner > Exporting and Publishing a Report

Instead of printing the report, you may want to export it into a file and distribute it electronically to your clients or co-workers. The Designer supports the following export formats:

| Format | Description |
|---|---|
| Paged HTML (*.htm) | Creates one HTML file for each page in the report. The HTML pages contain links that let the user navigate the report. |
| Drilldown HTML (*.htm) | Creates a single HTML file with sections that can be collapsed and expanded by the user by clicking on them. |
| Plain HTML (*.htm) | Creates a single HTML file with no drill-down functionality. |
| Table-based HTML (*.htm) | Creates a table-based HTML file that avoids use of absolute positioning. |
| PDF with system fonts (*.pdf) | Creates a PDF file that can be viewed on any computer equipped with Adobe's Acrobat viewer or browser plug-ins. |
| PDF with embedded fonts (*.pdf) | Creates a PDF file with embedded font information for extra portability. This option significantly increases the size of the PDF file. |
| RTF (*.rtf) | Creates an RTF file that can be opened by most popular word processors (for example, Microsoft Word, WordPad). |
| RTF with fixed positioning (*.rtf) | Creates an RTF file with fixed positioning that can be opened by most popular word processors (for example, Microsoft Word, WordPad). |
| Microsoft Excel 97 (*.xls) | Creates an XLS file that can be opened by Microsoft Excel. |
| Microsoft Excel 2007/2010 Open XML (*.xlsx) | Creates an XLS file that can be opened by Microsoft Excel 2007 and later. |
| TIFF (*.tif) | Creates a multi-page TIFF (Tag Image File Format) file. |
| Text (*.txt) | Creates a plain text file. |
| Single Page Text (*.txt) | Creates a single-page plain text file. |
| Compressed Metafile (*.txt) | Creates a compressed metafile text file. |
| XML Paper Specification(*.xps) | Creates a file of XPS format |
| ComponentOne OpenXml Document (*.c1dx) | Creates a file of C1DX format. |

Please note that the export version of PDF that is supported is 1.3 and above.

To create an export file, select **File | Export** from the menu and use the **Export Report to File** dialog box to specify the location, **File name** and **Save as type**.

**Note:** When a document is exported to the RTF or the DOCX formats with the "preserve pagination" option selected, text is placed in text boxes and the ability to reflow text in the resulting document may be limited.

## *Managing Report Definition Files*

Working with OIBRWDesigner > Managing Report Definition Files

A report definition file may contain several reports. Occasionally, you may want to copy or move a report from one file to another.

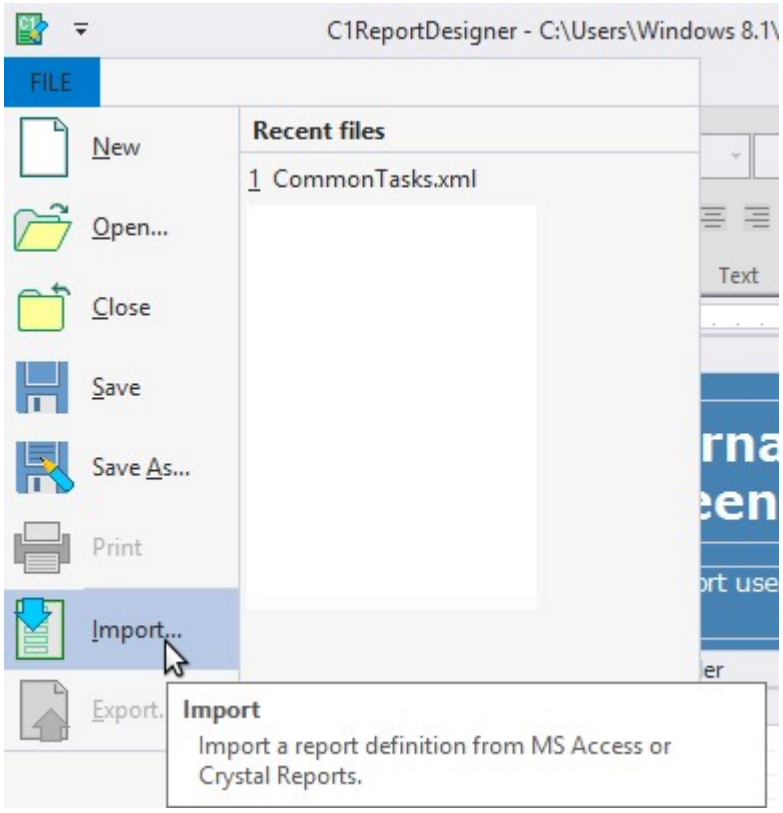
To move a report from one file to another, open two instances of the **OIBRWDesigner** application, find the report you wish to move/duplicate in the list of reports, right-click on the report name and choose "cut" to move, or "copy" to duplicate, the report. Then go to the second instance of the BRWDesigner, right click in the list of report names, and choose "paste".

You can also copy a report within a single file. This creates a new copy of the report, which is a good way to start designing a new report that is similar to an existing one.

Note that the report definition files are saved in XML, so you can also edit and maintain them using any text editor.

## *Importing Crystal Reports*

Working with OIBRWDesigner > Importing Crystal Reports
The **OIBRWDesigner** application can import Crystal report definition files (.rpt files).



To import reports from a Crystal report definition file:

1. Click the **File** menu and select **Import**. The **Import Report Definition** dialog box opens and prompts you for the name of the file you want to import.
2. Select a Crystal report definition file (.rpt). The **OIBRWDesigner** application converts the report into the OIBRW format.

The **OIBRWDesigner** application supports some of the following conversions on import of Crystal Reports:

- Reports bound to internal or external data sources can be imported and run without any changes required to the original data source path.
- Date, time, and number formats are retained during conversion.
- Expressions with percentage aggregates are sucessfully imported.
- **RAS API** is supported for TextObject conversion of some expressions such as those where a single Textbox containing combination of text and other expressions (**Text + [Expression]**) are to be interpreted.

Note: Before you import the report, please ensure that you have compatible versions of Crystal Reports and Visual Studio insta

Also note that if you have Crystal Reports 2013 installed on the system, then on conversion of Crystal Report to OIBRW, the da

xtreme.mdb will have to be changed manually in order to run the report.

The import process handles most elements of the source reports, with a few exceptions for elements that are not exposed by the Crystal object model or not supported by OIBRW. The exceptions include image fields, charts, and cross-tab fields.

## *Charting in Reports*

Working with OIBRWDesigner > Charting in Reports

Aggregate charting is a powerful, yet simple and easy-to-use feature. **Reports for WinForms** supports chart fields using its extensible custom field architecture. The **Chart** field is implemented as a custom field in the C1.Win.OIBRW.CustomFields.2.dll assembly, which is installed with the report designer application and is also included as a sample with full source code (**CustomFields**). In the following topics, you'll see how you can customize chart fields in reports using the **OIBRWDesigner** application. The **OIBRWDesigner** application is installed with both **Reports for WinForms** and **Reports for WPF**.

