Auto Count Sdn Bhd

Learning AutoCount Accounting Report Designer

Choo, Chin Peng

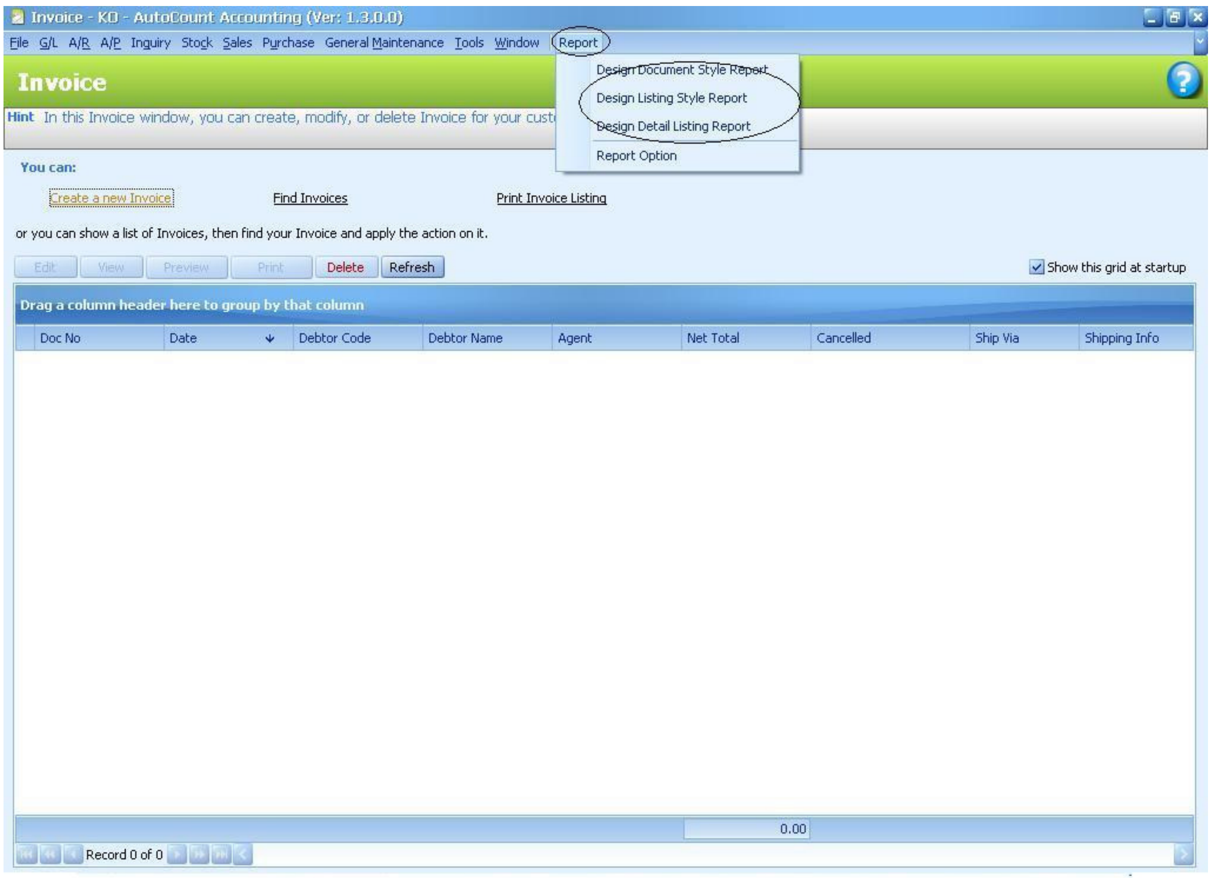
2/13/2008

Table of Contents

1. How to start customizing a report?

1.1. Start Report Designer from individual function Design Report menu

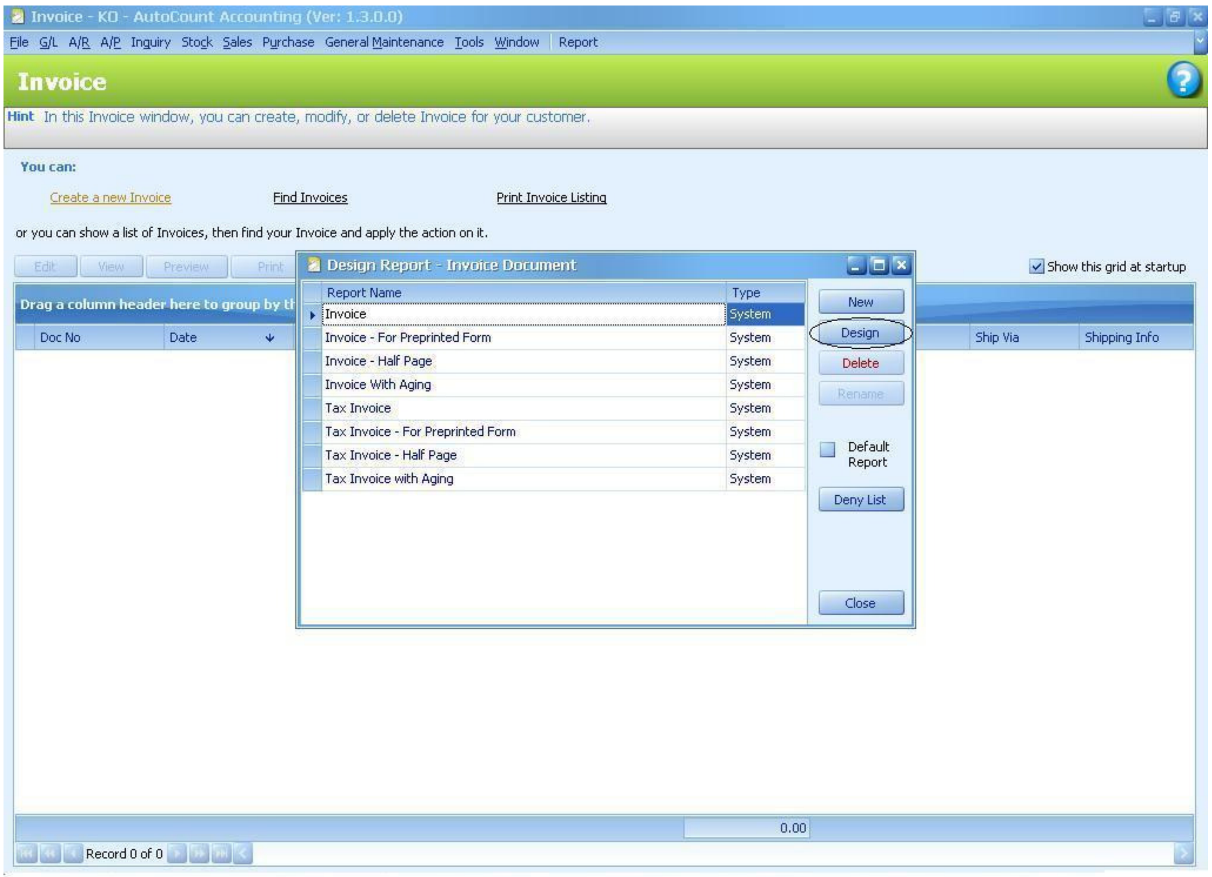
If the system default report cannot fulfill your requirement, you can start redesign or customize the report with the built-in Report Designer. You can find a Report menu in the function you have selected and under the Report menu, there is a list of sub-menu display such as Design Document Style Reports. The following illustrates the Design Document Style Report in Invoice List window.


Invoice - KD - AutoCount Accounting (Ver: 1.3.0.0)
File G/L A/R A/P Inquiry Stock Sales Purchase General Maintenance Tools Window Report
Design Document Style Report
Design Listing Style Report
Design Detail Listing Report
Report Option
Invoice
You can:
Create a new Invoice
Find Invoices
Print Invoice Listing
Edit View Preview Print Delete Refresh
Show this grid at startup
Drag a column header here to group by that column
Doc No
Date
Debtor Code
Debtor Name
Agent
Net Total
Cancelled
Ship Via
Shipping Info
0.00
Record 0 of 0

Hint In this Invoice window, you can create, modify, or delete Invoice for your cust

or you can show a list of Invoices, then find your Invoice and apply the action on it.

When you click on the Design Report sub-menu, the system will show a list of system default report in a dialog. Choose the system default report you want to modify and click **Design** button to start the Report Designer.


Invoice - KO - AutoCount Accounting (Ver: 1.3.0.0)

Hint In this Invoice window, you can create, modify, or delete Invoice for your customer.

You can:

Create a new Invoice   Find Invoices   Print Invoice Listing

or you can show a list of Invoices, then find your Invoice and apply the action on it.

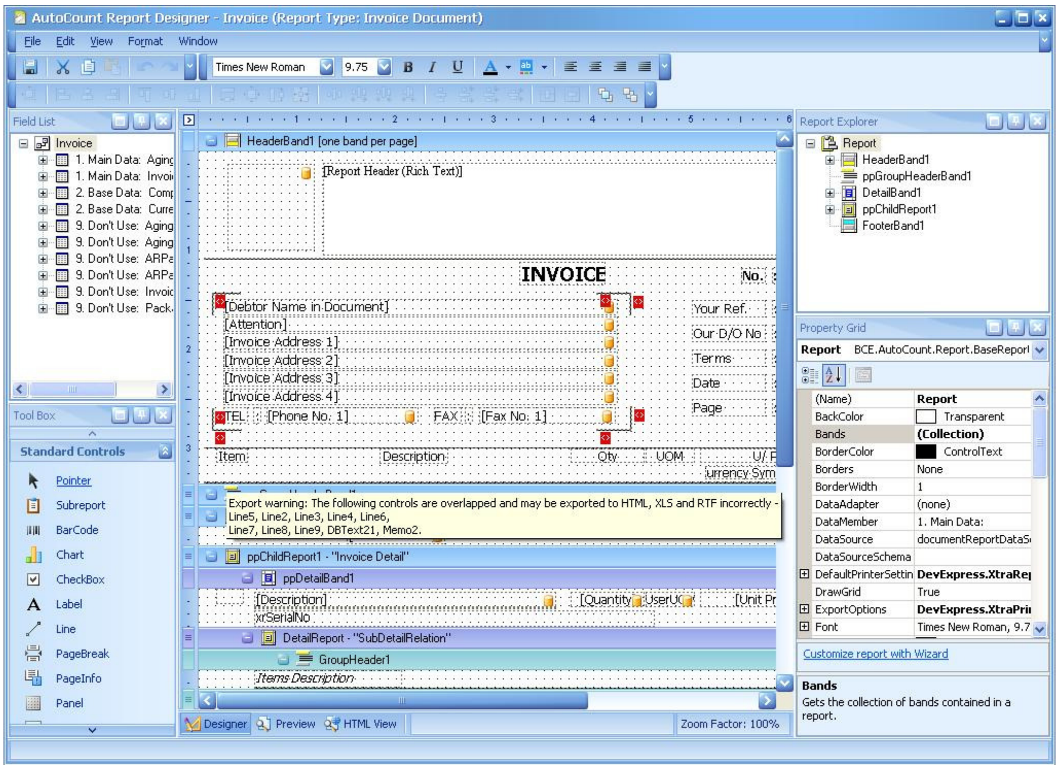
Design Report - Invoice Document

Report Name	Type
Invoice	System
Invoice - For Preprinted Form	System
Invoice - Half Page	System
Invoice With Aging	System
Tax Invoice	System
Tax Invoice - For Preprinted Form	System
Tax Invoice - Half Page	System
Tax Invoice with Aging	System

Note

System reports cannot be deleted nor overwritten.

The system will display a Report Designer page as shown below:-



Export warning: The following controls are overlapped and may be exported to HTML, XLS and RTF incorrectly - Line5, Line2, Line3, Line4, Line6, Line7, Line8, Line9, DBText21, Memo2.

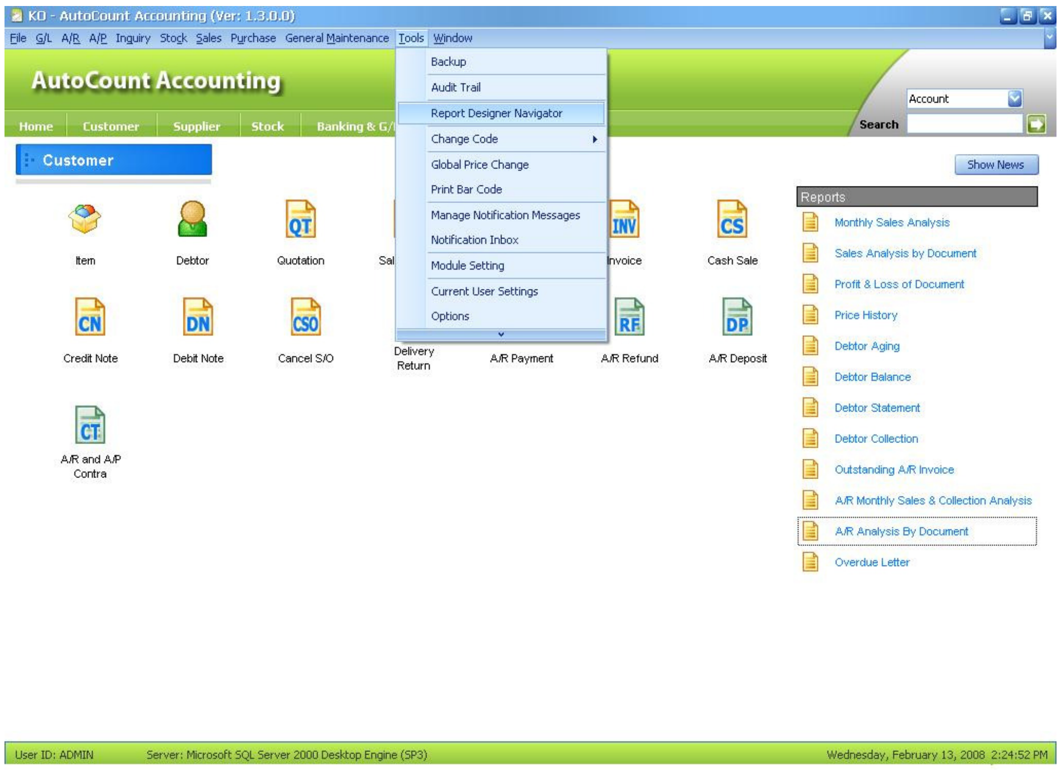
Report BCE.AutoCount.Report.BaseReport

(Name)	Report
BackColor	Transparent
Bands	(Collection)
BorderColor	ControlText
Borders	None
BorderWidth	1
DataAdapter	(none)
DataMember	1. Main Data:
DataSource	documentReportDataS
DataSourceSchema	
DefaultPrinterSettin	DevExpress.XtraRe
DrawGrid	True
ExportOptions	DevExpress.XtraPri
Font	Times New Roman, 9.7

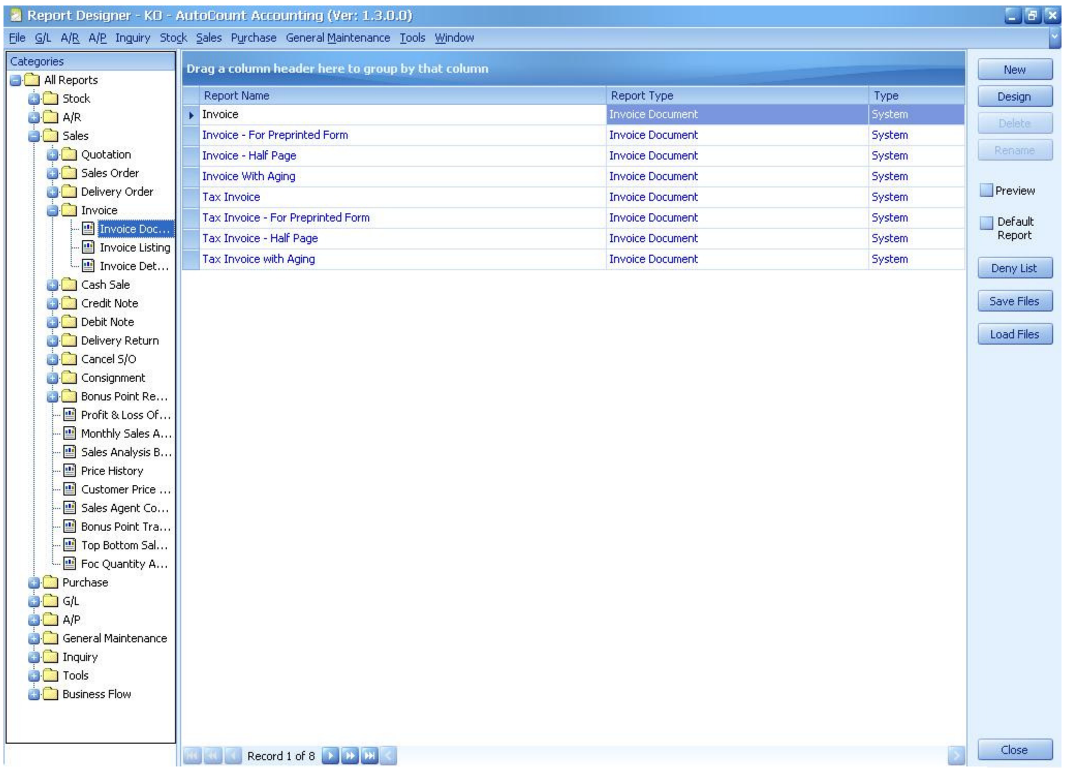
Bands
Gets the collection of bands contained in a report.

1.2. Start Report Designer from Tools menu

Alternatively, you can go to **Tools | Report Designer** navigator to customize your report.

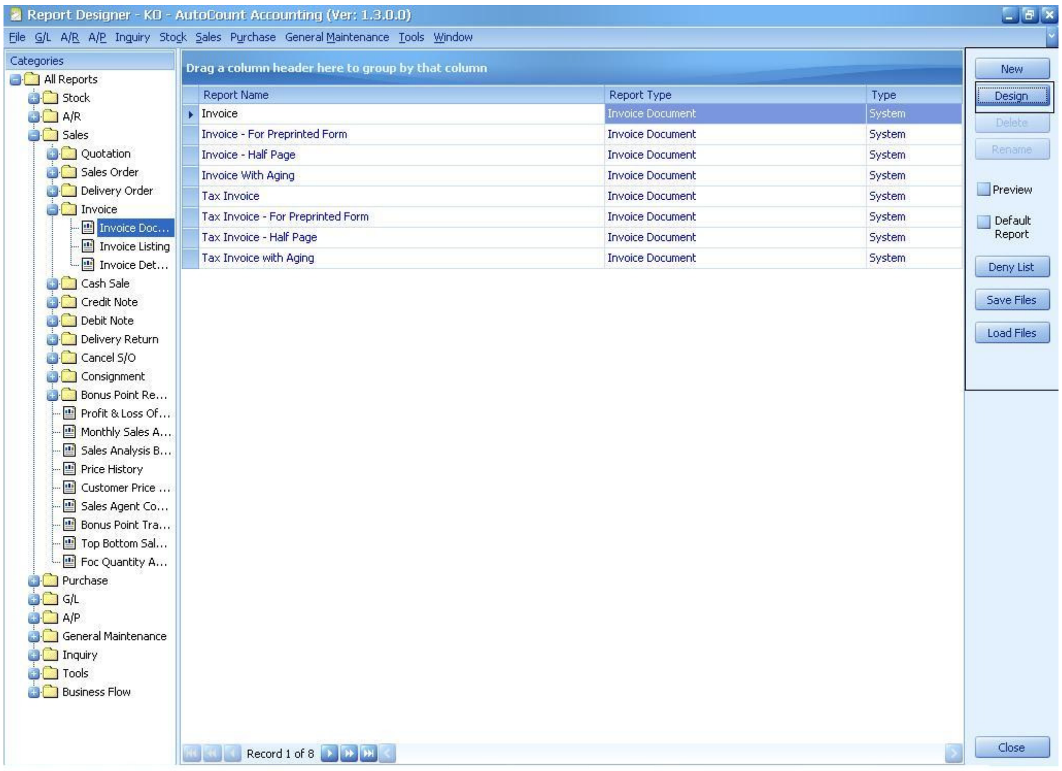


The Report Designer Navigator shows all reports of the system in a hierarchical structure, use the left pane **Categories** to navigate to your **report type**, then the main pane will show all reports belong to this report type.



Report Name	Report Type	Type
Invoice	Invoice Document	System
Invoice - For Preprinted Form	Invoice Document	System
Invoice - Half Page	Invoice Document	System
Invoice With Aging	Invoice Document	System
Tax Invoice	Invoice Document	System
Tax Invoice - For Preprinted Form	Invoice Document	System
Tax Invoice - Half Page	Invoice Document	System
Tax Invoice with Aging	Invoice Document	System

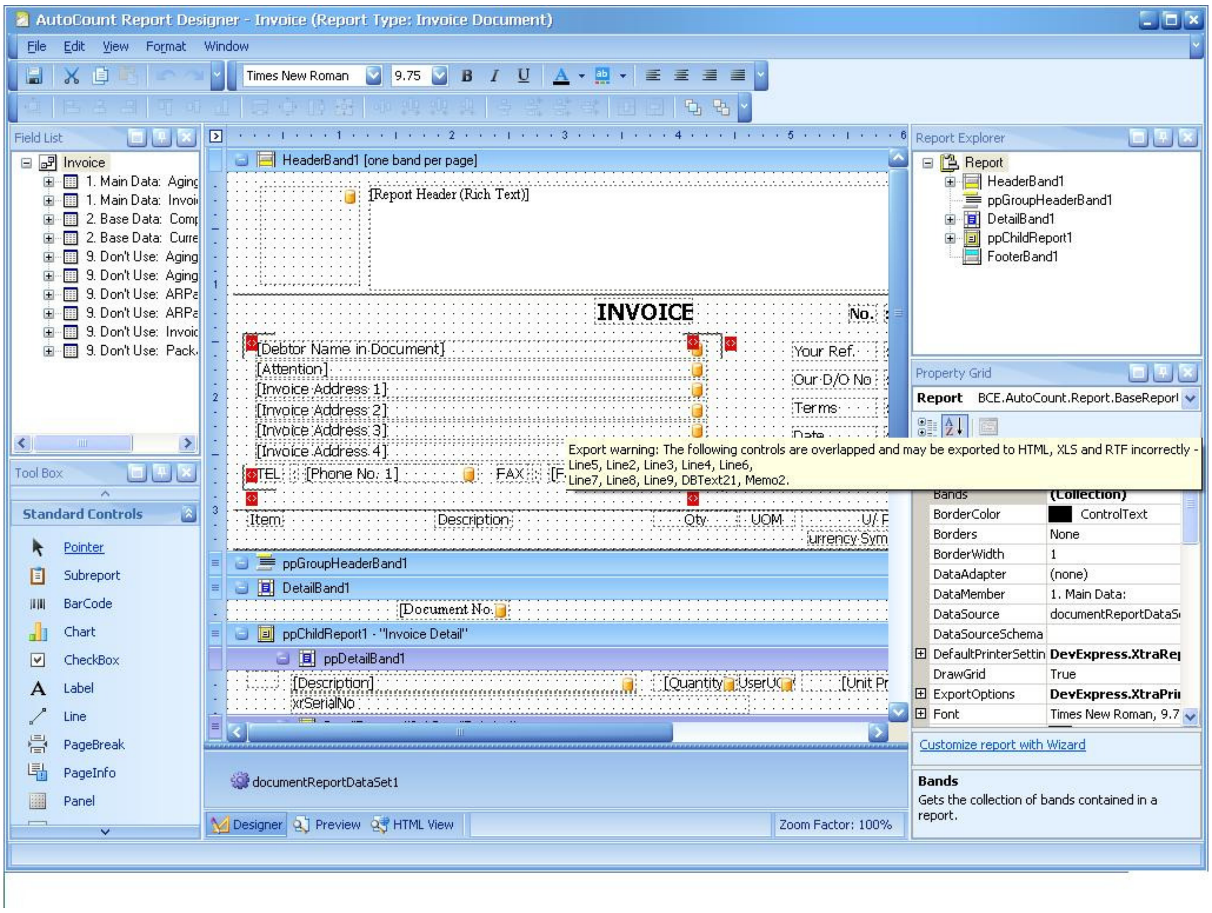
Then click on the **Design** button to start the Report Designer.



Report Name	Report Type	Type
Invoice	Invoice Document	System
Invoice - For Preprinted Form	Invoice Document	System
Invoice - Half Page	Invoice Document	System
Invoice With Aging	Invoice Document	System
Tax Invoice	Invoice Document	System
Tax Invoice - For Preprinted Form	Invoice Document	System
Tax Invoice - Half Page	Invoice Document	System
Tax Invoice with Aging	Invoice Document	System

1.3. Report Designer

When the Report Designer started, you will see the following window.



Export warning: The following controls are overlapped and may be exported to HTML, XLS and RTF incorrectly - Line5, Line2, Line3, Line4, Line6, Line7, Line8, Line9, DBText21, Memo2.

Bands	(Collection)
BorderColor	ControlText
Borders	None
BorderWidth	1
DataAdapter	(none)
DataMember	1. Main Data:
DataSource	documentReportDataS
DataSourceSchema	
DefaultPrinterSettin	DevExpress.XtraRe
DrawGrid	True
ExportOptions	DevExpress.XtraPri
Font	Times New Roman, 9.7

Bands
Gets the collection of bands contained in a report.

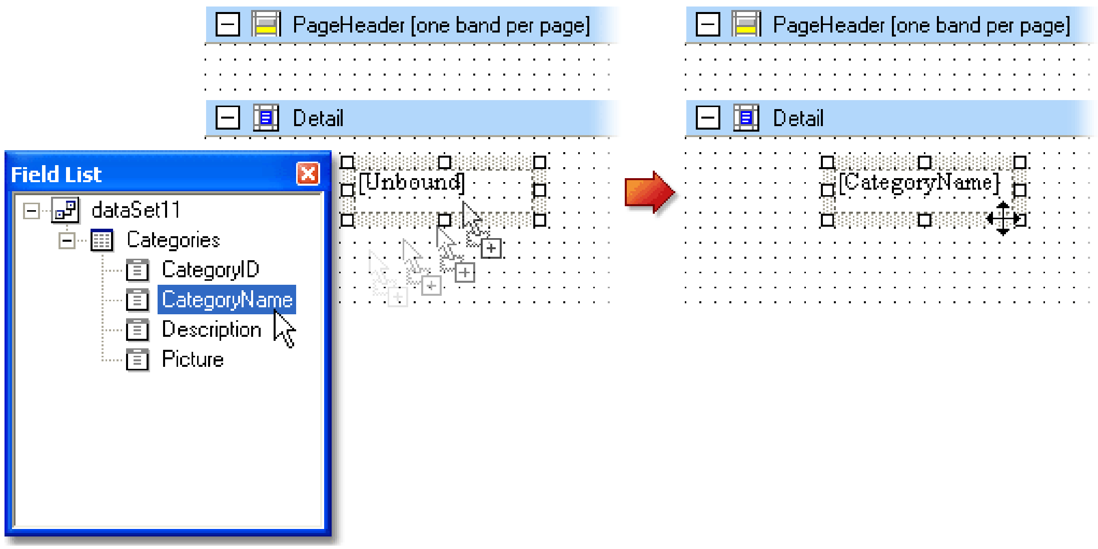
The standard **Report Designer** window basically can be represented by 7 major elements, which provide the basic editing capabilities of the report designer.

	Designer Elements	Short Description
1.	Field List	Represents the Field List tree in the report designer. This tree shows the structure of the data source which is bound to the report currently being edited in the End-User Designer, and is used to bind report controls to data.
2.	ToolBox	Represents the Toolbox in the report designer. This toolbox contains all the report controls and is used to drag and drop new controls onto the report's area.
3.	Formatting ToolBar	Represents the Formatting Toolbar in the report designer. It contains some default buttons for manipulating text.
4.	Panel	The main element in the End-User Designer which allows a report's layout to be edited. It contains two rulers, design and an **instant** preview tab, a status panel and the current report's surface, including its bands and controls.
5.	ToolBar	Represents the **Toolbar** in the report designer. This tool bar contains buttons which provide the ability to save, load, edit report layouts in the End-User Designer.
6.	Report Explorer	Represents the Report Explorer tree in the report designer. It shows a report's structure in a tree form and provides easy navigation through the report.
7.	PropertyGrid	Represents the **Properties Window** in the report designer. This window is used to change some of the properties of the report elements (bands and controls) by the end-User.

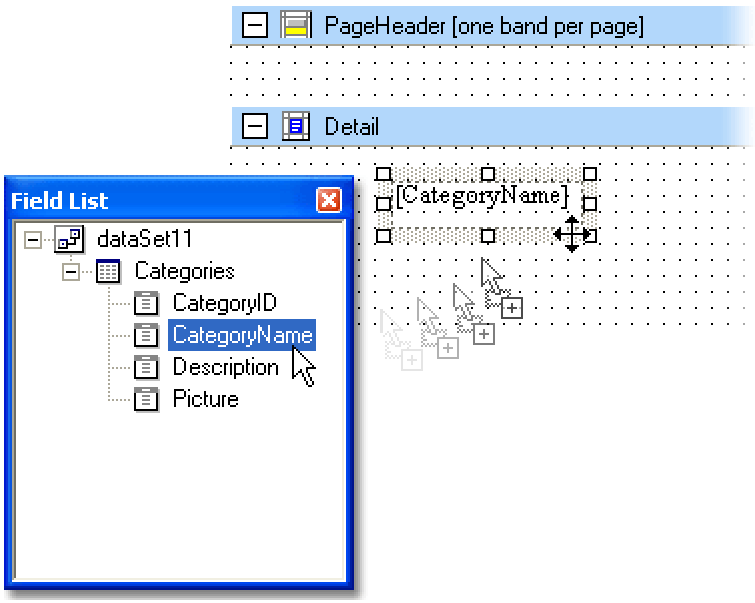
1.4. Field List

The **Field List** is intended to display the schema of the data source which is currently bound to a report. Also, this window may be used to bind existing report controls to data, or to create new bound report controls.

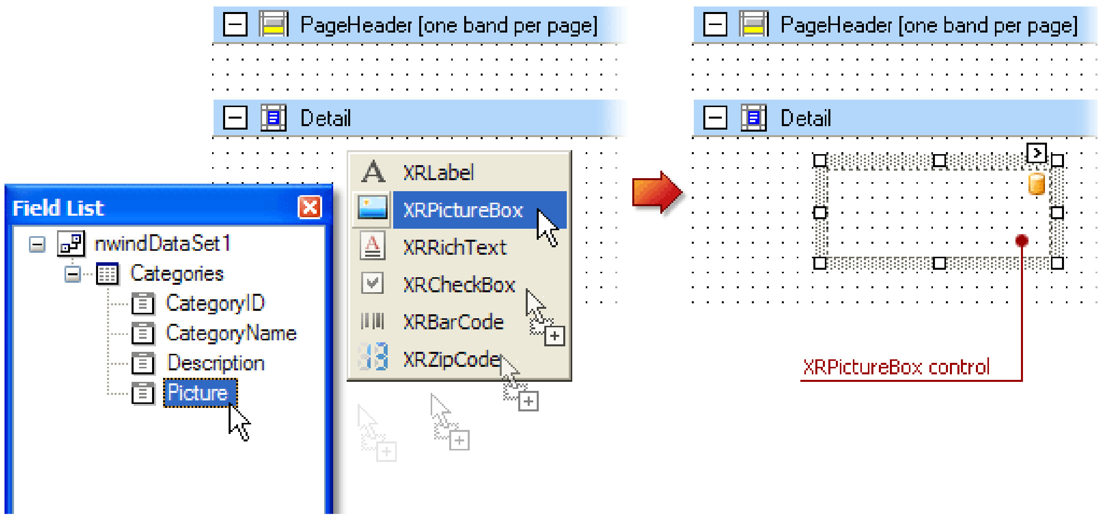
To bind an existing report control, control click the desired field item in the Field List window, and then drag and drop it onto the bindable report control. This control will then be bound to the selected data field.



To add a new bound report control, simply click the desired field item in the Field List window, and then drag and drop it onto the report band. Then, an XRLabel control bound to the selected data field will appear.



Another way of creating new bound report controls is to ***right-click*** a Field List item, and then drag and drop it onto a report. This will invoke the context menu shown in the image below. Simply choose the item you need from the list, and the selected control bound to the appropriate data field will be created and added to the report.

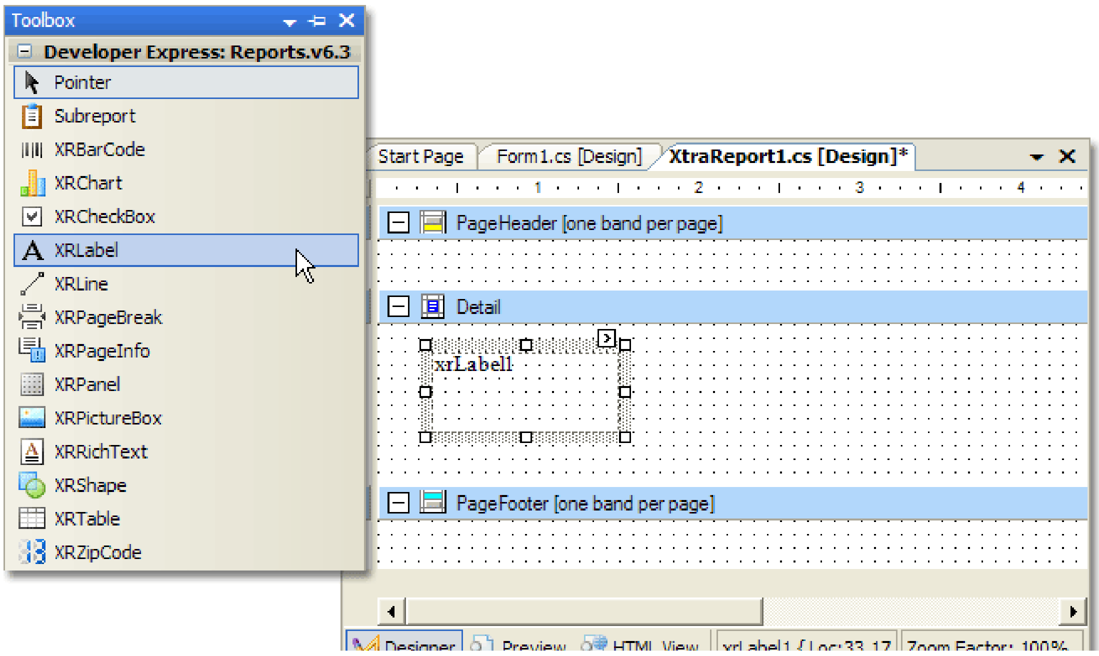

PageHeader [one band per page]
Detail
Field List
nwindDataSet1
Categories
CategoryID
CategoryName
Description
Picture
XRLabel
XRPictureBox
XRRichText
XRCheckBox
XRBarCode
XRZipCode
PageHeader [one band per page]
Detail
XRPictureBox control

1.5. Toolbox Tab

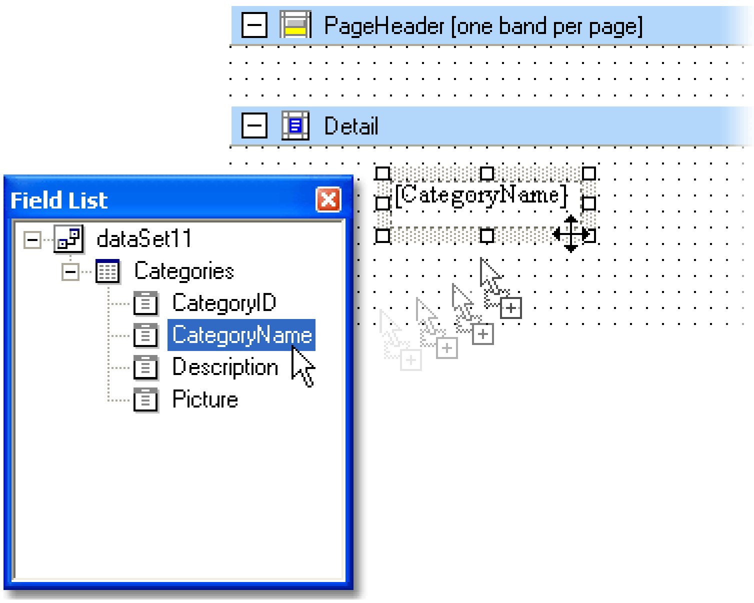
This is a toolbox tabs that contains all the standard report controls which can be inserted into a report. To use the toolbox tab, select the report control you need to insert into a report and then drop it onto one of the report bands.

Report Control Class	Description
BarCode	Represents a barcode control, which allows inserting many different barcode types into a report.
Chart	Represents a chart control, which may be used to represent your data as a series view.
CheckBox	Represents a check box control, which is intended to display a True/False or Checked/Unchecked/Indeterminate state in a report.
Label	Represents a label control. This is the basic control which allows inserting single-line or multi-line text into a report. Note that this text may be either static, or dynamically populated from a report's datasource.
Line	Represents a line control, which is intended to draw vertical, horizontal or diagonal lines in a report.
PageBreak	Represents a page break control. This control serves to mark the place where a report should start a new page. Note also that you may use the Band.PageBreak property instead of this control, if you want to break the page just before or after a particular band.
PageInfo	Represents a control which may be used to display some auxiliary information in a report. Use this control to display page numbers, the current date or user information in your report.
Panel	Represents a panel control which can contain other report controls. Use it to group controls together, to make their manipulation easier.
PictureBox	Represents a picture box control that can be used to display an image in a report. Use this control to insert images into your report.
RichText	Represents a rich text control which is intended to display, enter and manipulate formatted text. You can enter and format its text at design time, load it from an external file, or bind this control to a data field and populate it from a report's datasource.
Shape	Represents a shape control which can be used to embed any simple graphics into a report
Table	Represents a table control for inserting tables, containing rows and cells. This control is invaluable if you need to show your data in tabular form.
ZipCode	Represents a zipcode control, which allows the insertion of numbers representing a zipcode into a report.

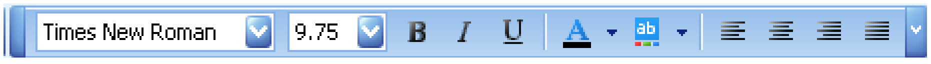
At design time report controls can be added to a report via the toolbox tab. Simply click the necessary report control in the toolbox tab, and then drag and drop it onto the report.



Another way of adding report controls to a report at design time is to use the Field List window. In this instance, an added report control is automatically bound to the specified data field.



1.6. Formatting ToolBar



It contains some default buttons for manipulating text. Example select font type, font size, edit text alignment.

1.7. Panel

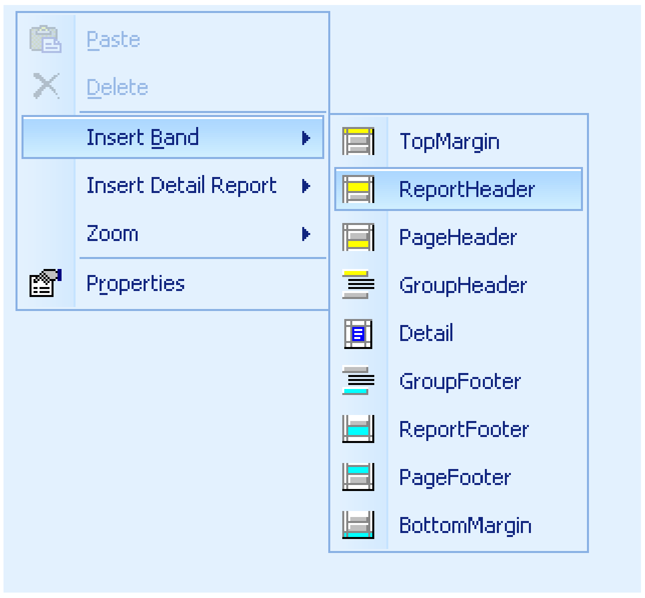
Panel contains two rulers, Design , HTML View and an **instant** Preview tab, a status panel and the current report's.

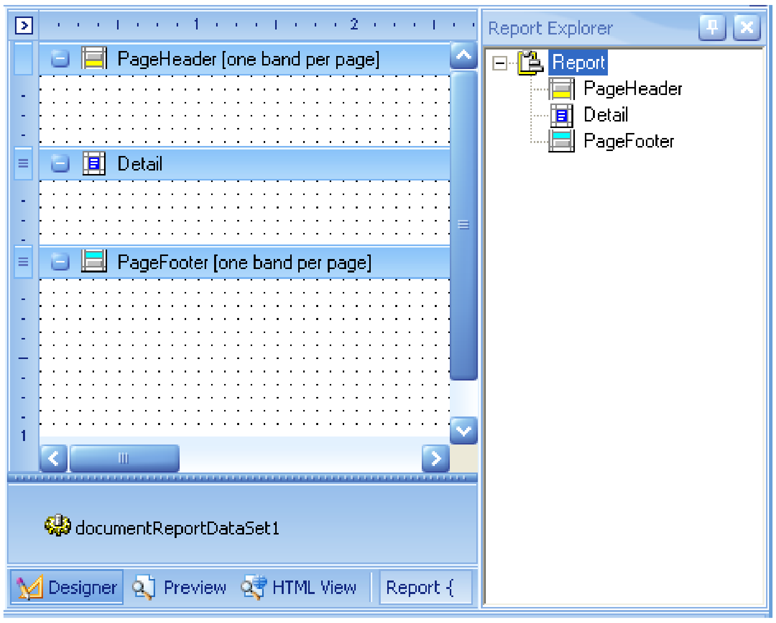
1.7.1. Report Band

Report Band represents a specific area on a report design page, which is used to define how to render report controls which belong to it. Every band is an instance of the Band class descendant. This class provides the Band.Height property which specifies the space that a band occupies on a page, along with other specific properties which define a band's behavior.

In the report designer, report bands are represented by the parts of the design surface divided via band strips.

After you've added a new blank report to a project, by default, it looks as shown in the image below. You can insert the band by right click at the report.





As you can see, the report's area is divided into three basic bands

- PageHeader band
- Detail band
- PageFooter band

that provide space for placing different report controls on them.

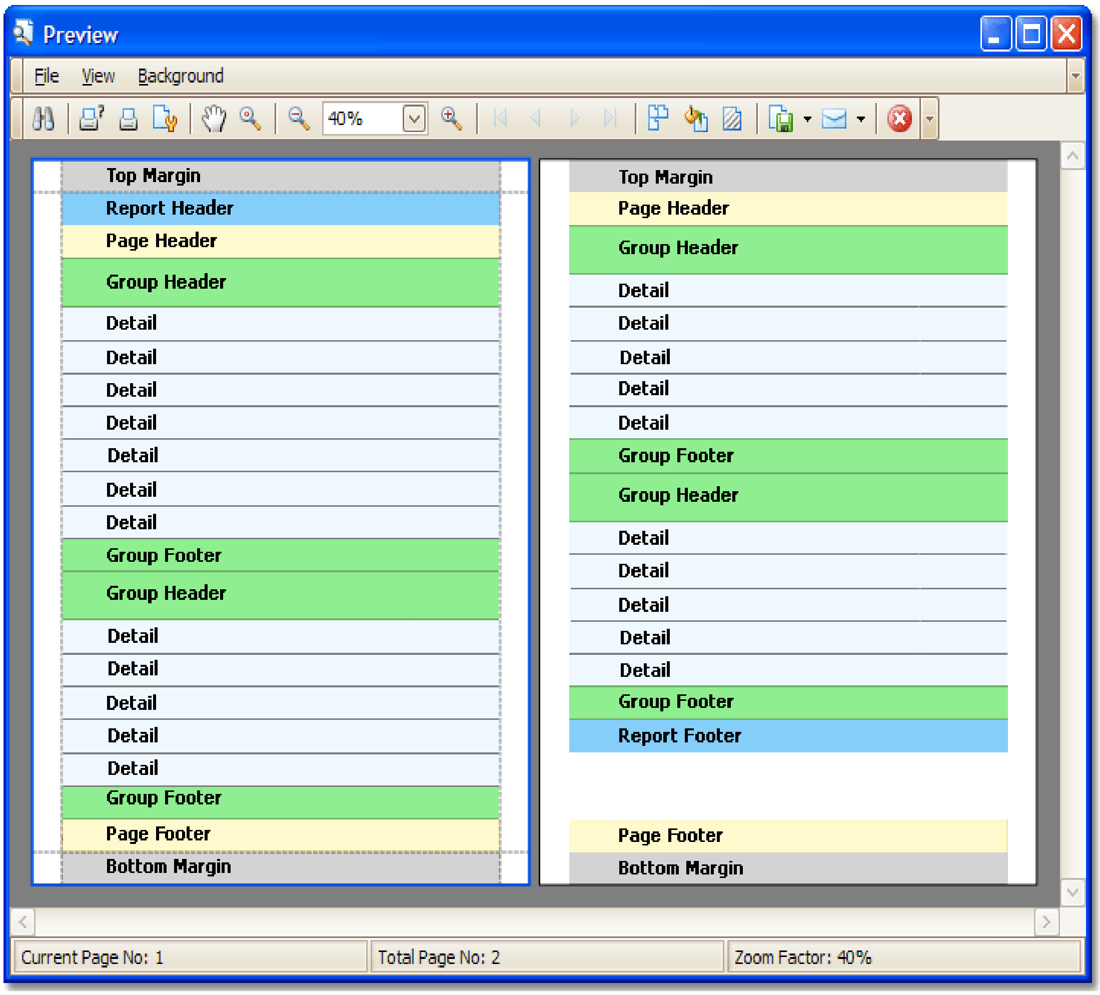
A particular band type specifies how the controls located on this band, are rendered, their rendering order and how many times they are rendered . Note that in the report designer, some of band strips may display tips with information on how bands will be rendered. For instance, for the **PageHeader** and **PageFooter** bands, the "one band per page" tip is displayed.

Later on, when creating a particular report, you can add or remove these or any other bands. There are different band types available in **XtraReports**, and each individual band is a descendant of the Band class. The table below lists them.

Report Band Class	**Description**
TopMarginBand	Represents a band located on the top margin of every page.
ReportHeaderBand	Represents a band used as a report header. Objects placed in this band are rendered only once - at the beginning of a report.
PageHeaderBand	Represents a band located at the beginning of every report page, below the TopMarginBand. It is mainly intended to display the header of a table, continued from the previous page.
GroupHeaderBand	Represents a band used to specify grouping criteria and to display information at the beginning of a group of records shown in the DetailBand. For more information about using this band type, see the Data Grouping document.
DetailBand	Represents a band used to display a single record from the bound datasource at a time, or just to hold controls (if there is no bound datasource). For more information about data binding, see the Providing Data document.

DetailReportBand	Represents a band used to create a master-detail report (the **DetailReport** band holds the detail report). The master-detail relationship for this detail report is specified by the XtraReportBase.DataMember property. To learn more about detail reports, see the Master-Detail Report Using Detail Report Bands document.
GroupFooterBand	Represents a band used to display controls at the end of a group of records shown in the DetailBand.
ReportFooterBand	Represents a band used as a report footer. Objects placed in this band are rendered only once, - at the end of a report.
PageFooterBand	Represents a band located at the bottom of every report page, above the **BottomMarginBand**. It is mainly intended to display the footer of a table, which has been continued on the following page.
BottomMarginBand	Represents a band located on the bottom margin of every page.

The following image illustrates the relative positions of different band types, and how many times they are rendered in a report.



The **PageHeaderBand, PageFooterBand, TopMarginBand** and **BottomMarginBand** bands are rendered in the report preview on every page.

The **ReportHeaderBand** and **ReportFooterBand** bands are rendered in the report preview only once.

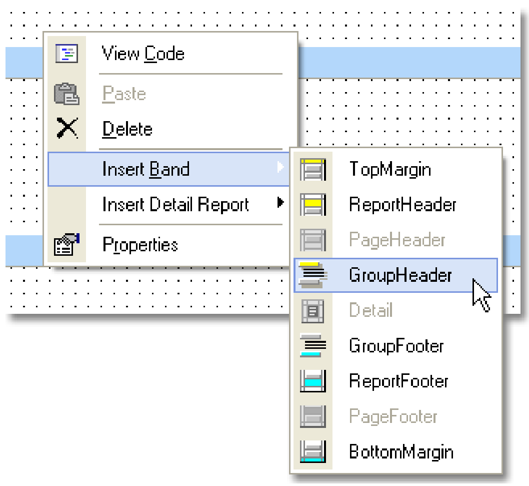
The **GroupHeaderBand** and **GroupFooterBand** bands are rendered for every group of records in a report.

The number of times the **DetailBand** band is rendered in a report depends upon the number of records returned from the bound datasource, - one band per record.

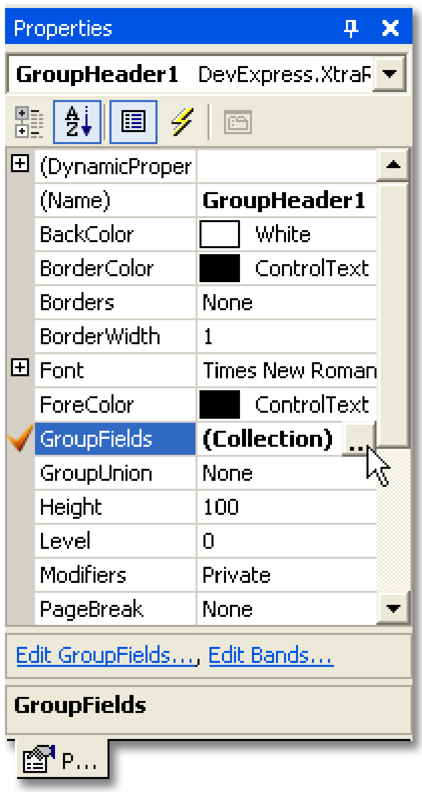
1.7.2. Data Grouping

Data grouping can be done by one or more data fields, which are called group fields and represented by GroupField objects. When you group data by a single or a few data fields, records with identical values in these group fields are arranged into corresponding data groups.

By right-clicking the report area in the report designer invokes the context menu. Then, add a **GroupHeader** band to the report as is shown in the image below.



Click the **GroupFields** property item of the created **GroupHeader1** band to invoke the **GroupField Collection** editor.



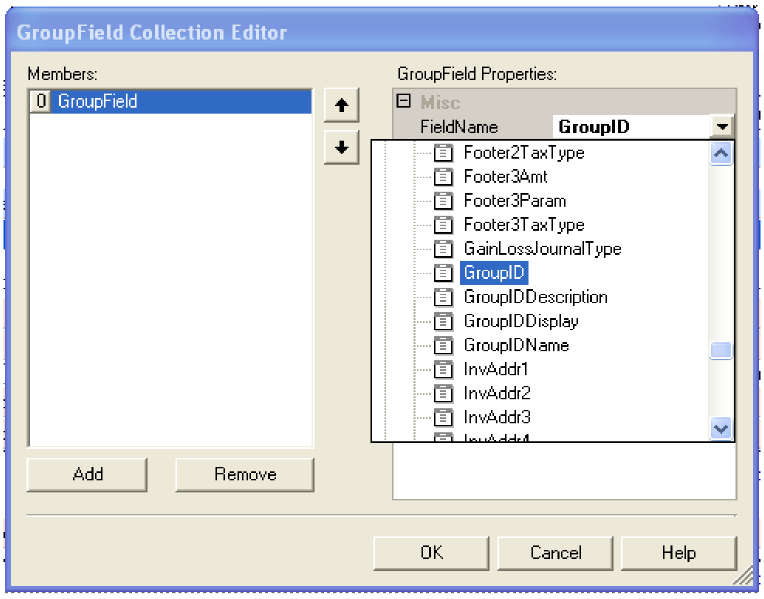
GroupHeader1 DevExpress.XtraR

(DynamicProper	
(Name)	GroupHeader1
BackColor	White
BorderColor	ControlText
Borders	None
BorderWidth	1
Font	Times New Roman
ForeColor	ControlText
GroupFields	(Collection)
GroupUnion	None
Height	100
Level	0
Modifiers	Private
PageBreak	None

Edit GroupFields..., Edit Bands...

GroupFields

As you can see there is already a GroupID inside the fieldname, this mean that the grouping is based on the GroupID determined by the user on screen. Click the OK button to close the editor. User also can group other data fieldname beside GroupID.



Another usage is sorting which is the similar way as grouping; just simply select the SortID instead of GroupID.

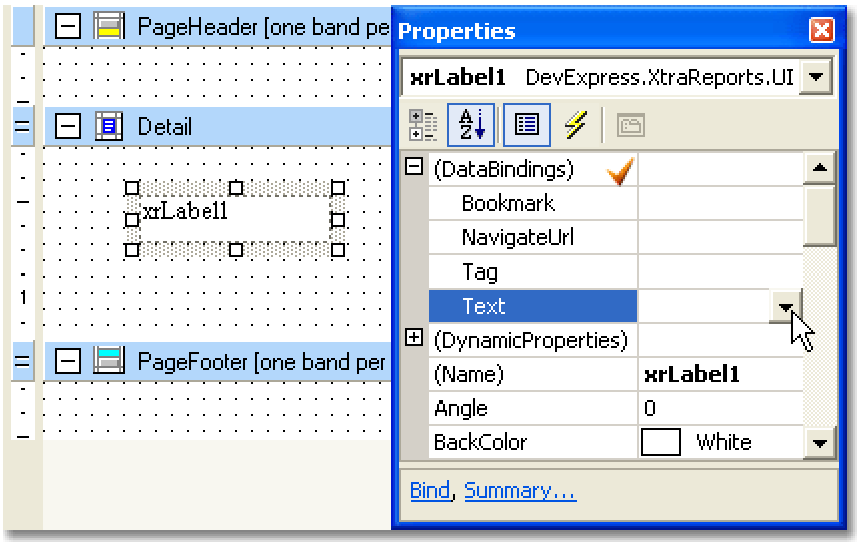
1.7.3. Report Controls to Data

This topic describes how to bind report controls to data fields, introduces the differences between binding controls situated on different report bands, and provides design-time and runtime samples. Note that report controls can be bound to any existing datasource in a report, regardless of whether the report is bound to the data or not.

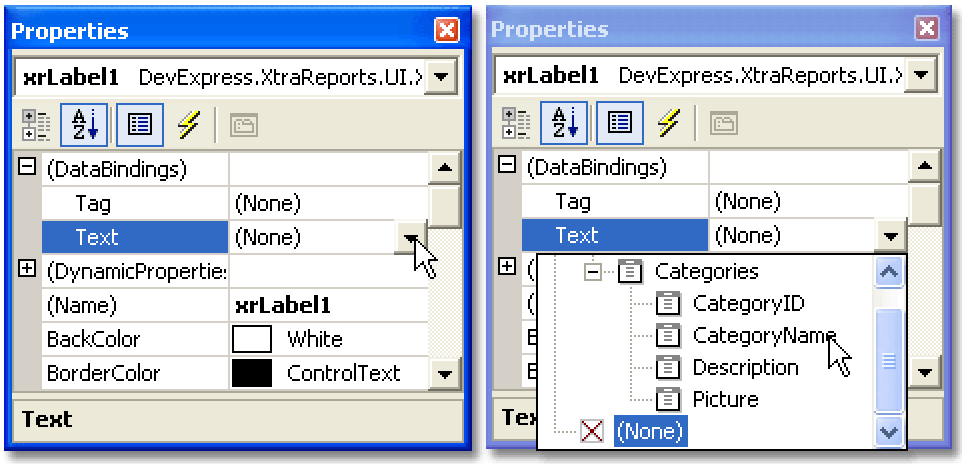
1.7.3.1. Methods for Binding Controls

Report controls can be bound to data from any datasource that exists in the report. If the report is also bound to the same, then this datasource will be the **primary datasource** for bound controls. It's also possible to bind a report control to a data field in a *secondary datasource*, but in this case you should manually advance the record cursor in the datasource. Otherwise it will always display the current record. Note, that if the XtraReportBase.DataAdapter property is not specified for the **primary datasource**, or if a *secondary datasource* is used, you are responsible for populating it with data and advancing its record cursor.

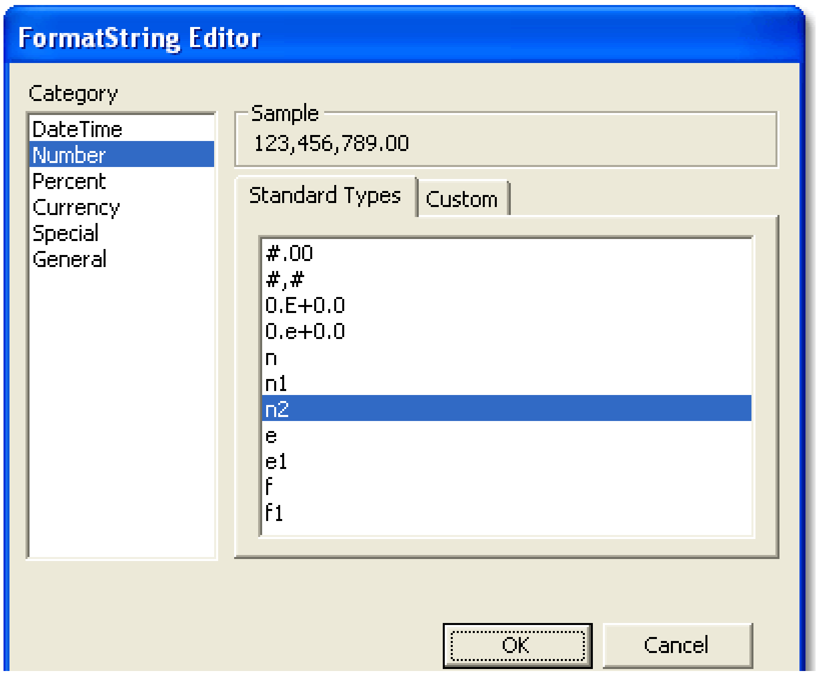
A control can be bound to data by setting the control's bindable properties to the fields in a table. Each report control has different bindable properties. Each binding is represented by an XRBinding object, and a collection of all the bindings for a control can be accessed via the XRControl.DataBindings property. At design time all bindable properties are also shown in the **(DataBindings)** group at the top of the Properties window. The bindable properties of an XRLabel control are shown in the picture below.



You can bind a **report control** to data both at design and runtime. Expand the **(DataBindings)** group in the Properties window. Then click the property item you want to bind. This will invoke the same binding editor as the one described above. The image below illustrates this process.



Here you can also specify a format for output values. For this purpose simply click the ellipsis button in the XRBinding.FormatString property item.


FormatString Editor

Category: DateTime, Number (selected), Percent, Currency, Special, General

Sample: 123,456,789.00

Standard Types | Custom

```
#.00
#,#
0.E+0.0
0.e+0.0
n
n1
n2
e
e1
f
f1
```

Selected: n2

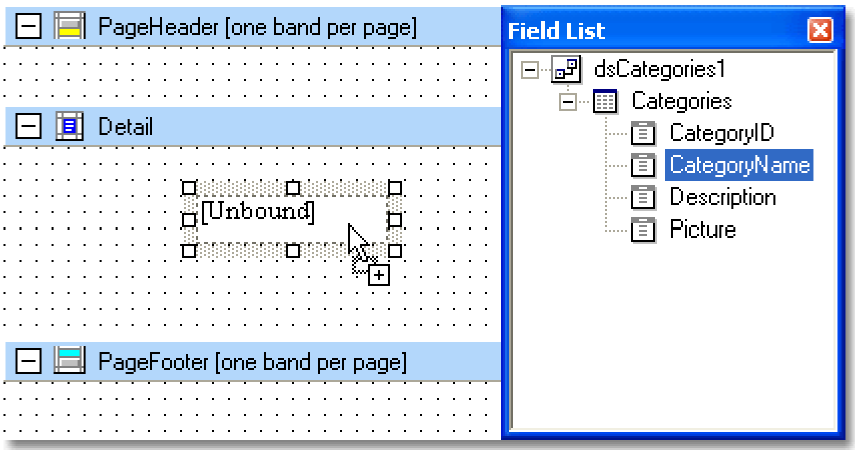
OK | Cancel

Note

The Format String Editor can also be invoked via the Format String context link.

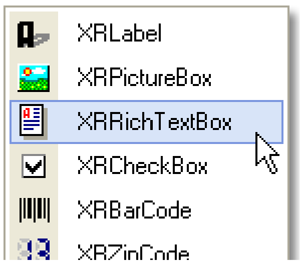
It is also very easy to bind an existing report control or to add a new bound control using the Field List window. Simply click the desired field item in the **Field List** window, then drag and drop it onto a bindable report control. This control will then be bound to the selected data field. Note that in this case the bound property will be the property specified in the control's **DefaultBindableProperty** attribute.

Also, if a field item is dropped onto any band's area in a report a new bound XRLabel object will appear at the drop point.