For more information on this topic, see the blog post on Charting in Reports for WinForms

**See Also**

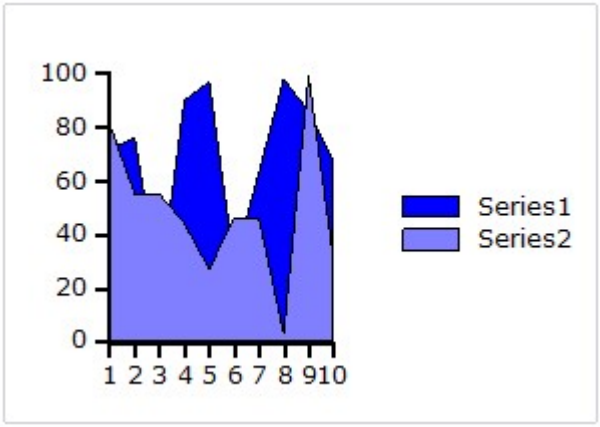
Plotting Data in Charts
Chart Properties
Charts with Multiple Series
Charts in Grouped Reports
Creating Aggregate Charts
Chart Types

## Chart Types

Working with OIBRWDesigner > Charting in Reports > Chart Types

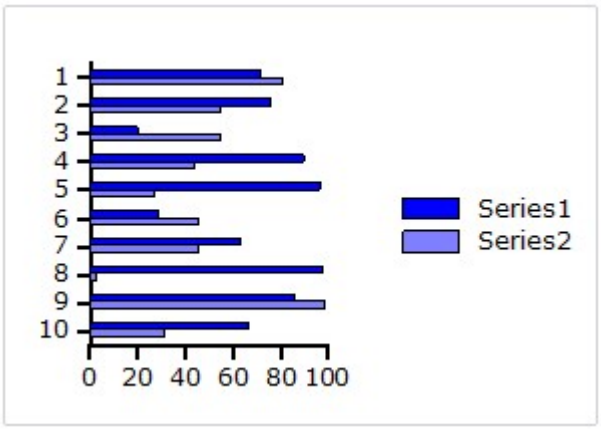
The **Chart Field** in **Report for WinForms** allows you to insert various types of charts using ChartTypeEnum. There are ten chart types that are supported in **OIBRW**: **Area**, **Bar** (horizontal bars), **Column** (vertical columns), **Scatter** (X-Y values), **Line**, **Pie**, **Step**, **Histogram**, **Radar**, and **Polar**. The chart types can be easily selected using the **ChartType** property in the Properties pane of the **OIBRWDesigner**.

**Area chart:** An Area chart draws each series as connected points of data, filled below the points. Each series is drawn on top of the preceding series.

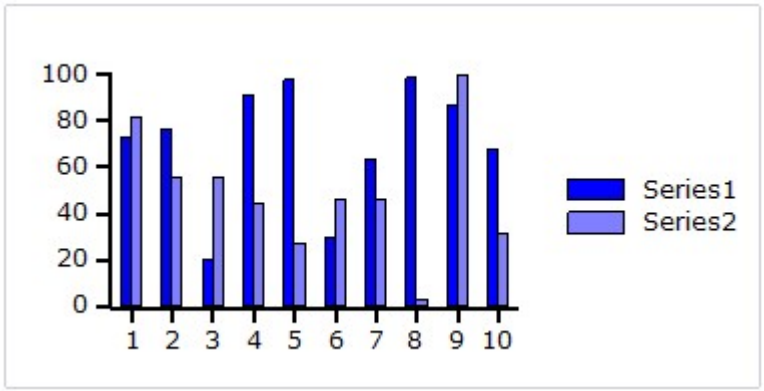



**Bar and Column charts:** A Bar chart or a Column chart represents each series in the form of bars of the same color and width, whose length is determined by its value. Each new series is plotted in the form of bars next to the bars of the preceding series. A Bar or Column chart draws each series as a bar in a cluster. The number of clusters is the number of points in the data. Each cluster displays the nth data point in each series. When the bars are arranged horizontally, the chart is called a bar chart and when the bars are arranged vertically, the chart is called column chart.

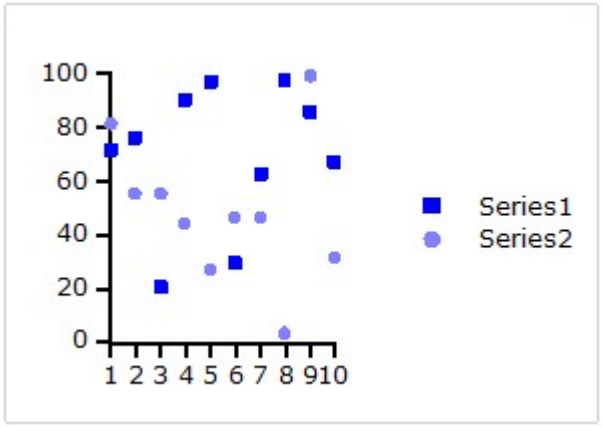
The following image represents a **Bar** chart:



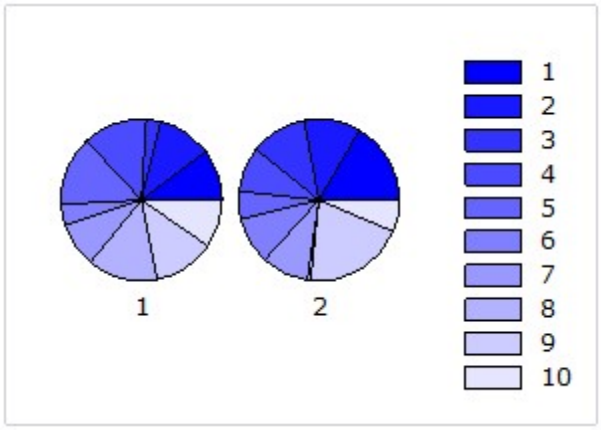

The following image represents a **Column** chart:



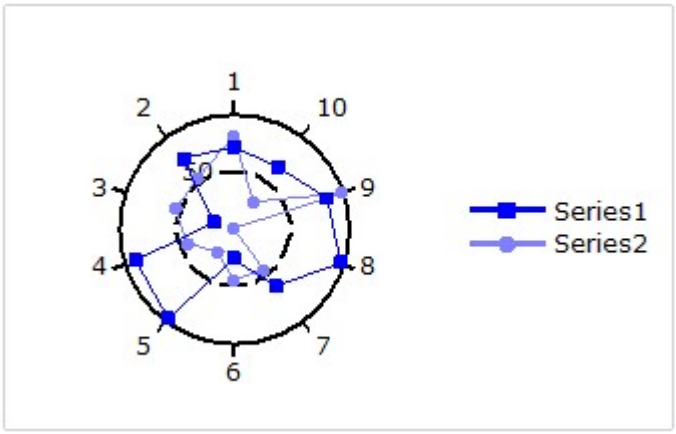

**Scatter chart:** A Scatter chart uses two values to represent each data point. This type of chart is often used to support statistical techniques that quantify the relationship between the variables (typically Linear Regression Analysis).