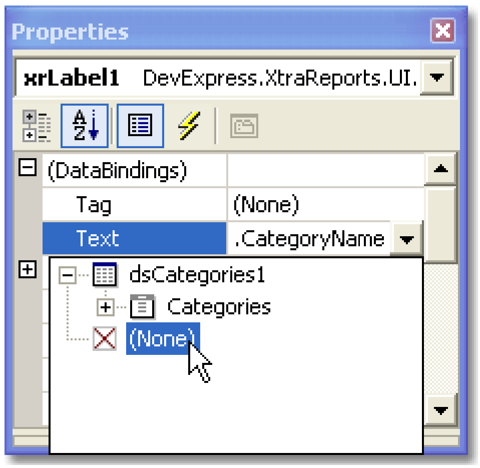


Note

You can also create a bound control of a specific type by dragging and dropping an item from the **Field List** window using the right mouse button. In this case after the item is dropped a context menu containing a list of the available report controls that can be created for the item will be displayed.



Controls can be unbound at design time. This can be done using the same binding editor by selecting the **(None)** item.

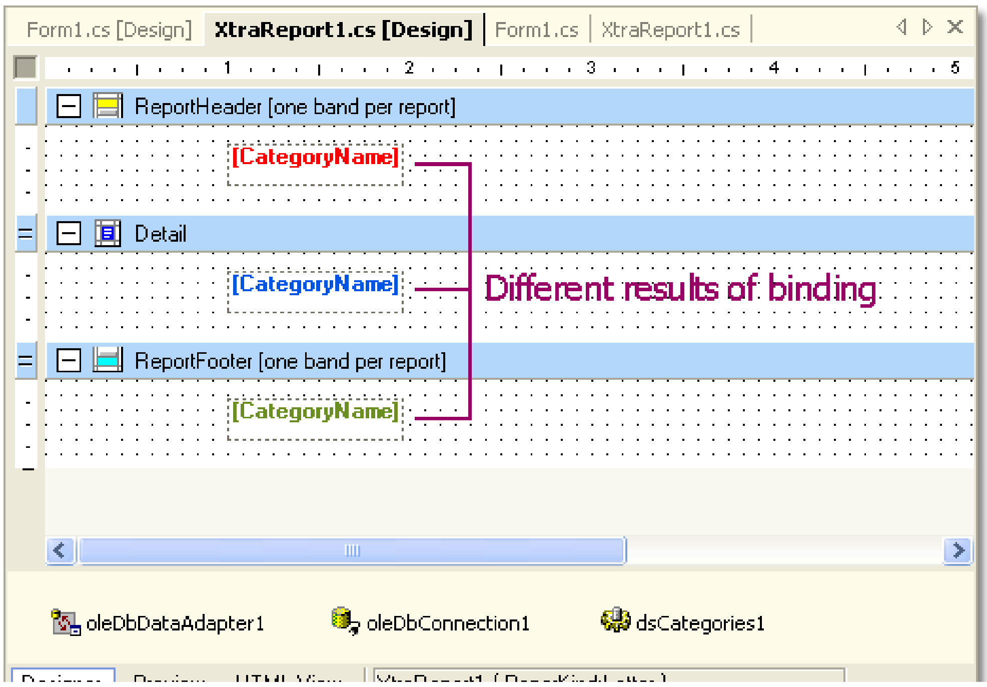


1.7.3.2. Binding Controls on Different Bands

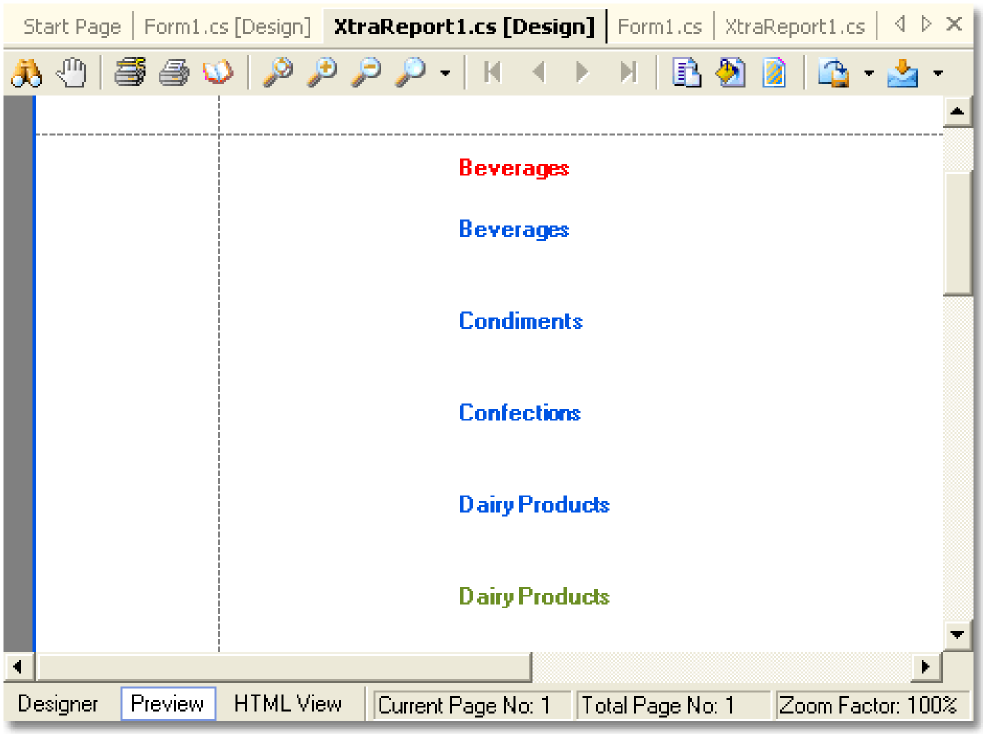
When binding a **report control** to a data field the data returned to the control depends on which band the bound control is located in. So, if a control is situated in the **Detail** band, every single record in the datasource will be returned and displayed in the control. Bound controls in grouping bands show data by dividing all the data into groups. Lastly, bound controls in any other bands display data from the record which is current when the band they are in is being printed/displayed. For more information about report bands see the Report Bands topic.

This feature is illustrated in the images below.

Report Design:



Report preview:



1.7.4. Smart Tag

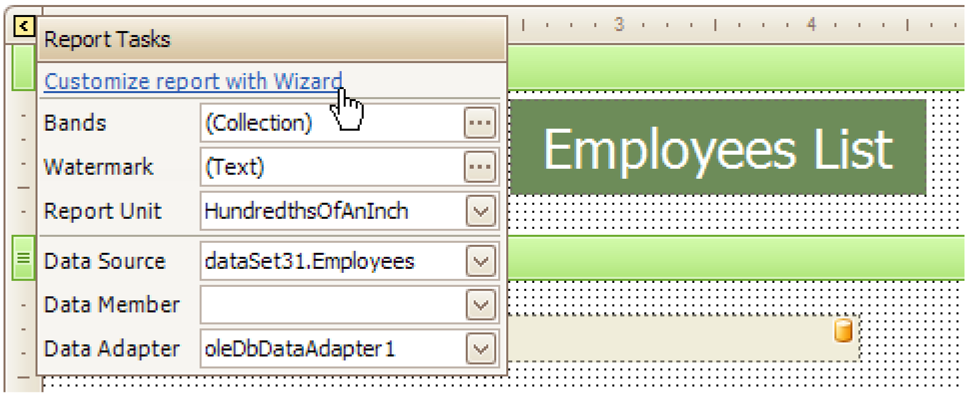
The smart-tag feature enables report controls and bands to display context-sensitive information and commands. The smart tag can be thought of as a replacement for designer verbs, because you can choose to display a smart tag item in both the smart-tag panel, and in the shortcut menu associated with a report control or a band.

To invoke a smart tag you first need to select any report element, and then to click the smart tag icon (- looks like a right arrow) of the currently selected report element. Then the smart tag panel is invoked on the left side of the smart tag icon, thus allowing you or an end-user to quickly adjust the selected report element.

The smart-tag feature is available for the following report elements.

XtraReport

A report's smart tag icon is located at the top left corner of a report designer.



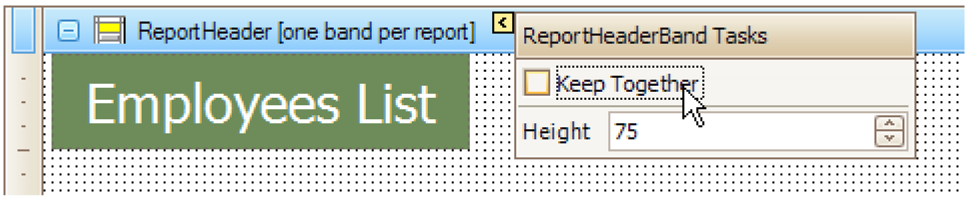
Report Tasks

Customize report with Wizard

Bands	(Collection)
Watermark	(Text)
Report Unit	HundredthsOfAnInch
Data Source	dataSet31.Employees
Data Member	
Data Adapter	oleDbDataAdapter1

Report Bands

A band's smart tag icon is located on the band strip right next to the caption. For instance, the smart tag for the Report Header band is shown in the image below.

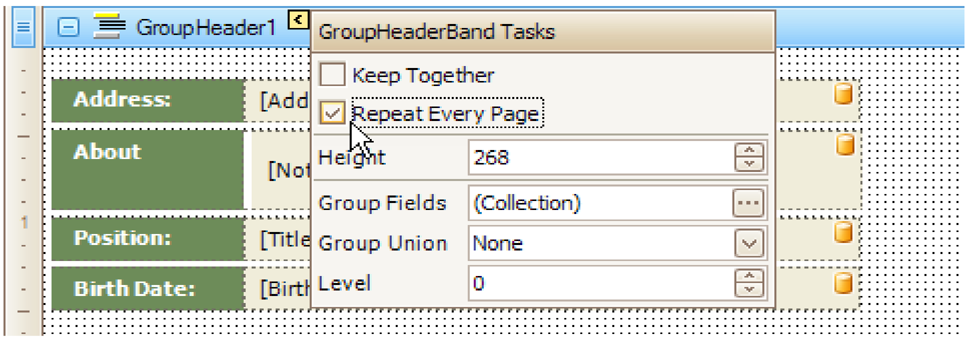


ReportHeaderBand Tasks

- [ ] Keep Together

Height	75

The following is the smart tag for the Group Header band.

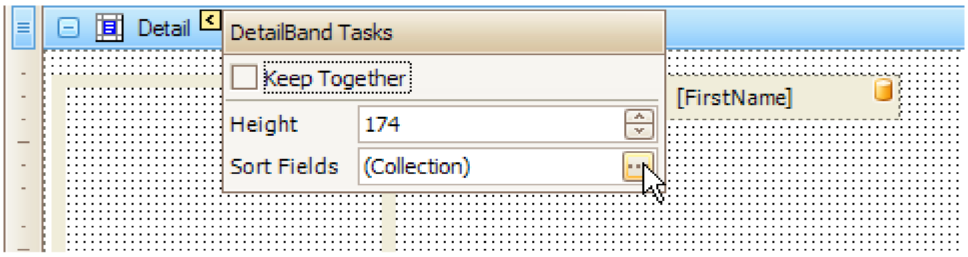


GroupHeaderBand Tasks

- [ ] Keep Together
- [✓] Repeat Every Page

Height	268
Group Fields	(Collection)
Group Union	None
Level	0

The following is the smart tag for the Detail band.


Detail

DetailBand Tasks

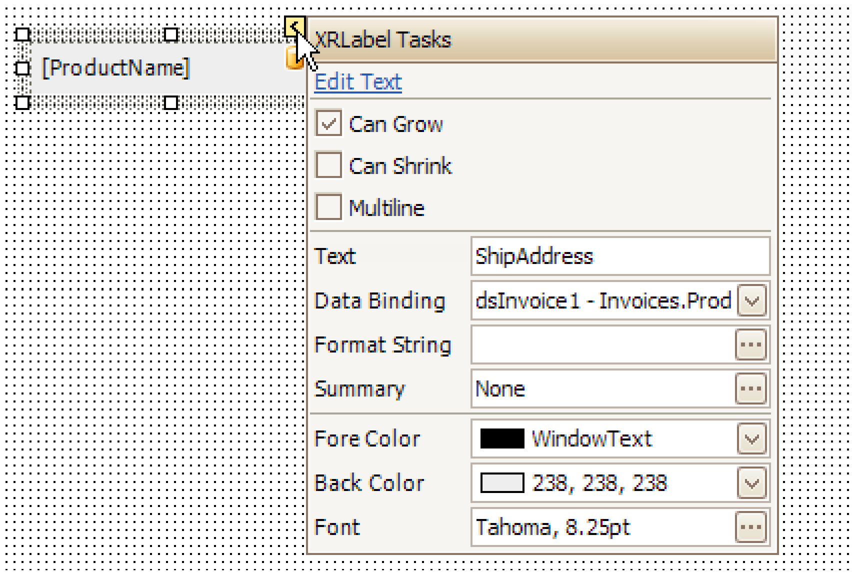
Keep Together

Height	174
Sort Fields	(Collection)

[FirstName]

Report Controls

A control's smart tag icon is located at the top right corner of the control. For instance, the smart tag for the label is shown in the image below.


[ProductName]

XRLabel Tasks

Edit Text

☑ Can Grow

☐ Can Shrink

☐ Multiline

Text	ShipAddress
Data Binding	dsInvoice1 - Invoices.Prod
Format String	
Summary	None
Fore Color	■ WindowText
Back Color	□ 238, 238, 238
Font	Tahoma, 8.25pt

The following is the smart tag for the picture box.

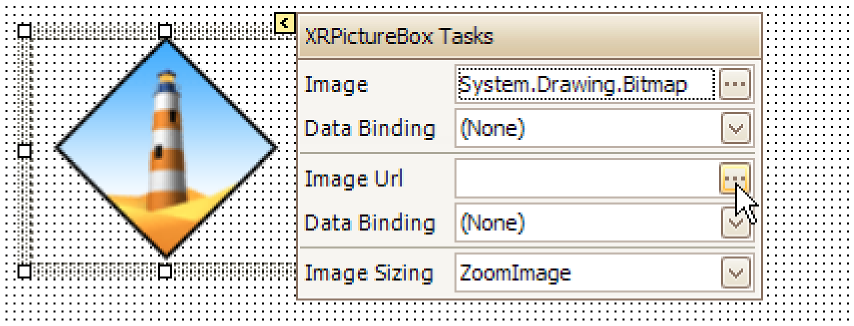
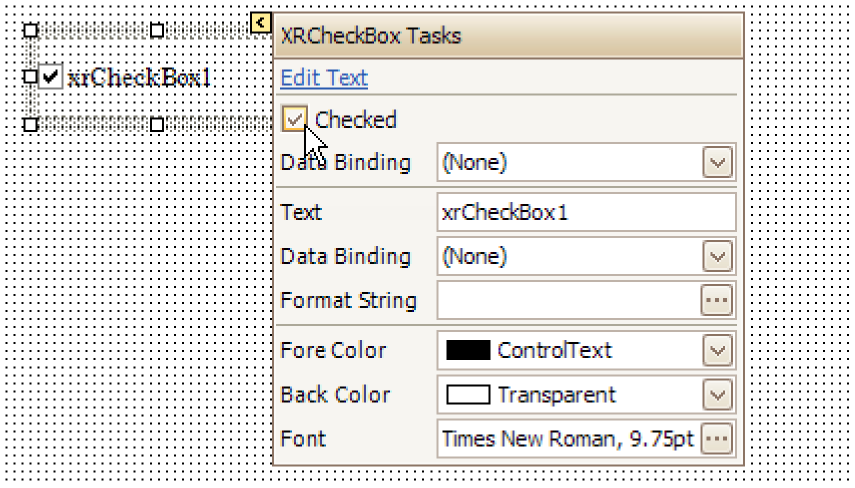

XRPictureBox Tasks

Image	System.Drawing.Bitmap
Data Binding	(None)
Image Url	
Data Binding	(None)
Image Sizing	ZoomImage

The following is the smart tag for the check box.



Checked

Data Binding (None)

Text xrCheckBox1

Data Binding (None)

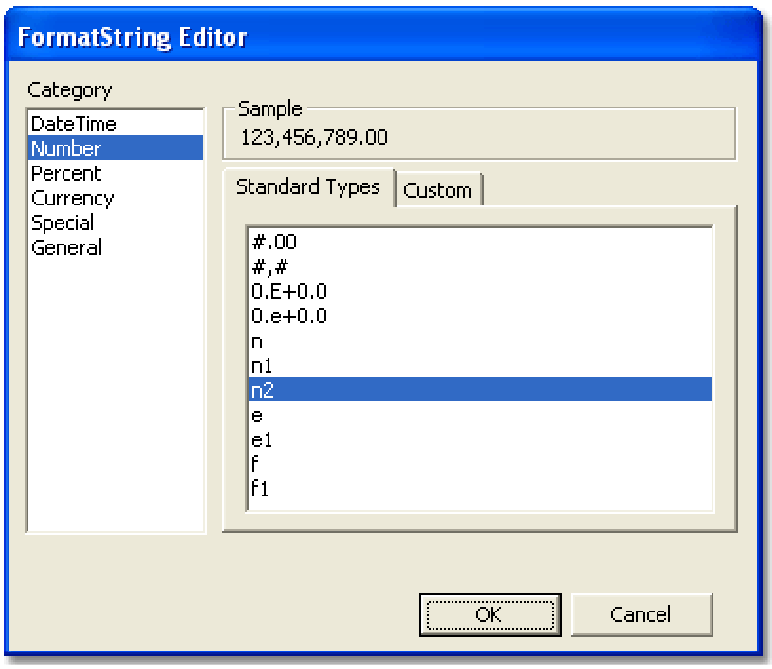
Format String

Fore Color ControlText

Back Color Transparent

Font Times New Roman, 9.75pt

1.7.5. Format String



Standard Format Strings for Numeric Values

Standard format strings for numeric values are specified in the *Axx* format. Here *A* is a character called the format specifier. *xx* is a sequence of digits called the precision specifier. The format specifier denotes whether values should be transformed to currency format, scientific notation, etc. This specifier must be set to one of the predefined characters listed in the **Standard Numeric Format Strings**. The table below gives some commonly used values.

Format Specifier	Description	Sample Format String	Sample Output
c or C	The number is converted to a string that represents a currency amount. The precision specifier indicates the desired number of decimal places. If the precision specifier is omitted, the default currency precision from the	c2	$1,234.00

	current regional options is used.		
e or E	The number is converted to a string of the form "-d.ddd...E+ddd" or "-d.ddd...e+ddd", where each 'd' indicates a digit (0-9). The string starts with a minus sign if the number is negative. One digit always precedes the decimal point. The precision specifier indicates the desired number of digits after the decimal point. If the precision specifier is omitted, a default of six digits after the decimal point is used. The case of the format specifier indicates whether to prefix the exponent with an 'E' or an 'e'. The exponent always consists of a plus or minus sign and a minimum of three digits. The exponent is padded with zeros to meet this minimum, if required.	E1	1.2E+003
n or N	The number is converted to a string of the form "-d,ddd,ddd.ddd...", where each 'd' indicates a digit (0-9). The string starts with a minus sign if the number is negative. Thousand separators are inserted between each group of three digits to the left of the decimal point. The precision specifier indicates the desired number of decimal places. If the precision specifier is omitted, the default currency precision from the current regional options is used.	n0	1,234
x or X	The number is converted to a string of hexadecimal digits. The case of the format specifier indicates whether uppercase or lowercase characters are used for hexadecimal digits greater than 9. The precision specifier indicates the minimum number of digits in the resulting string. If required, the number is padded with zeros to its left to produce the number of digits given by the precision specifier. This format is supported for integral types only.	X8	000004D2

Standard Format Strings for Date/Time Values

Standard date and time format strings contain a single character. This character defines the pattern used to represent the value (whether and how to display year numbers, month numbers, etc). The table below lists the most commonly used format characters.

Format Specifier	Description	Sample Output
d	Short date pattern.	3/12/2003
D	Long date pattern.	Wednesday, March 12, 2003
t	Short time pattern.	12:00 AM
T	Long time pattern.	12:00:00 AM
f	Full date/time pattern (short time).	Wednesday, March 12, 2003 12:00 AM
F	Full date/time pattern (full time).	Wednesday, March 12, 2003 12:00:00 AM
g	General date/time pattern (short	3/12/2003 12:00 AM

	time).	
G	General date/time pattern (full time).	3/12/2003 12:00:00 AM

Custom Format Strings for Numeric Values

Custom format strings are used to construct format patterns manually. You only need to use them when the standard format strings do not meet your requirements. All format strings represented by a literal character followed by one or two digits are treated as standard format strings and so all other strings are interpreted as custom format strings. The table below lists the most commonly used characters that can construct a custom format string.

Character	Meaning
0	The digit is always displayed.
#	The digit is displayed only when needed (i.e. use to suppress leading zeros).
.	Specifies the position of the decimal point. The appearance of the point depends upon the regional settings.
,	Specifies the position of a comma. The appearance of the comma depends upon the regional settings.

Note that custom format strings can also contain other characters and they will be copied to the formatted string. This can be used to add explanatory text to the value. If you need to display one of the reserved characters, it must be preceded by the '\' symbol.

When formatting numeric values, you can apply different formats to positive, negative and zero values. To do this, the format string must contain three parts delimited by semicolons. The first part sets the positive values format, the second is applied to negative values and the third represents zero values. For example, the custom format strings to display blank when the numeric value is 0 is coded as "**#,0.00;-#,0.00;** ". Another example, the custom format strings to display negative when the numeric value is positive, and to display positive when the numeric value is negative is coded as "**-#,0.00;#,0.00;**".

Custom Format Strings for Date/Time Values

To create format patterns for date and time values, you need to combine the strings listed in the tables below. These strings represent the year, month, day number and so on in different formats.

The following table lists the most commonly used strings that can be used to format dates. (Samples assume that the formatted date is 9/2/2003).

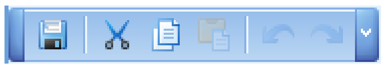
Symbol	Meaning	Value
yy	The last two digits of the year.	03
yyyy	Four digit year.	2003
MM	The number of the month.	09
MMM	The short text description of the month.	Sep
MMMM	The full name of the month.	September
dd	The number of the day.	02
ddd	The short text for the day of the week.	Tue

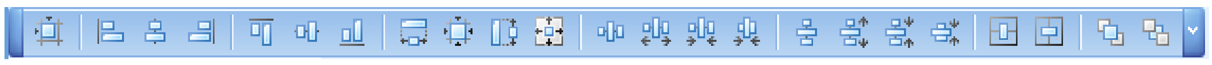
dddd	The full name of the day of the week.	Tuesday
/	Date separator. Its appearance depends upon the regional settings.	

The next table lists strings that are used to format time values.

Symbol	Meaning
hh	Hours.
mm	Minutes.
ss	Seconds.
tt	If present, represents data in AM/PM format.
:	Time separator. Its appearance depends upon the regional settings.

1.8. ToolBar





This tool bar contains buttons which provide the ability to save, load, edit report layouts in the End-User Designer.

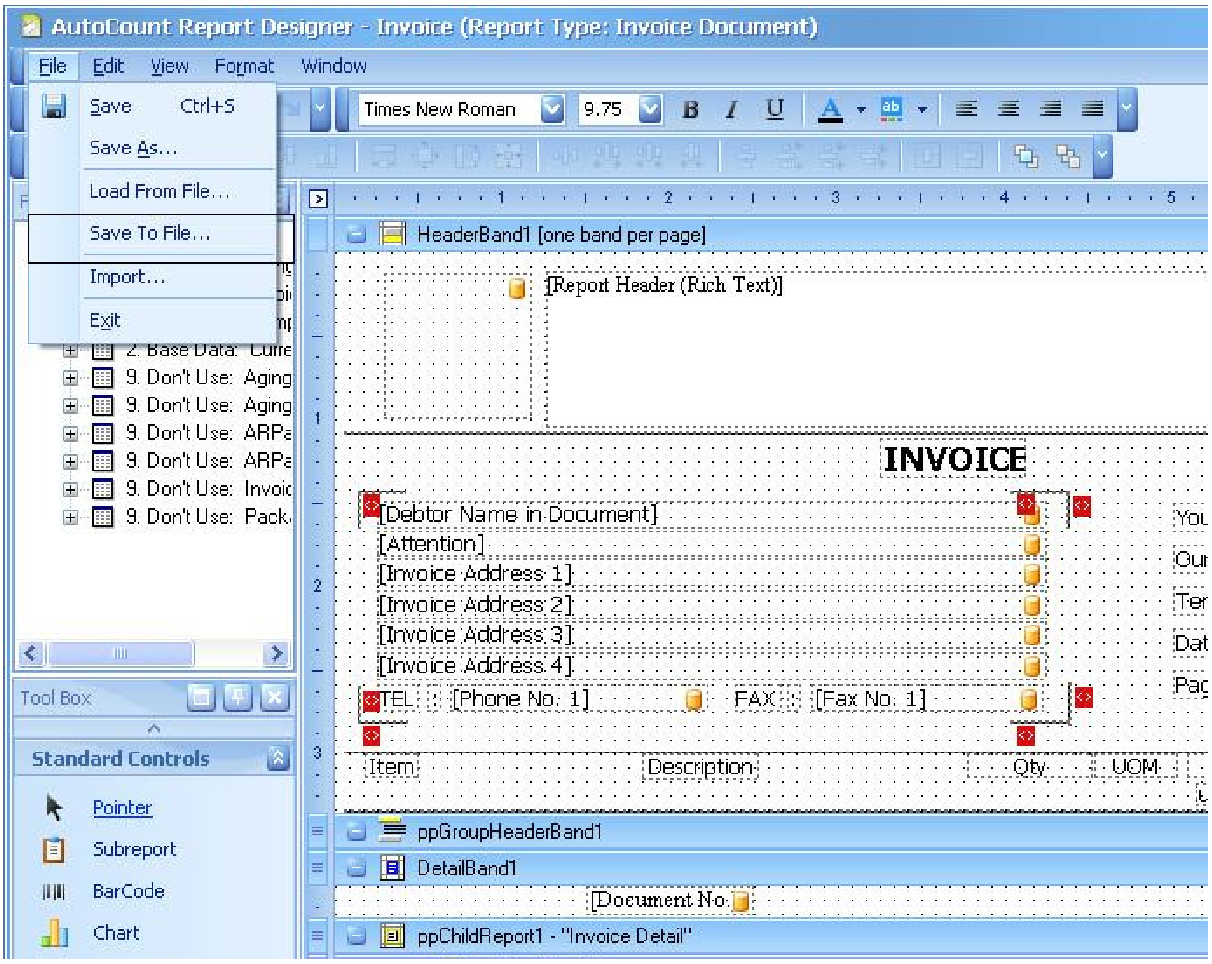
1.8.1. Save and Load to external file

AutoCount 2006 report designer provides the ability to save and load a report's layout to external files. (All the bands and controls contained in a report, along with its datasource object and other settings).

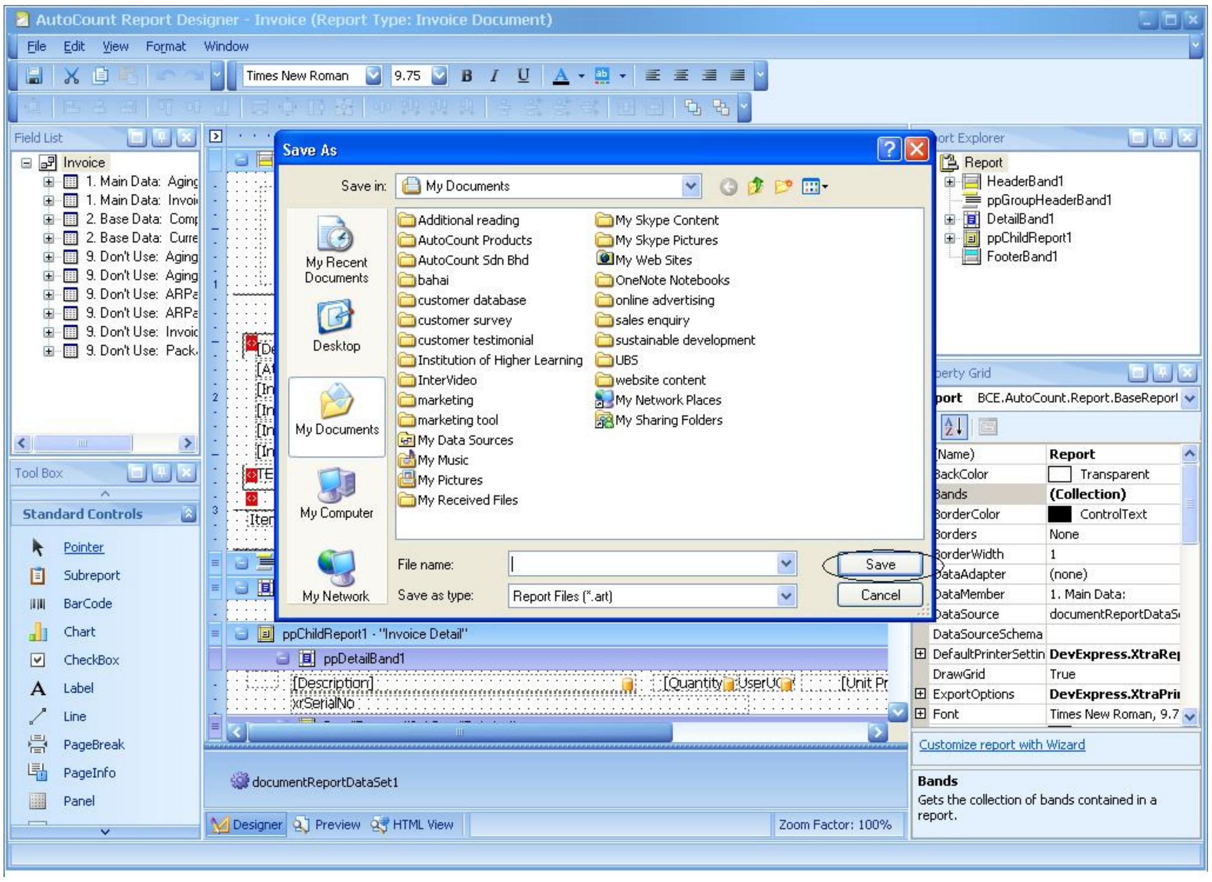
Once a report has been created, you're able to store it (to edit, preview and print it later). A report is usually saved to a file of the **.art** format.