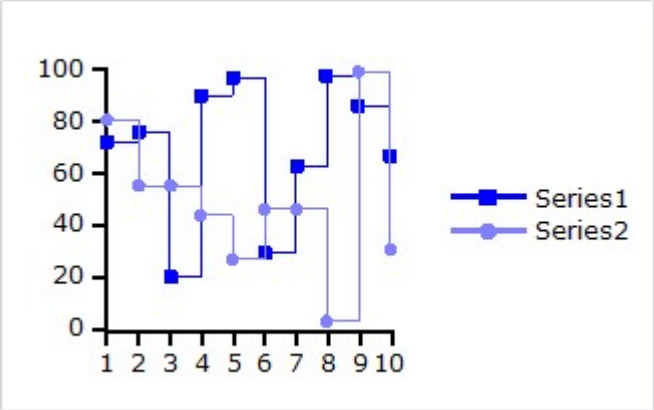

**Pie chart:** A Pie chart draws each series as a slice in a pie. The number of pies is the number of points in the data. Each pie displays the nth data point in each series. You can also customize Pie charts for displaying legends and labels; see Chart Properties for more information.



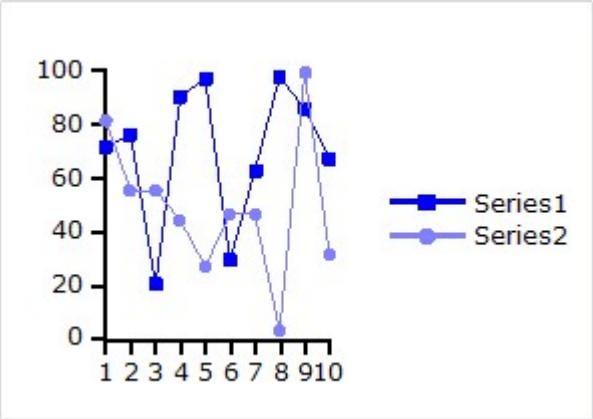

**Radar chart:** A Radar chart draws the y value in each data set along a radar line (the x value is ignored except for labels). If the data has n unique points, then the chart plane is divided into n equal angle segments, and a radar line is drawn (representing each point) at n/360 degree increments. By default, the radar line representing the first point is drawn vertically (at 90 degrees). Radar charts can be further customized; see Chart Properties for more information.



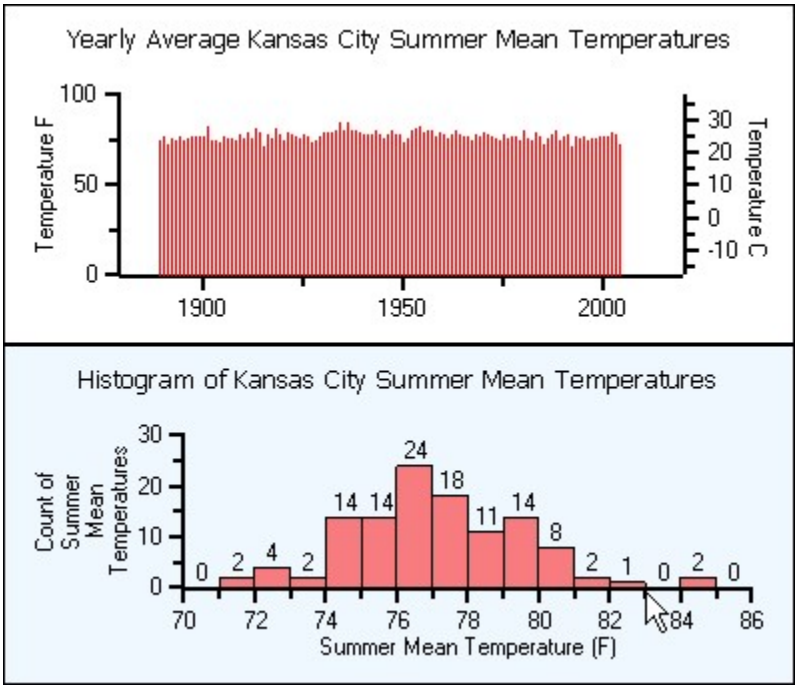

**Step chart:** A Step chart is a form of XY plot chart that draws series as connected points of data. These charts are often used when Y values change by discret amounts, at specific values of X with a sudden change of value. A simple, everyday example would be a plot of a checkbook balance with time. As each deposit is made, and each check is written, the balance (Y value) of the check register changes suddenly, rather than gradually, as time passes (X value). During the time that no deposits are made, or checks written, the balance (Y value) remains constant as time passes.




**Line chart:** A Line chart draws each series as connected points of data. It is the most effective way of denoting changes in values between different groups of data. These charts are commonly used to show trends and performance over time.



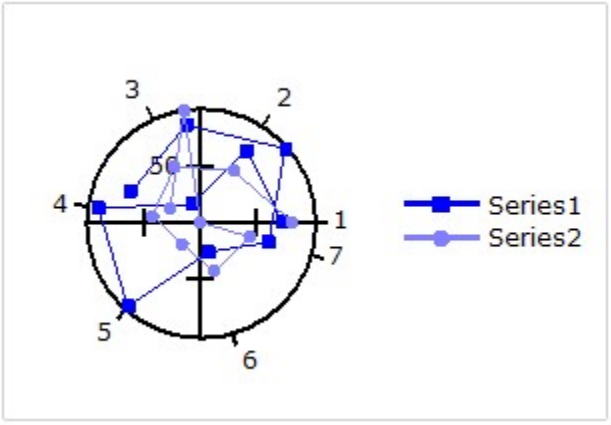

**Histogram chart:** A Histogram chart takes a collection of raw data values and plots the frequency distribution. It is frequently used with grouped data, which is generated by measuring a collection of raw data and plotting the number of data values that fall within defined intervals. Note that raw values are not used to generate data for a histogram, but are used to generate a frequency instead. While showing similarities to bar charts, it is important to note that histograms are used with quantitative variables whereas bar charts are commonly used with qualitative variables.
While the histogram and bar charts' appearances relate, their functionality does not. A bar chart is created from data points whereas a Histogram is created from the frequency distribution of the data. The charts following illustrate the difference between a bar chart and a histogram chart. Both of the charts use exactly the same Y data. The bar chart (top) shows each average mean temperature for each year in which it occurred. The histogram chart (bottom) using the same input temperature data automatically tabulates the number of temperatures that fall within each interval and draws the resulting histogram. For convenience, chart labels with the count in each interval have been added at the top of each interval.




A histogram is beneficial for pinpointing prominent features of the distribution of data for a quantitative variable. The important features for a quantitative variable include the following:

- It reveals the typical average value.
- The data yields a general shape. The data values can be distributed symmetrically around the middle or they can be skewed.
- If there are distant values from the group of data it shows them as outlier values.
- The data values can be near or far to the typical value.
- The distribution may result in a single peak or multiple peaks and valleys.

**Polar chart**: A Polar chart draws the x and y coordinates in each series as (theta,r), where theta is amount of rotation from the origin and r is the distance from the origin. Theta may be specified in either degrees (default) or radians. Since the X-axis is a circle, the X-axis maximum and minimum values are fixed. The series can be drawn independently, or stacked. Polar charts can be further customized; see Chart Properties for more information.




## Chart Properties

Working with OIBRWDesigner > Charting in Reports > Chart Properties

The appearance of charts can be customized by using several other chart properties provided by the Chart field. Few chart properties that are commonly used are as follows:

- **Aggregate:** This property allows you to create charts that automatically aggregate data values that have the same category using an aggregate function of your choice (sum, average, standard deviation, and so on). See Creating Aggregate Charts for more details.
- **DataColor**: This property selects the color used to draw the bars, columns, areas, scatter symbols, and pie slices. If the chart contains multiple series, then the **Chart** field automatically generates different shades of the selected color for each series. If you want to select specific colors for each series, use the **Palette** property instead, and set its value to a semi-colon separated list containing the colors to use (for example "Red;Green;Blue").
- **DataX, DataY:** These properties allows you to set the fields and the plot data required to be displayed in the chart. See Plotting Data in Charts for more details.
- **FormatX, FormatY**: These properties determine the format used to display the values along each axis. For example, setting **FormatY** to "c" causes the **Chart** field to format the values along the Y axis as currency values. This is analogous to the **Format** property in regular report fields.
- **XMin, XMax, YMin, YMax**: These properties allow you to specify ranges for each axis. Setting any of them to -1 cause the **Chart** to calculate the range automatically. For example, if you set the **YMax** property to 100, then any values higher than 100 will be truncated and won't appear on the chart.

These properties apply to all chart types. There are some additional properties which are specific to the chart types.

The following properties apply only to **Pie** charts:

- **ShowPercentages**: Each pie slice has a legend that shows the X value for the slice. If the **ShowPercentages** property is set to true, the legend will also include a percentage value that indicates the size of the slice with respect to the pie. The percentage is formatted using the value specified by the **FormatY** property. For example, if you set **FormatY** to "p2", then the legends will include the X value and the percentage with two decimal points (for example "North Region (15.23%)").
- **RadialLabels**: This property specifies that instead of showing a legend on the right side of the chart, labels with connecting lines should be attached to each slice. This works well for pies with few slices (up to about ten).

The following properties apply to **Radar** charts:

- **Start**: Specifies the starting angle of these charts on a 360 degree circle. The angle is measured in the counter-clockwise direction. The measuring unit of the start angle is degrees if the **Degrees** property is set to true, otherwise it is radians.
- **Degrees**: Specifies the measuring unit of the starting angle for these charts. If this property is set to true, the measuring unit is degrees, otherwise it is radians.
- **Filled**: The area enclosed by the data points in the radar charts can be set to filled by setting this property to true.
- **FlatGridLines**: The radar charts by default have circular Y coordinate gridlines. Using FlatGridLines property, the gridlines can be set to flat Y coordinate gridlines.

The following properties apply to **Polar** charts:

- **Start**: Specifies the starting angle of these charts on a 360 degree circle. The angle is measured in the counter-clockwise direction. The measuring unit of the start angle is degrees if the **Degrees** property is set to true, otherwise it is radians.
- **Degrees**: Specifies the measuring unit of the starting angle for these charts. If this property is set to true, the measuring unit is degrees, otherwise it is radians.
- **PiRatioAnnotations**: If **Degrees** is set to False and the chart reflects radian values, then C1Chart provides the option of having the chart annotated with ratios of Pi rather than radians. Setting this property to true annotates the values on the polar chart in ratios of Pi.

The following properties apply only to **Histogram** charts, exposed by **HistogramOptions**:

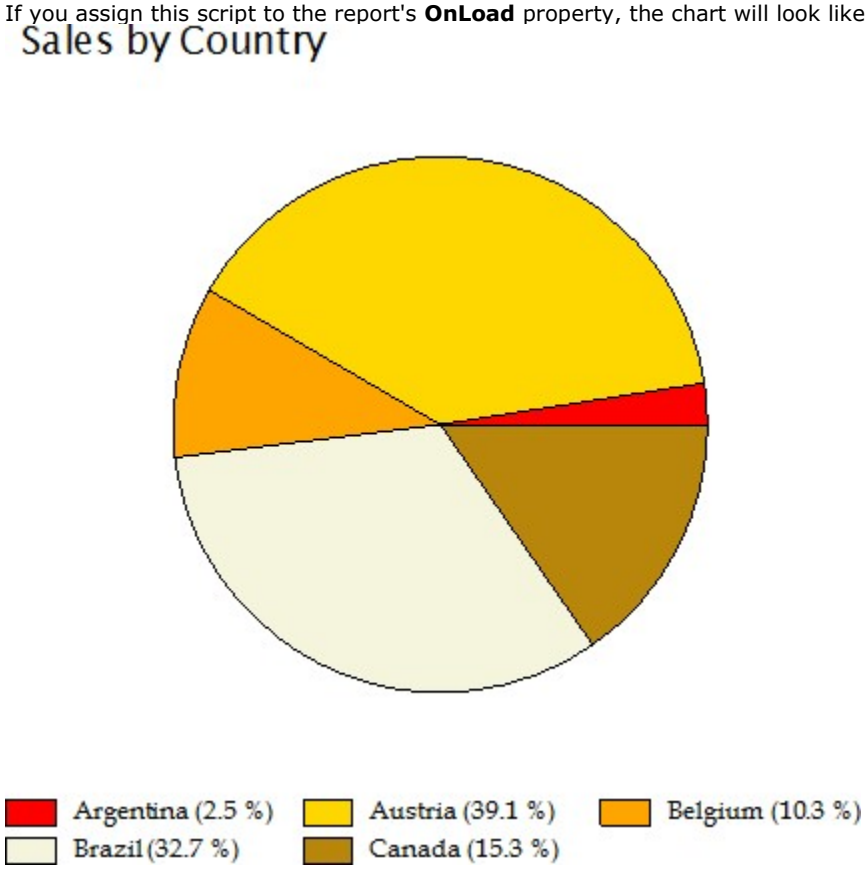
- **DisplayType**: Specifies the method in which frequency data should be displayed for a particular series. This is useful for displaying different frequency data uniquely in a single chart group.
- **IntervalCreationMethod**: This property is used to specify different interval boundaries in histogram charts. You can choose one of the following three methods from the IntervalCreationMethod property:
  - **Automatic**: When the Automatic method is used, the chart calculates the upper and lower limits of the intervals using the maximum and minimum data values, and restricting the intervals to lie within 3 standard deviations of the data mean. The number of intervals is optional. Interval boundaries are calculated uniformly.
  - **SemiAutomatic**: When the SemiAutomatic method is used, the upper and lower limits of the intervals are specified together with the number of intervals. Interval boundaries are calculated uniformly. The IntervalStart, IntervalWidth, and IntervalNumber properties are available when you select the SemiAutomatic method. The IntervalStart property gets or sets the numeric value of the beginning of the first interval.
  - **XDataBoundaries**: When the XDataBoundaries method is used, the X values of the data series are used to explicitly set each interval boundary. The X values are sorted and duplicate values are eliminated. Each ascending value of the result is used determine the next interval boundary. Thus, the first and second resulting X values define the first interval and each successive X value specifies the end of the next interval. Note that specification of N intervals requires N+1 unique X values.