To save to a file,

Click “File” then select “Save To File…”

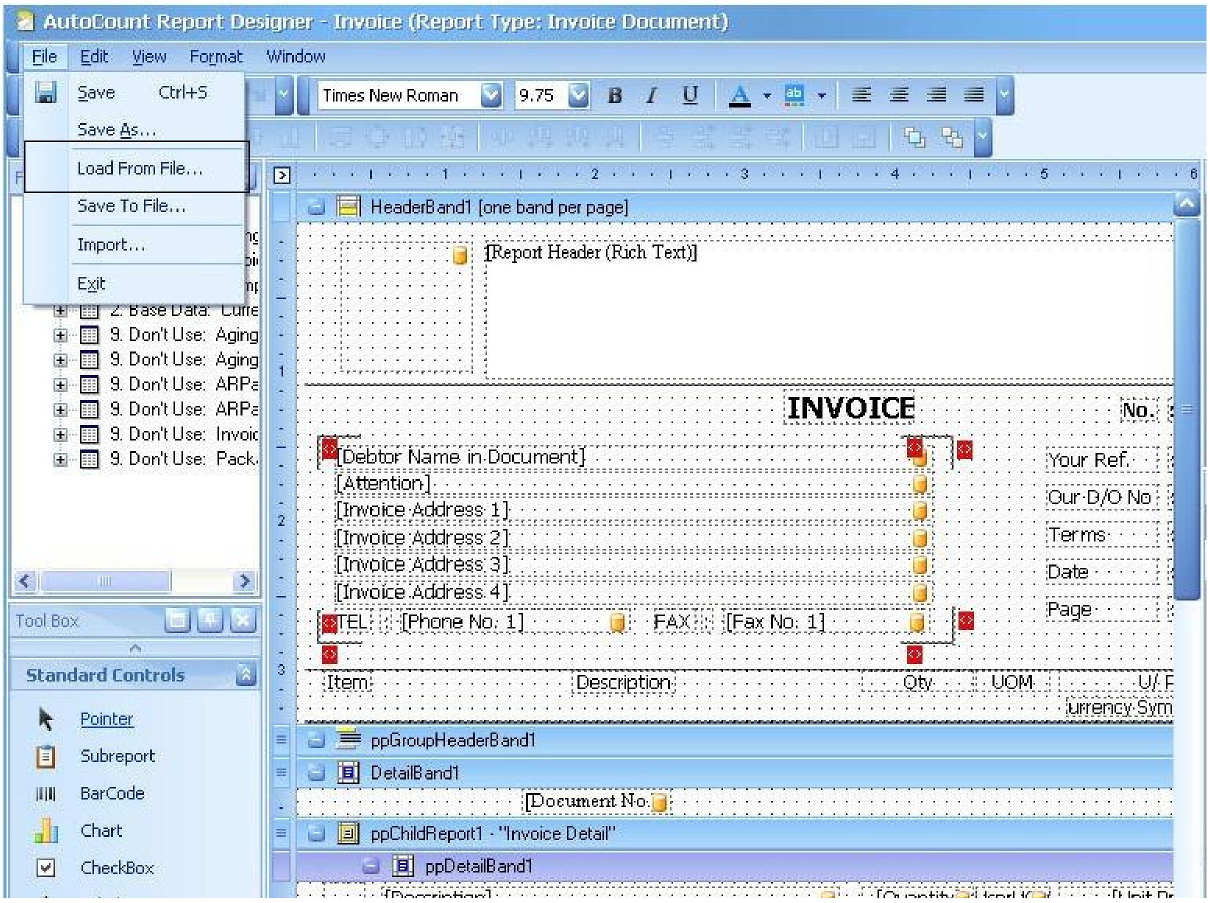


Type the file name and click “Save” button. The file will save in .art file.

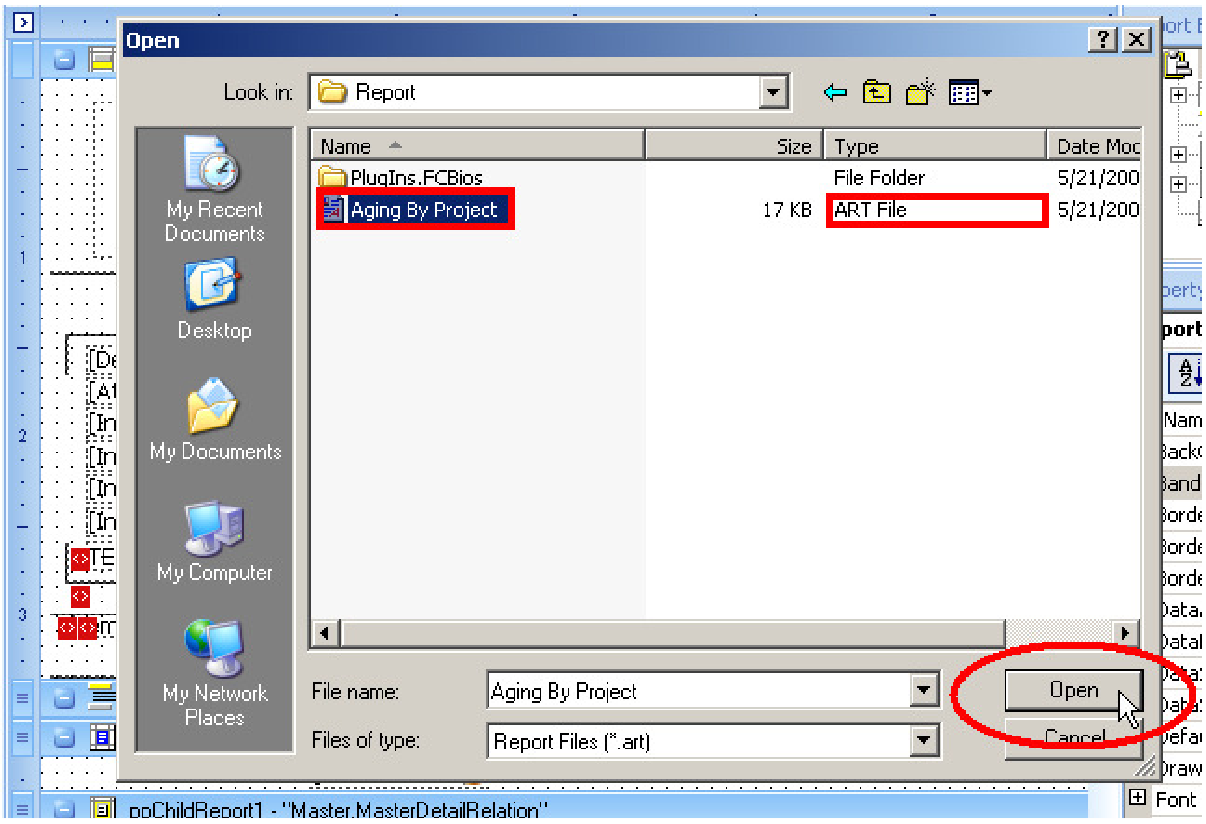


To load from a file,

Click “File” then select “Load From File…”



Select the file with .art type file and load it with "Open" button.



Open

Look in: Report

Name	Size	Type	Date Moc
PlugIns.FCBios		File Folder	5/21/200
Aging By Project	17 KB	ART File	5/21/200

File name: Aging By Project

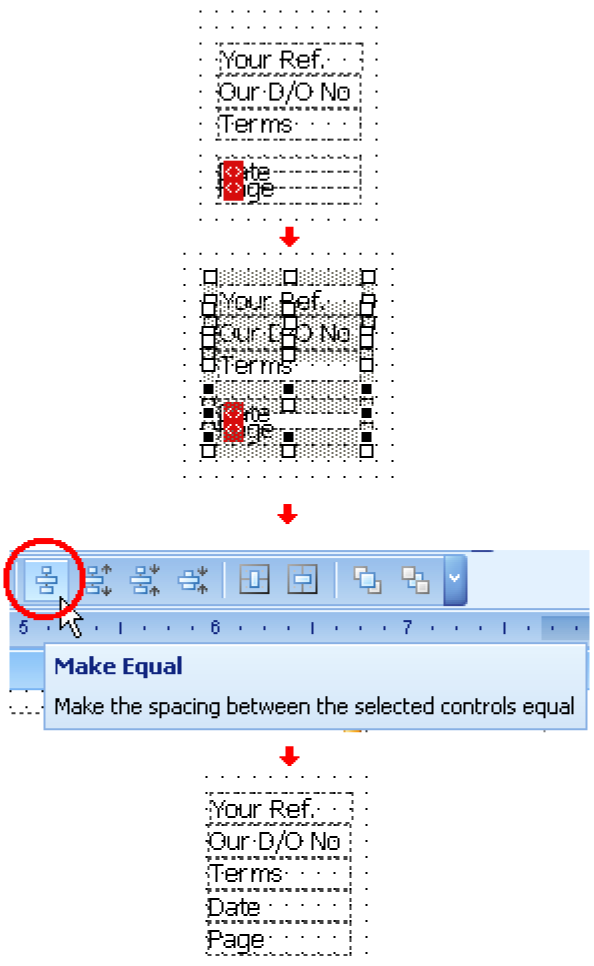
Files of type: Report Files (*.art)

1.8.2. Make Equal

If you would like to make a few labels in equal spacing, select all the labels and click the button "Make Equal" then all the label will place in equal size.

Note

-mean there are overlapping between the report controls.

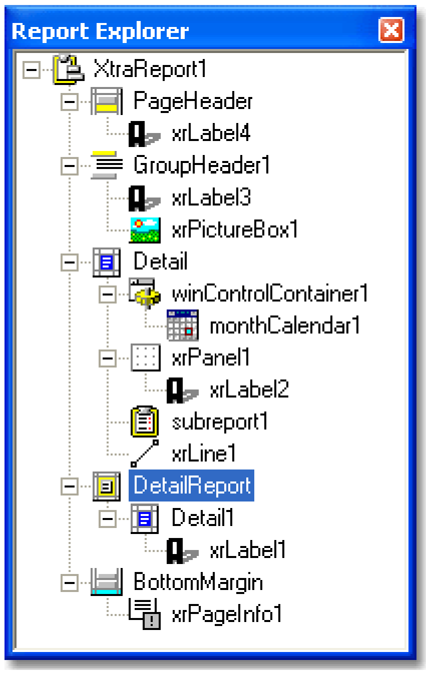


Make Equal

Make the spacing between the selected controls equal

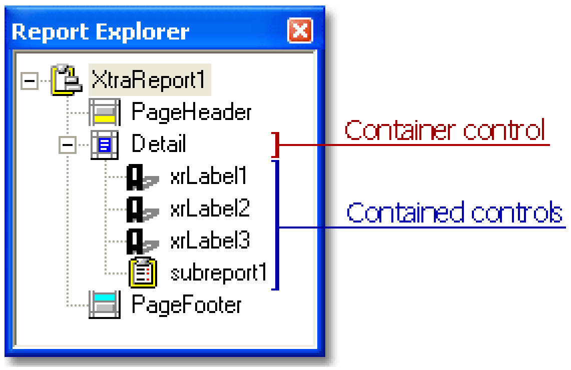
1.9. Report Explorer

The Report Explorer is a helpful tool which provides easy navigation through report elements. You can use it when building a report to quickly access all the elements of a report and their properties, and to see the report's structure. Report Explorer can be moved, sized or docked in the same manner as other IDE windows and usually has the following look.



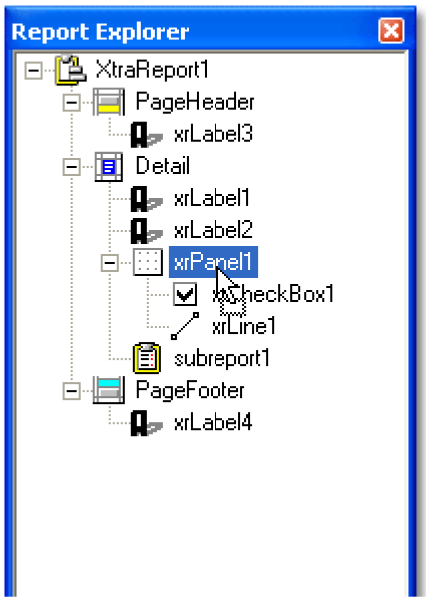
The Report Explorer displays a created report's structure using a tree view. Report elements in the tree view are displayed in a vertical order which corresponds to their vertical position on the report. When you choose an element in the Report Explorer tree the selected report element receives focus, so you can edit this element and its properties.

Usually a standard report consists of several bands containing some report controls. All child nodes in the tree view of the Report Explorer window are contained in their corresponding parent nodes . The picture below illustrates these relations.

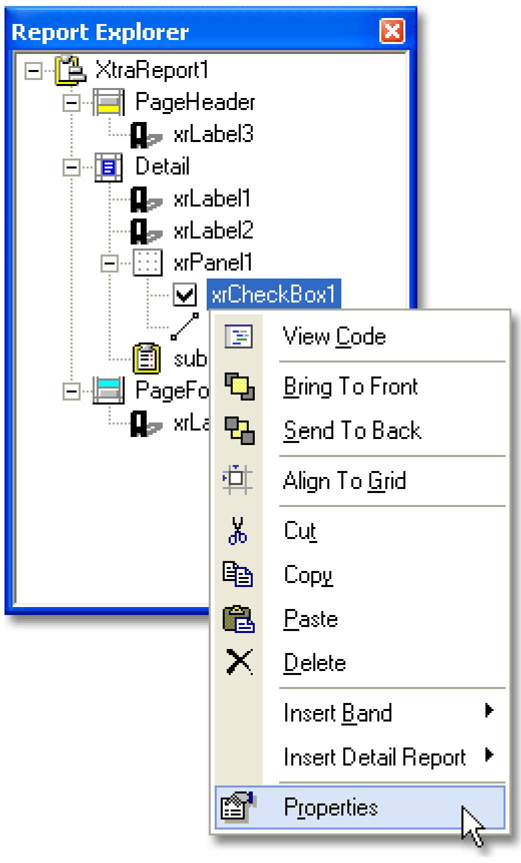


As the Report Explorer shows the report's structure, note that there are special rules for report building (for instance, XRControl objects should be contained in Band objects, not vice versa).

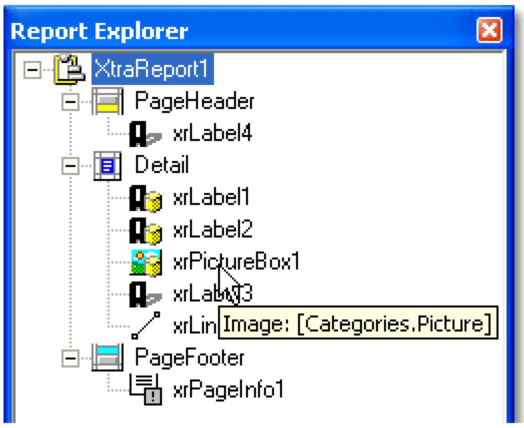
The Report Explorer supports drag and drop. This means that items representing report controls can be dragged and dropped onto band items or onto XRPanel control items. The corresponding report control will then be moved in the report.



Additionally, any item in the **Report Explorer** tree can be right clicked to invoke the context menu for this report element (the same as in the report designer window). Use this menu to quickly perform commonly used functions for the report element. An example of this menu is shown in the image below.

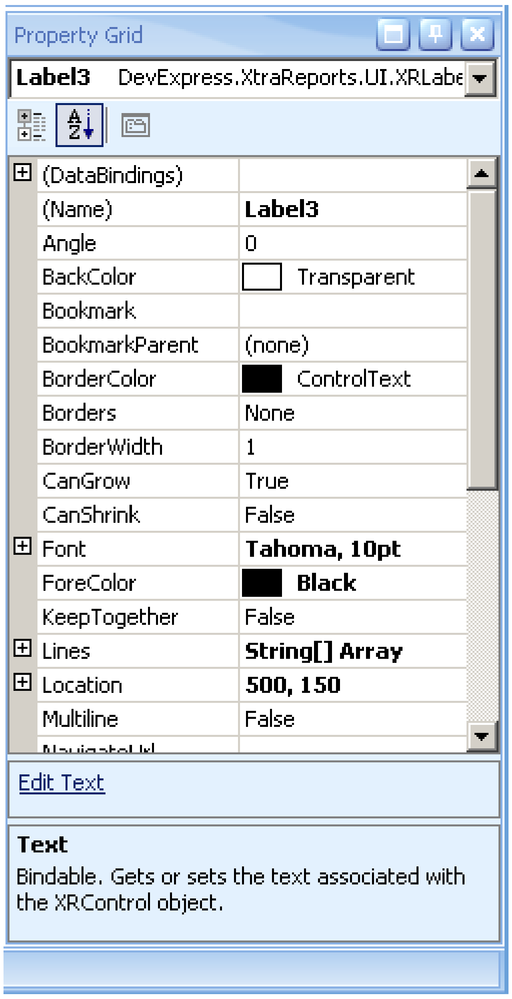


If there are any bound controls in a report, they will be marked in the **Report Explorer** with the yellow database icon as shown in the image below. Note that when a mouse pointer hovers over a bound item, a tooltip displaying the binding information is shown.


Report Explorer
XtraReport1
PageHeader
xrLabel4
Detail
xrLabel1
xrLabel2
xrPictureBox1
xrLabel3
xrLin
Image: [Categories.Picture]
PageFooter
xrPageInfo1

1.10. Property Grid

This window is used to change some of the properties of the report elements (bands and controls).



Label3 DevExpress.XtraReports.UI.XRLabe

⊞ (DataBindings)	
(Name)	**Label3**
Angle	0
BackColor	☐ Transparent
Bookmark	
BookmarkParent	(none)
BorderColor	■ ControlText
Borders	None
BorderWidth	1
CanGrow	True
CanShrink	False
⊞ Font	**Tahoma, 10pt**
ForeColor	■ **Black**
KeepTogether	False
⊞ Lines	**String[] Array**
⊞ Location	**500, 150**
Multiline	False

Text
Bindable. Gets or sets the text associated with the XRControl object.

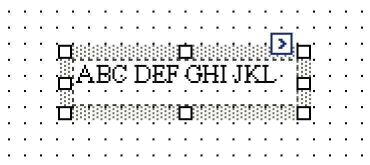
1.10.1. CanGrow property

true if the control's height can grow in order to display all its text; otherwise, **false**.

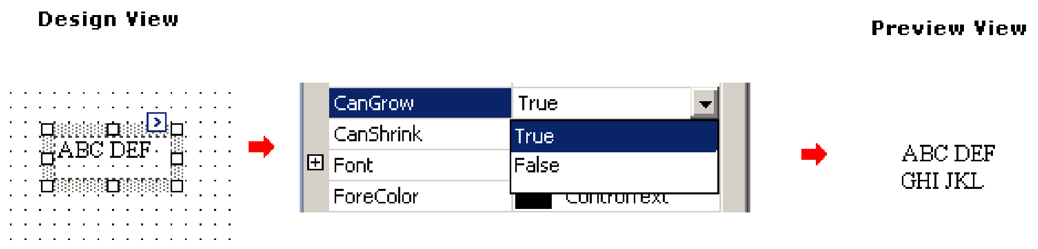
When the **CanGrow** property is set to **true** the control's height will be automatically increased (if required) so that all the text it contains is displayed. If there are other controls below the given control they will be moved down to prevent them from being overlapped.

Example:

We take this label as example



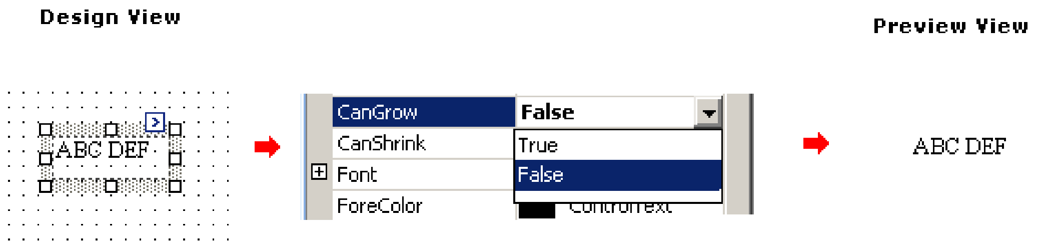
When set **CanGrow** to **True**, it is shown as below.



CanGrow True
CanShrink True
Font False
ForeColor ControlText

ABC DEF
GHI JKL

When set **CanGrow** to **False**, it is shown as below



CanGrow False
CanShrink True
Font False
ForeColor ControlText

ABC DEF

1.10.2. CanShrink property

true if the control's height can decrease in order to remove the unused space; otherwise, **false**.

When the **CanShrink** property is set to **true** and the control's text doesn't completely fill the control, then its height will be decreased to the height of its text. If there are other controls below the current control they will be moved up to avoid the unused space.

1.10.3. KeepTogether Property

true to keep the contents of the entire control on a single page; otherwise, false. The default is **false**.

This property is overridden to change its default value to **false**.

The **KeepTogether** property specifies whether the contents of the control can be horizontally split across pages. In other words, if the contents of the control are larger than the remaining space on the page, this property specifies whether the control's contents should be split across the current page and the next page, or whether the control will be printed in its entirety on the next page. Note, that this property is in effect only when a control's contents don't fit onto the current page.

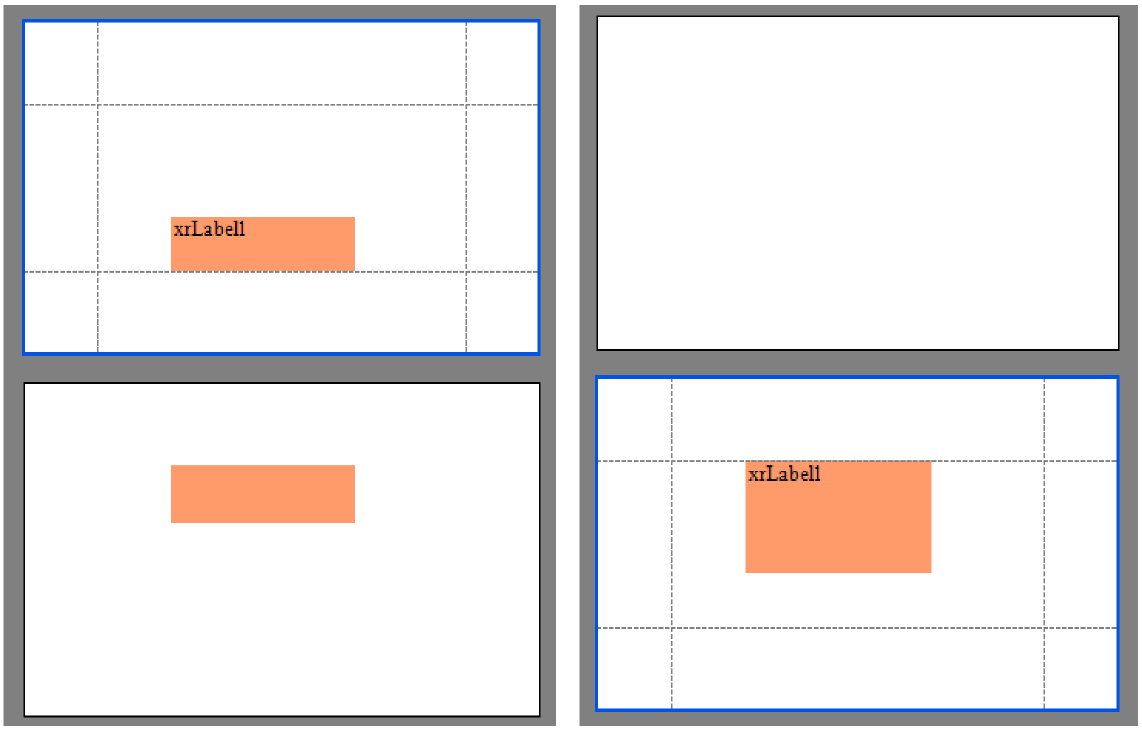
Note

If the control still can't be printed in it's entirety on the next page (there isn't enough space to keep all the control contents together), then it will be split as though this property's value is set to **false**.

The following example demonstrates how the **KeepTogether** property is in effect for an XRLabel control.

The image below demonstrates a simple report with a label, in which the **KeepTogether** property is set to **false**. In this instance, a label is automatically split across two pages.

The next image illustrates the same label, in which the **KeepTogether** property is set to **true**. Note that in this instance, a label is entirely moved to the next page to "keep its contents together".



Note

This property is in effect when a control is split in a horizontal direction only. In a vertical direction, if a control contents are outside the right page margin, this part will always be moved to the next page.

Band.KeepTogether

true to keep the contents of the entire band on a single page; otherwise, **false**. The default is **false**.

The **KeepTogether** property specifies whether the contents of the band can be horizontally split across pages. In other words, if the content of the band is larger than the remaining space on the page, this property specifies whether the band's content should be split across the current page and the next page, or whether the band will be printed in its entirety on the next page. Note, that this property is in effect only when a band's contents don't fit onto the current page.

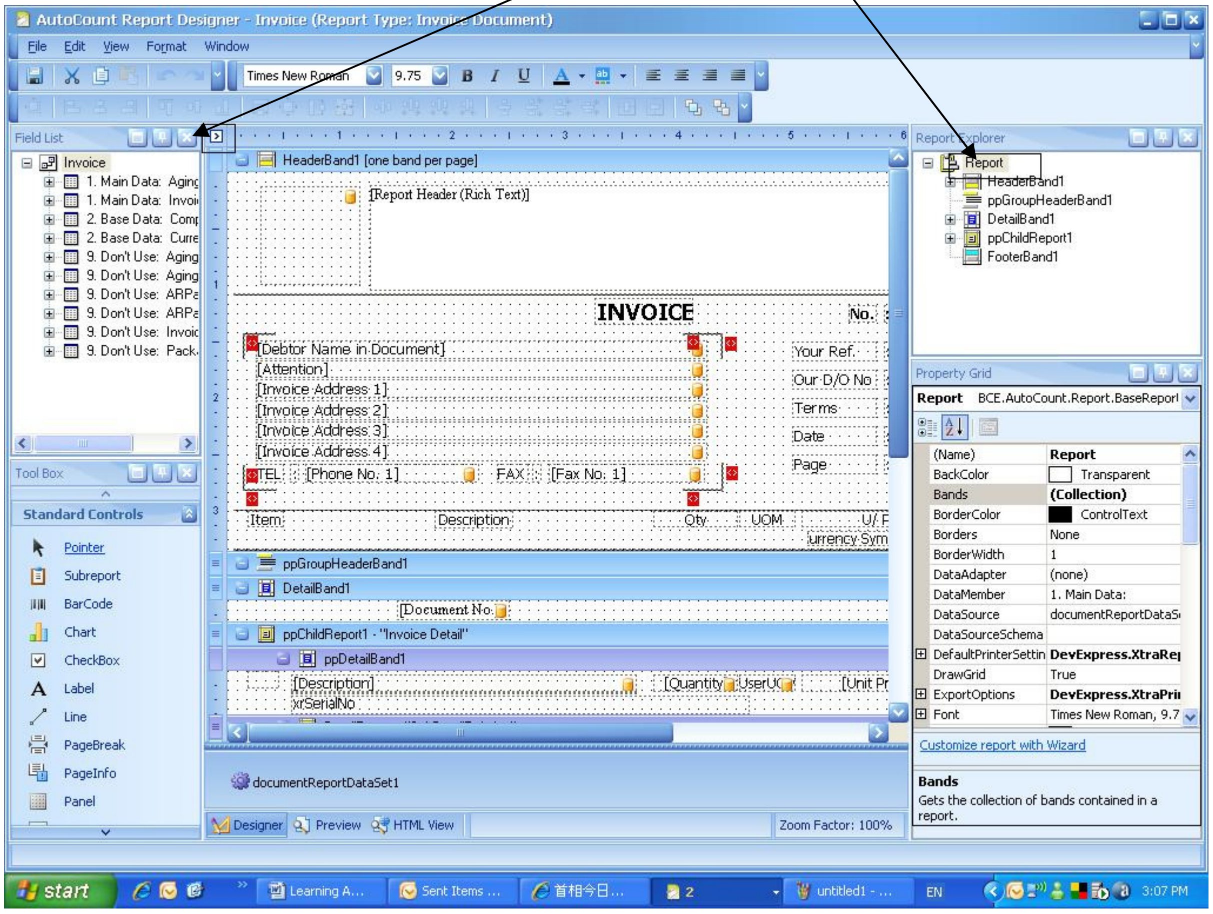
Note

If the band still can't be printed in it's entirety on the next page (there isn't enough space to keep all the band contents together), then it will be split as though this property's value is set to **false**.

1.10.4. Report Property Grid

At Report Property Grid, User are able to setting the paper kind, margins, font size, landscape and many other function for the report.

To go to the Report Property Grid, user can either select at 1 and 2 as shown at below, the property grid will show the the the Report Property Grid. User are able to select the function at the property grid.

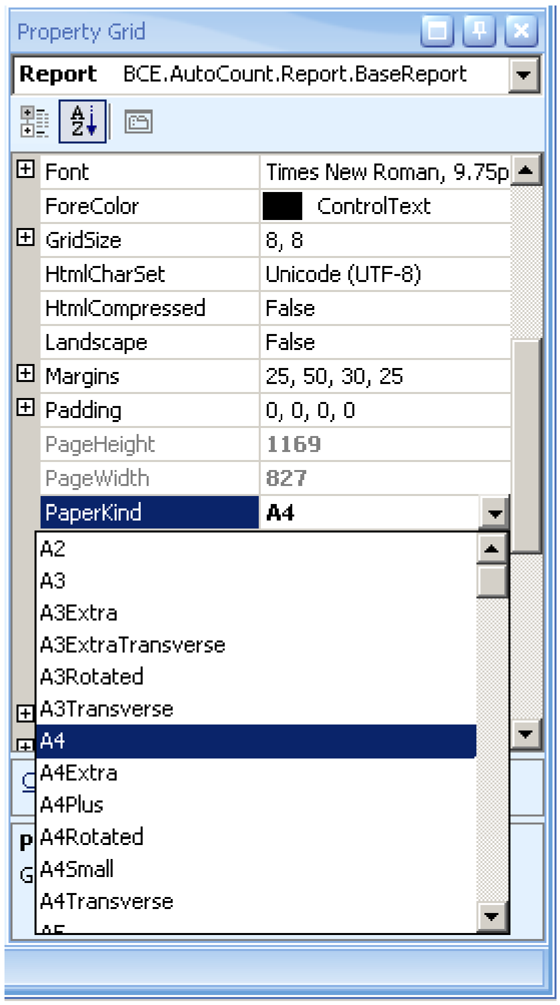


(Name)	Report
BackColor	Transparent
Bands	(Collection)
BorderColor	ControlText
Borders	None
BorderWidth	1
DataAdapter	(none)
DataMember	1. Main Data:
DataSource	documentReportDataS
DataSourceSchema	
DefaultPrinterSettin	DevExpress.XtraRe
DrawGrid	True
ExportOptions	DevExpress.XtraPri
Font	Times New Roman, 9.7

Bands
Gets the collection of bands contained in a report.

1.10.4.1. Paper Kind

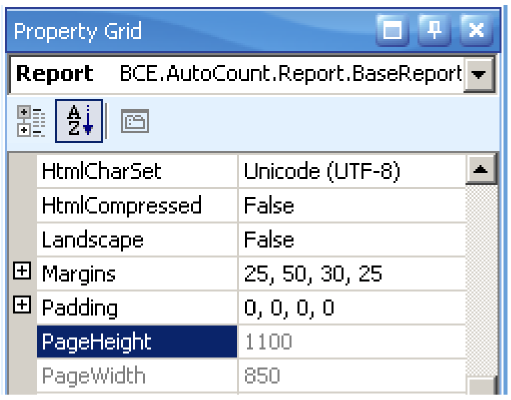
User are able to select the paper size needed design as over here. A lot of selection like A2, A3, A4, Letter, and many are available.



Font	Times New Roman, 9.75p
ForeColor	■ ControlText
GridSize	8, 8
HtmlCharSet	Unicode (UTF-8)
HtmlCompressed	False
Landscape	False
Margins	25, 50, 30, 25
Padding	0, 0, 0, 0
PageHeight	1169
PageWidth	827
PaperKind	A4

A2
A3
A3Extra
A3ExtraTransverse
A3Rotated
A3Transverse
A4
A4Extra
A4Plus
A4Rotated
A4Small
A4Transverse

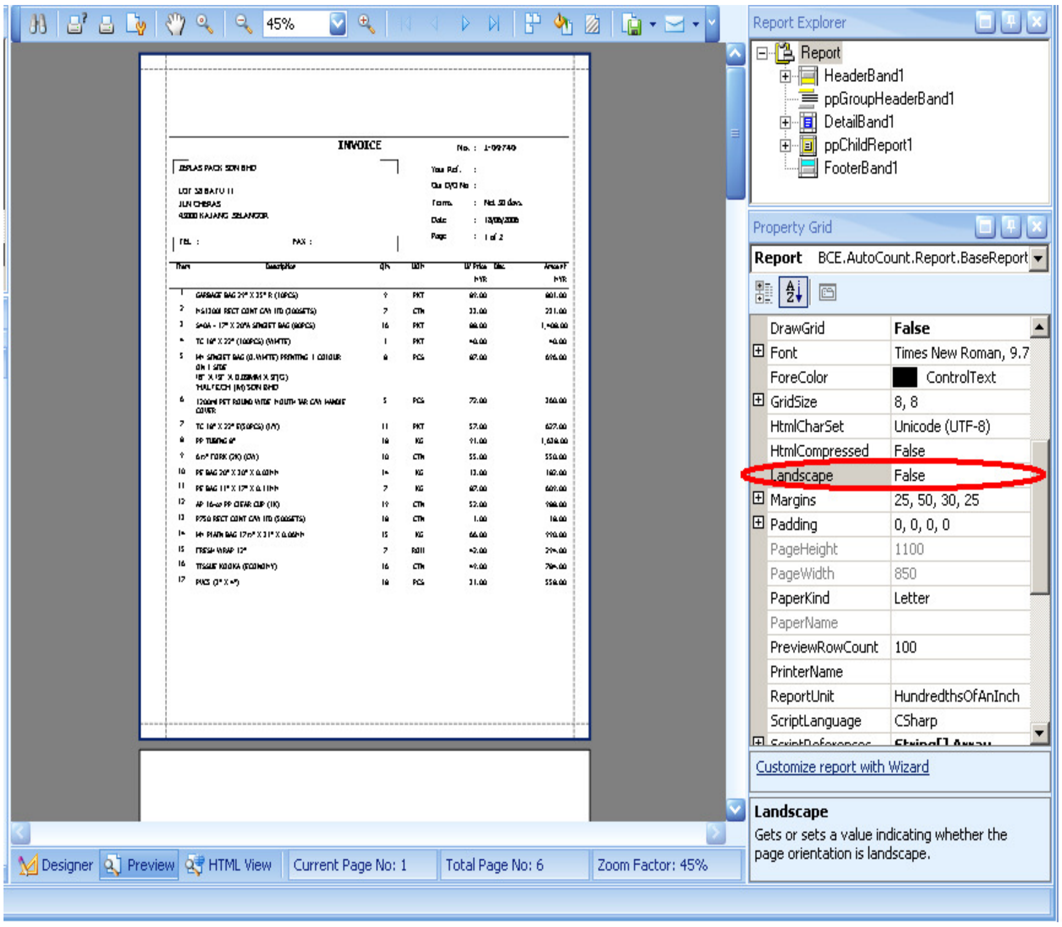
If the PaperKind property is set to Custom, then the printer paper will be selected according to the PaperName property's value. In this case it's also necessary to set the PageWidth and PageHeight properties to the corresponding values of the paper selected.



HtmlCharSet	Unicode (UTF-8)
HtmlCompressed	False
Landscape	False
Margins	25, 50, 30, 25
Padding	0, 0, 0, 0
PageHeight	1100
PageWidth	850

1.10.4.2. Landscape

User are able to set the report print as Landscape or Portrait. As default Landscape is set as false which mean is in Portrait mode. To set as Landscape select it as True for Landscape.


Report Explorer

- Report
  - HeaderBand1
  - ppGroupHeaderBand1
  - DetailBand1
  - ppChildReport1
  - FooterBand1

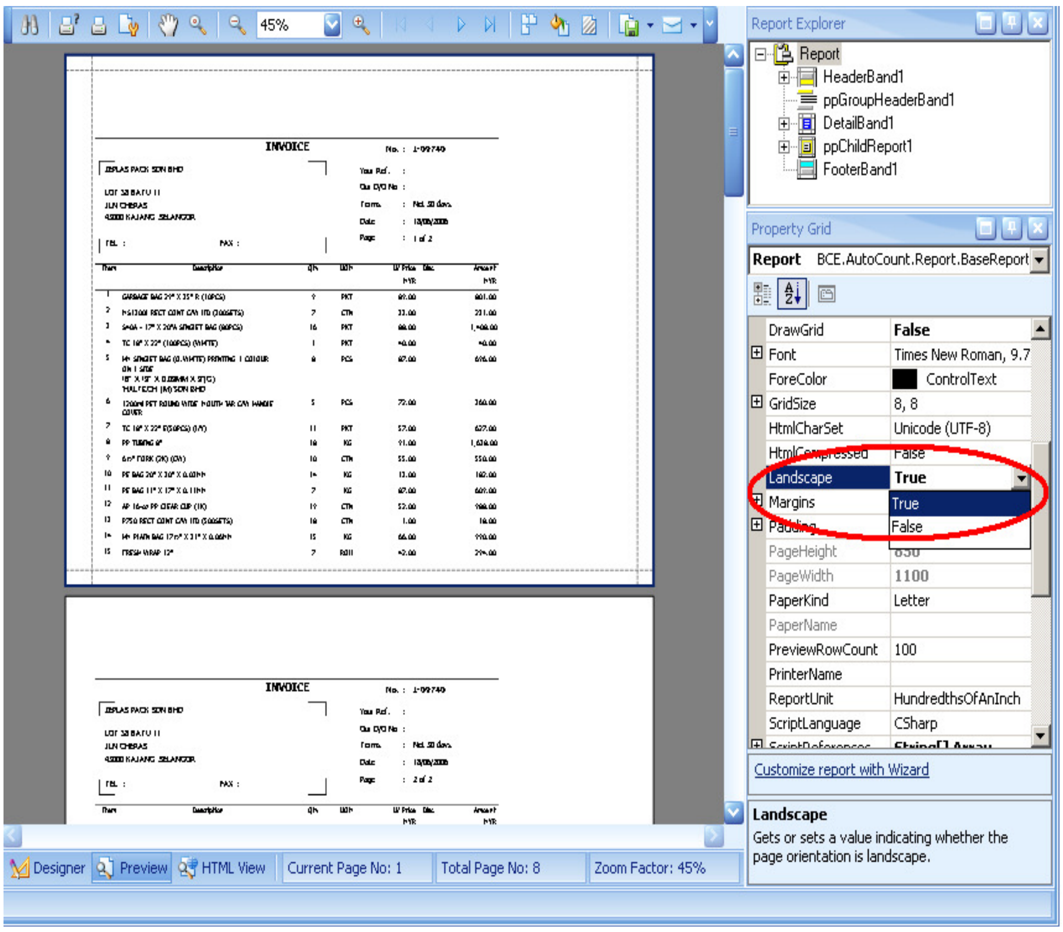
Property Grid

Report BCE.AutoCount.Report.BaseReport

DrawGrid	**False**
Font	Times New Roman, 9.7
ForeColor	ControlText
GridSize	8, 8
HtmlCharSet	Unicode (UTF-8)
HtmlCompressed	False
Landscape	False
Margins	25, 50, 30, 25
Padding	0, 0, 0, 0
PageHeight	1100
PageWidth	850
PaperKind	Letter
PaperName	
PreviewRowCount	100
PrinterName	
ReportUnit	HundredthsOfAnInch
ScriptLanguage	CSharp

Customize report with Wizard

Landscape
Gets or sets a value indicating whether the page orientation is landscape.



Report Explorer

- Report
  - HeaderBand1
  - ppGroupHeaderBand1
  - DetailBand1
  - ppChildReport1
  - FooterBand1

Property Grid

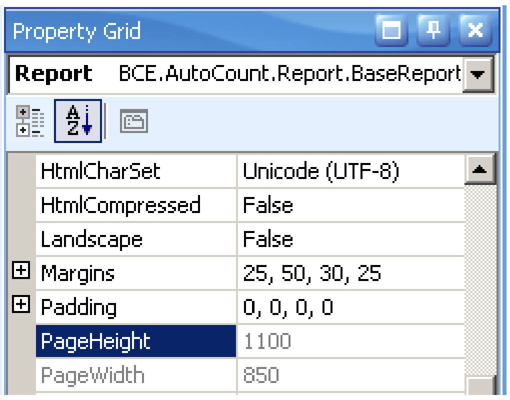
Report BCE.AutoCount.Report.BaseReport

DrawGrid	**False**
Font	Times New Roman, 9.7
ForeColor	■ ControlText
GridSize	8, 8
HtmlCharSet	Unicode (UTF-8)
HtmlCompressed	False
Landscape	**True**
Margins	True
Padding	False
PageHeight	850
PageWidth	1100
PaperKind	Letter
PaperName	
PreviewRowCount	100
PrinterName	
ReportUnit	HundredthsOfAnInch
ScriptLanguage	CSharp

Customize report with Wizard

Landscape
Gets or sets a value indicating whether the page orientation is landscape.

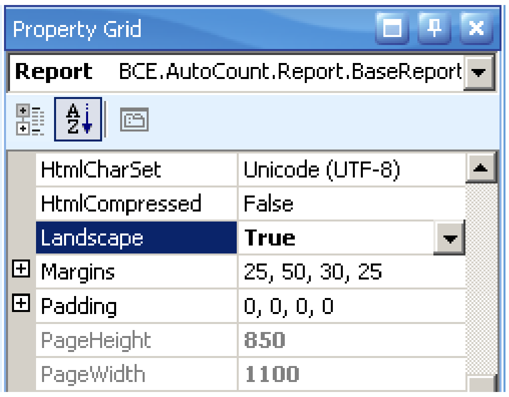
When the Landscape property value is changed, then the PageWidth and PageHeight properties values will be automatically swapped with each other.



Property Grid

Report BCE.AutoCount.Report.BaseReport

HtmlCharSet	Unicode (UTF-8)
HtmlCompressed	False
Landscape	False
Margins	25, 50, 30, 25
Padding	0, 0, 0, 0
PageHeight	1100
PageWidth	850



Property Grid

Report BCE.AutoCount.Report.BaseReport

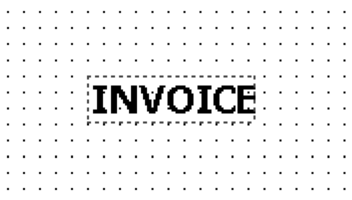
HtmlCharSet	Unicode (UTF-8)
HtmlCompressed	False
Landscape	**True**
Margins	25, 50, 30, 25
Padding	0, 0, 0, 0
PageHeight	**850**
PageWidth	**1100**

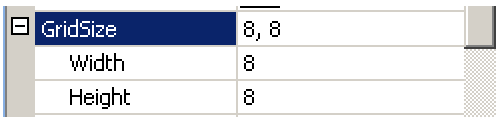
1.10.4.3. Grid Size

The GridSize default value is '8 , 8', user can set the value of the GridSize to make the design work more easy. The smaller value can let user easy to place the label.

Below is an example that GridSize set as.

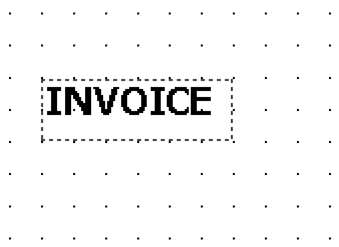
GridSize '8 , 8'

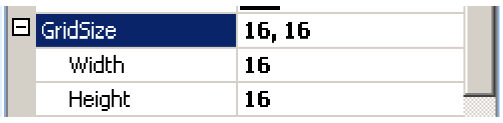




GridSize	8, 8
Width	8
Height	8

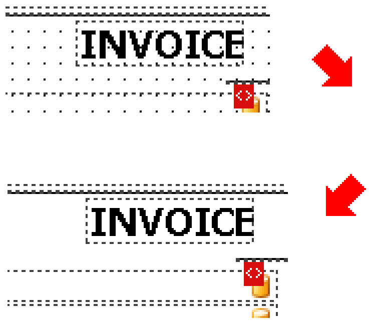
GridSize '16 , 16'

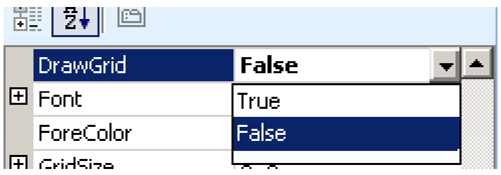




GridSize	16, 16
Width	16
Height	16

User are able to make the grid into invisible mode for more easy work. Set the DrawGrid to false then the grid will become invisible.



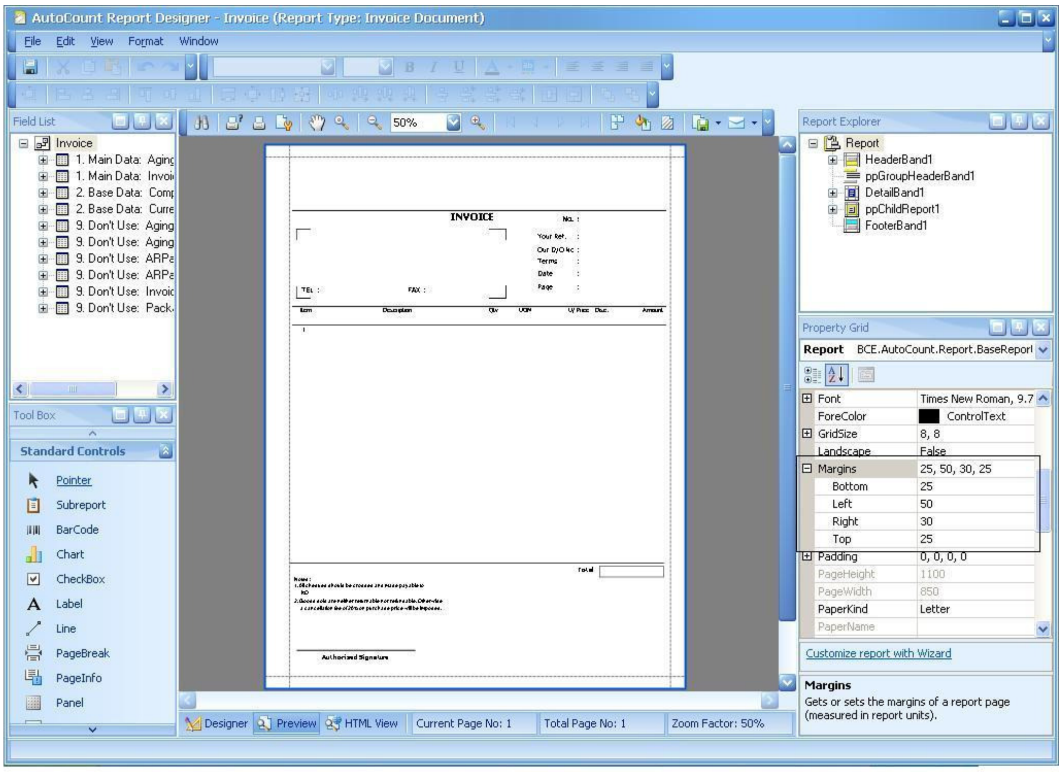


DrawGrid	False
Font	True
ForeColor	False

44

1.10.4.4. Margins

User are able to set the report margin by setting the value at the property grid as shown as below. the margins can be set is “Bottom”, “Left”,”Right” and “Top”. The margins are measured in report units (either in hundredths of an inch, or in tenths of a millimeter).



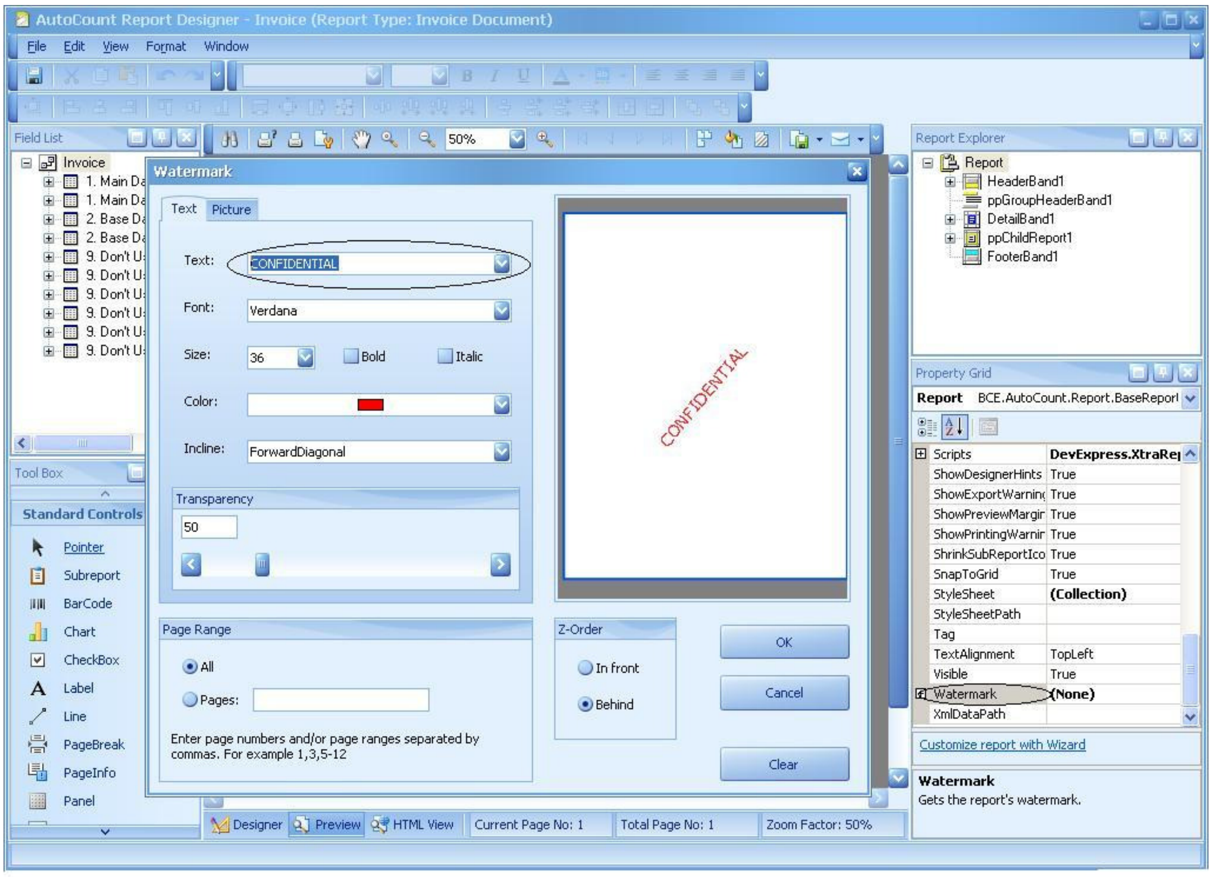
Report BCE.AutoCount.Report.BaseReport

Font	Times New Roman, 9.7
ForeColor	ControlText
GridSize	8, 8
Landscape	False
Margins	25, 50, 30, 25
Bottom	25
Left	50
Right	30
Top	25
Padding	0, 0, 0, 0
PageHeight	1100
PageWidth	850
PaperKind	Letter
PaperName	

Margins
Gets or sets the margins of a report page (measured in report units).

1.10.4.5. Watermark

A report can contain a collection of watermarks set either for all or specified report pages. The report’s watermark can be represented either by text or a picture.
To edit a watermark click the button on property grid. The watermark text can be select or can type iby user. User also can set the font, size, color, incline, transparency and the text properties. Beside text picture watermark also available.


AutoCount Report Designer - Invoice (Report Type: Invoice Document)

File Edit View Format Window

Field List
- Invoice
  - 1. Main Da
  - 1. Main Da
  - 2. Base Da
  - 2. Base Da
  - 9. Don't U
  - 9. Don't U
  - 9. Don't U
  - 9. Don't U
  - 9. Don't U
  - 9. Don't U

Tool Box

Standard Controls
- Pointer
- Subreport
- BarCode
- Chart
- CheckBox
- Label
- Line
- PageBreak
- PageInfo
- Panel

Watermark

Text | Picture

Text: CONFIDENTIAL

Font: Verdana

Size: 36 Bold Italic

Color: 🟥

Incline: ForwardDiagonal

Transparency: 50

Page Range: [✓ All] Pages:

Enter page numbers and/or page ranges separated by commas. For example 1,3,5-12

Z-Order: In front [✓ Behind]

OK | Cancel | Clear

Designer | Preview | HTML View | Current Page No: 1 | Total Page No: 1 | Zoom Factor: 50%

Report Explorer
- Report
  - HeaderBand1
  - ppGroupHeaderBand1
  - DetailBand1
  - ppChildReport1
  - FooterBand1

Property Grid

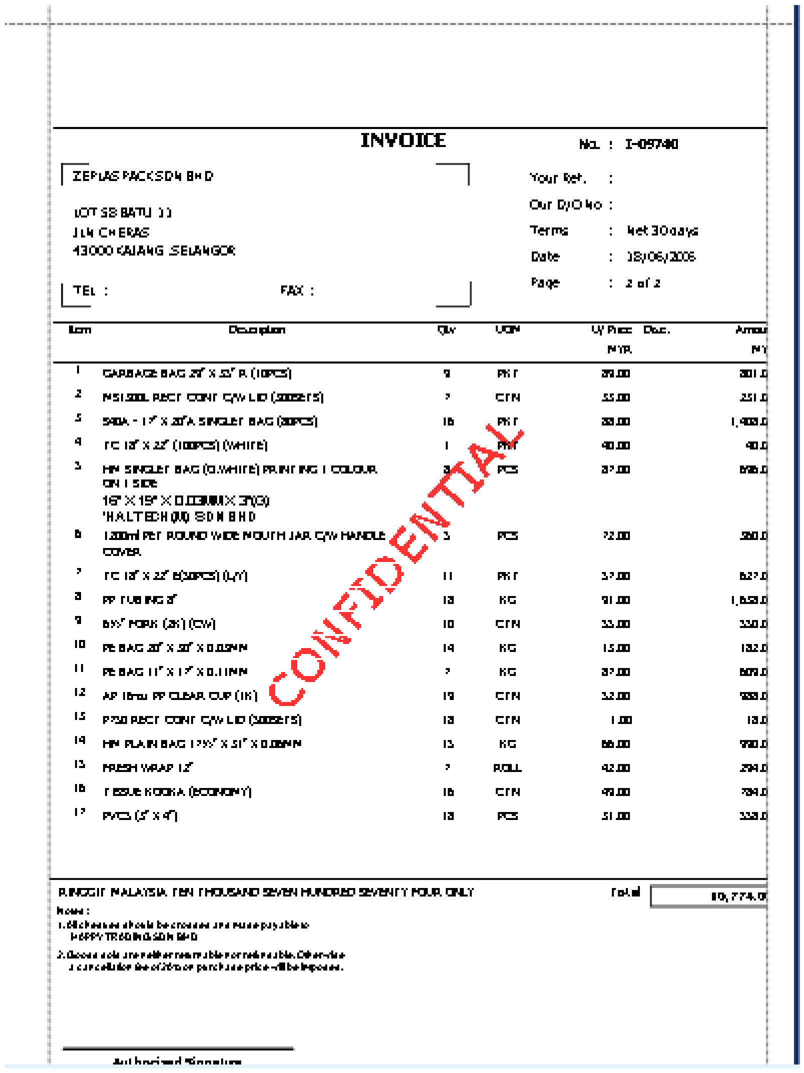
Report BCE.AutoCount.Report.BaseReport

Scripts	DevExpress.XtraRe
ShowDesignerHints	True
ShowExportWarning	True
ShowPreviewMargir	True
ShowPrintingWarnir	True
ShrinkSubReportIco	True
SnapToGrid	True
StyleSheet	(Collection)
StyleSheetPath	
Tag	
TextAlignment	TopLeft
Visible	True
Watermark	(None)
XmlDataPath	

Customize report with Wizard

Watermark
Gets the report's watermark.

The output of the watermark is shown as below


CONFIDENTIAL

INVOICE

No. : I-09740

ZEPLAS PACK SDN BHD

LOT 58 BATU 11
JLN CHERAS
43000 KAJANG SELANGOR

TEL :　　　FAX :

Your Ref. :
Our D/O No :
Terms : Net 30days
Date : 18/06/2006
Page : 2 of 2

Item	Description	Qty	UOM	U/ Price MYR	Disc.	Amou MY
1	GARBAGE BAG 27" X 32" R (10PCS)	9	PKT	39.00		301.0
2	MS1200L RECT CONT C/W LID (200SETS)	7	CTN	55.00		251.0
3	540A - 17" X 28"A SINGLET BAG (30PCS)	16	PKT	88.00		1,408.0
4	TC 18" X 22" (100PCS) (WHITE)	1	PKT	40.00		40.0
5	HM SINGLET BAG (D.WHITE) PRINTING 1 COLOUR ON 1 SIDE 16" X 19" X 0.013MM X 37(3) 'HALTECH (M) SDN BHD	3	PCS	37.00		888.0
6	1200ml PET ROUND WIDE MOUTH JAR C/W HANDLE COVER	5	PCS	72.00		360.0
7	TC 18" X 22" B(30PCS) (L/Y)	11	PKT	57.00		627.0
8	PP TUBING 8"	18	KG	91.00		1,638.0
9	6½" FORK (2K) (CW)	10	CTN	33.00		330.0
10	PE BAG 20" X 30" X 0.05MM	14	KG	13.00		182.0
11	PE BAG 11" X 17" X 0.11MM	7	KG	87.00		609.0
12	AP 16oz PP CLEAR CUP (1K)	19	CTN	52.00		988.0
13	P750 RECT CONT C/W LID (300SETS)	18	CTN	1.00		18.0
14	HM PLAIN BAG 17½" X 31" X 0.08MM	15	KG	66.00		990.0
15	FRESH WRAP 12"	7	ROLL	42.00		294.0
16	TISSUE KOOKA (ECONOMY)	16	CTN	49.00		784.0
17	PVC3 (3" X 4")	18	PCS	51.00		338.0

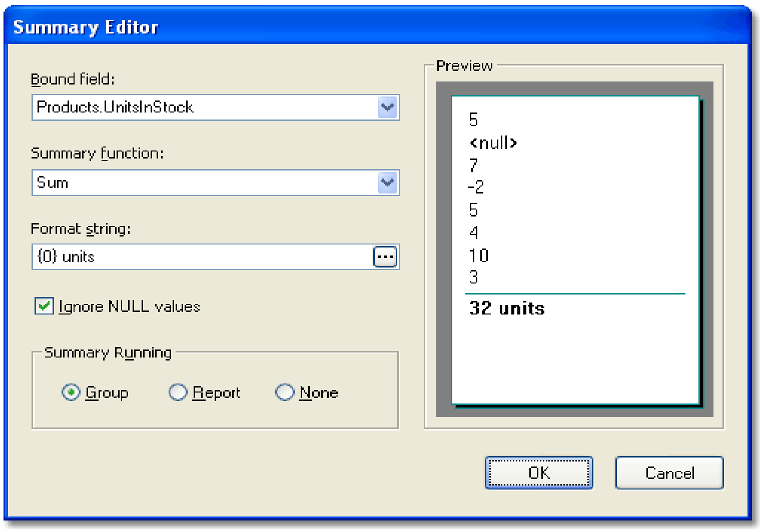
RINGGIT MALAYSIA TEN THOUSAND SEVEN HUNDRED SEVENTY FOUR ONLY　　Total 10,774.0

Notes :
1. All cheques should be crossed and made payable to
HAPPY TRADING SDN BHD
2. Goods sold are neither returnable nor refundable. Otherwise a cancellation fee of 20% on purchase price will be imposed.

Authorized Signature

1.11. Calculating an Automatic Summary

Reports also support automatic calculation of summary functions (totals, maximum, minimum, averages, etc.) only for Label controls (or its descendants) which are bound to data. At design time you can use the **Summary Editor** to set all the summary options in one place.