- **IntervalNumber**: Specifies the number of intervals for the histograms created by **Automatic** and **SemiAutomatic** methods.
- **IntervalStart**: Specifies the numeric value of the beginning of the first interval for the histograms created by **SemiAutomatic** method.
- **IntervalWidth**: Specifies the numeric value of width of the interval for the the histograms created by **SemiAutomatic** method.
- **NormalDisplay**: Specifies the properties that are used to display the Normal (Gaussian) curve for comparison with the histogram.
- **NormalizationInterval**: Specifies the normalization interval width for the histograms with non-uniform intervals. It preserves the shape of the histogram by normalizing the width such that each interval height represents the same frequency per unit width.
- **Normalized**: Specifies if each histogram series interval is normalized.

The **Chart** field is actually a wrapper for a **C1Chart** control, which provides all the charting services and has an extremely rich object model of its own. If you want to customize the **Chart** field even further, you can use the ChartControl property to access the inner **C1Chart** object using scripts.
For example, the **Chart** field does not have a property to control the position of the legend. But the **C1Chart** control does, and you can access this property through the **ChartControl** property. For example, the script below causes the chart legend to be positioned below the chart instead of on the right:

```
' place legend below the chart
chartField.ChartControl.Legend.Compass = "South"
```

If you assign this script to the report's **OnLoad** property, the chart will look like the image below:




The other properties used to create these chart are as follows:

**ChartType** = Pie
**FormatY** = "p1"
**ShowPercentage** = true
**Palette** = "Red;Gold;Orange;Beige;DarkGoldenrod;Goldenrod;"

## Charts with Multiple Series

Working with OIBRWDesigner > Charting in Reports > Charts with Multiple Series

To create charts with multiple series, simply set the **DataY** property to a string that contains the names of each data field you want to chart, separated by semi-colons.
For example, to create a chart showing product prices and discounts you would set the **DataY** property as shown below:

**DataY** = "UnitPrice;Discount"

If you want to specify the color used to display each series, set the **Palette** property to a list of colors separated by semi-colons. For example, the value displayed below would cause the chart to show the UnitPrice" series in red and the "Discount" series in blue:

**Palette** = "Red;Blue"

## Charts in Grouped Reports

Working with OIBRWDesigner > Charting in Reports > Charts in Grouped Reports

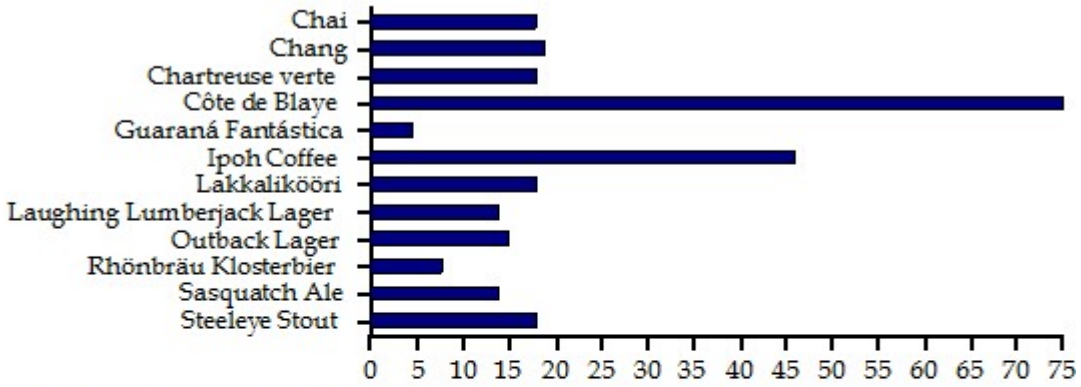
**Reports for WinForms** allows you to create reports with multiple groups. For example, instead of listing all products in a single flat report, you could group products by category. Each group has a header and a footer section that allow you to display information about the group, including titles and subtotals, for example.
If you add a chart to a group header, the chart will display only the data for the current group. By contrast, adding a chart to the report header or footer would include all the data in the report.

To illustrate this, here is a diagram depicting a report definition as shown in the report designer and showing the effect of adding a **Chart** field to the report header and to a group header:

| |
|---|
| **Report Header section**<br><br>*A chart field here would generate only one chart for the entire report.*<br><br>*The chart would show all the data in the report's data source.* |
| **Page Header section** |
| **Group Header section (CategoryName)**<br><br>*A chart field here would generate one chart for each CategoryName value.*<br><br>*Each chart would show all the data for the current CategoryName.* |
| **Detail section** |
| **Group Footer section (CategoryName)** |
| **Page Footer section** |
| **Report Footer section** |

Continuing with the example mentioned above, if you added a chart to the group header and set the **DataX** property to "ProductName" and the **DataY** property to "UnitPrice", the final report would contain one chart for each category, and each chart would display the unit prices for the products in that category.
The images below show screenshots of the report described above with the group headers, the charts they contain, and a few detail records to illustrate:

## Beverages

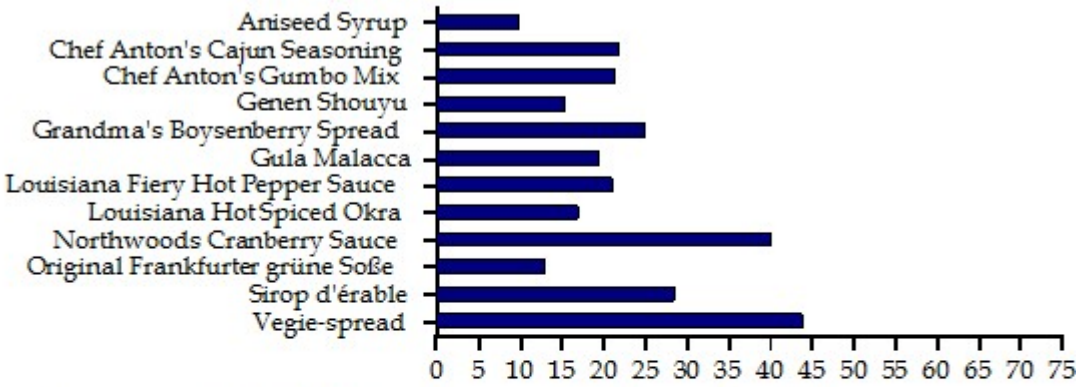



"Unit prices per product: "& CategoryName & " (chart truncated to $75)

| ProductName | QuantityPerUnit | UnitPrice |
|---|---|---|
| Chai | 10 boxes x 20 bags | 18.00 |
| Chang | 24 - 12 oz bottles | 19.00 |
| Chartreuse verte | 750 cc per bottle | 18.00 |
| Côte de Blaye | 12 - 75 cl bottles | 263.50 |
| Guaraná Fantástica | 12 - 355 ml cans | 4.50 |
| Ipoh Coffee | 16 - 500 g tins | 46.00 |

The above chart shows unit prices for products in the "Beverages" category. The below chart shows unit prices for products in the "Condiments" category.

Condiments




"Unit prices per product: "& CategoryName & " (chart truncated to $75)

| ProductName | QuantityPerUnit | UnitPrice |
|---|---|---|
| Aniseed Syrup | 12 - 550 ml bottles | 10.00 |
| Chef Anton's Cajun Seasoning | 48 - 6 oz jars | 22.00 |
| Chef Anton's Gumbo Mix | 36 boxes | 21.35 |
| Genen Shouyu | 24 - 250 ml bottles | 15.50 |
| Grandma's Boysenberry Spread | 12 - 8 oz jars | 25.00 |
| Gula Malacca | 20 - 2 kg bags | 19.45 |

**DataX** = "Product Name"
**DataY** = "Unit Price"

Because the chart automatically selects the data based on the scope of the section that contains it, creating charts in grouped reports is very easy.

## Plotting Data in Charts

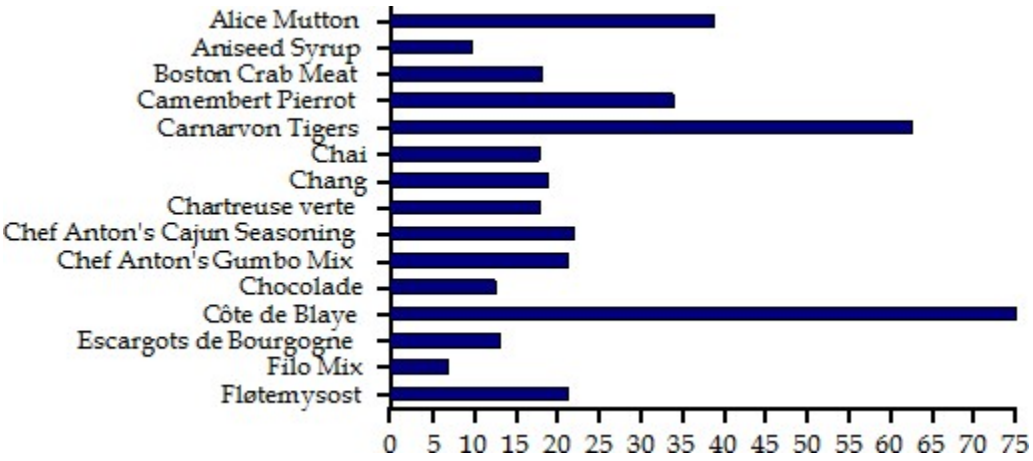
Working with OIBRWDesigner > Charting in Reports > Plotting Data in Charts

To plot data in a simple chart, the properties **DataX** and **DataY** are set as follows:

1. Open the **OIBRWDesigner** application and create or open a report definition file.
2. Add a **Chart** field to the report, then select it to show its properties in the designer's property window.
3. Set the chart's **DataX** property to the name of the field whose values should be displayed in the X axis (chart categories).
4. Set the chart's **DataY** property to the name of the field whose values should be displayed in the Y axis (chart values).
5. Optionally set additional properties such as **ChartType** and **DataColor**.

For example, the chart below was created based on the NorthWind Products table. In this case, the following properties were set:

**DataX** = "ProductName"
**DataY** = "UnitPrice"




Note that for this chart type (Bar), the value axis (where the **DataY** field is displayed) is the horizontal one, and the category axis is the vertical one.

In this case, a filter was applied to the data in order to limit the number of values shown. Without the filter, the chart would contain too many values and the vertical axis would not be readable.

The **DataY** property is not restricted to field names. You can also plot series with calculated values. The strings that specify the series are actually treated as full expressions, and are calculated like any regular field in the report.

For example, to create a chart showing the actual price of each field you could set the **DataY** property to the value shown below:

**DataY** = "UnitPrice * (1 - Discount)"

## Creating Aggregate Charts

Working with OIBRWDesigner > Charting in Reports > Creating Aggregate Charts

The **Chart** field of **Reports for WinForms** has a powerful feature called "aggregated charting". This feature allows you to create charts that automatically aggregate data values (**DataY**) that have the same category (**DataX**) using one of the following aggregate functions:

- Sum
- Count
- Average
- Minimum
- Maximum
- Variance
- VariancePop
- StandardDeviation
- StandardDeviationPop

To illustrate this feature, consider an "Invoices" report that groups data by country, customer, and order ID. The general outline for the report is as follows:

| |
|---|
| **Report Header section** |
| **Page Header section** |
| **Group Header section (Country)** |
| **Group Header section (Customer)** |
| **Group Header section (OrderID)** |

| |
|---|
| **Detail section** |
| **Group Footer section (OrderID)** |
| **Group Footer section (Customer)** |
| **Group Footer section (Country)** |
| **Page Footer section** |
| **Report Footer section** |

Now imagine that you would like to add a chart to each **Country** header displaying the total value of all orders placed by each customer in the current country.
You would start you adding a **Chart** field to the "Country" header section and set the **DataX** and **DataY** properties as follows:

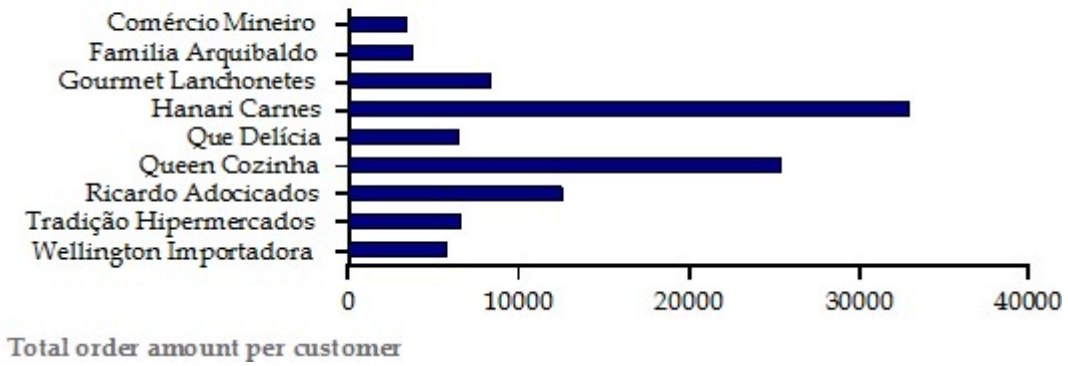
**DataX** = "CustomerName"
**DataY** = "ExtendedPrice"

This would not work. The data for each country usually includes several records for each customer, and the chart would create one data point for each record. The chart would not be able to guess you really want to add the values for each customer into a single data point.

To address this scenario, we added an **Aggregate** property to the **Chart** field. This property tells the chart how to aggregate values that have the same category into a single point in the chart. The **Aggregate** property can be set to perform any of the common aggregation functions on the data: sum, average, count, maximum, minimum, standard deviation, and variance.

Continuing with our example, we can now simply set the chart's **Aggregate** property to "Sum". This will cause the chart to add all "ExtendedPrice" values for records that belong to the same customer into a single data point. The result is shown below:

Brazil



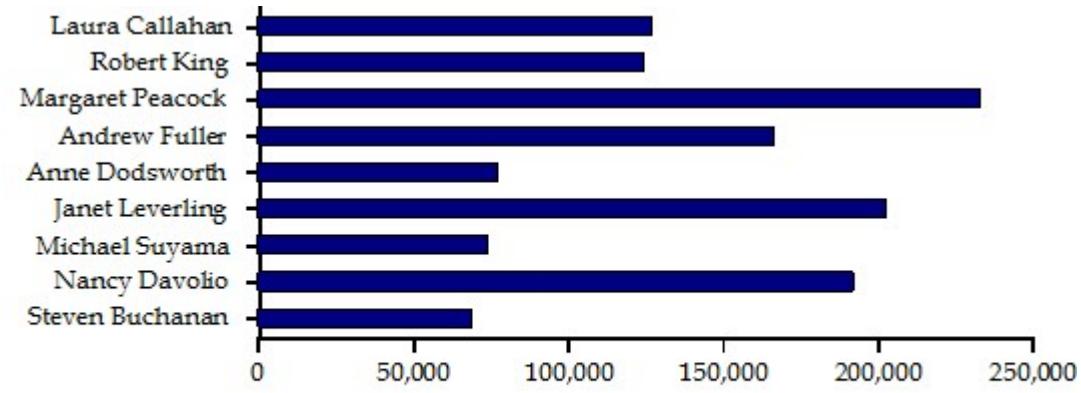

Notice how each customer appears only once. The values shown on the chart correspond to the sum of the "ExtendedPrice" values for all fields with the same "Customer".

Because the chart appears in the "Country" header field, it is repeated for each country, showing all the customers in that country.

If you place the chart in the report header section, it will aggregate data over the entire report. For example, suppose you want to start the "Invoices" report with a chart that shows the total amount ordered by each salesperson. To accomplish this, you would add a **Chart** field to the report header section and would set the following properties:

**DataX** = "Salesperson"
**DataY** = "ExtendedPrice"
**Aggregate** = "Sum"

The image below shows the resulting chart:




Total order amount per Salesperson

Since the chart is placed in the report header section, the values displayed include all countries and all customers. If you moved the chart field from the report header to the "Country" group header, you would obtain a similar chart for each country, showing the total amounts sold by each salesperson in that country.

# Command line interface for the OI BRW

To start up the Banded Report designer, you can do the following:

CALL RTI_BRWSUPPORT("LAUNCH")

Once you have your report(s) defined, you can generate them programmatically using the following:

CALL RTI_BRW_GENERATEREPORT(rptFile, rptName, outName, rptType, ovrListID, rptDetails, bUseGUI, ovrCfg)

where rptFile is the name of the report group you saved the desired report(s) into and rptName is the name of the specific report to run (or "*" to run all the reports in the group). You may also specify a semicolon-delimited list of report names if desired.

OutputName is the path and name of the file to save the output to (if producing a PDF, HTM, etc. document) - leave this blank to generate printed output.

rptType is the type of output to generate, and can be PDF, TIFF, HTML, TEXT, XLS, or XLSX (or PRINT to generate printed output, but that isn't really needed if you leave outputName blank).

overrideListID is the ID of a saved list that contains the record keys you want this report generation to use. If there is no overrideListID specified, but there *is* an active select list when RTI_BRW_GENERATEREPORT is called, then the active list will be used instead.

rptDetails will vary depending on the output type.
For rptType PDF you can optionally specify in rptDetails the access permissions for the PDF, the password(s) for PDF access, and page layout information as follows:
rptDetails<1,1> owner password
rptDetails<1,2> user password
rptDetails<1,3> access permissions. The access permissions are a string that can contain "C" (if copying content is allowed), "E" (if editing content is allowed), "A" (if editing annotations is allowed), and/or "P" (if printing is allowed).
rptDetails<1,4> printer name
rptDetails<1,5> number of copies
rptDetails<1,6> output orientation. Specify landscape ("1") or portrait ("0")
rptDetails<1,7,1> left page margin
rptDetails<1,7,2> right page margin
rptDetails<1,7,3> top page margin
rptDetails<1,7,4> bottom page margin
rptDetails<1,8> page size code
For rptType PRINT you can optionally specify in rptDetails the page layout information as follows:
rptDetails<1,1> printer name
rptDetails<1,2> number of copies
rptDetails<1,3> output orientation. Specify landscape ("1") or portrait ("0")
rptDetails<1,4,1> left page margin
rptDetails<1,4,2> right page margin
rptDetails<1,4,3> top page margin
rptDetails<1,4,4> bottom page margin
rptDetails<1,5> page size code
rptDetails<1,6> output window title
rptDetails<1,7> maximize window. Specify "1" to maximize the output window

For all report types, if field 1, value 1, subvalue 1 is the literal "UID", then field 1, value 1, subvalue 2 contains the unique identifier that the BRW will associate with this invocation. This is useful if you wish to customize the INIT or TERM behavior via a user-modified or created filter routine.

bUseGUI is a flag that indicates whether you want report generation to occur with a GUI (set bUseGUI to "1") or without a GUI, ie, silently (set bUseGUI to "0"). Note that if the output is going to the printer, you can also set bUseGUI to "2" - in this case, you'll get a full print preview window, instead of just a printer control window as you would if bUseGUI is "1". Note that the default if bUseGUI is not specified is "0" (no GUI).

ovrCfg is an "override configuration", a dynamic array which contains the same contents as the CFG_OIBRW record, changed (if needed) for individual circumstances – for example, to force the use of a specific engine. If not needed, please specify null ("") for this parameter.

You can also call, during SET_PRINTER, the LOADREPORT call; this will "embed" generated BRW output into OIPI output (this is *only* available when using VSPRINTER2, aka OIPI.NET). You'll pass in to the SET_PRINTER call the name of the report group, the name of the report to include ("*" for all in the group, or a semicolon-delimited list), the overrideListName, and a flag (0/1) to indicate whether the BRW report is appended to the current output (1) or

replaces the current output (0).

Note that in both cases (LOADREPORT in OIPI.NET, or RTI_BRW_GENERATEREPORT) you can if desired pass in a full report definition (which is just an XML string) in the "report group" parameter. In this way, you can actually access the report definition and modify it, or create one entirely programmatically. Note that the BRW identifies a full report definition by the presence of @FMs in the passed-in string, so be sure to delimit your XML string with @FMs.

If you wish to modify an existing BRW report, you can use the following to read an existing report definition:

rptdef = RTI_BRWSUPPORT("READ", reportGroupName, bLockFlag)

where reportGroupName is the name of the report group, and bLockFlag is a flag to indicate whether the record should be locked (1) or left unlocked (0). Please note, however, that the "READ" call returns a CRLF delimited XML string, rather than an @FM delimited string, so you should SWAP \0D0A\ WITH @FM in the returned result. Also note that the READ result will *not* include, and the BRW will not support, any custom dictionary definitions stored with the report group when the report definition XML is passed in explicitly.

By default, the RTI_BRW_GENERATEREPORT routine can generate one or more documents of the same output type (PDF, print, XLS, etc.) that are all contained in a single report group. For example, to invoke RTI_BRW_GENERATEREPORT to build a PDF based on the report "Sample Report" in the report group "SAMPLES":

```
DECLARE FUNCTION RTI_BRW_GENERATEREPORT
reportGroup = "SAMPLES"
reportName = "Sample Report"
outputName = "C:\TEMP\OUTPUT.PDF"
reportType = "PDF"
bUseGUI = "0"
rslt = RTI_BRW_GENERATEREPORT(reportGroup, reportName, outputName, reportType, '', '', bUseGUI, '')
```

If you wish to build multiple reports (contained in the same report group) of the same output type, multiple report names can be specified in a single request, using the semicolon delimiter. For example, to build a PDF based on the reports "Sample Report" and "Example Output", both contained in the "SAMPLES" report group:

```
DECLARE FUNCTION RTI_BRW_GENERATEREPORT
reportGroup = "SAMPLES"
reportName = "Sample Report;Example Output"
outputName = "C:\TEMP\OUTPUT.PDF"
reportType = "PDF"
bUseGUI = "0"
rslt = RTI_BRW_GENERATEREPORT(reportGroup, reportName, outputName, reportType, '', '', bUseGUI, '')
```

If you wish to generate multiple different output types (for example, both PDF and print output) or output from different report groups, these can also be created with a single call to RTI_BRW_GENERATEREPORT by using the record mark delimiter (@RM) to delimit each report file, report name collection, output name, report type, override list name, and GUI flag.

For example, to build a PDF based on the report "Sample Report" in the report group "SAMPLES", and an on-screen display of the report "Display Report" in the report group "DISPLAYS":

```
DECLARE FUNCTION RTI_BRW_GENERATEREPORT
reportGroup = "SAMPLES":@rm:"DISPLAYS"
reportName = "Sample Report":@rm:"Display Report"
outputName = "C:\TEMP\OUTPUT.PDF":@rm:""
reportType = "PDF":@rm:"PRINT"
bUseGUI = "0":@rm:"2"
rslt = RTI_BRW_GENERATEREPORT(reportGroup, reportName, outputName, reportType, '', '', bUseGUI, '')
```

As seen in the above examples, the RTI_BRW_GENERATEREPORT stored procedure is a function that can return a status value so the developer can determine the results from the request. Note that many of the available status values (and the parameter modifications, described below) will only be set when the BRW is configured to use a new engine to generate its output; when configured to 'share' the current engine (the default), most of these values will not be returned.

The return values are:

0: success
-1: Either parameter "reportFile" or "reportName" is null ("")
1: Report definition is null – specified report is not on file
2: unable to open SYSLISTS table
<error text> : error returned by call to the BRW component
<ole status>: error returned by call to the BRW component

In addition, the RTI_BRW_GENERATEREPORT routine may modify the passed-in parameters for "reportName" and "outputName". If multiple reports are specified in the reportName parameter (either by using the "*" wildcard, or explicitly by using multiple report names separated by semicolons (";")), when the RTI_BRW_GENERATEREPORT completes, the reportName parameter will be changed to a full list of all the individual report names that were accessed (@VM delimited), and the outputName parameter will be changed to a full list of all the individual output files that were generated (one per report name), @VM delimited.

RTI_BRWSUPPORT is a handy function with a few different actions. We've already discussed LAUNCH and READ; you can also call it with the following actions:

DISPLAY: pass in the report group name and specific report name (in parameters 2 and 3) and the report will be displayed via OIPI.NET;

REPORTS: this will return, as the result of the function call, an @VM delimited list of the report groups defined for your application

**Revelation Software, Inc**
99 Kinderkamack Road, First Floor
Westwood, NJ 07675
U.S.A
Toll Free: 800-262-4747
Phone: 201-594-1422
Fax: 201-722-9815
www.revelation.com

**Revelation Software Ltd.**
Boundary House
Boston Road
London, W7 2QE
U.K.
Phone: +44 0 208 912 1000
Fax: +44 0 208 912 1001
www.revsoft.co.uk

**BrightIdeas New Zealand**
44 Cockle Bay Rd
Howick
Auckland, 2014
New Zealand
Phone: +64 9 534 9134

info@revelationsoftware.asia

Revelation Software is a division of Revelation Technologies, Inc.

Part No 118-982