Summary Editor

Bound field: Products.UnitsInStock

Summary function: Sum

Format string: {0} units

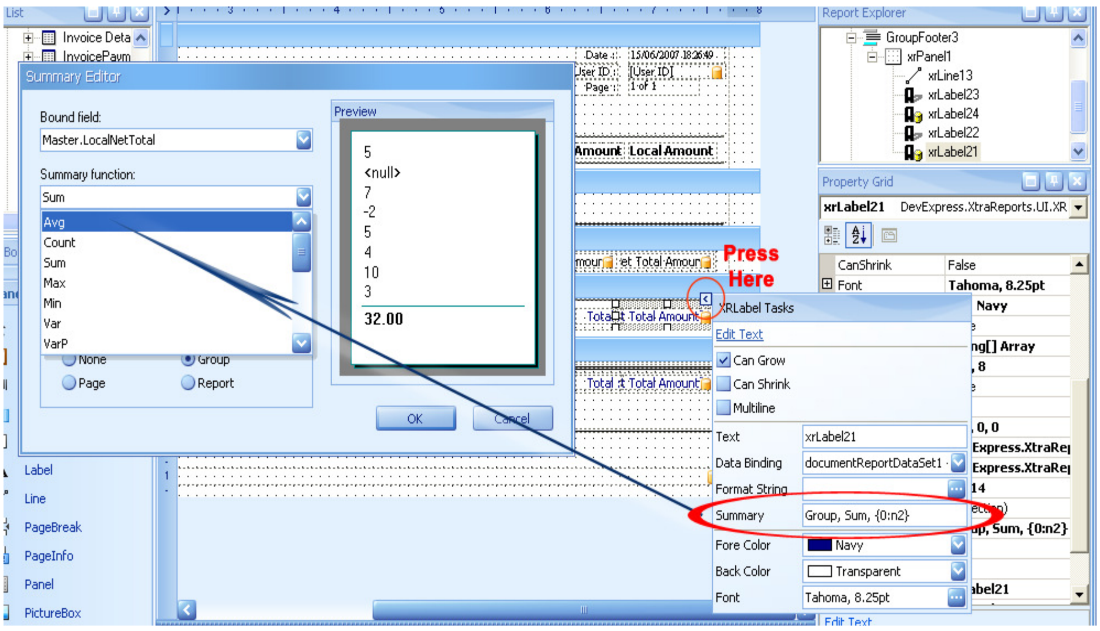
[✓] Ignore NULL values

Summary Running: [✓ Group] Report None

Preview:

```
5
<null>
7
-2
5
4
10
3
32 units
```

The following example demonstrates how to calculate the average of the values in the **Net Total** field of the **Invoice Listing**. As you can see at the summary editor, we have a list of calculation method. But the methods that often use are the Average, Count and Sum. Let us do an example now, we try use average and sum.


Summary Editor

Bound field: Master.LocalNetTotal

Summary function: Sum

Avg
Count
Sum
Max
Min
Var
VarP

None Group
Page Report

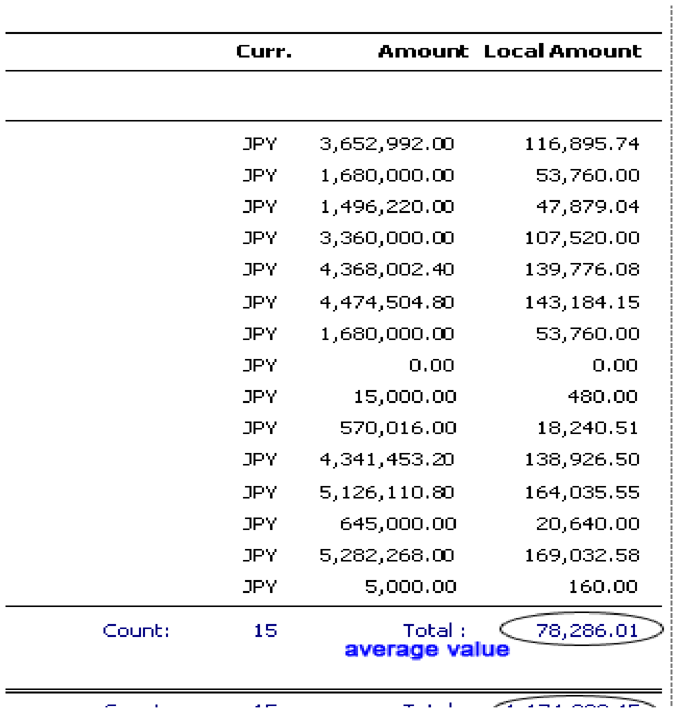
Preview:

```
5
<null>
7
-2
5
4
10
3
32.00
```

XRLabel Tasks

Text	xrLabel21
Data Binding	documentReportDataSet1
Format String	
Summary	Group, Sum, {0:n2}
Fore Color	Navy
Back Color	Transparent
Font	Tahoma, 8.25pt

The resulting report is shown in the image below

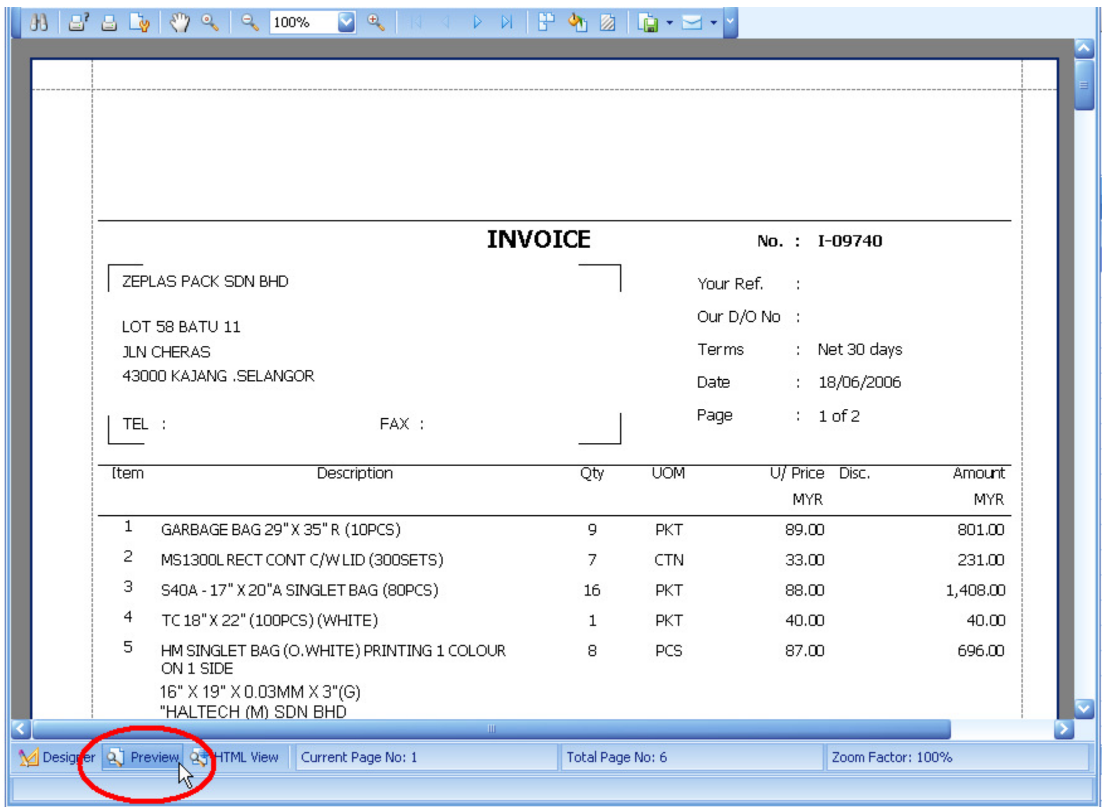


Curr.	Amount	Local Amount
JPY	3,652,992.00	116,895.74
JPY	1,680,000.00	53,760.00
JPY	1,496,220.00	47,879.04
JPY	3,360,000.00	107,520.00
JPY	4,368,002.40	139,776.08
JPY	4,474,504.80	143,184.15
JPY	1,680,000.00	53,760.00
JPY	0.00	0.00
JPY	15,000.00	480.00
JPY	570,016.00	18,240.51
JPY	4,341,453.20	138,926.50
JPY	5,126,110.80	164,035.55
JPY	645,000.00	20,640.00
JPY	5,282,268.00	169,032.58
JPY	5,000.00	160.00

Count: 15 Total : 78,286.01

1.9. Print Preview

After you have designed your report, you can preview your report immediately. Simply click the "preview" at the bottom part of the design page. At print preview page, what you see is what you will print out. At upper part of the page is a printing toolbar, user can easy print out while design report.



INVOICE

No. : **I-09740**

ZEPLAS PACK SDN BHD

LOT 58 BATU 11
JLN CHERAS
43000 KAJANG .SELANGOR

TEL :
FAX :

Your Ref.	:	
Our D/O No	:	
Terms	:	Net 30 days
Date	:	18/06/2006
Page	:	1 of 2

Item	Description	Qty	UOM	U/ Price MYR	Disc.	Amount MYR
1	GARBAGE BAG 29" X 35" R (10PCS)	9	PKT	89.00		801.00
2	MS1300L RECT CONT C/W LID (300SETS)	7	CTN	33.00		231.00
3	S40A - 17" X 20"A SINGLET BAG (80PCS)	16	PKT	88.00		1,408.00
4	TC 18" X 22" (100PCS) (WHITE)	1	PKT	40.00		40.00
5	HM SINGLET BAG (O.WHITE) PRINTING 1 COLOUR ON 1 SIDE 16" X 19" X 0.03MM X 3"(G) "HALTECH (M) SDN BHD	8	PCS	87.00		696.00

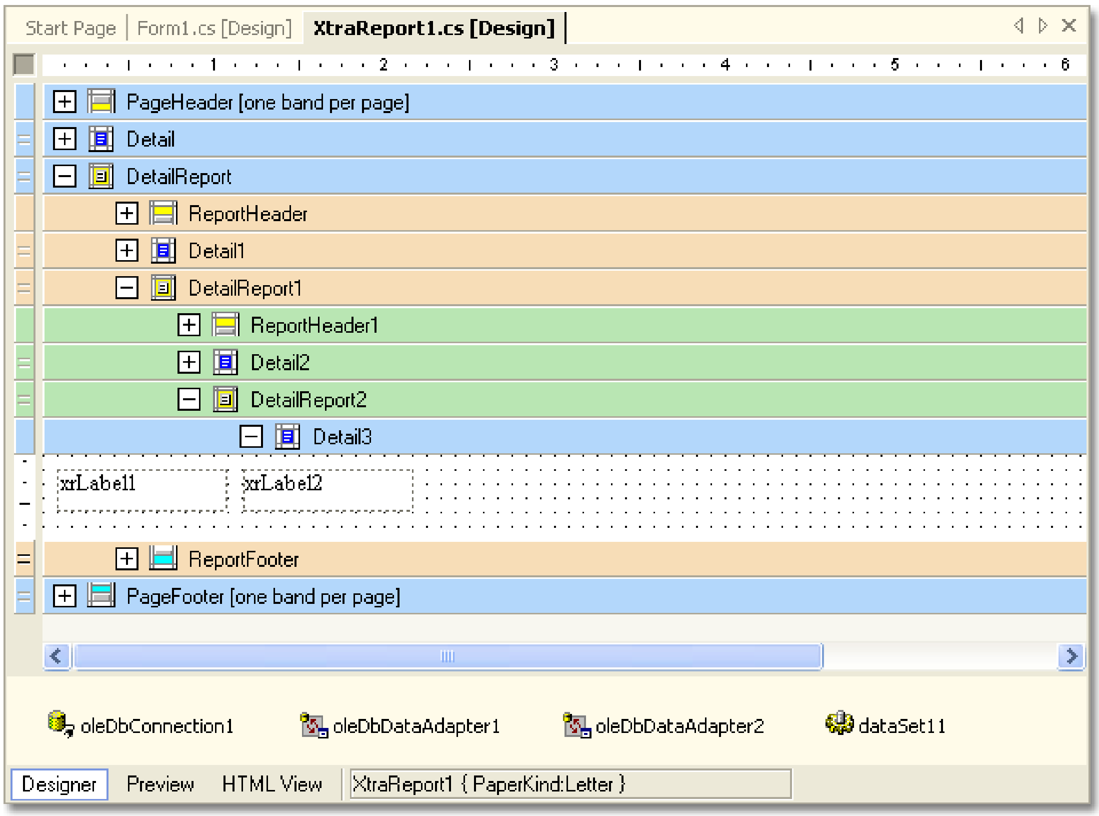
Current Page No: 1 | Total Page No: 6 | Zoom Factor: 100%

2. Master-Detail Report Using Detail Report Bands

XtraReports has a type of band that can be used to incorporate one report into another for the purpose of creating master-detail reports. This is done by using DetailReportBand objects which represent detail reports nested into a master report. Note that in this case the master and detail reports are shown in the same report designer and are bound to the same datasource which contains a data relationship. Use detail report bands instead of subreport controls to create a master-detail report.

2.1. Detail Report Bands

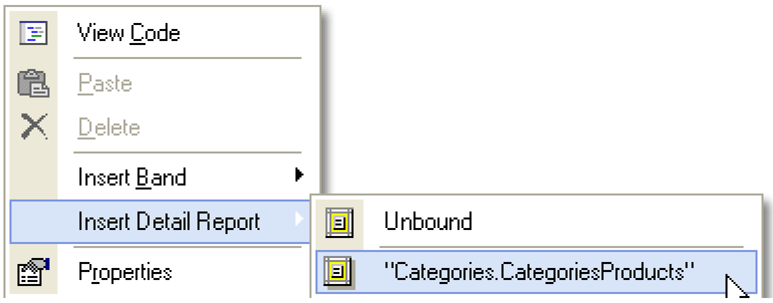
DetailReportBand objects are used in **XtraReports** to create master-detail reports based on a master-detail ADO .NET relationship. The **Detail Report** bands allow both the *master report* and *detail report* to be represented in a single report class. This means that a *master report* has a special band that contains another report including all the *detail report*'s bands. There can be an unlimited number of DetailReportBands nested in each other, and every new group of bands is colored accordingly in the report designer as shown in the image below.



Note

You can add several *detail reports* at the same nesting level.

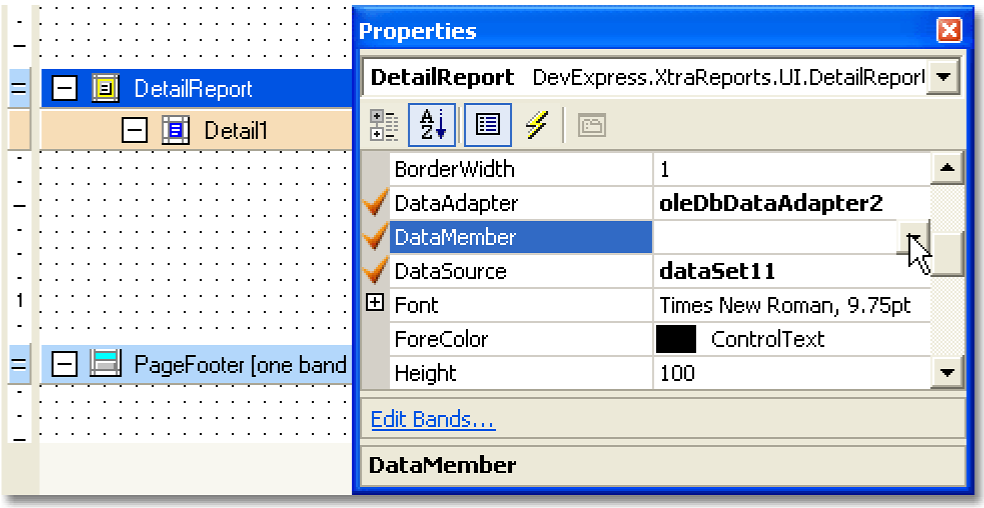
To add a DetailReportBand, just right-click the report designer and choose the **Insert Detail Report** item from the context menu. If an ADO .NET relationship exists in the datasource bound to the report, then the submenu will contain an item which has the name of that relationship. You can also add an unbound detail report and specify its main properties later on.



To bind a *detail report* to the detail data in a report's datasource the following properties should be set:

- XtraReportBase.DataSource should be set to the datasource in the *master report*.
- XtraReportBase.DataAdapter should be set to the data adapter providing the *detail* data for the report. Usually, it's different than the data adapter of the *main report*.
- XtraReportBase.DataMember should be set to the name of the data relationship used in the bound data source.

The image below illustrates these properties in the VS IDE.



BorderWidth	1
DataAdapter	oleDbDataAdapter2
DataMember	
DataSource	dataSet11
Font	Times New Roman, 9.75pt
ForeColor	ControlText
Height	100

Note

When using either the XtraReportBase.GetCurrentRow or XtraReportBase.GetCurrentColumnValue methods in a detail report's event handlers, you should provide the columns' names according to the level of the detail report. Do not call these methods for the XtraReport object itself, but for the DetailReportBand instead. For instance:

C#

```
// Don't use the GetCurrentRow method in this way in a detail report.
//
((DataRowView)GetCurrentRow()).Row["Categories.CategoriesProducts.ProductName
"].ToString();


// Get the current value of a data row in a detail report.
xrLabel1.Text =
((DataRowView)DetailReport.GetCurrentRow()).Row["ProductName"].ToString();


// Don't use the GetCurrentColumnValue method in this way in a detail report.
```

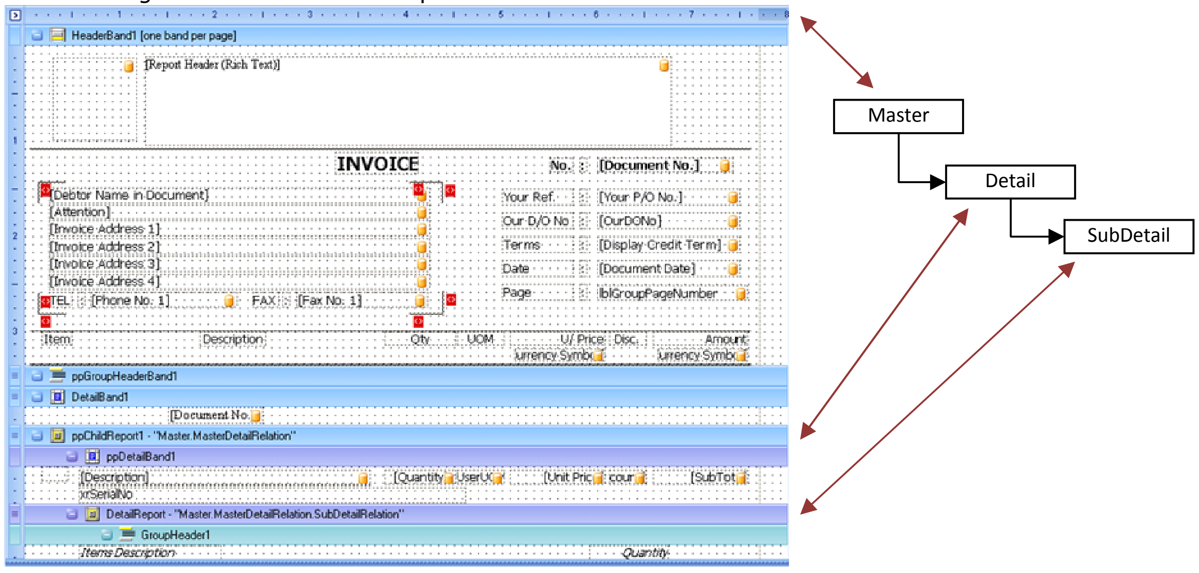
```
//
GetCurrentColumnValue("Categories.CategoriesProducts.ProductName").ToString()
;


// Get the current value of the CategoryName data column in a detail report.
xrLabel2.Text = DetailReport.GetCurrentColumnValue("ProductName").ToString();
```

**Visual Basi
c**

```
' Don't use the GetCurrentRow method in this way in a detail report.
' GetCurrentRow().Row("Categories.CategoriesProducts.ProductName").ToString()


' Get the current value of a data row in a detail report.
XrTableCell1.Text =
DetailReport.GetCurrentRow().Row("ProductName").ToString()


' Don't use the GetCurrentColumnValue method in this way in a detail report.
'
GetCurrentColumnValue("Categories.CategoriesProducts.ProductName").ToString()


' Get the current value of the CategoryName data column in a detail report.
XrTableCell2.Text =
DetailReport.GetCurrentColumnValue("ProductName").ToString()
```

2.2. Example of Master-Detail Report: Invoice

The following illustrates the DetailReportBand and its master-detail tables.



As illustrated in the above diagram. Master table will be the primary datasource of main report, Detail table will be the primary datasource of DetailReportBand ppChildReport1, and SubDetail table will be the primary datasource of DetailReportBand DetailReport.

Master table contains the Invoice’s master fields.

Detail table contains the Invoice’s detail fields.

SubDetail table contains the Invoice’s Item Package detail fields.

3. Scripting

The XtraReports suite provides the ability to use scripts to handle events of either report controls, its bands, or a report itself. This document describes the basic principles of using scripts in XtraReports, lists the main properties required for using scripts, and gives an example on how scripting can be used in a report. Review the Calculating a Custom Summary Using Scripts topic to learn how to calculate a custom summary with scripts.

3.1. Scripting Overview

Scripting is a runtime-only feature of **XtraReports**. It allows you to insert scripts into the code of a report and execute them at runtime. This feature is mostly intended to be used to slightly customize a report. In this instance, it's intended that an you are familiar with one of the scripting languages supported by XtraReports.

The programming language of scripts can be different from that of the language used when creating the report. Though all scripts in the report must be in the same language (a report object only supports scripting in one language at a time), **XtraReports** supports scripting in C#, Visual Basic .NET and JScript .NET. The scripting languages that can be used in **XtraReports** are listed by the ScriptLanguage enumeration values.

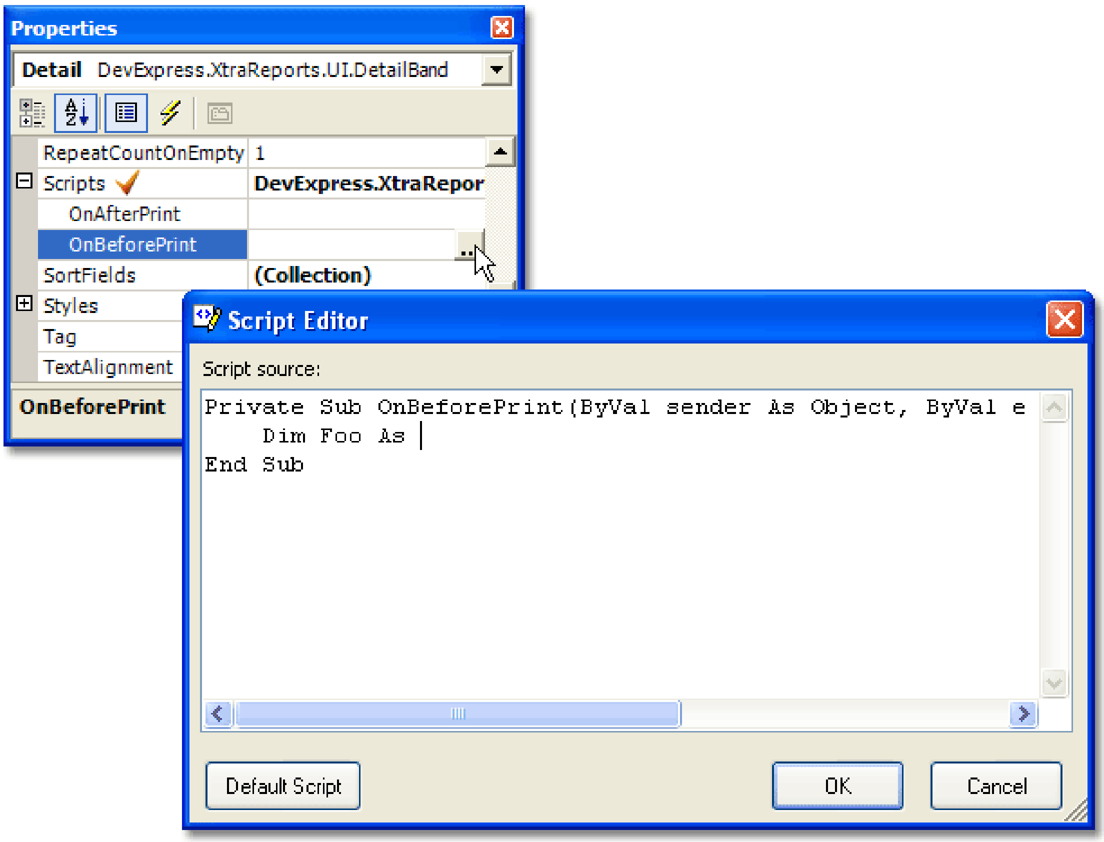
Note

The script language used for a report's scripts has to be supported on the client-side. For instance, since JScript .NET isn't supported in the .NET Framework by default, it needs to be installed on an end-user's machine before scripts in JScipt are executed.

The following is a list of the main properties which are used to implement scripting in **XtraReports**.

Property	**Description**
XtraReport.ScriptLanguage	Specifies the scripting language used to write all the scripts of the XtraReport object. All scripts in a report's object must be in the same language.
XtraReport.ScriptReferences	Specifies the collection of strings that represent the full paths to the assemblies used by the scripts in a report. This property should be used when scripts include references to non-standard assemblies. A list and description of the standard assemblies is available in the XtraReport.ScriptReferences topic.
XRControl.Scripts	Specifies an object of the XRControlEvents class which contains the scripts used for all the XRControl object's events. These events are similar to the events of corresponding descendants of the XRControl class.
XRLabelScripts.OnSummaryReset	Gets or sets the script which handles the XRLabel.SummaryReset event when a custom summary is calculated.
XRLabelScripts.OnSummaryRowChanged	Gets or sets the script which handles the XRLabel.SummaryRowChanged event when a custom summary is calculated.
XRLabelScripts.OnSummaryGetResult	Gets or sets the script which handles the XRLabel.SummaryGetResult event when a custom summary is calculated.

To create any scripting in XtraReports, it's necessary to assign particular scripts to the appropriate properties of an object returned by the XRControl.Scripts. When you start editing any of the script properties at design time, the **Script Editor** window is invoked. If there isn't any particular script defined for this property, this window will contain a code template written in the language defined by the XtraReport.ScriptLanguage property.



```
Private Sub OnBeforePrint(ByVal sender As Object, ByVal e
    Dim Foo As
End Sub
```

The entered script will be used to handle the corresponding event of the control. Scripting events are raised in the same manner as ordinary event handlers. Note, that if an XRControl object has both a script method and a standard event-handler method for the same event, then **both** of them will be executed.

In **XtraReports** scripting is carried out in the following order:

XtraReports generates a temporary class in memory and adds the variables corresponding to the report object, its bands and controls. The names of the variables are defined by the **Name** properties of the objects they represent.

The scripts are preprocessed. While preprocessing, the **using**-like directives are cut from the script code and added to the namespace where the temporary class is defined.

After preprocessing, all the user's scripts are placed in the code of the temporary class, just like text. Then the resulting class is compiled in memory and when required its methods are called to execute the user's scripts.

Note

Since the scripting code placed into the temporary class can contain any code except for the **using**-like directives, it offers a lot of possibilities: you can declare classes (they will become inner classes), declare new variables, methods, classes, etc. The advantage of this approach is that a variable, for instance, declared in one script, can be accessed in another script as it is, in fact, a variable of the temporary class.

3.2. Example. Using Scripts

The following example demonstrates how scripts can be used in **XtraReports**. It represents the script methods used to handle the events of the report's Detail band (changing the table cells color), and events of the label (to calculate a minimum value for the data column as a custom summary). Note that for this example to work correctly, the application should contain a report object bound to the **Products** table in the Northwind database (**nwind.mdb** located in the demos\data directory where you installed **XtraReports**).

These are the scripts used in a report:

C#

```
   // === Detail.Scripts.OnBeforePrint ===
   private void OnBeforePrint(object sender,
System.Drawing.Printing.PrintEventArgs e) {
      XRTableCell[] cells = new XRTableCell[] { pidCell, productNameCell,
productPriceCell };
      System.Decimal price =
(System.Decimal)GetCurrentColumnValue("UnitPrice");
      if (price < 20)
         ChangeCellsColor(cells, Color.Red);
      else if (price > 60)
         ChangeCellsColor(cells, Color.Green);
      else
         ChangeCellsColor(cells, Color.Black);
   }
   void ChangeCellsColor(XRTableCell[] cells, Color color) {
      int count = cells.Length;
      for (int i = 0; i < count; i++)
         cells[i].ForeColor = color;
   }
   // === Detail.Scripts.OnBeforePrint ===
   // === xrLabel1.Scripts.OnSummaryReset ===
   using MyAssembly;
   System.Decimal minPrice = System.Decimal.MaxValue;

   private void OnSummaryReset(object sender, System.EventArgs e) {
      minPrice = System.Decimal.MaxValue;
   }
   // === xrLabel1.Scripts.OnSummaryReset ===

   // === xrLabel1.Scripts.OnSummaryRowChanged ===
   private void OnSummaryRowChanged(object sender, System.EventArgs e) {
       minPrice = Math.Min(minPrice,
(System.Decimal)GetCurrentColumnValue("UnitPrice"));
   }
   // === xrLabel1.Scripts.OnSummaryRowChanged ===

   // === xrLabel1.Scripts.OnSummaryGetResult ===
   private void OnSummaryGetResult(object sender,
DevExpress.XtraReports.UI.SummaryGetResultEventArgs e) {
       e.Result = minPrice;
       e.Handled = true;
   }
   // === xrLabel1.Scripts.OnSummaryGetResult ===
```

Visual Basic

```
   ' === Detail.Scripts.OnBeforePrint ===
   Private Sub OnBeforePrint(sender As Object, e As
System.Drawing.Printing.PrintEventArgs)
      Dim cells() As XRTableCell = {pidCell, productNameCell,
productPriceCell}
      Dim price As System.Decimal = GetCurrentColumnValue("UnitPrice")
      If price < 20 Then
         ChangeCellsColor(cells, Color.Red)
      Else
         If price > 60 Then
            ChangeCellsColor(cells, Color.Green)
         Else
            ChangeCellsColor(cells, Color.Black)
         End If
      End If
   End Sub

   Sub ChangeCellsColor(cells() As XRTableCell, color As Color)
      Dim count As Integer = cells.Length
      Dim i As Integer
      For i = 0 To count - 1
         cells(i).ForeColor = color
      Next i
   End Sub
   ' === Detail.Scripts.OnBeforePrint ===



   ' === xrLabel1.Scripts.OnSummaryReset ===
   Imports MyAssembly

   Dim minPrice As System.Decimal = System.Decimal.MaxValue

   Private Sub OnSummaryReset(ByVal sender As Object, ByVal e As
System.EventArgs)
      minPrice = System.Decimal.MaxValue
   End Sub
   ' === xrLabel1.Scripts.OnSummaryReset ===
   ' === xrLabel1.Scripts.OnSummaryRowChanged ===
   Private Sub OnSummaryRowChanged(ByVal sender As Object, ByVal e As
System.EventArgs)
       minPrice = Math.Min(minPrice, GetCurrentColumnValue("UnitPrice"))
   End Sub
   ' === xrLabel1.Scripts.OnSummaryRowChanged ===

   ' === xrLabel1.Scripts.OnSummaryGetResult ===
   Private Sub OnSummaryGetResult(ByVal sender As Object, ByVal e As
DevExpress.XtraReports.UI.SummaryGetResultEventArgs)
       e.Result = minPrice
       e.Handled = True
   End Sub
   ' === xrLabel1.Scripts.OnSummaryGetResult ===
```

The resulting code will be a class generated in memory which will contain the following code. Note that this code is for internal use only, and it only demonstrates the main principles of using scripts in **XtraReports**.

C#

```
namespace AutogeneratedNamespace
{
   using System;
   using System.Collections;
   using System.Drawing;
   using DevExpress.Data;
   using DevExpress.Utils;
   using DevExpress.XtraPrinting;
   using DevExpress.XtraReports;
   using DevExpress.XtraReports.UI;

   using MyAssembly;
   // other usings

   class AutogeneratedClass
   {
      private XRTableCell pidCell;
      private XRTableCell productNameCell;
      private XRTableCell productPriceCell;
      private XtraReport XtraReport1;
      // other variables

      private void DetailOnBeforePrint(object sender,
System.Drawing.Printing.PrintEventArgs e) {
            XRTableCell[] cells = new XRTableCell[] { pidCell,
productNameCell, productPriceCell };
            System.Decimal price =
(System.Decimal)GetCurrentColumnValue("UnitPrice");
            if (price < 20)
               ChangeCellsColor(cells, Color.Red);
            else if (price > 60)
               ChangeCellsColor(cells, Color.Green);
            else
               ChangeCellsColor(cells, Color.Black);
         }

         void ChangeCellsColor(XRTableCell[] cells, Color color) {
            int count = cells.Length;
            for (int i = 0; i < count; i++)
               cells[i].ForeColor = color;
         }

         System.Decimal minPrice = System.Decimal.MaxValue;

         private void xrLabel1OnSummaryReset(object sender,
System.EventArgs e) {
            minPrice = System.Decimal.MaxValue;
         }

         private void xrLabel1OnSummaryRowChanged(object sender,
System.EventArgs e) {
            minPrice = Math.Min(minPrice,
(System.Decimal)GetCurrentColumnValue("UnitPrice"));
         }

         private void xrLabel1OnSummaryGetResult(object sender,
DevExpress.XtraReports.UI.SummaryGetResultEventArgs e) {
            e.Result = minPrice;
            e.Handled = true;
         }
      }
   }
```

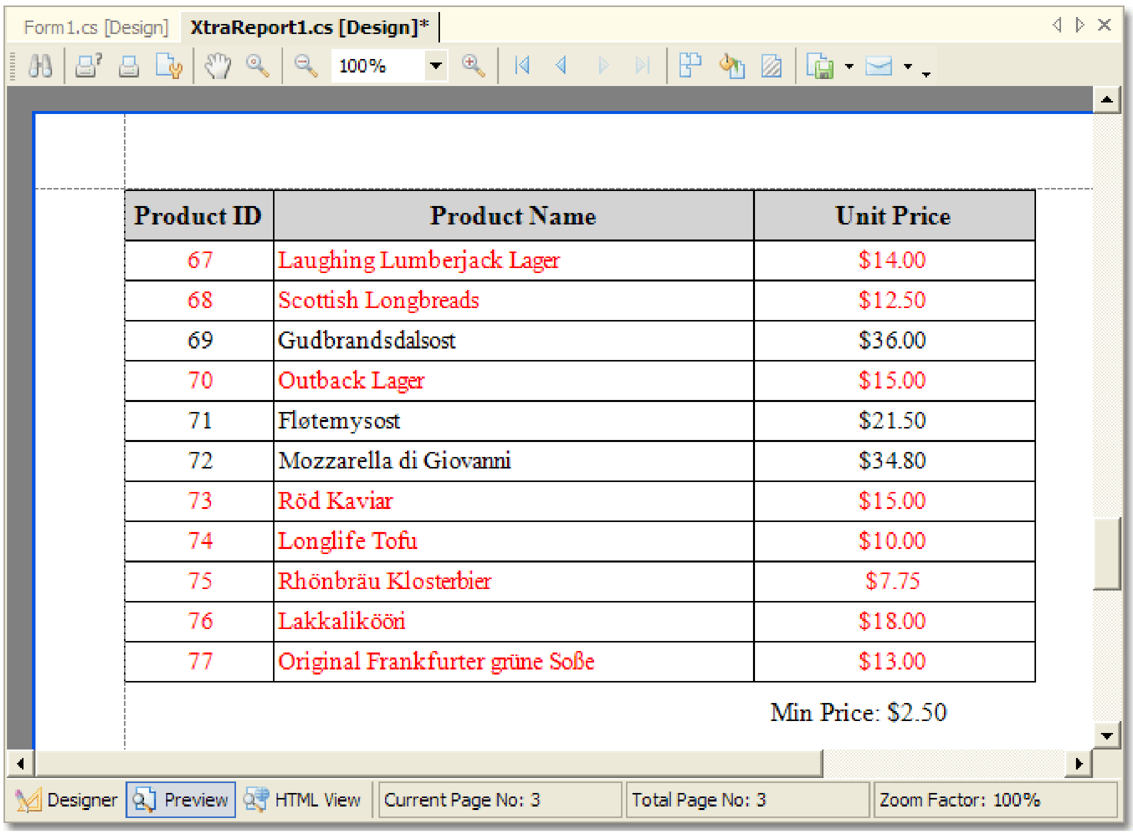
Visual Basic

```
Namespace AutogeneratedNamespace

   Imports System
   Imports System.Collections
```

```
      Imports System.Drawing
      Imports DevExpress.Data
      Imports DevExpress.Utils
      Imports DevExpress.XtraPrinting
      Imports DevExpress.XtraReports
      Imports DevExpress.XtraReports.UI

      Imports MyAssembly
      ' other usings

      Class AutogeneratedClass
         Private pidCell As XRTableCell
         Private productNameCell As XRTableCell
         Private productPriceCell As XRTableCell
         Private XtraReport1 As XtraReport

         ' other variables
         Private Sub DetailOnBeforePrint(sender As Object, e As
System.Drawing.Printing.PrintEventArgs)
            Dim cells() As XRTableCell = {pidCell, productNameCell,
productPriceCell}
            Dim price As System.Decimal =
GetCurrentColumnValue("UnitPrice")
            If price < 20 Then
               ChangeCellsColor(cells, Color.Red)
            Else
               If price > 60 Then
                  ChangeCellsColor(cells, Color.Green)
               Else
                  ChangeCellsColor(cells, Color.Black)
               End If
            End If
         End Sub

         Sub ChangeCellsColor(cells() As XRTableCell, color As Color)
            Dim count As Integer = cells.Length
            Dim i As Integer
            For i = 0 To count - 1
               cells(i).ForeColor = color
            Next i
         End Sub

         Private minPrice As System.Decimal = System.Decimal.MaxValue
```

```
        Private Sub xrLabel1OnSummaryReset(ByVal sender As Object, ByVal
e As System.EventArgs)
            minPrice = System.Decimal.MaxValue
        End Sub

        Private Sub xrLabel1OnSummaryRowChanged(ByVal sender As Object,
ByVal e As System.EventArgs)
            minPrice = Math.Min(minPrice,
GetCurrentColumnValue("UnitPrice"))
        End Sub

        Private Sub xrLabel1OnSummaryGetResult(ByVal sender As Object,
ByVal e As DevExpress.XtraReports.UI.SummaryGetResultEventArgs)
            e.Result = minPrice
            e.Handled = True
        End Sub
    End Class
End Namespace
```

The following image demonstrates the resulting report. Note that the scripts are processed for both runtime and design-time previews.



Product ID	Product Name	Unit Price
67	Laughing Lumberjack Lager	$14.00
68	Scottish Longbreads	$12.50
69	Gudbrandsdalsost	$36.00
70	Outback Lager	$15.00
71	Fløtemysost	$21.50
72	Mozzarella di Giovanni	$34.80
73	Röd Kaviar	$15.00
74	Longlife Tofu	$10.00
75	Rhönbräu Klosterbier	$7.75
76	Lakkalikööri	$18.00
77	Original Frankfurter grüne Soße	$13.00

Min Price: $2.50

3.3. GetCurrentColumnValue Method

XtraReportBase.GetCurrentColumnValue Method

Gets the current value of the specified column in the **primary datasource**.

⊟ Syntax

Visual Basic

```
Public Function GetCurrentColumnValue(
   ByVal columnName As String
) As Object
```

C#

```
public object GetCurrentColumnValue(
   string columnName
);
```

Parameters

columnName

A System.String containing the name of the specified column.

Return Value

An object which represents the current value of the specified column in the **primary datasource**.

⊟ Remarks

If the specified column was not found this method returns **null** .The report's **primary datasource** is determined by the DataSource property.

⊟ Example

This example demonstrates how the GetCurrentRow and GetCurrentColumnValue methods should be used for both master and detail reports. Note that for this example to work properly, the report should be bound to the **Categories** and **Products** tables in the demo Northwind database (**nwind.mdb** located in the demos\data directory where you installed **XtraReports**), and the dataset for these tables should contain the **CategoriesProducts** master-detail relationship. Then, a report should be configured as a master-detail report.

Note

In the code below, "DetailReport" is the name of the DetailReportBand instance, which should be created when creating a master-detail report. This name may be different in another application (for instance, it may be set to "detailReportBand1").

C#

```
using System.Data;
using System.Drawing.Printing;
// ...
```

```
   private void Detail_BeforePrint(object sender, PrintEventArgs e) {
      // Get the value of the current row in the master report.
      xrLabel1.Text =
((DataRowView)GetCurrentRow()).Row["CategoryName"].ToString();

      // Get the value of the current cell in the CategoryName column in
the master report.
      xrLabel2.Text = GetCurrentColumnValue("CategoryName").ToString();
   }

   private void Detail1_BeforePrint(object sender, PrintEventArgs e) {
      // You shouldn't use the GetCurrentRow method in this way in a
detail report.
      //
((DataRowView)GetCurrentRow()).Row["Categories.CategoriesProducts.ProductN
ame"].ToString();

      // Get the value of the current row in the detail report.
      xrLabel3.Text =
((DataRowView)DetailReport.GetCurrentRow()).Row["ProductName"].ToString();

      // You shouldn't use the GetCurrentColumnValue method in this way
in a detail report.
      //
GetCurrentColumnValue("Categories.CategoriesProducts.ProductName").ToStrin
g();

      // Get the current value of the CategoryName data column in a
detail report.
      xrLabel4.Text =
DetailReport.GetCurrentColumnValue("ProductName").ToString();
   }
```

Visual Basic

```
   Imports System.Data
   Imports System.Drawing.Printing
   ' ...

   Private Sub Detail_BeforePrint(ByVal sender As Object, ByVal e As
PrintEventArgs) Handles Detail.BeforePrint
      ' Get the value of the current row in the master report.
      XrTableCell1.Text = GetCurrentRow().Row("CategoryName").ToString()

     ' Get the value of the current cell in the CategoryName column in
the master report.
      XrTableCell2.Text =
GetCurrentColumnValue("CategoryName").ToString()
```

```
    End Sub


    Private Sub Detail1_BeforePrint(ByVal sender As Object, ByVal e As
PrintEventArgs) Handles Detail1.BeforePrint
        ' You shouldn't use the GetCurrentRow method in this way in a
detail report.
        '
GetCurrentRow().Row("Categories.CategoriesProducts.ProductName").ToString(
)


        ' Get the value of the current row in the detail report.
        XrTableCell3.Text =
DetailReport.GetCurrentRow().Row("ProductName").ToString()


        ' You shouldn't use the GetCurrentColumnValue method in this way in
a detail report.
        '
GetCurrentColumnValue("Categories.CategoriesProducts.ProductName").ToStrin
g()


        ' Get the value of the current cell in the CategoryName column in
the detail report.
        XrTableCell4.Text =
DetailReport.GetCurrentColumnValue("ProductName").ToString()
    End Sub
```

3.4. GetCurrentRow Method

XtraReportBase.GetCurrentRow Method

Gets the current row in the **primary datasource**.

Syntax

Visual Basic

```
Public Function GetCurrentRow() As Object
```

C#

```
public object GetCurrentRow();
```

Return Value

If the **primary datasource** is represented by a System.Data.DataTable object the return value will be an object of the System.Data.DataRowView class. If it's represented by an object implementing the System.Collections.IList interface the return value will be an item from the collection represented by this object.

Remarks

The report's **primary datasource** is determined by the DataSource property.

Example

This example demonstrates how the GetCurrentRow and GetCurrentColumnValue methods should be used for both master and detail reports. Note that for this example to work properly, the report should be bound to the **Categories** and **Products** tables in the demo Northwind database (**nwind.mdb** located in the demos\data directory where you installed **XtraReports**), and the dataset for these tables should contain the **CategoriesProducts** master-detail relationship. Then, a report should be configured as a master-detail report.

Note

In the code below, "DetailReport" is the name of the DetailReportBand instance, which should be created when creating a master-detail report. This name may be different in another application (for instance, it may be set to "detailReportBand1").

C#

```
    using System.Data;
    using System.Drawing.Printing;
    // ...

    private void Detail_BeforePrint(object sender, PrintEventArgs e) {
       // Get the value of the current row in the master report.
       xrLabel1.Text =
((DataRowView)GetCurrentRow()).Row["CategoryName"].ToString();

       // Get the value of the current cell in the CategoryName column in
the master report.
```

```
        xrLabel2.Text = GetCurrentColumnValue("CategoryName").ToString();
    }


    private void Detail1_BeforePrint(object sender, PrintEventArgs e) {
        // You shouldn't use the GetCurrentRow method in this way in a
detail report.
        //
((DataRowView)GetCurrentRow()).Row["Categories.CategoriesProducts.ProductN
ame"].ToString();


        // Get the value of the current row in the detail report.
        xrLabel3.Text =
((DataRowView)DetailReport.GetCurrentRow()).Row["ProductName"].ToString();


        // You shouldn't use the GetCurrentColumnValue method in this way in
a detail report.
        //
GetCurrentColumnValue("Categories.CategoriesProducts.ProductName").ToStrin
g();


        // Get the current value of the CategoryName data column in a detail
report.
        xrLabel4.Text =
DetailReport.GetCurrentColumnValue("ProductName").ToString();
    }
```

Visual Basic

```
    Imports System.Data
    Imports System.Drawing.Printing
    ' ...


    Private Sub Detail_BeforePrint(ByVal sender As Object, ByVal e As
PrintEventArgs) Handles Detail.BeforePrint
        ' Get the value of the current row in the master report.
        XrTableCell1.Text = GetCurrentRow().Row("CategoryName").ToString()


       ' Get the value of the current cell in the CategoryName column in the
master report.
        XrTableCell2.Text = GetCurrentColumnValue("CategoryName").ToString()
    End Sub


    Private Sub Detail1_BeforePrint(ByVal sender As Object, ByVal e As
PrintEventArgs) Handles Detail1.BeforePrint
        ' You shouldn't use the GetCurrentRow method in this way in a detail
report.
        '
GetCurrentRow().Row("Categories.CategoriesProducts.ProductName").ToString()


        ' Get the value of the current row in the detail report.
        XrTableCell3.Text =
DetailReport.GetCurrentRow().Row("ProductName").ToString()


       ' You shouldn't use the GetCurrentColumnValue method in this way in a
detail report.
       '
GetCurrentColumnValue("Categories.CategoriesProducts.ProductName").ToString()


       ' Get the value of the current cell in the CategoryName column in the
detail report.
       XrTableCell4.Text =
DetailReport.GetCurrentColumnValue("ProductName").ToString()
    End Sub
```

3.5. Useful static methods in BCE.AutoCount.Application class

The following table lists out some of the useful static methods in BCE.AutoCount.Application class.

Method	Description
FormatQuantity	Format a decimal according to system defined quantity decimal.

Sample Code:
`decimal qty = 123;`
`xrLabel.Text = BCE.AutoCount.Application.FormatQuantity(qty);`	

FormatPrice	Format a decimal according to system defined price decimal.

Sample Code:
`decimal price = 123.56;`
`xrLabel.Text = BCE.AutoCount.Application.FormatPrice(price);`	

FormatCurrency	Format a decimal according to system defined currency decimal.

Sample Code:
`decimal subTotal = 133.26;`
`xrLabel.Text = BCE.AutoCount.Application.FormatCurrency(subTotal);`	

RoundQuantity	Round a decimal or object according to system defined quantity decimal.

Sample Code:
`object obj = GetCurrentColumnValue(“Qty”);`
`decimal qty = BCE.AutoCount.Application.RoundQuantity(obj);`	

RoundPrice	Round a decimal or object according to system defined price decimal.

Sample Code:
`object obj = GetCurrentColumnValue(“UnitPrice”);`
`decimal unitPrice = BCE.AutoCount.Application.RoundPrice(obj);`	

RoundCurrency	Round a decimal or object according to system defined currency decimal.

Sample Code:
`object obj = GetCurrentColumnValue(“SubTotal”);`
`decimal subTotal = BCE.AutoCount.Application.RoundCurrency(obj);`	

3.6. Useful data access methods in BCE.AutoCount.Application.DBSetting object

The following table lists out some of the useful methods in BCE.AutoCount.Application.DBSetting object.

3.6.1. ExecuteScalar Method

BCE.AutoCount.Application.DBSetting.ExecuteScalar Method

Executes the query, and returns the first column of the first row in the result set returned by the query. Additional columns or rows are ignored.

Syntax

C#

```
   public object BCE.AutoCount.Application.DBSetting.ExecuteScalar(string
sql, params SqlParameter[] sqlParams));
```

Return Value

The first column of the first row in the result set, or a null reference (**Nothing** in Visual Basic) if the result set is empty.

3.6.2. ExecuteNonQuery Method

BCE.AutoCount.Application.DBSetting.ExecuteNonQuery Method

Executes a Transact-SQL statement against the connection and returns the number of rows affected.

Syntax

C#

```
   public int BCE.AutoCount.Application.DBSetting.ExecuteNonQuery(string
sql, params SqlParameter[] sqlParams));
```

Return Value

The number of rows affected.

3.6.3. GetFirstDataRow Method

BCE.AutoCount.Application.DBSetting.GetFirstDataRow Method

Executes the query, and returns the first row in the result set returned by the query. Additional rows are ignored.

Syntax

C#

```
   public DataRow
BCE.AutoCount.Application.DBSetting.GetFirstDataRow(string sql, params
SqlParameter[] sqlParams));
```

Return Value

The first row of the result set.

3.6.4. GetDataTable Method

BCE.AutoCount.Application.DBSetting.GetDataTable Method

Executes the query, and returns the result set returned by the query in a DataTable object.

Syntax

C#

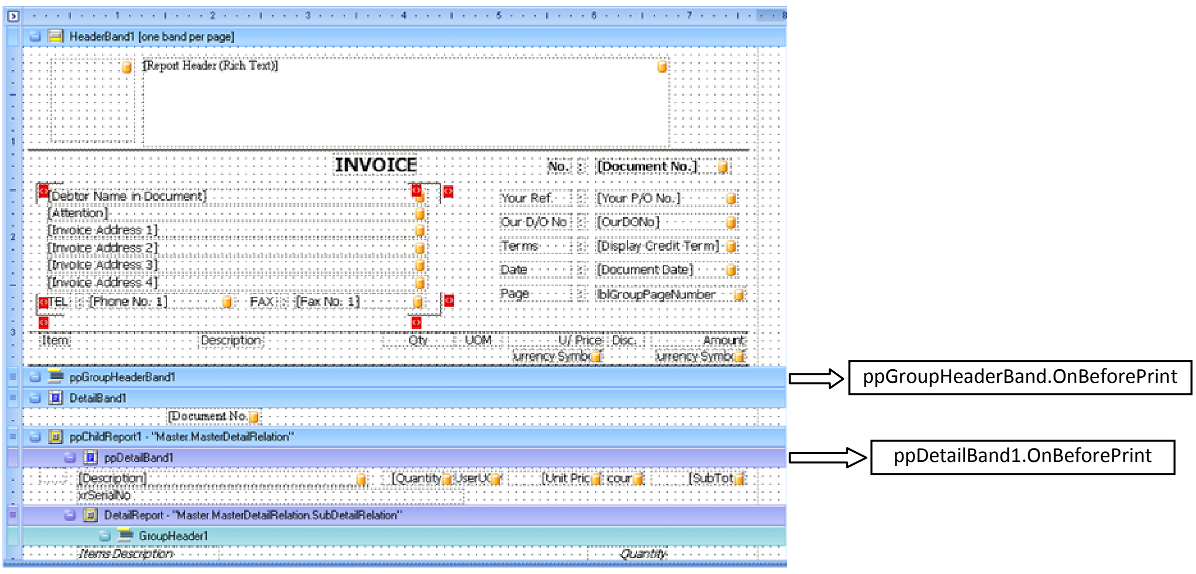
```
   public DataTable
BCE.AutoCount.Application.DBSetting.GetDataTable(string sql, bool
loadSchema, params SqlParameter[] sqlParams));
```

Return Value

A DataTable object.

3.7. Example 1: Using Scripts to show simple Item Count in Invoice

The following example demonstrates how scripts can be used to show simple Item Count in Invoice.



C# ppGroupHeaderBand.OnBeforePrint

```
  int detailCount;

  private void OnBeforePrint(object sender,
System.Drawing.Printing.PrintEventArgs e)

  {

    detailCount = 0;

  }
```

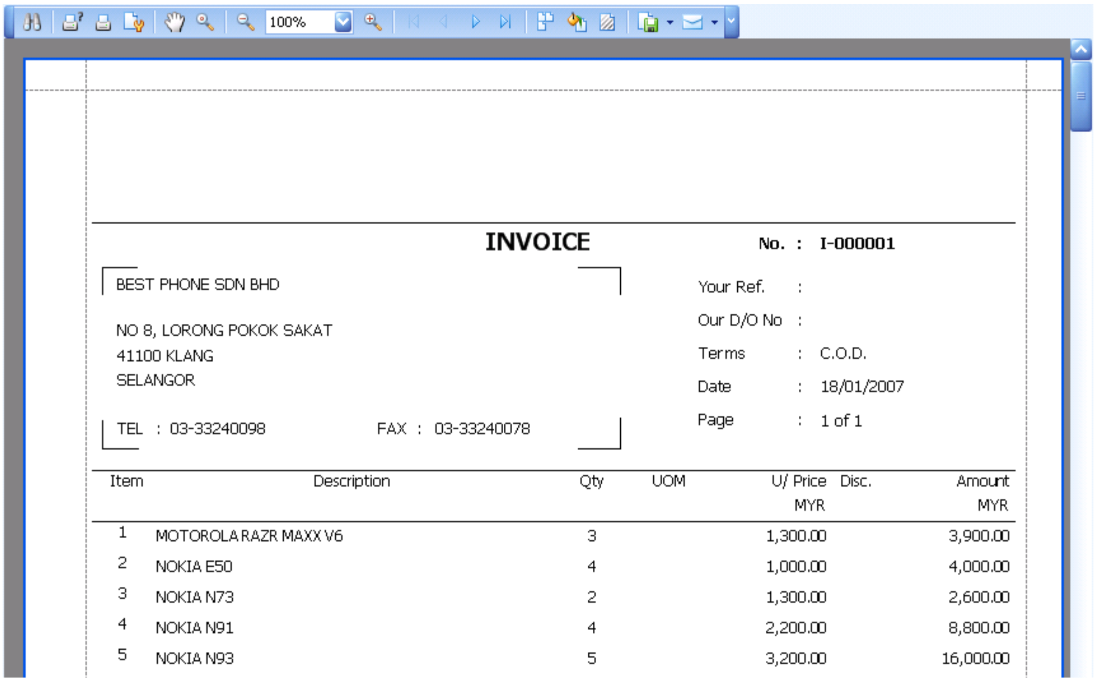
C# ppDetailBand1.OnBeforePrint

```
 private void OnBeforePrint(object sender,
System.Drawing.Printing.PrintEventArgs e)

 {

   detailCount++;

   DBCalc1.Text = detailCount.ToString();

 }
```

In ppGroupHeaderBand.DoBeforePrint event, we declare a variable detailCount, and we reset it to 0 in the OnBeforePrint event, this is necessary because the GroupHeaderBand here is to control different invoices, so when start a new invoice, we must reset the detailCount to 1.

In ppDetailBand1.OnBeforePrint event, we increment the detailCount by 1, and then assign its value to DBCalc1.Text property.

Preview result:



INVOICE

No. :	I-000001
Your Ref. :	
Our D/O No :	
Terms :	C.O.D.
Date :	18/01/2007
Page :	1 of 1

BEST PHONE SDN BHD

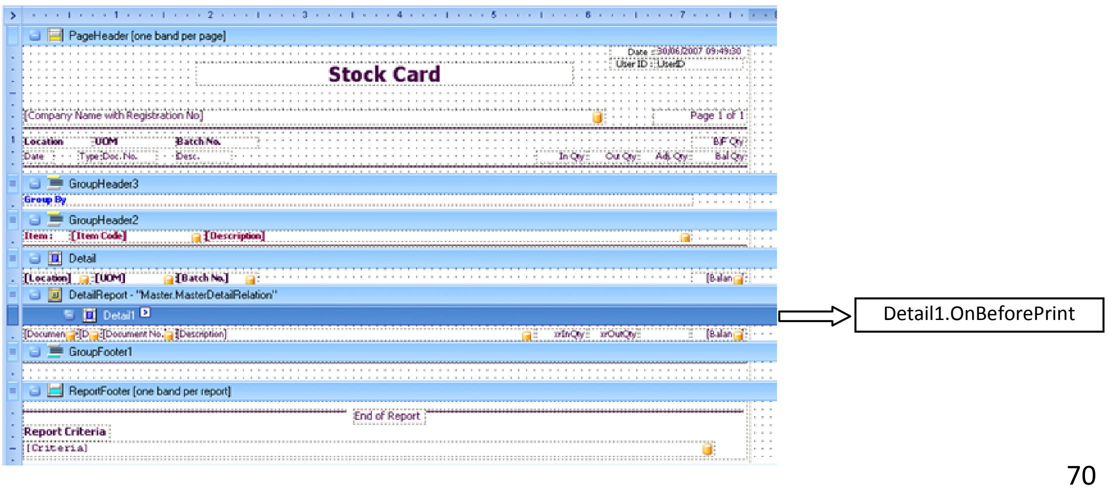
NO 8, LORONG POKOK SAKAT
41100 KLANG
SELANGOR

TEL : 03-33240098 FAX : 03-33240078

Item	Description	Qty	UOM	U/ Price MYR	Disc.	Amount MYR
1	MOTOROLA RAZR MAXX V6	3		1,300.00		3,900.00
2	NOKIA E50	4		1,000.00		4,000.00
3	NOKIA N73	2		1,300.00		2,600.00
4	NOKIA N91	4		2,200.00		8,800.00
5	NOKIA N93	5		3,200.00		16,000.00

3.8. Example 2: Using Scripts to show Stock Adjustment Quantity in a separate column in Stock Card Report

The following example demonstrates how scripts can be used to show Stock Adjustment Quantity in a separate column in Stock Card Report.



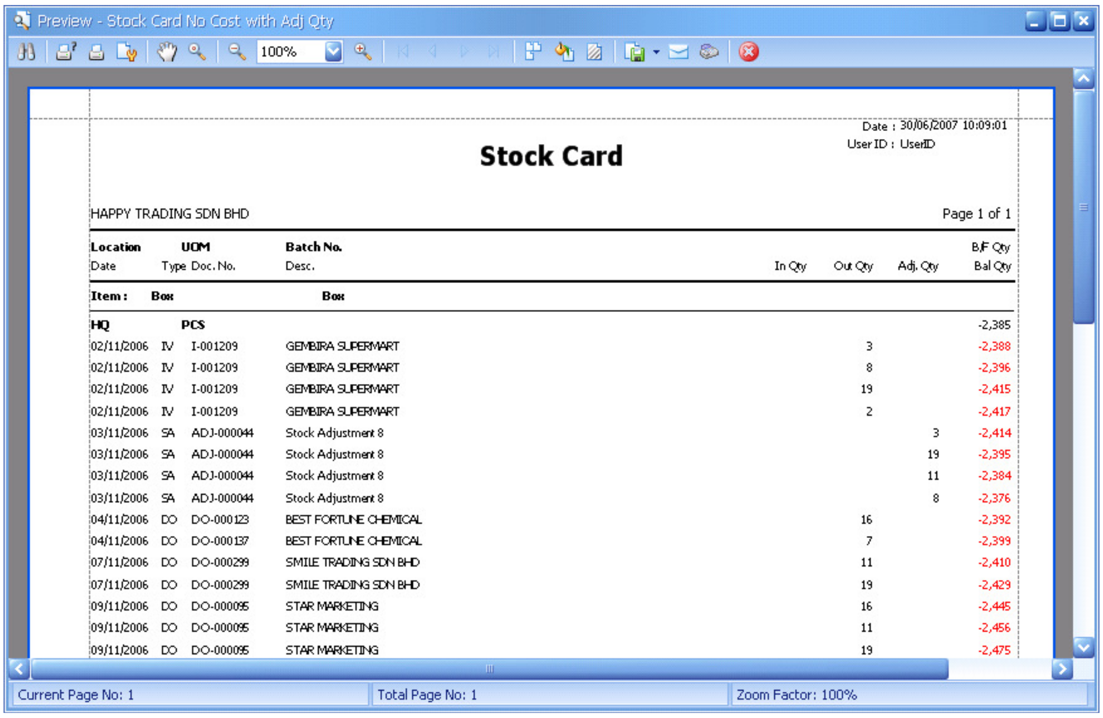
C# Detail1.OnBeforePrint

```
 private void OnBeforePrint(object sender,
System.Drawing.Printing.PrintEventArgs e)
 {
   object obj = DetailReport.GetCurrentColumnValue("DocType");
   if (obj == null) return;
   string docType = obj.ToString();
   decimal inQty =
BCE.AutoCount.Application.RoundQuantity(DetailReport.GetCurrentColumnValue
("InQty"));
   decimal outQty =
BCE.AutoCount.Application.RoundQuantity(DetailReport.GetCurrentColumnValue
("OutQty"));
   decimal inOutQty = inQty + outQty;
   if (docType == "SA")
   {
     xrAdjQty.Text = BCE.AutoCount.Application.FormatQuantity(inOutQty);
     xrInQty.Text = "";
     xrOutQty.Text = "";
   }
   else
   {
     if (inOutQty > 0)
     {
       xrInQty.Text = BCE.AutoCount.Application.FormatQuantity(inOutQty);
       xrOutQty.Text = "";
     }
     else if (inOutQty < 0)
     {
       xrInQty.Text = "";
       xrOutQty.Text = BCE.AutoCount.Application.FormatQuantity(-
inOutQty);
     }
     else
     {
       xrInQty.Text = "";
       xrOutQty.Text = "";
     }
     xrAdjQty.Text = "";
   }
   decimal balQty =
BCE.AutoCount.Application.RoundQuantity(DetailReport.GetCurrentColumnValue
("Balance"));
   if (balQty < 0)
```

```
        xrBalQty.ForeColor = Color.Red;
      else
        xrBalQty.ForeColor = Color.Black;
  }
```

The above script will extract the InOutQty from current DataRow, if current DataRow is of Stock Adjustment Type, then it will assign the InOutQty directly to the xrAdjQty label, otherwise, it will assign the InOutQty to xrInQty or xrOutQty label, depends on the sign of the InOutQty. At the final section of the script, it checks the BalQty of current DataRow, if it is negative, it will set the xrBalQty label text color to red color.

Preview result:


Preview - Stock Card No Cost with Adj Qty

100%

Date : 30/06/2007 10:09:01
User ID : UserID

Stock Card

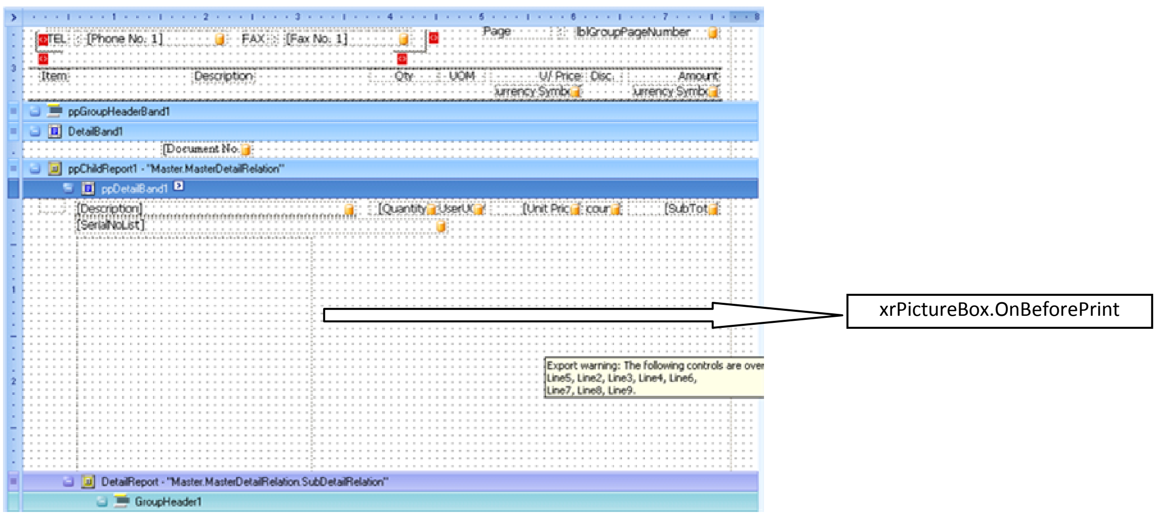
HAPPY TRADING SDN BHD — Page 1 of 1

Location / Date	UOM / Type	Doc. No.	Batch No. / Desc.	In Qty	Out Qty	Adj. Qty	B/F Qty / Bal Qty
Item :	Box		Box				
HQ	PCS						-2,385
02/11/2006	IV	I-001209	GEMBIRA SUPERMART		3		-2,388
02/11/2006	IV	I-001209	GEMBIRA SUPERMART		8		-2,396
02/11/2006	IV	I-001209	GEMBIRA SUPERMART		19		-2,415
02/11/2006	IV	I-001209	GEMBIRA SUPERMART		2		-2,417
03/11/2006	SA	ADJ-000044	Stock Adjustment 8			3	-2,414
03/11/2006	SA	ADJ-000044	Stock Adjustment 8			19	-2,395
03/11/2006	SA	ADJ-000044	Stock Adjustment 8			11	-2,384
03/11/2006	SA	ADJ-000044	Stock Adjustment 8			8	-2,376
04/11/2006	DO	DO-000123	BEST FORTUNE CHEMICAL		16		-2,392
04/11/2006	DO	DO-000137	BEST FORTUNE CHEMICAL		7		-2,399
07/11/2006	DO	DO-000299	SMILE TRADING SDN BHD		11		-2,410
07/11/2006	DO	DO-000299	SMILE TRADING SDN BHD		19		-2,429
09/11/2006	DO	DO-000095	STAR MARKETING		16		-2,445
09/11/2006	DO	DO-000095	STAR MARKETING		11		-2,456
09/11/2006	DO	DO-000095	STAR MARKETING		19		-2,475

Current Page No: 1 | Total Page No: 1 | Zoom Factor: 100%

3.9. Example 3: Using Scripts to show Stock Item Picture in Invoice

The following example demonstrates how scripts can be used to show Stock Item Picture in Invoice Report.

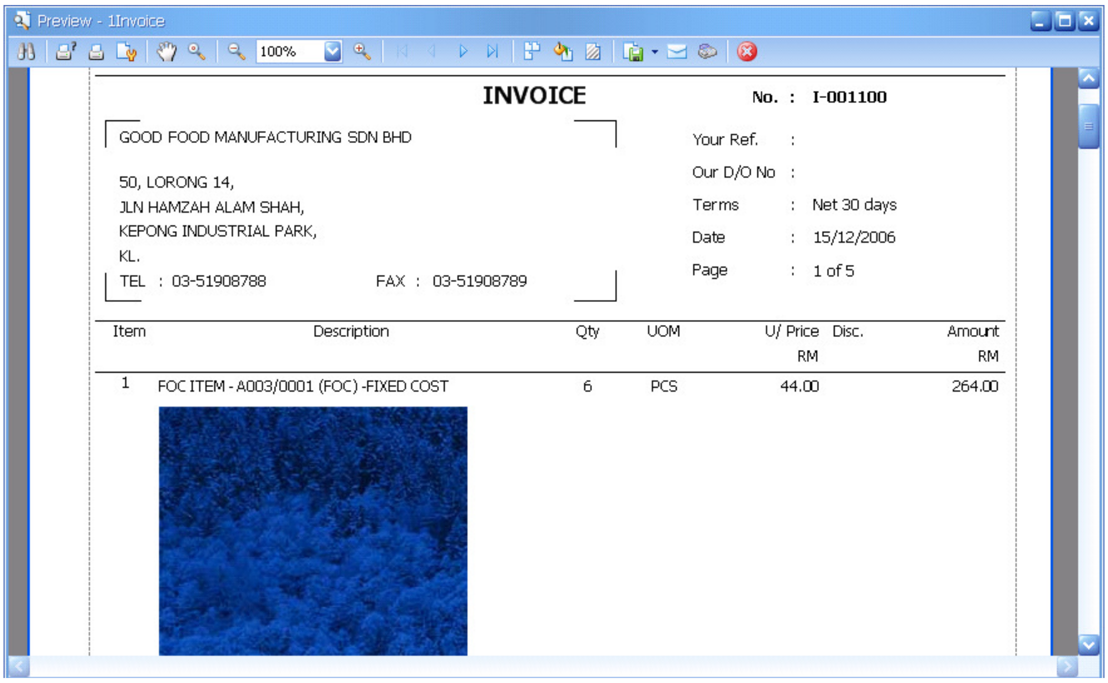


C# xrPictureBox.OnBeforePrint

```
 private void OnBeforePrint(object sender,
System.Drawing.Printing.PrintEventArgs e)
  {
    string imageFileName =
ppChildReport1.GetCurrentColumnValue("ImageFileName").ToString();


    try
    {
      if (imageFileName.Length > 0 &&
System.IO.File.Exists(imageFileName))
      {
        System.Drawing.Image image =
System.Drawing.Image.FromFile(imageFileName);


        xrPictureBox.Size = new System.Drawing.Size(256, 256);
        xrPictureBox.Image = image;
        xrPictureBox.Visible = true;
      }
      else
      {
        xrPictureBox.Visible = false;
        xrPictureBox.Size = new System.Drawing.Size(0, 0);
        ppDetailBand1.Height = 0;
      }
    }
    catch
    {
        xrPictureBox.Visible = false;
        xrPictureBox.Size = new System.Drawing.Size(0, 0);
        ppDetailBand1.Height = 0;
    }
  }
```

The above script will extract the ImageFileName property of Invoice Detail, and check whether the image file is exist or not, if it it exist, then it will load the image file into a System.Drawing.Image object, and then assign this object to xrPictureBox.Image. If the image file is not found or error occurs during loading, it will set the size of the xrPictureBox to 0, and minimize the height of the ppDetailBand1.

Preview result:



INVOICE

No. : I-001100

GOOD FOOD MANUFACTURING SDN BHD

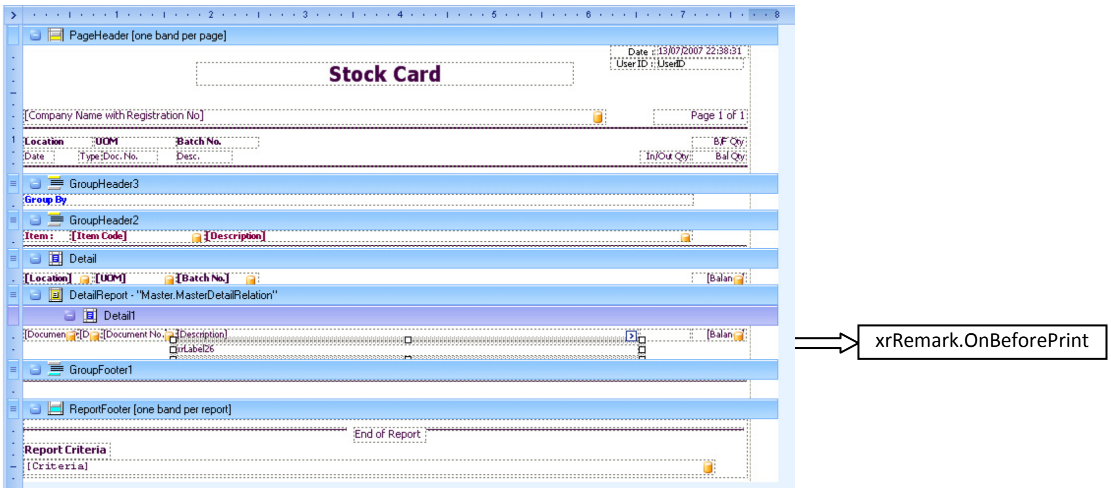
50, LORONG 14,
JLN HAMZAH ALAM SHAH,
KEPONG INDUSTRIAL PARK,
KL.
TEL : 03-51908788 FAX : 03-51908789

Your Ref.	:	
Our D/O No	:	
Terms	:	Net 30 days
Date	:	15/12/2006
Page	:	1 of 5

Item	Description	Qty	UOM	U/ Price RM	Disc.	Amount RM
1	FOC ITEM - A003/0001 (FOC) -FIXED COST	6	PCS	44.00		264.00

3.10. Example 4: Using Scripts to show an UDF from D/O in Stock Card Report

The following example demonstrates how scripts can be used to show an UDF from D/O in Stock Card Report.

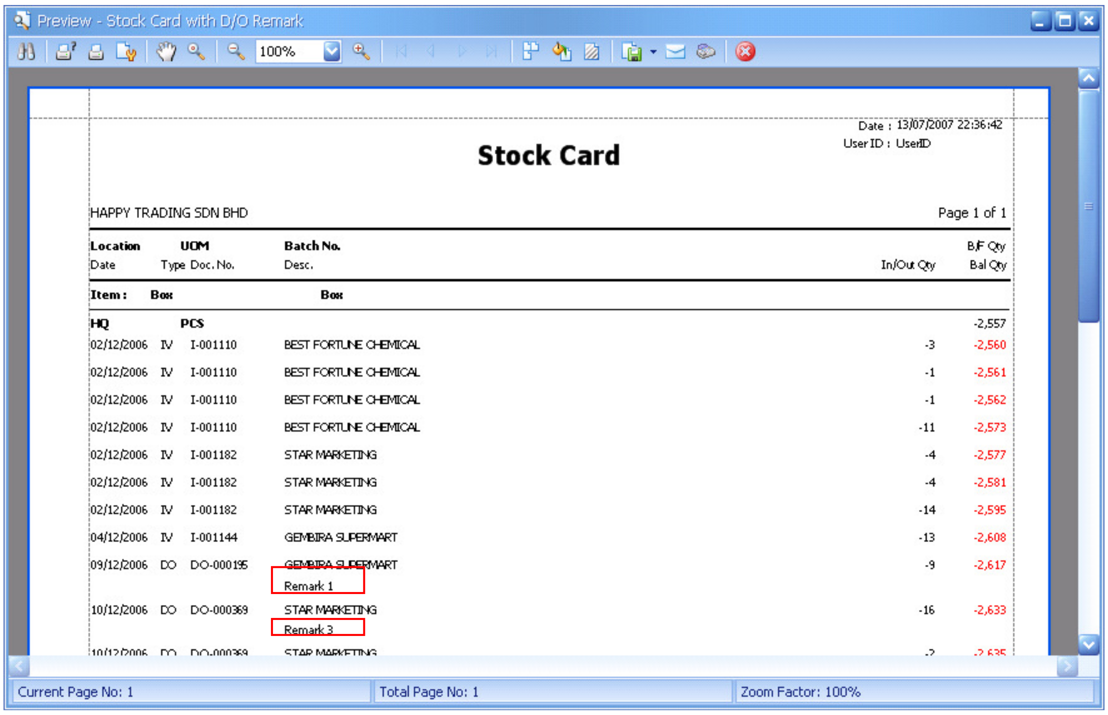


C# Detail1.OnBeforePrint

```
 private void OnBeforePrint(object sender,
System.Drawing.Printing.PrintEventArgs e)
  {
    xrRemark.Text = "";
    object obj = DetailReport.GetCurrentColumnValue("DocType");
    if (obj == null) return;


    string docType = obj.ToString();
    if (docType == "DO")
    {
      obj = DetailReport.GetCurrentColumnValue("DtlKey");
      string dtlKey = obj.ToString();

      obj = BCE.AutoCount.Application.DBSetting.ExecuteScalar("SELECT
UDF_Remark FROM DODTL WHERE DtlKey=" + dtlKey);
      if (obj != null)
        xrRemark.Text = obj.ToString();
    }
  }
```

The above script will get the value of the DocType column from current DataRow, if the value is equal to DO, then it will get the value of the DtlKey column from current DataRow, then it will execute a SQL statement to get the UDF_Remark column from DODTL table, then assign the value to xrRemark.Text property.

Preview result:


Preview - Stock Card with D/O Remark

100%

Stock Card

Date : 13/07/2007 22:36:42
User ID : UserID

HAPPY TRADING SDN BHD — Page 1 of 1

Location		UOM	Batch No.		B/F Qty
Date	Type	Doc. No.	Desc.	In/Out Qty	Bal Qty
Item :	**Box**		**Box**		
HQ		**PCS**			-2,557
02/12/2006	IV	I-001110	BEST FORTUNE CHEMICAL	-3	-2,560
02/12/2006	IV	I-001110	BEST FORTUNE CHEMICAL	-1	-2,561
02/12/2006	IV	I-001110	BEST FORTUNE CHEMICAL	-1	-2,562
02/12/2006	IV	I-001110	BEST FORTUNE CHEMICAL	-11	-2,573
02/12/2006	IV	I-001182	STAR MARKETING	-4	-2,577
02/12/2006	IV	I-001182	STAR MARKETING	-4	-2,581
02/12/2006	IV	I-001182	STAR MARKETING	-14	-2,595
04/12/2006	IV	I-001144	GEMBIRA SUPERMART	-13	-2,608
09/12/2006	DO	DO-000195	GEMBIRA SUPERMART Remark 1	-9	-2,617
10/12/2006	DO	DO-000369	STAR MARKETING Remark 3	-16	-2,633
10/12/2006	DO	DO-000369	STAR MARKETING	-2	-2,635

Current Page No: 1 | Total Page No: 1 | Zoom Factor: 100%

See the red box value.

4. How To

4.1. How to make Panasonic KX-P1121 printer able to print to the margin area?

Just install Epson LQ850 printer driver will do. Epson LQ850 printer driver is fully compatible with Panasonic KX-P1121 printer.

4.2. How to make Panasonic KX-P3624 printer able to print to the margin area?

Just install Epson LQ2500 printer driver will do. Epson LQ2500 printer driver is fully compatible with Panasonic KX-P3624 printer.

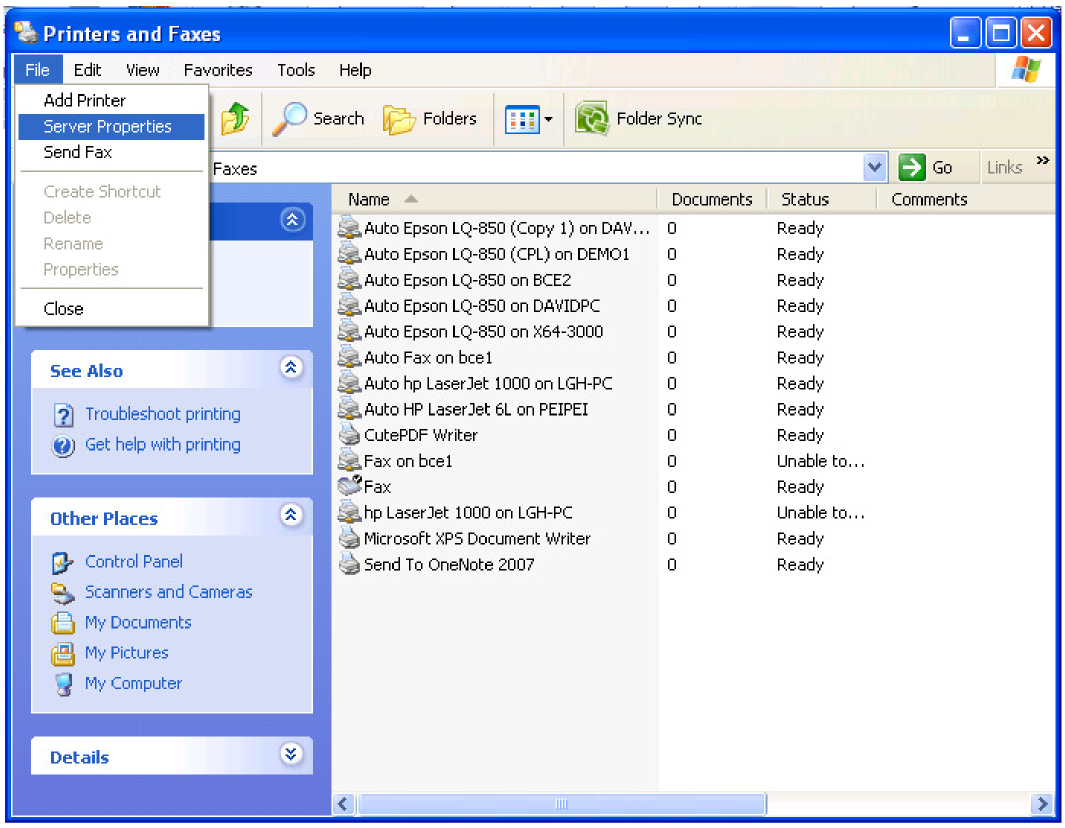
4.3. How to print custom paper size report?

To print a custom paper size report such as half letter size payment voucher or receipt voucher, it requires a bit trick, please complete the following 3 steps.

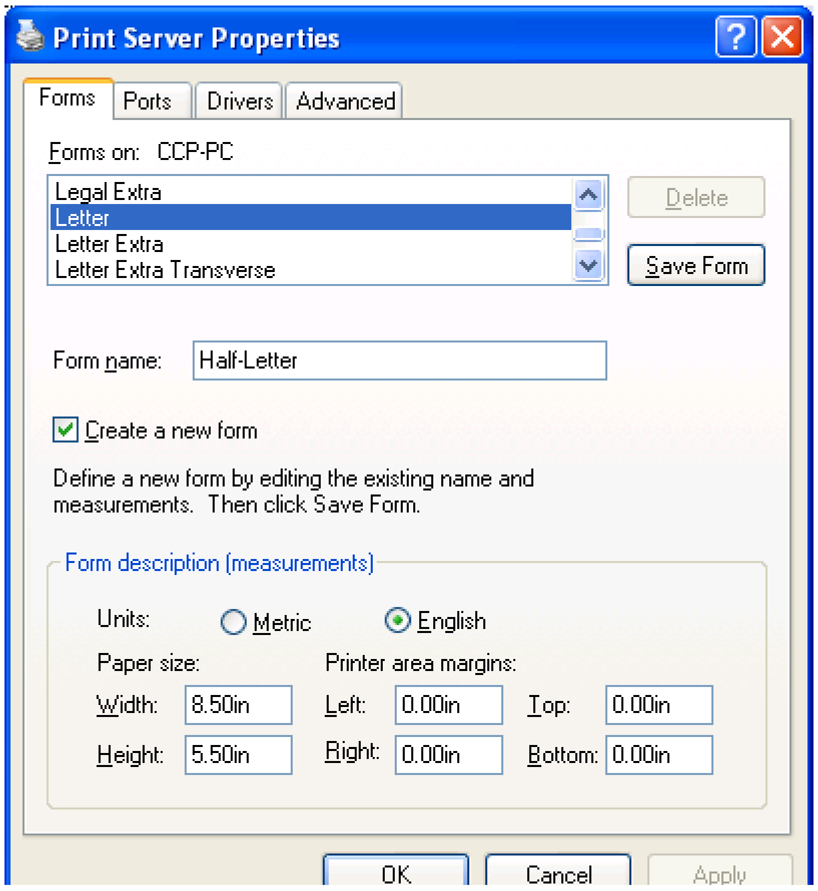
4.3.1. Create a custom Report Form

To create a custom report form, please follow the steps below:

1. Open windows Report and Faxes folder, then choose File | Server Properties.



Name	Documents	Status	Comments
Auto Epson LQ-850 (Copy 1) on DAV...	0	Ready	
Auto Epson LQ-850 (CPL) on DEMO1	0	Ready	
Auto Epson LQ-850 on BCE2	0	Ready	
Auto Epson LQ-850 on DAVIDPC	0	Ready	
Auto Epson LQ-850 on X64-3000	0	Ready	
Auto Fax on bce1	0	Ready	
Auto hp LaserJet 1000 on LGH-PC	0	Ready	
Auto HP LaserJet 6L on PEIPEI	0	Ready	
CutePDF Writer	0	Ready	
Fax on bce1	0	Unable to...	
Fax	0	Ready	
hp LaserJet 1000 on LGH-PC	0	Unable to...	
Microsoft XPS Document Writer	0	Ready	
Send To OneNote 2007	0	Ready	



Forms on: CCP-PC

Legal Extra
Letter
Letter Extra
Letter Extra Transverse

Form name: Half-Letter

[✓] Create a new form

Define a new form by editing the existing name and measurements. Then click Save Form.

Form description (measurements)

Units: Metric [● English]

Paper size:
Width: 8.50in
Height: 5.50in

Printer area margins:
Left: 0.00in
Right: 0.00in
Top: 0.00in
Bottom: 0.00in

2. Then click on Create a new form.

3. Then enter your Form name, and Form measurements, like Width and Height.

4. Then click Save Form to save it.

4.3.2. Modify AutoCount 2006 Report Paper Size

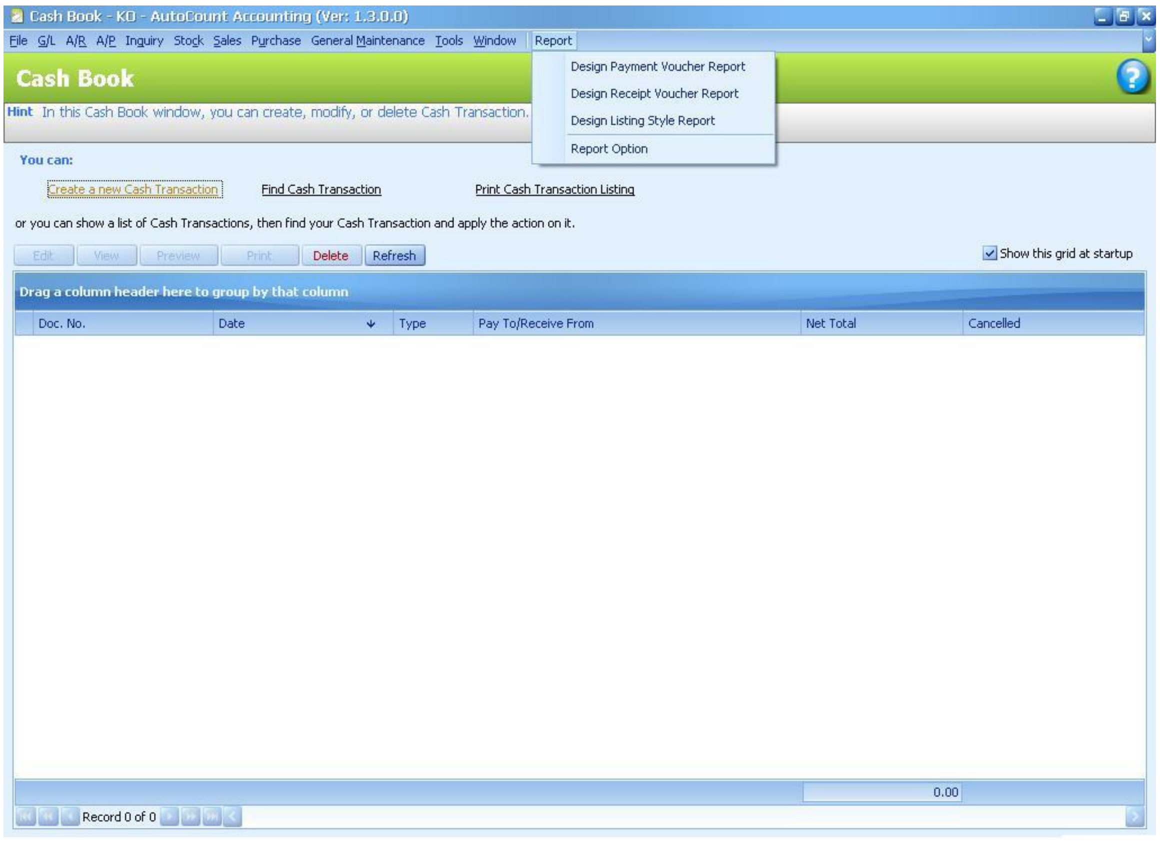
To create a custom report form, please follow the steps below:

1. Please refer to 2.7.4 on how to set your report as a custom paper size. The paper size must same as defined in previous section.

4.3.3. Set Custom Paper Name in Report Option

The last step is to set the Custom Paper Name in Report Option to the Form Name as defined in 5.1.1.

1. Go to your report form, click on the Report menu, choose Report Option.


Cash Book - KO - AutoCount Accounting (Ver: 1.3.0.0)
File G/L A/R A/P Inquiry Stock Sales Purchase General Maintenance Tools Window Report
Design Payment Voucher Report
Design Receipt Voucher Report
Design Listing Style Report
Report Option
Cash Book
Create a new Cash Transaction
Find Cash Transaction
Print Cash Transaction Listing
Edit
View
Preview
Print
Delete
Refresh
Show this grid at startup
Drag a column header here to group by that column
Doc. No.
Date
Type
Pay To/Receive From
Net Total
Cancelled
Record 0 of 0

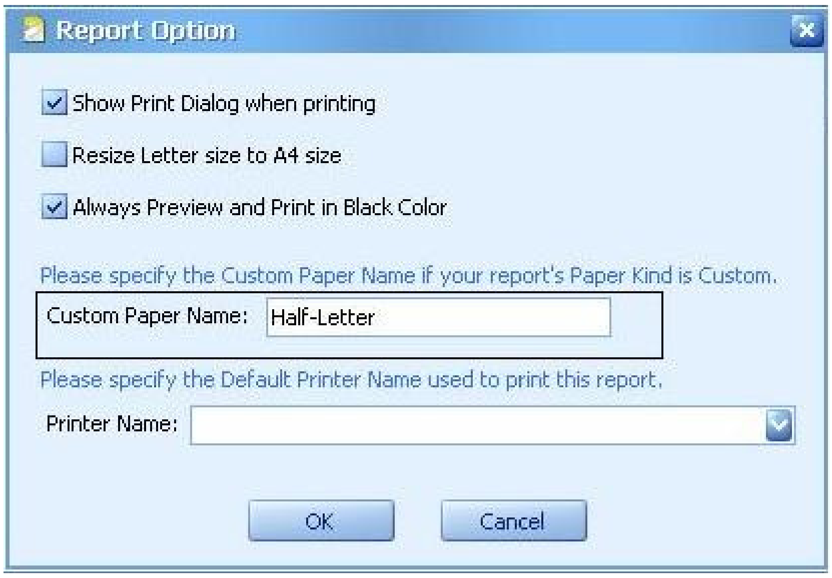
Hint In this Cash Book window, you can create, modify, or delete Cash Transaction.

You can:

or you can show a list of Cash Transactions, then find your Cash Transaction and apply the action on it.

0.00

2. In the Report Option window, enter your Custom Paper Name, in the above example, it is **Half-Letter**.



☑ Show Print Dialog when printing

☐ Resize Letter size to A4 size

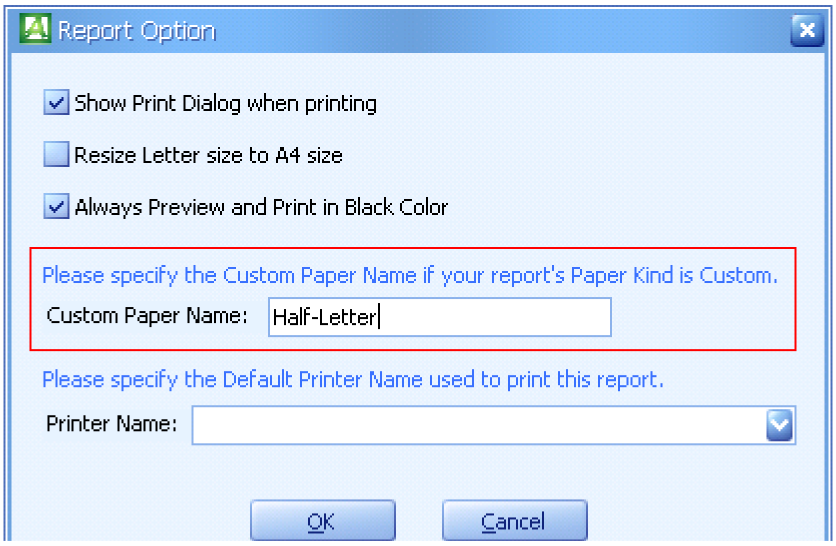
☑ Always Preview and Print in Black Color

Please specify the Custom Paper Name if your report's Paper Kind is Custom.

Custom Paper Name: Half-Letter

Please specify the Default Printer Name used to print this report.

Printer Name:



Report Option

[✓] Show Print Dialog when printing

[ ] Resize Letter size to A4 size

[✓] Always Preview and Print in Black Color

Please specify the Custom Paper Name if your report's Paper Kind is Custom.

Custom Paper Name: Half-Letter

Please specify the Default Printer Name used to print this report.

Printer Name: