ComponentOne

FlexReport for WinForms

ComponentOne, a division of GrapeCity
201 South Highland Avenue, Third Floor
Pittsburgh, PA 15206 USA

Website: http://www.componentone.com
Sales: sales@componentone.com
Telephone: 1.800.858.2739 or 1.412.681.4343 (Pittsburgh, PA USA Office)

Trademarks

Warranty

ComponentOne warrants that the media on which the software is delivered is free from defects in material and workmanship, assuming normal use, for a period of 90 days from the date of purchase. If a defect occurs during this time, you may return the defective media to ComponentOne, along with a dated proof of purchase, and ComponentOne will replace it at no charge. After 90 days, you can obtain a replacement for the defective media by sending it and a check for $2 5 (to cover postage and handling) to ComponentOne.

Except for the express warranty of the original media on which the software is delivered is set forth here, ComponentOne makes no other warranties, express or implied. Every attempt has been made to ensure that the information contained in this manual is correct as of the time it was written. ComponentOne is not responsible for any errors or omissions. ComponentOne's liability is limited to the amount you paid for the product. ComponentOne is not liable for any special, consequential, or other damages for any reason.

Copying and Distribution

Table of Contents

FlexReport for WinForms Overview

ComponentOne Studio introduces **FlexReport for WinForms** - newer, updated, and faster version of C1Report.

FlexReport is a comprehensive reporting tool that provides complete reporting solution - from building complex reports to previewing, exporting, and printing. With a rich object model, and modern user interfaces in its previewing control and designer application, FlexReport provides flexibility to generate attractive and feature-rich reports.
To work with FlexReport and/or FlexViewer, the minimal system requirements are Windows 7 SP1 or Windows Server 2008 R2 SP1 with Platform Update (KB2670838).

Present your data in a consolidated format, design and customize the reports, and take important business decisions from the reports generated through FlexReport. High quality rendering, accurate calculations, and ease of use make FlexReport a must have control for advanced as well as basic level report designers.

Help with WinForms Edition

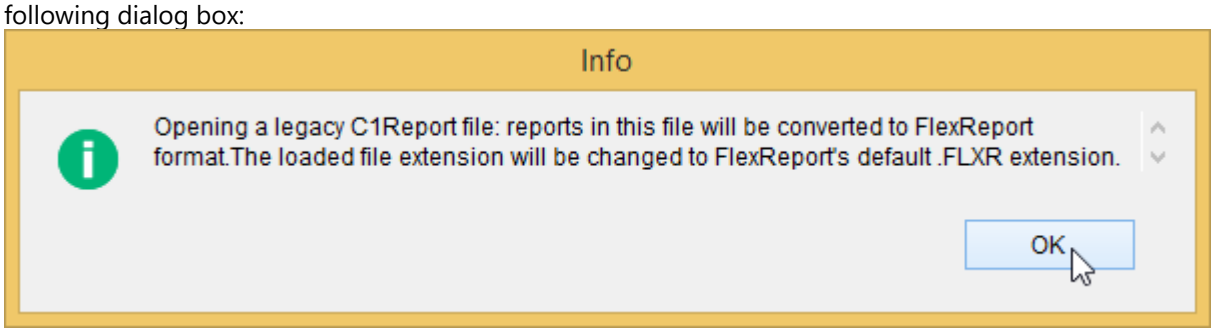
For information on installing **ComponentOne Studio WinForms Edition**, licensing, technical support, namespaces and creating a project with the control, please visit Getting Started with WinForms Edition.

Upgrading C1Report to FlexReport

The reports created using **C1Report** are fully compatible with C1FlexReport. The following are the two simple ways to upgrade or convert your existing C1Report report definition (.xml) to C1FlexReport report definition (.flxr):

Upgrading C1Report report definition in the designer

1. Run **C1FlexReportDesigner.4.exe**.
2. Go to **File|Open** and select the C1Report report definition (.xml) that you want to upgrade. You will see the following dialog box:



3. Click **OK**.
4. Go to **File|Save**.
5. In the **Save Report Definition File** dialog box, specify the **File name** and click **Save**.

The report definition is saved as type **.flxr**. Your report definition is now converted from C1Report's .xml to C1FlexReport's .flxr.

Upgrading the existing C1Report Windows Application Project to FlexReport Windows Application Project in Visual Studio

1. Open existing Windows Application project that contains C1Report (.xml) file.
2. Delete C1Report from the Form.
3. Delete C1Report's previewing control C1PrintPreview or C1RibbonPreviewControl from the Form.
4. Add C1FlexReport component to the **Toolbox**:
    a. Right-click a tab and select **Choose items...** A **Choose Toolbox Items** dialog box appears.

   b. Click Browse and select **C1.Win.FlexReport.4.dll** from the bin\v4.0 folder. The **C1FlexReport** component will be added in the Toolbox.
5. Add **FlexViewer** control to the **Toolbox**:
   a. Right-click a tab and select **Choose items...** A **Choose Toolbox Items** dialog box appears.
   b. Click Browse and select **C1.Win.FlexViewer.4.dll** from the bin\v4.0 folder. The FlexViewer control will be added in the **Toolbox**.
6. Drop **C1FlexReport** on the Form. The following dlls with the same version as the version of C1FlexReport should get added to the references:
   - C1.C1Pdf.4
   - C1.Win.4
   - C1.Win.BarCode.4
   - C1.Win.C1Document.4
   - C1.Win.FlexReport.4

   If these references do not have the same version, you need to add them manually.
7. Drop FlexViewer control on the Form. The following dlls with the same version as the version of C1FlexViewer should get added to the references:
   - C1.C1Zip.4
   - C1.Win.C1DX.4
   - C1.Win.C1Ribbon.4
   - C1.Win.FlexViewer.4
   - C1.Win.ImportServices.4

   If these references do not have the same version, you need to add them manually.
8. Add the following dlls to the references:
   - C1.C1Word.4
   - C1.C1Excel.4
   - C1.Win.C1Chart.4
   - C1.Win.C1Chart3D.4

   To use map and super-label custom fields, add following dlls to the references:
   - C1.Win.FlexReport.CustomFields.4
   - C1.WPF.Maps.4
   - C1.WPF.4
9. Rename **C1Report** component to **C1FlexReport** component. In code, you can change the name of the component as follows:

Visual Basic

```
Dim report As New C1Report()
'To
Dim report As New C1FlexReport()
```

C#

```
C1Report report = new C1Report();

//To

C1FlexReport report = new C1FlexReport();
```

10. Change the name of namespace from C1.C1Report to **C1.Win.FlexReport** in code-behind.
11. Delete all references to the dlls of C1Report and its dependencies - C1.C1Report, C1.Win.C1Report, C1.Win.C1Barcode, and C1.Win.C1RibbonPreview.
12. Delete namespace C1.Win.C1Preview.
13. Delete license entires of C1Report and the referenced viewer (C1Preview or C1RibbonPreview) from

licenses.licx.

14. In order to bind C1FlexReport with C1FlexViewer, following code will have to be changed as follows:

Visual Basic

```
Dim c1r As C1.C1Report.C1Report = New C1Report()
c1r.Load(filepath, reportname)
C1PrintPreviewControl1.Document = clr
'To
Dim report As New C1FlexReport()
report.Load(filepath, reportname)
C1FlexViewer1.DocumentSource = report
```

C#

```
C1.C1Report.C1Report c1r = new C1Report();
c1r.Load(filepath, reportname);
c1PrintPreviewControl1.Document=clr;

//To
C1FlexReport report = new C1FlexReport();
report.Load(filepath, reportname);
c1FlexViewer1.DocumentSource = report;
```

Note that FlexReport can be previewed at runtime by using FlexViewer control only. The FlexViewer control is not compatible with C1PrintPreviewControl or C1RibbonPreviewControl.

FlexReport versus C1Report

FlexReport is the newer and improved C1Report, with the following main differences:

Hierarchy of report field types

The structure of Field objects in FlexReport is hierarchical, with FieldBase as a base class, and other different class types to represent different fields. So, there are different types of report fields to represent text, image, shape, subreports, and other different types of data. The advantage of having a hierarchy of Field objects in FlexReports is that it makes working with fields quite easy and flexible.

C1Report, on the other hand, has complex field objects that require defining the **C1.C1Report.Field.Text** property and setting **Calculated** to True. Here, each field is interpreted as an expression which is evaluated individually for each record, which slows down the process of rendering of data in C1Report's fields.

Regardless of this, the field types in FlexReport are fully compatible with the field types in C1Report, which enables loading or rendering of C1Report definitions in FlexReport.

Multiple data sources

A FlexReport definition can have several data sources each identified by a unique name. In case a report contains multiple data sources, one of the data sources acts as the main data source for the report; the values from the other data sources can be used for adding report parameters and creating chart fields. For more information, see Data Sources in FlexReport.

Improved data sorting

FlexReport provides improved data source sorting capability. You can define several sort expressions for a data to specify the sorting condition and its direction using DataSource.SortDefinitions property.

In addition, sorting and grouping of data are independent of each other, that is, you can apply grouping on a set of records as well as control the order of records through sorting. A simple example is sorting the data - City while grouping by - Country. This results in improved compatibility with Crystal Reports. See Sorting Data for more information.

Improved data filtering

FlexReport provides improved data filtering that can be specified in a regular VBScript (as all other expressions in C1Report or FlexReport) or in DataView (several expressions to specify the criteria to filter the data). The syntax type for filtering data in FlexReport is specified by setting FilterExpressionSyntax enum to DataView (which is default) or VBScript.

Filtering in C1Report, on the other hand, is specified by **C1Report.Filter** property, which uses the limited syntax as the **DataView.Filter** property. Regardless of this, filtering in FlexReport is fully compatible with filtering in C1Report. For more information, see Filtering Data.

Calculated fields for data source

In FlexReport, data source supports Calculated fields, that is, calculated fields can be defined in the data sources to fetch calculated data. The expressions in the Calculated fields are specified through VBScript expressions in DataSource.CalculatedFields collection. These expressions can use other data source fields, report parameters, and so on. For more information, see Defining Calculated Fields.

Report parameters

The report parameters in a report are used to modify the default values of the data and hence update the values when the report is rendered. In FlexReport, report parameters can be defined in the C1FlexReport.Parameters collection, where each element is an instance of the ReportParameter class, with some additional properties, that are described in Adding Parameters topic.

In C1Report, parameters could be specified either in **DataSource.RecordSource** or in **DataSource**, using PARAMETERS "key word" in the connection string, such as the following string:

```
C1Report.DataSource.RecordSource = "PARAMETERS param1 int 0; select * from Customers where id < param1".
```

When C1Report report definitions are loaded in FlexReport, parameters specified using DataSource.RecordSource and DataSource.Filter are imported correctly into the **C1FlexReport.Parameters** collection.

Multiple Sub-sections

In C1FlexReport, each section contains at least one sub-section. The sub-sections, just like sections, contain report fields. The advantage of adding sub-sections is that they help in enhancing the data present in their parent section. Sub-sections can be accessed through Section.SubSections collection property. For more information, see Adding Multiple Sub-sections.

Visual properties

- **Borders**
  In FlexReport, borders can be specified for fields, sub-sections, and sections using VisualReportObject.Border property. In addition, borders can have each side having its own style and each corner having a different radii for rounded corners. See Border for more information.
  In C1Report, borders can be defined only on fields.
- **Backgrounds**
  In FlexReport, backgrounds can be specified for fields, sub-sections, and sections. The background color can be solid or gradient, which can be set using VisualReportObject.Background property. See Background for more information.
  In C1Report, **BackColor** property is used to set background colors for fields and sections.

C1Report to FlexReport Breaking Changes

FlexReport code has been written from scratch; as a result, you will find following breaking changes in the API on migrating from C1Report to FlexReport:

- In C1Report, parameters can only be specified directly in the **DataSource.RecordSource**, immediately before the SQL statement and after the key word PARAMETERS.

  In FlexReport, there is a separate dedicated report-level collection C1FlexReport.Parameters, where the report parameters can be specified. When importing a C1Report report definition from a .xml file, any parameters specified old-style using PARAMETERS keyword are automatically added to the **C1FlexReport.Parameters** collection.

- In FlexReport, the **OnOpen** script is fired AFTER the data source has been opened, so any changes to the main data source made in that script does not affect the report. In order to change something in the data source before the report is generated, use **GlobalScripts**. GlobalScripts can contain function and procedure definitions, and codes that are not within these definitions; all such definitions and codes are now executed when the report starts rendering, before the data source is opened.

- In C1Report, Custom fields derived from **Field** overrides the **GetRenderContent()** method. The method's signature in C1FlexReport has been changed to:

  ```
  public virtual void GetDesignerRenderContent(
  ref string text,
  ref Image image,
  ref bool disposeImage);
  ```

  If the overriding method sets disposeImage to true, C1FlexReport calls **Dispose()** on the image after it has been used.

- The **C1Report.OutlineRootLevel** property has been removed. To control the outline structure, use properties OutlineLabel

and OutlineParent. To turn off outlines generated by a subreport, use the SubreportField.OutlinesVisible property.

- In C1Report, there are two slightly different methods to generate/layout text - default and 'gdi+' (if **C1Report.UseGdiPlusTextRendering** were set to true - non-default). These methods can produce slightly different text layouts, e.g. line breaks could be in different places etc. FlexReport always generates/lays out text like C1Report with UseGdiPlusTextRendering set, but still there may be differences in line breaks between C1Report with UseGdiPlusTextRendering set, and FlexReport.
- The **AddOutlineEntry** event has been removed. To change the text of the outline entry generated by a field/section/sub-section, use the **OutlineLabel** property.
- A new specialized event type has been added for the **C1FlexReport.ReportError** event: ReportErrorEventHandler, accepting ReportErrorEventArgs event arguments. The ReportEventArgs type has been modified - Exception and Handled have been removed from the event arguments.
- If a report contains Map custom field(s), it must be generated synchronously (call Render() rather than **RenderAsync()**), else the map field cannot be displayed in the FlexViewer control. You should set the FlexViewer.UseAsyncRendering property to False.
- In C1Report, EndReport event is not fired in case of an error. In FlexReport, the **EndReport** event is fired even if a fatal error occurs during rendering.
- Following C1Report methods/properties have been removed from C1FlexReport:
  - C1Report.Document.Generate()
  - C1Report.Document.Export()
  - C1Report.Document.CreationDpi
  - C1Report.DocumentInternal
  - C1Report.Document.DoEvents
  - C1Report.Document.HasEditableTags
  - C1Report.CreationDevice
  - StartReport(), StartSection(), EndSection(), RenderField()
  - C1Report.DataSource.DataObject: In C1FlexReport, the same can be accessed through DataSource.Recordset property.
  - C1Report.EmfType: In C1FlexReport, use EMF+ instead.
  - C1Report.GetReportInfo(): In C1FlexReport, use C1FlexReport.ReportInfo instead.
  - C1Report.PageRenderingMode: In C1FlexReport, use C1FlexReport.GetPageImage in order to get a page's metafile.
- C1Report Render<X> method cannot be accessed in code behind with C1FlexReport. These methods are for internal use.
- C1Report Render<X> object cannot be used by using following code:

  ```
  c1Report1.Document.Body.Children.Add(RenderGraphic obj)
  ```

  This method is not supported in FlexReport.
- C1FlexReport.Document cannot be converted to System.Drawing.Printing.PrintDocument. PrintDocument should not be used with FlexReport since PrintDocument's **C1Report.Document** property does not exist.
- In FlexReport, **IC1FlexReportRecordset** does not have ApplyFilter() and ApplySort() methods. Instead, DataSource in FlexReport has filters/sort definition, so these should be used. IC1FlexReportRecordset is assigned to DataSource.Recordset while filters/sorts can be defined on the DataSource.
- C1Report's **FieldBase** object can no longer be used with FlexReport.

  Field is a 'legacy' type with C1FlexReport. Now specialized types are derived from FieldBase in C1FlexReport. Following field objects should be directly created in code-behind:

  - TextField text = new TextField();
  - BarCodeField barcode = new BarCodeField();
  - SubreportField subreport = new SubreportField();
  - ChartField chart = new ChartField();
  - RTFField rtf=new RTFField();
  - CheckBoxField checkbox = new CheckBoxField();
  - CalculatedField calField=new CalculatedField();
  - ImageField img = new ImageField();
  - ShapeField shape = new ShapeField();

  Use corresponding properties, same as used to be set for C1Report FieldBase object.

- CanGrow and CanShrink properties of C1Report have been renamed in C1FlexReport.
  Use C1FlexReport.AutoHeight, C1FlexReport.AutoWidth, and C1FlexReport.AutoSizeBehavior instead. CanGrow=True and CanShrink=True can be used as C1FlexReport.AutoSizeBehavior.GrowAndShrink.
- In FlexReport, instead of the **AddScriptObject** event there is GetScriptObject event. So in this case, instead of

```
private void c1flxr_StartReport(object sender, System.EventArgs e)
{
  c1flxr.AddScriptObject("LookUp", new LookUpObject());
}

this works:


c1flxr.GetScriptObject += c1flxr_GetScriptObject;
...

void c1flxr_GetScriptObject(object sender, C1.Win.FlexReport.ReportGetScriptObjectEventArgs e)
{
  if (e.Name.ToLower() == "lookup")
    e.Object = new LookUpObject();
}
```

- C1FlexReport caches rendered content and does not regenerate a report if the report template is not changed. To ensure that report is regenerated, you can call C1FlexReport.SetDirty() method.

Key Features

The key features of **FlexReport for WinForms** are as follows:

- **Light-weight and Fast**
  FlexReport is light-weight and fast in particular for smaller reports. FlexReport is rendered twice as fast as C1Report; major exports such as PDF and HTML are much faster than C1Report.

- **High Quality Rendering**
  FlexReport uses DirectWrite/Direct2D to draw and generate high performance and quality report content that does not depend on measurement context like printer, screen, etc.

- **Single Viewer for all Document types**
  FlexViewer is a new control that has been introduced in 2015 v3 with FlexReport. It can be used to view multiple document types such as C1Report, C1FlexReport, SSRS, and C1Document. It gives you capabilities of using/resetting Parameters, options to Refresh and Cancel Report rendering, Bookmarks to jump to report locations, and more.
- **More accurate Crystal Report Migration**
  FlexReport supports some features that improve its compatibility with Crystal Reports. The proper migration for following features are supported:
  - Sub-sections
  - Complex Expressions
  - Special Ordered Group
  - Enhanced Border Styles

    See Importing Crystal Reports for more information.
- **Modern UI**
  Report designing application (FlexReportDesigner) and Previewing tool (FlexViewer) provide a Ribbon-based UI offering an intuitive and rich user experience with easy to access and well placed designer and viewer options.

- **New Designer Application with Rich User Experience**
  The FlexReportDesigner application has additional features that makes report designing much easier than before. The new features are:
  - Snap Lines to show/align distance from controls.
  - Collapsible/Expandable Sections and Sub-sections.
  - Show Captions settings for optionally displaying Section/Sub-sections header strips.
  - Chart Editors at Design time to set Chart Field's Properties, Data source, and Visual Effects
  - Data Tab to add, edit and remove Data Sources, Parameters, Sort Expressions, and Calculated Fields.
  - Ability to edit expressions through 'Edit Expression' in context menu.
  - Ability to align numbers to left and other values to right using Align General button in the designer.
  - Errors tab to show errors and warnings while importing or previewing a report.

    See About FlexReportDesigner for more information.
- **New and extensive set of Charts:**
  FlexReport provides more than 70 Chart types to choose from. Binding data to charts, setting properties, adding visual effects, and other related tasks can be performed easily through the Design-Time editors available in the FlexReportDesigner application.

- **Added support for SQLite database**
  The SQLite connection can be specified in the FlexReportDesigner to fetch data, just like any other database. For that, you should have SQLite ADO.NET provider installed on your system, see https://system.data.sqlite.org for more details.

Feature Comparison Matrix

Explore all of the features offered by FlexReport, Visual Studio Reporting, Report Builder 3.0, and Crystal Reports. You can download the matrix in PDF.

End User Designer for Section Report

Features	FlexReport	Visual Studio Reporting Control	Report Builder 3.0	Crystal Reports
Designer available with installer	✓	Available with MS SQL Server	Separate Installer	Available with Crystal Reports Installer
Create blank report	✓	✓	✓	Through Database Expert Wizard
Add chart to blank report	✓	✓	✓	✓
Add map to blank report	✓	✓	✓	✓
Add CrossTab to blank report		✓	✓	✓
Interface style	Modern UI, Ribbon Interface	✓	Modern UI, Ribbon Interface	Traditional WinForms interface

Visual Studio Templates

Features	FlexReport	Visual Studio Reporting Control	Report Builder 3.0	Crystal Reports
Reporting Application Available		✓		WinForms, WPF
Template applications that bind Report with Viewer		✓		✓
Report (blank), Report Wizard available in "Add new item"		✓		✓

Report Wizard

Features	FlexReport	Visual Studio Reporting Control	Report Builder 3.0	Crystal Reports
Report Wizard	✓	✓	✓	✓
Table list	Any data source	Any data source	SQL	✓

			Query for Access, SQL Server data source	
View list	Any data source	Any data source	SQL Query for Access	✓
Stored procedure list	Any data source		SQL Query for Access, SQL Server data source	✓
Grouping set during Report Wizard binding	✓	✓	✓	✓
Grouping set only for Table/Matrix		✓	✓	

Report Styles

Features	FlexReport	Visual Studio Reporting Control	Report Builder 3.0	Crystal Reports
Built-in Report Styles	34 types	6 types	6 types	12 types
Columnar	✓			
Tabular	✓	✓	✓	✓
Justified	✓			
Labels	✓			✓
Stepped	✓			
Outline	✓			
Aligned	✓			
Standard				✓
Form Letters				✓
OLAP				✓

Query Builder

Features	FlexReport	Visual Studio	Report	Crystal Reports

		Reporting Control	Builder 3.0	
Graphical Query Designer	Any datasource		SQL Server Relational Databases only	
Text-based Query Designer		✓	Oracle, OLEDB, ODBC, TeraData	Any datasource

Report Layout & Controls

Features	FlexReport	Visual Studio Reporting Control	Report Builder 3.0	Crystal Reports
Section-based design template	.FLXR			.RPT
Page-based design template		.RDL/.RDLC	.RDL/.RDLC	
Multiple reports in a report	✓			

Data Binding

Features	FlexReport	Visual Studio Reporting Control	Report Builder 3.0	Crystal Reports
MS SQL Server	✓	✓	✓	
Object Datasource	✓			✓
OLEDB	✓	✓	✓	✓
SQLite	✓			
Stored Procedure	✓			✓
XML	✓	✓	✓	✓
Bind report to multiple tables	✓	✓	✓	
Drag & drop bound fields to report	✓	✓	✓	✓

Microsoft SQL Server Analysis Services for MDX		✓	✓	
DMX		✓	✓	
Microsoft Power Pivot and tabular models		✓	✓	
Microsoft Azure SQL Database		✓	✓	
SQL Server Parallel Data Warehouse		✓	✓	
Oracle		✓	✓	
SAP NetWeaver BI		✓	✓	
Hyperion Essbase		✓	✓	
Microsoft SharePoint List		✓	✓	
Teradata		✓	✓	
ODBC		✓	✓	✓
Access	✓	✓	✓	✓
Excel				✓
ADO.NET				✓
Java Beans				✓
Salesforce				✓
OLAP				✓
SAP Table				✓
SAP BW				✓
Outlook				✓

Design Preview

Features	FlexReport	Visual Studio	Report	Crystal

		Reporting Control	Builder 3.0	Reports
Preview Reports	Designer; not Visual Studio		Designer; not Visual Studio	Designer; not Visual Studio
HTML preview				Designer Edition only
Edit reports in preview				Designer Edition only
Portrait/landscape control in preview	✓	✓	✓	

Viewers

Features	FlexReport	Visual Studio Reporting Control	Report Builder 3.0	Crystal Reports
HTML	✓			WebViewer
MVC	✓			
UWP	✓			
Windows	✓	✓	✓	✓
WinForms	✓	✓		✓
WPF	✓	✓		✓
Web		✓		✓

Report Features

Features	FlexReport	Visual Studio Reporting Control	Report Builder 3.0	Crystal Reports
Adding Sort Expressions on Groups at report level	✓			
Drillthrough		✓	✓	
Filtering	✓	✓	✓	✓
Formatting controls - Done through tabs and groups of Ribbon	✓	✓	✓	✓
Grouping	✓	✓	✓	✓
Hyperlinks	✓	✓	✓	✓

Page Number	✓	✓	✓	✓
ReportEvents	✓			
Runtime Date & time	✓	✓	✓	✓
Sorting	✓	✓	✓	✓
Summary Totals	✓	✓	✓	✓
Built-in Special Fields	✓	✓	✓	✓
Report Scheduling				✓
Adding Sort Expressions on Groups available on Data Regions	✓	✓	✓	
Subreports	✓	✓	✓	✓
Interactive Sorting at Runtime	✓	✓	✓	✓

Parameters

Features	FlexReport	Visual Studio Reporting Control	Report Builder 3.0	Crystal Reports
AllowBlank Parameters	✓	✓	✓	
Boolean	✓	✓	✓	✓
DateTime	✓	✓	✓	✓
Float	✓	✓	✓	
Hidden Parameters	✓		✓	
Integer	✓	✓	✓	
Multivalue	✓	✓	✓	
Nullable	✓	✓	✓	
Passing parameters to reports	✓	✓	✓	✓
String	✓	✓	✓	✓

Time	✓			✓
Date	✓			✓
Currency				✓
Number	✓	✓	✓	✓
Internal Parameters			✓	
Options to refresh report with change in Parameter values	✓	✓	✓	✓

Formats

Features	FlexReport	Visual Studio Reporting Control	Report Builder 3.0	Crystal Reports
Boolean	✓			✓
Currency	✓	✓	✓	✓
Date	✓	✓	✓	✓
LongDate	✓	✓	✓	✓
LongTime	✓	✓	✓	✓
Number	✓	✓	✓	✓
Percentage	✓	✓	✓	✓
Short Date	✓	✓	✓	✓
Short Time	✓	✓	✓	✓
Time	✓	✓	✓	✓
Decimal		✓	✓	✓
Full Date/time long		✓	✓	✓
Full Date/time short		✓	✓	✓
General		✓	✓	✓
General Date/time long		✓	✓	✓
General Date/time short		✓	✓	✓
Month day		✓	✓	✓
RFC1123 pattern		✓	✓	
Round trip		✓	✓	
Scientific		✓	✓	

Year Month		✓	✓	✓

Charts

Features	FlexReport	Visual Studio Reporting Control	Report Builder 3.0	Crystal Reports
Total Chart Types	70+	50+	50+	40
2D	38	✓	✓	33
3D	36	✓	✓	7
Area	✓	✓	✓	✓
Bar	✓	✓	✓	✓
Column	✓	✓	✓	✓
Cone	✓			
Cylinder	✓	✓	✓	
Doughnut	✓	✓	✓	
Gantt	✓			✓
Histogram	✓			✓
Line	✓	✓	✓	✓
Pie	✓	✓	✓	✓
Polar	✓	✓	✓	
Pyramid	✓	✓	✓	
Radar	✓	✓	✓	✓
Scatter	✓	✓	✓	✓
Stacked	✓	✓	✓	✓
Step	✓			
Stock	✓	✓	✓	✓
Bubble		✓	✓	✓
Candlestick		✓	✓	
Funnel		✓	✓	✓
Range		✓	✓	

Chart Wizards

Features	FlexReport	Visual Studio Reporting Control	Report Builder 3.0	Crystal Reports
Properties	✓			✓

Bind data source	✓	✓	✓	✓
Visual Effects	Advanced		✓	Basic
Choose chart type	✓	✓	✓	✓

Export

Features	FlexReport	Visual Studio Reporting Control	Report Builder 3.0	Crystal Reports
Total Formats	14	3	7	10
BMP	✓			
Excel (.xls)	✓			✓
Open XML Excel (.xlsx)	✓		✓	✓
GIF	✓			
HTML	✓			
JPEG	✓			
PDF	✓	✓	✓	✓
PNG	✓			
RTF	✓			✓
TIFF	✓	✓	✓	
Open XML Word (.docx)	✓	✓	✓	
Zip compressed files	✓			
Crystal Reports				✓
XML				✓
CSV		✓	✓	✓
MHTML		✓	✓	
Word (.doc)		✓	✓	✓

Import

Features	FlexReport	Visual Studio Reporting Control	Report Builder 3.0	Crystal Reports
C1Report	✓			
ADP Reports	✓			
MDB Reports	✓		✓	
Crystal Reports	✓			✓

Code Behind

Features	FlexReport	Visual Studio Reporting Control	Report Builder 3.0	Crystal Reports
Can create report in code behind	✓			✓

Scripting & Expressions

Features	FlexReport	Visual Studio Reporting Control	Report Builder 3.0	Crystal Reports
VB Script Editor	✓	✓	✓	
Syntax check available	✓	✓	✓	✓
C# Editor				✓

Deployment Licensing

Features	FlexReport	Visual Studio Reporting Control	Report Builder 3.0	Crystal Reports
Internal business use for client application	No license required	✓	✓	No license required
Third-party servers				License required
Royalty-free development	✓	✓		

Sample Reports

Features	FlexReport	Visual Studio Reporting Control	Report Builder 3.0	Crystal Reports
Installed with setup	✓	✓		Not installed at setup; available separately
Available for download	✓			✓
Medical Report	✓			
Inventory	✓			

Management				
Balance Sheet	✓			✓
Budget Report	✓			
Hospital Bills				✓
SPC Process Control Chart				✓
Telephone bill for FlexReport	✓			

IntelliSense

Features	FlexReport	Visual Studio Reporting Control	Report Builder 3.0	Crystal Reports
Report Fields	✓			
Database Fields	✓			
Calculated Fields	✓			
Parameters	✓			
VB Script Functions	✓			
Aggregates	✓			
Associated Properties	✓			
Property description	✓			
Report Design				
Add/insert resize actions for report sections	✓	✓	✓	
Can grow, Grow and Shrink, Can shrink, Keep Together available to control layout of text	✓	✓	✓	
Caption settings like All, Section, Hairline, Hidden	✓			
Expand/Collapse buttons for sections/subsections	✓			
Group header/footer	✓			
Reorder	✓			
Resize actions for report sections available	✓			

Snap to grid	✓			
Subsections	✓			

Server Edition

Features	FlexReport	Visual Studio Reporting Control	Report Builder 3.0	Crystal Reports
SAP Crystal Reports Server	Jul-16			✓
SAP Business Objects Enterprise				✓

FlexReport Dissection

Before starting with FlexReport, it is important to know about the components and controls that are shipped with FlexReport, the object model of FlexReport, and sections or bands available in FlexReport. The following sections provide indepth information about these.

Components and Controls

FlexReport for WinForms consists of following assemblies:

C1.Win.FlexReport dll

It includes all report rendering and document generating functionalities through following component:

- **C1FlexReport:**
  The C1FlexReport component is a report generating component that generates data-based banded reports. You can render reports directly to a printer or preview control, or export to various portable formats (including XLS, PDF, HTML, text, and images). The FlexReport component also exposes a rich object model for creating, customizing, loading, and saving report definitions.

C1.Win.FlexViewer dll

It includes all viewing functionalities through following controls or components:

- **C1FlexViewer:**
  The FlexViewer control is a multiple document previewing control. It has a ribbon-based UI with all preview related options (viewer pane and status bar) easily accessible.
- **C1FlexViewerPane:**
  The FlexViewerPane control displays the pages of the document being previewed, allows panning, zooming and other preview operations. In the form designer, standard toolbars and status bar can be created on the current form through context menu items.
- **C1FlexViewerDialog:**
  The FlexViewerDialog is a form that is displayed with the nested FlexViewer control.

Included Applications

In addition to the reporting components and controls, FlexReport also includes following stand-alone applications:

- **C1FlexReportDesigner.4 exe and C1FlexReportDesigner32.4.exe**
  These are built-in applications for creating and editing C1FlexReport report definition files. The designers allows you to create, edit, load, and save files (XML) that can be read by the C1FlexReport component.

Object Model Summary

C1FlexReport has a rich object model, which is largely based on the Microsoft Access model. The objects, collections, and the associated properties and methods together provide an ease and flexibility in generating FlexReport. The following table lists objects and their main properties and methods:

C1FlexReport Object
ReportName, Load, GetReportList, Save, Clear, Render, Parameters, Document, DoEvents, Cancel, Page, MaxPages, Font, OnOpen, OnClose, OnNoData, OnPage, OnError, Evaluate, Execute
Layout Object
Width, MarginLeft, MarginTop, MarginRight, MarginBottom, PaperSize, Orientation, Columns,

ColumnLayout, PageHeader, PageFooter
DataSource Object
CalculatedFields, ConnectionString, Filter, RecordSource, SortDefinitions
DataSourceCollection
Report, Add, RemoveAt
SortDefinition Object
Direction, Expression
SortDefinitionCollection
Owner, Report
CalculatedField Object
DataSource, Expression, Type
CalculatedFieldCollection
Owner, Report
Group
GroupBy, KeepTogether, SectionHeader, SectionFooter, Sort, SortExpression
GroupCollection
Add, Clear, RemoveAt, Report
ReportParameter Object
AllowedValuesDefinition, DisplayText, ParentReport, SetName
ReportParameterCollection
InsertItem, RemoveItem, SetItem, Report
AllowedValuesDefinition Object
AssignFrom, Binding, Values
Section Object
Calculated, Fields, Height, KeepTogether, SplitBehavior, SubSections
SectionCollection
Detail, Footer, Header, PageFooter, PageHeader
SubSection Object
Calculated, Fields, Height, ParentReport, ParentSection, SplitBehavior, Visible
SubSectionCollection
Add, Remove, RemoveAt, Report
FieldBase Object
Anchor, Height, KeepTogether, ForcePageBreak, MarginBottom, MarginLeft, MarginRight, MarginTop, Section, SplitHorzBehavior, SplitVertBehavior
FieldCollection

Add, Remove, RemoveAt
BarCodeField Object
BarCode, BarCodeOptions, Font, Text
CheckBoxField Object
CheckAlign, CheckMark, Text, ThreeState, Value
DataField Object
Calculated, Name, Type, Value
ImageField Object
AssignFrom, PictureAlign, PictureScale
RtfField Object
AssignFrom, DetectUrls, Text
ShapeField Object
Line, Shape, ShapeBackColor, ShapeBackground, ShapeType
SubreportField Object
ParameterValues, Subreport, SubreportFilter
TextField Object
Format, Text
VisualReportObject
Background, Border, BordersSplitHorzMode, BordersSplitVertMode, OutlineLabel
BehaviorOptions
AssignFrom, Reset, IgnoreInvisibleFieldsInGrowShrinkSections

Sections of FlexReport

Every report consists of the following five basic sections:

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

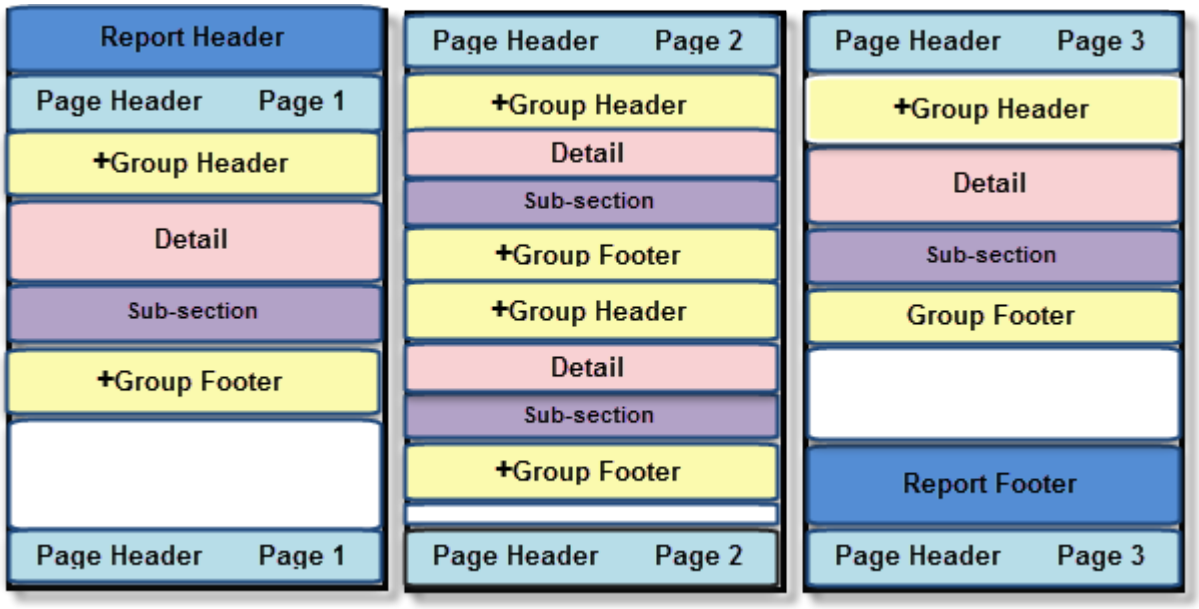
There are two additional sections for each group: a **Group Header** and a **Group Footer** Section. For example, a report

with 3 grouping levels will have 11 sections.

An additional section, called **Sub-section** can also be added to any section of a report.

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

The following diagram shows how each section is rendered on a typical report. It also displays how a sub-section in the Detail section is rendered in the report:



Report Header

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

Page Header

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

Group Headers and Group Footers

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

Detail

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

Page Footer

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

Report Footer

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

Sub-section

The sub-section can be added to any section of a report; by default it gets added at the bottom of the section that is currently selected. This section contains the additional data that enhances the data present in its parent section. A section's height is determined by the sum of heights of its sub-sections.

Customized sections

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

FlexReport Quick Start

Although you can use C1FlexReport in many different scenarios, on the desktop, the main sequence of steps is always the same:

1. **Create a report definition**

   This can be done directly with the **FlexReportDesigner** application or using the report designer in Microsoft Access and Crystal Report, and then importing it into the **FlexReportDesigner**. You can also do it using code, either by using the object model to add groups and fields or by writing a custom XML file.

2. **Load the report into the C1FlexReport component**

   This can be done at design time, using the **Load Report** context menu, or programmatically using the C1FlexReport.Load method. If you load the report at design time, it will be persisted (saved) with the control and you won't need to distribute the report definition file.

3. **Render the report**

   You can render the report into a **FlexViewer** control using the C1FlexViewer.DocumentSource property. The preview control will display the report on the screen, and users will be able to preview it with full zooming, panning, and so on. Note that report rendering is supported only for desktop applications.

The following steps will show you how to create a report definition, load the report into the C1FlexReport component, and render the report.

Step 1 of 4: Creating a Report Definition

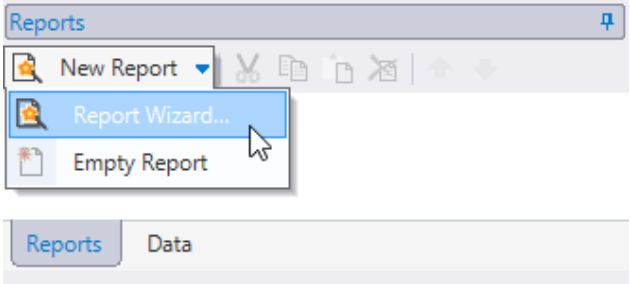
The following topic shows how you can create a report definition using the **FlexReportDesigner** application and the **code**. Note that creating a report definition is different from rendering a report. To render a report, you can simply load an existing definition and call the C1FlexReport.Render method. The easiest way to create a report definition is to use the **C1FlexReportDesigner**, which is a stand-alone application similar to the report designer in Microsoft Access and Crystal Report. You can find the detailed information about FlexReportDesigner and how it looks in About FlexReportDesigner.

The **C1FlexReportDesigner.exe** for 64 bit platform and **C1FlexReportDesigner32.4.exe** for 32 bit platform are located at **C:\Program Files (x86)\ComponentOne\Apps\v4.0** on your computer.

Creating a Report Definition Using the FlexReportDesigner:

You can create a new report definition in FlexReportDesigner using **FlexReport Wizard**. The **FlexReport Wizard** walks you through the steps of creating a new report from start to finish. To begin, complete the following steps:

1. Run the **C1FlexReportDesigner.exe** file from the location discussed in About FlexReportDesigner.
2. Go to File Menu in the menu bar and select **New** command.
   Blank space appears in the FlexReportdesigner to create a new report.
3. Click **New Report** drop down from the **Reports** tab located on the extreme left of designer and select **Report Wizard**.



The **FlexReport Wizard** opens.

From the **C1FlexReport Wizard**, complete the following five steps to create your report:

1. **Select the data source for the new report.**

   Use this page to select the DataSource.ConnectionString and DataSource.RecordSource that will be used to retrieve the data for the report.

   You can specify the **DataSource.ConnectionString** in three ways:

   - Type the string directly into the editor.
   - Use the drop-down list to select a recently used connection string (the Designer keeps a record of the last eight connection strings).
   - Click the **ellipses** button **(...)** to bring up the standard connection string builder.

   You can specify the **DataSource.RecordSource** string in two ways:

   - Click the **Table** option and select a table from the list.
   - Click the **SQL** option and type (or paste) an SQL statement into the editor.

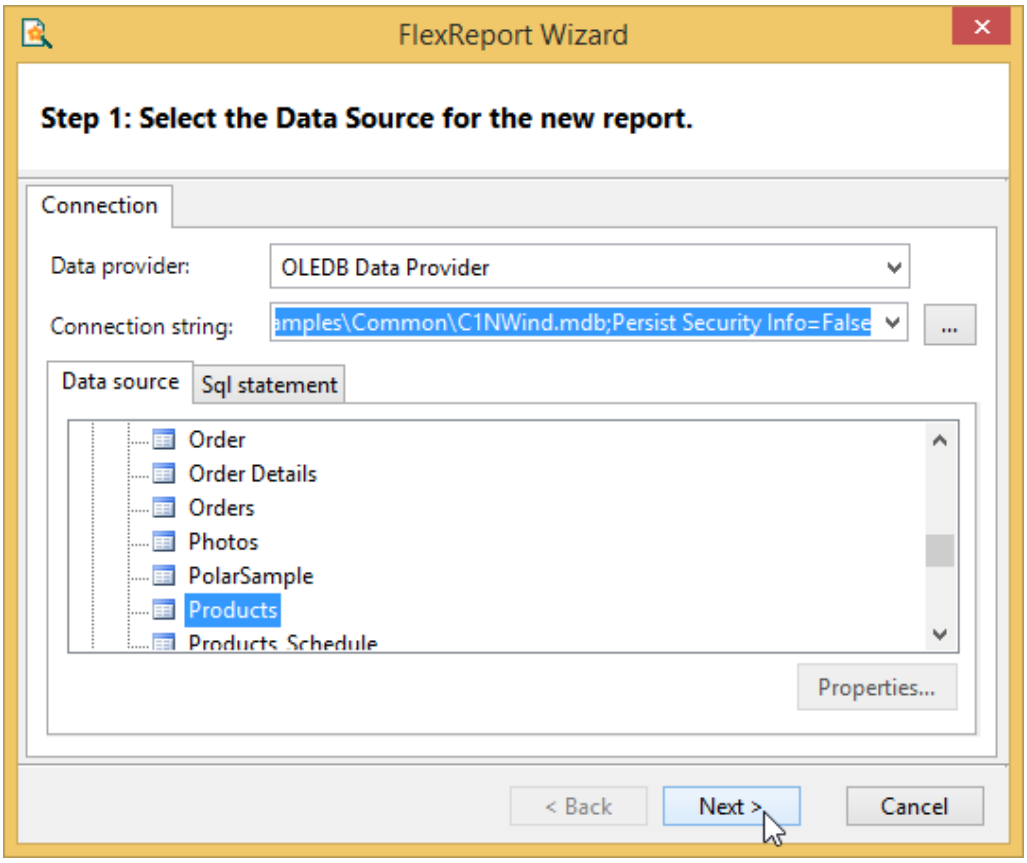
   Complete Step 1:

   Complete the following steps:

   1. Click the **ellipses** button to bring up the standard connection string builder. The **Data Link Properties** dialog box opens.
   2. Select the **Provider** tab and select a data provider from the list. For this example, select **Microsoft Jet 4.0 OLE DB Provider**.
   3. Click the **Next** button or select the **Connection** tab. Now you must choose a data source.
   4. Click the **ellipses** button to select a database. The **Select Access Database** dialog box appears. For this example, select the **C1NWind.mdb** located in the **Common** folder in the **ComponentOne Samples** directory (by default installed in the **Documents** folder). Note that this directory reflects the default installation path and its path may be different if you made changes to the installation path.
   5. Click **Open**. You can test the connection and click **OK**.
   6. Click **OK** to close the **Data Link Properties** dialog box.
   7. Once you have selected your data source, you can select a table, view, or stored procedure to provide the actual data. You can specify the DataSource.RecordSource string in two ways:
      - Select the **Data source** tab and select the **Products** table from the **Tables** list.
      - Select the **SQL** tab and type (or paste) an SQL statement into the editor.

        For example:

        ```
        select * from products
        ```



8. Click **Next**. The wizard will walk you through the remaining steps.

2. **Select the fields you want to include in the report.**

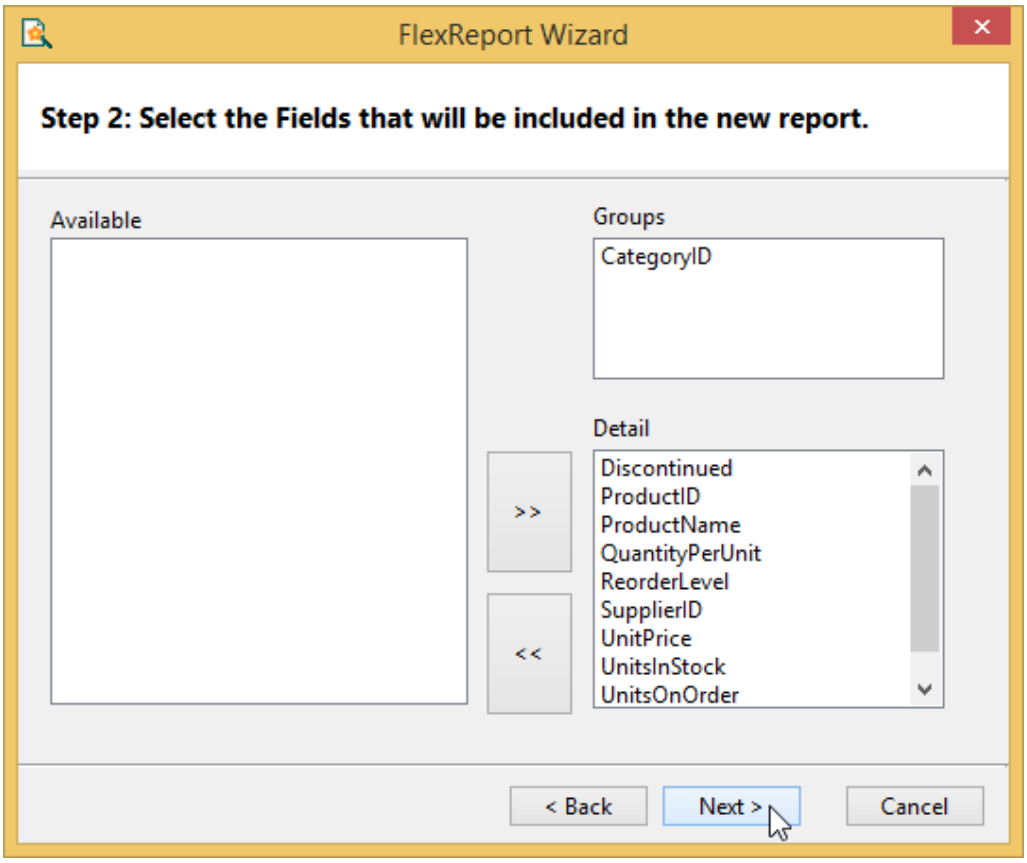
   This page contains a list of the fields available from the recordset you selected in Step 1, and two lists that define the group and detail fields for the report. Group fields define how the data will be sorted and summarized, and detail fields define what information you want to appear in the report.

   You can move fields from one list to another by dragging them with your mouse pointer. Drag fields into the **Detail** list to include them in the report, or drag within the list to change their order. Drag fields back into the **Available** list to remove them from the report.

   Complete Step 2:

   Complete the following steps:

   1. With your mouse pointer, select the **CategoryID** field and drag it into the **Groups** list.
   2. Press the >> button to move the remaining fields into the **Detail** list.



3. Click **Next**. The wizard will walk you through the remaining steps.

3. **Select the layout for the new report.**

   This page offers you several options to define how the data will be organized on the page. When you select a layout, a thumbnail preview appears on the left to give you an idea of what the layout will look like on the page. There are two groups of layouts, one for reports that have no groups and one for reports with groups. Select the layout that best approximates what you want the final report to look like.

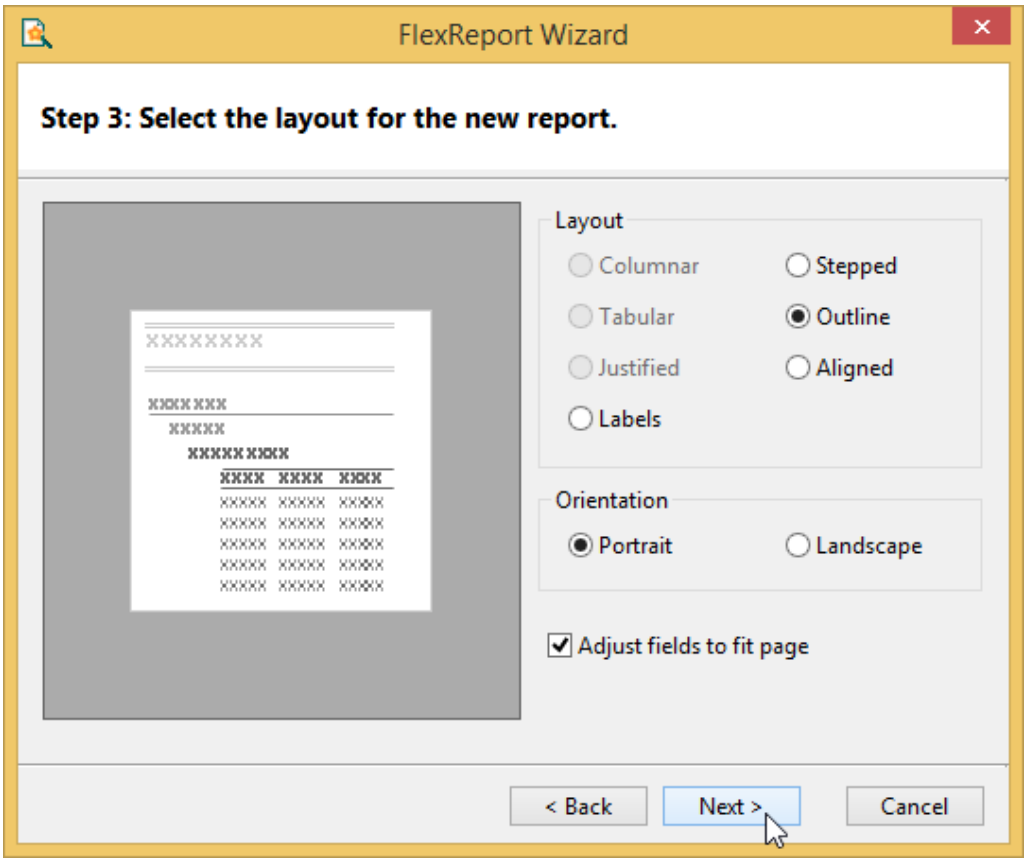
   This page also allows you to select the page orientation and whether fields should be adjusted to fit the page width.

   The **Labels** layout option is used to print Avery-style labels. If you select this option, you will see a page that prompts you for the type of label you want to print.

   Complete Step 3:

   Complete the following steps:

   1. Keep the **Outline** layout.



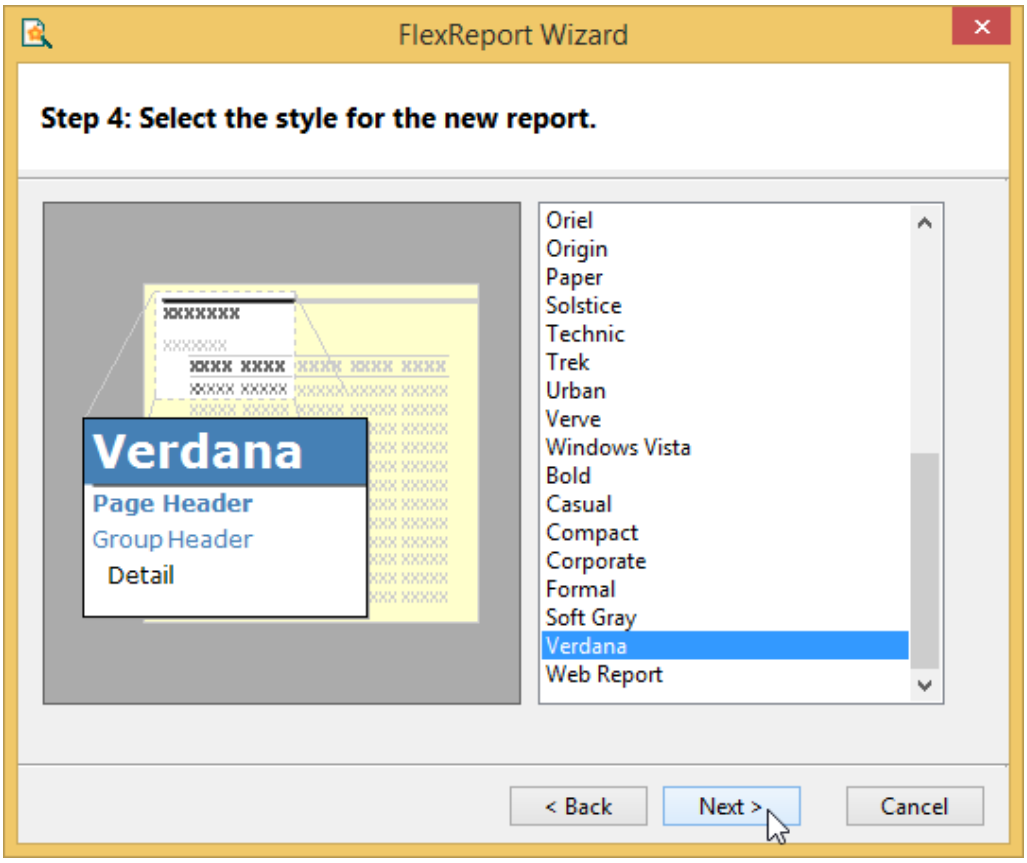
2. Click **Next**. The wizard will walk you through the remaining steps.

4. **Select the style for the new report.**

   This page allows you to select the fonts and colors that will be used in the new report. Like the previous page, it shows a preview to give you an idea of what each style looks like. Select the one that you like best (and remember, you can refine it and adjust the details later).

   Complete Step 4:

   1. Select the **Verdana** style.



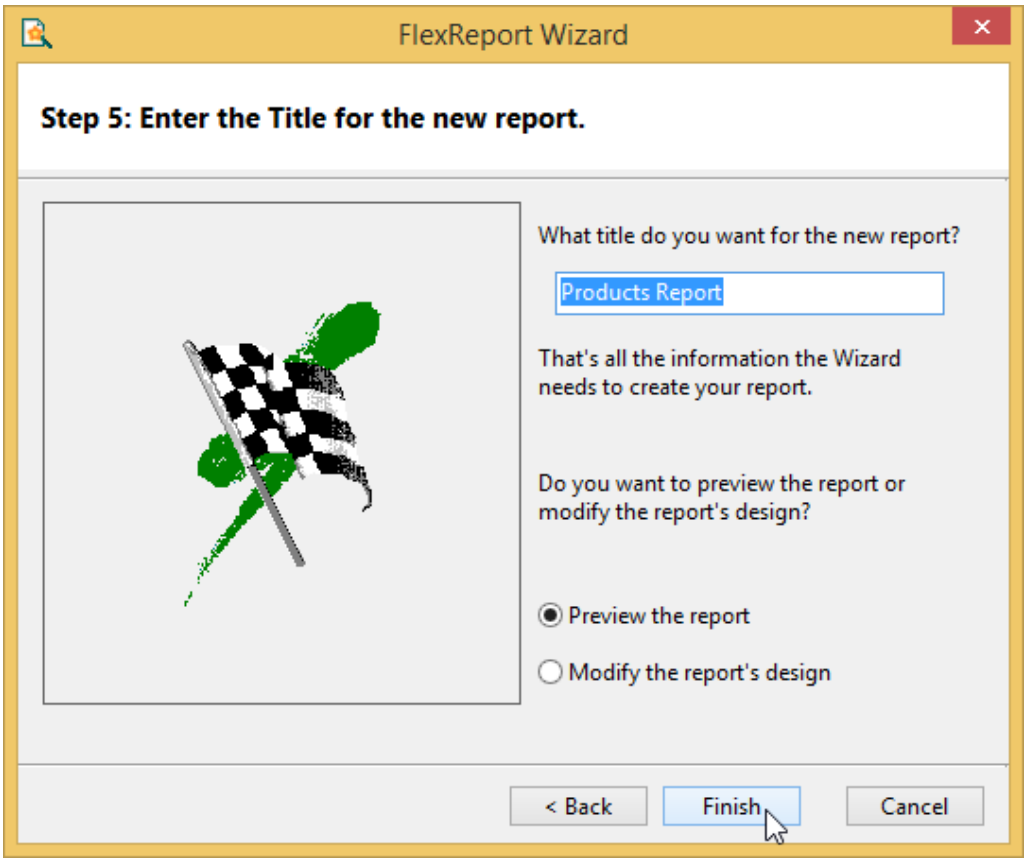
2. Click **Next**. The wizard will walk you through the remaining steps.

5. **Select a title for the new report.**

   This last page allows you to select a title for the new report and to decide whether you would like to preview the new report right away or whether you would like to go into edit mode and start improving the design before previewing it.

   Complete Step 5:

   1. Enter a title for the new report, **Products Report**, for example.



2. Choose to **Preview the report** and click **Finish**.

   You will immediately see the report in the preview pane of the **Designer**.

You will notice that the report will require some adjustments.

You can invoke **C1FlexReportDesigner** from Visual Studio as well. To do so, complete the following steps:

1. Create a .NET project and add the **C1FlexReport** component to your Toolbox.
2. From the Toolbox, double-click the **C1FlexReport** icon to add the component to your project. Note that the component will appear below the form in the Component Tray.
3. Click the **C1FlexReport** component's smart tag and select **Edit Report** from its **Tasks** menu.
   The **C1FlexReportDesigner** opens and the **C1FlexReport Wizard** is ready to guide you through the five easy steps discussed above.

Creating a Report Definition Using Code:

The following steps show how you can create a report definition using the **FlexReportDesigner** application or using code. You can even write your own report designer or ad-hoc report generator.

The example uses code to create a simple tabular report definition based on the C1NWind database. The code is commented and illustrates the most important elements of the C1FlexReport object model. Complete the following steps:

1. First, add a button control, C1FlexReport component, and FlexViewer control to your form. Set the following properties:

   Button.Name = **btnEmployees**

   C1FlexReport.Name = **c1FlexReport1** (default name in C#)

   C1FlexViewer.Name = **c1FlexViewer1** (default name in C#)

2. Initialize the control, named c1FlexReport1, using the Clear method to clear its contents and set the control font (this is the font that will be assigned to new fields):

Visual Basic

```
Private Sub RenderEmployees()

C1FlexReport1.DataSource.RecordSourceType = RecordSourceType.Auto '
clear any existing fields
C1FlexReport1.Clear()

' set default font for all controls
C1FlexReport1.Font.Name = "Tahoma"
C1FlexReport1.Font.Size = 8

End Sub
```

C#

```
private void RenderEmployees()
{
c1FlexReport1.DataSource.RecordSourceType = RecordSourceType.Auto;
   // clear any existing fields
   c1FlexReport1.Clear();
   // set default font for all controls
   c1FlexReport1.Font.Name = "Tahoma";
   c1FlexReport1.Font.Size = 8;
}
```

3. Next, set up the **DataSource** object to retrieve the data that you want from the C1NWind.mdb database. This is done using the **ConnectionString** and **RecordSource** properties:

Visual Basic

```
' initialize DataSource
Dim ds As DataSource = C1FlexReport1.DataSource
ds.ConnectionString = "Provider=Microsoft.Jet.OLEDB.4.0;Data
Source=C:\...\ComponentOne Samples\Common\C1NWind.mdb;"
ds.RecordSource = "Employees"
```

C#

```
//initialize DataSource
DataSource ds = c1FlexReport1.DataSource;
ds.ConnectionString = @"Provider=Microsoft.Jet.OLEDB.4.0; Data
Source=C:\...\ComponentOne Samples\Common\C1NWind.mdb;";
ds.RecordSource = "Employees";
```

4. Next, initialize the Layout object that defines how the report will be laid out on the page. In this case, render the report in Portrait mode and set its Width to 6.5 inches (8.5 page width minus one inch for margins on either side):

Visual Basic

```
'initialize Layout
Dim l As Layout = C1FlexReport1.Layout
l.Orientation = OrientationEnum.Portrait
l.Width = 6.5 * 1440 ' 8.5 - margins, in twips
```

C#

```
//initialize Layout
Layout l = c1FlexReport1.Layout;
l.Orientation = OrientationEnum.Portrait;
l.Width = 6.5 * 1440; // 8.5 - margins, in twips
```

5. Now comes the interesting part. Every report has five basic sections: **Detail**, **Report Header**, **Report Footer**, **Page Header**, and **Page Footer**. Use the following code to set up the report header by setting a couple of properties and adding a title field to it:

Visual Basic

```
' create a report header
Dim s As Section = C1FlexReport1.Sections(SectionTypeEnum.Header)
s.Height = 1440
s.Visible = True
s.BackColor = Color.FromArgb(200, 200, 200)
Dim textFld1 As New TextField()
textFld1.Name = "FldTitle"
textFld1.Text = "Employees Report"
textFld1.Left = 0
textFld1.Top = 0
textFld1.Width = 8000
textFld1.Height = 1440
textFld1.Font.Size = 24
textFld1.Font.Bold = True
textFld1.ForeColor = Color.FromArgb(0, 0, 100)
C1FlexReport1.Sections.Header.Fields.Add(textFld1)
```

C#

```
// create a report header
Section s = c1FlexReport1.Sections[SectionTypeEnum.Header];
s.Height = 1440;
s.Visible = true;
s.BackColor = Color.FromArgb(200, 200, 200);
TextField textFld1 = new TextField();
textFld1.Name = "FldTitle";
textFld1.Text = "Employees Report";
textFld1.Left = 0;
textFld1.Top = 0;
textFld1.Width = 8000;
textFld1.Height = 1440;
textFld1.Font.Size = 24;
textFld1.Font.Bold = true;
textFld1.ForeColor = Color.FromArgb(0, 0, 100);
c1FlexReport1.Sections.Header.Fields.Add(textFld1);
```

The section object has a Fields collection. The collection's **Add** method creates a new field and assigns it to the Section. The parameters specify the new field's Name, Text, Left, Top, Width, and Height properties. By default, the field has the same font as the control. Since this is a title, it makes sense to change the font and make it larger. Note that the field should be tall enough to accommodate the font size, or nothing will appear in it.

6. Next, set up the Page Footer Section. This section is more interesting because it contains expressions in the text fields. To evaluate the expressions in the text field, simply set the expression in the **TextField.Text.Expression** property. The expression in the text field is evaluated when the report is rendered. In C1Report, same is achieved by setting the field's Calculated property to **True**. To create a page footer, add the following code:

VB

```
' create a page footer
s = C1FlexReport1.Sections(SectionTypeEnum.PageFooter)
s.Height = 500
s.Visible = True

Dim textFld2 As New TextField()
textFld2.Name = "FldFtrLeft"
textFld2.Text.Expression = ""
"Employees: Printed on "
" & Now"
textFld2.Left = 0
textFld2.Top = 0
textFld2.Width = 4000
textFld2.Height = 300
textFld2.Font.Size = 8
textFld2.Font.Bold = False
C1FlexReport1.Sections.PageFooter.Fields.Add(textFld2)

Dim textFld3 As New TextField()
textFld3.Name = "FldFtrRight"
textFld3.Text.Expression = ""
"Page "
" + Page + "
" of "
" & Pages"
textFld3.Left = 4000
textFld3.Top = 0
textFld3.Width = 4000
textFld3.Height = 300
textFld3.Align = FieldAlignEnum.RightTop
textFld3.Width = C1FlexReport1.Layout.Width - textFld3.Left
C1FlexReport1.Sections.PageFooter.Fields.Add(textFld3)


'add a line before page footer
Dim shpfld As New ShapeField()
shpfld.Name = "FldLine"
shpfld.ShapeType = ShapeType.Line
shpfld.Left = 0
shpfld.Top = 0
shpfld.Width = C1FlexReport1.Layout.Width
shpfld.Height = 20
C1FlexReport1.Sections.PageFooter.Fields.Add(shpfld)
```

C#

```
// create a page footer
s = c1FlexReport1.Sections[SectionTypeEnum.PageFooter];
s.Height = 500;
s.Visible = true;

TextField textFld2 = new TextField();
textFld2.Name = "FldFtrLeft";
```

```
textFld2.Text.Expression = @ ""
"Employees: Printed on "
" & Now";
textFld2.Left = 0;
textFld2.Top = 0;
textFld2.Width = 4000;
textFld2.Height = 300;
textFld2.Font.Size = 8;
textFld2.Font.Bold = false;
c1FlexReport1.Sections.PageFooter.Fields.Add(textFld2);

TextField textFld3 = new TextField();
textFld3.Name = "FldFtrRight";
textFld3.Text.Expression = @ ""
"Page "
" + Page + "
" of "
" & Pages";
textFld3.Left = 4000;
textFld3.Top = 0;
textFld3.Width = 4000;
textFld3.Height = 300;
textFld3.Align = FieldAlignEnum.RightTop;
textFld3.Width = c1FlexReport1.Layout.Width - textFld3.Left;
c1FlexReport1.Sections.PageFooter.Fields.Add(textFld3);

//add a line before page footer
ShapeField shpfld = new ShapeField();
shpfld.Name = "FldLine";
shpfld.ShapeType = ShapeType.Line;
shpfld.Left = 0;
shpfld.Top = 0;
shpfld.Width = c1FlexReport1.Layout.Width;
shpfld.Height = 20;
c1FlexReport1.Sections.PageFooter.Fields.Add(shpfld);
```

The Page Footer section uses expressions with variables that are not intrinsic to VBScript, but are defined by C1FlexReport.Page and **Pages** are variables that contain the current page number and the total page count. The section also uses a field configured to look like a line.

7. Next, set up the Page Header Section. This section gets rendered at the top of every page and will display the field labels. Using a Page Header section to display field labels is a common technique in tabular reports. The code is simple, but looks a bit messy because of all the field measurements. In a real application, these values would not be hard-wired into the program. To create a page header with field labels, add the following code:

Visual Basic

```
'create a page header with field labels
s = C1FlexReport1.Sections(SectionTypeEnum.PageHeader)
s.Height = 500
s.Visible = True
C1FlexReport1.Font.Bold = True

Dim textFld4 As New TextField()
textFld4.Name = "LblID"
```

```
textFld4.Text = "ID"
textFld4.Left = 0
textFld4.Top = 50
textFld4.Width = 400
textFld4.Height = 300
textFld4.Align = FieldAlignEnum.RightTop
C1FlexReport1.Sections.PageHeader.Fields.Add(textFld4)

Dim textFld5 As New TextField()
textFld5.Name = "LblFirstName"
textFld5.Text = "First"
textFld5.Left = 500
textFld5.Top = 50
textFld5.Width = 900
textFld5.Height = 300
C1FlexReport1.Sections.PageHeader.Fields.Add(textFld5)

Dim textFld6 As New TextField()
textFld6.Name = "LblLastName"
textFld6.Text = "Last"
textFld6.Left = 1500
textFld6.Top = 50
textFld6.Width = 900
textFld6.Height = 300
C1FlexReport1.Sections.PageHeader.Fields.Add(textFld6)

Dim textFld7 As New TextField()
textFld7.Name = "LblTitle"
textFld7.Text = "Title"
textFld7.Left = 2500
textFld7.Top = 50
textFld7.Width = 2400
textFld7.Height = 300
C1FlexReport1.Sections.PageHeader.Fields.Add(textFld7)

Dim textFld8 As New TextField()
textFld8.Name = "LblTitle"
textFld8.Text = "Notes"
textFld8.Left = 5000
textFld8.Top = 50
textFld8.Width = 8000
textFld8.Height = 300
C1FlexReport1.Sections.PageHeader.Fields.Add(textFld8)
C1FlexReport1.Font.Bold = False

Dim shpfld2 As New ShapeField()
shpfld2.Name = "FldLine"
shpfld2.ShapeType = ShapeType.Line
shpfld2.Left = 0
shpfld2.Top = 400
shpfld2.Width = C1FlexReport1.Layout.Width
shpfld2.Height = 20
C1FlexReport1.Sections.PageHeader.Fields.Add(shpfld2)
```

C#

```
//create a page header with field labels
s = c1FlexReport1.Sections[SectionTypeEnum.PageHeader];
s.Height = 500;
s.Visible = true;
c1FlexReport1.Font.Bold = true;

TextField textFld4 = new TextField();
textFld4.Name = "LblID";
textFld4.Text = "ID";
textFld4.Left = 0;
textFld4.Top = 50;
textFld4.Width = 400;
textFld4.Height = 300;
textFld4.Align = FieldAlignEnum.RightTop;
c1FlexReport1.Sections.PageHeader.Fields.Add(textFld4);

TextField textFld5 = new TextField();
textFld5.Name = "LblFirstName";
textFld5.Text = "First";
textFld5.Left = 500;
textFld5.Top = 50;
textFld5.Width = 900;
textFld5.Height = 300;
c1FlexReport1.Sections.PageHeader.Fields.Add(textFld5);

TextField textFld6 = new TextField();
textFld6.Name = "LblLastName";
textFld6.Text = "Last";
textFld6.Left = 1500;
textFld6.Top = 50;
textFld6.Width = 900;
textFld6.Height = 300;
c1FlexReport1.Sections.PageHeader.Fields.Add(textFld6);

TextField textFld7 = new TextField();
textFld7.Name = "LblTitle";
textFld7.Text = "Title";
textFld7.Left = 2500;
textFld7.Top = 50;
textFld7.Width = 2400;
textFld7.Height = 300;
c1FlexReport1.Sections.PageHeader.Fields.Add(textFld7);

TextField textFld8 = new TextField();
textFld8.Name = "LblTitle";
textFld8.Text = "Notes";
textFld8.Left = 5000;
textFld8.Top = 50;
textFld8.Width = 8000;
textFld8.Height = 300;
c1FlexReport1.Sections.PageHeader.Fields.Add(textFld8);
c1FlexReport1.Font.Bold = false;

ShapeField shpfld2 = new ShapeField();
```

```
shpfld2.Name = "FldLine";
shpfld2.ShapeType = ShapeType.Line;
shpfld2.Left = 0;
shpfld2.Top = 400;
shpfld2.Width = c1FlexReport1.Layout.Width;
shpfld2.Height = 20;
c1FlexReport1.Sections.PageHeader.Fields.Add(shpfld2);
```

This code illustrates a powerful technique for handling fonts. Since every field inherits the control font when it is created, set the control's **Font.Bold** property to **True** before creating the fields, and set it back to **False** afterwards. As a result, all controls in the Page Header section have a bold font.

8. To finalize the report, add the Detail Section. This is the section that shows the actual data. It contains expressions in the text fields below each label in the Page Header Section. To create the Detail section, add the following code:

Visual Basic

```
' create the Detail section
s = C1FlexReport1.Sections(SectionTypeEnum.Detail)
s.Height = 330
s.Visible = True

Dim textField As New TextField()
textField.Name = "FldID"
textField.Text.Expression = "EmployeeID"
textField.Left = 0
textField.Top = 0
textField.Width = 400
textField.Height = 300
C1FlexReport1.Sections.Detail.Fields.Add(textField)

Dim textField1 As New TextField()
textField1.Name = "FldFirstName"

textField1.Text.Expression = "FirstName"
textField1.Left = 500
textField1.Top = 0
textField1.Width = 900
textField1.Height = 300
C1FlexReport1.Sections.Detail.Fields.Add(textField1)

Dim textField2 As New TextField()
textField2.Name = "FldLastName"
textField2.Text.Expression = "LastName"
textField2.Left = 1500
textField2.Top = 0
textField2.Width = 900
textField2.Height = 300
C1FlexReport1.Sections.Detail.Fields.Add(textField2)

Dim textField3 As New TextField()
textField3.Name = "FldTitle"
textField3.Text.Expression = "Title"
textField3.Left = 2500
textField3.Top = 0
```

```
textField3.Width = 2400
textField3.Height = 300
C1FlexReport1.Sections.Detail.Fields.Add(textField3)

Dim textField4 As New TextField()
textField4.Name = "FldNotes"
textField4.Text.Expression = "Notes"
textField4.Left = 5000
textField4.Top = 0
textField4.Width = 8000
textField4.Height = 300
C1FlexReport1.Sections.Detail.Fields.Add(textField4)

textField4.Width = C1FlexReport1.Layout.Width - textField4.Left
textField4.AutoHeight = AutoSizeBehavior.CanGrow
textField4.Font.Size = 6
textField4.Align = FieldAlignEnum.JustTop

  'add a line after each field in detail section
Dim shpfld3 As New ShapeField()
shpfld3.Name = "FldLine"
shpfld3.ShapeType = ShapeType.Line
shpfld3.Left = 0
shpfld3.Top = 310
shpfld3.Width = C1FlexReport1.Layout.Width
shpfld3.Height = 20
C1FlexReport1.Sections.Detail.Fields.Add(shpfld3)
```

C#

```
// create the Detail section
s = c1FlexReport1.Sections[SectionTypeEnum.Detail];
s.Height = 330;
s.Visible = true;

TextField textField = new TextField();
textField.Name = "FldID";
textField.Text.Expression = "EmployeeID";
textField.Left = 0;
textField.Top = 0;
textField.Width = 400;
textField.Height = 300;
c1FlexReport1.Sections.Detail.Fields.Add(textField);

TextField textField1 = new TextField();
textField1.Name = "FldFirstName";

textField1.Text.Expression = "FirstName";
textField1.Left = 500;
textField1.Top = 0;
textField1.Width = 900;
textField1.Height = 300;
c1FlexReport1.Sections.Detail.Fields.Add(textField1);

TextField textField2 = new TextField();
```

```
textField2.Name = "FldLastName";
textField2.Text.Expression = "LastName";
textField2.Left = 1500;
textField2.Top = 0;
textField2.Width = 900;
textField2.Height = 300;
c1FlexReport1.Sections.Detail.Fields.Add(textField2);

TextField textField3 = new TextField();
textField3.Name = "FldTitle";
textField3.Text.Expression = "Title";
textField3.Left = 2500;
textField3.Top = 0;
textField3.Width = 2400;
textField3.Height = 300;
c1FlexReport1.Sections.Detail.Fields.Add(textField3);

TextField textField4 = new TextField();
textField4.Name = "FldNotes";
textField4.Text.Expression = "Notes";
textField4.Left = 5000;
textField4.Top = 0;
textField4.Width = 8000;
textField4.Height = 300;
c1FlexReport1.Sections.Detail.Fields.Add(textField4);

textField4.Width = c1FlexReport1.Layout.Width - textField4.Left;
textField4.AutoHeight = AutoSizeBehavior.CanGrow;
textField4.Font.Size = 6;
textField4.Align = FieldAlignEnum.JustTop;

//add a line after each field in detail section
ShapeField shpfld3 = new ShapeField();
shpfld3.Name = "FldLine";
shpfld3.ShapeType = ShapeType.Line;
shpfld3.Left = 0;
shpfld3.Top = 310;
shpfld3.Width = c1FlexReport1.Layout.Width;
shpfld3.Height = 20;
c1FlexReport1.Sections.Detail.Fields.Add(shpfld3);
```

Note that all text fields contain expressions and each text field corresponds to the names of fields in the source record source. Setting the expressions in the **TextField.Text.Expression** property ensures that the Text property is interpreted as a database field name, as opposed to being rendered literally. It is important to adopt a naming convention for report fields that makes them unique, different from recordset field names. If you had two fields named "LastName", an expression such as "Left(LastName,1)" would be ambiguous. This example has adopted the convention of beginning all report field names with "Fld".

Note that the "FldNotes" field has its AutoHeight property set to **CanGrow**, and a smaller font than the others. This was done to make sure that the "Notes" field in the database, which contains a lot of text, will appear in the report. Rather than make the field very tall and waste space, setting the **AutoHeight** property to CanGrow tells the control to expand the field as needed to fit its contents; it also sets the containing section's **AutoHeight** property to True, so the field doesn't spill off the Section.

9. The report definition is done. To render the report into the **FlexViewer** control, double-click the **Employees** button to add an event handler for the **btnEmployees_Click** event. The Code Editor will open with the insertion point placed

within the event handler. Enter the following code:

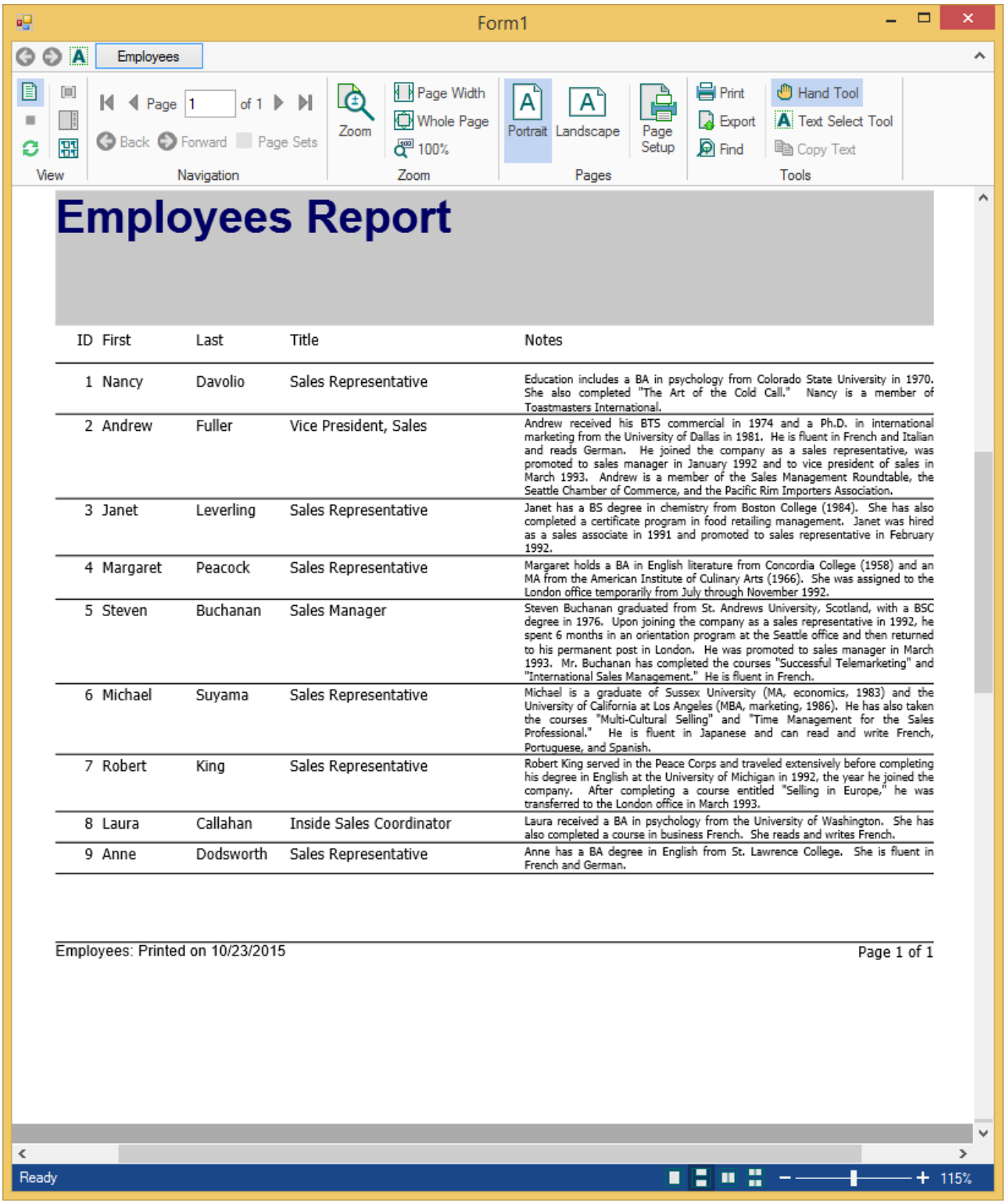
Visual Basic

```
RenderEmployees()
C1FlexViewer1.DocumentSource = C1FlexReport1
```

C#

```
RenderEmployees();
c1FlexViewer1.DocumentSource = c1FlexReport1;
```

Here's what the basic report looks like:



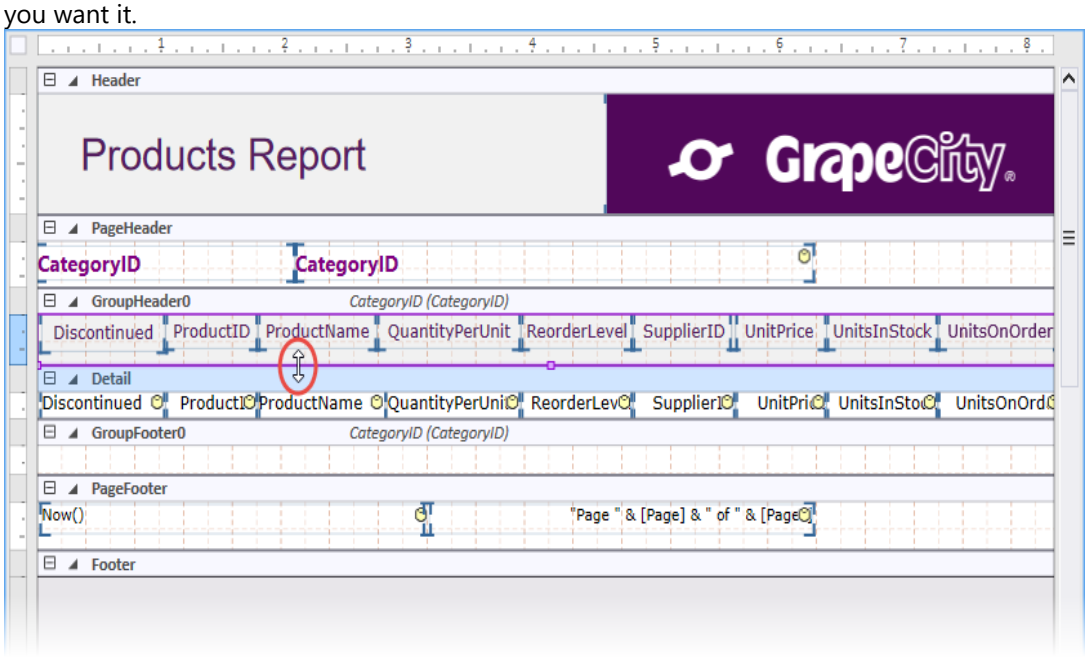
Step 2 of 4: Modifying the Report

With the **FlexReportDesigner** in preview mode, you cannot make any adjustments to the report. Click the **Design** button to switch to Design mode and begin making modifications. The right pane of the main window switches from Preview mode to Design mode, and it shows the controls and fields that make up the report.

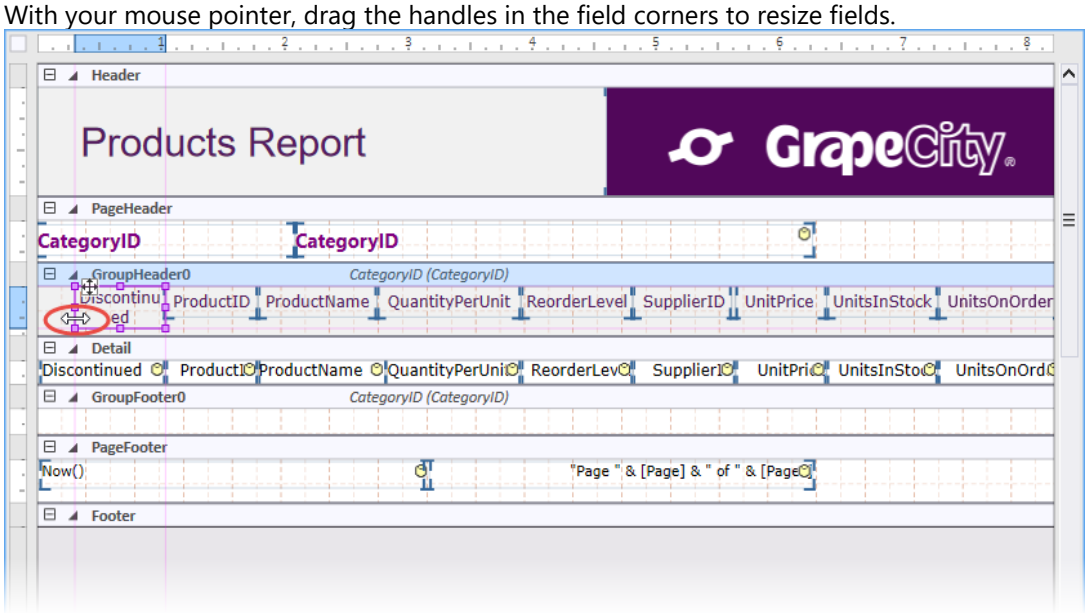
Modify the Report:

For this example, we will resize the Group Header section and fields as well as format a field value. To do this, complete the following steps:

1. To resize the Group Header section, select its border and with your mouse pointer drag to the position where you want it.

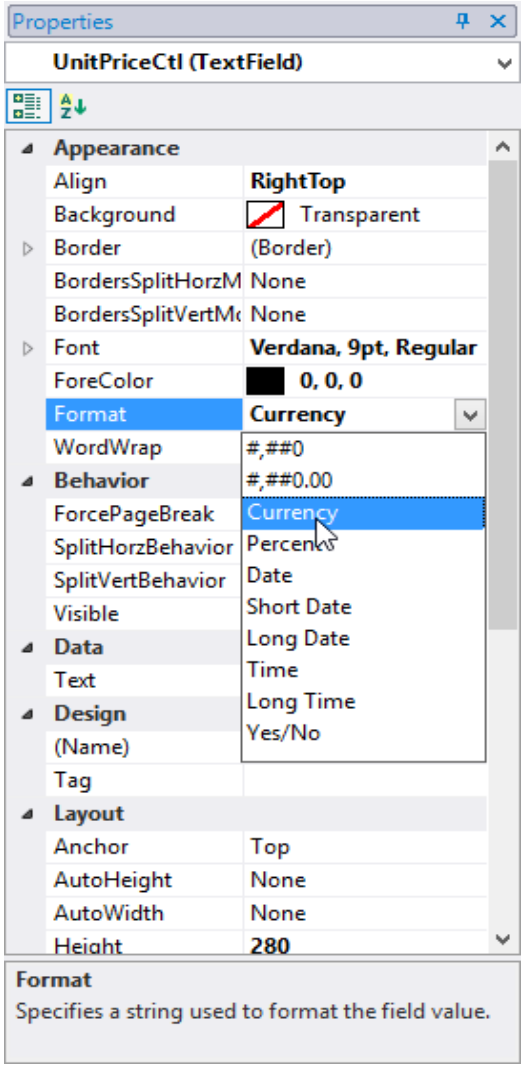


2. With your mouse pointer, drag the handles in the field corners to resize fields.

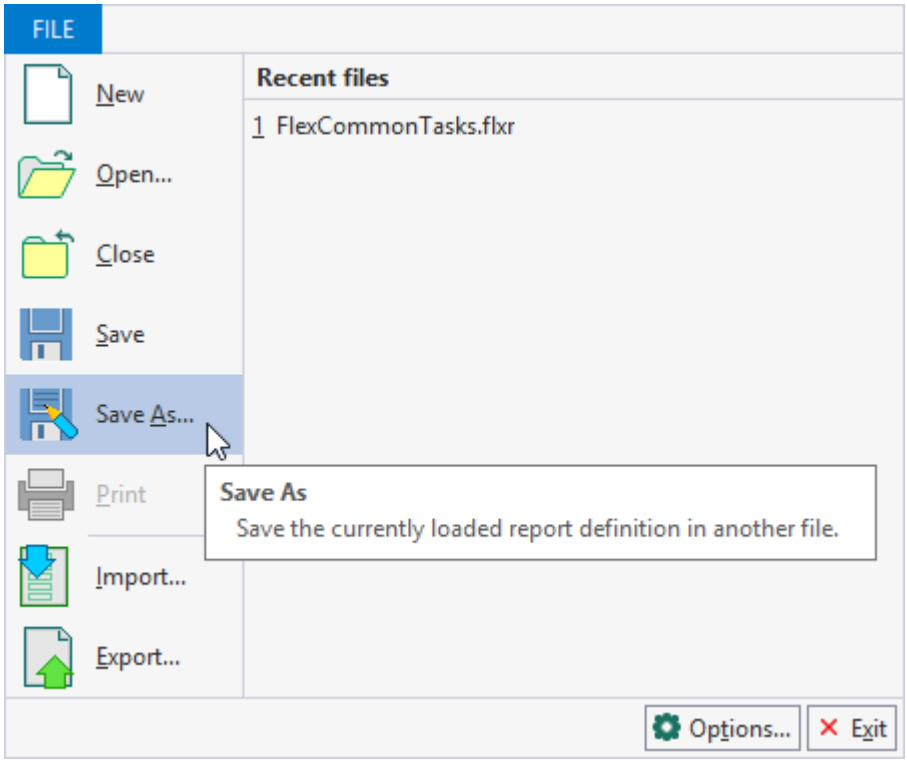


> **Tip**: If text is not fitting in the field, set the **Appearance.WordWrap** property for the field to True in the Properties window.

3. In the Properties window, select the **UnitPriceCtl** field in Detail section under the *Unit Price* column.
4. Set the **Appearance.Format** property for the field to **Currency**.



5. Click the **Preview** button to switch to Preview mode and see your modifications.
6. Click **Design** button to switch from Preview mode to Design mode.
7. Click the **File** menu and select **Save As** from the menu that appears.



8. In the **Save Report Definition File** dialog box, enter **ProductsReport.flxr** in the **File name** box. Save the file to a location that you will remember for later use.
9. Close the **Designer** and return to your Visual Studio project.

You have successfully created a report definition file; in the next step you will load the report in the **C1FlexReport** component.

Step 3 of 4: Loading the Report in the C1FlexReport Component

To load a report definition from a file at design time, complete one of the following tasks:

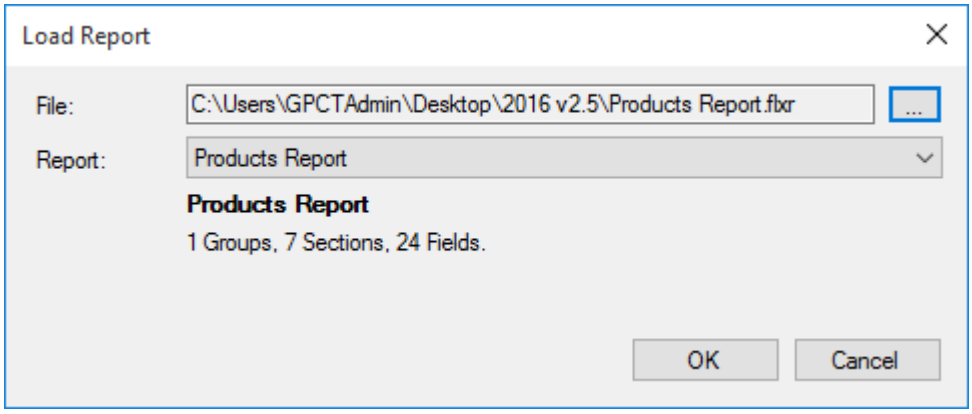
- Right-click the C1FlexReport component and select the **Load Report** menu option.

  OR

- Click the smart tag (▶) above the **C1FlexReport** component and select **Load Report** from the **C1FlexReport Tasks** menu.

Using the **Load Report** dialog box to select the report you want, complete the following tasks:

1. Click the **ellipses** button. The **Open** dialog box appears.
2. Browse to the location that you just saved your **Products Report.flxr** file, select it, and click **Open**.
3. The available report definitions are listed in the **Report** drop-down box. Select the **Products Report** definition to load.
4. Click **Load** and **OK** to close the dialog box.



In the next step you will render the report into a preview control.

Step 4 of 4: Rendering the Report

Once the report definition has been created, a data source defined and loaded into the C1FlexReport component, you can render the report to the printer, to the preview control - C1FlexViewer, or export to different file formats.

To preview the report in the FlexViewer control, the steps are as follows:

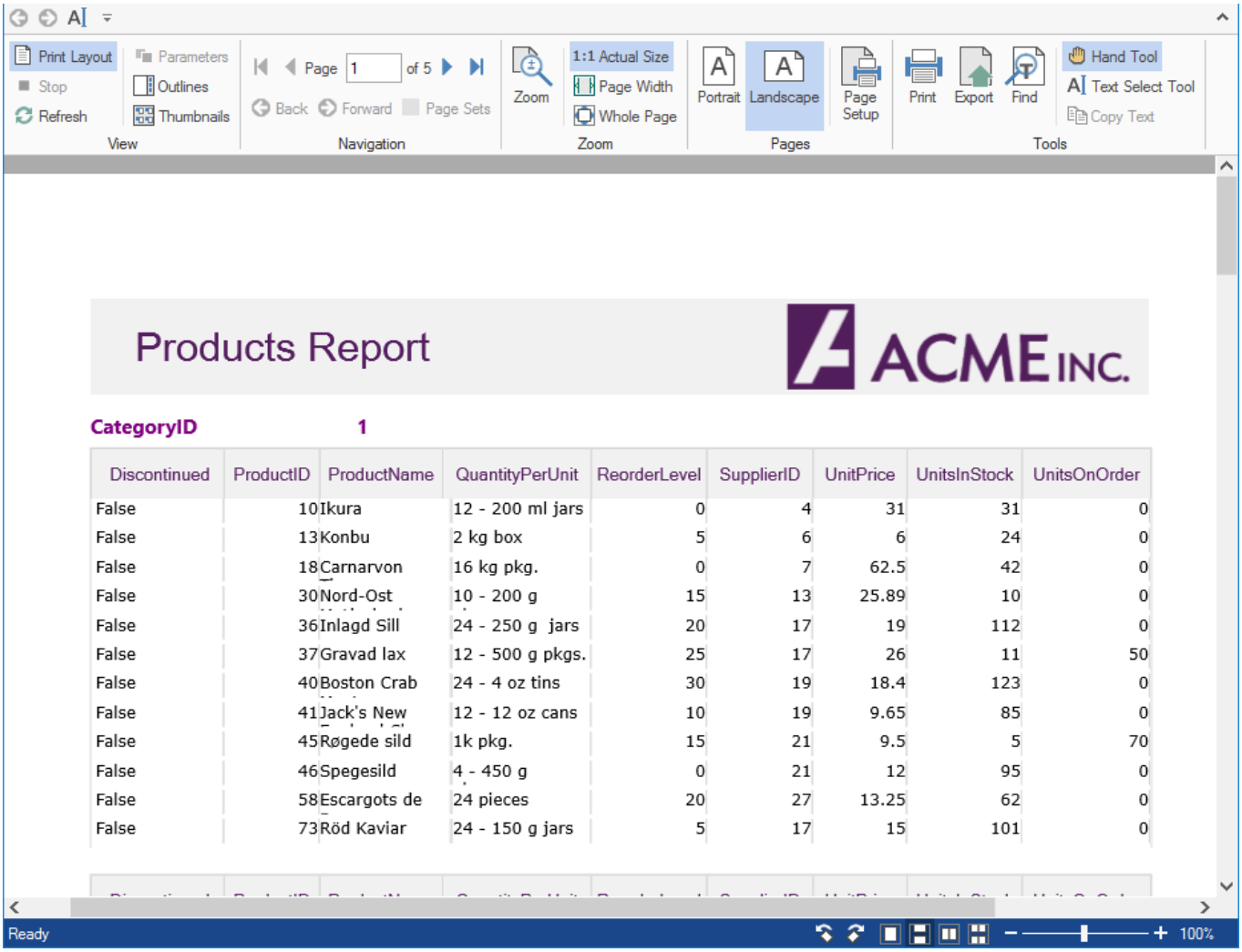
1. From the **Toolbox**, double-click the **C1FlexViewer** control to add it to your project.
2. From the Properties window, set the **C1FlexViewer.Dock** property to Fill.
3. Select the Windows Form with your mouse and drag the corners to resize it.
4. Double-click the form and enter the following code in the Form1_Load event handler:

Visual Basic

```
'load report definition
c1FlexReport1.Load("..\..\Products Report.flxr", "Products Report")
'preview the report
c1FlexViewer1.DocumentSource = c1FlexReport1
```

- **C#**

```
//load report definition
c1FlexReport1.Load(@"..\..\Products Report.flxr", "Products Report");
//preview the report
c1FlexViewer1.DocumentSource = c1FlexReport1;
```



Design-Time Support

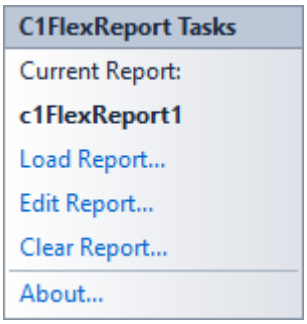
You can easily configure **FlexReport for WinForms** at design time using the property grid, menus, and designers in Visual Studio. The following sections describe how to use design-time environment for configuring **FlexReport** and **FlexViewer**.

C1FlexReport Tasks Menu

In Visual Studio, the C1FlexReport component includes a smart tag. A smart tag represents a short-cut tasks menu that provides the most commonly used properties.

The C1FlexReport component provides quick and easy access to the **FlexReport Wizard** (for reports definitions that have not been created) or the **FlexReportDesigner** (for report definitions that already exist in the project), as well as loads reports through its smart tag.

To access the **C1FlexReport Tasks** menu, click the smart tag (▶) in the upper-right corner of the C1FlexReport component.



The **C1FlexReport Tasks** menu operates as follows:

- **Current Report:**

  Shows the name of the current report, specified in the **ReportName** property.

- **Load Report**

  Clicking **Load Report** opens the **Load Report** dialog box. For more information about loading a report, see Load FlexReport at Design Time.

- **Edit Report**

  Clicking **Edit Report** opens the **FlexReport Wizard** if you have not already created a report definition or the **C1FlexReportDesigner** if you have already created a report.

  For more information on using the **FlexReport Wizard**, see Step 1 of 4: Creating a Report Definition. For details on using the **C1FlexReportDesigner**, see Working with FlexReportDesigner.

- **Clear Report**

  Clears the report loaded in C1FlexReport. On clicking Clear Report, user is asked - 'Are you sure to clear Report [Report Name]?'.

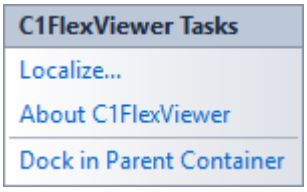
- **About**

  Clicking **About** displays the **About** dialog box, which is helpful in finding the version number of **C1.Win.FlexReport**, as well as information about licensing and online resources.

C1FlexViewer Tasks Menu

In **C1FlexViewer Tasks** menu, you can quickly and easily dock the FlexViewer control in the parent container and access the **Localize** dialog box.

To access the **FlexViewer Tasks** menu, click the smart tag (▶) in the upper right corner of the control.



The **C1FlexViewer Tasks** menu operates as follows:

- **Localize**

  Clicking **Localize** opens the **Localize** dialog box. In the **Localize** dialog box, you can customize your localization settings.

- **About FlexViewer**

  Clicking **About** displays the control's **About** dialog box, which is helpful in finding the build version of the control.

- **Dock in Parent Container/Undock in Parent Container**

  Clicking **Dock in Parent Container** sets the **Dock** property for **C1FlexViewer** to **Fill**.

  If C1FlexViewer is docked in the parent container, the option to undock it from the parent container will be available. **Clicking Undock in Parent Container** sets the **Dock** property for **C1FlexViewer** to **None**.

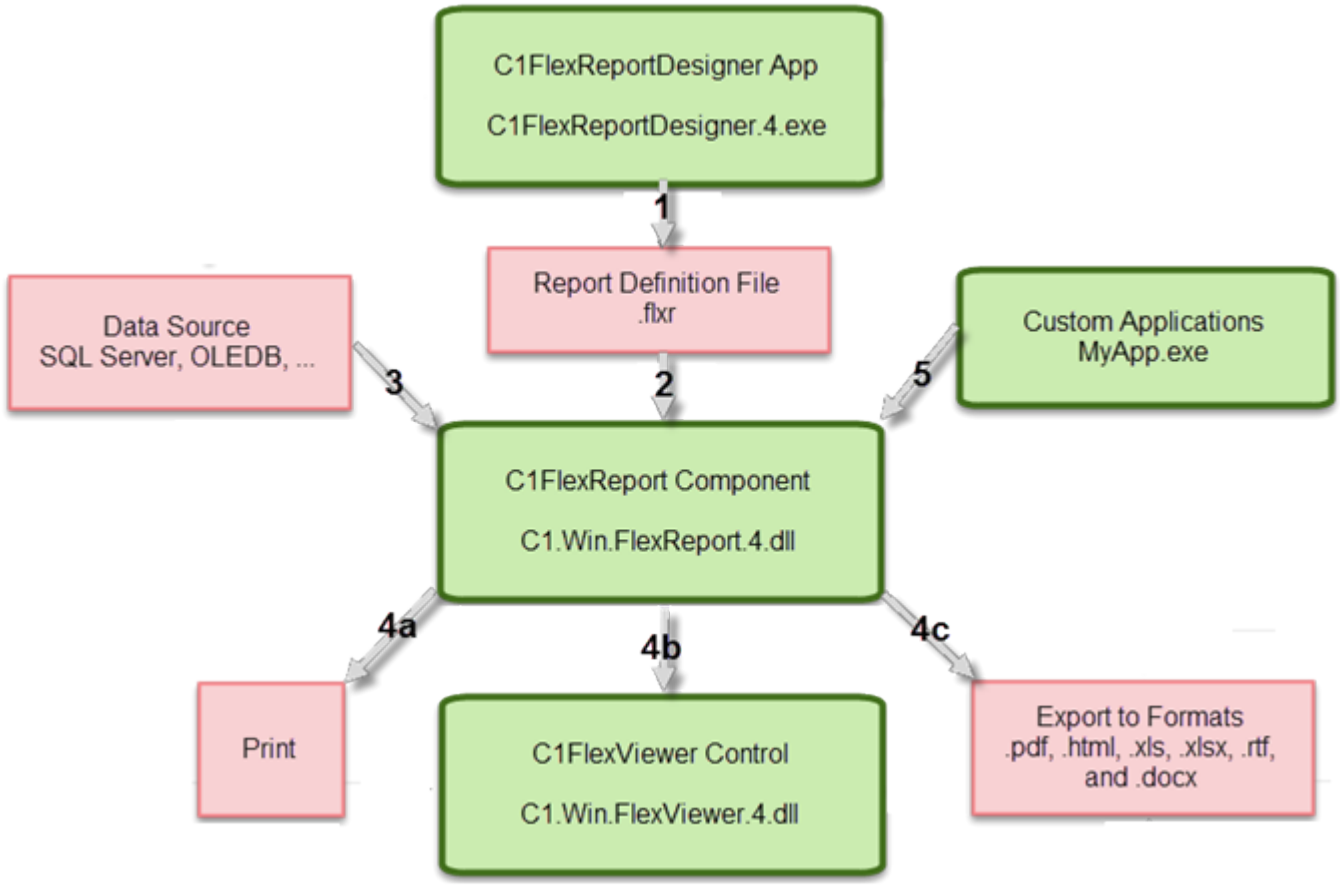
Working with FlexReport

The main sequence of working with C1FlexReport is as follows:

1. You start by creating a report using the **FlexReportDesigner** application to create report definitions; report definitions are saved as .flxr files, and can be designed from scratch or imported from existing Microsoft Access Reports and Crystal Reports. You can then modify the basic report using the designer.
2. The **C1FlexReport** component reads the report definitions and renders the reports using data from any standard .NET data source.
3. The report definitions can be loaded at design time, and embedded in your application, or they can be read and modified at run time. (You can even create report definitions from scratch, using the C1FlexReport object model.)
4. Reports can be rendered directly to a printer, into a **FlexViewer** control, or exported to different formats.

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

> **Note:** Boxes with a bold border represent code components (controls and applications). Boxes with a thin border represent files containing information (report definitions, data, and finished reports).



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

1. Use the **C1FlexReportDesigner** application to create, edit, and save FLXR report definition files.
2. The **C1FlexReport** component loads report definitions from the FLXR files created with the designer. This can be done at design time (in this case the FLXR file is persisted with the control and not needed at run time) or at run time using the Load method.
3. The C1FlexReport component loads data from the data source specified in the report definition file. Alternatively, you can provide your own custom data source.

4. The C1FlexReport component formats the data according to the report definition and renders reports to a (a) printer, (b) to a previewing control, (c) or to one of several file formats.
5. Custom applications can communicate with the C1FlexReport component using a rich object model, so you can easily customize your reports or generate entirely new ones. **C1FlexReportDesigner** is a good example of such an application.

Note: To work with FlexReport the operating system must support NT 6.1 core. Therefore, the minimal system requirements are Windows 7 SP1 or Windows Server 2008 R2 SP1 with Platform Update (KB2670838).

C1FlexReport and C1Document

C1FlexReport uses the **C1.Win.C1Document** assembly, that further exposes the following two classes, which are base classes for C1FlexReport and C1SsrsDocumentSource:

- **C1Document:** represents ComponentOne document. Its main functions are as follows:
  - Persists the document as a RenderObjects tree.
  - Renders the document content.
  - Highlights the Text selection.
  - Checks a specific location of an object with respect to other elements through methods such as HitTest.

  C1Document is similar to our WinForms' C1Framework library. It is a set of rather low level utility classes used by other components. Currently it is used by **C1SsrsDocumentSource**, **C1SsrsViewer**, and **C1FlexReport**.

- **C1DocumentSource:** provides functionality to work with different documents and report types. It is a base class to build C1Document objects. Its main functions are as follows:
  - Provides infrastructure for asynchronous rendering of documents.
  - Provides parameter support while generating the document.
  - Provides support for text search in the generated document.

  C1FlexReport is derived from **C1DocumentSource**. It uses **C1DocumentSource** to provide asynchronous rendering, parameter support and text search. This also means that it will be easy to port C1FlexReport to other platforms (XAML).

C1Document Breaking Changes

With the development of FlexReport, there are some breaking changes that took place in C1.Win.C1Document.C1DocumentSource and C1.Win.C1Document.C1SSRSDocumentSource. You should be aware of the following breaking changes while using **C1.Win.C1Document.C1DocumentSource** and **C1.Win.C1Document.C1SSRSDocumentSource**.

- All OpenXXX methods have been removed:
  - public void Open(C1DocumentLocation documentLocation);
  - public IAsyncActionWithProgress<double> OpenAsyncEx(C1DocumentLocation documentLocation);
  - public IAsyncActionWithProgress<double> OpenAsyncEx();
  - public void Open();
  - public Task OpenAsync();
  - public Task OpenAsync(C1DocumentLocation documentLocation);

  Now C1SSRSDocumentSource.Generate() can be used immediately after defining all necessary properties like **DocumentLocation**, **ConnectionOptions**, etc.

- All GenerateXXX methods are now accessible only in C1SSRSDocumentSource; in C1DocumentSource these

methods are internal:
  - public void Generate();
  - public IAsyncActionWithProgress<double> GenerateAsyncEx();
  - public Task GenerateAsync();
- All ApplyParameterValuesXXX and CheckParameterValuesXXX methods have been removed:
  - public List<ParameterValueError> ApplyParameterValues();
  - public IAsyncOperationWithProgress<List<ParameterValueError>, double> ApplyParameterValuesAsyncEx();
  - public Task<List<ParameterValueError>> ApplyParameterValuesAsync();
  - public List<ParameterValueError> CheckParameterValues();
  - public Task<List<ParameterValueError>> CheckParameterValuesAsync();
  - public IAsyncOperationWithProgress<List<ParameterValueError>, double> CheckParameterValuesAsyncEx();

  Use C1SSRSDocumentSource.ValidateParameters(...) instead. The values of parameters are applied automatically before generating.
- Methods ExecuteCustomActionXXX, GetPageXXX and GetBookmarkPositionXXX() are now internal and inaccessible:
  - public IAsyncOperationWithProgress<C1BookmarkPosition, double> ExecuteCustomActionAsyncEx(CustomAction action);
  - public Task<C1BookmarkPosition> ExecuteCustomActionAsync(CustomAction action);
  - public C1Page GetPage(int pageIndex);
  - public Task<C1Page> GetPageAsync(int pageIndex);
  - public IAsyncOperationWithProgress<C1Page, double> GetPageAsyncEx(int pageIndex);
  - public virtual C1Page GetLoadedPage(int pageIndex);
  - public bool IsPageLoading(int pageIndex);
  - public C1BookmarkPosition GetBookmarkPosition(string bookmark);
  - public IAsyncOperationWithProgress<C1BookmarkPosition, double> GetBookmarkPositionAsyncEx(string bookmark);
  - public Task<C1BookmarkPosition> GetBookmarkPositionAsync(string bookmark);
- All functionality related to text searching has been moved to C1FlexViewer. In addition, following methods and properties have been removed:
  - public C1FoundPosition FindTextStart(int startPageIndex, bool wholeDocument, C1FindTextParams findParams);
  - public IAsyncOperationWithProgress<C1FoundPosition, double> FindTextStartAsyncEx(int startPageIndex, bool wholeDocument, C1FindTextParams findParams);
  - public Task<C1FoundPosition> FindTextStartAsync(int startPageIndex, bool wholeDocument, C1FindTextParams findParams);
  - public C1FoundPosition FindTextNext(C1FoundPosition foundPosition);
  - public IAsyncOperationWithProgress<C1FoundPosition, double> FindTextNextAsyncEx(C1FoundPosition foundPosition);
  - public Task<C1FoundPosition> FindTextNextAsync(C1FoundPosition foundPosition);
  - public C1FoundPosition FindTextPrevious(C1FoundPosition foundPosition);
  - public IAsyncOperationWithProgress<C1FoundPosition, double> FindTextPreviousAsyncEx(C1FoundPosition foundPosition);
  - public Task<C1FoundPosition> FindTextPreviousAsync(C1FoundPosition foundPosition);
  - public void FindTextReset();
  - public C1HighlightAttrs FindMatchHighlight { get; set; }
  - public C1HighlightAttrs ActiveFindMatchHighlight { get; set; }
  - public IList<C1FoundPosition> FoundPositions { get; }
  - public C1FoundPosition ActiveFoundPosition { get; set; }
- The following properties have been removed (text selection functionality has been moved to viewers):
  - public C1DocumentRange SelectedRange { get; set; }
  - public C1HighlightAttrs SelectionHighlight { get; set; }
- The property - C1DocumentSource.State { get; set; } and enum - C1DocumentSourceState have been removed.

Instead, use C1DocumentSource.BusyState property as follows:

```
public C1DocumentSourceBusyState BusyState { get; }

// Summary:
// Describes the busy state of a C1DocumentSource object.
public enum C1DocumentSourceBusyState
{
// Summary:
// The document is ready (not busy).
Ready,
// Summary:
// The document is currently generating.
Generating,
// Summary:
// The document is currently exporting.
Exporting,
// Summary:
// The document is currently printing.
Printing,
}
```

- The following properties can be used to determine the current state of the C1DocumentSource:

```
// Summary:
//   Gets the value indicating whether the current C1DocumentSource busy.
public bool IsBusy { get; }
// Summary:
//   Gets a value indicating whether this C1DocumentSource is disposed and can not be longer used.
public bool IsDisposed { get; }
```

- The following events have been removed:
  - event EventHandler<ExecuteCustomActionCompletedEventArgs> ExecuteCustomActionCompleted;
  - event EventHandler<GetPageCompletedEventArgs> GetPageCompleted;
  - event EventHandler<GetLinkTargetPositionCompletedEventArgs> GetLinkTargetPositionCompleted;
  - event EventHandler PagesClear;

- The following events have been moved to C1SSRSDocumentSource:
  - event EventHandler<AsyncCompletedEventArgs> GenerateCompleted;
  - event EventHandler<ValidateParametersCompletedEventArgs> ValidateParametersCompleted;
  - event EventHandler<ExportCompletedEventArgs> ExportCompleted;
- The following methods are accessible only through C1SSRSDocumentSource():
  - public void Clear();
  - public void Cancel();
- Methods ValidateParameterXXX have been added to replace ApplyParameterValues and CheckParameterValues:
  - public List<ParameterValidationError> ValidateParameters();
  - Async variants:
    - public new IAsyncOperationWithProgress<List<ParameterValidationError>, double> ValidateParametersAsyncEx();
    - public Task<List<ParameterValidationError>> ValidateParametersAsync();

These methods validate the current parameter values and refresh their valid values lists if the values are valid. Note that parameter values are now applied automatically when a report generation starts.

Data Binding in FlexReport

In addition to a report definition, FlexReport needs the actual data to create the report. In most cases, the data comes from a database, but there are other options. The following topics explore how to retrieve data from other sources.

Retrieving Data from a Database

For retrieving or loading the report data in FlexReport, following DataSource properties of C1FlexReport should be set:

- ConnectionString
- RecordSource

To set data source:

Visual Basic

```
'initialize data source
         Dim ds As DataSource = C1FlexReport1.DataSource
        ds.ConnectionString = "Provider=Microsoft.Jet.OLEDB.4.0;Data
Source=C:\...\ComponentOne Samples\Common\C1Nwind.mdb;"
        ds.RecordSource = "Employees"
```

C#

```
//initialize DataSource
          DataSource ds = c1FlexReport1.DataSource;
          ds.ConnectionString = @"Provider=Microsoft.Jet.OLEDB.4.0; Data
Source=C:\...\ComponentOne Samples\Common\C1NWind.mdb;";
          ds.RecordSource = "Employees";
```

If these properties are set, C1FlexReport initializes the data source and uses them to load the data from the database automatically. This is same as initializing data source through the code or the designer as illustrated in Step 1 of 4: Creating a Report Definition.

Retrieving Data from a Stored Procedure

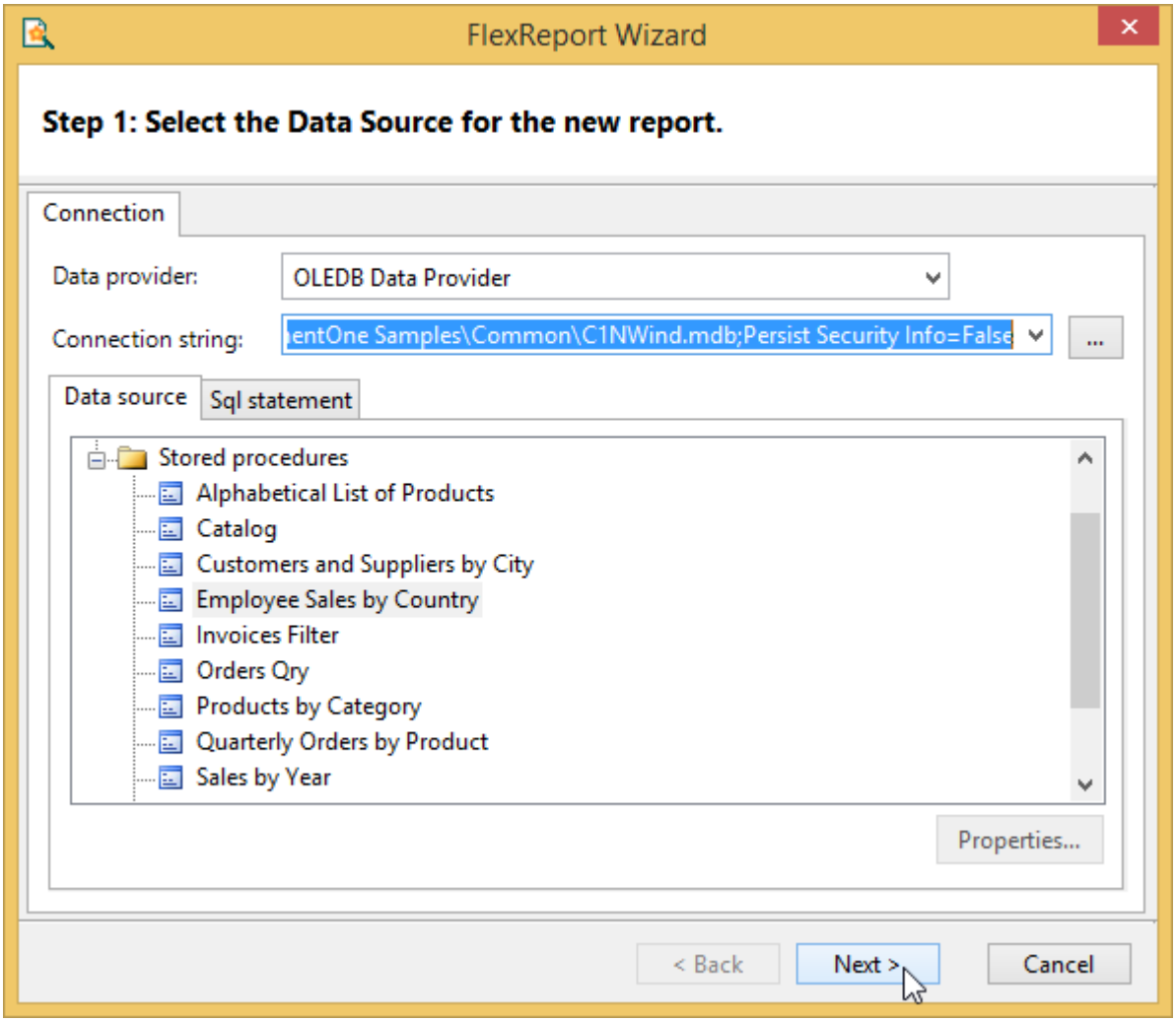
Stored procedures (or sprocs) can assist you in achieving a consistent implementation of logic across applications, improve performance, and shield users from needing to know the details of the tables in the database. One of the major advantages of stored procedures is you can pass in parameters to have the database filter the recordset. This returns a smaller set of data, which is quicker and easier for the report to manipulate.

You can populate a report from a stored procedure in the **C1FlexReport Wizard**. To open the **C1FlexReport Wizard** complete one of the following:

- In Visual Studio by selecting **Edit Report** from the **C1FlexReport** context menu
- In Visual Studio by selecting **Edit Report** from the **C1FlexReport Tasks** menu
- From the **C1FlexReportDesigner** application, click the **New Report** button from the **Reports** tab

For more information on accessing the **Edit Report** link, see Design-Time Support.

Populating a report from a stored procedure is no different than using SQL statements or straight tables. In the first screen of the **C1FlexReport Wizard**, click the **ellipses** button to choose a datasource. Then choose a **Stored Procedure** from the list of available **Data sources**:



Select **Next** and continue through the wizard.

As with loading other forms of data, you have two options:

- You can use the DataSource's ConnectionString and RecordSource properties to select the datasource:

  In the Designer, use the **DataSource** dialog box to select the connection string (by clicking the ellipses button "..."), then pick the table or sproc you want to use from the list. For example:

  ```
  connectionstring = "Provider=Microsoft.Jet.OLEDB.4.0;Data Source=C:\Users\Windows 8.1\Documents\ComponentOne Samples\Common\C1NWind.mdb;Persist Security Info=False;" RecordSource = "[Products by Category]"
  ```

  (In this case the stored procedure name has spaces, so it's enclosed in square brackets).

- You can create the data source using whatever method you want, then assign it to the DataSource's **Recordset** property:

  This method requires you to write code, and is useful when you have your data cached somewhere and want to use it to produce several reports. It overrides the previous method (if you specify **ConnectionString**, **RecordSource**, and Recordset, **C1FlexReport** uses the **Recordset**).

  The syntax is different depending on the type of connection/adapter you want to use (OleDb, SQL, Oracle, and so on). The easier way to get the syntax right is to drag tables or sprocs from Visual Studio's Server Explorer onto a form. This adds all the cryptic elements required, and then you can go over the code and pick up the pieces you want.

  You can specify stored procedures as data sources by their name. If the sproc has parameters, you pass them

as parameters. For example, in a report definition built against MSSQL and ADVENTURE_WORKS.mdf database, the SQL request that is specified in **C1FlexReportDesigner** (adjust the path to ADVENTUREWORKS_DATA.MDF as needed) is:

```
PARAMETERS Employee Int 290;
DECLARE @RC int
DECLARE @EmployeeID int
set @EmployeeID = [Empoyee]
EXECUTE @RC = [C:\ADVENTUREWORKS_DATA.MDF].[dbo].[uspGetEmployeeManagers]
@EmployeeID
```

Using Data Table Object as Data Source

Many applications need to work on the data outside **C1FlexReport** and load it into DataTable objects. In such cases, you can use DataTable objects as report data sources, avoiding the need to load them again while rendering the report.

This approach is also useful in applications where:

- Security restrictions dictate that connection strings must be kept private and only the data itself may be exposed (not its source).
- The database is not supported by OleDb (the provider used internally by C1FlexReport).
- The data does not come from a database at all. Instead, the DataTable is created and populated using custom code.

To use a DataTable object as a **C1FlexReport** data source, simply load the report definition and then assign the DataTable to C1FlexReport.DataSource.Recordset property. For example:

Visual Basic

```
' load DataTable from cache or from a secure/custom provider
Dim dt As DataTable = GetMyDataTable()

' load report definition (before setting the data source)
C1FlexReport1.Load(@"reportFile", "reportName")

' use DataTable as the data source
C1FlexReport1.DataSource.Recordset = dt
```

C#

```
// load DataTable from cache or from a secure/custom provider
DataTable dt = GetMyDataTable();

// load report definition (before setting the data source)
c1FlexReport1.Load(@"reportFile", "reportName");

// use DataTable as the data source
c1FlexReport1.DataSource.Recordset = dt;
```

Using Custom Data Source Objects

You can use custom objects as data sources. The only requirement is that the custom object must implement the **IC1FlexReportRecordset** interface.

IC1FlexReportRecordset is a simple and easy-to-implement interface that can be added to virtually any collection of data with ease. This is often more efficient than creating a **DataTable** object and copying all the data into it. For example, you could use custom data source objects to wrap a file system or custom .xml or .flxr files.

To use custom data source objects, load the report definition and then assign the object to the C1FlexReport's Recordset property. For example:

Visual Basic

```
' get custom data source object
Dim rs As IC1FlexReportRecordset = CType(GetMyCustomDataSource(),
IC1FlexReportRecordset)

' load report definition (before setting the data source)
C1FlexReport1.Load(@"reportFile", "reportName")

' use custom data source object in C1FlexReport component
C1FlexReport1.DataSource.Recordset = rs
```

C#

```
// get custom data source object
IC1FlexReportRecordset rs =
(IC1FlexReportRecordset)GetMyCustomDataSource();

// load report definition (before setting the data source)
c1FlexReport1.Load(@"reportFile", "reportName");

// use custom data source object in C1FlexReport component
c1FlexReport1.DataSource.Recordset = rs;
```

Data Sources in FlexReport

A FlexReport definition can include several data sources, which are accessible through C1FlexReport.DataSources collection. The data sources in this collection are identified by unique names. These data sources can be used as:

- **Main data source:** It is the main data source for a report. The main data source is specified by using C1FlexReport.DataSourceName property on the report. If the main data source is not specified (**DataSourceName** is empty or contains a name not found in the DataSources collection), C1FlexReport is rendered in unbound mode, containing a single instance of the Detail section.
- **Data source for Parameters:** It is the source of valid values for the report parameters (elements in the C1FlexReport.Parameters collection). The data source for parameters is specified using **ReportParameter.AllowedValuesDefinition.Binding.DataSourceName** property.
- **Data source for Charts:** It is the data source for the Chart field. The data source for charts is specified using ChartField.DataSource property.

The list of supported data source types in FlexReport are as follows:

- OLE DB
- ODBC
- XML
- Object in external assembly
- Microsoft SQL Server Compact Data Provider version 3.5 and 4.0

- SQLite

For backwards compatibility with C1Report, C1FlexReport has a DataSource property which points to DataSources[DataSourceName]. When a new C1FlexReport is created, a single element with the name 'Main' is added to its **C1FlexReport.DataSources** collection, and 'Main' is assigned to the C1FlexReport.DataSourceName property.

Note that in C1Report, Main data source is the only data source for the report.

Connecting to Multiple Data Sources using Code

You have learned how to create a report bound to a main data source in FlexReport Quick Start. Since a report can have multiple data sources, you should know how to connect to these data sources while using charts and parameters.

The following sections dive into how to bind data to charts and parameters in the reports with multiple data sources.

Binding Data to Charts in Multiple Data Source Report

When you add a **Chart** field to your report, the first step is to bind the chart to a data source.

Let's say your report has two data sources, 'Employees' and 'Products'. You want to create two charts, one that displays FullName and Age from Employees data source, and other that displays CategoryName and Sum(UnitsInStock) from Products data source.

The steps to achieve this scenerio are as follows:

1. Create two data sources ("Employees", "Products") in the report.
2. Define two calculated fields ("FullName", "Age") in Employees data source.
3. Define two calculated fields ("CategoryName", "Sum(UnitsInStock)") in Products data source.
4. Create two chart fields which bind to "Employees" and "Products" data sources separately.

The following code illustrates the scenerio:

Visual Basic

```
Private report As C1FlexReport
    Private Function CreateChartSampleReport() As C1FlexReport
        report = New C1FlexReport() With { _
            .ReportName = "ChartSample" _
        }
        ' add data source "Employees"
        Dim dsEmployees = New DataSource() With { _
            .Name = "Employees", _
            .ConnectionString =
"Provider=Microsoft.Jet.OLEDB.4.0;Data
Source=..\..\Reports\C1Nwind.mdb;Persist Security Info=False", _
            .DataSource = "Select * from Employees" _
        }
        report.DataSources.Add(dsEmployees)
        ' add calculated field "FullName".
        Dim calcFullName = New CalculatedField("FullName",
GetType(String), "=LastName & "" "" & FirstName")
        dsEmployees.CalculatedFields.Add(calcFullName)
        ' add calculated field "Age".
```

```
        Dim calcAge = New CalculatedField("Age", GetType(Integer),
"=Year(Now())-Year(BirthDate) + 1")
        dsEmployees.CalculatedFields.Add(calcAge)
        ' add data source "Products"
        Dim dsProducts = New DataSource() With { _
            .Name = "Products", _
            .ConnectionString =
"Provider=Microsoft.Jet.OLEDB.4.0;Data
Source=..\..\Reports\C1Nwind.mdb;Persist Security Info=False", _
            .RecordSource = "Select Products.CategoryID as
CategoryID, Categories.CategoryName as CategoryName,
Products.UnitsInStock as UnitsInStock from Products inner join
Categories on Products.CategoryID = Categories.CategoryID" _
        }
        report.DataSources.Add(dsProducts)
        report.Sections.Header.Visible = True
        ' add ChartField using Employees data source.
        Dim sectionEmployees =
report.Sections.Header.SubSections.Add()
        sectionEmployees.Name = "ChartWithEmployees"
        sectionEmployees.Height = 5200
        sectionEmployees.Visible = True
        sectionEmployees.Fields.Add(CreateChartForEmployees())
        ' add ChartField using Products data source.
        Dim sectionProducts =
report.Sections.Header.SubSections.Add()
        sectionProducts.Name = "ChartWithProducts"
        sectionProducts.Height = 5200
        sectionProducts.Visible = True
        sectionProducts.Fields.Add(CreateChartForProducts())
        Return report
    End Function
    Private Function CreateChartForEmployees() As ChartField
        Dim chart = CreateChartField("Chart1", "Employees")
        chart.Header.Text = "Employees Age"
        chart.ChartArea2D.Inverted = True
        chart.ChartArea2D.AxisX.OnTop = True
        Dim group = chart.ChartGroups2D.Group0
        group.ChartType = Chart2DType.Bar
        Dim data = group.ChartData
        data.IsForEachRecord = True        ' show value of each
record in data source
        data.CategoryGroups.AddNewGroup("=FullName")        ' group
by FullName
        Dim seriesTemplate = data.SeriesValues.AddNewSeries()
        seriesTemplate.DataValues.AddNewValue("=Age")        ' show
Age in AxisY
        Return chart
    End Function
    Private Function CreateChartForProducts() As ChartField
        Dim chart = CreateChartField("Chart2", "Products")
```

```
        chart.Header.Text = "Sum of UnitsInStock by Category"
        chart.ChartArea2D.Inverted = True
        chart.ChartArea2D.AxisX.OnTop = True
        Dim group = chart.ChartGroups2D.Group0
        group.ChartType = Chart2DType.Bar
        Dim data = group.ChartData
        Dim categoryGroup =
data.CategoryGroups.AddNewGroup("=CategoryID")
        ' group by each CategoryID
        categoryGroup.LabelExpression = "=CategoryName"  ' show the
CategoryName in AxisX
        Dim seriesTemplate = data.SeriesValues.AddNewSeries()
        seriesTemplate.DataValues.AddNewValue("=Sum(UnitsInStock)")
        ' show sum of UnitsInStock in AxisY
        Return chart
    End Function
    Private Function CreateChartField(name As String, datasource As
String) As ChartField
        Dim chart = New ChartField() With { _
            .Name = name, _
             .Width = 7500, _
            .Height = 5000, _
             .Top = 100, _
             .Left = 100, _
             .DataSource = datasource _
        }
        chart.Border.Color = Color.Black
        chart.Border.Width = 15
        chart.Border.Style = DashStyle.Solid
        chart.Border.CornerRadius = New CornerRadius(200.0)
        chart.ChartArea2D.AxisY.AutoMin = False
        Return chart
    End Function
    Private Sub Button1_Click(sender As Object, e As EventArgs)
Handles Button1.Click
        CreateChartSampleReport()
        C1FlexViewer1.DocumentSource = report
    End Sub
```

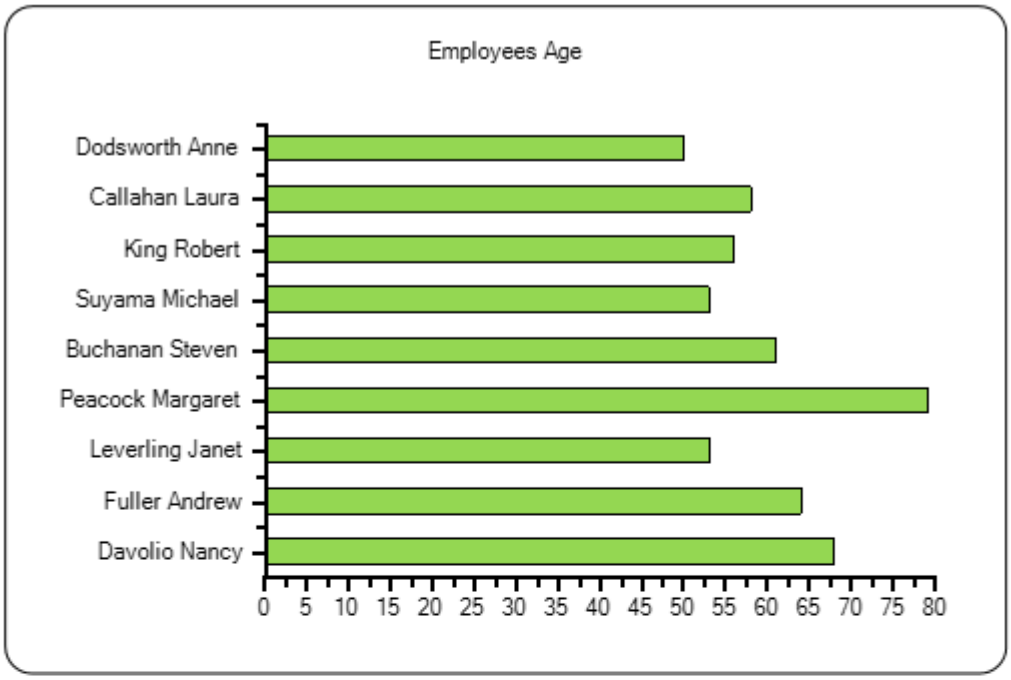
C#

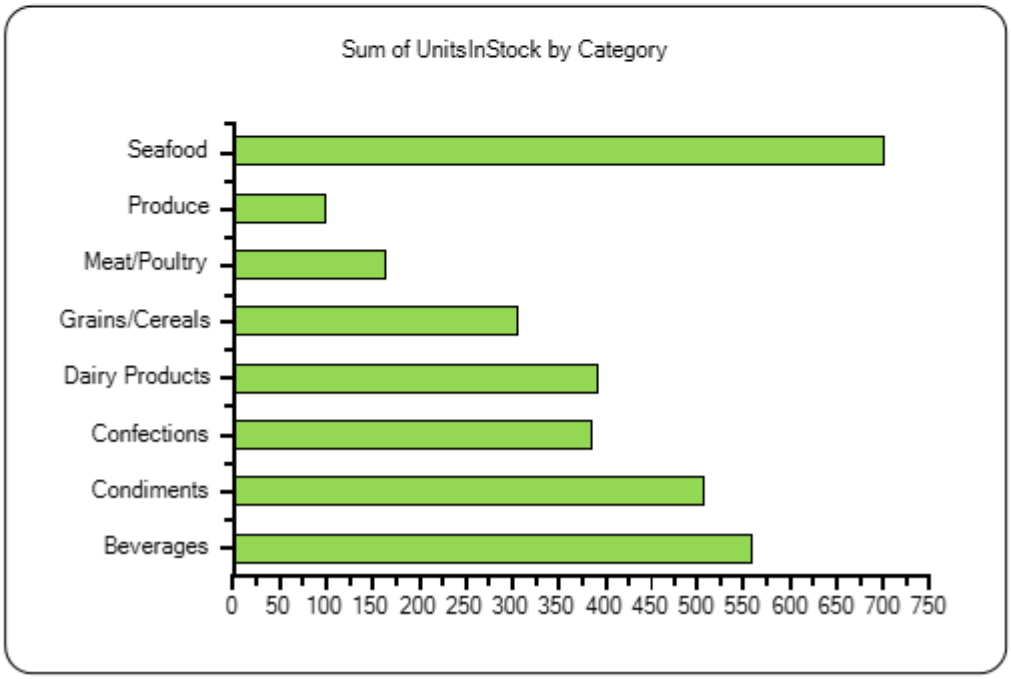
```
 private C1FlexReport CreateChartSampleReport()
        {

           var report = new C1FlexReport { ReportName =
"ChartSample" };
            // add data source "Employees"
            var dsEmployees = new DataSource
            {
                Name = "Employees",
                ConnectionString =
                    @"Provider=Microsoft.Jet.OLEDB.4.0;Data
```

```
Source=..\..\Reports\C1Nwind.mdb;Persist Security Info=False",
                RecordSource = "Select * from Employees",
            };
            report.DataSources.Add(dsEmployees);
            // add calculated field "FullName".
            var calcFullName = new CalculatedField("FullName",
typeof(string), "=LastName & \" \" & FirstName");
            dsEmployees.CalculatedFields.Add(calcFullName);

            // add calculated field "Age".
            var calcAge = new CalculatedField("Age", typeof(int),
"=Year(Now())-Year(BirthDate) + 1");
            dsEmployees.CalculatedFields.Add(calcAge);
            // add data source "Products"
            var dsProducts = new DataSource
            {
                Name = "Products",
                ConnectionString =
                    @"Provider=Microsoft.Jet.OLEDB.4.0;Data
Source=..\..\Reports\C1Nwind.mdb;Persist Security Info=False",
                RecordSource =
                    "Select Products.CategoryID as CategoryID,
Categories.CategoryName as CategoryName, Products.UnitsInStock as
UnitsInStock from Products inner join Categories on
Products.CategoryID = Categories.CategoryID"
            };
            report.DataSources.Add(dsProducts);
            report.Sections.Header.Visible = true;
            // add ChartField using Employees data source.
            var sectionEmployees =
report.Sections.Header.SubSections.Add();
            sectionEmployees.Name = "ChartWithEmployees";
            sectionEmployees.Height = 5200;
            sectionEmployees.Visible = true;
            sectionEmployees.Fields.Add(CreateChartForEmployees());
            // add ChartField using Products data source.
            var sectionProducts =
report.Sections.Header.SubSections.Add();
            sectionProducts.Name = "ChartWithProducts";
            sectionProducts.Height = 5200;
            sectionProducts.Visible = true;
            sectionProducts.Fields.Add(CreateChartForProducts());

            return report;
        }

        private ChartField CreateChartForEmployees()
        {
            var chart = CreateChartField("Chart1", "Employees");
            chart.Header.Text = "Employees Age";
            chart.ChartArea2D.Inverted = true;
```

```
            chart.ChartArea2D.AxisX.OnTop = true;
            var group = chart.ChartGroups2D.Group0;
            group.ChartType = Chart2DType.Bar;
            var data = group.ChartData;
            data.IsForEachRecord = true; // show value of each
record in data source
            data.CategoryGroups.AddNewGroup("=FullName"); // group
by FullName
            var seriesTemplate = data.SeriesValues.AddNewSeries();
            seriesTemplate.DataValues.AddNewValue("=Age"); // show
Age in AxisY
            return chart;
        }
        private ChartField CreateChartForProducts()
        {
            var chart = CreateChartField("Chart2", "Products");
            chart.Header.Text = "Sum of UnitsInStock by Category";
            chart.ChartArea2D.Inverted = true;
            chart.ChartArea2D.AxisX.OnTop = true;
            var group = chart.ChartGroups2D.Group0;
            group.ChartType = Chart2DType.Bar;
            var data = group.ChartData;
            var categoryGroup =
data.CategoryGroups.AddNewGroup("=CategoryID"); // group by each
CategoryID
            categoryGroup.LabelExpression = "=CategoryName"; // show
the CategoryName in AxisX.
            var seriesTemplate = data.SeriesValues.AddNewSeries();

seriesTemplate.DataValues.AddNewValue("=Sum(UnitsInStock)"); // show
sum of UnitsInStock in AxisY.
            return chart;
        }
        private ChartField CreateChartField(string name, string
datasource)
        {
            var chart = new ChartField
            {
                Name = name,
                Width = 7500,
                Height = 5000,
                Top = 100,
                Left = 100,
                DataSource = datasource,
            };
            chart.Border.Color = Color.Black;
            chart.Border.Width = 15;
            chart.Border.Style = DashStyle.Solid;
            chart.Border.CornerRadius = new CornerRadius(200d);
            chart.ChartArea2D.AxisY.AutoMin = false;
            return chart;
```

```
        }
private void button1_Click(object sender, EventArgs e)
        {
            CreateChartSampleReport();
            c1FlexViewer1.DocumentSource = report;
        }
```





Binding Data to Parameters in Multiple Data Source Report

Binding data to parameters defines the valid values for the report parameters (elements in the C1FlexReport.Parameters collection). The **ReportParameter.AllowedValuesDefinition.Binding.DataSourceName**

property indicates the data source which is used to build the list of possible values in the parameters. The following code illustrates how to bind data to the parameters in a report with multiple data sources.

Visual Basic

```
' add datasource and parameter using this datasource
Dim mds As DataSource = C1FlexReport.DataSource
Dim ds As New DataSource()
ds.Name = "CategoriesDS"
ds.ConnectionString = mds.ConnectionString
ds.RecordSource = "select * from categories"
ds.DataProvider = DataProvider.OLEDB
C1FlexReport.DataSources.Add(ds)
mds.RecordSource = "select * from products where categoryid =
[CategoryParam]"
Dim rp As New ReportParameter()
rp.DataType = Doc.ParameterType.[Integer]
rp.Prompt = "Category"
rp.Name = "CategoryParam"
rp.AllowedValuesDefinition.Binding.DataSourceName = "CategoriesDS"
rp.AllowedValuesDefinition.Binding.ValueExpression = "CategoryID"
rp.AllowedValuesDefinition.Binding.LabelExpression = "CategoryName"
C1FlexReport.Parameters.Add(rp)
```

C#

```
// add datasource and parameter using this datasource
DataSource mds = c1FlexReport.DataSource;
DataSource ds = new DataSource();
ds.Name = "CategoriesDS";
ds.ConnectionString = mds.ConnectionString;
ds.RecordSource = "select * from categories";
ds.DataProvider = DataProvider.OLEDB;
c1FlexReport.DataSources.Add(ds);
mds.RecordSource = "select * from products where categoryid =
[CategoryParam]";
ReportParameter rp = new ReportParameter();
rp.DataType = Doc.ParameterType.Integer;
rp.Prompt = "Category";
rp.Name = "CategoryParam";
rp.AllowedValuesDefinition.Binding.DataSourceName = "CategoriesDS";
rp.AllowedValuesDefinition.Binding.ValueExpression = "CategoryID";
rp.AllowedValuesDefinition.Binding.LabelExpression = "CategoryName";
c1FlexReport.Parameters.Add(rp);
```

Defining Calculated Fields

Calculated fields contain expressions that are evaluated at run-time. These can be added to a data source using **DataSource.CalculatedFields.Add** method.

The code to add a calculated field, say 'Calc1' that calculates 'CategoryID * 2' , of the Integer type, is as follows:

Visual Basic

```
Dim ds As DataSource = C1FlexReport1.DataSources(0)
ds.CalculatedFields.Add(New CalculatedField("Calc1", GetType(Integer),
"CategoryID * 2"))
```

C#

```
DataSource ds = c1FlexReport1.DataSources[0];
ds.CalculatedFields.Add(new CalculatedField("Calc1", typeof(int),
"CategoryID * 2"));
```

Note that if there are more than one Calculated field, they must have unique names.

Developing FlexReport for Desktop

In typical desktop scenarios, C1FlexReport runs on the same computer where the reports are generated and viewed (the report data itself may still come from a remote server). The following sections assume that FlexReport is hosted in a Visual Studio environment.

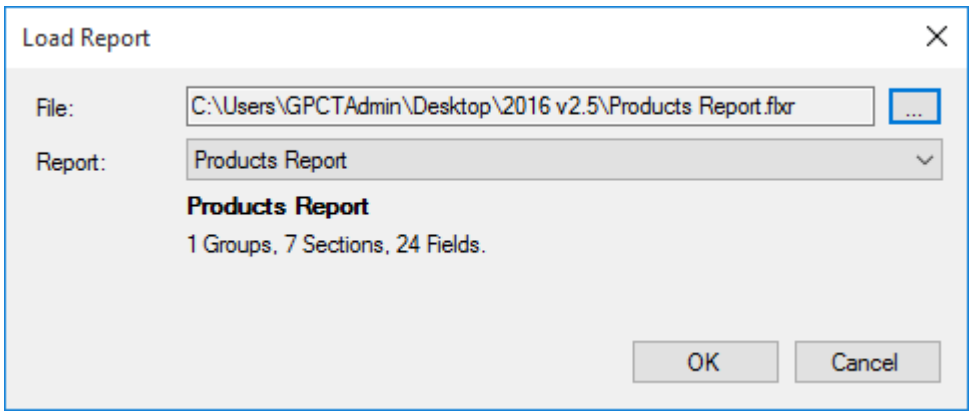
Load FlexReport at Design Time

In this scenario, an application generates reports using a fixed set of report definitions that are built into the application. This type of application does not rely on any external report definition files, and end-users can not modify these reports.

The main advantage of this type of application is that you don't need to distribute the report definition file, and you can be sure that no one can modify the report format. The disadvantage is that to make any modifications to the report, you must recompile the application.

If you want to use a report definition that you already have, without any modifications, follow these steps:

1. Add one **C1FlexReport** component for each report definition you want to distribute. You may want to name each control after the report it will render (this will make your code easier to maintain).
2. Right-click each C1FlexReport component and select the **Load Report** menu option to load report definitions into each control. (You can also click the smart tag ( ) above the component to open the **C1FlexReport Tasks** menu and choose the **Load Report** option.)
   The **Load Report** dialog box appears, which allows you to select a report definition file and then a report within that file.
   To load a report, click the **ellipses** button to select the report definition file you created in Step 1, then select the report from the drop-down list and click **OK**. The **Load Report** dialog box shows the name of the report you selected and a count of groups, sections, and fields. This is what the dialog box looks like:



3. Add **FlexViewer** control to the form. Also, add a control that will allow the user to pick a report (this could be a menu, a list box, or a group of buttons).
4. Add code to render the report selected by the user. For example, if you added a button in the previous step with the name **btnProductsReport**, the code would look like this:

Visual Basic

```
Private Sub btnProductsReport_Click(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles btnProductsReport.Click
  c1FlexViewer1.DocumentSource =rptProducts
End Sub
```

C#

```
private void btnProductsReport_Click(object sender, System.EventArgs
e)
{
c1FlexViewer1.DocumentSource =rptProducts;
}
```

Note that **rptProducts** is the name of the C1FlexReport component that contains the report selected by the user and c1FlexViewer1 is the name of the **FlexViewer** control.

Create FlexReport at Design Time

The **Load Report** option, described in Load FlexReport at Design Time, makes it easy to embed reports you already have in your application. In some cases, however, you may want to customize the report, or use data source objects that are defined in your Visual Studio application rather than using connection strings and record sources. In these situations, use the **Edit Report** command instead.

To create or edit reports at design time, right-click the C1FlexReport component and select the **Edit Report** menu option to invoke the **C1FlexReportDesigner** application (you can also click the smart tag (▶) above the component to open the **C1FlexReport Tasks** menu and select the **Edit Report** option).

Note: If the **Edit Report** command doesn't appear on the context menu and Properties window, it is probably because the control could not find the **C1FlexReportDesigner** application. To fix this, simply run the **C1FlexReportDesigner** application once in stand-alone mode. The designer will save its location to the registry, and the C1FlexReport component should be able to find it afterwards.

The **C1FlexReportDesigner** application shows the report that is currently loaded in the C1FlexReport component. If the C1FlexReport component is empty, the Designer shows the **C1FlexReport Wizard** so you can create a new report.

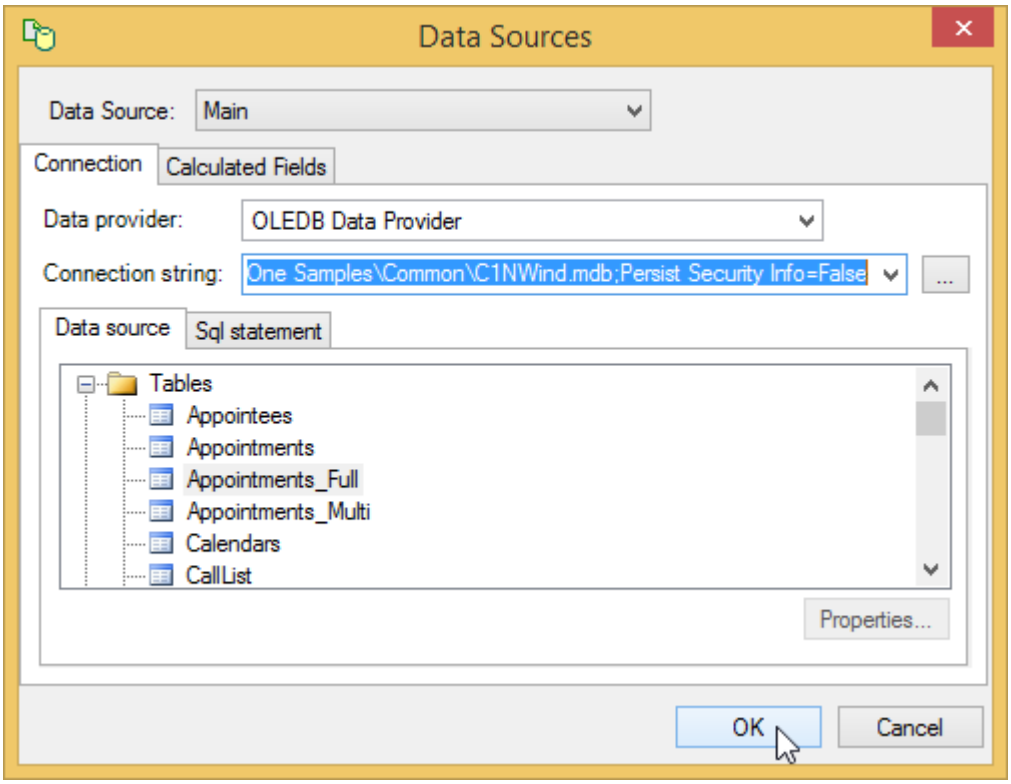
This is the same **C1FlexReportDesigner** application that is used in stand-alone mode. The only differences you will notice when you start the **C1FlexReportDesigner** application in this mode are:

- You can use the data source objects defined in your application as data sources for your new reports.
- When you close the Designer, any changes you made will be saved back into the C1FlexReport component on the form (unless you choose to discard the changes by selecting **File | Exit** from the Designer's menu, and select **No** to saving the changes).

To use data source objects defined in your application, click the **Data Source** button in the Designer, then select the **Tables** option in the **Select a Data Source** dialog box.

The **Tables** page shows a list of data objects currently defined on the form (the page will not be visible if there aren't any valid data sources on the form). Alternatively, you can use the **Connection string** to build and select a connection string and record source as usual.

For example, if the main form contains a DataSet object with several DataTables attached to it, the data source picker dialog box might look like this:



Once you are done creating or editing the report, you can close the Designer by selecting **File | Save** and **File | Exit** from the menu. This will save the report definition directly into the component (as if you had loaded it from a file using the **Load Report** command).

If you change your mind and decide to cancel the changes, quit the Designer selecting **File | Exit** from the menu and choose **No** to saving the changes.

Load FlexReport at Run Time

Loading reports at run time requires a report definition file and a viewer. The main advantage of this type of application is that if you modify the report format, there's no need to update the application. Simply send the new report definition file to the users and you are done.

To create an application with reports loaded at run time, follow these steps:

1. Create all the required reports in the **C1FlexReportDesigner** application. For more information, see Working

with C1FlexReportDesigner.

2. Add the following controls to the application:
   - **C1FlexReport** component named c1FlexReport1
   - **C1FlexViewer** control named **fv**
   - **ComboBox** control named **cmbReport**
   - **Button** control named **button1**
3. Add the following Import statements to the top of the file:

Visual Basic

```
Imports C1.Win.FlexReport
Imports System.IO
```

C#

```
using C1.Win.FlexReport;
using System.IO;
```

This allows you to reference the **C1FlexReport** and **System.IO** classes and objects without having to specify the full namespaces.

4. Add the following code in the button click event to read the report definition file and build a list of all reports:

   - **Visual Basic**

```
' get application path
Dim appPath As String
appPath = Path.GetDirectoryName(Application.ExecutablePath).ToLower()
Dim i As Integer = appPath.IndexOf(vbBack & "in")
If (i < 0) Then
    i = appPath.IndexOf(vbBack & "in")
End If
If (i > 0) Then
    appPath = appPath.Remove(i, appPath.Length - i)
End If
' get names of reports in the report definition file
m_ReportDefinitionFile = appPath & Convert.ToString("\Data\Products Report.flxr")
Dim reports As String() = C1FlexReport.GetReportList(m_ReportDefinitionFile)
' populate combo box
cmbReport.Items.Clear()

For Each report As String In reports
    cmbReport.Items.Add(report)
Next
```

   - **C#**

```
// get application path
string appPath;
appPath = Path.GetDirectoryName(Application.ExecutablePath).ToLower();
int i = appPath.IndexOf("\bin");
if ((i < 0)) { i = appPath.IndexOf("\bin"); }
if ((i > 0)) { appPath = appPath.Remove(i, appPath.Length - i); }
// get names of reports in the report definition file
m_ReportDefinitionFile = appPath + @"\Data\Products Report.flxr";
string[] reports = C1FlexReport.GetReportList(m_ReportDefinitionFile);
// populate combo box
cmbReport.Items.Clear();
```

```
foreach (string report in reports)
{
    cmbReport.Items.Add(report);
}
```

The code starts by getting the location of the file that contains the report definitions. This is done using static methods in the system-defined **Path** and **Application** classes. You may have to adjust the code to reflect the location and name of your report definition file.

Then it uses the GetReportList method to retrieve an array containing the names of all reports in the report definition file (created in step 1), and populates the combo box allowing users to select the report.

5. Add code to render the report selected by the user. For example:
   - **Visual Basic**

```
Try
    Cursor = Cursors.WaitCursor

    ' load report
    fv.StatusText = "Loading" + cmbReport.Text
    C1FlexReport1.Load(m_ReportDefinitionFile, cmbReport.Text)


    ' render into print preview control
    fv.StatusText = "Rendering" + cmbReport.Text
    fv.DocumentSource = C1FlexReport1

    ' give focus to print preview control
    fv.Focus()
Finally
    Cursor = Cursors.[Default]
End Try
```

   - **C#**

```
try
{
    Cursor = Cursors.WaitCursor;

    // load report
    fv.StatusText = "Loading" + cmbReport.Text;
    c1FlexReport1.Load(m_ReportDefinitionFile, cmbReport.Text);


    // render into print preview control
    fv.StatusText = "Rendering" + cmbReport.Text;
    fv.DocumentSource = c1FlexReport1;

    // give focus to print preview control
    fv.Focus();
}
finally
{
    Cursor = Cursors.Default;
}
```

6. Run the project.

Adding Parameters

Parameters are an important part of any report. They influence the data populated by manipulating the data passed in the report. Parameters can be used for modifying the default values of data and applying filtering to the data. You can also select more than one value using multi-value parameters.

FlexReport has parameters collection, C1FlexReport.Parameters, where parameters can be defined to specify types, captions, default value, possible values, and so on.

Each element that is defined as parameters in the **C1FlexReport.Parameters** collection is an instance of the ReportParameter class, with the following properties:

Nullable	Gets or sets a value indicating whether the value of this parameter can be Null. Cannot be true if this is a multi-value parameter.
AllowBlank	Gets or sets a value indicating whether the value of this parameter can be an empty string. Ignored unless DataType is String.
MultiValue	Gets or sets a value indicating whether this is a multivalue parameter (a parameter that can take a set of values).
Hidden	Gets or sets a value indicating whether the parameter should be hidden from the end user (however, it will still be available for programmatic use with subreports, drill-through reports etc.)
Prompt	Gets or sets the prompt shown to the end user when prompting for parameter values.
Value	Gets or sets the parameter value. Value can be specified as an array if MultiValue is true (in this case all items should have the same item type).
DataType	Gets or sets the data type of the Parameter.
AllowedValuesDefinition	Gets AllowedValuesDefinition defining the list of allowed values for this parameter. Allowed values can be specified as a static list using AllowedValuesDefinition.Values property, or as a dynamic list bound to one of report's data sources using AllowedValuesDefinition.Binding property.

Report parameters can be easily added through the FlexReportDesigner application. For more information, see Working with Parameters and Binding Data to Parameters in Multiple Data Source Report .

Grouping Data

Grouping is the most commonly used method to represent data in an organized manner. After designing the basic layout, you may decide to segregate the records by certain fields, or other criteria that would make the report easier to read. By grouping data, you can separate groups of records and display introductory and summary data for each group. The group break is based on a grouping expression. This expression is usually based on one or more recordset fields but it can be as complex as you want.

In FlexReport, grouping is achieved by using C1FlexReport.Groups.

Lets say you want to view a list of employees falling under a designation or title. In this case, the list should be grouped by Title. The following steps illustrate how to Group the list of employees by the Title. This example uses sample created in FlexReport Quick Start.

1. Add a **C1CheckBox** to the Form in the FlexReport Quick Start project.
2. Set the C1CheckBox **Name** to 'groupC1CheckBox' and **Text** to 'Group Report by Title'.
3. Create **CheckedChanged** event as c1CheckBox1_CheckedChanged.
4. Add the following code.

Visual Basic

```
Private grp As Group
Private s As Section
Private Sub c1CheckBox1_CheckedChanged(sender As Object, e As EventArgs)
```

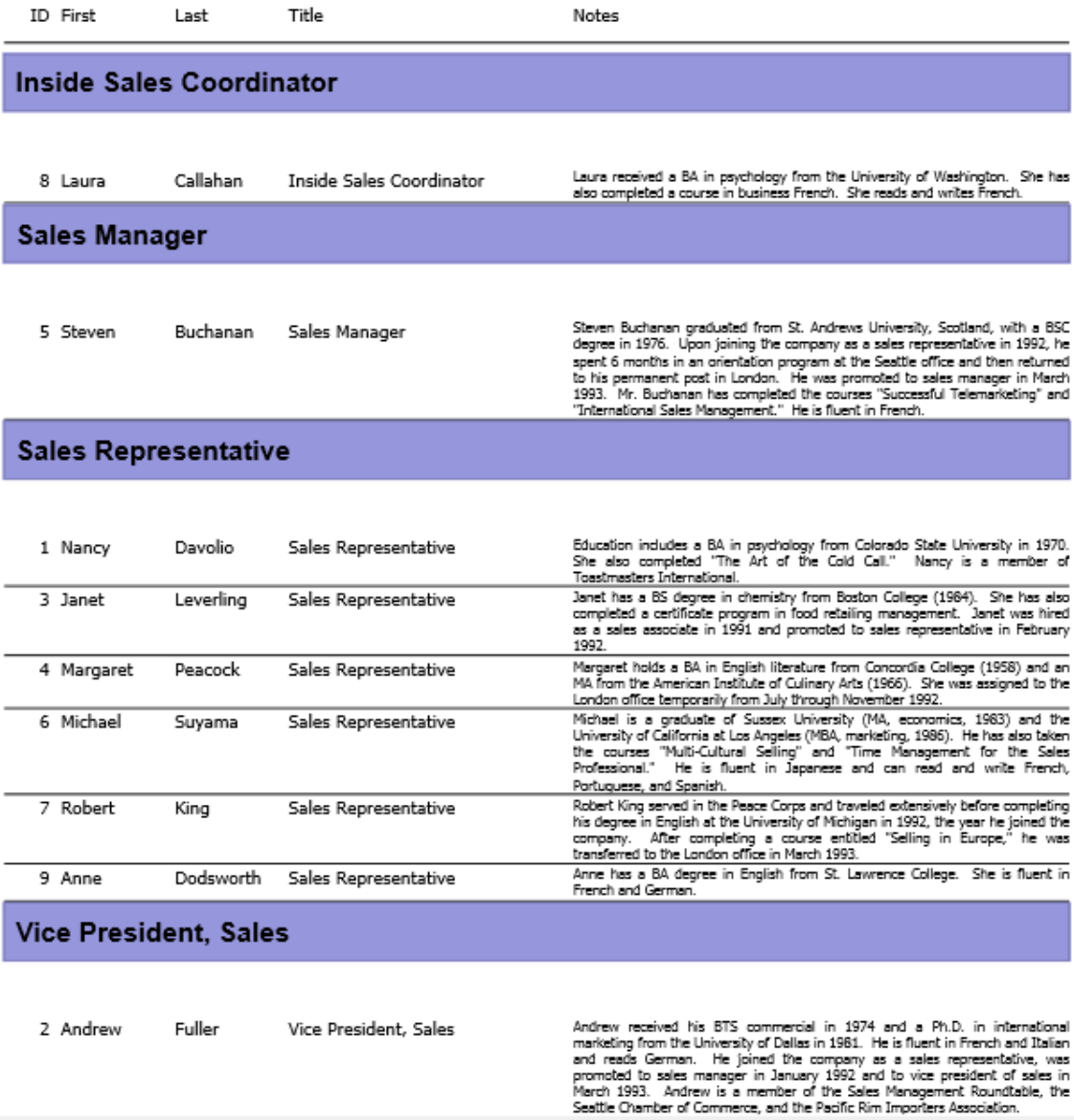
```
    If groupC1CheckBox.Checked Then
        ' group employees by title and sort titles in ascending order
        grp = C1FlexReport1.Groups.Add("GrpTitle", "Title",
SortEnum.Ascending)
        ' format the Header section for the new group
        s = grp.SectionHeader
        s.Height = 1000
        s.Visible = True
        Dim f As New TextField()
        f.Name = "Title"
        f.Text.Expression = "Title"
        f.Left = 0
        f.Top = 0
        f.Width = C1FlexReport1.Layout.Width
        f.Height = 500
        f.Align = FieldAlignEnum.LeftMiddle
        f.Font.Bold = True
        f.Font.Size = 12
        f.Border = New Border(2, Color.Black, DashStyle.Solid)
        f.BackColor = Color.FromArgb(150, 150, 220)
        f.MarginLeft = 100
        s.Fields.Add(f)
        C1FlexReport1.Render()
    Else
          btnEmployees.PerformClick()
    End If
End Sub
```

C#

```
Group grp;
Section s;
private void c1CheckBox1_CheckedChanged(object sender, EventArgs e)
        {
            if (groupC1CheckBox.Checked)
            {
             // group employees by title and sort titles in ascending
order
                grp = c1FlexReport1.Groups.Add("GrpTitle", "Title",
SortEnum.Ascending);
             // format the Header section for the new group
                s = grp.SectionHeader;
                s.Height = 1000;
                s.Visible = true;

                TextField f = new TextField();
                f.Name = "Title";
                f.Text.Expression = "Title";
                f.Left = 0;
                f.Top = 0;
                f.Width = c1FlexReport1.Layout.Width;
                f.Height = 500;
                f.Align = FieldAlignEnum.LeftMiddle;
                f.Font.Bold = true;
                f.Font.Size = 12;
```

```
                f.Border = new Border(2, Color.Black, DashStyle.Solid);
                f.BackColor = Color.FromArgb(150, 150, 220);
                f.MarginLeft = 100;
                s.Fields.Add(f);
                c1FlexReport1.Render();
            }
            else
            {
                btnEmployees.PerformClick();
            }
        }
```

5. Run the project. Click Employees button to render the report.
6. Click 'Group Report by Title' checkbox to view grouping the report. Observe that the titles are sorted in the ascending order.



ID	First	Last	Title	Notes
Inside Sales Coordinator				
8	Laura	Callahan	Inside Sales Coordinator	Laura received a BA in psychology from the University of Washington. She has also completed a course in business French. She reads and writes French.
Sales Manager				
5	Steven	Buchanan	Sales Manager	Steven Buchanan graduated from St. Andrews University, Scotland, with a BSC degree in 1976. Upon joining the company as a sales representative in 1992, he spent 6 months in an orientation program at the Seattle office and then returned to his permanent post in London. He was promoted to sales manager in March 1993. Mr. Buchanan has completed the courses "Successful Telemarketing" and "International Sales Management." He is fluent in French.
Sales Representative				
1	Nancy	Davolio	Sales Representative	Education includes a BA in psychology from Colorado State University in 1970. She also completed "The Art of the Cold Call." Nancy is a member of Toastmasters International.
3	Janet	Leverling	Sales Representative	Janet has a BS degree in chemistry from Boston College (1984). She has also completed a certificate program in food retailing management. Janet was hired as a sales associate in 1991 and promoted to sales representative in February 1992.
4	Margaret	Peacock	Sales Representative	Margaret holds a BA in English literature from Concordia College (1958) and an MA from the American Institute of Culinary Arts (1966). She was assigned to the London office temporarily from July through November 1992.
6	Michael	Suyama	Sales Representative	Michael is a graduate of Sussex University (MA, economics, 1983) and the University of California at Los Angeles (MBA, marketing, 1986). He has also taken the courses "Multi-Cultural Selling" and "Time Management for the Sales Professional." He is fluent in Japanese and can read and write French, Portuguese, and Spanish.
7	Robert	King	Sales Representative	Robert King served in the Peace Corps and traveled extensively before completing his degree in English at the University of Michigan in 1992, the year he joined the company. After completing a course entitled "Selling in Europe," he was transferred to the London office in March 1993.
9	Anne	Dodsworth	Sales Representative	Anne has a BA degree in English from St. Lawrence College. She is fluent in French and German.
Vice President, Sales				
2	Andrew	Fuller	Vice President, Sales	Andrew received his BTS commercial in 1974 and a Ph.D. in international marketing from the University of Dallas in 1981. He is fluent in French and Italian and reads German. He joined the company as a sales representative, was promoted to sales manager in January 1992 and to vice president of sales in March 1993. Andrew is a member of the Sales Management Roundtable, the Seattle Chamber of Commerce, and the Pacific Rim Importers Association.

Adding Subtotals and Other Aggregates

C1FlexReport supports aggregate expressions in all its calculated fields. The aggregate expressions includes

aggregates - Sum, Min, Max, Avg, Count, Range, Var, and so on.

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

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

The following aggregate expression would return the total amount of sales taxes paid for all values in the report (assuming an 8.5% sales tax):`Sum(Sales * 0.085)`

The following example uses **Count** aggregate to calculate number of records for employees falling under a designation.

1. Add the following code in the check box event (c1CheckBox1_CheckedChanged) created in the sample Grouping Data.

Visual Basic

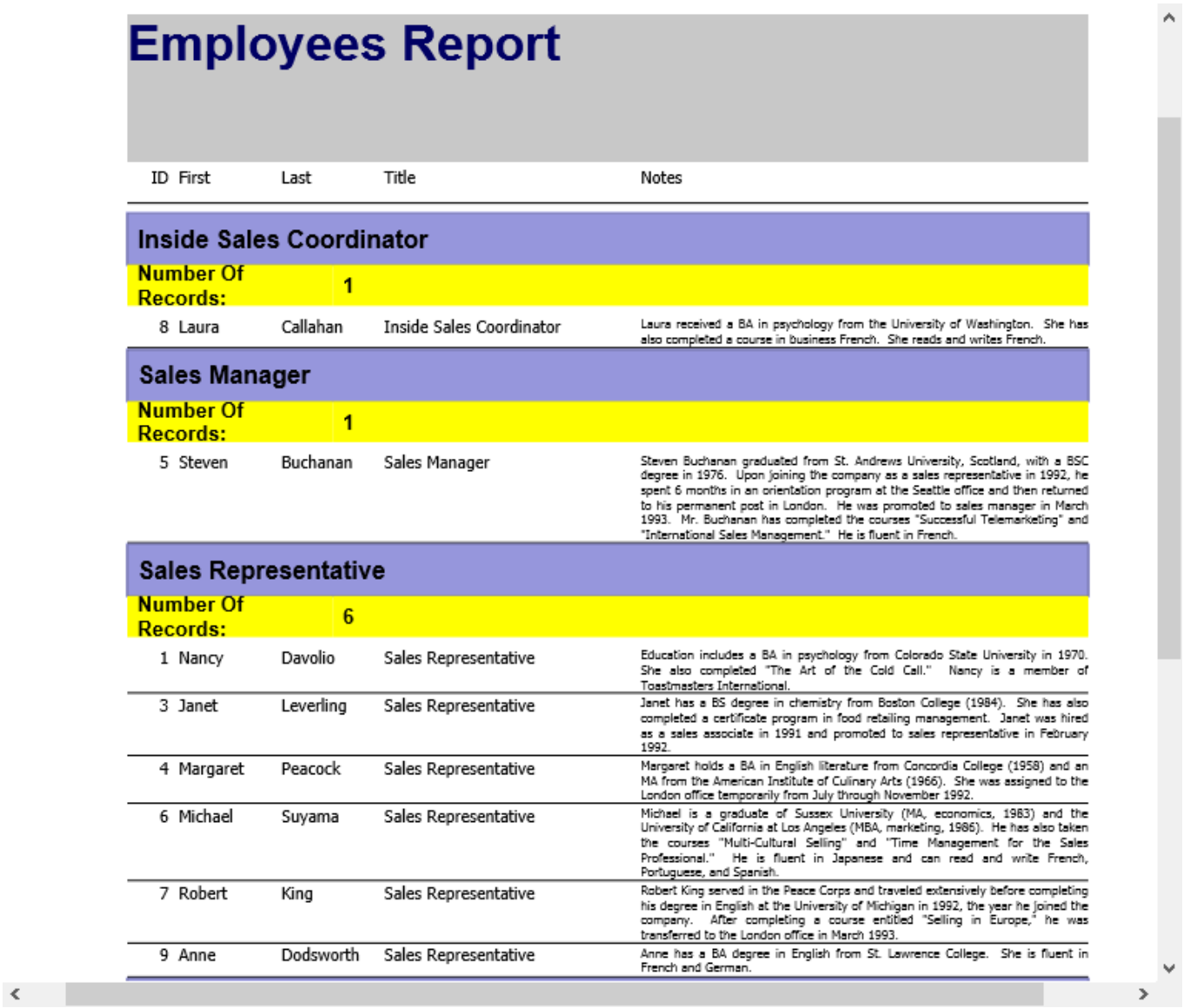
```
Dim f1 As New Field()
f1.Name = "CountRecords"
f1.Text = "Count(GrpTitle)"
f1.Left = 2000
f1.Top = 500
f1.Width = C1FlexReport1.Layout.Width - 2000
f1.Height = 400
f1.Align = FieldAlignEnum.LeftMiddle
f1.MarginLeft = 100
f1.Calculated = True
f1.Visible = True
f1.BackColor = Color.Yellow
f1.Font.Bold = True
f1.Font.Size = 10
s.Fields.Add(f1)
Dim tf As New TextField()
tf.Name = "Text"
tf.Text = "Number Of Records: "
tf.Left = 0
tf.Top = 500
tf.Width = C1FlexReport1.Layout.Width - f1.Width
tf.Height = 400
tf.Align = FieldAlignEnum.LeftMiddle
tf.Font.Bold = True
tf.Font.Size = 10
tf.BackColor = Color.Transparent
tf.BackColor = Color.Yellow
tf.MarginLeft = 100
tf.Visible = True
s.Fields.Add(tf)
```

C#

```
Field f1 = new Field();
f1.Name = "CountRecords";
f1.Text = "Count(GrpTitle)";
```

```
f1.Left = 2000;
f1.Top = 500;
f1.Width = c1FlexReport1.Layout.Width - 2000;
f1.Height = 400;
f1.Align = FieldAlignEnum.LeftMiddle;
f1.MarginLeft = 100;
f1.Calculated = true;
f1.Visible = true;
f1.BackColor = Color.Yellow;
f1.Font.Bold = true;
f1.Font.Size = 10;
s.Fields.Add(f1);
TextField tf = new TextField();
tf.Name = "Text";
tf.Text = "Number Of Records: ";
tf.Left = 0;
tf.Top = 500;
tf.Width = c1FlexReport1.Layout.Width - f1.Width;
tf.Height = 400;
tf.Align = FieldAlignEnum.LeftMiddle;
tf.Font.Bold = true;
tf.Font.Size = 10;
tf.BackColor = Color.Transparent;
tf.BackColor = Color.Yellow;
tf.MarginLeft = 100;
tf.Visible = true;
s.Fields.Add(tf);
```

2. Run the project. Click 'Employees' button to render the report.
3. Click 'Group Report by Title' checkbox to view grouping in the report. Observe that Number of Records are calculated for each group.



Employees Report

ID	First	Last	Title	Notes
Inside Sales Coordinator				
Number Of Records:	**1**			
8	Laura	Callahan	Inside Sales Coordinator	Laura received a BA in psychology from the University of Washington. She has also completed a course in business French. She reads and writes French.
Sales Manager				
Number Of Records:	**1**			
5	Steven	Buchanan	Sales Manager	Steven Buchanan graduated from St. Andrews University, Scotland, with a BSC degree in 1976. Upon joining the company as a sales representative in 1992, he spent 6 months in an orientation program at the Seattle office and then returned to his permanent post in London. He was promoted to sales manager in March 1993. Mr. Buchanan has completed the courses "Successful Telemarketing" and "International Sales Management." He is fluent in French.
Sales Representative				
Number Of Records:	**6**			
1	Nancy	Davolio	Sales Representative	Education includes a BA in psychology from Colorado State University in 1970. She also completed "The Art of the Cold Call." Nancy is a member of Toastmasters International.
3	Janet	Leverling	Sales Representative	Janet has a BS degree in chemistry from Boston College (1984). She has also completed a certificate program in food retailing management. Janet was hired as a sales associate in 1991 and promoted to sales representative in February 1992.
4	Margaret	Peacock	Sales Representative	Margaret holds a BA in English literature from Concordia College (1958) and an MA from the American Institute of Culinary Arts (1966). She was assigned to the London office temporarily from July through November 1992.
6	Michael	Suyama	Sales Representative	Michael is a graduate of Sussex University (MA, economics, 1983) and the University of California at Los Angeles (MBA, marketing, 1986). He has also taken the courses "Multi-Cultural Selling" and "Time Management for the Sales Professional." He is fluent in Japanese and can read and write French, Portuguese, and Spanish.
7	Robert	King	Sales Representative	Robert King served in the Peace Corps and traveled extensively before completing his degree in English at the University of Michigan in 1992, the year he joined the company. After completing a course entitled "Selling in Europe," he was transferred to the London office in March 1993.
9	Anne	Dodsworth	Sales Representative	Anne has a BA degree in English from St. Lawrence College. She is fluent in French and German.

Creating Cross-Tab Reports

Cross-tab reports group data in two dimensions (down and across). They are useful for summarizing large amounts of data in a format that cross-references information.

The following steps create a cross-tab report in the report created in FlexReport Quick Start.

1. Add a **Button** to the Form in the FlexReport Quick Start project.
2. Set the Button **Name** to 'crossC1Button' and **Text** to 'View Cross Tab-Report'.
3. Create **Click** event as **crossC1Button_Click**.
4. Write the following code.

Visual Basic

```
Private grp2 As Group
Private Sub crossC1Button_Click(sender As Object, e As EventArgs)
    btnEmployees.PerformClick()
    c1FlexReport1.Sections.Detail.Visible = False
    grp2 = c1FlexReport1.Groups.Add("GrpCountry", "Country",
SortEnum.Ascending)
    ' format the Header section for the new group
```

```
c1FlexReport1.Sections.PageHeader.Height = 600
shpfld2.Top = 600
s = grp2.SectionHeader
s.Height = 400
s.AutoHeight = AutoSizeBehavior.GrowAndShrink
s.Visible = True
textFld4.Text = "Country"
textFld4.Width = 1000
textFld4.Align = FieldAlignEnum.CenterMiddle
textFld4.Height = 400
textFld4.Font.Bold = True
textFld4.Font.Size = 10
textFld5.Text = "Total"
textFld5.Width = 1000
textFld5.Left = 1000
textFld5.Align = FieldAlignEnum.CenterMiddle
textFld5.Height = 400
textFld5.Font.Bold = True
textFld5.Font.Size = 10
textFld6.Text = "Sales Representative"
textFld6.Width = 2000
textFld6.Left = 2000
textFld6.Align = FieldAlignEnum.CenterMiddle
textFld6.Height = 500
textFld6.Font.Bold = True
textFld6.Font.Size = 10
textFld7.Text = "Vice President"
textFld7.Width = 1500
textFld7.Left = 4000
textFld7.Align = FieldAlignEnum.CenterMiddle
textFld7.Height = 400
textFld7.Font.Bold = True
textFld7.Font.Size = 10
textFld8.Text = "Sales Manager"
textFld8.Width = 1500
textFld8.Left = 5800
textFld8.Align = FieldAlignEnum.CenterMiddle
textFld8.Height = 400
textFld8.Font.Bold = True
textFld8.Font.Size = 10
Dim isc As New TextField()
isc.Text = "Inside Sales Coordinator"
isc.Width = 1500
isc.Left = 7500
isc.Align = FieldAlignEnum.CenterMiddle
isc.Height = 400
isc.Font.Bold = True
isc.Font.Size = 10
isc.Visible = True
c1FlexReport1.Sections.PageHeader.Fields.Add(isc)
Dim f1 As New Field()
f1.Name = "Country"
f1.Text = "Country"
f1.Left = 80
```

```
f1.Top = 0
f1.Width = 1000
f1.Height = 400
f1.Align = FieldAlignEnum.CenterMiddle
f1.MarginLeft = 100
f1.Calculated = True
f1.Visible = True
f1.BackColor = Color.Transparent
f1.Font.Bold = True
f1.Font.Size = 10
s.Fields.Add(f1)
Dim f7 As New TextField()
f7.Name = "RunCount"
f7.Text = "=Count(Title)"
f7.Left = 1080
f7.Top = 0
f7.Width = 1000
f7.Height = 400
f7.Align = FieldAlignEnum.CenterMiddle
f7.MarginLeft = 100
f7.Visible = True
f7.Font.Bold = True
f7.Font.Size = 10
s.Fields.Add(f7)
Dim f8 As New TextField()
f8.Name = "SRCount"
f8.Text = "=Count(Title, Title = ""Sales Representative"")"
f8.Left = 2080
f8.Top = 0
f8.Width = 1000
f8.Height = 400
f8.Align = FieldAlignEnum.CenterMiddle
f8.MarginLeft = 100
f8.Visible = True
f8.Font.Bold = True
f8.Font.Size = 10
s.Fields.Add(f8)
Dim f9 As New TextField()
f9.Name = "VPCount"
f9.Text = "=Count(Title, Title = ""Vice President, Sales"")"
f9.Left = 4000
f9.Top = 0
f9.Width = 1000
f9.Height = 400
f9.Align = FieldAlignEnum.CenterMiddle
f9.MarginLeft = 100
f9.Visible = True
f9.Font.Bold = True
f9.Font.Size = 10
s.Fields.Add(f9)
Dim f10 As New TextField()
f10.Name = "SMCount"
f10.Text = "=Count(Title, Title = ""Sales Manager"")"
f10.Left = 5800
```

```
    f10.Top = 0
    f10.Width = 1000
    f10.Height = 400
    f10.Align = FieldAlignEnum.CenterMiddle
    f10.MarginLeft = 100
    f10.Visible = True
    f10.Font.Bold = True
    f10.Font.Size = 10
    s.Fields.Add(f10)
    Dim f11 As New TextField()
    f11.Name = "ISCCount"
    f11.Text = "=Count(Title, Title = ""Inside Sales Coordinator"")"
    f11.Left = 7500
    f11.Top = 0
    f11.Width = 1000
    f11.Height = 400
    f11.Align = FieldAlignEnum.CenterMiddle
    f11.MarginLeft = 100
    f11.Visible = True
    f11.Font.Bold = True
    f11.Font.Size = 10
    s.Fields.Add(f11)
    c1FlexReport1.Render()
End Sub
```

C#

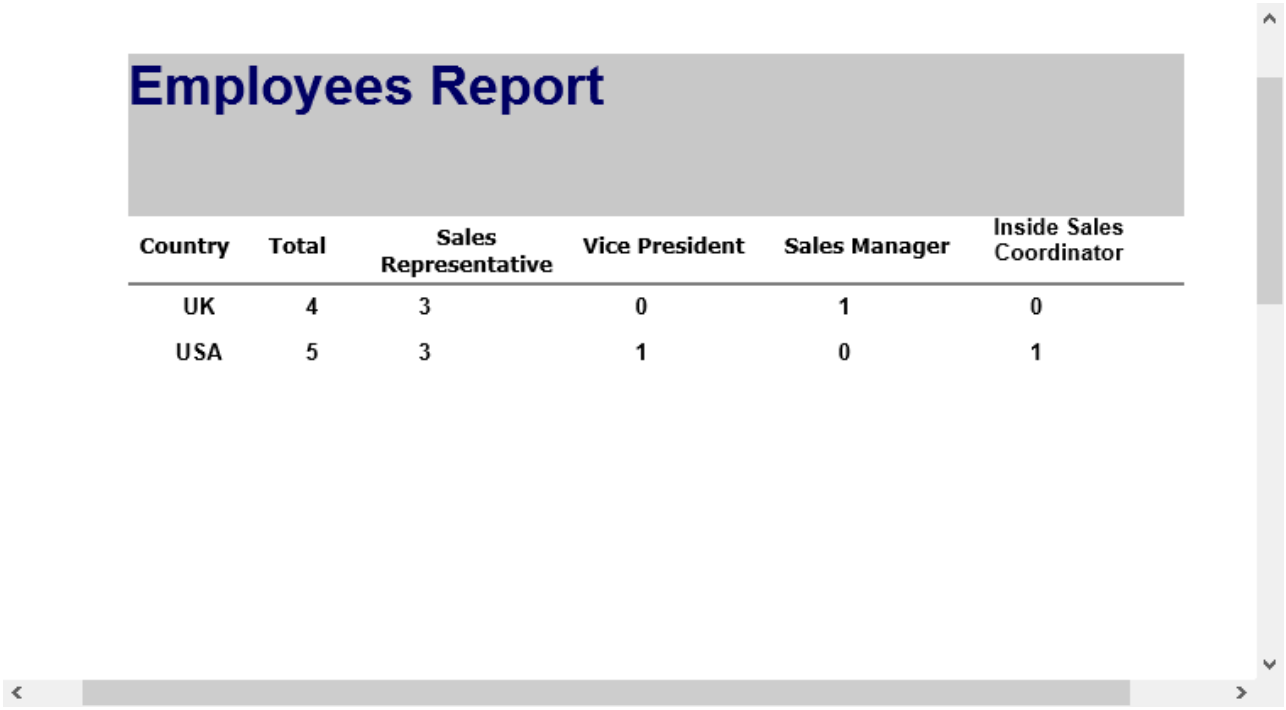
```
Group grp2;
        private void crossC1Button_Click(object sender, EventArgs e)
        {

    btnEmployees.PerformClick();
    c1FlexReport1.Sections.Detail.Visible = false;

    grp2 = c1FlexReport1.Groups.Add("GrpCountry", "Country",
SortEnum.Ascending);
            // format the Header section for the new group
    c1FlexReport1.Sections.PageHeader.Height = 600;
    shpfld2.Top = 600;
    s = grp2.SectionHeader;
    s.Height = 400;
    s.AutoHeight = AutoSizeBehavior.GrowAndShrink;
    s.Visible = true;
    textFld4.Text = "Country";
    textFld4.Width = 1000;
    textFld4.Align = FieldAlignEnum.CenterMiddle;
    textFld4.Height = 400;
    textFld4.Font.Bold = true;
    textFld4.Font.Size = 10;
    textFld5.Text = "Total";
    textFld5.Width = 1000;
    textFld5.Left = 1000;
    textFld5.Align = FieldAlignEnum.CenterMiddle;
    textFld5.Height = 400;
    textFld5.Font.Bold = true;
```

```
textFld5.Font.Size = 10;
textFld6.Text = "Sales Representative";
textFld6.Width = 2000;
textFld6.Left = 2000;
textFld6.Align = FieldAlignEnum.CenterMiddle;
textFld6.Height = 500;
textFld6.Font.Bold = true;
textFld6.Font.Size = 10;
textFld7.Text = "Vice President";
textFld7.Width = 1500;
textFld7.Left = 4000;
textFld7.Align = FieldAlignEnum.CenterMiddle;
textFld7.Height = 400;
textFld7.Font.Bold = true;
textFld7.Font.Size = 10;
textFld8.Text = "Sales Manager";
textFld8.Width = 1500;
textFld8.Left = 5800;
textFld8.Align = FieldAlignEnum.CenterMiddle;
textFld8.Height = 400;
textFld8.Font.Bold = true;
textFld8.Font.Size = 10;
TextField isc = new TextField();
 isc.Text = "Inside Sales Coordinator";
 isc.Width = 1500;
 isc.Left = 7500;
 isc.Align = FieldAlignEnum.CenterMiddle;
 isc.Height = 400;
 isc.Font.Bold = true;
 isc.Font.Size = 10;
 isc.Visible = true;
 c1FlexReport1.Sections.PageHeader.Fields.Add(isc);
 Field f1 = new Field();
 f1.Name = "Country";
 f1.Text = "Country";
 f1.Left = 80;
 f1.Top = 0;
 f1.Width = 1000;
 f1.Height = 400;
 f1.Align = FieldAlignEnum.CenterMiddle;
 f1.MarginLeft = 100;
 f1.Calculated = true;
 f1.Visible = true;
 f1.BackColor = Color.Transparent;
 f1.Font.Bold = true;
 f1.Font.Size = 10;
 s.Fields.Add(f1);
 TextField f7 = new TextField();
 f7.Name = "RunCount";
 f7.Text = "=Count(Title)";
 f7.Left = 1080;
 f7.Top = 0;
 f7.Width = 1000;
 f7.Height = 400;
```

```
f7.Align = FieldAlignEnum.CenterMiddle;
f7.MarginLeft = 100;
f7.Visible = true;
f7.Font.Bold = true;
f7.Font.Size = 10;
s.Fields.Add(f7);
TextField f8 = new TextField();
f8.Name = "SRCount";
f8.Text = "=Count(Title, Title = \"Sales Representative\")";
f8.Left = 2080;
f8.Top = 0;
f8.Width = 1000;
f8.Height = 400;
f8.Align = FieldAlignEnum.CenterMiddle;
f8.MarginLeft = 100;
f8.Visible = true;
f8.Font.Bold = true;
f8.Font.Size = 10;
s.Fields.Add(f8);
TextField f9 = new TextField();
f9.Name = "VPCount";
f9.Text = "=Count(Title, Title = \"Vice President, Sales\")";
f9.Left = 4000;
f9.Top = 0;
f9.Width = 1000;
f9.Height = 400;
f9.Align = FieldAlignEnum.CenterMiddle;
f9.MarginLeft = 100;
f9.Visible = true;
f9.Font.Bold = true;
f9.Font.Size = 10;
s.Fields.Add(f9);
TextField f10 = new TextField();
f10.Name = "SMCount";
f10.Text = "=Count(Title, Title = \"Sales Manager\")";
f10.Left = 5800;
f10.Top = 0;
f10.Width = 1000;
f10.Height = 400;
f10.Align = FieldAlignEnum.CenterMiddle;
f10.MarginLeft = 100;
f10.Visible = true;
f10.Font.Bold = true;
f10.Font.Size = 10;
s.Fields.Add(f10);
TextField f11 = new TextField();
f11.Name = "ISCCount";
f11.Text = "=Count(Title, Title = \"Inside Sales Coordinator\")";
f11.Left = 7500;
f11.Top = 0;
f11.Width = 1000;
f11.Height = 400;
f11.Align = FieldAlignEnum.CenterMiddle;
f11.MarginLeft = 100;
```

```
        f11.Visible = true;
        f11.Font.Bold = true;
        f11.Font.Size = 10;
        s.Fields.Add(f11);
        c1FlexReport1.Render();
        }
```

5. Run the project. Click 'View Cross Tab-Report' button to view cross-tab report. Observe that the details corresponding to the employee titles in the two countries are shown.



Employees Report

Country	Total	Sales Representative	Vice President	Sales Manager	Inside Sales Coordinator
UK	4	3	0	1	0
USA	5	3	1	0	1

Sorting Data

Sorting is another way to organize data in ascending or descending order.

In FlexReport, sorting is achieved by using DataSource.SortDefinitions.

Lets say you want to view the list of employees with their names in ascending order. In this case the list should be sorted by First Name. The following steps illustrate how to Sort the names of the list of employees in alphabetical order. This example uses sample created in FlexReport Quick Start.

1. Add a **C1Button** to the form in the FlexReport Quick Start project.
2. Set the C1Button **Name** to 'sortC1Button' and **Text** to 'Sort Report by Employee First Name'.
3. Create **Click** event as sortC1Button_Click.
4. Add the following code.

Visual Studio

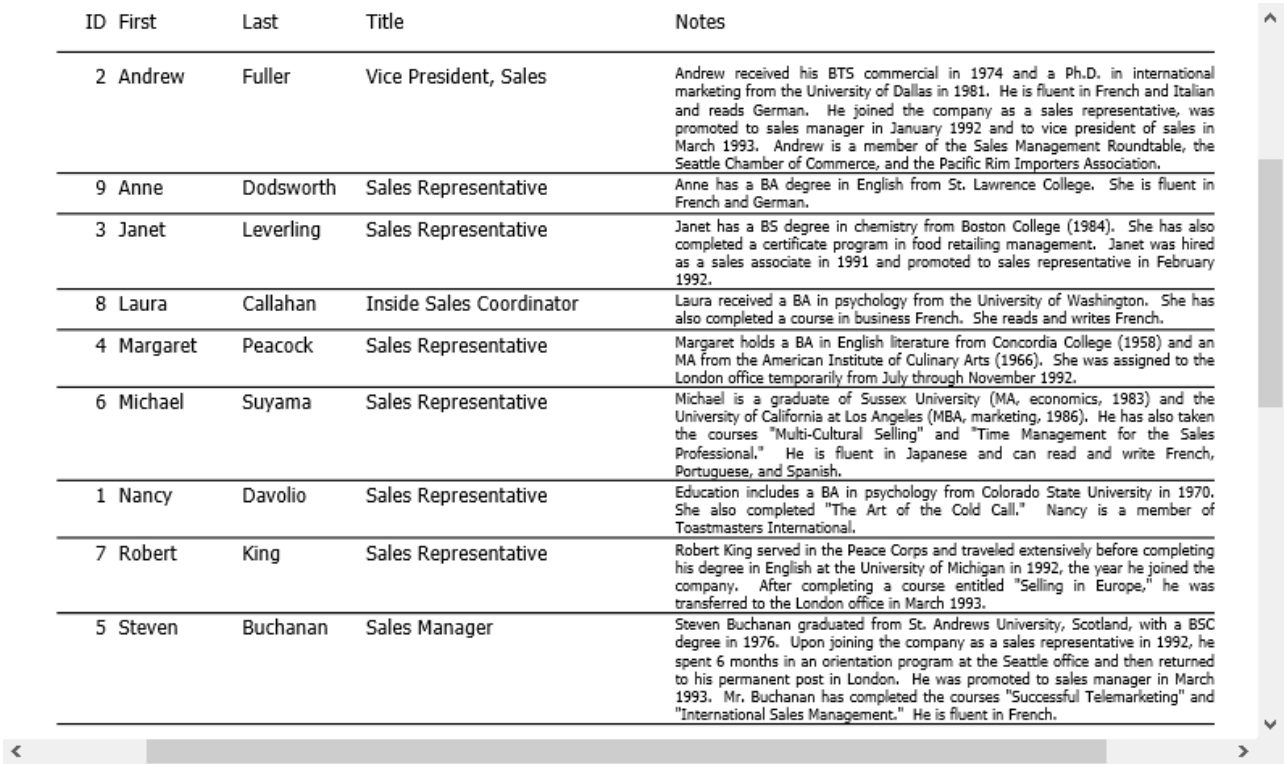
```
Private asc As Boolean = True
    Private Sub sortC1Button_Click(sender As Object, e As EventArgs)
Handles Button2.Click
        If asc Then
            Dim sd As New SortDefinition("[FirstName]",
SortDirection.Ascending)
            C1FlexReport1.DataSource.SortDefinitions.Add(sd)
            asc = False
        Else
```

```
            btnEmployees.PerformClick()
            asc = True
        End If
        C1FlexReport1.Render()
    End Sub
```

C#

```
 bool asc = true;
        private void sortC1Button_Click(object sender, EventArgs e)
        {
            if (asc)
            {
                SortDefinition sd = new SortDefinition("[FirstName]",
SortDirection.Ascending);
                c1FlexReport1.DataSource.SortDefinitions.Add(sd);
                asc = false;
            }
            else
            {
                btnEmployees.PerformClick();
                asc = true;
            }
            c1FlexReport1.Render();
        }
```

5. Preview the report. Click Employees button to render the report.
6. Click 'Sort Report by Employee First Name' button to view sorting in the report.



ID	First	Last	Title	Notes
2	Andrew	Fuller	Vice President, Sales	Andrew received his BTS commercial in 1974 and a Ph.D. in international marketing from the University of Dallas in 1981. He is fluent in French and Italian and reads German. He joined the company as a sales representative, was promoted to sales manager in January 1992 and to vice president of sales in March 1993. Andrew is a member of the Sales Management Roundtable, the Seattle Chamber of Commerce, and the Pacific Rim Importers Association.
9	Anne	Dodsworth	Sales Representative	Anne has a BA degree in English from St. Lawrence College. She is fluent in French and German.
3	Janet	Leverling	Sales Representative	Janet has a BS degree in chemistry from Boston College (1984). She has also completed a certificate program in food retailing management. Janet was hired as a sales associate in 1991 and promoted to sales representative in February 1992.
8	Laura	Callahan	Inside Sales Coordinator	Laura received a BA in psychology from the University of Washington. She has also completed a course in business French. She reads and writes French.
4	Margaret	Peacock	Sales Representative	Margaret holds a BA in English literature from Concordia College (1958) and an MA from the American Institute of Culinary Arts (1966). She was assigned to the London office temporarily from July through November 1992.
6	Michael	Suyama	Sales Representative	Michael is a graduate of Sussex University (MA, economics, 1983) and the University of California at Los Angeles (MBA, marketing, 1986). He has also taken the courses "Multi-Cultural Selling" and "Time Management for the Sales Professional." He is fluent in Japanese and can read and write French, Portuguese, and Spanish.
1	Nancy	Davolio	Sales Representative	Education includes a BA in psychology from Colorado State University in 1970. She also completed "The Art of the Cold Call." Nancy is a member of Toastmasters International.
7	Robert	King	Sales Representative	Robert King served in the Peace Corps and traveled extensively before completing his degree in English at the University of Michigan in 1992, the year he joined the company. After completing a course entitled "Selling in Europe," he was transferred to the London office in March 1993.
5	Steven	Buchanan	Sales Manager	Steven Buchanan graduated from St. Andrews University, Scotland, with a BSC degree in 1976. Upon joining the company as a sales representative in 1992, he spent 6 months in an orientation program at the Seattle office and then returned to his permanent post in London. He was promoted to sales manager in March 1993. Mr. Buchanan has completed the courses "Successful Telemarketing" and "International Sales Management." He is fluent in French.

Filtering Data

Filtering a data is important where you want to view only a portion of data based on certain criteria. In FlexReport, the

data is filtered by using DataSource.Filter.

Lets say you want to view the employee detail corresponding to an Employee ID for the report created in FlexReport Quick Start. Add the following code where EmployeeID field is added in the Detail section, to Filter the employee detail corresponding to the 'EmployeeID = 2'.

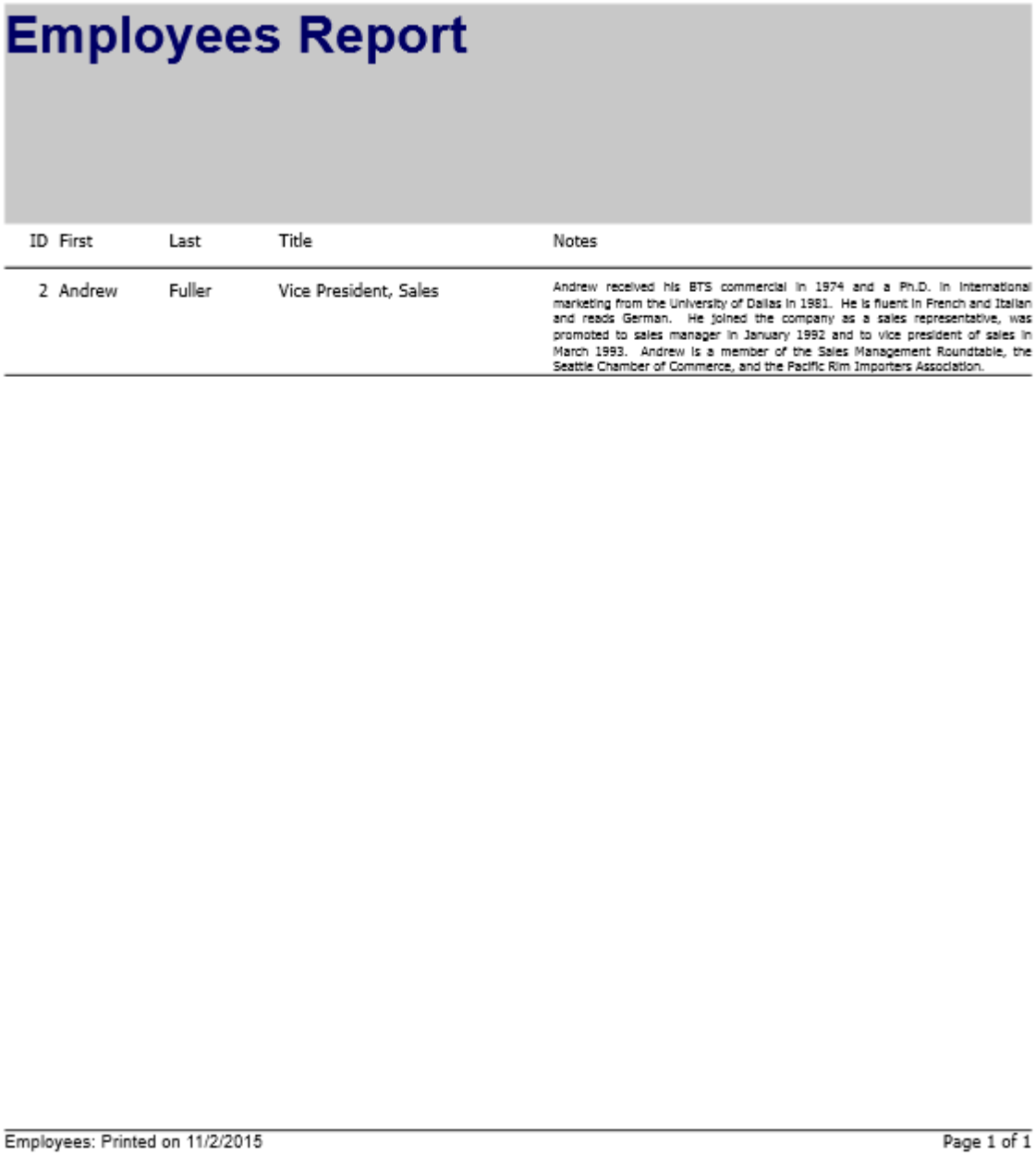
Visual Basic

```
C1FlexReport1.DataSource.Filter = "EmployeeID = 2"
```

C#

```
c1FlexReport1.DataSource.Filter = "EmployeeID = 2";
```

Run the project. You see that the First name, Last name, Title, and Notes for the employee with EmployeeID = 2 is displayed.



Exporting Reports to Various Formats

Export using FlexViewer

With **FlexViewer** control, you can preview the reports as well as export them. The reports can be exported to several file formats: .pdf, .html, .rtf, .docx, .xls, .xlsx, .zip, .tiff, .bmp, .png, .jpg, and .gif. The following code describes how to export a FlexReport using Export method of C1FlexViewer Class:

Visual Basic

```
'Load report definition
C1FlexReport1.Load(@"reportFile", "reportName")

'Specify the report shown by the viewer
C1FlexViewer1.DocumentSource = C1FlexReport1
```

```
'Export
C1FlexViewer1.Export()
```

C#

```
//Load report definition
c1FlexReport1.Load(@"reportFile", "reportName");

//Specify the report shown by the viewer
c1FlexViewer1.DocumentSource = c1FlexReport1;

//Export
c1FlexViewer1.Export();
```

Export to PDF using FlexReport

The following code describes how to export a FlexReport to PDF using PdfFilter class. Similarly, you can also export the report to other formats mentioned above.

- **Visual Basic**

```
'create report object
Dim c1FlexReport1 As New C1FlexReport()

'Load a report
c1FlexReport1.Load("../../ProductsReport.flxr", "Products Report")
c1FlexReport1.Render()

'Create PdfFilter object
Dim filter As New C1.Win.C1Document.Export.PdfFilter()
filter.ShowOptions = False

'Give file name and path where exported file will be saved
filter.FileName = "Products Report" + "../../ProductsReport.pdf"
'The report is exported as ProductsReport.pdf in bin\debug folder

'Export
c1FlexReport1.RenderToFilter(filter)
```

- **C#**

```
//create report object
C1FlexReport c1FlexReport1 = new C1FlexReport();

//Load a report
c1FlexReport1.Load(@"..\..\ProductsReport.flxr", "Products Report");
c1FlexReport1.Render();

//Create PdfFilter object
C1.Win.C1Document.Export.PdfFilter filter = new C1.Win.C1Document.Export.PdfFilter();
filter.ShowOptions = false;

//Give file name and path where exported file will be saved
filter.FileName = "Products Report" + @"..\..\ProductsReport.pdf";
//The report is exported as ProductsReport.pdf in bin\debug folder

//Export
c1FlexReport1.RenderToFilter(filter);
```

Working with VBScript

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

Note the following differences between VBScript expressions and statements:

- **Expressions** return values, you can assign them to things like **Field.Text**, for example:
  `Field1.Text.Expression = "iif( 1=1, 1+2, 1+3 )"`
- **Statements** don't return values. You can assign them to event properties like **OnFormat**. For example:
  `c1FlexReport.OnOpen = "if 1=1 then msgbox("OK!!!") else msgbox("ooops")"`

C1FlexReport relies on VBScript to evaluate expressions in calculated fields and to handle report events.

VBScript is a full-featured language, and you have access to all its methods and functions when writing C1FlexReport expressions. For the intrinsic features of the VBScript language, refer to the Microsoft Developer's Network (MSDN).

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

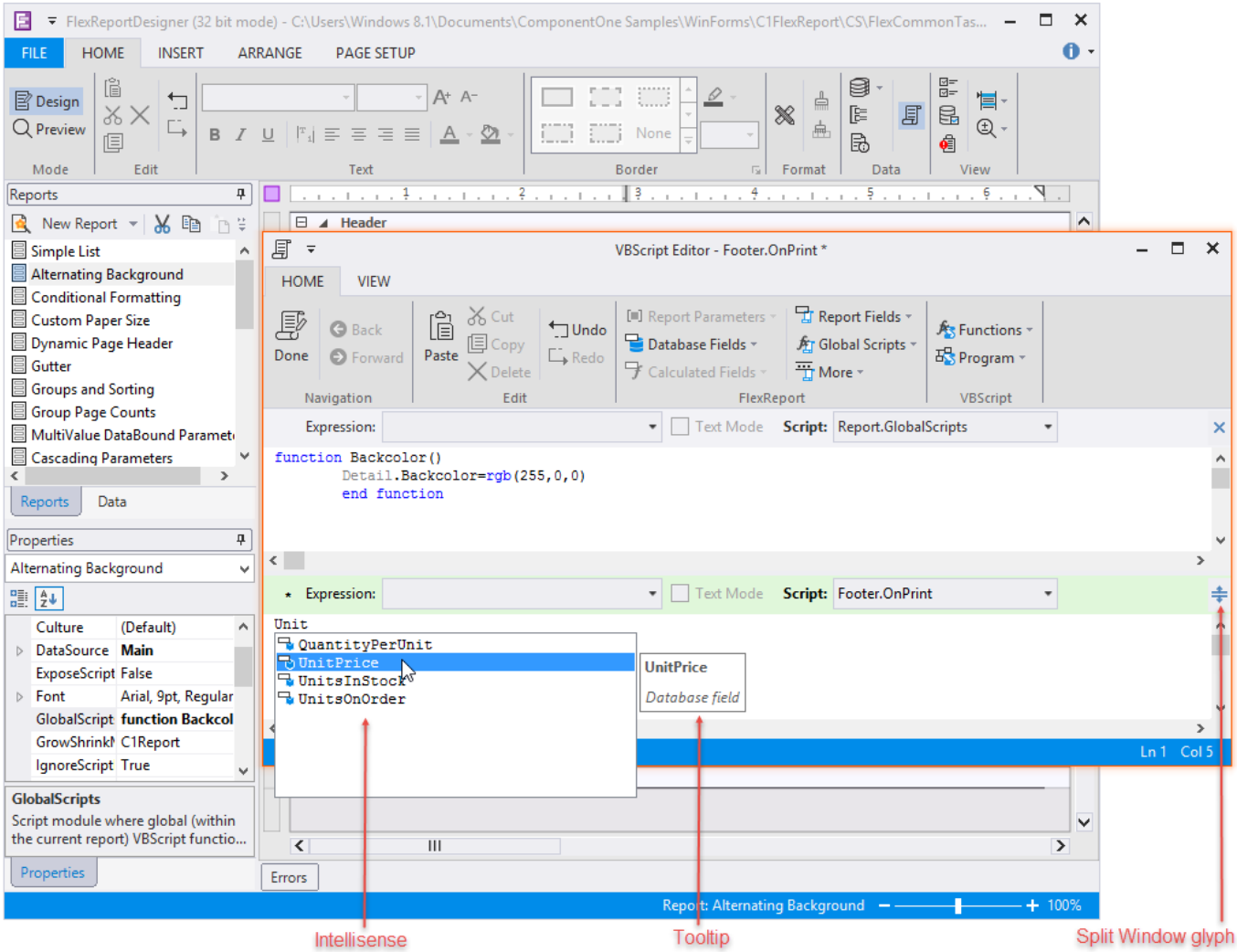
To write global scripts using **VBScript Editor** option,

1. Go to Home Tab of **C1FlexReportDesigner**.
2. Click **VBScript Editor** and write desired global script; for example,

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

You can also write global scripts using **GlobalScripts** property of **C1FlexReportDesigner** as follows:

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



So, you have defined a global function 'Backcolor()', which can be used throughout the report.

The VBScript Editor has the following additional features:

- **IntelliSense**: Provides auto code completion prompts for the scripts supported by VBScript Editor. IntelliSense in VBScript Editor has following features:
  - The IntelliSense window that displays a context-dependent list of available words also displays a detailed help on VBScript functions and keywords in a small tooltip or help window. The italic font on the detailed help basically shows the category to which the current item belongs (such as 'VBScript function', 'C1FlexReport aggregate script function', .NET object property, and so on).
  - On editing DataSource.Filter, the editor opens as **Expression Editor - DataSource.Filter** and IntelliSense shows keywords or functions available in that with corresponding help.
  - Icons associated with IntelliSense entries indicate the type of the entry. The icons' color palette is different for VBScript, report built-in stuff, and DataSource.Filter.
  - When a user types and Intellisense window is opened, the list is filtered according to the letters being typed for example, typing 't' will only show words that contain the letter 't', typing 'te' will narrow the list to words that contain 'te', and so on.
  - Backspace in the IntelliSense window undoes the last filter.
  - Pressing square bracket '**[**' shows the list of available db fields.
  - Pressing dot '**.**' after the name of an object such as a report, field, or section shows the .NET properties available for that object
  - Pressing Ctrl+J, Ctrl+Space, or a letter after a non-letter character shows the list of available VBScript functions, keywords, etc. depending on the context.

- **Split Window**: Lets you view or write two same or different scripts in single VBScriptEditor. By default, the VBScript editor opens as a single window.

  To switch to Split Window
  Switch to the split window mode by clicking the split window glyph and dragging it down to open another editor at the top. The windows can be resized by dragging the divider between the windows.

  To switch back to the single window
  Click the 'x' glyph on the top right corner of the window to close the top window, turn the split mode off, and zoom out the bottom window. The enabled or disabled state of ribbon buttons depends on the current window, which is shown with a light green caption bar. The split window mode has following additional functionalities:
  - Switch between the two windows by pressing **F6**.
  - Hide the top window in a split window mode by dragging the split window glyph or the divider line high enough across the top window.


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

C1FlexReport extends VBScript by exposing additional objects, variables, and functions. These extensions are described in the following sections.

VBScript Elements, Objects, and Variables

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

Operators

The following table contains the VBScript operators:

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

Reserved symbols

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

Keyword	Description
True	The **True** keyword has a value equal to -1.
False	The **False** keyword has a value equal to 0.
Nothing	Used to disassociate an object variable from any actual object. To assign **Nothing** to an object variable, use the **Set** statement, for example: `Set MyObject = Nothing` Several object variables can refer to the same actual object. When **Nothing** is assigned to an object variable, that variable no longer refers to any actual object. When several object variables refer to the same object, memory and system resources associated with the object to which the

Keyword	Description
	variables refer are released only after all of them have been set to **Nothing**, either explicitly using **Set**, or implicitly after the last object variable set to **Nothing**.
Null	The **Null** keyword is used to indicate that a variable contains no valid data.
vbCr	When you call print and display functions, you can use the following constants in your code in place of the actual values.
vbCrLf	When you call print and display functions, you can use the following constants in your code in place of the actual values.
vbLf	When you call print and display functions, you can use the following constants in your code in place of the actual values.
vbFormFeed	When you call print and display functions, you can use the following constants in your code in place of the actual values.
vbNewLine	When you call print and display functions, you can use the following constants in your code in place of the actual values.
vbNullChar	When you call print and display functions, you can use the following constants in your code in place of the actual values.
vbTab	When you call print and display functions, you can use the following constants in your code in place of the actual values.
vbVerticalTab	When you call print and display functions, you can use the following constants in your code in place of the actual values.
vbBlack	Black. Value = 0x0.
vbRed	Red. Value = 0xFF.
vbGreen	Green. Value = 0xFF00.
vbYellow	Yellow. Value = 0xFFFF.
vbBlue	Blue. Value = 0xFF0000.
vbMagenta	Magenta. Value = 0xFF00FF.
vbCyan	Cyan. Value = 0xFFFF00.
vbWhite	White. Value = 0xFFFFFF.

Built-in functions

The VBScript built-in functions are listed below:

Abs	Date	Iif	Minute	Sign
Acos	DateAdd	InputBox	Month	Space
Asc	DateDiff	InStr	MonthName	Sqr
Asin	DatePart	InStrRev	MsgBox	StrComp
Atn	DateSerial	Int	Now	String
CBool	DateValue	IsDate	Oct	Tan

CByte	Day	IsEmpty	Pi	Time
CCur	Exp	IsNull	Replace	Timer
CDate	Fix	IsNumeric	RGB	TimeSerial
CDbl	Format	IsObject	Right	TimeValue
Chr	FormatCurrency	LCase	Rnd	Trim
CInt	FormatDateTime	Left	Round	TypeName
CLng	FormatNumber	Len	RTrim	UCase
Cos	FormatPercent	Log	Second	WeekDay
CSng	Hex	LTrim	Sgn	WeekDayName
CStr	Hour	Mid	Sin	Year

For more information on the VBScript functions, see the MSDN documentation.

The key features of VBScript that are part of C1FlexReport are as follows:

- Aggregate functions (Sum, Average, StDev, Var, Count, and so on)
- Report and Database field names
- Page/Pages variables
- Report objects
- String functions (Chr, Format, and so on)
- Data Conversion (CBool, CByte, an so on)
- Math functions (cos, sin, and so on)
- Date/Time functions (DateAdd, Hour, and so on)
- Functions and Subs
- Conditional statements
- Built-in functions (Like and In)

Built-in script functions, **Like** and **In** have functionality similar to SQL operators LIKE and IN and return True or False. Like(str, template): Compares 'str' to 'template', which can contain wildcard '%'. Some examples of **Like** function are as follows:

- Like("abc", "%bc") returns True.
- Like("abc", "%bcd") returns False.
- Like("abc", "ab%") returns True.
- Like("abc", "abd%") returns False.
- Like("abc", "%b%") returns True.
- Like("abc", "%d%") returns False.
- Like("abc", "abc") returns True.
- Like("abc", "abcd") returns False.
- Like("Abc", "abc") returns False.

In(obj, obj1, ... objN): Tests whether 'obj' is among objects 'obj1', ... , 'objN'. Some examples of **In** function are as follows:

- In(1,1,2,3) returns True.
- In(1,2,3) returns False.
- In("a", "a", "b", "c") returns True.
- In("a", "b", "c") returns False.
- In("A", "a", "b", "c") returns False.

As you can observe, both the functions are case-sensitive, so "abc" is not the same as "Abc".

Note that the following VBScript features are **not** supported in C1FlexReport:

- Arrays
- Select/Case statements

Statement keywords

The VBScript statement keywords are listed below:

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

Report Field Names

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

```
MyField.BackColor = RGB(200,250,100)
MyField.Font.Size = 14
MyField * 2 ' (same as MyField.Value * 2)
```

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

Report Section Names

Names of Section objects are evaluated and return a reference to the object, so you can access the section's properties. The default property for the Section object is Name. For example:
If Page = 1 Then [Page Footer].Visible = False

Database Field Names

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

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

Report Variables

Page

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

```
If Country <> LastCountry Then Page = 1
LastCountry = Country
```

Pages

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

Report Object

The report object returns a reference to the control object, so you can access the full C1FlexReport object model from your scripts and expressions. For example:

```
"Fields: " & Report.Fields.Count
```

Cancel

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

```
If Page > 100 Then Cancel = True
```

Compatibility Functions: Iif and Format

To increase compatibility with code written in Visual Basic and Microsoft Access (VBA), C1FlexReport exposes two functions that are not available in VBScript: **Iif** and **Format**.

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

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

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

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

Value Type	Format String	Description
Number	Percent, %	Formats a number as a percentage, with zero or two decimal places. For example: `Format(0.33, "Percent") = "33%"` `Format(0.3333333, "Percent") = "33.33%"`
	#,###.##0	Formats a number using a mask. The following symbols are recognized: # digit placeholder 0 digit placeholder, force display , use thousand separators (enclose negative values in parenthesis % format as percentage For example: `Format(1234.1234, "#,###.##") = "1,234.12"` `Format(-1234, "#.00") = "(1234.12)"` `Format(.1234, "#.##") = ".12"` `Format(.1234, "0.##") = "0.12"` `Format(.3, "#.##%") = "30.00%"`
Currency	Currency, $	Formats a number as a currency value. Displays number with thousand separator, if appropriate; displays two digits to the right of the decimal separator. For example: `Format(1234, "$") = "$1,234.00"`
Boolean	Yes/No	Returns "Yes" or "No".
Date	Long Date	`Format(#12/5/1#, "long date") = "December 5, 2001"`
	Short Date	`Format(#12/5/1#, "short date") = "12/5/2001"`
	Medium Date	`Format(#12/5/1#, "medium date") = "05-Dec-01"`
	q,m,d,w,yyyy	Returns a date part (quarter, month, day of the month, week of the year, year). For example: `Format(#12/5/1#, "q") = "4"`
String	@@-@@/@@	Formats a string using a mask. The "@" character is a placeholder for a

Value Type	Format String	Description
		single character (or for the whole value string if there is a single "@"). Other characters are intercodeted as literals. For example: `Format("AC55512", "@@-@@/@@") = "AC-555/12"` `Format("AC55512", "@") = "AC55512"`
	@;Missing	Uses the format string on the left of the semi-colon if the value is not null or an empty string, otherwise returns the part on the right of the semi-colon. For example: `Format("@;Missing", "UK") = "UK"` `Format("@;Missing", "") = "Missing"`

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

Aggregate Functions

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

All FlexReport aggregate functions take two arguments:

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

FlexReport defines the following aggregate functions:

Function	Description
Avg	Average value of the expression within the current group. For example, the following expression calculates the average sales for the whole group and the average sales for a certain type of product: `Avg(SalesAmount)` `Avg(SalesAmount, ProductType = 3)`
Sum	Sum of all values in the group.
Count	Count of records in the group with non-null values. Use an asterisk for the expression to include all records. For example, the following expressions count the number of employees with valid (non-null) addresses and the total number of employees: `Count(Employees.Address)` `Count(*)`
CountDistinct	Count of records in the group with distinct non-null values.
Min, Max	Minimum and maximum values for the expression. For example: `"Min Sale = " & Max(SaleAmount)`
Range	Range between minimum and maximum values for the expression.
StDev, Var	Standard deviation and variance of the expression in the current group. The values are calculated using the sample (n-1)

Function	Description
	formulas, as in SQL and Microsoft Excel.
StDevP, VarP	Standard deviation and variance of the expression in the current group. These values are calculated using the population (n) formulas, as in SQL and Microsoft Excel.
Median	Returns median from the values in the group.
Mode	Returns mode from the values in the group.

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

For example, the "Employee Sales by Country" report in the sample **NWind.xml** file contains several aggregate fields. The report groups records by Country and by Employee.

The **SalespersonTotal** field in the Footer section of the Employee group contains the following expression:

```
=Sum([SaleAmount])
```

Because the field is in the Employee group footer, the expression returns the total sales for the current employee.

The **CountryTotal** and **GrandTotal** fields contain exactly the same expression. However, because these fields are in the Country group footer and report footer, the expression returns the total sales for the current country and for the entire recordset.

You may need to refer to a higher-level aggregate from within a group. For example, in the "Employee Sales by Country" report, there is a field that shows sales in the current country as a percentage of the grand total. Since all aggregates calculated within a country group refer to the current country, the report cannot calculate this directly. Instead, the **PercentOfGrandTotal** field uses the following expression:

```
=[CountryTotal]/[GrandTotal]
```

CountryTotal and **GrandTotal** are fields located in the Country and Report Footer sections. Therefore, **CountryTotal** holds the total for the current country and **GrandTotal** holds the total for the whole recordset.

It is important to realize that evaluating aggregate functions is time-consuming, since it requires the control to traverse the recordset. Because of this, you should try to use aggregate functions in a few calculated fields only. Other fields can then read the aggregate value directly from these fields, rather than evaluating the aggregate expression again.

For example, the "Employee Sales by Country" report in the NorthWind database has a detail field, **PercentOfCountryTotal**, that shows each sale as a percentage of the country's total sales. This field contains the following expression:

```
=[SaleAmount]/[CountryTotal]
```

SaleAmount is a reference to a recordset field, which varies for each detail record. **CountryTotal** is a reference to a report field that contains an aggregate function. When the control evaluates this expression, it gets the aggregate value directly from the report field, and does not recalculate the aggregate.

For the complete report, see report "Employee Sales by Country" in the **Nwind.xml** report definition file, which is available in the **ComponentOne Samples** folder.

Managing Splitting of FlexReport Objects

Since creating reports is all about representing data, it is important to control the fit of the objects depending upon their height and width.

In FlexReports, any Section or Sub-section can be forced to split or not to split (keep together) between the pages by setting SplitBehavior property to **SplitIfNeeded** or **KeepTogether**. Similarly, the splitting of Fields and Borders is governed by SplitHorzBehavior and SplitVertBehavior properties.

The following code sets the **SplitBehavior** for a Section and a Sub-section:

Visual Studio

```
'Allow section to split if needed
  C1FlexReport1.Sections.Header.SplitBehavior =
SplitBehavior.SplitIfNeeded

'Allow sub-section to split if needed
  C1FlexReport1.Sections.Header.SubSections(0).SplitBehavior =
SplitBehavior.SplitIfNeeded
```

C#

```
// Allow section to split if needed
   c1FlexReport1.Sections.Header.SplitBehavior =
SplitBehavior.SplitIfNeeded;

// Allow sub-section to split if needed
   c1FlexReport1.Sections.Header.SubSections[0].SplitBehavior =
SplitBehavior.SplitIfNeeded;
```

Modifying the Fields

You are not restricted to using VBScript to evaluate expressions in calculated fields. You can also specify scripts that are triggered when the report is rendered, and you can use those to change the formatting of the report.These scripts are contained in *event properties*. An event property is similar to a Visual Basic event handler, except that the scripts are executed in the scope of the report rather than in the scope of the application that is displaying the report.

For example, you can use an event property to set a field's Font and ForeColor properties depending on its value. This behavior would then be a part of the report itself, and would be preserved regardless of the application used to render it.

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

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

Object	Property	Description
C1FlexReport	OnOpen	Fired when the report starts rendering. Can be used to modify the ConnectionString or RecordSource properties, or to initialize VBScript variables.
	OnClose	Fired when the report finishes rendering. Can be used to perform clean-up tasks.
	OnNoData	Fired when a report starts rendering but the source recordset is empty. You can set the **Cancel** property to **True** to prevent the report from being generated. You could also show a dialog box to alert the user as to the reason why no

Object	Property	Description
		report is being displayed.
	OnPage	Fired when a new page starts. Can be used to set the **Visible** property of sections of fields depending on a set of conditions. The control maintains a Page variable that is incremented automatically when a new page starts.
	OnError	Fired when an error occurs.
Section	OnFormat	Fired before the fields in a section are evaluated. At this point, the fields in the source recordset reflect the values that will be rendered, but the report fields do not.
	OnPrint	Fired before the fields in a section are printed. At this point, the fields have already been evaluated and you can do conditional formatting.

The following topics illustrate typical uses for these properties.

Formatting a Field According to Its Value

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

To highlight products that are below the reorder level using code:

To highlight products that are below the reorder level by displaying their name in bold red characters, use an event script that looks like this:

Visual Basic

```
Dim script As String = _
  "If UnitsInStock < ReorderLevel Then" & vbCrLf & _
  "ProductNameCtl.ForeColor = RGB(255,0,0)" & vbCrLf & _
  "ProductNameCtl.Font.Bold = True" & vbCrLf & _
  "Else" & vbCrLf & _
  "ProductNameCtl.ForeColor = RGB(0,0,0)" & vbCrLf & _
  "ProductNameCtl.Font.Bold = False" & vbCrLf & _
  "End If"
C1Flexreport.Sections.Detail.OnPrint = script
```

C#

```
 string script =
      "if (UnitsInStock& ReorderLevel) then\r\n" +
      "ProductNameCtl.ForeColor = rgb(255,0,0)\r\n" +
      "ProductNameCtl.Font.Bold = true\r\n" +
      "else\r\n" +
      "ProductNameCtl.ForeColor = rgb(0,0,0)\r\n" +
      "ProductNameCtl.Font.Bold = false\r\n" +
      "end if\r\n";
 c1FlexReport1.Sections.Detail.OnPrint = script;
```

To highlight products that are below the reorder level using FlexReportDesigner:

Alternatively, instead of writing the code, you can use the **C1FlexReportDesigner** application to type the following script code directly into the VBScript Editor of the Detail section's **Section.OnPrint** property. Complete the following steps:

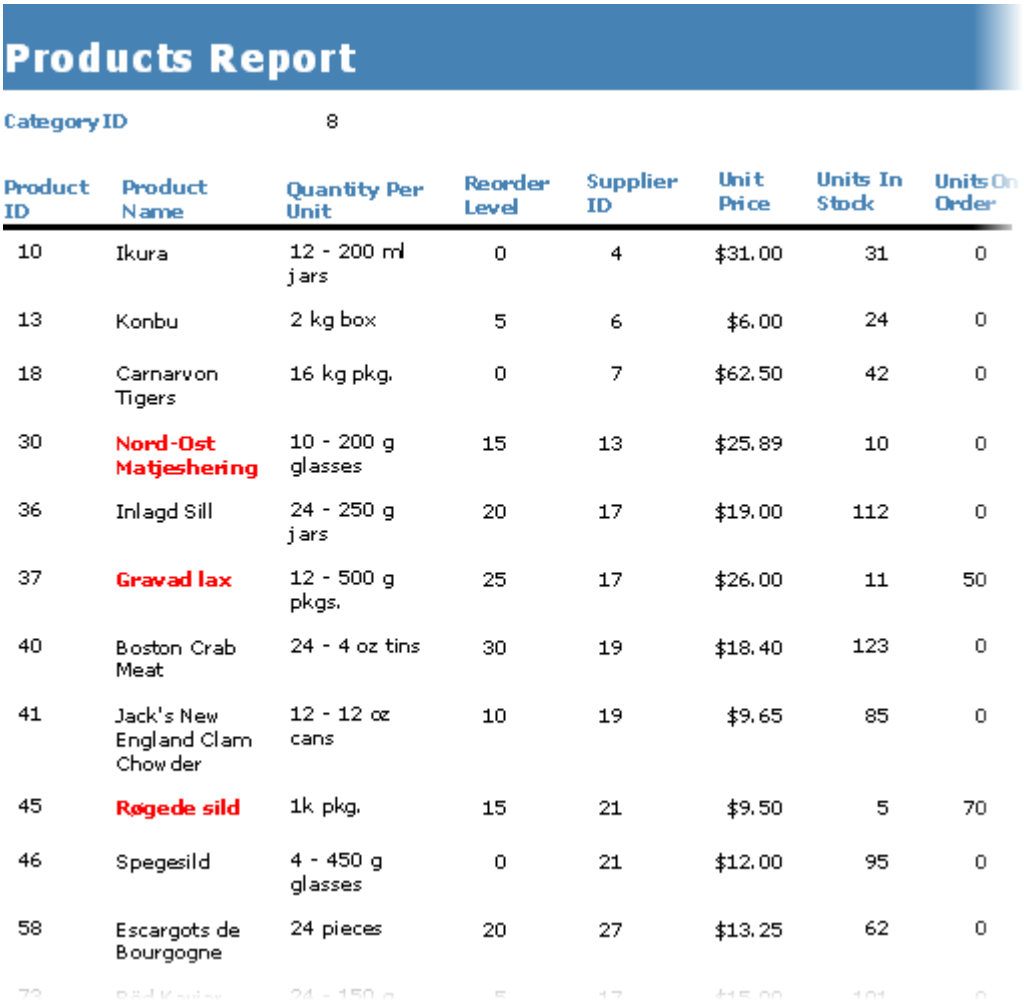
1. Select **Detail** from the Properties window drop-down list in the Designer. This reveals the section's available properties.
2. Click the empty box next to the **Section**.**OnPrint** property, then click the drop-down arrow, and select **Expression Editor** from the list. **VBScript Editor** window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

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

4. Click **OK** to close the editor.

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

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



Products Report

Category ID 8

Product ID	Product Name	Quantity Per Unit	Reorder Level	Supplier ID	Unit Price	Units In Stock	Units On Order
10	Ikura	12 - 200 ml jars	0	4	$31.00	31	0
13	Konbu	2 kg box	5	6	$6.00	24	0
18	Carnarvon Tigers	16 kg pkg.	0	7	$62.50	42	0
30	Nord-Ost Matjeshering	10 - 200 g glasses	15	13	$25.89	10	0
36	Inlagd Sill	24 - 250 g jars	20	17	$19.00	112	0
37	Gravad lax	12 - 500 g pkgs.	25	17	$26.00	11	50
40	Boston Crab Meat	24 - 4 oz tins	30	19	$18.40	123	0
41	Jack's New England Clam Chowder	12 - 12 oz cans	10	19	$9.65	85	0
45	Røgede sild	1k pkg.	15	21	$9.50	5	70
46	Spegesild	4 - 450 g glasses	0	21	$12.00	95	0
58	Escargots de Bourgogne	24 pieces	20	27	$13.25	62	0

Hiding a Section If there is No Data

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

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

```
If isnull(PictureFieldName) Then
  Detail.Visible = false
 Else
  Detail.Visible = true
End If
```

To hide a section if there is no data for it using code:

To hide a section if there is no data, in this case a record's image, for it, use an event script that looks like this:

Visual Basic

```
C1FlexReport1.Sections.Detail.OnFormat = "Detail.Visible =
notisnull(PictureFieldName)"
```

C#

```
c1FlexReport1.Sections.Detail.OnFormat = "Detail.Visible =
notisnull(PictureFieldName)";
```

To hide a section if there is no data for it using FlexReportDesigner:

Alternatively, instead of writing the code, you can use the **C1FlexReportDesigner** to type the following script code directly into the VBScript Editor of the Detail section's **OnFormat** property. Complete the following steps:

1. Select **Detail** from the Properties window drop-down list in the Designer. This reveals the section's available properties.
2. Click the empty box next to the **Section.OnFormat** property, then click the drop-down arrow, and select **Expression Editor** from the list. **VBScript Editor** window appears.
3. In the **VBScript Editor**:
    - Simply type the following script in the window:
      ```
      If isnull(PictureFieldName) Then
      Detail.Visible = false
      Else
      Detail.Visible = true
      End If
      ```
    - Or you could use the more concise version:
      ```
      Detail.Visible = not isnull(PictureFieldName)
      ```

Showing or Hiding a Field Depending on a Value

Instead of changing the field format to highlight its contents, you could set another field's **Visible** property to **True** or **False** to create special effects. For example, if you inserted a new field Shape field named "Shapefld" around the product name and set its Shape property to True, then you could write the script as follows:

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

To highlight products that are below the reorder level using code:

To highlight products that are below the reorder level by displaying a box, use an event script that looks like this:

Visual Basic

```
Dim script As String = _
  "If UnitsInStock < ReorderLevel Then" & vbCrLf & _
  "  BoxCtl.Visible = True" & vbCrLf & _
  "Else" & vbCrLf & _
  "  BoxCtl.Visible = False" & vbCrLf & _
  "End If"
C1FlexReport1.Sections.Detail.OnPrint = script
```

C#

```
string script =
  "if (UnitsInStock < ReorderLevel) then\r\n" +
  "BoxCtl.Visible = true\r\n" +
  "else\r\n" +
  "BoxCtl.Visible = false\r\n" +
  "end if\r\n";
c1FlexReport1.Sections.Detail.OnPrint = script;
```

The code builds a string containing the VBScript event handler, and then assigns it to the section's OnPrint property.

To highlight products that are below the reorder level using FlexReportDesigner:

Alternatively, instead of writing the code, you can use the **C1FlexReportDesigner** application to type the following script code directly into the VBScript Editor of the Detail section's OnPrint property. Complete the following steps:

1. Select **Detail** from the Properties window drop-down list in the Designer. This reveals the section's available properties.
2. Click the ellipses next to the **OnPrint** property, to open **VBScript Editor**.
3. In the **VBScript Editor**, simply type the following script:

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

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

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

Resetting Page Counter

The C1FlexReport.Page variable is created and automatically updated by the control. It is useful for adding page numbers to page headers or footers. In some cases, you may want to reset the page counter when a group starts. For example, in a report that groups records by country. You can do this by adding code or using the Designer.

Using Code:

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

Visual Basic

```
C1FlexReport1.Fields("PageFooter").Text = "[ShipCountry] & "" "" &
[Page]"
```

C#

```
c1FlexReport1.Fields["PageFooter"].Text = "[ShipCountry] + [Page]";
```

Using FlexReportDesigner:

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

1. Select the PageFooter's page number field from the Properties window drop-down list in the Designer or select the field from the design pane. This reveals the field's available properties.
2. Click the box next to the **Text** property, then click the drop-down arrow, and select **Expression Editor** from the list. **VBScript Editor** windows appears.
3. In the **VBScript Editor**, type the following script:
   `="Page " & GroupPage(0) & " of " & GroupPages(0) & " for " & Country`
4. Click **OK** to close the editor.

Adding Sub-sections

Sub-sections are the additional sections that can be added to any section of a report. A FlexReport generally contains - Detail, Header, Footer, PageHeader, Page Footer, Group Header and Group Footer - sections as described in Sections of FlexReport.

Each of these sections contains atleast one sub-section, but you can add as many sub-sections in a section.

To add a sub-section in the Header section of a report, the following code should be used:

Visual Studio

```
 'create a subsection in the header section
    Dim ss As SubSection =
C1FlexReport1.Sections.Header.SubSections.Add()
  'set height to 10 mm
    ss.Height = 10 * 1440 / 25.4
```

C#

```
//create a subsection in the header section
   SubSection ss = rep.Sections.Header.SubSections.Add();
// set height to 10 mm
   ss.Height = 10 * 1440 / 25.4;
```

Working with FlexReportDesigner

FlexReportDesigner is a stand-alone application for designing FlexReport, similar to the report designer in Microsoft Access. The default location of the designer is C:\Program Files (x86)\ComponentOne\Apps\v4.0\. There are two designer applications:

- **C1FlexReportDesigner.4.exe** targets 'Any CPU' so the application runs in 64 bit mode on 64 bit systems and in 32 bit mode on 32 bit systems. This application does not support use of 32 bit only data providers such as Microsoft.Jet.OLEDB.4.0.
- **C1FlexReportDesigner32.4.exe** targets x86 that allows using 32 bit only data providers such as Microsoft.Jet.OLEDB.4.0.

You can create a basic report definition file, modify, print, and export the report definition. The following topics explain all about the FlexReportDesigner application.

About FlexReportDesigner

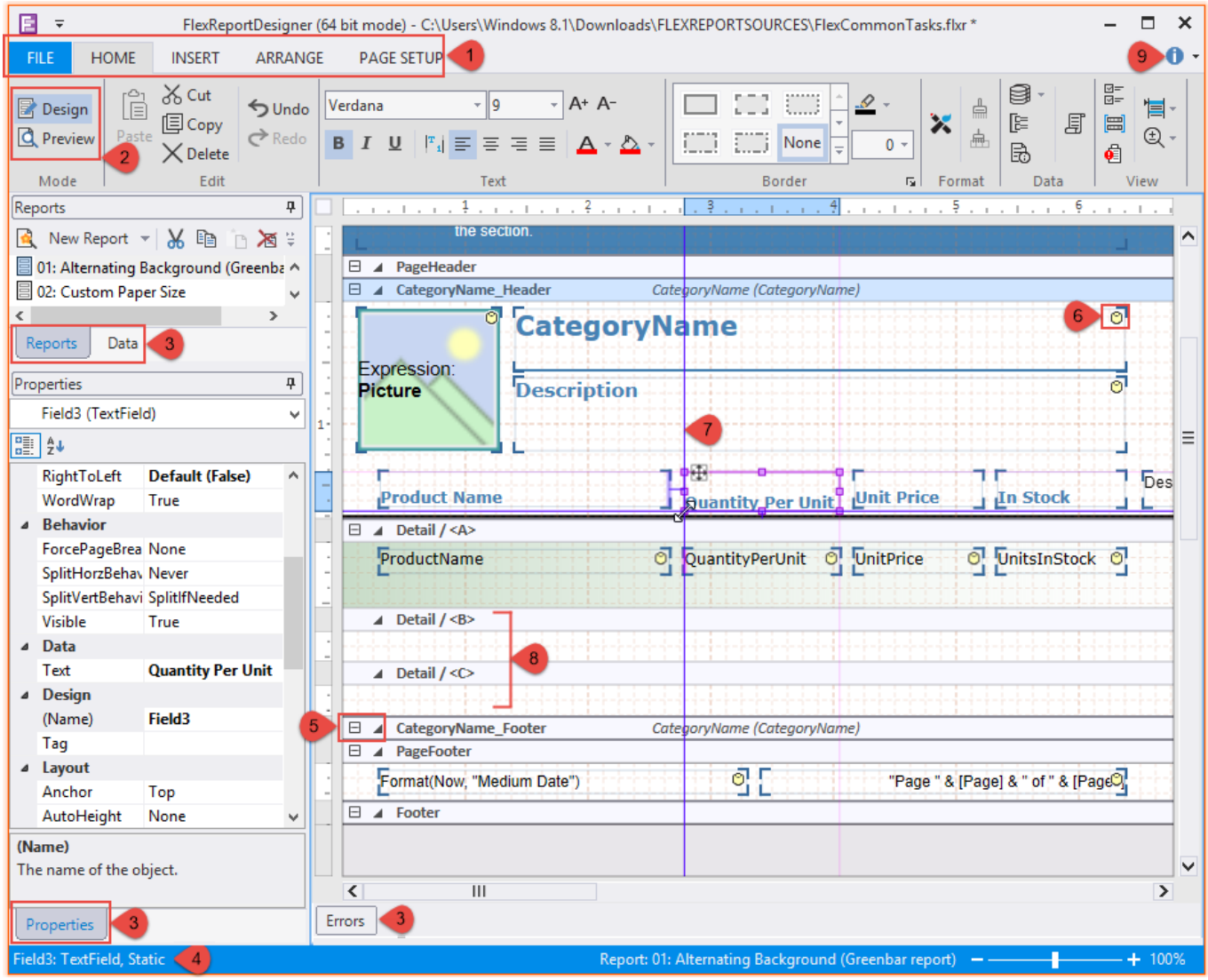
The **FlexReportDesigner** application is a tool used for creating and editing **C1FlexReport** report definition files. The Designer allows you to create, edit, load, and save files (FLXR) that can be read by the C1Report component. It also allows you to import report definitions from Microsoft Access files (.mdb) and Crystal Reports (.rpt).

To run the Designer, double-click **C1FlexReportDesigner.exe** for 64 bit platform and **C1FlexReportDesigner32.4.exe** for 32 bit platform located at the following location on your computer:

C:\Program Files (x86)\ComponentOne\Apps\v4.0

Note that this directory reflects the default installation path and its path may be different if you made changes to the installation path.

Here's what the Designer looks like with the FlexCommonTasks.flxr file opened:



The main Designer window has the following components:

1	• **File menu:** Contains information for handling report definition files - load, save, import and export. • **Tabs - Home, Insert, Arrange, PageSetup** - provide all functionalities related to a report definition - editing, formatting, inserting fields and sections, positioning and sizing, and page layout and printer settings.
2	• **Design mode**: Provides shortcuts to the Edit, Text, Data, etc. menu functions. By default, Design Mode is selected which consists of Home, Insert, Arrange, and Page Setup Tabs. • **Preview mode**: Provides a preview of the report. See Preview Mode for more information.
3	• **Reports tab**: Lists all reports contained in the current report definition file. You can double-click a report name to preview or edit the report. You can also use the list to rename, copy, and delete reports. • **Data tab**: Lists all the Data Sources and Parameters in the current report. The data sources and parameters can be added or edited from here. • **Properties tab**: Allows you to edit properties for the objects that are selected in the Designer. • **Error tab**: Displays list of errors, their severity and count, generated when importing or previewing a report.

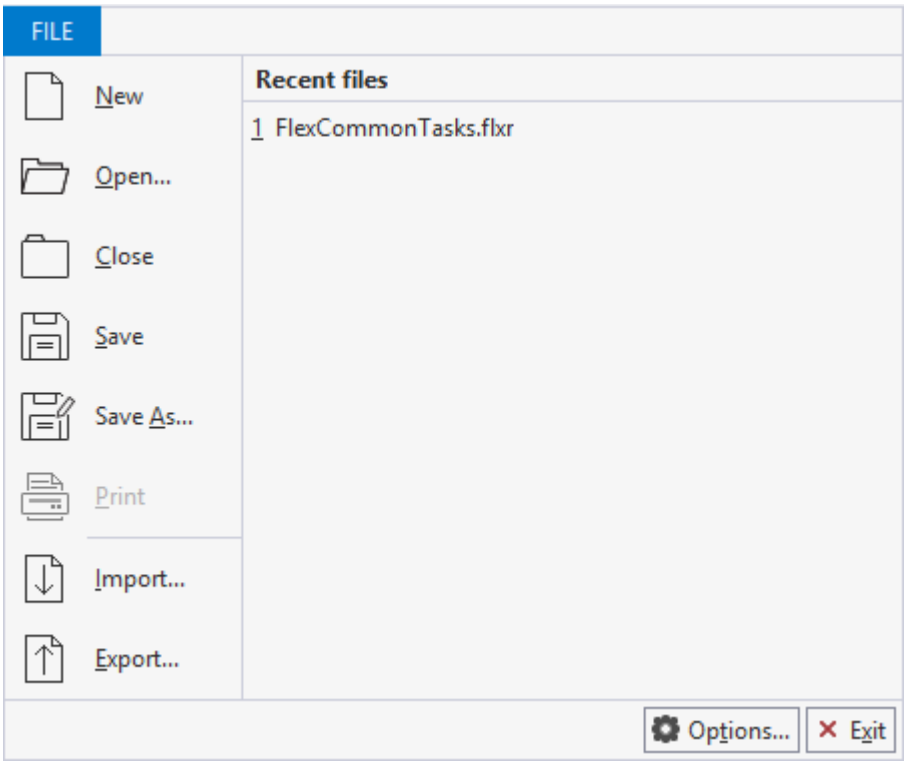
4	• **Status bar:** Displays information about what the Designer is working on. If a field is selected, status bar displays selected field's name, type, and if the field is data bound (calculated) or static. If a section is selected, status bar displays the name of the section, section type, and its visibility if the section is hidden. It also displays processes such as loading, saving, printing, rendering, importing, and so on. You can zoom in and out of a selected report by dragging the zoom slider at the right of the status bar.
5	• **Collapse/Expand Glyphs:** Each section provides option to expand or collapse the sub-sections contained within them by clicking expand (⊞) or collapse (⊟)glyphs.
6	• **Database icon:** Indicates that a field is bound to a data source (i.e., a calculated field), if the data base icon appears on the top-right corner of the field. If the data base icon does not appear, it means the field is static.
7	• **Snap Lines:** Help in the alignment of the fields. When the size of a field is increased or decreased, vertical and horizontal snap lines appear, that help in positioning the fields relative to each other.
8	• **Sub-sections:** Sub-sections are sections within sections. Sub-sections, by default, appear at the bottom of a section and are automatically named as /<B>, /<C>..., and so on.
9	• **Help button:** Provides options to open the online help file and view the **About** screen, which displays information about the application.

The topics that follow explain how you can use the **C1FlexReportDesigner** application to create, edit, use, and save report definition files.

File Menu

The **File** menu provides shortcut to load and save report definition files and to import and export report definitions. You can also access the **C1FlexReportDesigner** application's options through the **File** menu.

The following image displays the **File** menu:

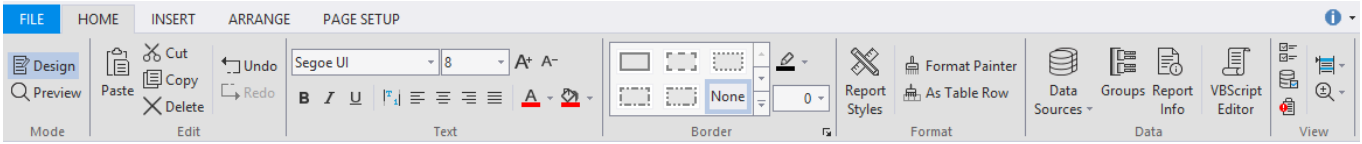


The menu includes the following options:

- **New**: Creates a new report definition file.
- **Open**: Brings up the **Open Report Definition File** dialog box, enabling you to select an existing file to open.
- **Close**: Closes the current report definition file.
- **Save**: Saves the report definition file, to the location previously saved.
- **Save As**: Opens the **Save Report Definition** dialog box allowing you to save your report definition as an .flxr file.
- **Print**: Prints the current report. Note that **Print** button is enabled only in preview mode of **C1FlexReportDesigner** application.
- **Import**: Opens the **Import Report Definition** dialog box enabling you to import Microsoft Access (.mdb and .adp) files and Crystal Reports (.rpt) files. See Importing Microsoft Access Reports and Importing Crystal Reports for more information.
- **Export**: Exports the current report file as an HTML, PDF /A, PDF, RTF, DOCX, XLS, XLSX, TIFF, BMP, PNG, JPG, ZIP, or GIF. See Exporting and Publishing a Report for more information.
- **Recent files**: Lists recently opened report definition files. To reopen a file, select it from the list.
- **Options**: Opens the **C1FlexReportDesigner Options** dialog box which allows you to customize the default appearance and behavior of the **C1FlexReportDesigner** application. See Setting C1FlexReportDesigner Options for more information.
- **Exit**: Closes the **C1FlexReportDesigner** application.

Design Mode

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

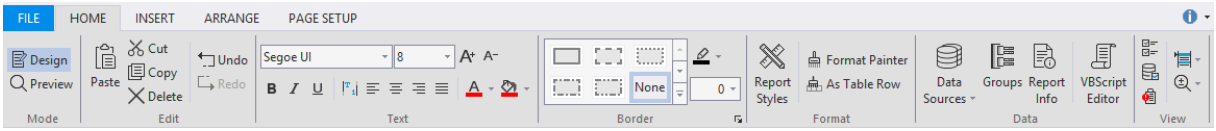


- **Home tab**: Provides shortcuts to the Edit, Text, Border, Format, Data, and View menu functions. See Home Tab

for more information.

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

Home Tab



Home tab consists of several menu functions arranged in following groups:

Edit group: It consists of the following options:

- **Paste**: Pastes the last copied item.
- **Cut**: Cuts the selected item, removing it from the report and allowing it to be pasted elsewhere.
- **Copy**: Copies the selected item so that it can be pasted elsewhere.
- **Delete**: Deletes the selected item.
- **Undo**: Undoes the last change that was made to the report definition.
- **Redo**: Redoes the last change that was made to the report definition.

Text group: It consists of the following options:

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

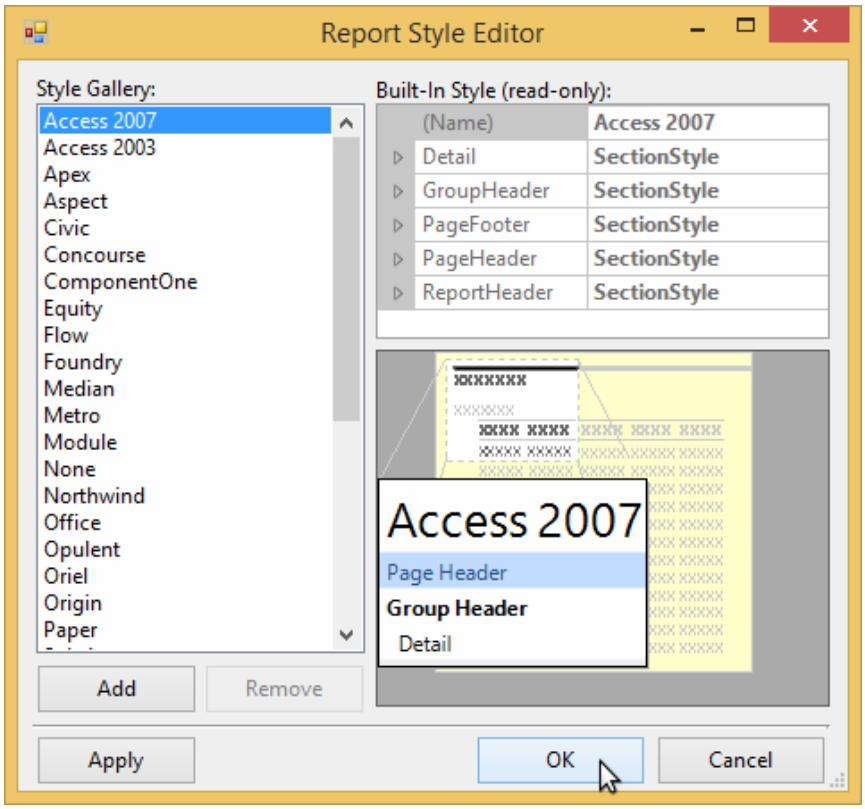
Border group: It consists of the following options:

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

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

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

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

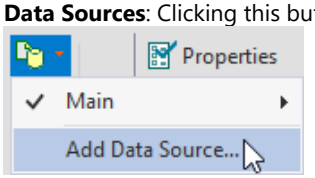


It consists of following elements:

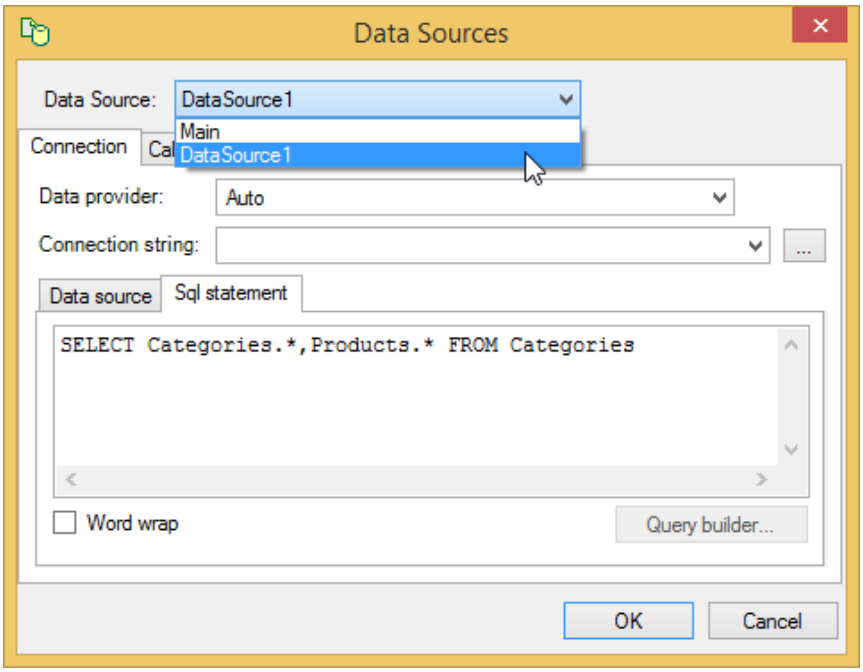
- **Style Gallery List**: Displays all the currently available built-in and custom styles. See Style Gallery for information about the available built-in styles.
- **Add button**: Adds a custom style to the Style Gallery list. The style that is added is based on the style selected in the Style Gallery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

Data group: It consists of the following options:

- **Data Sources**: Clicking this button lists down the options Main and Add Data Source.

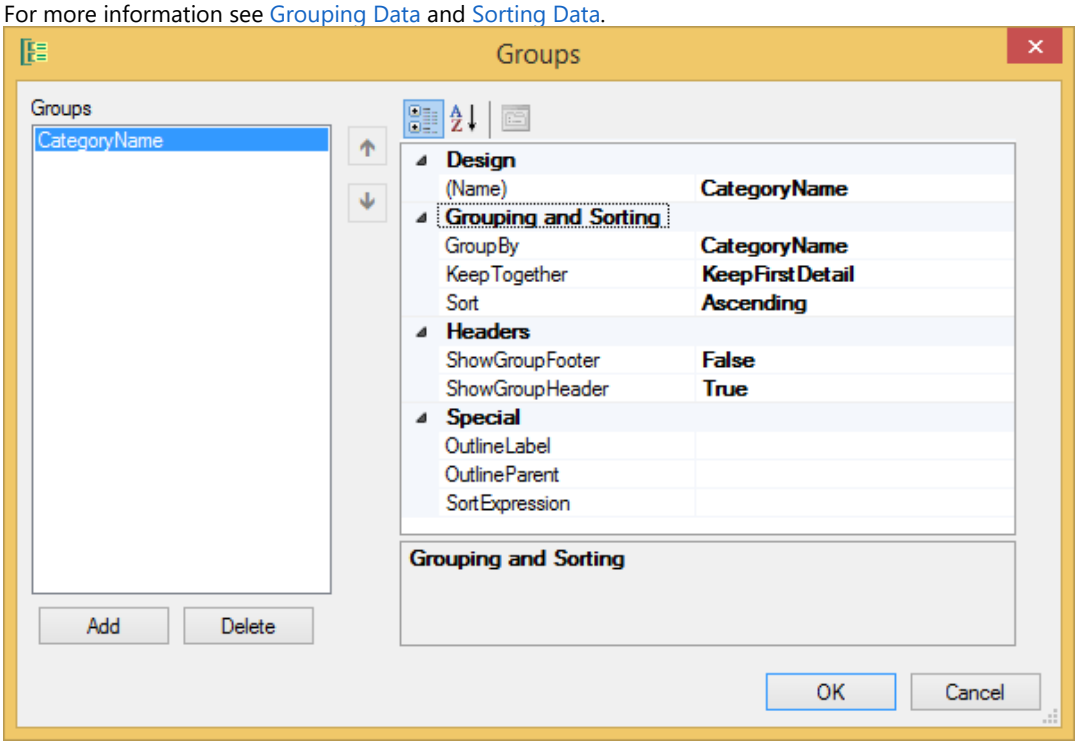


Main option lets you Edit or Remove the main data source to which the report is bound. Clicking **Edit** or **Add Data Source** opens the **Data Sources** dialog box. You can then you to choose a new data source, change the connection string, and edit the Sql statement. Clicking the drop down next to **Data Source** option displays the list of data sources present in the report. From the **Data source** tab, you can select the tables, views, and stored procedures in the current data source. Clicking the **Sql statement** tab allows you to view the current SQL statement:

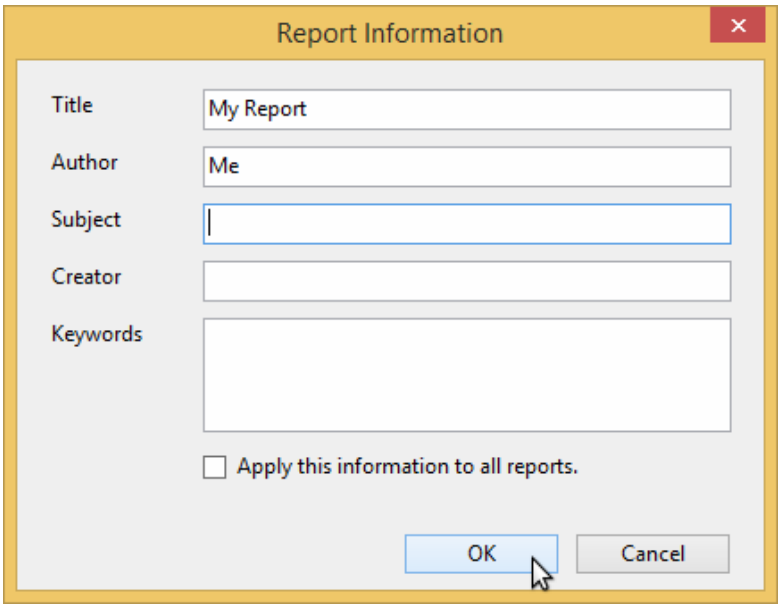


To change the connection string, click the ellipses button. This will open the **Data Link Properties** dialog box. To edit or change the SQL statement, click the **Query builder...** button which will open the **Sql Builder** dialog box.

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



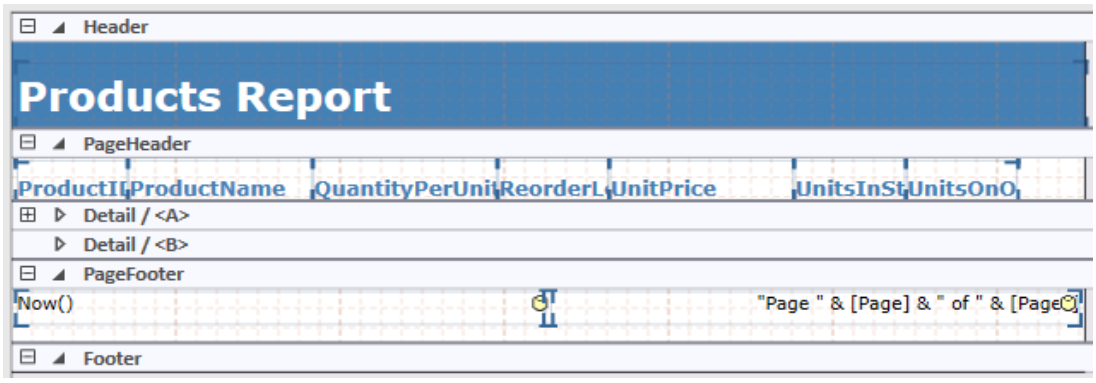
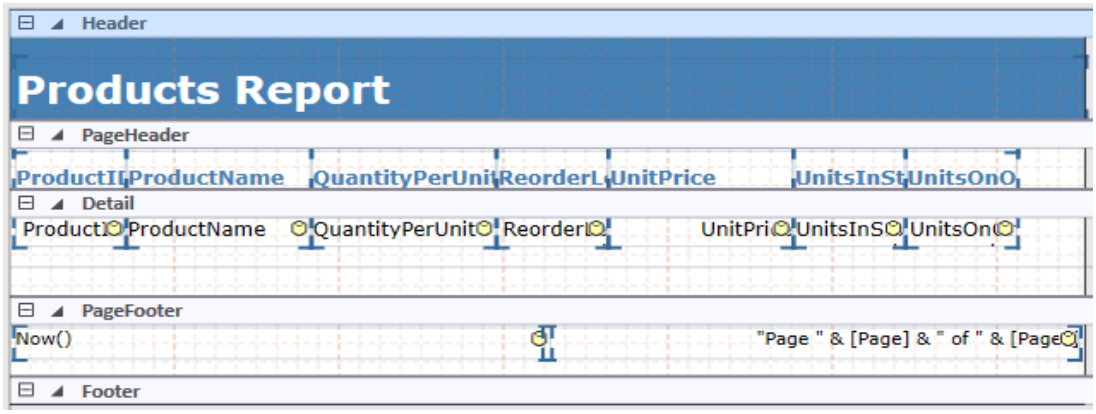
- **Report Info** : Opens the **Report Information** dialog box. This dialog box allows you to set the report's Title, Author, Subject, Creator, and Keywords. You can also choose to apply report information to all reports.



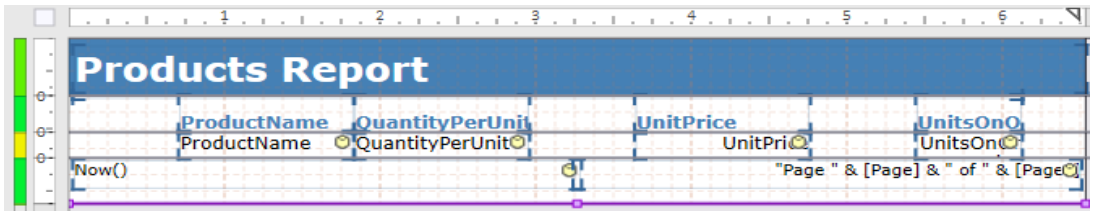
- **VBScript Editor**: Opens **VBScript Editor - Report.GlobalScripts** dialog box. Multiple scripts can be easily edited in the **VBScript Editor**, allowing users to switch between statements and expressions.

View group: It consists of the following options:

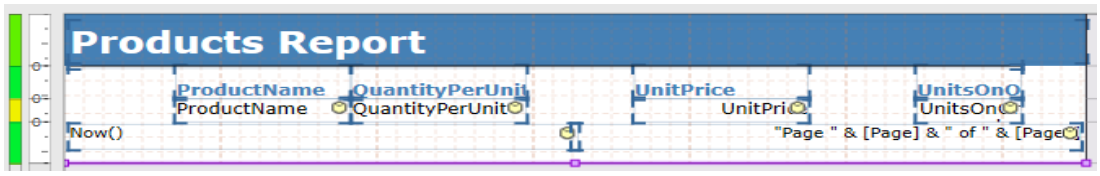
- **Properties**: Brings the **Properties** tab into view on the left pane. The shortcut key for viewing the **Properties** tab is F4.
- **Data**: Brings the **Data** tab into view on the left pane. The shortcut key combination for viewing the **Data** tab is Shift+F4.
- **Error List:** Displays the list of warnings and errors generated while importing or previewing a report.
- **Captions**: Lets you make a choice of viewing captions for sections or sub-sections in the designer panel. You can choose options - **All** (shows strip on all), **Section** (shows strip on Sections and not subsections), **Hairline** (hides the section header strips), **Hidden, Expand All** and **Collapse All** options. The drop-down consists of the following options:

○ All	
○ Section	

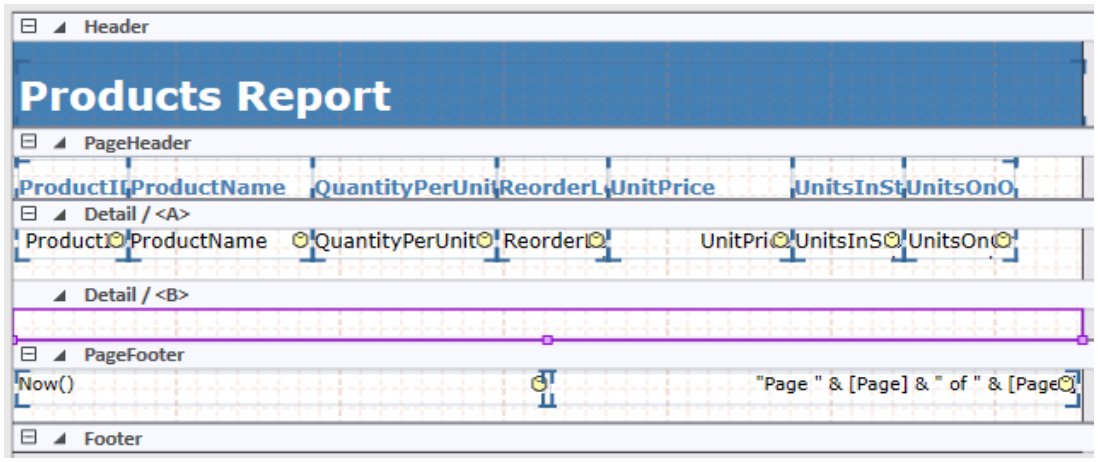
- Hairline



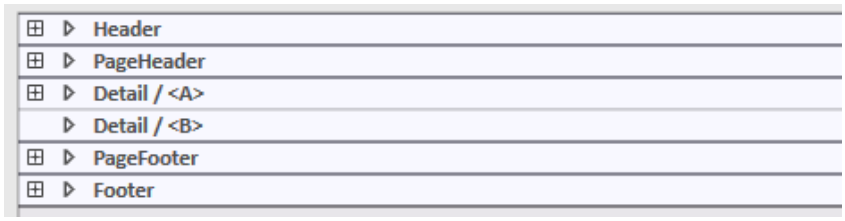
- Hidden



- Expand All

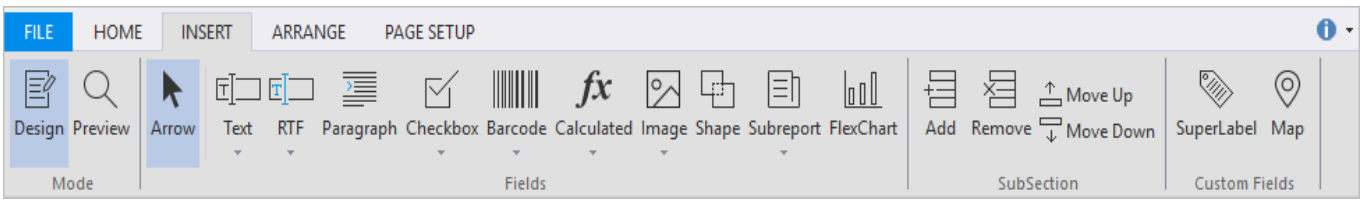


- Collapse All



- **Zoom**: Allows you to select a value to set the zoom level of the report. You can also press Ctrl+Plus or Ctrl+Minus to zoom in or zoom out the designer panel.

Insert Tab



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

Fields group: It consists of the following items:

- **Arrow**: Returns the cross mouse cursor to an arrow cursor.
- **Text**: Creates a field bound to the source recordset or an unbound (static) text label. When you click this button, a list appears and you can select the recordset field. Bound fields are not limited to displaying raw data from the database. You can edit their **Text** property and use any VBScript expression.

- **RTF**: Creates an RTF field. When you click this button, a list appears where you can select other fields that are contained in the same report definition file to be displayed in RTF format.
- **Paragraph:** Creates an Paragraph field. For more information, see Paragraph Field.
- **Checkbox**: Creates a bound field that displays a Boolean value as a check box. By default, the check box displays a regular check mark. You can change it into a radio button or cross mark by changing the value of the field's **CheckMark.Style** property after it has been created.
- **Barcode**: Creates a field that displays a barcode. When you click this button, a menu appears where you can select other fields that are contained in the same report definition file to be displayed as a barcode. See Barcode Field for more information.
- **Calculated**: Creates a calculated field. When you click this button, the code editor dialog box appears so you can enter the VBScript expression or an arbitrary formula whose value you want to evaluate. When you click the drop down, you can select commonly used expressions that render the date or time when the report was created or printed, the page number, page count, or "page n of m", or the report name.
- **Image**: Creates a field for data bound stored in the recordset image or static (unbound) image. When you click this button, 'Open' dialog box appears that lets you choose an image that is static (unbound), such as a logo. When you click the drop-down, you can select an image field in the source recordset (if there is one; not all recordsets contain this type of field).
- **Shape**: Creates a geometric shape - Line, Isosceles Triangle, Right Triangle, Rectangle, Ellipse, and Arc. These shapes can be used to enhance the look of a report.
- **Subreport**: Creates a field that displays another report. When you click this button, a list appears and you can select other reports that are contained in the same report definition file.
- **FlexChart**: Creates a field that displays a FlexChart. For more information, see FlexChart Field.
- **Chart**: Creates a field that displays a chart. See Chart Field for more information.

> **Note:** By default, Chart Field is hidden in the Insert tab and can be used as a Legacy Chart Field.

Subsection group: It consists of the following items:

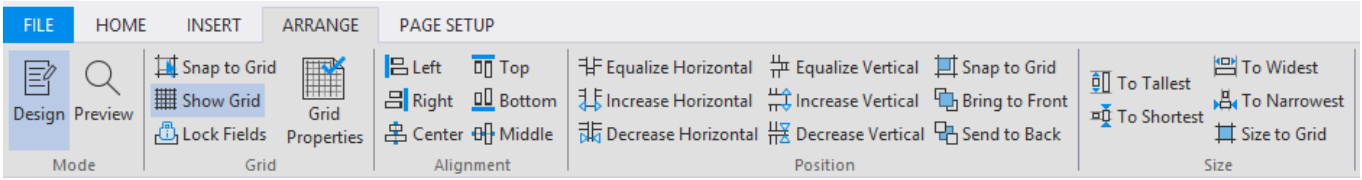
- **Add**: Adds a sub-section at the bottom of the current section.
- **Remove**: Removes the currently selected sub-section.
- **Move Up**: Moves the currently selected sub-section upward, jumping one sub-section at a time.
- **Move Down**: Moves the currently selected sub-section downward, jumping one sub-section at a time.

Custom Fields group: It consists of the following items:

- **SuperLabel**: Creates a field that renders HTML formatted text. The text property of the field is set to any HTML text that is required to be rendered.
- **Map**: Creates a field that displays a region of earth, i.e., a map. See Maps in Reports for more information.

For more information on adding fields to your report, see Adding FlexReport Fields and Adding FlexReport Custom Fields.

Arrange Tab



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

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

- **Snap to Grid**: Snaps fields to the grid. When this item is selected fields cannot be placed in between lines of the grid.

- **Show Grid**: Shows a grid in the background of the report in the preview. The grid can help you place and align fields. By default, this option is selected.
- **Lock Fields**: Locks and unlocks the fields in the report. After you've placed the fields in the desired positions, you can lock them to prevent inadvertent moving of fields with mouse or keyboard.
- **Grid Properties**: Opens the **FlexReportDesigner Options** dialog box. For details see Setting C1FlexReportDesigner Options.

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

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

Note that the elements in a report can be both horizontally and vertically aligned - so, for example, an element can be both left and top aligned.

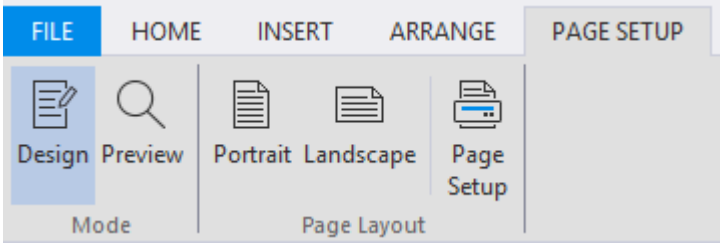
Position group: The **Position** group controls spacing between elements and how elements are layered. It consists of the following items:

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

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

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

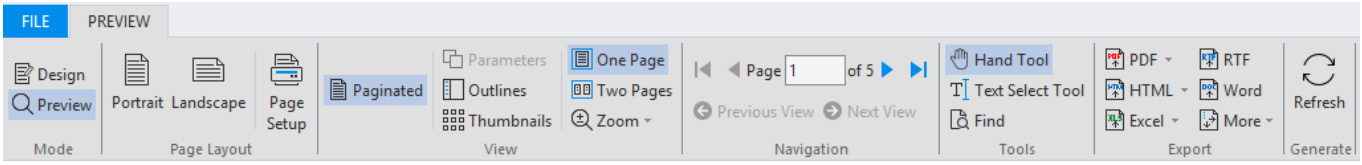
Page Setup Tab



The **Page Setup** tab provides shortcuts to Page Layout menu functions. It consists of the following options:

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

Preview Mode



The **Preview** mode displays preview of the current report. The ribbon on the **Preview** mode consists of the following items:

Page Layout group: It consists of the following options:

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

View group: It consists of the following options:

- **Paginated:** Switch between paginated and non-paginated views of the report.
- **Parameters:** Show or hide report parameter panel. It is enabled only if the report contains parameters.
- **Outlines:** Displays a text outline of the document.
- **Thumbnails:** Switch between normal and thunbnail view.
- **One page**: Allows you to preview one page at a time.
- **Two pages**: Allows you to preview two pages at a time.
- **Zoom**: Zooms the page in to a specific percent or to fit in the window. The shortcut key for zooming in is Ctrl+Plus and for zooming out is 'Ctrl+Minus'.

Navigation group: It consists of the following options:

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

Tools group: It consists of the following options:

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

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

- **PDF**: Exports the report to a PDF file. The drop-down arrow includes options for PDF with non-embedded (linked) fonts and **PDF /A (embedded fonts)** to choose if you want to use system fronts or embed your chosen fonts in the PDF file.
- **HTML**: Exports the report to an HTML file. You can then copy and paste this text to another application. The drop-down arrow includes options for **Plain HTML**, **Paged HTML**, and **Drilldown HTML**, and **Table-based HTML** allowing you to choose if you want to export to a plain HTML file, multiple HTML files that can be paged using included arrow links.
- **Excel**: Exports the report to a Microsoft Excel file. The drop-down arrow includes options for **Microsoft Excel**

97 and **Microsoft Excel 2007 - OpenXML** allowing you to choose if you want to save the document as an XLS or XLSX file.

- **RTF**: Exports the report to a Rich Text File (RTF).
- **Word:** Exports the report to Open XML Word (DOCX) format.
- **More**: Clicking the **More** drop-down arrow includes additional options to export the report including: Tagged Image File Format (export as TIFF), Compressed Metafiles (export as ZIP), BMP (Bitmap Images), PNG(Portable Network Graphic), JPEG or GIF.

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

Generate:

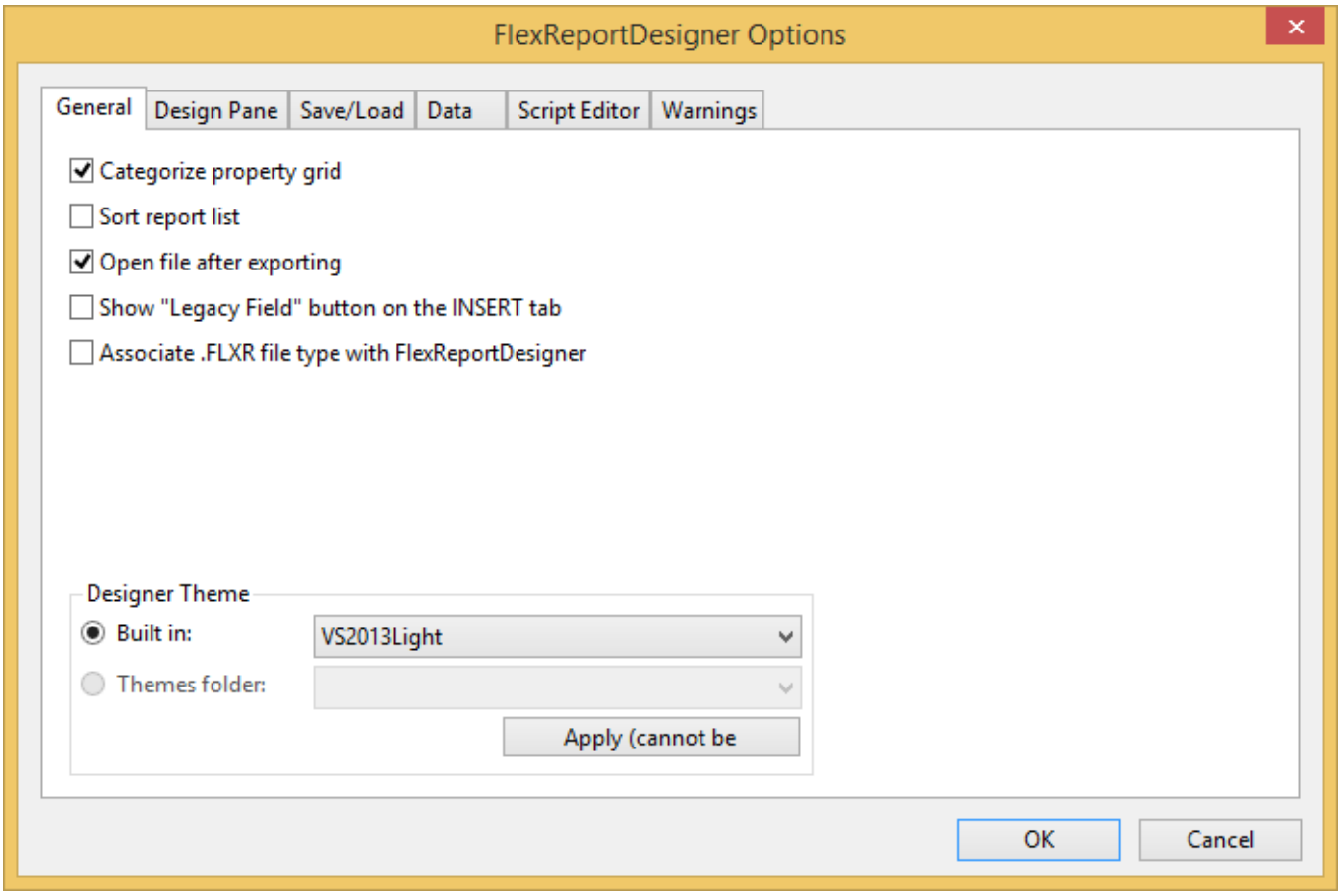
It consists of **Refresh** button. Clicking **Refresh** regenerates the current report. This button changes to **Stop** while the report is regenerating; so you can also stop regenerating the report.

Setting FlexReportDesigner Options

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

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

General tab:

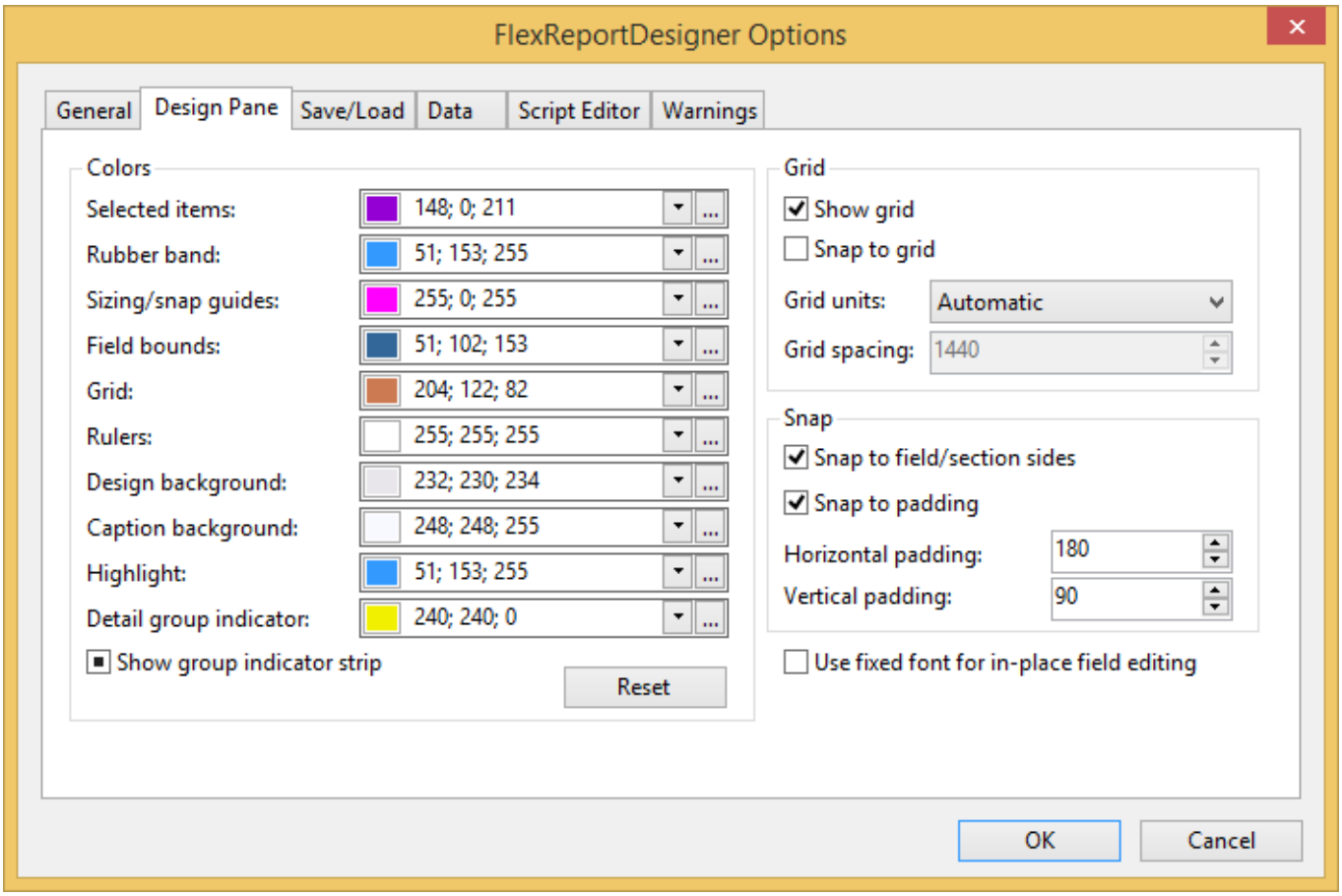


It consists of the following options:

- **Categorize property grid**: Categorizes the Properties grid by property type. The Properties grid can be accessed by clicking the **Properties** tab located in the bottom of the left pane in Design view.
- **Enable undo/redo**: Enables undo and redo in the application.

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

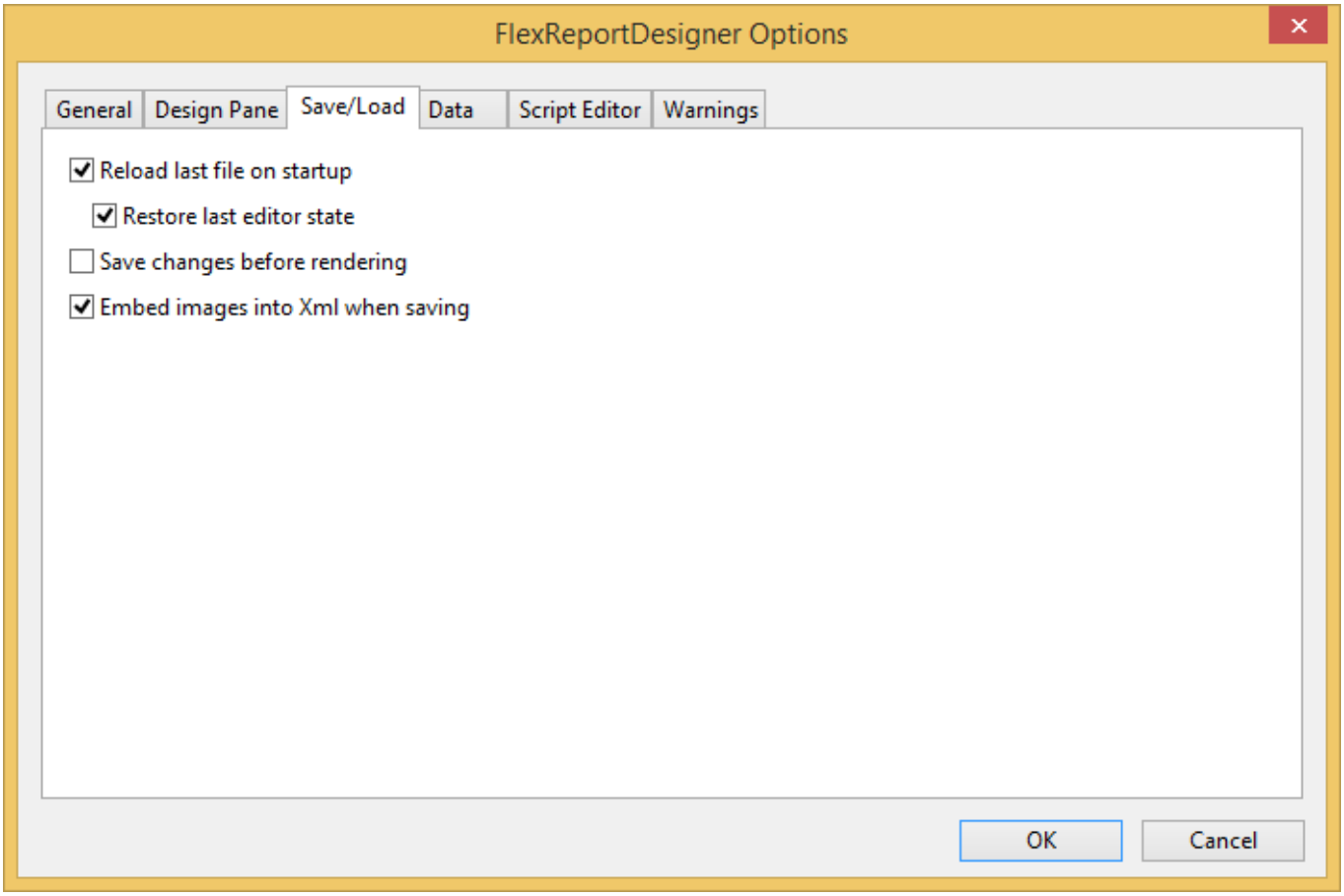
Design Pane tab:



It consists of the following options:

- **Show grid**: Shows the grid in the report preview window.
- **Snap to grid**: Snaps all objects the grid in the report. If this option is selected, you will not be able to place objects between grid lines.
- **Show subreport content**: Shows sub-report content in the report.
- **Use C1FlexReport.CreationGraphics as reference graphics to render fields**: Uses **C1FlexReport.CreationGraphics** as reference to render fields.
- **Grid units**: Indicates how the grid is spaced. Options include **Automatic**, **English (in)**, **Metric (cm)**, and **Custom**.
- **Grid spacing**: Sets the spacing of grid lines. This option is only available when the **Grid Units** option is set to **Custom**.
- **Grid major color**: Set the color of major grid lines.
- **Grid minor color**: Sets the color of minor grid lines.
- **Field edges color**: Sets the color of field edges in the report.

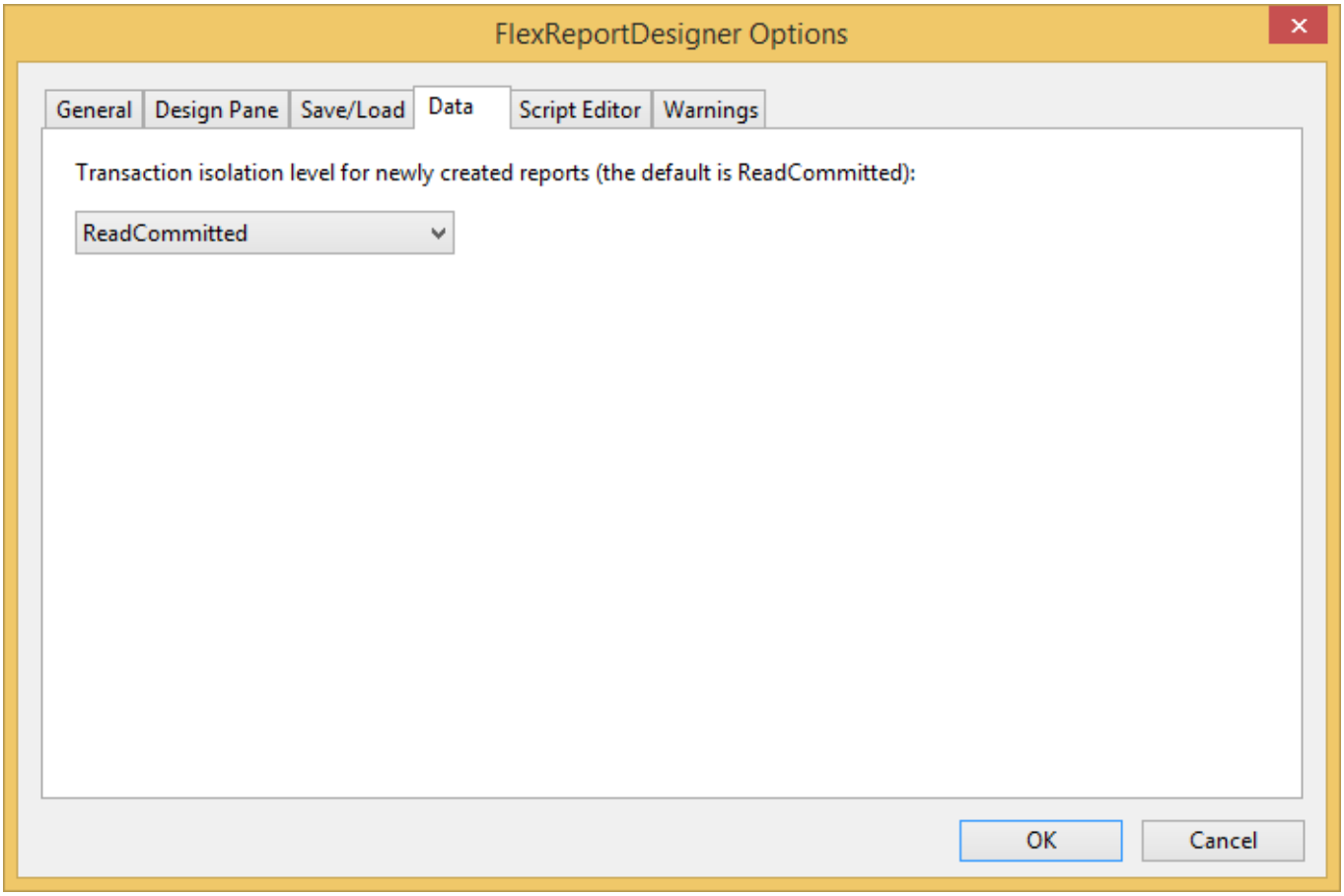
Save/Load tab:



It consists of the following options:

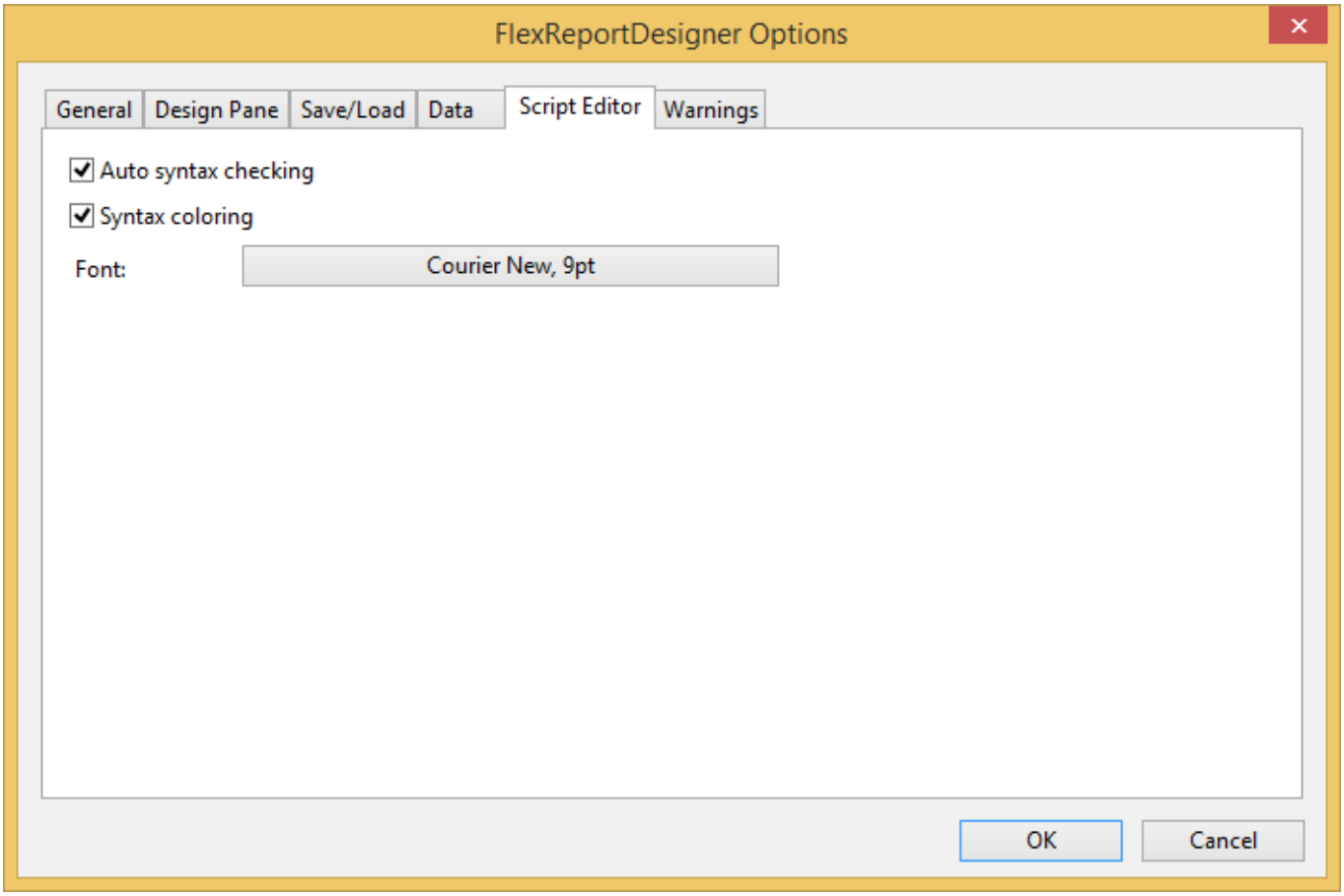
- **Reload last file on startup**: If this option is checked, the last opened file will appear whenever the **C1FlexReportDesigner** application is opened.
- **Save changes before rendering**: Checking this option saves the report before rendering.
- **Show options when exporting**: Checking this option saves the report's options when exporting.
- **Embed images into Xml when saving**: Checking this option embeds images into XML when the report is saved.
- **Default export format**: Sets the default export format. For more information about exporting see Exporting and Publishing a Report.

Connection tab:



It consists of the options for transaction isolation level.

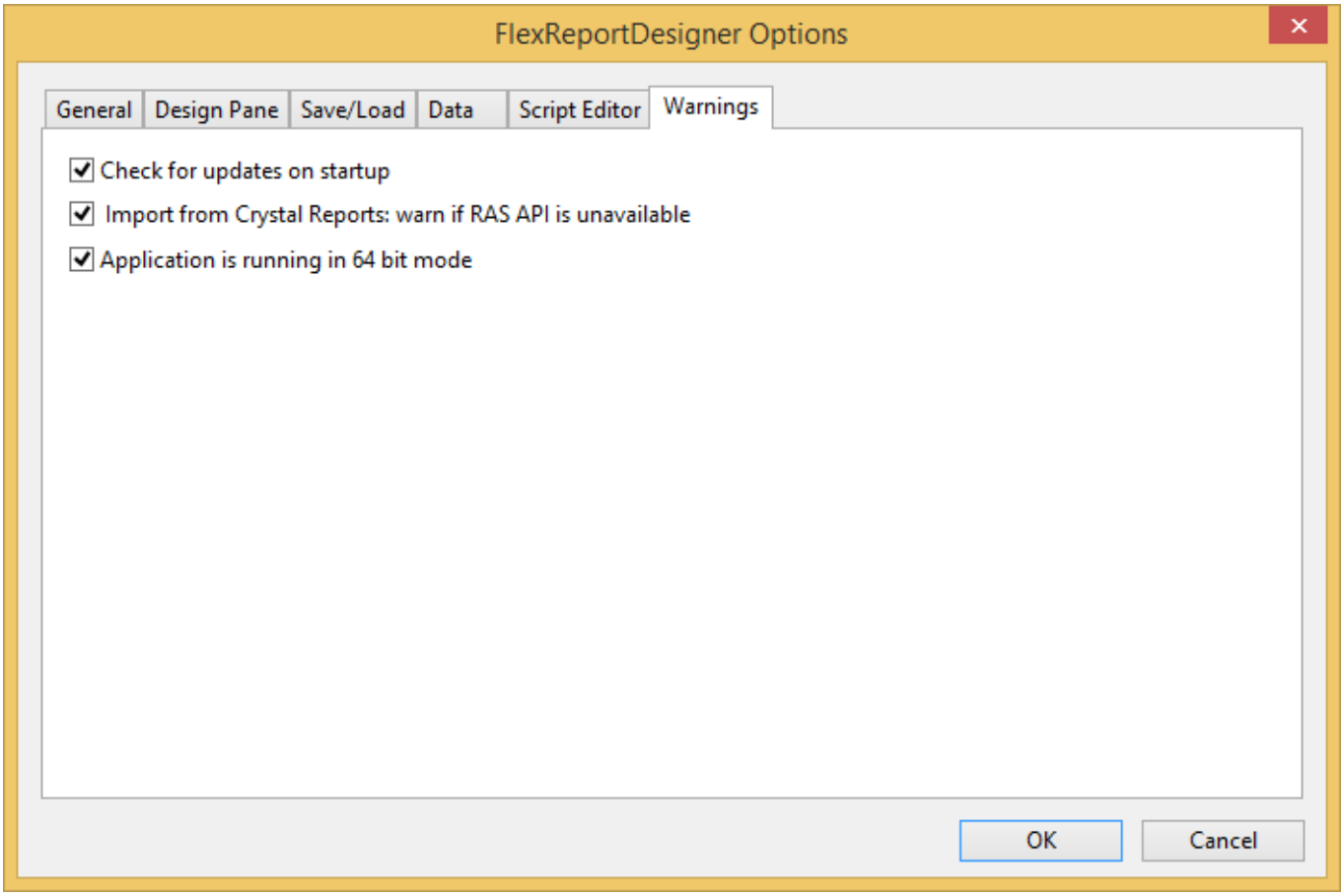
Script Editor:



It consists of the following options:

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

Warnings tab:



It consists of the following options:

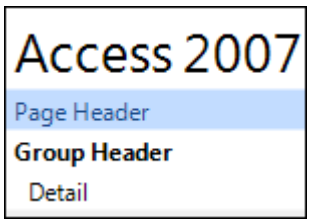
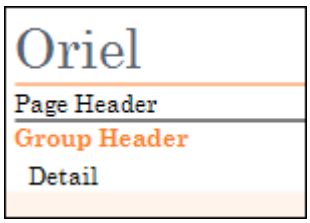
- **Check for updates on startup**: Checking this option checks for any updates when C1FlexReportDesigner application is opened.
- **Import from Crystal Reports: warn if RAS API is unavailable**: Checking this option throws warning if RAS API is unavailable while importing Crystal Reports in **C1FlexReportDesigner**.

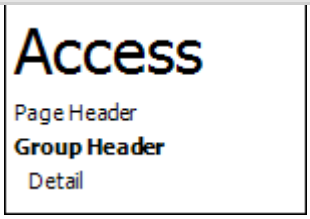
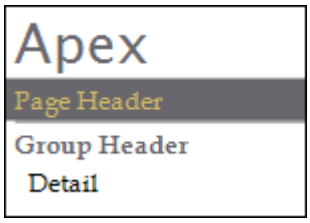
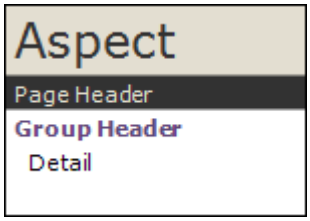
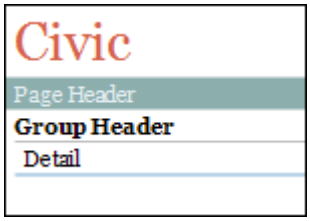
In each of the above tabs, clicking OK saves the changes and clicking Cancel cancels any changes that you have made in the **FlexReportDesigner Options** dialog box.

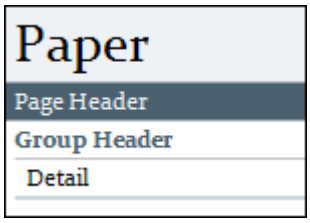
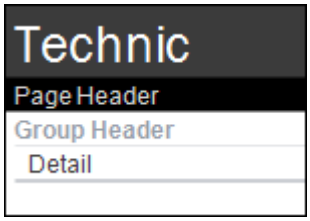
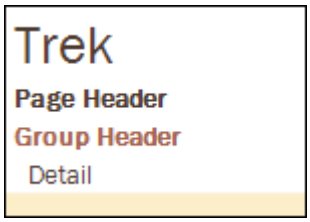
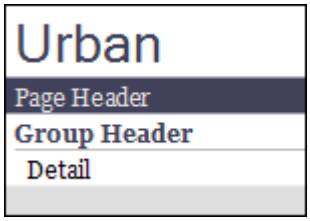
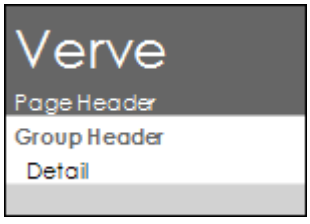
Style Gallery

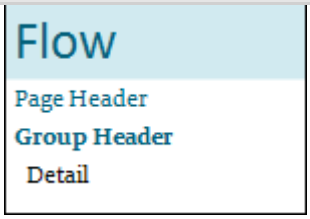
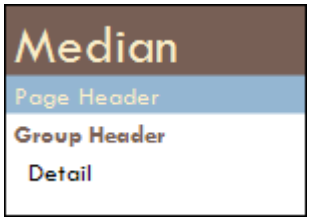
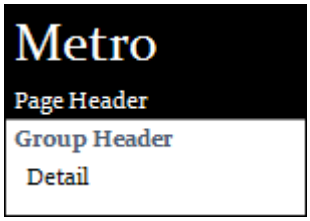
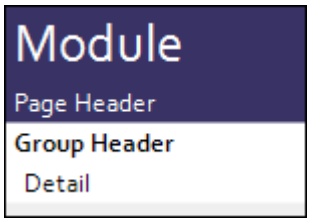
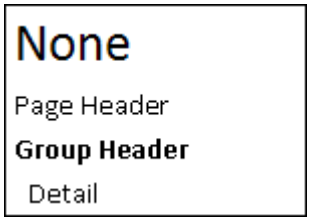
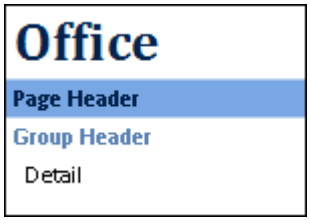
The **Style Gallery** dialog box details all the available built-in and custom styles that you can use to format your report. Built-in styles include standard Microsoft AutoFormat themes, including Vista and Office 2007 themes. You can access the **Style Gallery** from the **C1FlexReportDesigner** application by selecting the **Home** tab and clicking **Report Styles**.

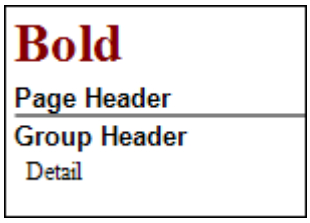
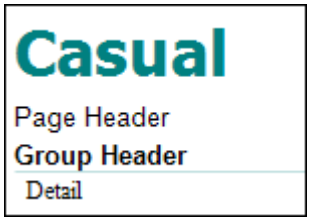
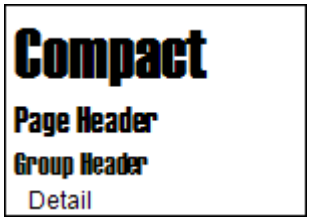
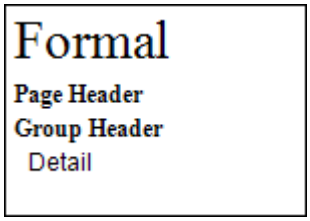
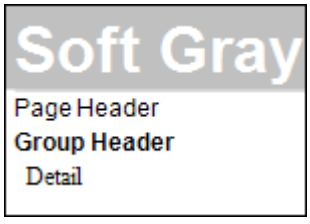
The following built-in styles are included:

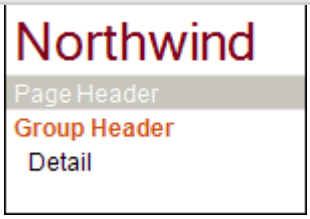
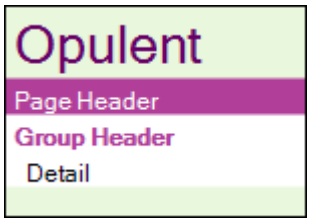
Style Name	Preview	Style Name	Preview
Access 2007		Oriel	

Style Name	Preview
Access 2003	
Apex	
Aspect	
Civic	
Concourse	
ComponentOne	
Equity	

Style Name	Preview
Origin	
Paper	
Solstice	
Technic	
Trek	
Urban	
Verve	

Style Name	Preview
Flow	
Foundry	
Median	
Metro	
Module	
None	
Northwind	

Style Name	Preview
Windows Vista	
Bold	
Casual	
Compact	
Corporate	
Formal	
Soft Gray	

Style Name	Preview	Style Name	Preview
Office		Verdana	
Opulent		WebReport	

Adding Multiple Sub-Sections

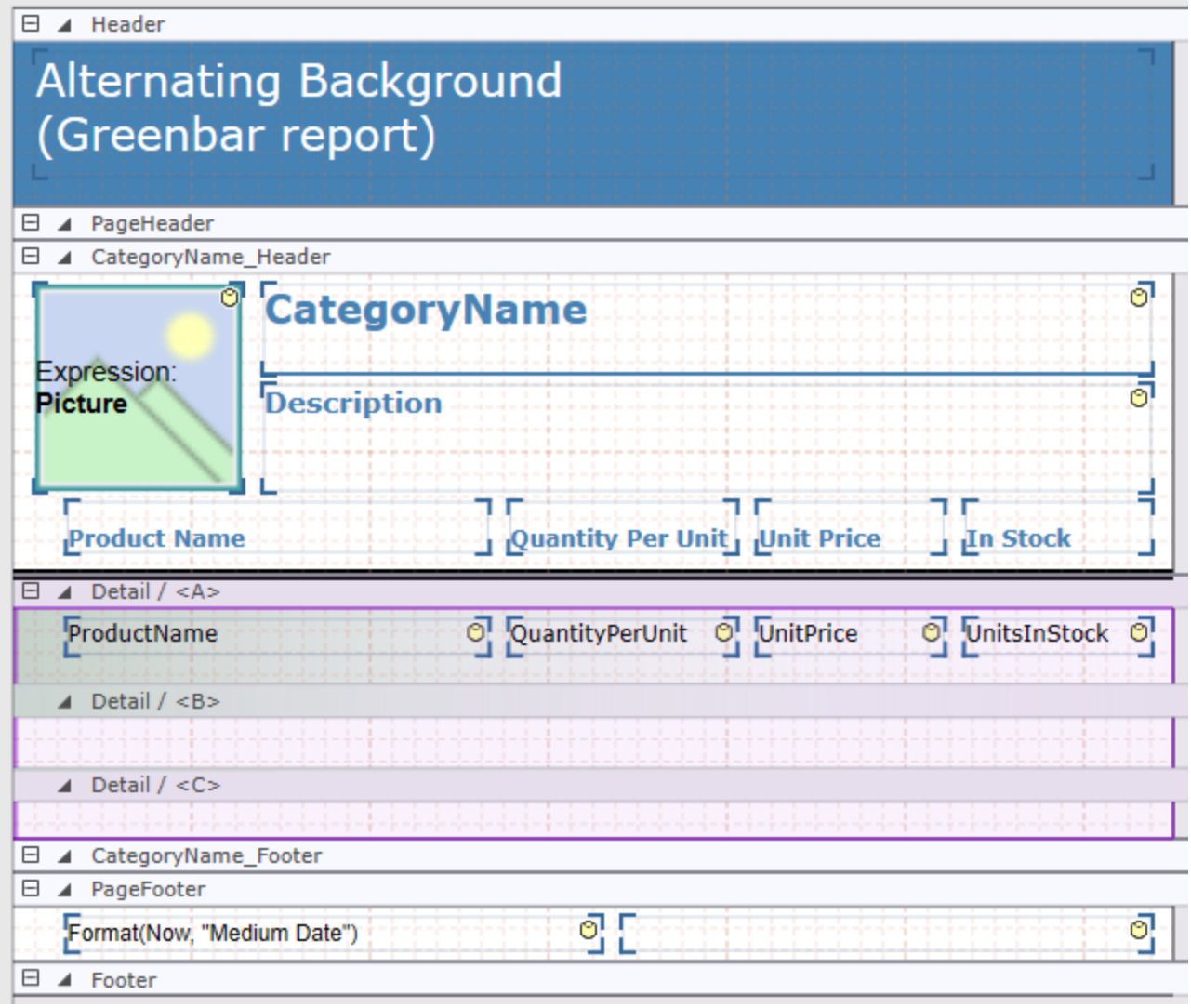
A FlexReport generally contains - Detail, Header, Footer, PageHeader, Page Footer, Group Header and Group Footer - sections. Each of these sections contains atleast one sub-section. You can add as many sub-sections in a section.

Sub-sections are useful in providing additional information about the data present in the parent section. You can add data fields in the same way as other sections. You can also enhance the look of a report by just adding a shape field to a sub-section.

To add Multiple Sub-sections

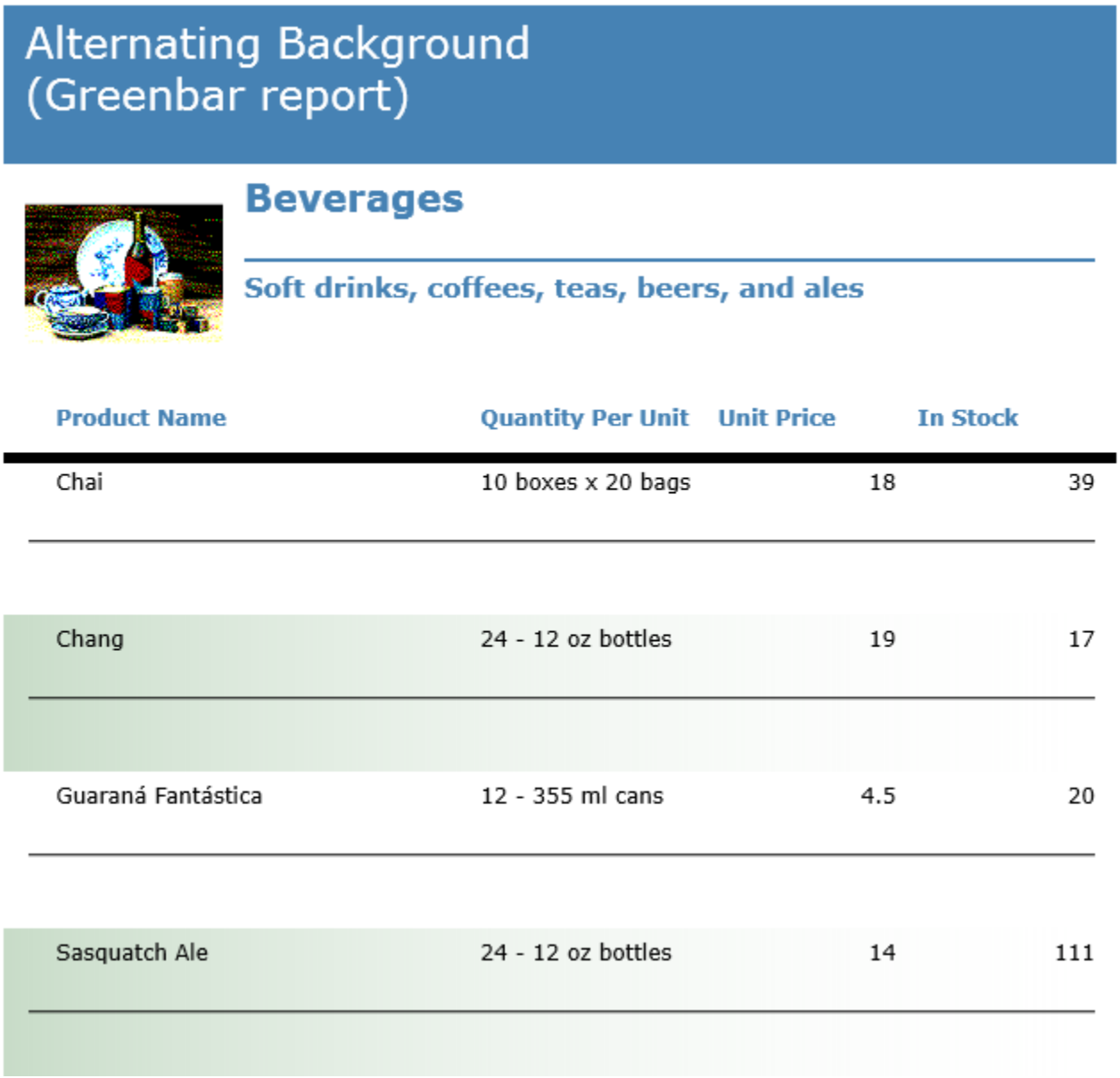
Following steps let you add multiple sub-sections to Detail section of a report:

1. Click the **Detail** section. Observe that it already has a sub-section named **Detail/ <A>**.
2. Go to **Insert** Tab and click **Add** from the **Subsection** group. This adds one sub-section in the Detail section, which is automatically named as **Detail/ <B>**.
3. Again click **Add** from the Subsection group to add one more sub-section. This adds another subsection named as **Detail/ <C>**.



Following steps let you add a field to a sub-section:

4. Go to **Insert** Tab and click **Shape** field from the **Subsection** group.
5. In the sub-section **Detail/ <B>**, drop the **Shape** field.
6. From the Properties window, set **Shape** property to **Line**. Drag the selection handle to increase or decrease length of the line.
7. Preview the report.



You see a line is drawn after every field in the Detail section.

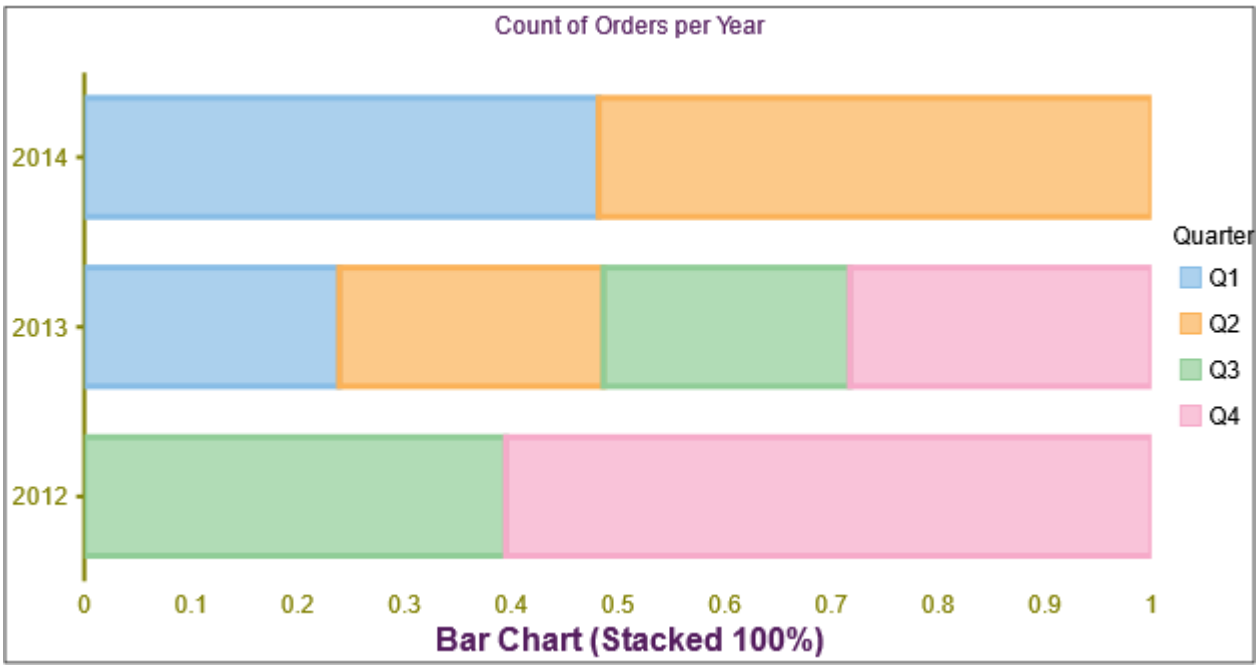
Adding FlexReport Fields

Adding fields in FlexReport is quite simple. All the available fields are provided in the **Insert** tab of C1FlexReportDesigner. You just have to click the field and drop it into the report.

The following sections explain various types of fields, their properties, and how to add them to your reports. Note that the database used for adding fields is C1NWind.mdb.

FlexChart Field

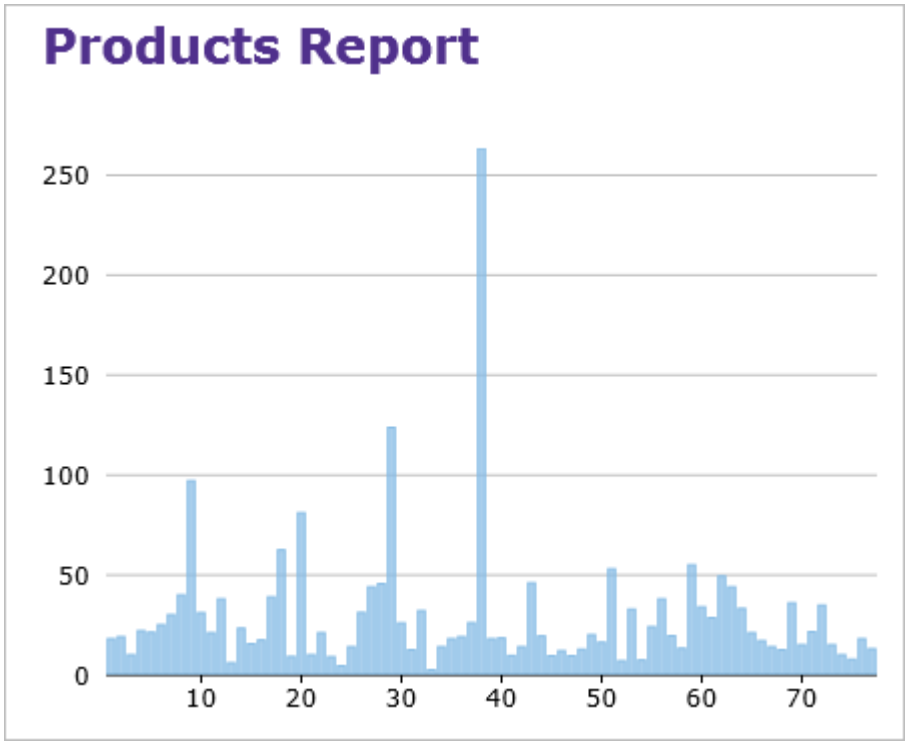
FlexReportDesigner provides a new field type, **FlexChart** field. The **FlexChart** field simplifies adding data visualization capabilities to FlexReport using flexible data binding, multiple chart types, and supports grouping and data aggregation. For more information on FlexChart, see FlexChart documentation.



Binding FlexChart Field with Data

When you add the **FlexChart** field to your report, you need to bind it to data for populating data in the displayed chart, say Column chart, as shown below. The following steps show how to bind the FlexChart field with data in FlexReportDesigner application.

1. Create a new report in C1FlexReportDesigner.
2. In **Data** tab, bind the Main data source with Products table of C1Nwind.mdb database.
3. Under **INSERT** tab, select FlexChart field and draw it in the Header section.
4. Select the FlexChart.
5. In the Properties window, set the desired Chart Type, say Column chart, from ChartType dropdown.
6. Navigate to Series and click the ellipsis button next to it.
7. In the Series Collection Editor, click Add button to add a series data group.
8. Navigate to Data and set the DataSourceName as Main.
9. Set Label as **ProductID**.
10. Set XExpression to **ProductID**.
11. Set YExpression to **UnitPrice**.
12. Close the Editor.
13. Click on Preview.



Difference Between FlexChartField and FlexChart

FlexChart field inherits majority of properties from FlexChart. However, since FlexChart field is used in reporting scenarios, few properties and their nature are different from FlexChart. The major difference between the two controls appears in binding the controls with data as the binding properties are different in FlexChart field from that of FlexChart. The following table shows the high-level object model difference between the FlexChartField (FlexReport) and FlexChart (WinForms).

FlexChartField (FlexReport)	FlexChart (WinForms)
-	AccessibleDescription
-	AccessibleName
-	AccessibleRole
-	AllowDrop
Anchor	Anchor
AutoHeight	-
AutoWidth	AutoWidth
AxisX	AxisX
AxisY	AxisY
Background	BackColor
-	BackgroundImage
-	BackgroundImageLayout
-	Binding

-	BindingMode
-	BindingX
Bookmark	-
Border	-
BodersPlitHorzMode	-
BordersSplitVertMode	-
-	CausesValidation
CategoryGroups	-
ChartType	ChartType
-	ContextMenuStrip
-	Cursor
DataLabel	DataLabel
-	DataMember
-	DataSource
-	Dock
DataSourceName	-
-	Enabled
Font	Font
Footer	Footer
-	ForeColor
ForcePageBreak	-
-	GenerateMember
Header	Header
Height	-
Hyperlink	-
-	ImeMode
Left	-
Legend	Legend
-	LegendToggle
-	Location
-	Locked
-	Margin
MarginBottom	-
MarginLeft	-

MarginRight	-
MarginTop	-
-	MaximumSize
-	MinimumSize
-	Modifiers
Options	Options
OutlineLabel	-
OutlineParent	-
-	Padding
Palette	Palette
PlotMargin	PlotMargin
PlotStyle	PlotStyle
-	RenderMode
RightToLeft	RightToLeft
Rotated	Rotated
-	SelectedIndex
-	SelectedMode
-	SelectionStyle
Series	Series
SeriesGroups	-
-	Size
-	SmoothingMode
SplitHorzBehavior	-
SplitVertBehavior	-
Stacking	Stacking
Tag	-
-	Text
-	ToolTip
-	ToolTip on ttCopy
Top	-
Visible	-
Width	-
XLabelExpression	-
-	UseWaitCursor

-	Visible
Zorder	-

The following table shows the Series Object Model comparison.

FlexChartField	FlexChart
AltStyle	AltStyle
	Binding
	BindingMode
	BindingX
ChartType	ChartType
	DataMember
	DataSource
DataSourceName	
Hyperlink	
Label	
	Name
Style	Style
SymbolMarker	SymbolMarker
SymbolSize	SymbolSize
SymbolStyle	SymbolStyle
Visibility	Visibility
XExpression	
Y1Expression	
Y2Expression	
Y3Expression	
YExpression	

FlexChart Field Data Object Model

The following table lists objects and the main properties of FlexChart field.

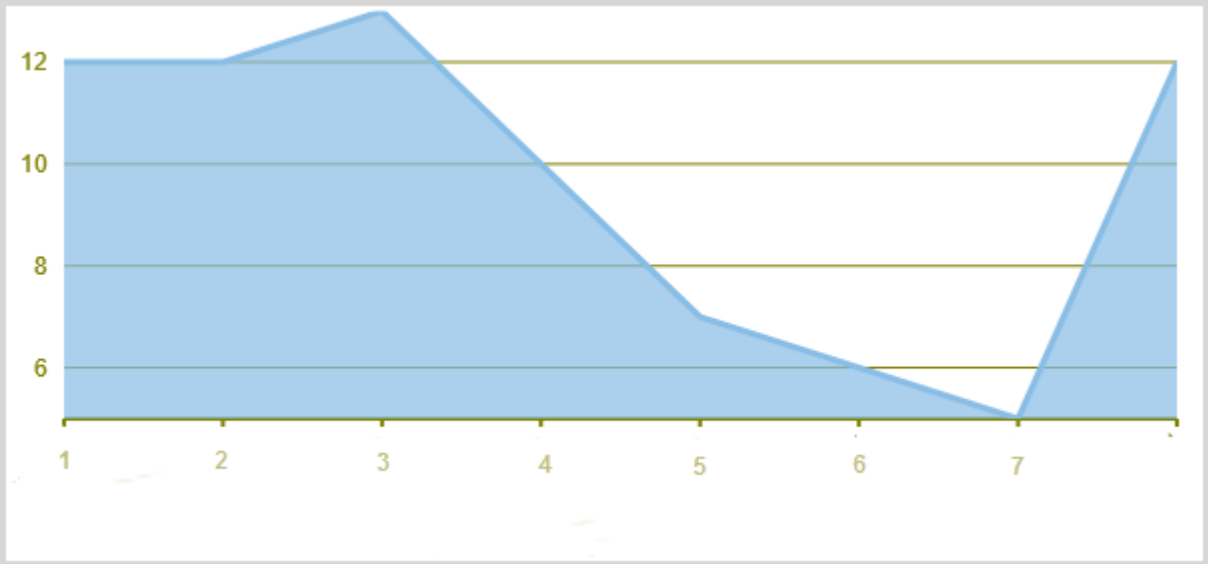
FlexChartField
Properties: DataSourceName, DataLabel.Content
CategoryGroups
Properties: GroupExpression, SortExpression, Sort, FilterExpression, FilterOutName
FlexChart Series

Properties: DataSourceName, XExpression, YExpression, Label
SeriesGroups
Properties: GroupExpression, SortExpression,Sort, FilterExpression, FilterOutName

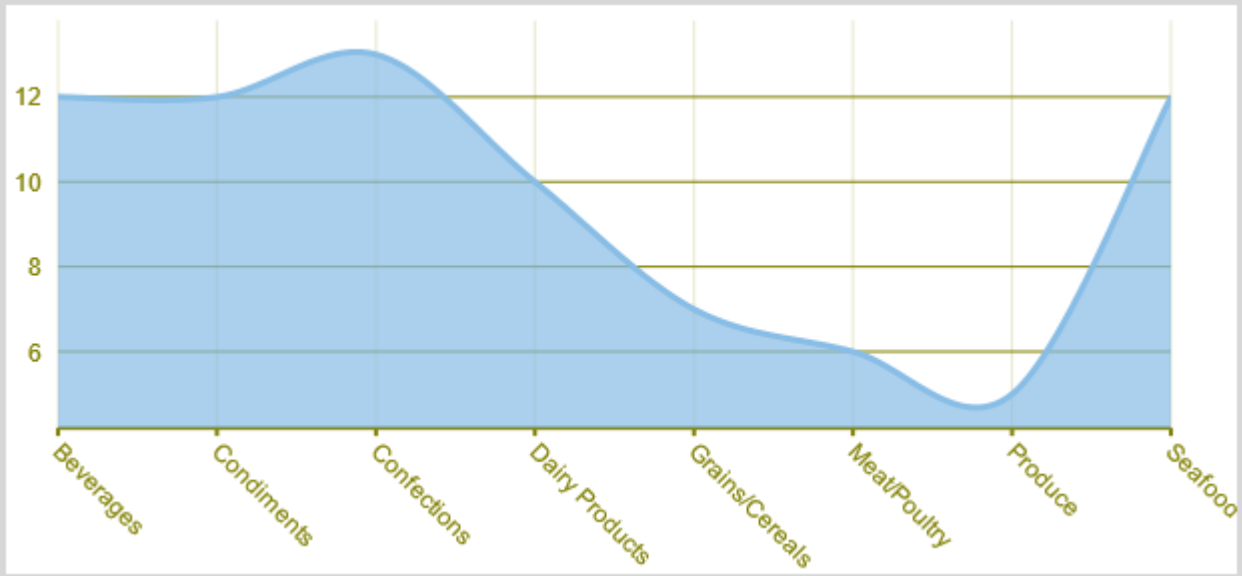
Supported Chart Types

The **FlexChart** field in **FlexReport** allows you to set a specific chart type using C1.Chart.ChartType property. It allows you to visualize data through thirteen different chart types, which includes Area, SplineArea, Bar, Bubble, Column, Scatter, Line, LineSymbols, Spline, SplineSymbols, Candlestick, HiLoOpenClose, and Funnel. The chart types can be easily selected using the ChartType property in the Properties window of the **C1FlexReportDesigner**.

Area chart: An Area chart draws data series by connecting data points against Y-axis and filling the area between the series and X-axis. Each series is drawn on top of the preceding series.

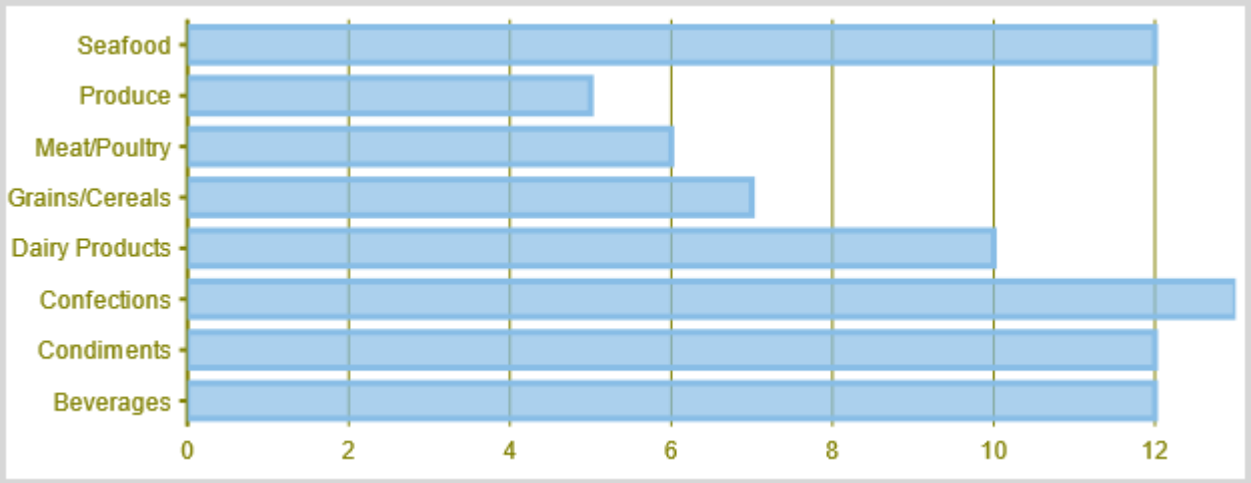


SplineArea chart: A SplineArea Chart is similar to an area chart. The only difference is that it connects data points using splines instead of straight lines and fills the area enclosed by the splines.

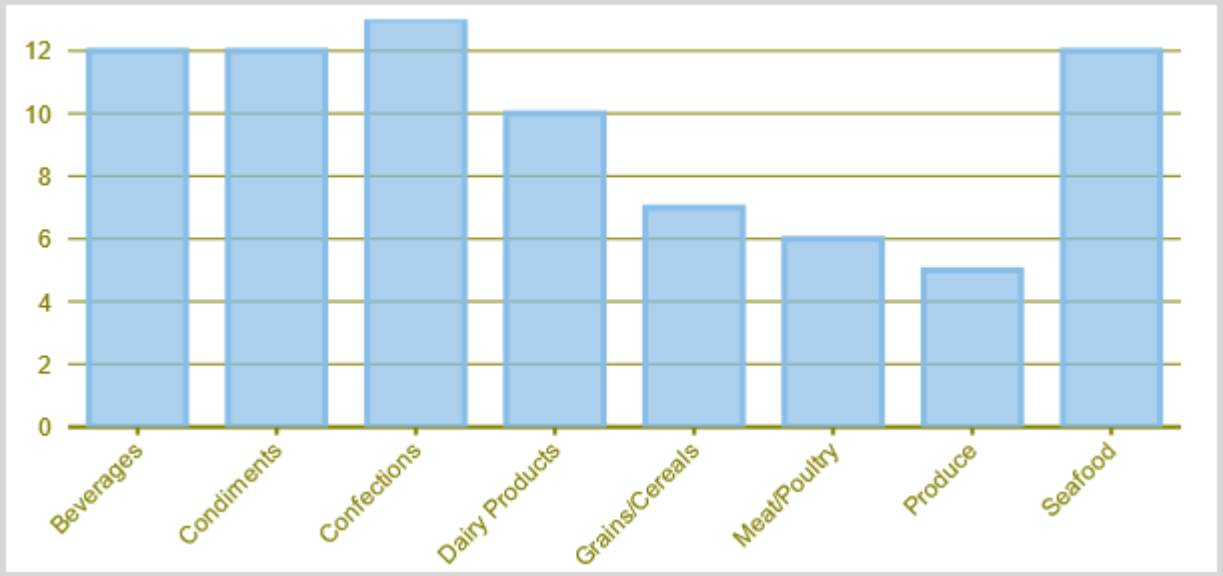


Bar chart: A Bar chart compares values across various categories or displays variations in a data series over time. It represents data series in the form of bars of the same color and width, whose length is determined by its value. Each

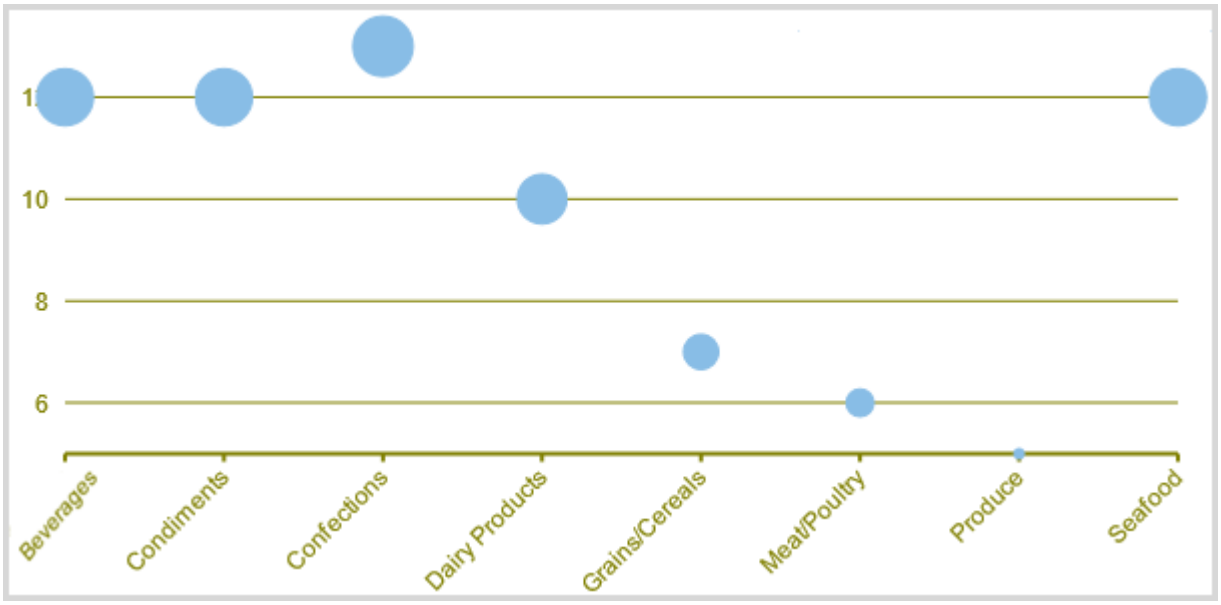
new series is displays horizontal bars for data series plotted against X-axis and arranges categories or items on Y-axis.



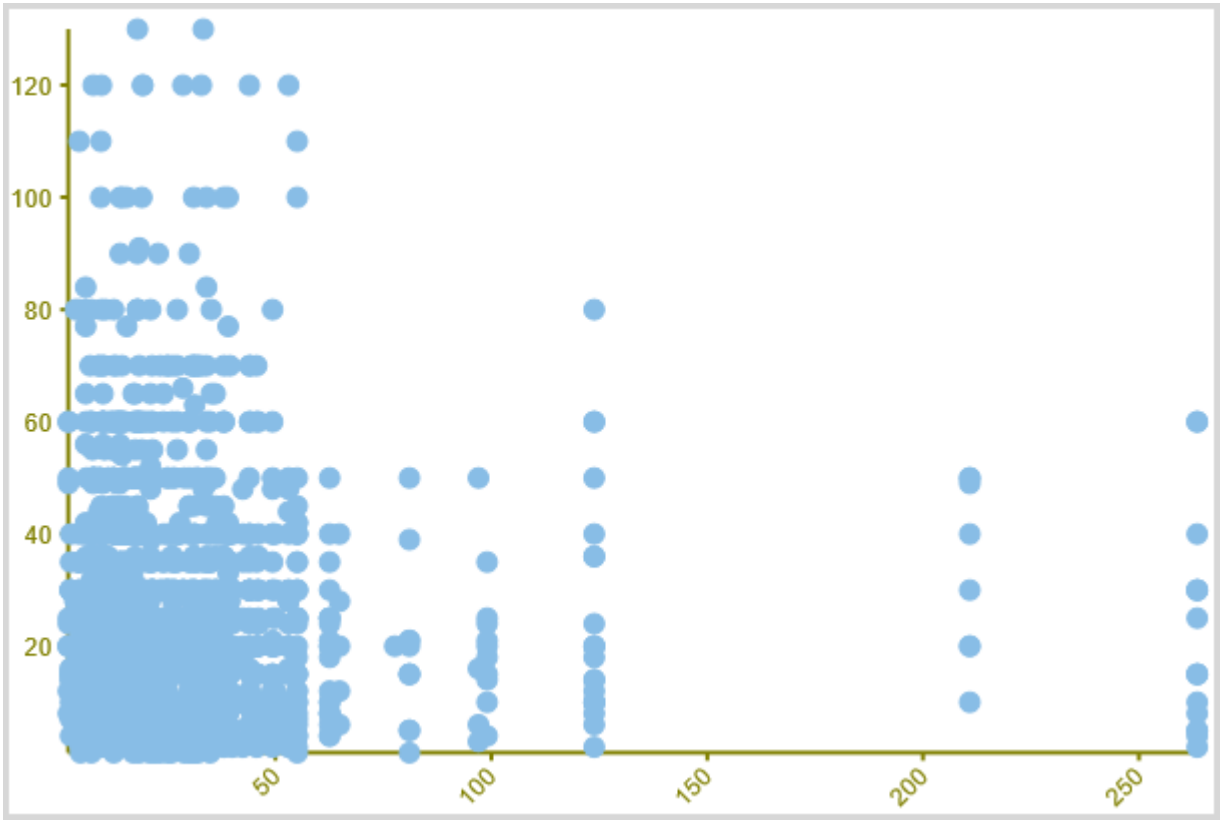
Column chart: . A Column chart, just like the Bar Chart, represents variation in a data series over time or compares different items. It displays values of one or more items as vertical bars against Y-axis and arranges items or categories on X-axis.



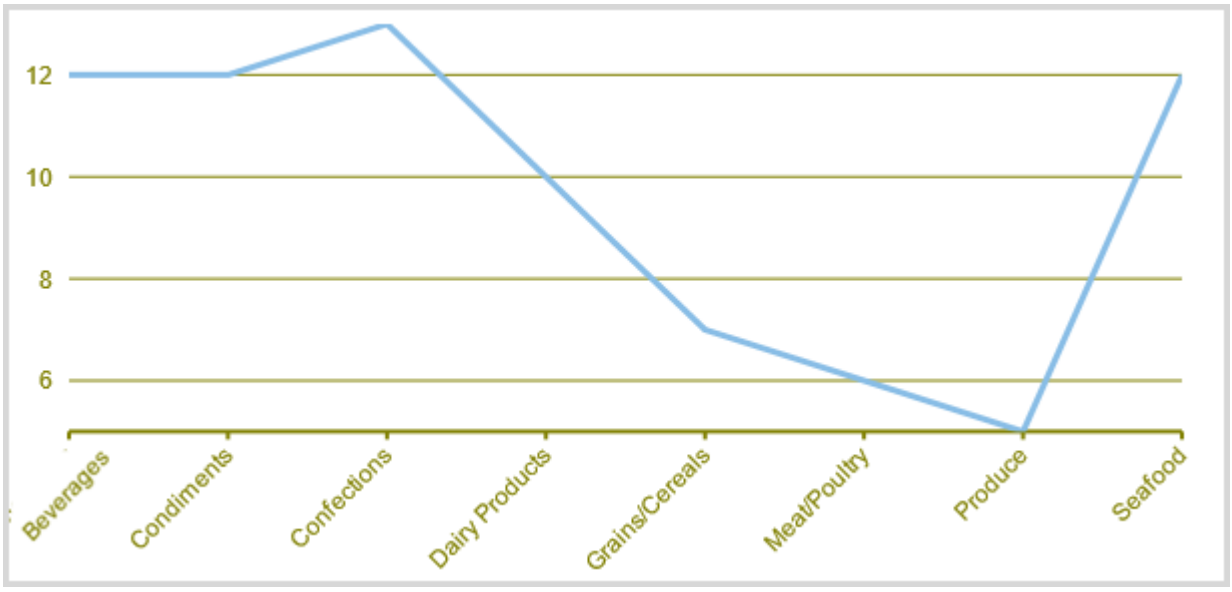
Bubble Chart: A Bubble chart combines two independent values to supply both the point y value and the point sizes. Bubble charts are used to represent an additional data value at each point by changing its size. The Y array elements determine the Cartesian position (as in a XY-Plot chart), and the Y1 element values determine the size of the bubble at each point. The size of the points can be encoded according to area or diameter.



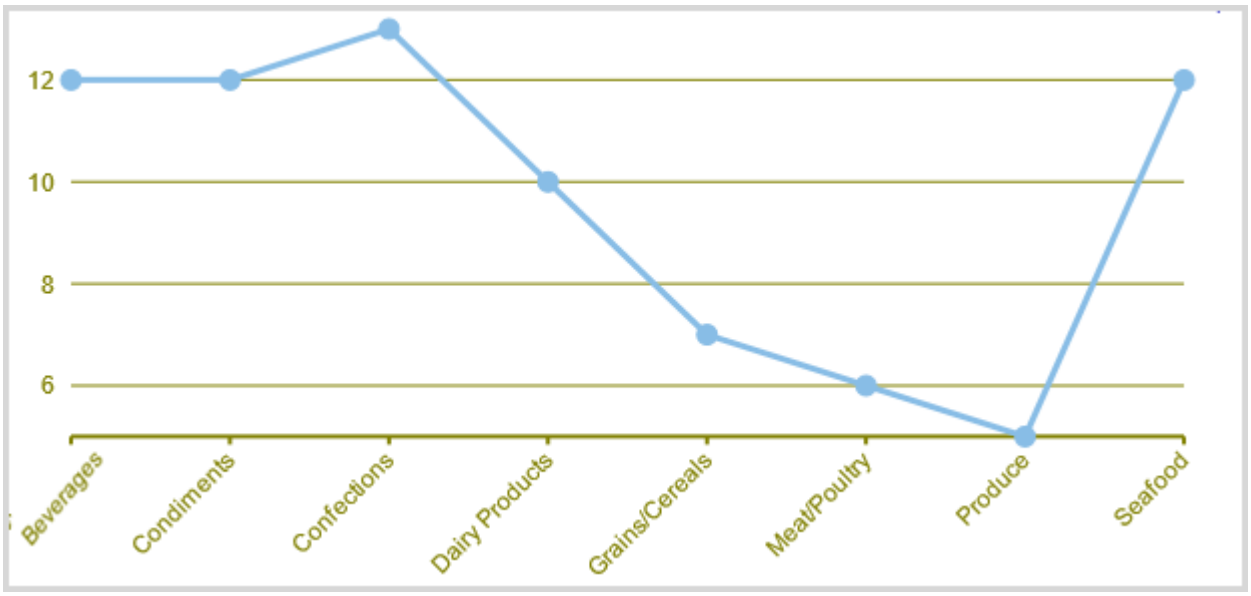
Scatter chart: A Scatter chart uses two values to represent each data point. It depicts relationship among items of different data series. This type of chart is often used to represent scientific data, and can highlight the deviation of assembled data from predicted data or result.



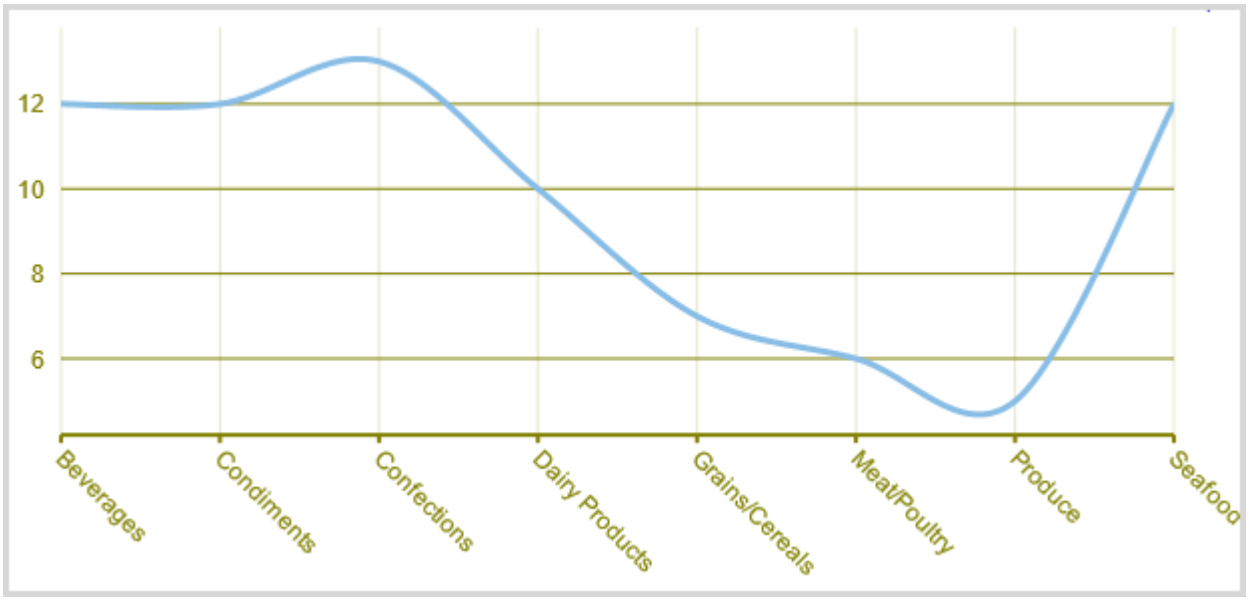
Line chart: A Line chart displays trends over a period of time by connecting different data points in a series with a straight line. It is the most effective way of denoting changes in the values between different groups of data.



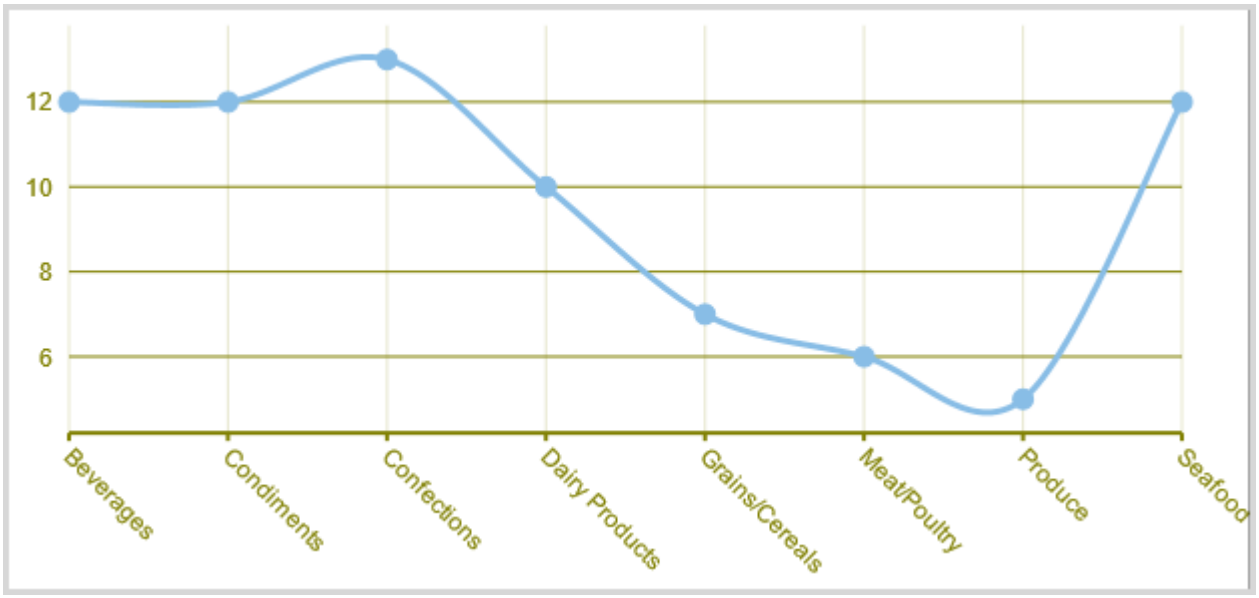
LineSymbols chart: A LineSymbols chart is a combination of the Line chart and the Scatter chart. The chart plots data points by using symbols and connects those data points by using lines.



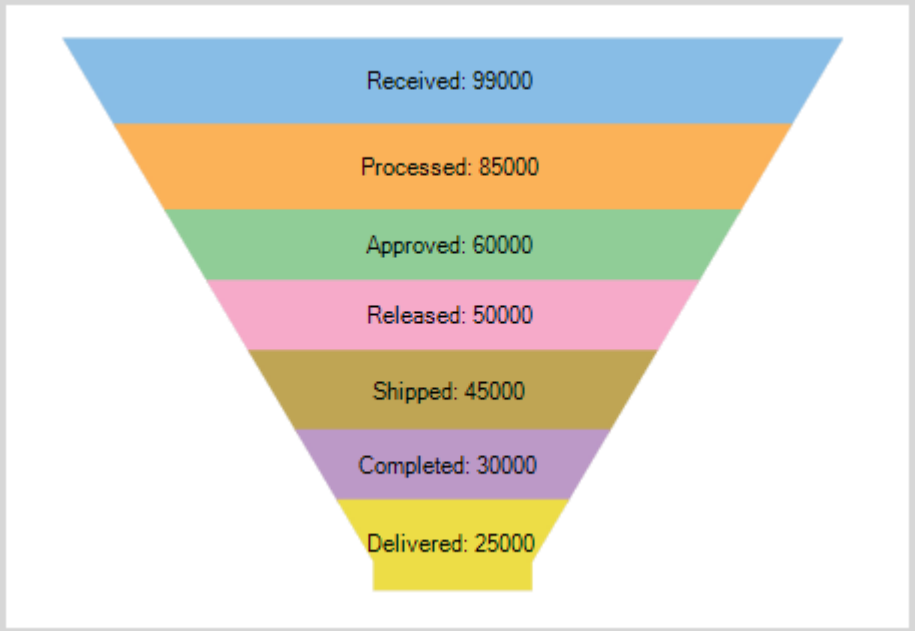
Spline chart: A Spline chart is similar to a line chart except that it connects data points by using splines rather than straight lines. It is specifically used for representing data that requires the use of curve fittings.



SplineSymbols: A SplineSymbols chart combines the Spline chart and the Scatter chart. The chart plots data points by using symbols and connects those data points by using splines.

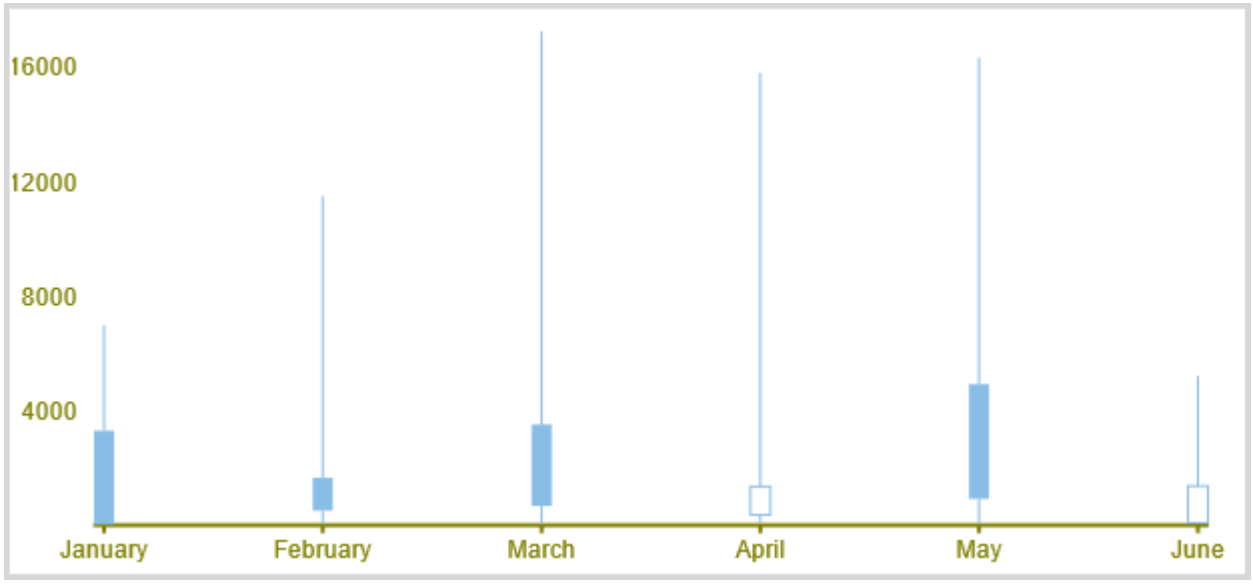


Funnel chart: A funnel chart represents sequential stages in a linear process. This chart can be useful in identifying potential problem areas in processes where it is noticeable at what stages and rate the values decrease.

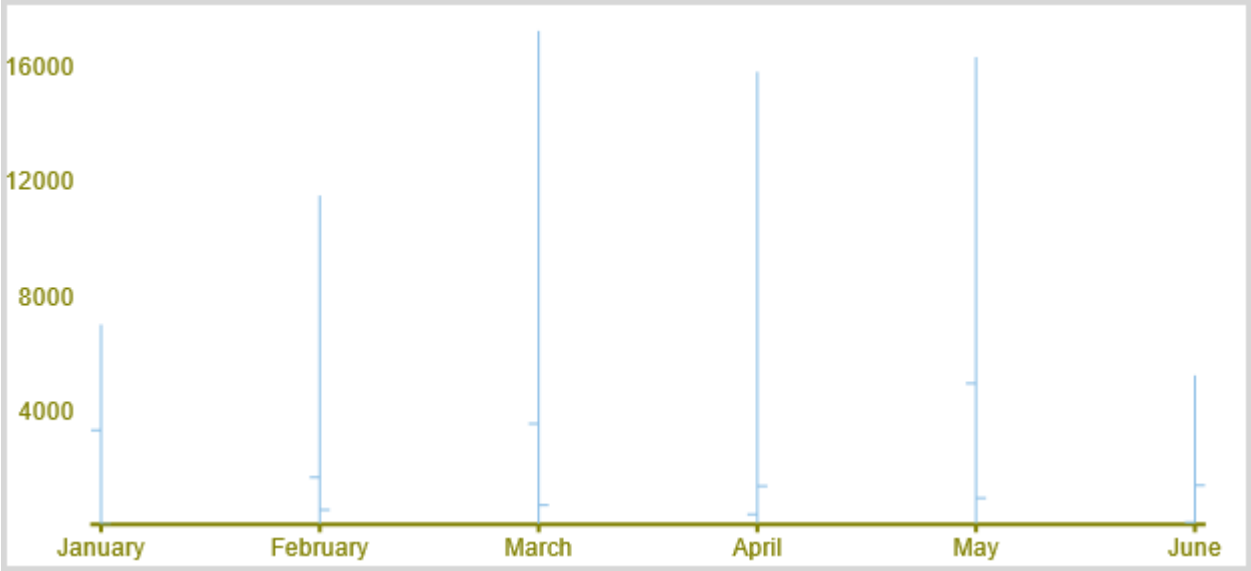


Stock Charts: Stock charts used in financial applications to show the opening, closing, high and low prices of a given stock. The types of stock charts are as follows:

- **Candlestick chart:** A Candlestick chart is a special type of HiLoOpenClose chart that integrates Bar and Line charts to depict a range of values over time. It consists of visual elements known as candles that are further comprised of three elements: body, wick, and tail. The body represents the opening and the closing value, while the wick and the tail represent the highest and the lowest value respectively.



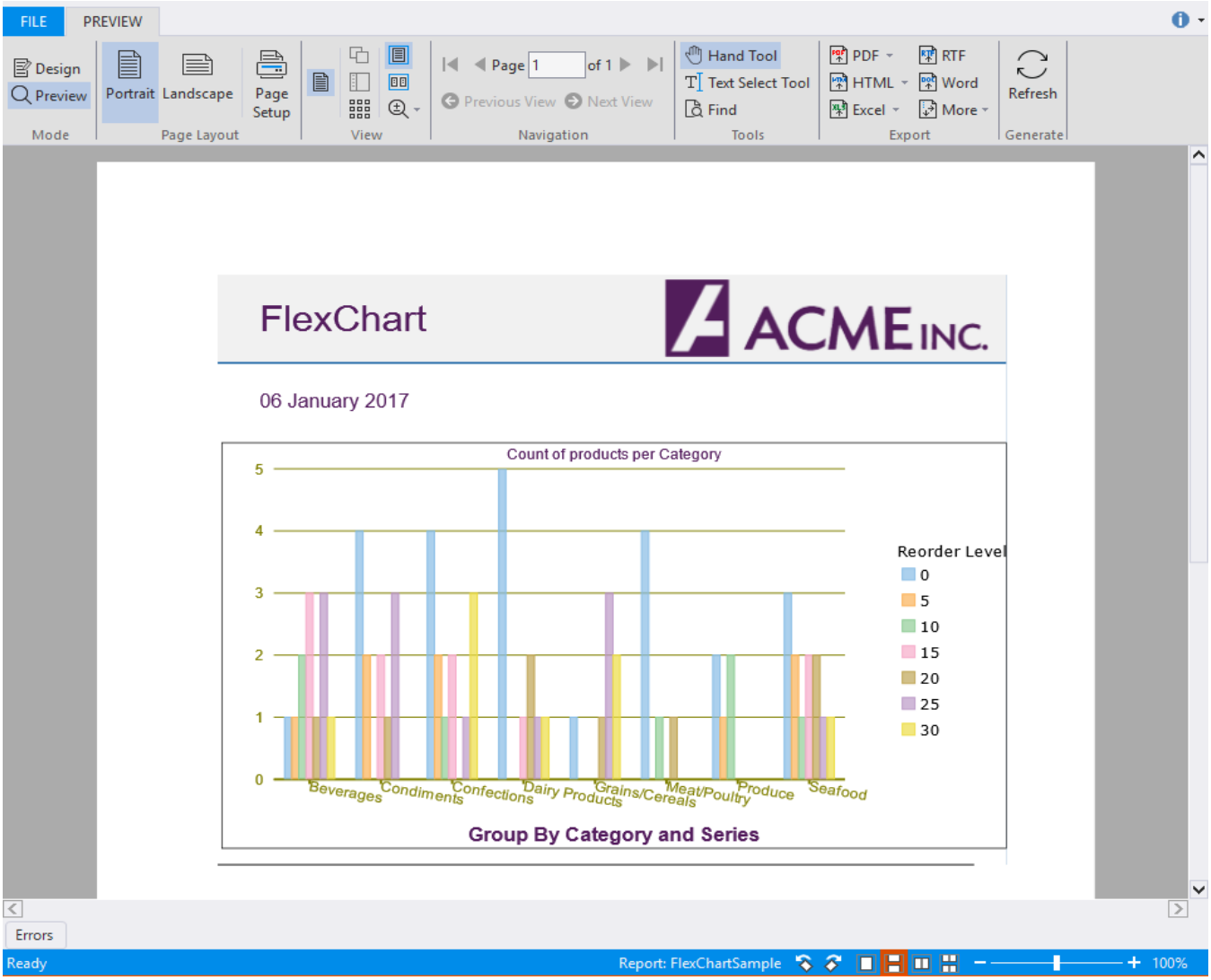
- **HiLoOpenClose:** HiLoOpenClose charts combine four independent values to supply high, low, open, and close data for a point in a series. In addition to showing the high and low value of a stock, the Y2 and Y3 array elements represent the stock's opening and closing price, respectively.



Grouping and Aggregates

FlexReport allows using **FlexChart** field to render a FlexChart in FlexReport and perform grouping and aggregation in it. Let us create a report to show grouping and aggregation of data.

The image below shows FlexReport containing FlexChart field where grouping and aggregation is performed on data.



To create a report showing grouping and aggregation of data in FlexChart, follow these steps.

1. In the **C1FlexReportDesigner** application, create a new report by navigating through the **Report Wizard**.
2. Bind the report to the **Main** data source, by specifying the following Sql Statement.
   ```
   SELECT Categories.*, Categories.CategoryID as CategoryID, Products.* FROM
   Categories INNER JOIN Products ON Categories.CategoryID = Products.CategoryID
   ```
3. Add a **SubSection** to the **Header** section and add **FlexChart** field to it.
4. In the **Properties** window, expand **AxisX** and set the value of LabelAngle property to **-10**.
5. Navigate to AxisX|Style and set Font properties to **Arial, 9pt, Regular**, StrokeColor to **Olive**, and StrokeWidth to **30**.
6. Navigate to AxisY and set Font properties to **Arial, 9pt, Regular**, StrokeColor to **Olive**, and StrokeWidth to **15**.
7. Set the Border Color to **Black**, Style to **Solid** and Width to **10**.
8. Navigate to CategoryGroups and click ellipsis button next to it.
   The **DataGroup Collection Editor** opens.
9. In the **DataGroup Collection Editor**, click **Add** button to add a data group and set it's GroupExpression to **CategoryID**.
10. Close the Editor.
11. Navigate to Footer|Content and set it to **Group By Category and Series**.
12. Navigate to Header|Content and set it to **Count of products per Category**.
13. Navigate to Legend|Title and set it to **Reorder Level**.
14. Navigate to Series and click the ellipsis button next to it.

15. In the **Series Collection Editor**, click **Add** button to add a series.
16. Navigate to Data|YExpression and set it's value to **Count(*)**.
    With **Count(*)**, we are setting aggregate data to the **FlexChartField**.
17. Close the Editor.
18. Navigate to SeriesGroups and click the ellipsis button next to it.
19. In the **Data Group Collection Editor**, click **Add** button to add a series and set it's GroupExpression to **RecordLevel**.
20. Close the Editor.
21. Navigate to XLabelExpression and set it to **CategoryName**.
22. Click **Preview** button to switch to Preview mode to view the report.

FlexChart Navigation

FlexChart field allows navigation to another report, a URL or script via the Hyperlink property of FlexChartField or a particular Series.

Suppose you want to show Total Orders per Year and monthly sales for a particular year. Both these tasks can be achieved in FlexReport using FlexChart navigation feature. In one report, the chart shows Total Orders per year and in the other, the chart shows Sales per Month along with the total orders and amount received every month. If you click on one series, the report navigates to the chart and pass the year for which Monthly sales are shown.

Let us create a report to navigate from a series to other report, or data within a report.

1. In the **C1FlexReportDesigner** application, create two new reports, **Orders Report** and **Sales Report**, by navigating through the **Report Wizard**.
2. Bind the reports to the **Main** data source, by specifying the following Sql Statement.
   ```
   SELECT o.OrderDate, od.Quantity * od.UnitPrice AS OrderItemSum FROM Orders AS o,
   [Order Details] AS od WHERE o.OrderId = od.OrderId
   ```
3. Add a parameter, **pYear**, to the report and set it's DataType to **Integer**, Prompt to **Year**, and Value to **2012**.
4. Add one **FlexChart** field in **Orders Report** and two **FlexChart** field in **Sales Report**.
5. Select **FlexChart** field in **Orders Report**.
6. In the **Properties** window, navigate to Series and click the ellipsis button next to it.
7. In the **Series Collection Editor**, click **Add** button to add a series data group, navigate to Data|YExpression and set it's value to **Sum(OrderItemSum)**.
8. Navigate to Hyperlink|LinkTarget and set it to **Bookmark**.
9. Click the ellipsis button next to the ParameterValues, add a parameter named **pYear** with **=Year(OrderDate)** value, and close the **ParametersValues** Editor.
10. Set the Hyperlink|Report to **Sales Report**.
11. Close the Editor.
12. Navigate to SeriesGroups and click the ellipsis button next to it.
13. In the **Data Group Collection Editor**, click **Add** button to add a series and set it's GroupExpression to **Year(OrderDate)**.
14. Close the Editor.
15. Select the first FlexChart field in Sales Report.
16. In the **Properties** window, navigate to Series and click the ellipsis button next to it.
17. In the **Series Collection Editor**, click **Add** button to add a series data group.
18. Navigate to Hyperlink|LinkTarget and set it to **Bookmark**.
19. Set the Bookmark to **=Month(OrderDate)**.
20. Close the Editor.
21. Navigate to SeriesGroups and click the ellipsis button next to it.
22. In the **Data Group Collection Editor**, click **Add** button to add a series and set it's GroupExpression to **Month(OrderDate)**.
23. Close the Editor.
24. Select the second **FlexChart** field in **Sales Report**.

25. In the **Properties** window, navigate to Series and click the ellipsis button next to it.
26. In the **Series Collection Editor**, click **Add** button to add a series data group, navigate to Data|YExpression and set it's value to **Sum(OrderSum)**.
27. Close the Editor.
28. Select the **Orders Report** and click **Preview** button to switch to the Preview mode to see how FlexChart navigation works in FlexReport.

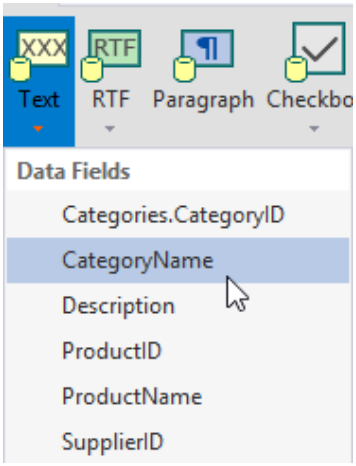
Text Field

The **Text** field is the most commonly used report field to display data. It is used to insert:

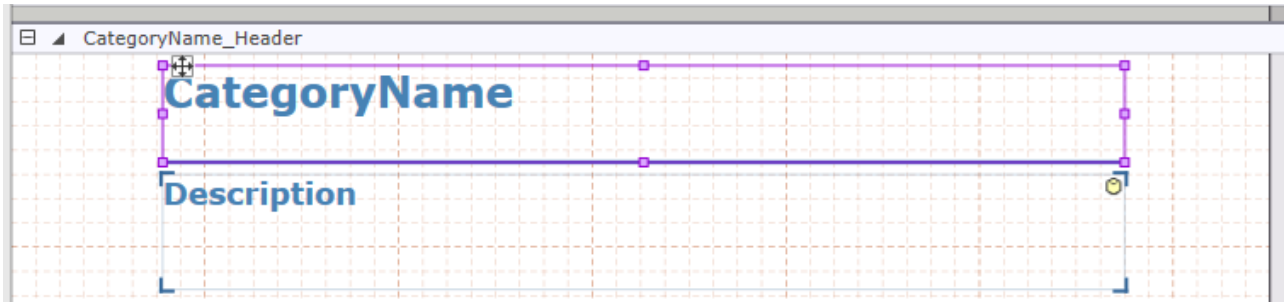
- a data-bound text field
- an unbound (static) text label

To add a data-bound text field using FlexReportDesigner application

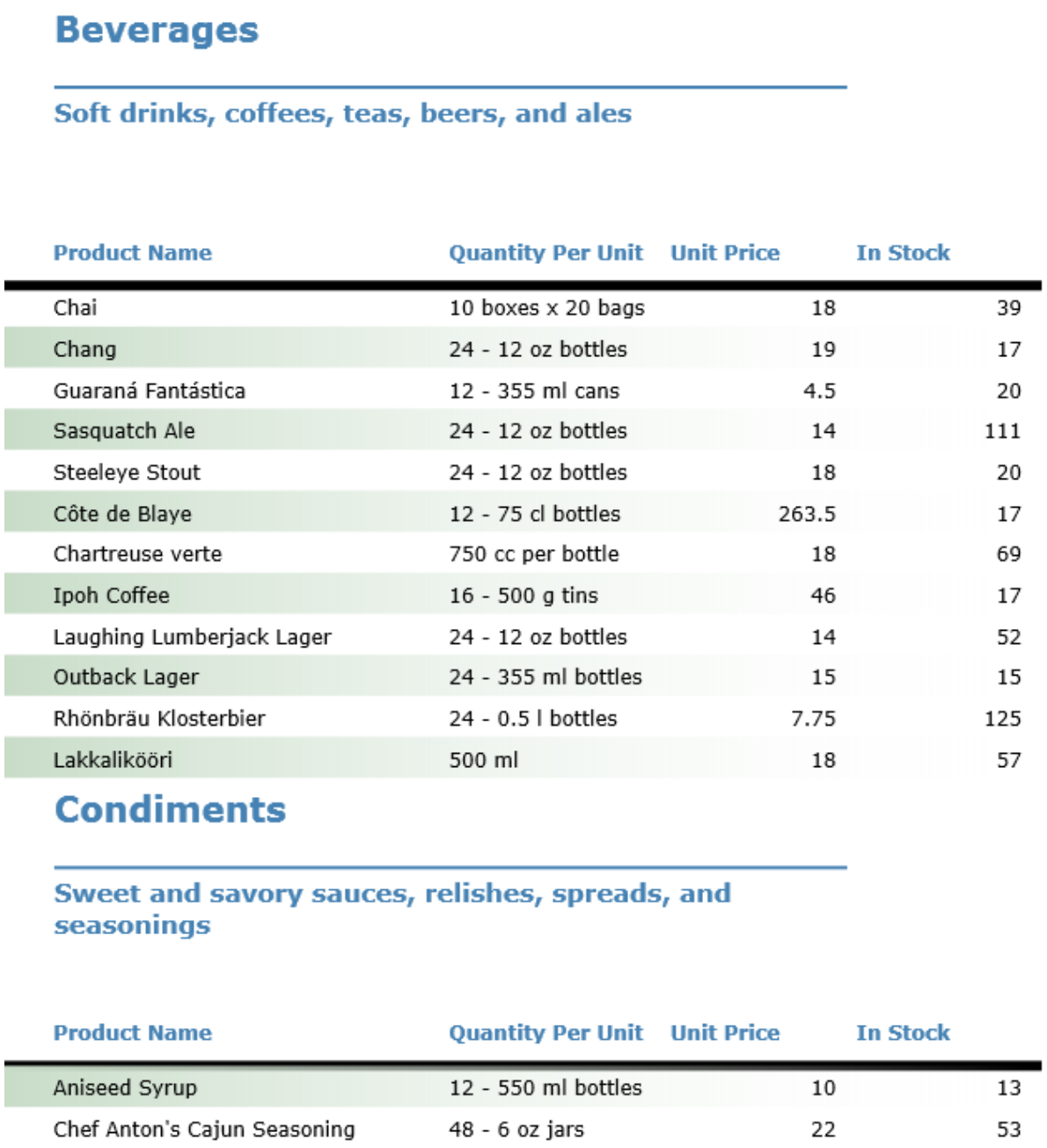
1. Create a new report in C1FlexReportDesigner.
2. Bind the report with Products Table.
3. Create a Group Header in which text field 'CategoryName' for the products will be displayed.
4. Go to the **Insert** tab and click the **Text** field icon. All the **Data Fields** (bound to the data source) are listed.



5. Add the 'CategoryName' data field to the group header section of the report.



6. Preview the report.



Beverages

Soft drinks, coffees, teas, beers, and ales

Product Name	Quantity Per Unit	Unit Price	In Stock
Chai	10 boxes x 20 bags	18	39
Chang	24 - 12 oz bottles	19	17
Guaraná Fantástica	12 - 355 ml cans	4.5	20
Sasquatch Ale	24 - 12 oz bottles	14	111
Steeleye Stout	24 - 12 oz bottles	18	20
Côte de Blaye	12 - 75 cl bottles	263.5	17
Chartreuse verte	750 cc per bottle	18	69
Ipoh Coffee	16 - 500 g tins	46	17
Laughing Lumberjack Lager	24 - 12 oz bottles	14	52
Outback Lager	24 - 355 ml bottles	15	15
Rhönbräu Klosterbier	24 - 0.5 l bottles	7.75	125
Lakkalikööri	500 ml	18	57

Condiments

Sweet and savory sauces, relishes, spreads, and seasonings

Product Name	Quantity Per Unit	Unit Price	In Stock
Aniseed Syrup	12 - 550 ml bottles	10	13
Chef Anton's Cajun Seasoning	48 - 6 oz jars	22	53

To add an unbound (static) text label using FlexReportDesigner:

1. Go to the **Insert** tab and click the **Text** field icon.
2. Drag the crosshair on the design area of the report and draw the field in the section in which you want the field to appear.
3. Click the field and enter the text you want to be displayed as a label.

Rtf Field

The **Rich Text Formatted (RTF)** field is used to display a formatted text. When you click this button, a menu appears where you can select other fields that are contained in the same report definition file to be displayed in RTF format. These are data-bound RTF fields.

RTF fields are particularly used in creating Mail Merge reports.

To create a mail merge using RTF field in FlexReportDesigner application:

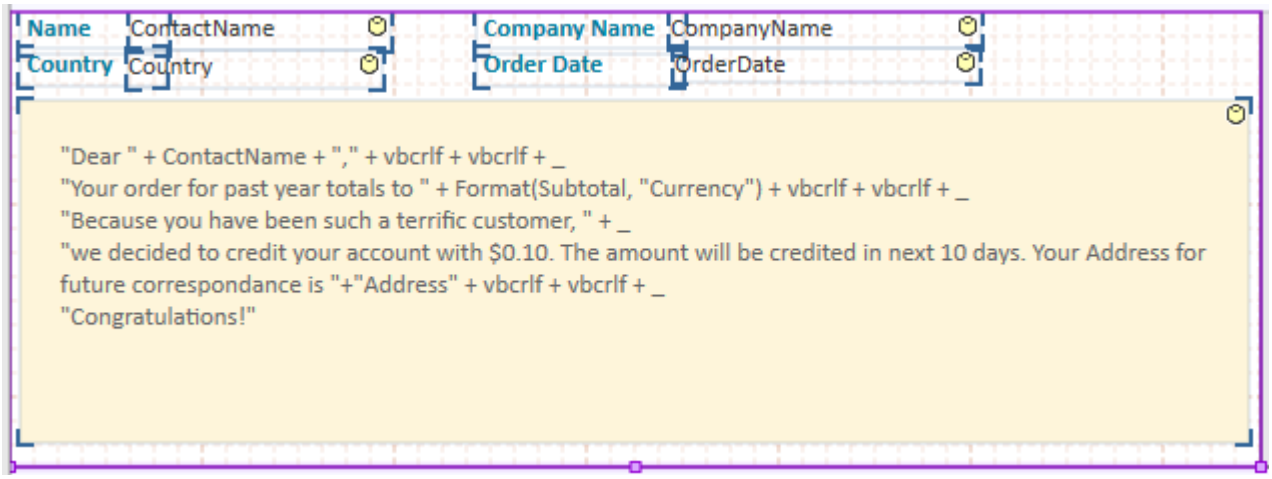
1. Create a new report C1FlexReportDesigner.
2. Bind the report to 'Customers' and 'Orders' Table by giving the following Sql statement:

```
SELECT
Customers.CustomerID, Customers.CompanyName,
Customers.ContactName, Customers.Address,
Customers.City, Customers.Region, Customers.Country, Customers.PostalCode,
Orders.OrderID, Orders.OrderDate, [Order Subtotals].Subtotal
FROM Customers INNER JOIN ([Order Subtotals] INNER JOIN Orders
ON [Order Subtotals].OrderID = Orders.OrderID)
ON Customers.CustomerID = Orders.CustomerID
WHERE CompanyName = "Ernst Handel"
```

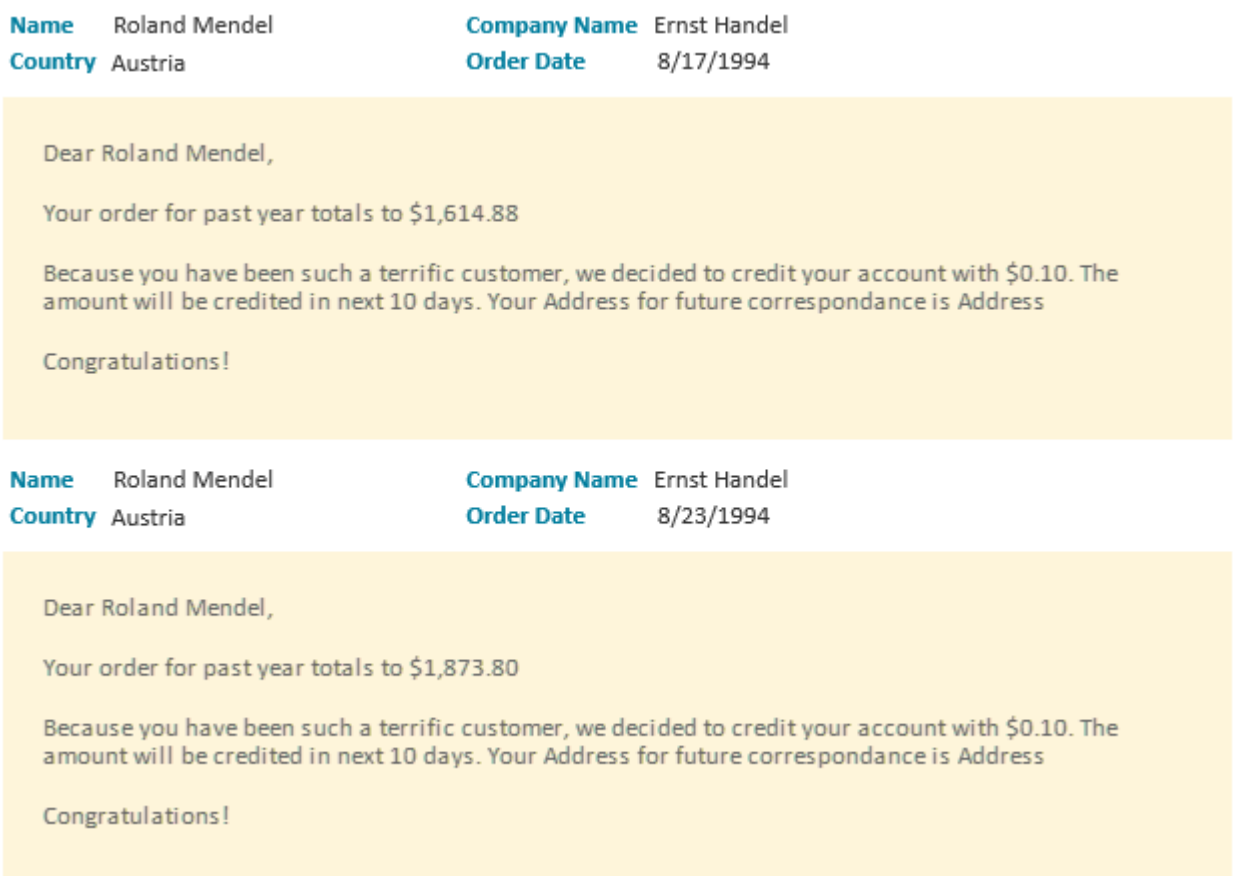
3. From the **Insert** tab add the **RTF** field.
4. From the Properties window, set the Background color to a light color.
5. Set the **Text** property to following expression:

```
"Dear " + ContactName + "," + vbcrlf + vbcrlf + _
"Your order for past year totals to " + Format(Subtotal, "Currency") + vbcrlf +
vbcrlf + _
"Because you have been such a terrific customer, " + _
"we decided to credit your account with $0.10. The amount will be credited in
next 10 days._
Your Address for future correspondance is "+"Address" + vbcrlf + vbcrlf + _
"Congratulations!"
```

6. Arrange the fields as shown.



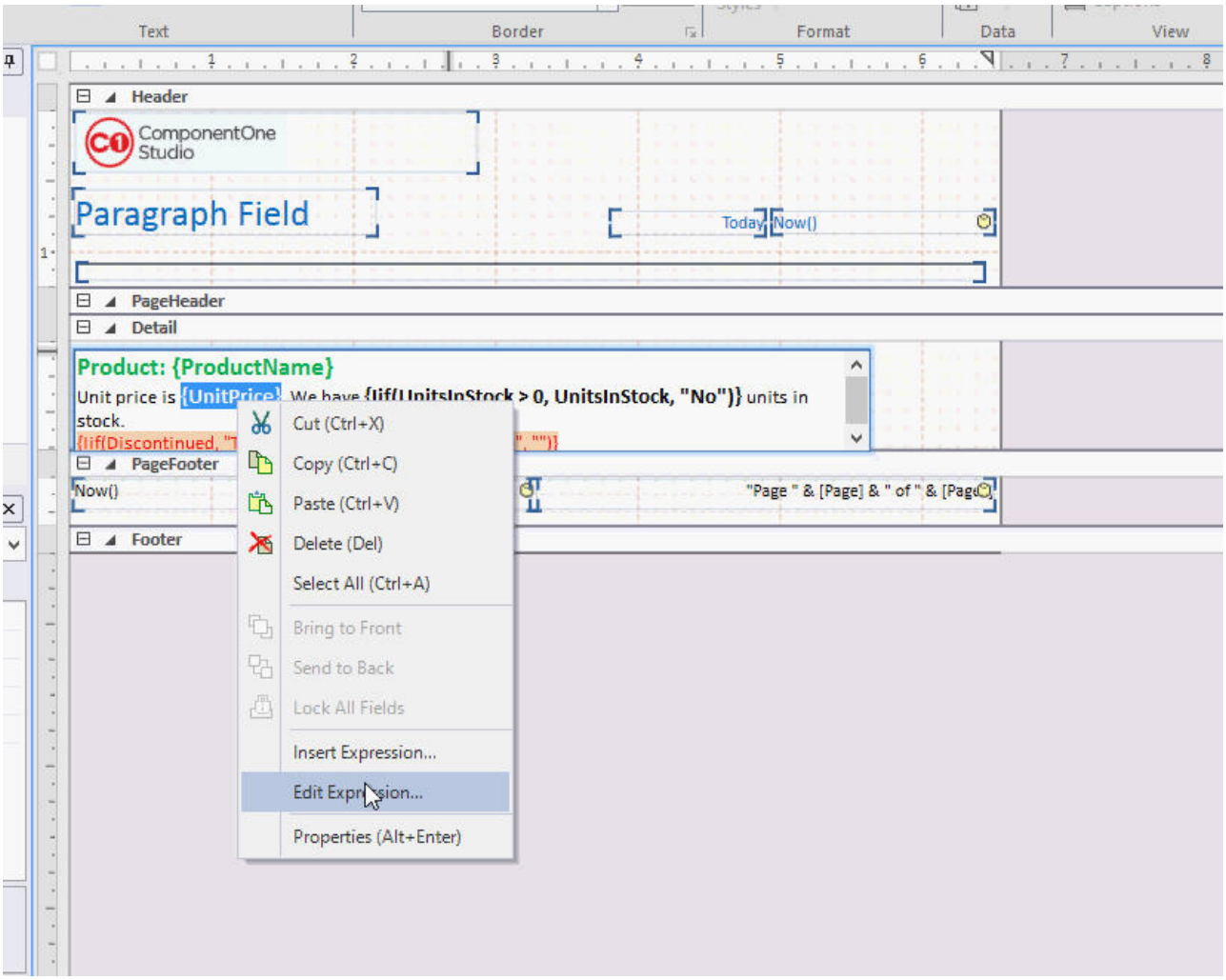
7. Preview the report.



Paragraph Field

In order to use multi formatted text, Database fields, scripts, Parameters, Calculated Fields within a single Field, **Paragraph** field is the best option for you. In edit mode, you can implement following features through Paragraph field:

- Type static text.
- Insert a newline by pressing Ctrl+Enter.
- Set font/fore color/back color for any part of the text.
- Insert expressions (like DB field values) within a single field together with the text.
- Press Alt+Enter to open Properties on any Field.
- Right-click any expression and choose 'Edit expression' to open the Expression editor.



To add a Paragraph field in FlexReportDesigner application:

1. Create a new FlexReport in C1FlexReportDesigner.
2. Bind it with **Products** Table.
3. From the Insert tab, add **Paragraph** field on the report.
4. Double-click the **Paragraph** field to enter in edit more.
5. Type "Product:"

To insert Database fields:

1. In edit mode, right-click the Paragraph field and select **Insert Expression**.
2. In the Expression editor, select 'ProductID' from **DatabaseFields** dropdown.
3. Click **Done**.
4. Select "Product:{ProductID}" and set **Forecolor** as Green from the Ribbon.

To change new line in edit mode:

1. In edit mode, press Ctrl+Enter.
2. In the next line, type static text "Unit Price is" and insert 'UnitPrice' Database Field using steps above.
3. Set {UnitPrice} as Bold.

To insert scripts:

1. After the text - "UnitPrice is {UnitPrice}", type - "We have".
2. Right-click and select **Insert Expression**.
3. Add following script: `Iif(UnitsInStock > 0, UnitsInStock, "No")`
4. Click **Done**.

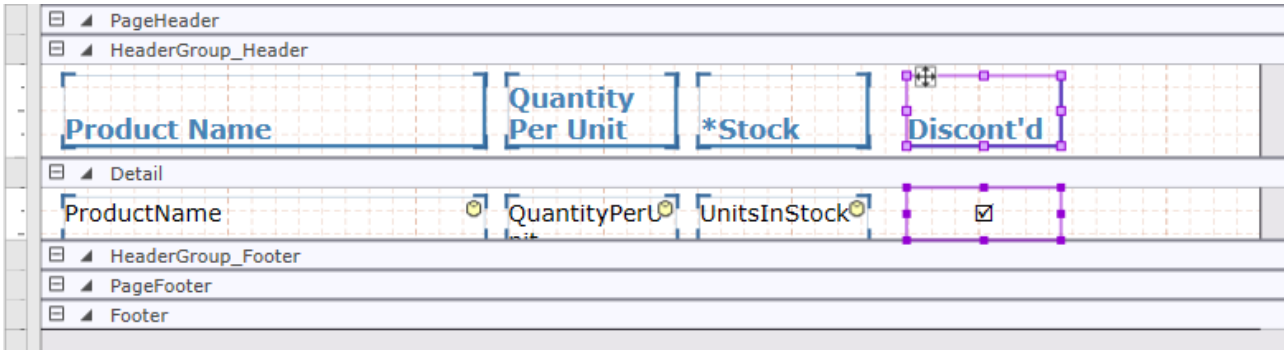
5. After this script expression, type static text - "units in stock."
6. Select the script while in edit mode and set it to Bold.
7. Press Ctrl+Enter.
8. In newline, right-click and select **Insert Expression**.
9. Type following script: `Iif(Discontinued, "This product is no longer available", "") /`
10. Click **Done**.
11. Select the above mentioned script and set **Highlight color** to 'Orange, Accent 6 Lighter 60%' and **Forecolor** to Red.
12. Preview the report.

Checkbox Field

The **Checkbox** field is used to add a visual for Yes or No. The checkbox field, by default, appears as an empty box with the text on the right. It takes the boolean value; if it evaluates to true, a check mark appears in the check box.

To add data bound check box in FlexReportDesigner application:

1. Create a new FlexReport in C1FlexReportDesigner.
2. Bind it with **Products** Table. Select these fields - 'Product Name', 'Quantity Per Unit', 'Stock', and 'Discontinued'.
3. From the Insert tab, click **Checkbox** field and select 'Discontinued' data field.
4. Drop the data field in the Detail section of the report, below the Discontinued label in the group header.
5. From Properties window, set **CheckAlign** to CenterMiddle.



6. Preview the report.

Product Name	Quantity Per Unit	*Stock	Discont'd
Chai	10 boxes x 20 bags	39	☐
Chang	24 - 12 oz bottles	17	☐
Aniseed Syrup	12 - 550 ml bottles	13	☐
Chef Anton's Cajun Seasoning	48 - 6 oz jars	53	☐
Chef Anton's Gumbo Mix	36 boxes	0	☑
Grandma's Boysenberry Spread	12 - 8 oz jars	120	☐
Uncle Bob's Organic Dried Pears	12 - 1 lb pkgs.	15	☐
Northwoods Cranberry Sauce	12 - 12 oz jars	6	☐
Mishi Kobe Niku	18 - 500 g pkgs.	29	☑
Ikura	12 - 200 ml jars	31	☐

Barcode Field

Barcodes in FlexReport let you integrate several industry-standard barcodes in Barcode field, that can be quickly and easily generated in your reports. Simply drop the barcode field on your report, select the barcode symbology, provide the text, and you are done!

The functionality of barcodes in FlexReport further extended by properties associated with them. The checksums to the value being encoded are automatically added to eliminate reader errors.

For barcodes, **C1.Win.Barcode** assembly is used.

Barcode Symbology

Barcode symbology specifies the encoding scheme used to convert character data into the pattern of wide and narrow bars and spaces in a barcode. The following table illustrates the barcode symbology used in **FlexReport**.

Style Name	Example	Description
Ansi39		ANSI 3 of 9 (Code 39) uses upper case, numbers, - , * $ / + %. This is the default barcode style.
Ansi39x		ANSI Extended 3 of 9 (Extended Code 39) uses the complete ASCII character set.
Codabar		Codabar uses A B C D + - : . / $ and numbers.

Code_128_A		Code 128 A uses control characters, numbers, punctuation, and upper case.
Code_128_B		Code 128 B uses punctuation, numbers, upper case and lower case.
Code_128_C		Code 128 C uses only numbers.
Code_128auto		Code 128 Auto uses the complete ASCII character set. Automatically selects between Code 128 A, B and C to give the smallest barcode.
Code_2_of_5		Code 2 of 5 uses only numbers.
Code93		Code 93 uses uppercase, % $ * / , + -, and numbers.
Code25intlv		Interleaved 2 of 5 uses only numbers.
Code39		Code 39 uses numbers, % * $ /. , - +, and upper case.
Code39x		Extended Code 39 uses the complete ASCII character set.
Code49		Code 49 is a 2D high-density stacked barcode containing two to eight rows of eight characters each. Each row has a start code and a stop code. Encodes the complete ASCII character set.
Code93x		Extended Code 93 uses the complete ASCII character set.

DataMatrix		Data Matrix is a high density, two-dimensional barcode with square modules arranged in a square or rectangular matrix pattern.
EAN_13		EAN-13 uses only numbers (12 numbers and a check digit). It takes only 12 numbers as a string to calculate a check digit (CheckSum) and add it to the thirteenth position. The check digit is an additional digit used to verify that a barcode has been scanned correctly. The check digit is added automatically when the CheckSum property is set to True.
EAN8		EAN-8 uses only numbers (7 numbers and a check digit).
EAN128FNC1		EAN-128 is an alphanumeric one-dimensional representation of Application Identifier (AI) data for marking containers in the shipping industry. This type of barcode contains the following sections: • Leading quiet zone (blank area) • Code 128 start character • FNC (function) 1 character which allows scanners to identify this as an EAN-128 barcode • Data (AI plus data field) • Symbol check character (Start code value plus product of each character position plus value of each character divided by 103. The checksum is the remainder value.) • Stop character • Trailing quiet zone (blank area) The AI in the Data section sets the type of the data to follow (i.e. ID, dates, quantity, measurements, etc.). There is a specific data structure for each type of data. This AI is what distinguishes the EAN-128 code from Code 128.

		Multiple AIs (along with their data) can be combined into a single barcode. EAN128FNC1 is a UCC/EAN-128 (EAN128) type barcode that allows you to insert FNC1 character at any place and adjust the bar size, etc., which is not available in UCC/EAN-128. To insert FNC1 character, set "\n" for C#, or "vbLf" for VB to Text property at runtime.
IntelligentMail		Intelligent Mail, formerly known as the 4-State Customer Barcode, is a 65-barcode used for domestic mail in the U.S.
JapanesePostal		This is the barcode used by the Japanese Postal system. Encodes alpha and numeric characters consisting of 18 digits including a 7-digit postal code number, optionally followed by block and house number information. The data to be encoded can include hyphens.
Matrix_2_of_5		Matrix 2 of 5 is a higher density barcode consisting of 3 black bars and 2 white bars.
MicroPDF417		MicroPDF417 is two-dimensional (2D), multi-row symbology, derived from PDF417. Micro-PDF417 is designed for applications that need to encode data in a two-dimensional (2D) symbol (up to 150 bytes, 250 alphanumeric characters, or 366 numeric digits) with the minimal symbol size. MicroPDF417 allows you to insert an FNC1 character as a field separator for variable length Application Identifiers (AIs). To insert FNC1 character, set "\n" for C#, or "vbLf" for VB to Text property at runtime.
MSI		MSI Code uses only numbers.

Pdf417		Pdf417 is a popular high-density 2-dimensional symbology that encodes up to 1108 bytes of information. This barcode consists of a stacked set of smaller barcodes. Encodes the full ASCII character set. It has ten error correction levels and three data compaction modes: Text, Byte, and Numeric. This symbology can encode up to 1,850 alphanumeric characters or 2,710 numeric characters.
PostNet		PostNet uses only numbers with a check digit.
QRCode		QRCode is a 2D symbology that is capable of handling numeric, alphanumeric and byte data as well as Japanese kanji and kana characters. This symbology can encode up to 7,366 characters.
RM4SCC		Royal Mail RM4SCC uses only letters and numbers (with a check digit). This is the barcode used by the Royal Mail in the United Kingdom.
RSS14		RSS14 is a 14-digit Reduced Space Symbology that uses EAN.UCC item identification for point-of-sale omnidirectional scanning.
RSS14Stacked		RSS14Stacked uses the EAN.UCC information with Indicator digits as in the RSS14Truncated, but stacked in two rows for a smaller width. RSS14Stacked allows you to set Composite Options, where you can select the type of the barcode in the **Type** drop-down list and the value of the composite barcode in the **Value** field.
RSS14StackedOmnidirectional		RSS14StackedOmnidirectional uses the EAN.UCC information with omnidirectional scanning as in the RSS14, but stacked in two rows for a smaller width.

RSS14Truncated		RSS14Truncated uses the EAN.UCC information as in the RSS14, but also includes Indicator digits of zero or one for use on small items not scanned at the point of sale.
RSSExpanded		RSSExpanded uses the EAN.UCC information as in the RSS14, but also adds AI elements such as weight and best-before dates. RSSExpanded allows you to insert an FNC1 character as a field separator for variable length Application Identifiers (AIs). To insert FNC1 character, set “\n” for C#, or “vbLf” for VB to Text property at runtime.
RSSExpandedStacked		RSSExpandedStacked uses the EAN.UCC information with AI elements as in the RSSExpanded, but stacked in two rows for a smaller width. RSSExpandedStacked allows you to insert an FNC1 character as a field separator for variable length Application Identifiers (AIs). To insert FNC1 character, set “\n” for C#, or “vbLf” for VB to Text property at runtime.
RSSLimited		RSS Limited uses the EAN.UCC information as in the RSS14, but also includes Indicator digits of zero or one for use on small items not scanned at the point of sale. RSSLimited allows you to set Composite Options, where you can select the type of the barcode in the **Type** drop-down list and the value of the composite barcode in the **Value** field.
UCCEAN128		UCC/EAN –128 uses the complete ASCII character Set. This is a special version of Code 128 used in HIBC applications.
UPC_A		UPC-A uses only numbers (11 numbers and a check digit).

UPC_E0		UPC-E0 uses only numbers. Used for zero-compression UPC symbols. For the Caption property, you may enter either a six-digit UPC-E code or a complete 11-digit (includes code type, which must bezero) UPC-A code. If an 11-digit code is entered, the Barcode control will convert it to a six-digit UPC-E code, if possible. If it is not possible to convert from the 11-digit code to the six-digit code, nothing is displayed.
UPC_E1		UPC-E1 uses only numbers. Used typically for shelf labeling in the retail environment. The length of the input string for U.P.C. E1 is six numeric characters.

You can directly insert a barcode field in the **FlexReportDesigner** using the **BarCode** property in the Properties window. You can also use **Barcode** to set the type of BarCode in the barcode field.

Note that the following barcodes support FNC1 characters:

- EAN128FNC1
- MicroPDF417
- RSSExpanded
- RSSExpandedStacked

Barcode Properties

The **BarcodeOptions** provides additional properties for rendering barcodes in **FlexReport**. Following are the common properties exposed by **BarcodeOptions**:

- **BarDirection**: Lets you select the barcode's direction, horizontally or vertically. The available options are LeftToRight, RightToLeft, TopToBottom, and BottomToTop. The direction of barcode can also be set using **BarDirectionEnum**.
- **CaptionGrouping**: Lets you split the text of the caption into groups for the barcode types it supports. Its value is either True or False.
- **CaptionPosition**: Lets you select the caption's vertical position relative to the barcode symbol. The available options are None, Above, and Below.
- **CheckSumEnabled**: Determines whether a checksum of the barcode will be computed and included in the barcode when applicable.
- **TextAlign:** Lets you select the caption text alignment. The available options are Left, Center, and Right.
- **SupplementNumber:** Lets you specify the supplement for the barcode data, supplement is 2 or 5 digit for EAN or UPC symbologies.
- **SizeOptions:**
  - **BarHeight:** Specifies the height of a barcode in twips.
  - **ModuleSize:** Specifies the module (narrowest bar width) of a barcode in twips.
  - **NarrowWideRatio:** Specifies the ratio between the width of narrow and wide bars.
  - **SizeMode:** Specifies the sizing mode of a barcode. The options available are:
    - **Normal:** Keeps the size of a barcode same as the original size.
    - **Scale:** Scales the barcode image to take as much field area as possible. Different type of barcodes are scaled in different ways; for example, in Bar type barcodes like Code128, the height

is increased and in barcodes such as Matrix, Rss, and Composite, height and width get scaled proportionally.

- **SupplementSpacing:** Specifies the spacing between the main and the supplement barcodes.

Other options exposed by **BarcodeOptions** corresponding to different barcode styles are as follows:

Code49:

- **Grouping:** Lets you use grouping in the barcode. Its value is either True or False.
- **Group:** Obtains or sets group numbers for barcode grouping. Its value is between 0 and 8.

DataMatrix:

- **EccMode:** Lets you select the ECC mode. The possible values are ECC000, ECC050, ECC080, ECC100, ECC140, or ECC200.
- **Ecc200SymbolSize:** Lets you select the size of the ECC200 symbol. The default value is SquareAuto.
- **Ecc200EncodingMode:** Lets you select the ECC200 encoding mode. The possible values are Auto, ASCII, C40, Text, X12, EDIFACT, or Base256.
- **Ecc000_140SymbolSize:** Lets you select the size of the ECC000_140 symbol.
- **StructuredAppend:** Lets you select whether the current barcode symbol is part of structured append symbols.
- **StructureNumber:** Lets you specify the structure number of the current symbol within the structured append symbols.
- **FileIdentifier:** Lets you specify the file identifier of a related group of structured append symbols. The valid file indentifier value should be within [1,254]. Setting file identifier to 0 lets the file identifier be calculated automatically.

GS1Composite:

- **Type:** Lets you select the composite symbol type. Its value can be None or CCA. CCA (Composite Component - Version A) is the smallest variant of the 2-dimensional composite component.
- **Value:** Gets or sets the CCA character data.

MicroPDF417:

- **CompactionMode:** Lets you select the type of CompactionMode. The possible values are Auto, TextCompactionMode, NumericCompactionMode, or ByteCompactionMode.
- **FileID:** Lets you specify the file id of the structured append symbol. It takes the value from 0 to 899.
- **SegmentCount:** Lets you specify the segment count of the structured append symbol. It takes the value from 0 to 99999.
- **SegmentIndex:** Lets you specify the segment index of the structured append symbol. It takes the value from 0 to 99998 and less than the value of segment count.
- **Version:** Lets you select the symbol size. The default value is ColumnPriorAuto.

PDF417:

- **Column:** Lets you specify the column numbers for the barcode.
- **Row:** Lets you specify the row numbers for the barcode.
- **ErrorLevel:** Lets you specify the error correction level for the barcode.
- **Type:** Lets you select the type of PDF417 barcode. The available types are Normal and Simple.

QRCode

- **Model:** Lets you select the model of QRCode. The available models are Model1 and Model2.
- **ErrorLevel:** Lets you select the error correction level for the barcode. The available options are Low, Medium, Quality, and High.
- **Version:** Lets you specify the version of the barcode.
- **Mask:** Lets you select the pattern used for masking barcode. In order to make sure QRCode being successfully read, mask process is required to balance brightness. The options available are Auto, Mask000, Mask001,

Mask010, Mask011, Mask100, Mask101, Mask110, and Mask111.

- **Connection:** Lets you select whether connection is used for the barcode. It takes the value True of False.
- **ConnectionNumber:** Lets you specify the connection number for the barcode. It takes the integer value ranging from 0 to 15.

RssExpandedStacked:

- **RowCount:** Lets you specify the number of stacked rows.

> The quiet zones for barcodes can be specified easily by using MarginBottom, MarginLeft, MarginRight, and MarginTop properties in the Property pane of the **C1FlexReportDesigner**.

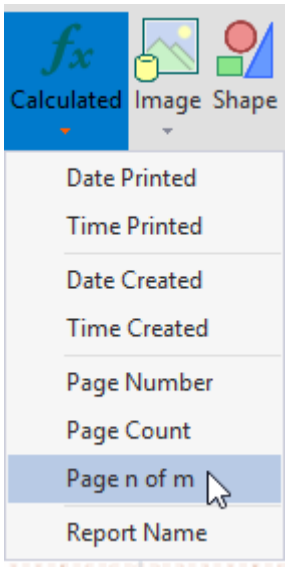
Calculated Field

In order to create fields that do not use raw data, **Calculated** fields are the best option. Calculated fields contain expressions that are evaluated at run-time. They are used to generate those data values that are not stored in the data base.

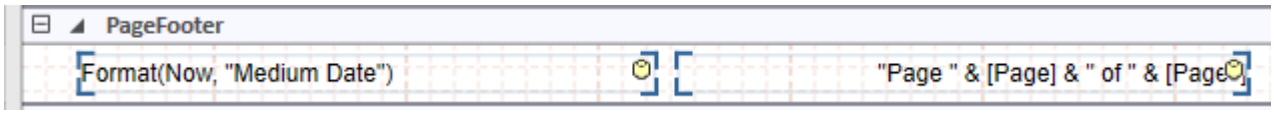
Some predefined expressions are available in the dropdown of Calculated Fields.

To add a predefined Calculated field using FlexReportDesigner application:

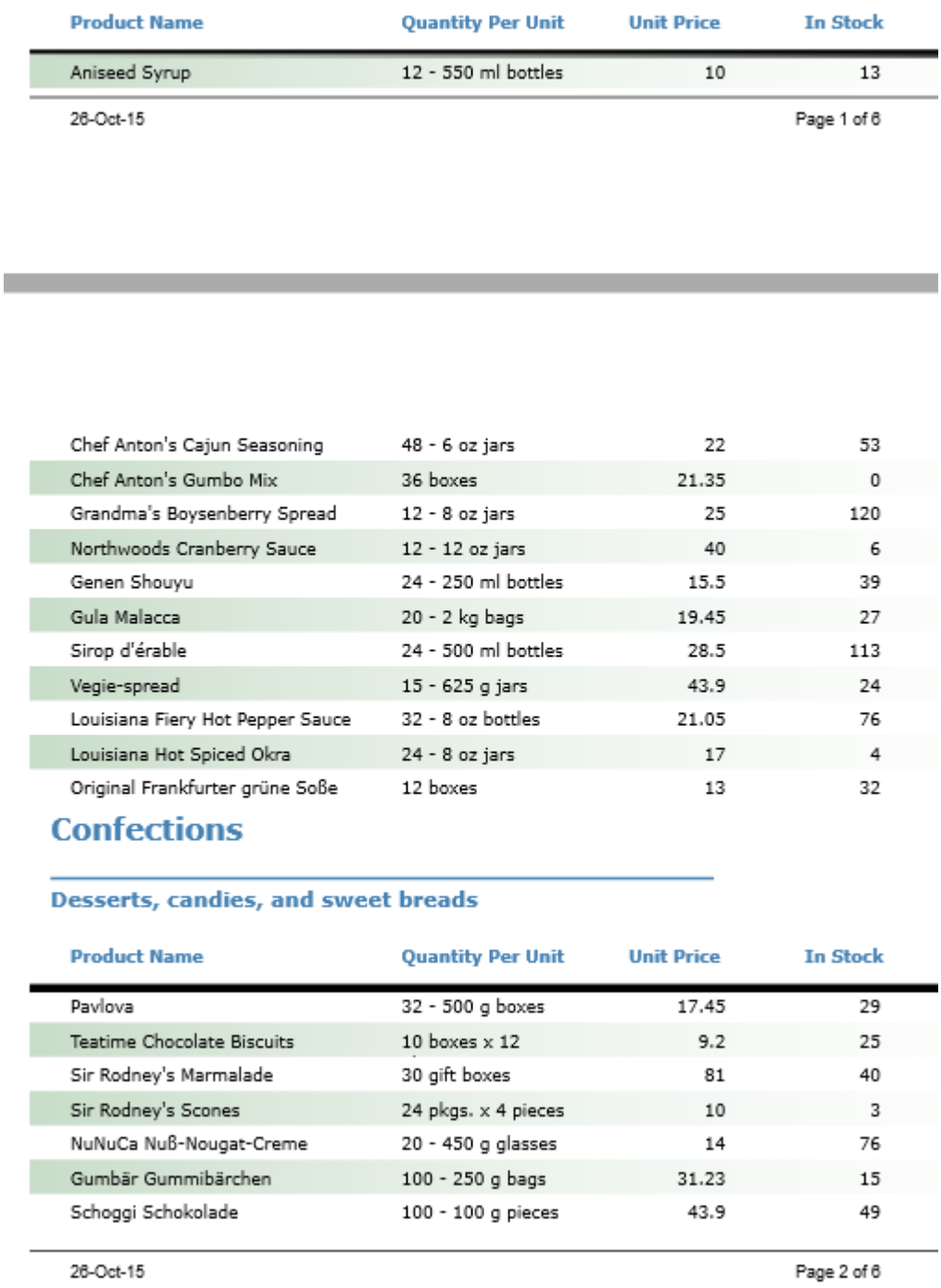
1. Create a new FlexReport in C1FlexReportDesigner.
2. From **Insert** tab, add two **Calculated** fields - 'Date Created' and 'Page n of m', in the PageFooter section:



3. Note the expressions in these calculated fields in the design area.



4. Preview the report.



Product Name	Quantity Per Unit	Unit Price	In Stock
Aniseed Syrup	12 - 550 ml bottles	10	13

26-Oct-15 Page 1 of 6

Chef Anton's Cajun Seasoning	48 - 6 oz jars	22	53
Chef Anton's Gumbo Mix	36 boxes	21.35	0
Grandma's Boysenberry Spread	12 - 8 oz jars	25	120
Northwoods Cranberry Sauce	12 - 12 oz jars	40	6
Genen Shouyu	24 - 250 ml bottles	15.5	39
Gula Malacca	20 - 2 kg bags	19.45	27
Sirop d'érable	24 - 500 ml bottles	28.5	113
Vegie-spread	15 - 625 g jars	43.9	24
Louisiana Fiery Hot Pepper Sauce	32 - 8 oz bottles	21.05	76
Louisiana Hot Spiced Okra	24 - 8 oz jars	17	4
Original Frankfurter grüne Soße	12 boxes	13	32

Confections

Desserts, candies, and sweet breads

Product Name	Quantity Per Unit	Unit Price	In Stock
Pavlova	32 - 500 g boxes	17.45	29
Teatime Chocolate Biscuits	10 boxes x 12	9.2	25
Sir Rodney's Marmalade	30 gift boxes	81	40
Sir Rodney's Scones	24 pkgs. x 4 pieces	10	3
NuNuCa Nuß-Nougat-Creme	20 - 450 g glasses	14	76
Gumbär Gummibärchen	100 - 250 g bags	31.23	15
Schoggi Schokolade	100 - 100 g pieces	43.9	49

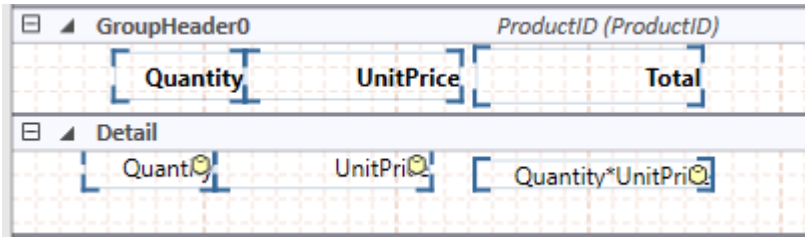
26-Oct-15 Page 2 of 6

To define a Calculated field and add the field in the FlexReportDesigner application:

1. Create a new report.
2. Bind the report with '**Orders Details**' Table.
3. In Group Header, add a **Text** field 'Total' to display label for the total price.
4. Go to **Data** tab, and right-click the Main data source and select **Add Calculated Field**. VBScript Editor appears.
5. In the **VBScript Editor**, write the following expression:

```
Quantity*UnitPrice
```

6. Drop the above mentioned Calculated field as shown.



7. Preview the report.

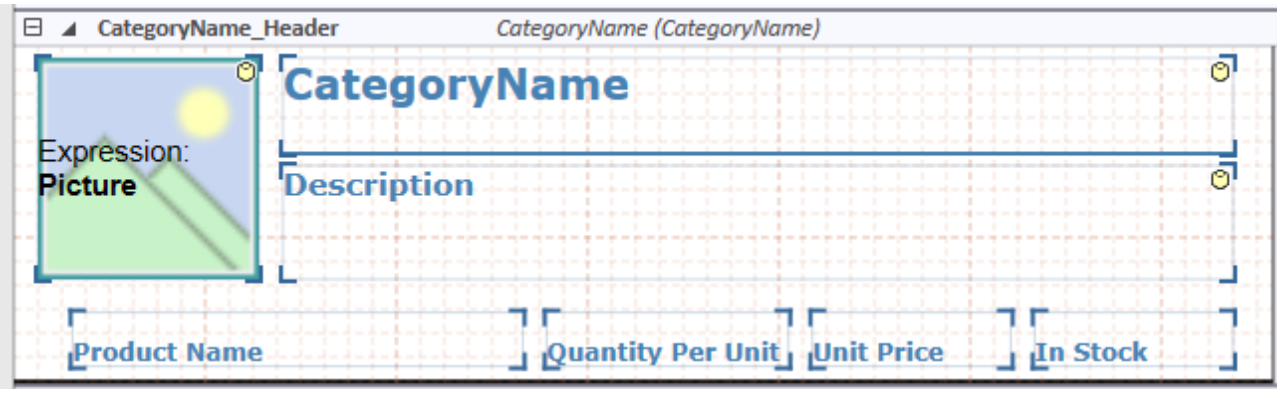
Quantity	UnitPrice	Total
12	10.4	124.8
15	10.4	156
10	10.4	104
5	10.4	52
7	10.4	72.8
14	10.4	145.6
35	10.4	364
10	10.4	104
55	10.4	572
30	10.4	312

Image Field

Image field is used to insert images that are data-dound or static. On clicking Image field, a dialog box appears that prompts you to select an image file to be inserted in the report. A copy of the image you select is made and placed in the same directory as the report file. You must distribute this file with the application unless you embed the report file in the application. When you embed a report file in your application, any unbound picture files are embedded too.

To add a Image field using FlexReportDesigner application:

1. Create a new report in C1FlexReportDesigner.
2. Bind the report with **Products** Table.
3. Create a Group Header in which text field 'Image' for the products will be displayed.
4. Go to the **Insert** tab and click the dropdown on the **Image** field icon.
5. Select 'Picture' and add the Image field to the group header section of the report.



6. Preview the report.

Beverages

Soft drinks, coffees, teas, beers, and ales

Product Name	Quantity Per Unit	Unit Price	In Stock
Chai	10 boxes x 20 bags	18	39
Chang	24 - 12 oz bottles	19	17
Guaraná Fantástica	12 - 355 ml cans	4.5	20
Sasquatch Ale	24 - 12 oz bottles	14	111
Steeleye Stout	24 - 12 oz bottles	18	20
Côte de Blaye	12 - 75 cl bottles	263.5	17
Chartreuse verte	750 cc per bottle	18	69
Ipoh Coffee	16 - 500 g tins	46	17
Laughing Lumberjack Lager	24 - 12 oz bottles	14	52
Outback Lager	24 - 355 ml bottles	15	15
Rhönbräu Klosterbier	24 - 0.5 l bottles	7.75	125
Lakkalikööri	500 ml	18	57

Condiments

Sweet and savory sauces, relishes, spreads, and seasonings

Product Name	Quantity Per Unit	Unit Price	In Stock
Aniseed Syrup	12 - 550 ml bottles	10	13
Chef Anton's Cajun Seasoning	48 - 6 oz jars	22	53

Shape Field

Shape fields are used to display geometric shapes in reports. Lines are often used as separators, rectangles are used to highlight groups of fields or to create tables and grids, and so on.

You can also set rules for defining visibility of a shape in the expressions. For example, if you want Product names to be enclosed in a rectangular shape when the Reorder Level is less than the Units in Stock, you can write following expression in the OnPrint property of the report, to define when to turn on the visibility of shape:

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

This scenario is discussed in detail in the topic Showing or Hiding a Field Depending on a Value.

Subreport Field

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

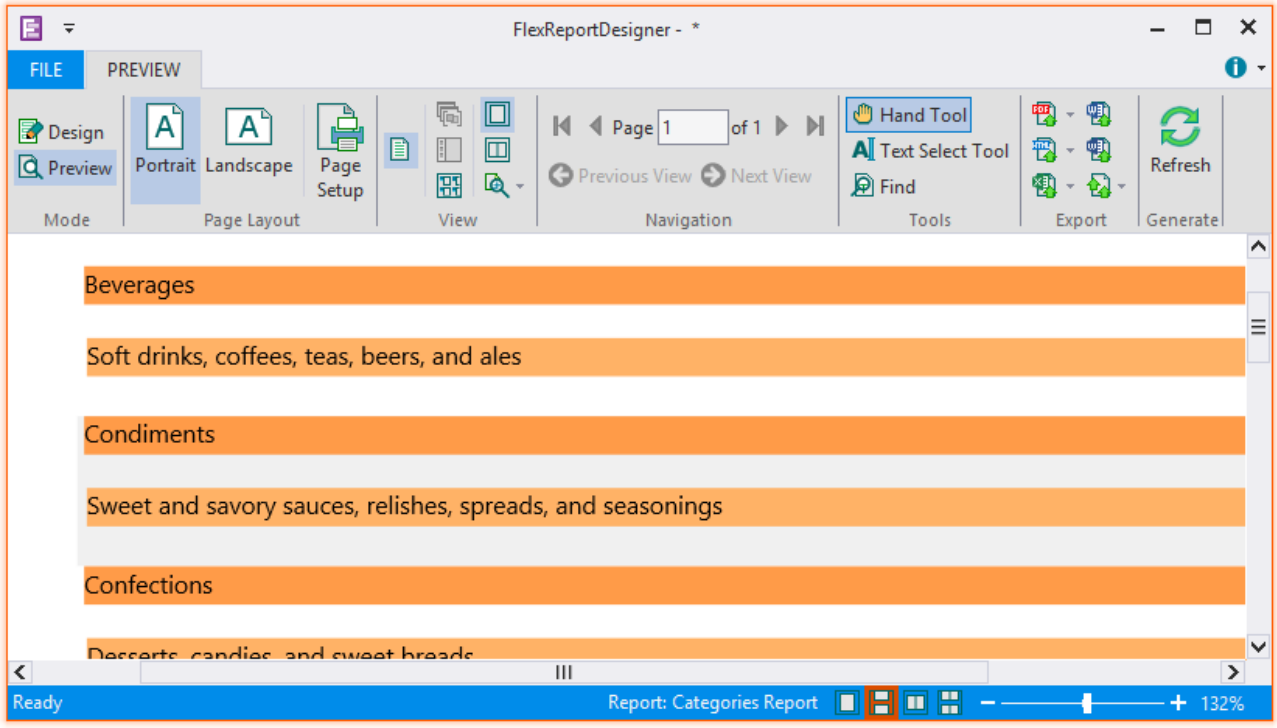
In the following example, the main report contains categories and the subreport in the Detail section contains product details for the current category:

Beverages				
Soft drinks, coffees, teas, beers, and ales				
Chai	10 boxes x 20 bags	18	39	0
Chang	24 - 12 oz bottles	19	17	40
Guaraná Fantástica	12 - 355 ml cans	4.5	20	0
Sasquatch Ale	24 - 12 oz bottles	14	111	0
Steeleye Stout	24 - 12 oz bottles	18	20	0
Côte de Blaye	12 - 75 cl bottles	263.5	17	0
Chartreuse verte	750 cc per bottle	18	69	0
Ipoh Coffee	16 - 500 g tins	46	17	10
Laughing	24 - 12 oz bottles	14	52	0
Outback Lager	24 - 355 ml bottles	15	15	10
Rhönbräu	24 - 0.5 l bottles	7.75	125	0
Lakkalikööri	500 ml	18	57	0
Condiments				
Sweet and savory sauces, relishes, spreads, and seasonings				
Aniseed Syrup	12 - 550 ml bottles	10	13	70
Chef Anton's Cajun	48 - 6 oz jars	22	53	0
Chef Anton's	36 boxes	21.35	0	0
Grandma's	12 - 8 oz jars	25	120	0
Northwoods	12 - 12 oz jars	40	6	0
Genen Shouyu	24 - 250 ml bottles	15.5	39	0
Gula Malacca	20 - 2 kg bags	19.45	27	0
Sirop d'érable	24 - 500 ml bottles	28.5	113	0
Vegie-spread	15 - 625 g jars	43.9	24	0
Louisiana Fiery Hot	32 - 8 oz bottles	21.05	76	0
Louisiana Hot	24 - 8 oz jars	17	4	100
Original Frankfurter	12 boxes	13	32	0

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

Step 1: Create the master report

1. Create a basic report definition using the **FlexReport Wizard**.
    1. Select the **Categories** table from the Northwind database (**C1NWind.mdb** located in the ComponentOne Samples\Common folder).
    2. Include the **CategoryName** and **Description** fields in the report.
2. In the **C1FlexReportDesigner** application, click the **Design** button to begin editing the report.
3. Set the Page Header and Header section's **Visible** property to **False**.
4. In the Detail section, select the **DescriptionCtl** and move it directly below the **CategoryNameCtl**.
5. Use the Properties window to change the Appearance settings (Background).
6. Select the **Preview** button, the Categories report should now look similar to the following image:



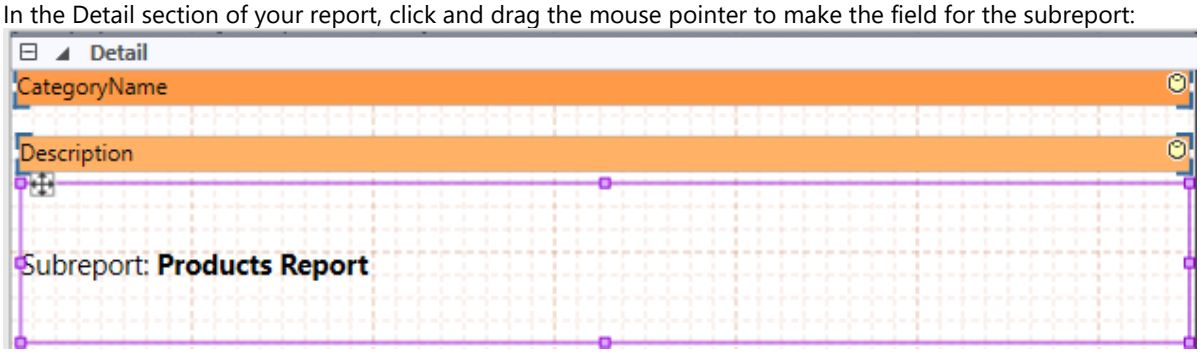
Step 2: Create the detail report

1. In the **C1FlexReportDesigner** application, click the **New Report** button to create a basic report definition using the **FlexReport Wizard**.
    1. Select the **Products** table from the C1NWind database.
    2. Include the following fields in the report: **ProductName**, **QuantityPerUnit**, **UnitPrice**, **UnitsInStock**, and **UnitsOnOrder**.
2. In the Report Designer, click the **Design** button to begin editing the report.
    1. Set the Page Header and Header section's **Visible** property to **False**.
    2. In the Detail section, arrange the controls so that they are aligned with the heading labels. Use the Properties window to change the Appearance settings.

Step 3: Create the Subreport field

The **C1FlexReportDesigner** application now has two separate reports**, Categories Report** and **Products Report**. The next step is to create a subreport:

1. From the Reports list in the Designer, select **Categories Report** (master report).
2. In design mode, from **Fields** group in **Insert** tab, click the **Subreport** icon and select **Products Report** from the drop-down menu.
3. In the Detail section of your report, click and drag the mouse pointer to make the field for the subreport:

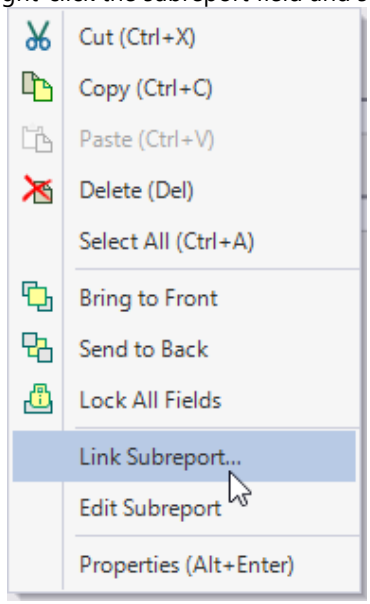


Step 4: Link the Subreport to the master report

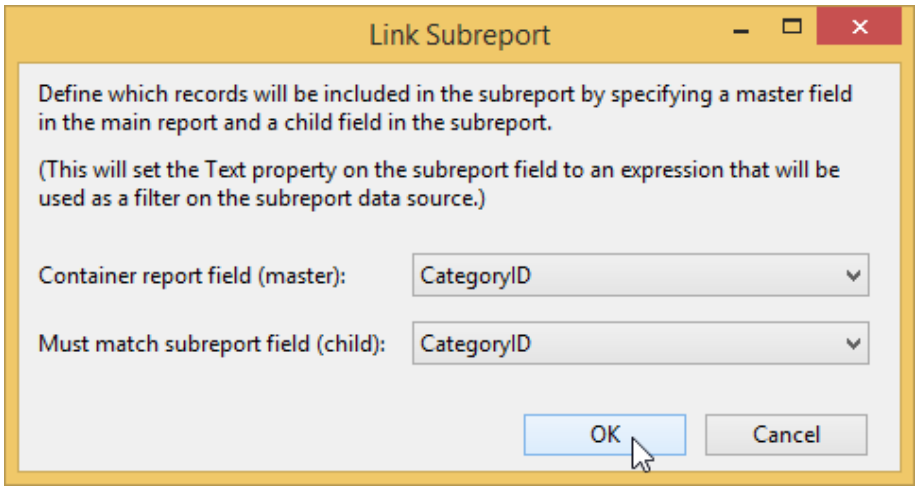
The master-detail relationship is controlled by the **SubreportFilter** property of the subreport field. This property should contain an expression that evaluates into a filter condition that can be applied to the subreport data source.

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

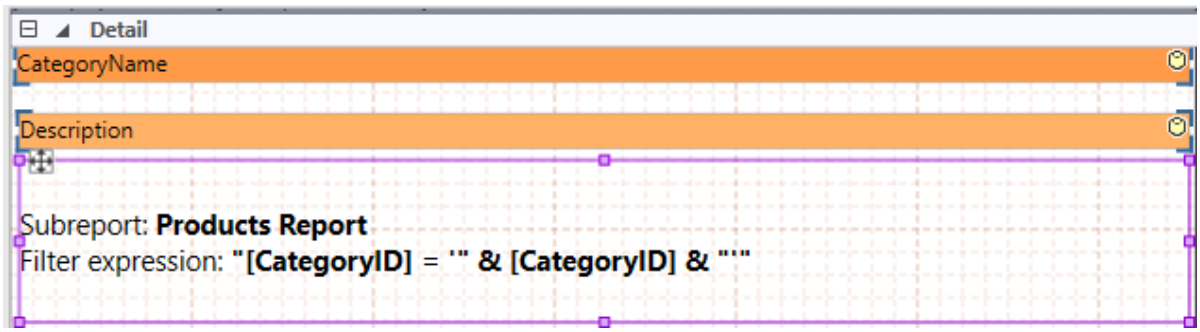
1. Right-click the subreport field and select **Link Subreport** from the menu.



Link Subreport dialog box appears that allows you to select which fields should be linked.



The Subreport field in design area now appears as follows:



2. Once you make a selection and click **OK**, the Report Designer builds the link expression and assigns it to the

SubreportFilter property of the subreport field. In this case, the expression is:

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

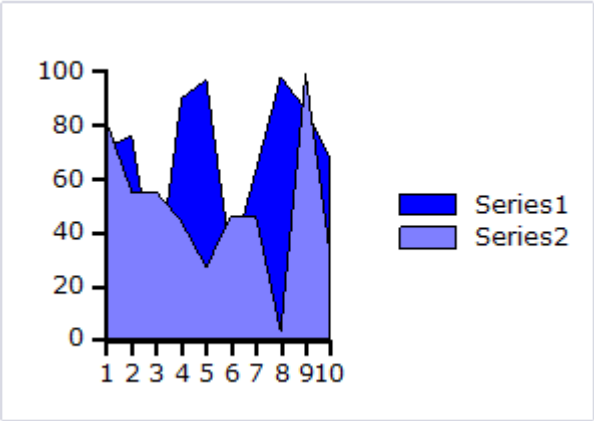
Legacy Chart Field

The **Legacy Chart** field in FlexReport is now provided built-in with FlexReportDesigner. It uses **C1.Win.C1Chart** and **C1.Win.C1Chart3D** assemblies to render chart fields consisting of 2D and 3D chart types, respectively. For more information on 2DCharts and 3DCharts, see 2DChart documentation and 3DChart documentation, respectively.

Chart Types

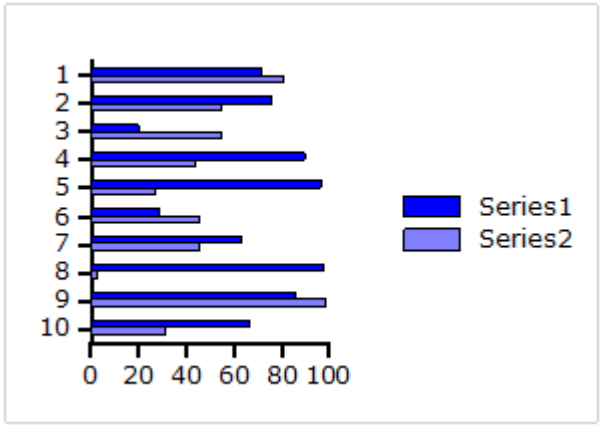
The **Chart Field** in **FlexReport** allows you to insert various types of charts using **Chart.Chart2DGroup.ChartType** and **Chart.Chart3DGroup.ChartType**. The chart types that are supported in **C1FlexReport** are **-** Area, Bar (horizontal bars), Column (vertical columns), Doughnut, Scatter (X-Y values), Line, Pie, Step, Stock, Histogram, Radar, Polar, and 3D charts-Cone, Cylinder, and Pyramid. The chart types can be easily selected using the **ChartType** property in the Properties window of the **C1FlexReportDesigner**.

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

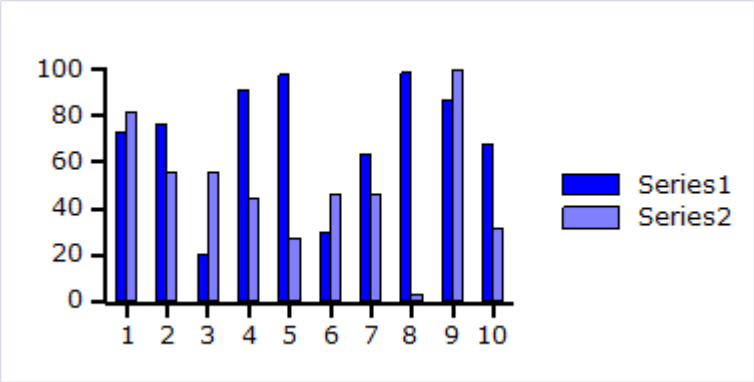


Bar and Column charts: A Bar chart or a Column chart represents each series in the form of bars of the same color and width, whose length is determined by its value. Each new series is plotted in the form of bars next to the bars of the preceding series. A Bar or Column chart draws each series as a bar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

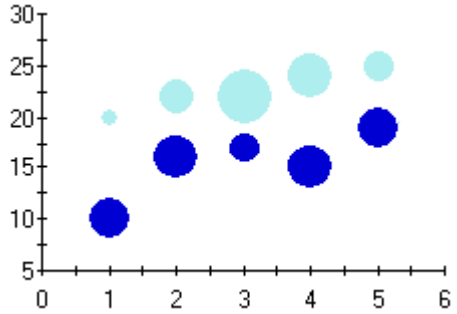
The following image represents a **Bar** chart:



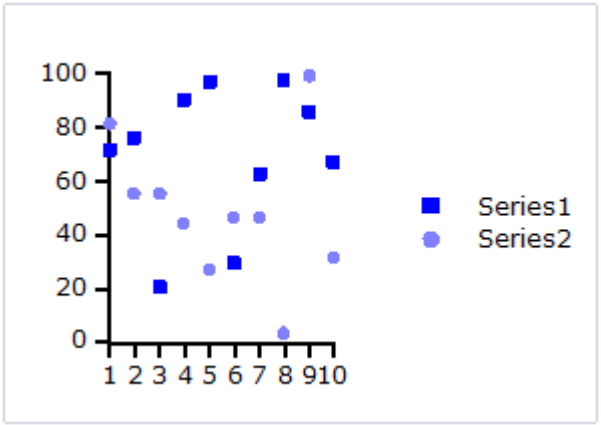
The following image represents a **Column** chart:



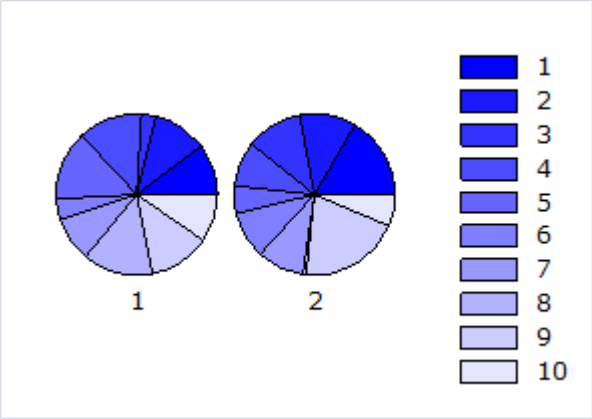
Bubble Chart: A Bubble chart combines two independent values to supply both the point y value and the point sizes. Bubble charts are used to represent an additional data value at each point by changing its size. The Y array elements determine the Cartesian position (as in a XY-Plot chart), and the Y1 element values determine the size of the bubble at each point. The size of the points can be encoded according to area or diameter.



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



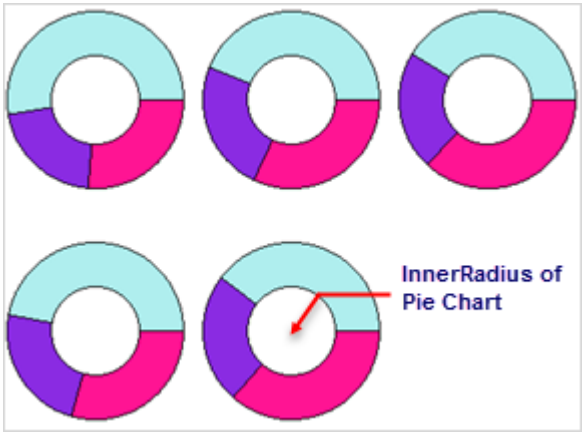
Pie chart: A Pie chart draws each series as a slice in a pie. The number of pies is the number of points in the data. Each pie displays the nth data point in each series. You can also customize Pie charts for displaying legends and labels.



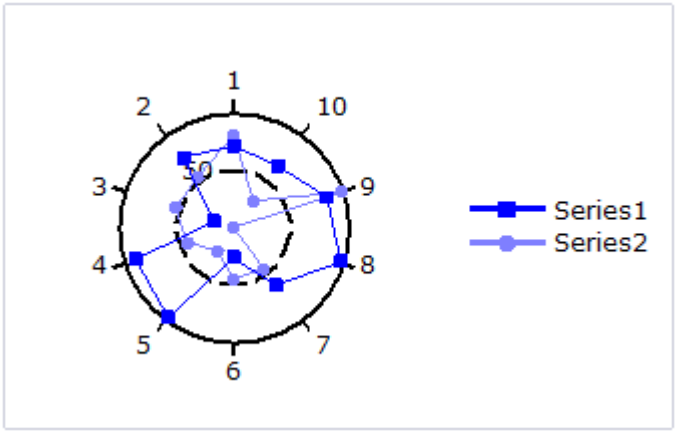
Doughnut Chart:

A doughnut chart is a pie chart with a non-zero radius and is identical in function to a pie chart, but can be used to increase aesthetic appeal, particularly when shown with 3D effects. As with all pie charts, each doughnut shows each series as a fraction of the whole at each data point. If multiple data points are specified, then multiple doughnuts appear in the chart.

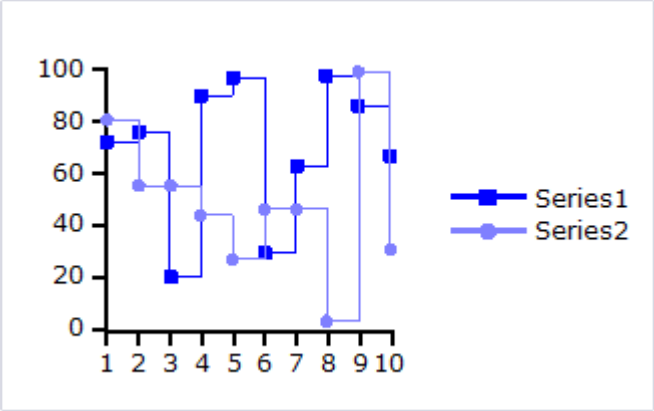
A doughnut chart can be created by setting the **InnerRadius** property of a pie chart to a non-zero value. The InnerRadius value represents the percentage of the full pie radius. The InnerRadius property can be accessed in the pie object of each Chart group.



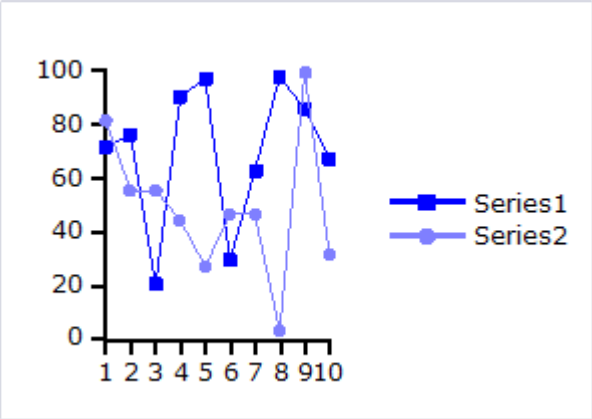
Radar chart: A Radar chart draws the y value in each data set along a radar line (the x value is ignored except for labels). If the data has n unique points, then the chart plane is divided into n equal angle segments, and a radar line is drawn (representing each point) at n/360 degree increments. By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



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



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

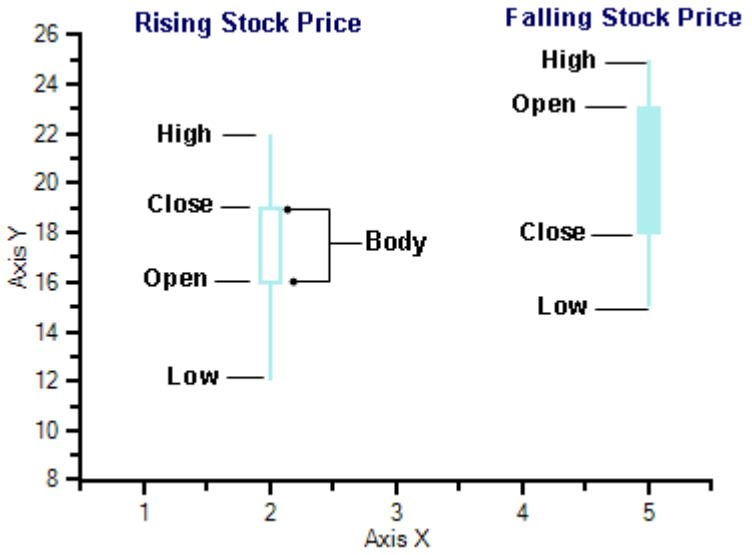


Stock Charts: Stock charts used in financial applications to show the opening, closing, high and low prices of a given stock. The types of stock charts are as follows:

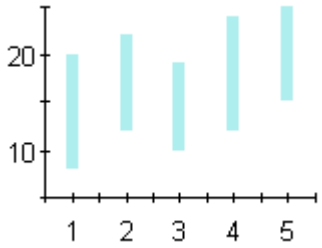
- **Candle:** A Candle chart is a special type of HiLoOpenClose chart that is used to show the relationship between the open and close as well as the high and low. Like, HiLoOpenClose charts, Candle charts use the same price data (high, low, open, and close values) except they include a thick candle-like body that uses the color and size of the body to reveal additional information about the relationship between the open and close values. For example, long transparent candles show buying pressure and long filled candles show selling pressure.

  The Candle chart is made up of the following elements: candle, wick, and tail. The candle or the body (the solid bar between the opening and closing values) represents the change in stock price from opening to closing. The thin lines, wick and tail, above and below the candle depict the high/low range. A hollow candle or transparent candle indicates a rising stock price (close was higher than open). In a hollow candle, the bottom of the body represents the opening price and the top of the body represents the closing price. A filled candle indicates a falling stock price (open was higher than close). In a filled candle the top of the body represents the opening price and the bottom of the body represents the closing price.

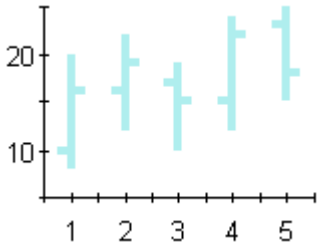
  C1Chart creates the Candle chart with using the Y value for the High, Y1 for the low, Y2 for the open, and Y3 for the close. C1Chart automatically fills the falling candle with the value of the line color.



- **HiLo:** A HiLo chart combines two independent values to supply high and low data for each point in a series. HiLo charts are used primarily in financial applications to show the high and low price for a given stock. The elements of the Y and Y1 arrays in each series of a HiLo chart represent the "high" value, and the "low" value.

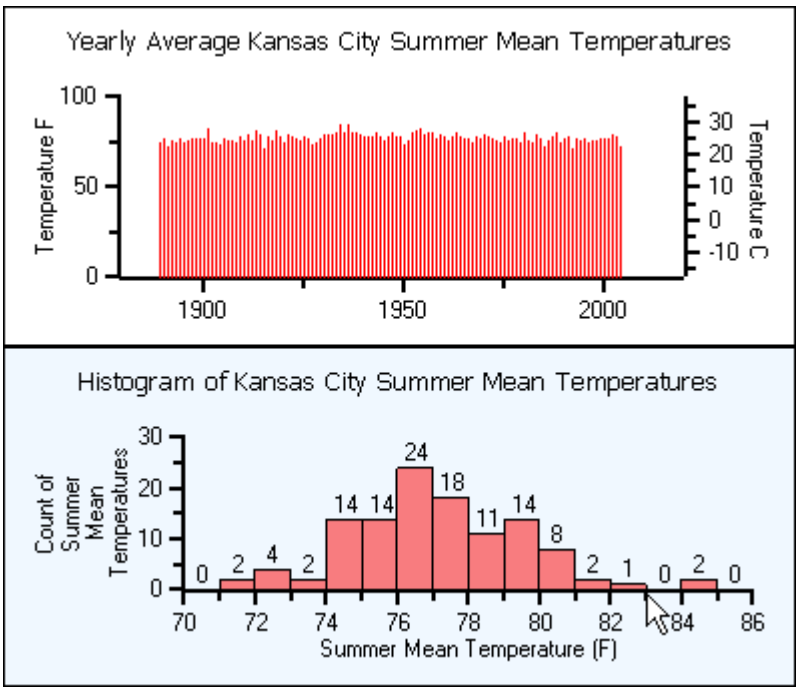


- **HiLoOpenClose:** HiLoOpenClose charts are similar to HiLo charts except that they combine four independent values to supply high, low, open, and close data for a point in a series. In addition to showing the high and low value of a stock, the Y2 and Y3 array elements represent the stock's opening and closing price, respectively.



Histogram chart: A Histogram chart takes a collection of raw data values and plots the frequency distribution. It is frequently used with grouped data, which is generated by measuring a collection of raw data and plotting the number of data values that fall within defined intervals. Note that raw values are not used to generate data for a histogram, but are used to generate a frequency instead. While showing similarities to bar charts, it is important to note that histograms are used with quantitative variables whereas bar charts are commonly used with qualitative variables.

While the histogram and bar charts' appearances relate, their functionality does not. A bar chart is created from data points whereas a Histogram is created from the frequency distribution of the data. The charts following illustrate the difference between a bar chart and a histogram chart. Both of the charts use exactly the same Y data. The bar chart (top) shows each average mean temperature for each year in which it occurred. The histogram chart (bottom) using the same input temperature data automatically tabulates the number of temperatures that fall within each interval and draws the resulting histogram. For convenience, chart labels with the count in each interval have been added at the top of each interval.

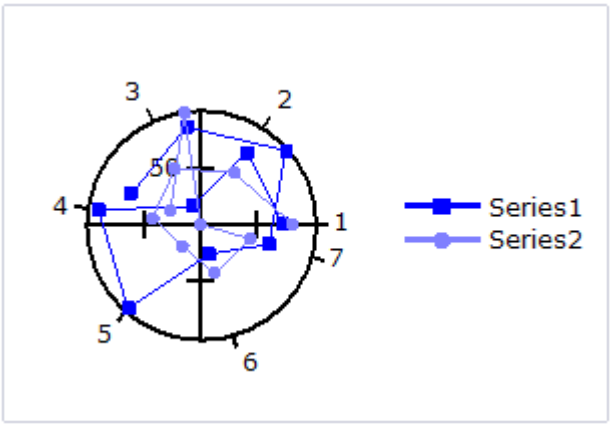


A histogram is beneficial for pinpointing prominent features of the distribution of data for a quantitative variable. The important features for a quantitative variable include the following:

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

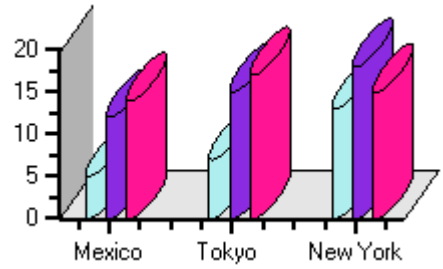
To select the chart type as Histogram, go to **ChartGroups|Group0** and set the **ChartType** to Histogram.

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

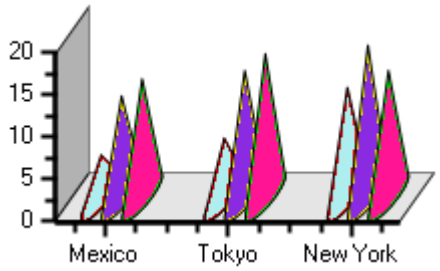


3D Charts

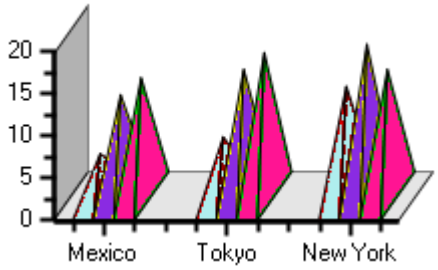
- **Cylinder chart:** A Cylinder chart is a variation of the Bar and Column charts. It represents the bars or columns as cylinders. The Cylinder chart creates long circular boxes of the same base on both ends. Like all bar and column charts, the Cylinder bar chart is appropriate for comparing individual items or groups of items.



- **Cone chart:** A Cone chart is a variation of the 3D Bar and Column charts. It represents the bars or columns as cones. The cone chart essentially is a rotated triangle. It has a flat circular base and one curved side topped by a higher point.



- **Pyramid chart:** A Pyramid chart is a variation the 3D-Bar and Column charts. It represents the bars or columns as pyramids. The Pyramid chart is similar to the cone chart except for their base. Pyramid charts are often used for geographical purposes.



Design Time Support

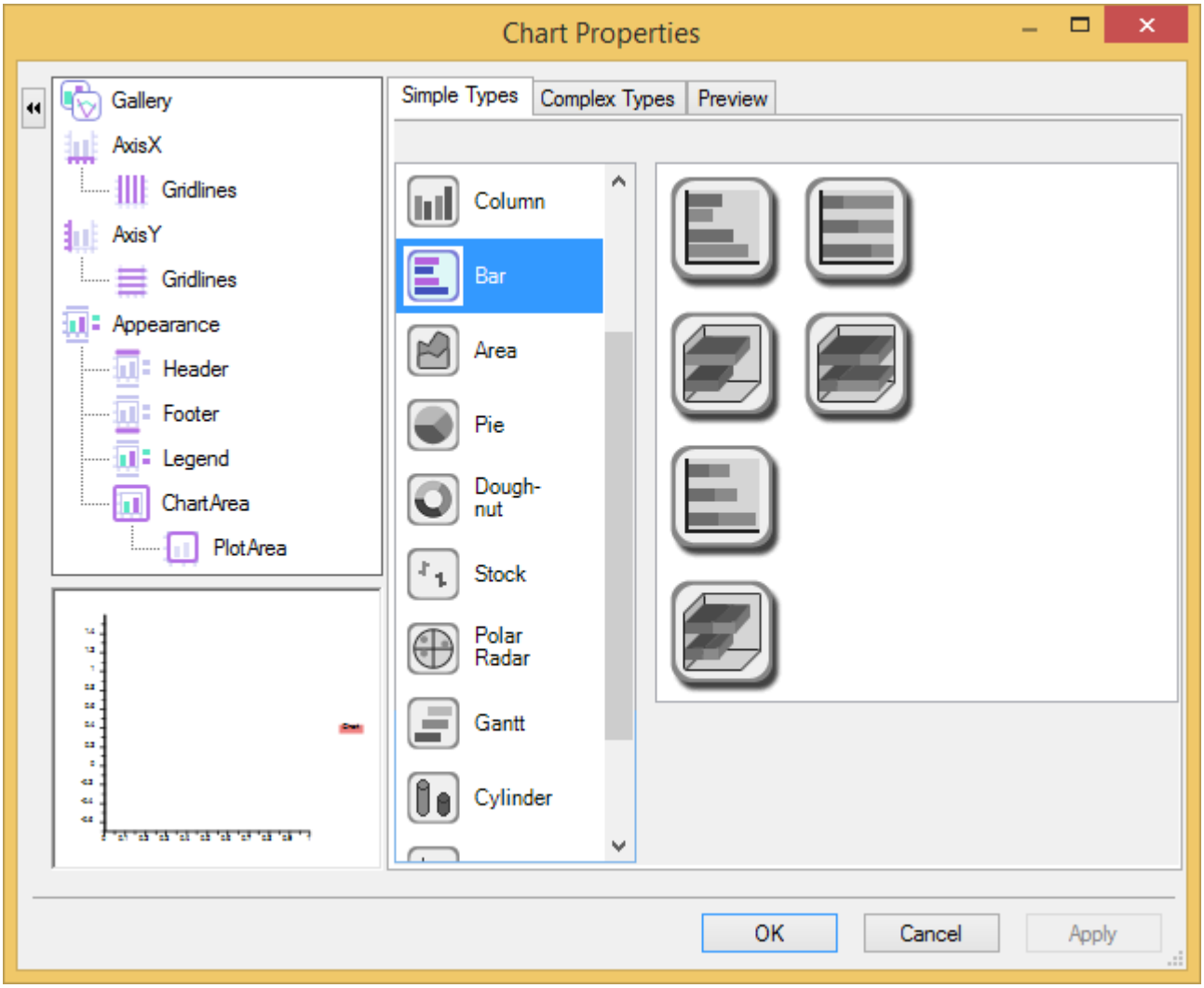
The **Chart** field in **FlexReport** provides design-time support through design-time editors and collection editors that simplify working with the charts.

The design-time editors for **Chart** field are as follows:

- Chart Properties
- Chart Data Source
- Chart Visual Effects

These editors can be invoked by right-clicking the chart field and selecting the required editor.

- Chart Properties

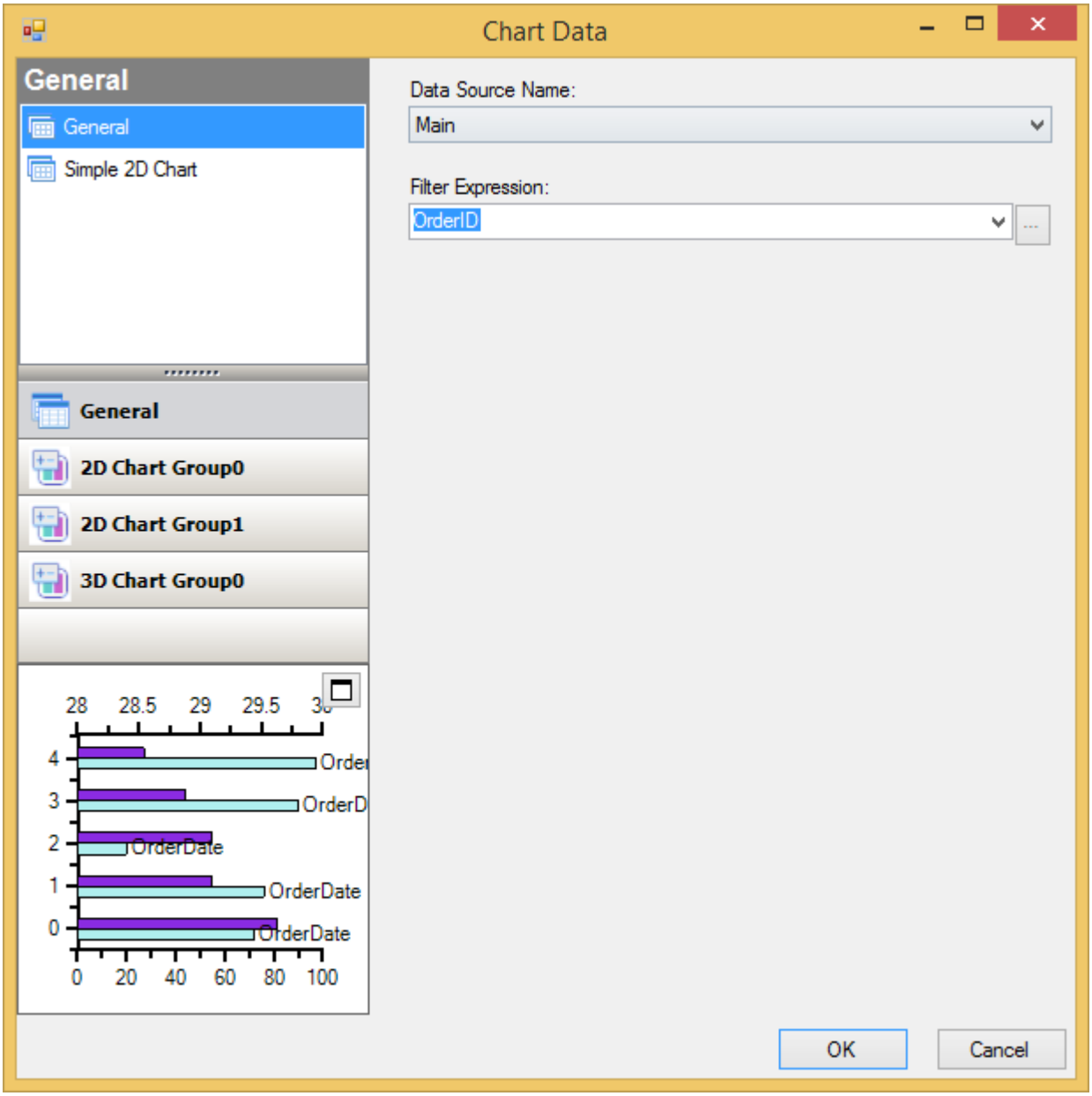


It provides an easy and interactive way to create and modify a new or existing chart. It contains the options to select the type of chart and . However, it also includes additional property settings for the x and y axis along with appearance settings for the header, footer, legend, chart area, and plot area of the chart. The Chart Properties dialog box provides more options to address specific details with the design of the chart you are developing. It consists of following elements:

- **Gallery Item:** The Gallery item in the left pane of the Chart Properties dialog box provides options for choosing a chart type and/or a sub-type of a chart. To see a description of all chart type selections, see Chart Types. You can choose from a variety of simple chart types or you could click on complex types to add more functionality to your chart.
- **Simple Types Tab:** In the Simple Types tab you can choose from one of many simple chart types and then you can select a specialized chart located in list box next to the simple chart types.
- **Complex Types Tab:** In the Complex Types tab you can specify whether you want to chart one or two chart groups. Also, you can select the type of chart you would like to create in the drop-down box for each group. For each group, you have the option to make the groups stacked and/or 3D.

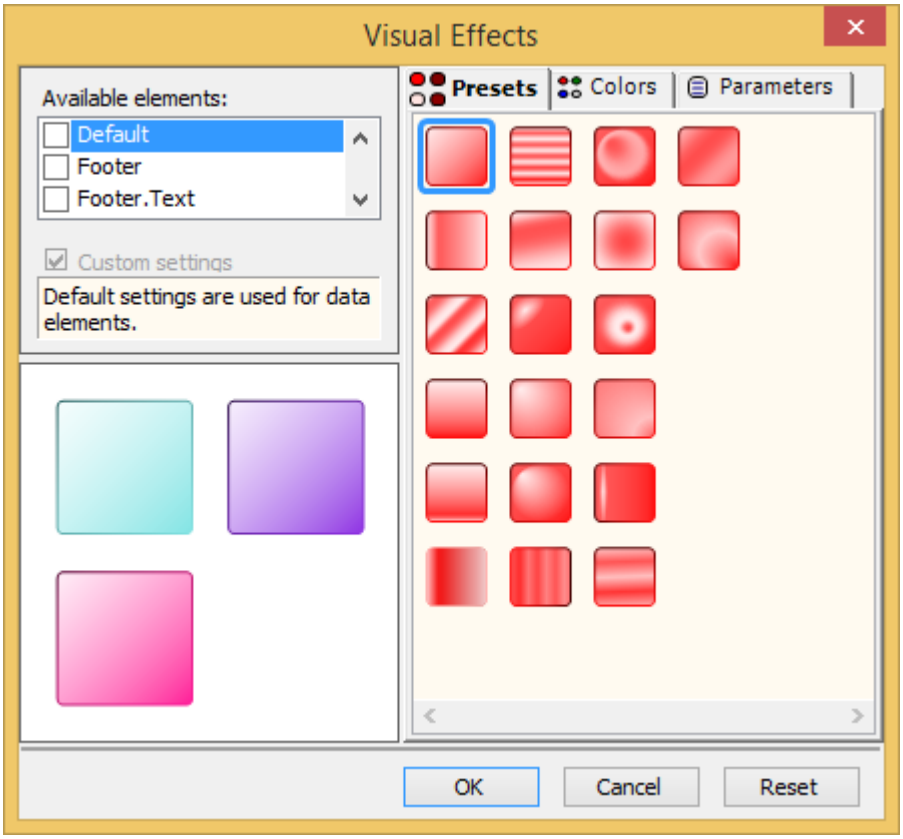
Note: If you don't select a chart type for data [Group1] then the elements for data [Group1] will not appear in the list box in the left pane of the Chart Properties dialog box.

- Chart Data Source



The Chart Data editor lets you bind the chart by setting data through Chart Data-add or remove data series in your chart, specify the label and color of each data series, and stack your data by checking the Show stacked data check box.

- Chart Visual Effects

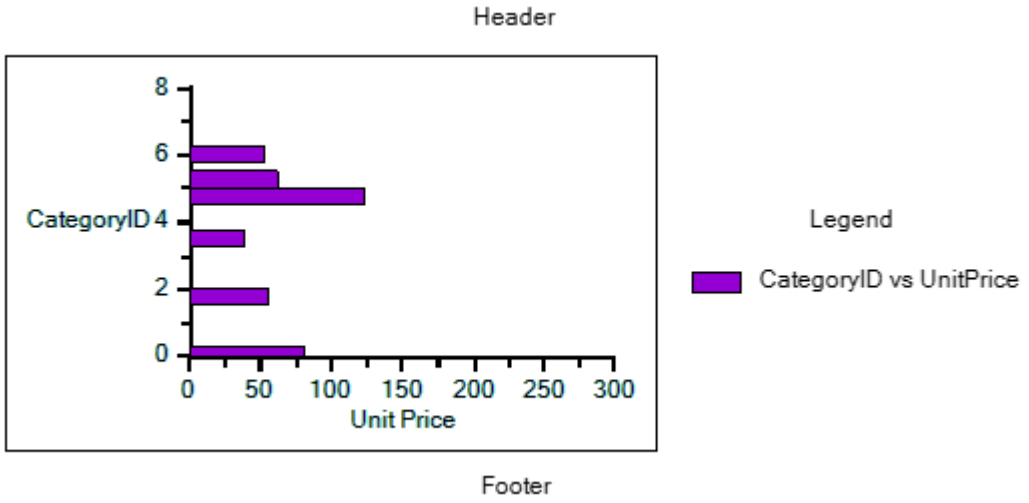


Visual Effects is a tool used for visually enhancing the Chart2D control's elements such as the data series, header, and footer. Any existing project can use the new features provided by this tool. The chart's appearance can dramatically improve in a few simple steps using the Visual Effects designer.

The collection editors for **Chart** field are as follows:

- **ValueLabel Collection Editor:** This collection editor can be assessed through the Property pane-**ChartArea|AxisX|ValueLabels**. For more information, see the documentation on ValueLabel CollectionEditor.
- **ChartDataSeries Collection Editor:** This collection editor can be assessed through the Property pane-**ChartGroups|Group0|ChartData|CategoryGroups**. For more information, see the documentation on ChartDataSeries CollectionEditor.
- **ChartLabel Collection Editor:** This collection editor can be assessed through the Property pane-**ChartLabels|DefaultLabelStyle|LabelsCollection**. For more information, see the documentation on LabelCollection Editor.
- **ChartVisualEffectsStyle Collection Editor**: This collection editor can be assessed through the Property pane-**VisualEffects|Styles**. For more information, see the documentation on VisualEffectsStyle Collection Editor.

Plotting Data in Data-Bound Charts



The steps to plot a simple 2D chart are as follows:

1. Open the **C1FlexReportDesigner** application.
2. Create a new report by navigating through the C1FlexReport wizard or open an existing report definition.
3. Add **Legacy Chart** field to the Detail section.

   > **Note:** To add **Legacy Chart** field to your report, you need to add it to the **INSERT** tab. In order to do that, go to **File|Options** in the designer and select **Show "Legacy Chart" button on the INSERT tab**.

4. Right-click the chart field, select **Chart Properties>Simple Types>Bar**.
5. Right-click the chart field and select **Chart Data Source**.
6. Click **Simple 2D Chart** and check the **Generate simple 2D chart** checkbox.
7. Specify **X** and **Y** values of the chart.

For example, the chart shown is created using C1NWind.mdb database and Products table, with following properties set:

- **Chart Type** as 'Bar'
- **X** and **Y** values of the chart as 'CategoryID' and 'UnitPrice', respectively.

Add series label as follows:

1. Select **2D Chart Group0**, and then **Series Values**.
2. Specify **Series Label** as ''CategoryID vs UnitPrice'.

Add legend as follows:

1. Go to the Properties window.
2. In the **Legend|Text**, set the legend name as 'Legend' and **Legend|Visible** to True.

Add Titles to the chart as follows:

1. Go to Properties window.
2. Select **Titles|Header**, set the header text as 'Header' and **Titles|Header|Visible** to True.
3. Select **Titles|Footer**, set the footer text as 'Footer' to add header and footer to the chart.

You can also add axis titles in the chart, add back color and border to chart area, and more. For more information on using properties for customizing charts, see 2DChart documentation.

Three dimensional data can be represented in the form of 3D Surface, 3D Bar, and many more by using 3D Chart types. To plot 3D charts from 2D charts, you can change the VisualizationType property to Chart3D. For more information on plotting 3D charts, see 3DChart documentation.

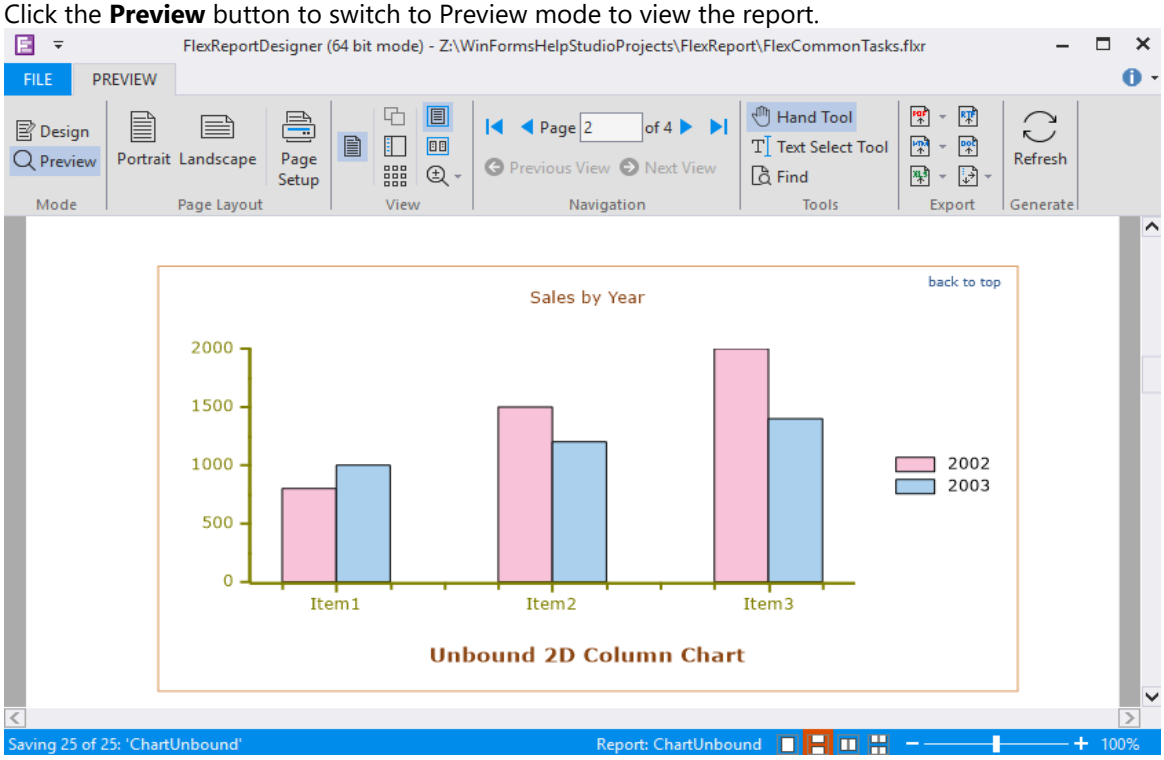
Plotting Data in Unbound Charts

FlexReport allows you to add data to the charts without binding the charts to a data source. You can plot series on chart and display it by setting an array of X and Y values directly in unbound mode. Let us add an unbound 2D chart by performing the following steps:

1. In the **C1FlexReportDesigner** application, create a new report by navigating through the **C1FlexReport wizard** or open an existing report definition.
2. Add **Chart** field to the **Detail** section.

   > **Note:** To add **Legacy Chart** field to your report, you need to add it to the **INSERT** tab. In order to do that, go to **File|Options** in the designer and select **Show "Legacy Chart" button on the INSERT tab**.

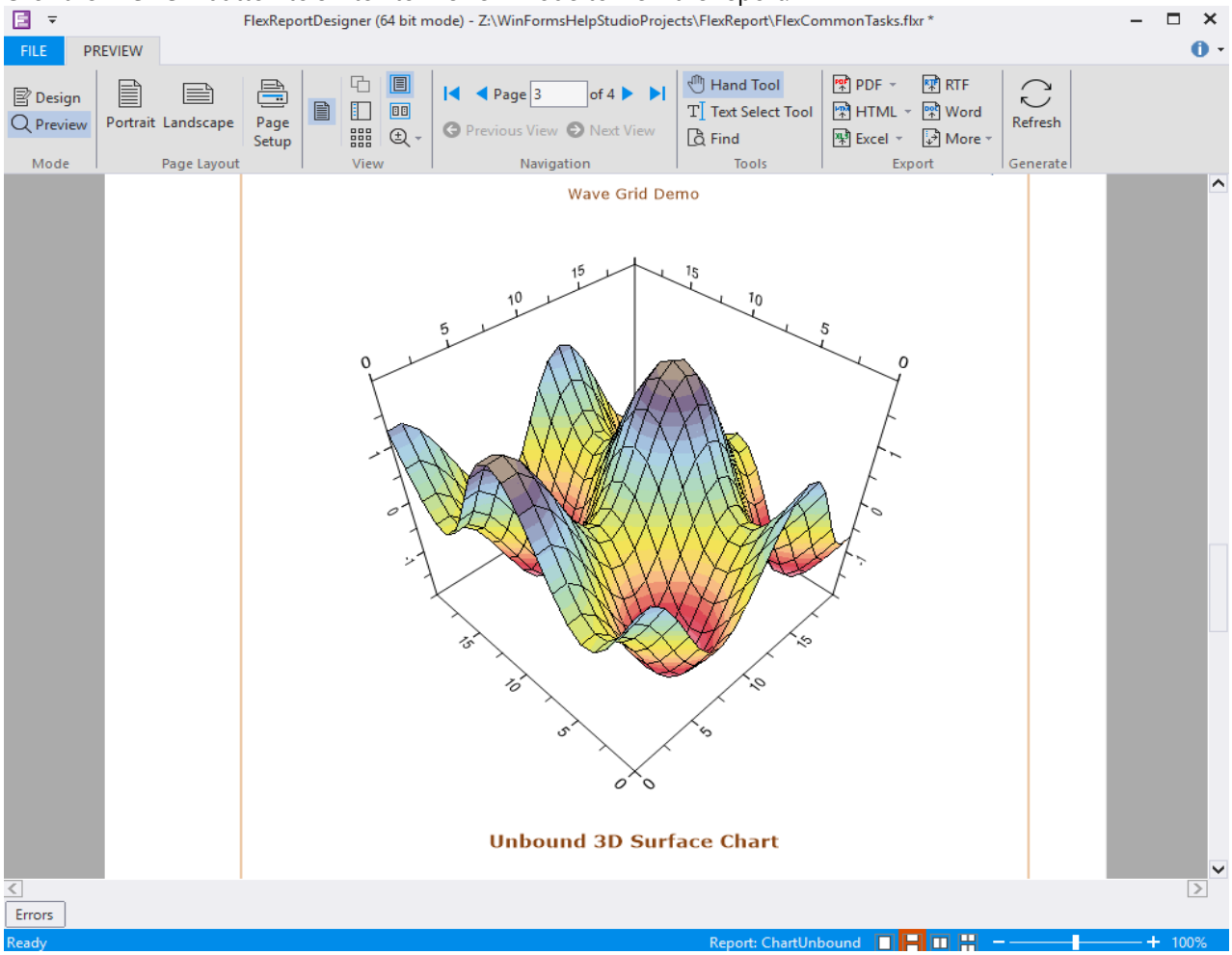
3. In **Properties** window, expand **ChartGroups|Group0|ChartType** and select a type of chart from the ChartType dropdown. In our case, we have selected Bar chart type.
4. Navigate to **ChartGroups|Group0|ChartData** and click the ellipsis button next to **UnboundSeriesList** property.
5. In the **ChartUnboundDataSeries Collection Editor**, add any number of series you want to plot in chart. Here, we have added two series.
6. Select one of the series and click the ellipsis button next to **SeriesData** property.
7. In the editor for series data, enter your data.
8. Click **OK** to close the editors.
9. Click the **Preview** button to switch to Preview mode to view the report.



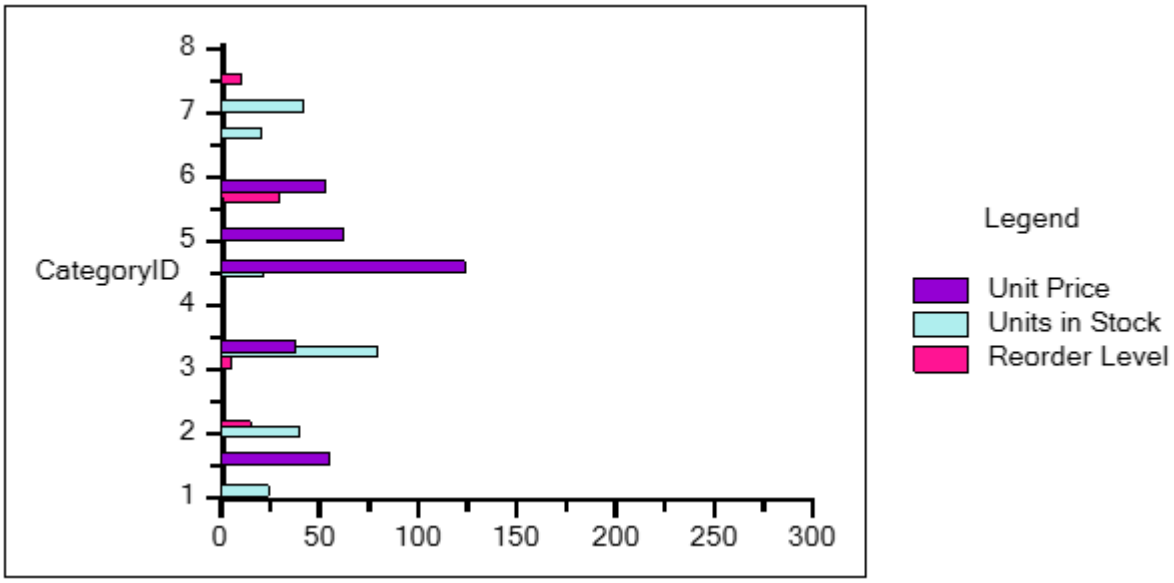
For 3D unbound chart, perform the following steps:

1. Add **Chart** field to the **Detail** section.
2. In **Properties** window, expand **ChartGroups|Group0|ChartType** and select a type of chart from the ChartType dropdown. In our case, we have selected Surface chart type.
3. Navigate to **ChartGroups|Group0|ChartData|UnboundData|GridSet**.
4. Set the **ColumnCount** and **RowCount** properties to the desired value, if required. In our case, both **ColumnCount** and **RowCount** properties are set to 0.
5. Click the ellipsis button next to the **GridData** property.
6. In **Single Array Editor**, enter data and close it.

7. Click the **Preview** button to switch to Preview mode to view the report.



Charts with Multiple Series



Sometimes you need to create charts that have multiple data sets or multiple series. To create such charts, the values for each series are required to be set.

Consider an example of Products table from C1Nwind.mdb database, where you want to view Unit Price, Units in Stock, and Reorder Level corresponding to the Category IDs. The steps to achieve this scenario are as follows:

1. Right-click the **Chart** field and select **Chart Data Source** .
2. Click **2DChartGroup0** and then click **Series Values**.
3. Click the plus glyph and specify **Series Label** of your choice and **Values** from the drop-down list. You can also add colors to the series from the **Fill** option in **Style** tab. In this case, the data specified for our multiple-series chart is as follows:

Series Label	Values	Style | Fill
Unit Price	UnitPrice	Blue Voilet
Units in Stock	UnitsinStock	Pale Turquoise
Reorder Level	ReorderLevel	Deep Pink

To create Legend:

1. Right-click **Legacy Chart** field.
2. Select **Chart Properties**.
3. Navigate to **Appearance | Legend**.
4. Select the **Visible** checkbox and write “Legend” as **Text**.

Charts in Grouped Reports

FlexReport for WinForms allows you to create reports with multiple groups. For example, instead of listing all products in a single flat report, you could group products by category. Each group has a header and a footer section that allow you to display information about the group, including titles and subtotals, for example.

If you add a chart to a group header, the chart will display only the data for the current group. By contrast, adding a chart to the report header or footer would include all the data in the report.

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

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

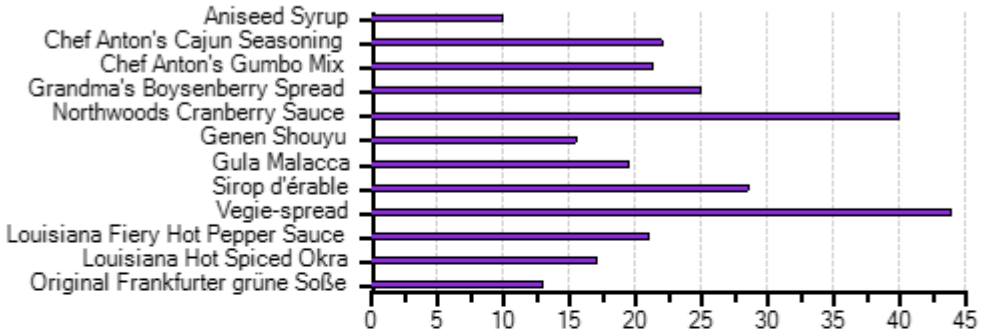
Report Footer section

Continuing with the example mentioned above, lets visualize the report data for each category through chart.

Add a **Legacy Chart** field to the Group Header section (Category Name) and create the chart as explained in the topic Plotting Data in Charts. Specify **X** and **Y** values of the chart as ProductName and UnitPrice and specify **Value** in **Series Values** as UnitPrice.

The images below show screenshots of the report described above with the group headers, the charts they contain, and a few detail records to illustrate:

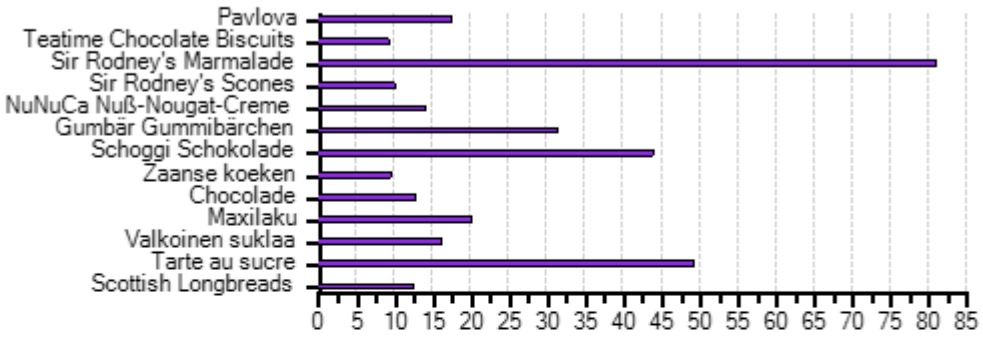
Condiments



Product Name	Quantity Per Unit	Unit Price
Aniseed Syrup	12 - 550 ml bottles	$10.00
Chef Anton's Cajun Seasoning	48 - 6 oz jars	$22.00
Chef Anton's Gumbo Mix	36 boxes	$21.35
Grandma's Boysenberry Spread	12 - 8 oz jars	$25.00
Northwoods Cranberry Sauce	12 - 12 oz jars	$40.00
Genen Shouyu	24 - 250 ml bottles	$15.50
Gula Malacca	20 - 2 kg bags	$19.45
Sirop d'érable	24 - 500 ml bottles	$28.50
Vegie-spread	15 - 625 g jars	$43.90
Louisiana Fiery Hot Pepper Sauce	32 - 8 oz bottles	$21.05
Louisiana Hot Spiced Okra	24 - 8 oz jars	$17.00
Original Frankfurter grüne Soße	12 boxes	$13.00

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

Confections



Product Name	Quantity Per Unit	Unit Price
Pavlova	32 - 500 g boxes	$17.45
Teatime Chocolate Biscuits	10 boxes x 12 pieces	$9.20
Sir Rodney's Marmalade	30 gift boxes	$81.00
Sir Rodney's Scones	24 pkgs. x 4 pieces	$10.00
NuNuCa Nuß-Nougat-Creme	20 - 450 g glasses	$14.00
Gumbär Gummibärchen	100 - 250 g bags	$31.23
Schoggi Schokolade	100 - 100 g pieces	$43.90
Zaanse koeken	10 - 4 oz boxes	$9.50
Chocolade	10 pkgs.	$12.75
Maxilaku	24 - 50 g pkgs.	$20.00
Valkoinen suklaa	12 - 100 g bars	$16.25
Tarte au sucre	48 pies	$49.30
Scottish Longbreads	10 boxes x 8 pieces	$12.50

Adding FlexReport Custom Fields

The custom fields in FlexReport are available in **C1.Win.FlexReport.CustomFields.4** assembly. You can create your own custom fields and add them to the Report Designer palette. To do that, you have to:

1. Create a custom field class that derives from **C1.Win.FlexReport.CustomFields**.
2. Register your custom field assembly in **C1FlexReportDesigner.4.exe.settings**.

For registering a custom field, say MyField, add your control to the <customfields> section in the C1FlexReportDesigner.4.exe.settings file as follows:

```
<customfields>
<!-- THIS LINE ADDS A NEW FIELD TO THE DESIGNER -->
<item value="MyCustomFieldAssembly;MyCustomFieldAssembly.MyField" />
</customfields>
```

Note that the code above assumes that your field is called "MyField" and it can be found in the assembly called "MyCustomFieldAssembly". Also, MyCustomFieldAssembly should be in the same folder as the designer.

Following are built-in standard custom fields that are loaded in the FlexReportDesigner application by default:

- SuperLabel
- Maps

The source code for implementing these custom fields is:

```
<customfields>
```

```
<item value="C1.Win.FlexReport.CustomFields;C1.Win.FlexReport.CustomFields.SuperLabel"
/>
<item value="C1.Win.FlexReport.CustomFields;C1.Win.FlexReport.CustomFields.Map" />
</customfields>
```

Map Custom Field

FlexReport supports map fields using its extensible custom field architecture. In the following topics, you'll see how you can customize map fields in reports using the **FlexReportDesigner** application. The Map custom field uses two assemblies, C1.Win and C1.Win.Map, which should be referenced by your project before you begin.

To start using the Map custom field in the **FlexReportDesigner** application, complete the following steps:

1. Run the **C1FlexReportDesigner** application.
2. Confirm that the map icon is present in the **C1FlexReportDesigner** toolbar. If it is not included, you may need to add the following line to the <customfields> section of the **C1FlexReportDesigner.4.exe.settings** file: `<item value="C1.Win.FlexReport.CustomFields.4;C1.Win.FlexReport.CustomFields.Map" />`
3. Create a new report or open an existing report. See Step 1 of 4: Creating a Report Definition for an example.
4. Click the map icon and drag it onto your report to add a Map field.

That's it! The main aspects of the Map field include:

- tile and data layers
- legends
- styles
- expressions
- auto zooming/centering and data tracking

> Note that if C1FlexReport definition contains map field and the report is generated asynchronously, the map field cannot be displayed in the FlexViewer control. As a workaround please set the C1FlexViewer.UseAsyncRendering property to False.

Map Custom Field Properties

The important properties of the Map custom field are discussed in the following sections.

Layers

The main part of a map is the tile layer which provides raster graphics representing the Earth surface or part of it, and zero or more layers representing spatial data.

The tile layer is specified by the **TileSource** property. It may be set to a VirtualEarth tile source (road, aerial, or hybrid). The tile source may be set to "none" in which case no tiles will be drawn on the map. This may be useful especially when other layers such as KML provide enough data for the map visualization.

Note that unless the tile source is "none", the tiles are loaded from a network location when the report runs, which may slow things down considerably.

Except for the tile layer, all other layers are contained in the **Layers** collection. Currently, three layer types are supported as described below:

- **Points Layer**: A points layer allows to show spatial data as points on the map. A marker is drawn for each record of the data source. A marker's location is specified either by a Longitude/Latitude pair, or by a **MapLocation**, as described in Spatial Locations. The following points describe important aspects of the points layer.

    - **Data access:** when a points layer is processed at run time, the record source (either the layer's own **DataSource** if specified, or the report's data source filtered by the current group) is looped through, and a mark is drawn for each data record.
    - **Visual Styles:** the way point markers look is determined by the applied marker style. A points layer provides a default **MarkerStyle** property that allows to specify markers' shape, color and so on. Additionally, a **MarkerStyleExpr** expression may be specified, in which case at runtime it will be evaluated for each data record, and if a matching marker style is found in the **MarkerStyles** collection of the current map, or failing that of other maps in the report, that style will be applied instead of the default. (As described above, a style expression should evaluate to a string matching a style name in the styles collection.)
    - **Clustering:** when several point markers are located close to one another they may be "clustered" together into a single marker. That marker always shows the number of clustered point markers it represents. The visual style of the cluster marker may differ from the style of the point markers, and may even vary depending on the number of points it represents. Cluster styles are specified by the points layer's **ClusterStyles** collection, if more than one styles are provided the specific style is determined by the cluster size. Relevant points layer properties are: **ClusterDistance**, **ClusterDistribution** and **ClusterStyles**.
    - **Tracking:** if the **Track** property is **True**, automatic centering and zooming includes all layer's points.
- **Lines Layer**: A lines layer is used to draw a straight line between two points for each data row on the map. Spatial location for each point may be specified in the same manner as for the Points Layer: either with two **Longitude**/**Latitude** pairs (one for each end of the line), or with two **MapLocations** used to request locations from an online service. The following points describe important aspects of the lines layer.
    - **Data access:** as with the points layer, a lines layer allows to specify its own **DataSource**, or uses the report's record source filtered by the current group.
    - **Visual Styles:** styles are handled much in the same manner as with points layers, but instead of **MarkerStyles**, the **LineStyles** collection is used.
    - **Tracking:** if the **Track** property is **True**, automatic centering and zooming includes all layer's lines.
- KML Layer: KML (Keyhole Markup Language) is an XML based language that allows to describe various geographic information. For more info on KML, see Keyhole Markup Language. A KML layer allows to load into the map and show a local or Web-based KML file. This layer renders a KML (Keyhole Markup Language) or KMZ (compressed KML) file on the map. The file name is specified by the **KmlFileName** property on the layer. The file may be loaded from a URL, from a local disk file, or embedded in the report. If the file is not embedded (**EmbedKmlFile** is **False**), and the directory is not specified, the file is loaded from the directory containing the report definition.
    - **KML item expressions:** when a KML layer is rendered, items present in the KML file are processed in sequence. As each item is loaded, several expressions specified on the layer are evaluated allowing to control the process - for example, only load certain items based on various criteria, or modify items' visual attributes. Additionally, if a **DataSource** is specified for the KML layer, the data may be filtered for each KML item prior to evaluating the item expressions. Following is a detailed explanation of the properties involved in evaluating KML item expressions. Note that in all those expressions, the special variable **kmlItemName** may be used, and refers to the KML item name that is currently being processed.
    - **ItemFilterExpr:** if (and only if) a **DataSource** is specified on the KML layer, this filter is applied to the retrieved data prior to evaluating other expressions. For example, if the layer's record source contains a Country field, and the KML file includes country items, the following filter:
      `kmlItemName=Country`
      will ensure that for each KML item, other item expressions will evaluate with data corresponding to the current item's country.
    - **ItemTrackExpr:** if specified, determines whether an item is used to automatically center/zoom the map. If left empty, true is assumed.
    - **ItemVisibleExpr:** if specified, determines an item's visibility. If left empty, true is assumed.
    - **ItemStyleExpr:** if this expression evaluates to a valid style name in the **KmlItemStyles** collection (of the current or any other Map in the report), this style is applied to the item. This may be used for instance to fill different states with different colors depending on a data value such as orders total for that state.

- **ItemStyle.ItemNameExpr:** the KML item style itself contains one calculated property, the item's name. This allows to suppress the name rendered on the map, or replace it with report data (such as orders total).

Tracking

The map shown by a Map field can automatically center and zoom in on the data shown on the map. This behavior is determined by two factors:

- The **AutoCenter** and **AutoZoom** properties' values specified for the whole Map field, together with several related properties fine-tuning the automatic centering and zooming (**AutoZoomPadLon**, **AutoZoomPadLat**, **MaxAutoZoom**, **RoundAutoZoom**).
- The spatial data represented by the layers, provided that data is "tracked". Tracking (such as whether or not a particular piece of spatial data should be used for automatic centering and zooming) is determined by the layer's **Track** property. Additionally, for KML layers an expression may be specified which will determine whether a specific KML item should be tracked or not.

Styles

Visual attributes of map elements are mostly defined by styles. There are several types of styles (point marker styles, line styles and KML item styles); the applicable type is determined by the context, such as points layers use point marker styles, lines layers use line styles, and so on. Usually a style may be specified as a data driven expression (so that the actual style depends on run time data), with a fallback style used by default. How style expressions are specified and evaluated is described next.

The Map custom field contains 3 style collections:

- MarkerStyles
- LineStyles
- KmlItemStyles

These styles are available to all layers defined on the **Map**, and also to other **Map** fields in the current report. The styles in each collection are addressable either by index or - preferably - by name (using the **Name** property). When a style expression evaluates to a string, that string is used to search for a matching style, first in the current map and if that fails, in all other maps on the current report (only matching type styles are searched; for example, only **MarkerStyles** collections are searched for a point marker style, and so on).

Spatial Locations

Points and lines layers provide two different ways to specify spatial locations for the data:

- As a pair of expressions that evaluate to a longitude/latitude pair at run time. Typically these would directly reference corresponding data fields (longitude and latitude) stored in the data source.

- As a **MapLocation**, an expression (or a list of expressions) that evaluates to a string that can be used to retrieve the corresponding spatial location using an external online service (Google Maps). If the specified **MapLocation** contains semicolons, it is treated as a list of semicolon-delimited expressions, each of which is evaluated separately and then combined to use as the query. A typical **MapLocation** could look like this:

  ```
  "Address;City;PostalCode;Country"
  ```

  which would fetch *Address*, *City*, *PostalCode*, and *Country* fields from the data source and then combine them to query the external service.

  Note that using **MapLocation** may be very time consuming due to Internet access. Hence by default the retrieved spatial data is stored in a local disk file. The path to that file is specified by the **Map.GeoCachePath** property. By default the file's name is "geocache.xml", and it is stored in the same directory as the report definition. Disabling geocaching is not recommended.

Legends

A map can have several associated legends, rendered within its bounds. To facilitate placing a legend outside the map's bounds, the legend can be associated with any map field in the report, so you can add an empty map field just to hold a legend describing another map.

Legends are contained within the **Legends** collection of the **Map** field. To add a legend, add an item to that collection. The location of a legend within its map's bounds is determined by the **LegendAlignment** property. Orientation determines whether items within the legend are placed vertically (default) or horizontally. Several other properties allow to fine-tune the way the legend looks.

Items within the legend are represented by the Items collection. That collection may be populated automatically with data from non-KML layers of the current map, if the **Automatic** property of the legend is set to **True**. In that case the Items collection cannot be edited. Otherwise, the legend items must be added manually.

The following types of legend items are supported:

- **LegendLayerStyleItem**: represents a layer style. The designer allows to select an existing layer or style represented by the legend item. Depending on the selected layer style, the legend item may represent a point marker (for points layers/styles), a line (for line layers/styles) or a color swatch (for KML item styles).
- **LegendColorSwatchItem**: represents an arbitrary color swatch.
- **LegendTextItem**: represents arbitrary text.

Adding Map Custom Field

Now that you familiar with the basics of **Map** field, let us add a map to a report showing Employees, Suppliers, and Customers summarized by City.

Complete the following steps:

1. **Create the base report.**

   Add a new report in the designer, with **C1NWind.mdb** as the data source, with the following SQL query:
   `Customers and Suppliers by City`

2. **Add the main map.**

   You'll add the map to the report's header:

   - Make some room for the map by dragging the header's bottom edge down in the report designer.
   - Click on the Map custom field icon and drag it onto the header.

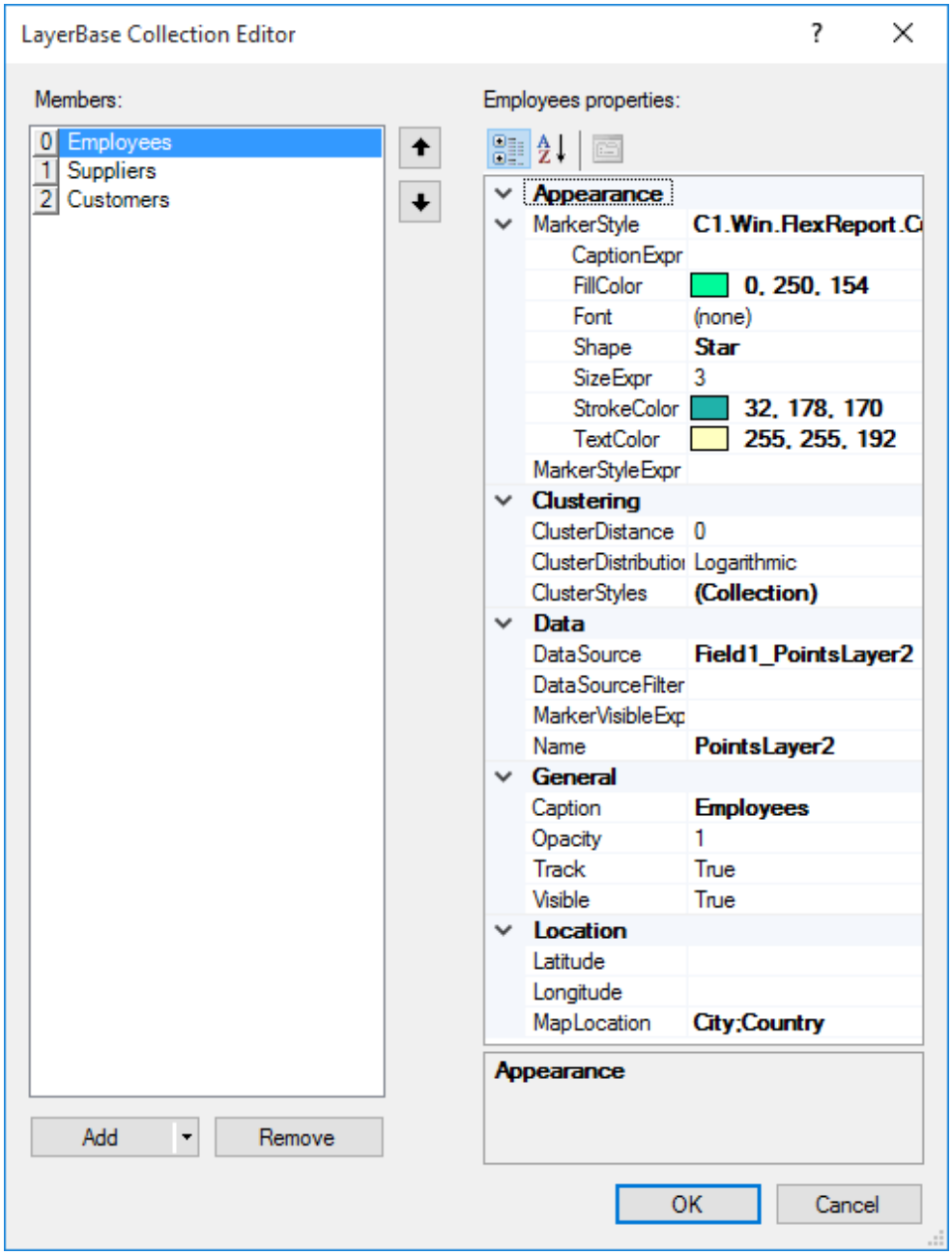
3. **Adjust the map's properties.**

   Set the map's properties as follows (only non-defaults are shown here):

   - **AutoCenter**: False
   - **AutoZoom**: False
   - **CenterLatitude**: 10
   - **CenterLongitude**: 15
   - **ShowScale**: True
   - **TileSource**: VirtualEarthAriel
   - **ZoomLevel**: .55

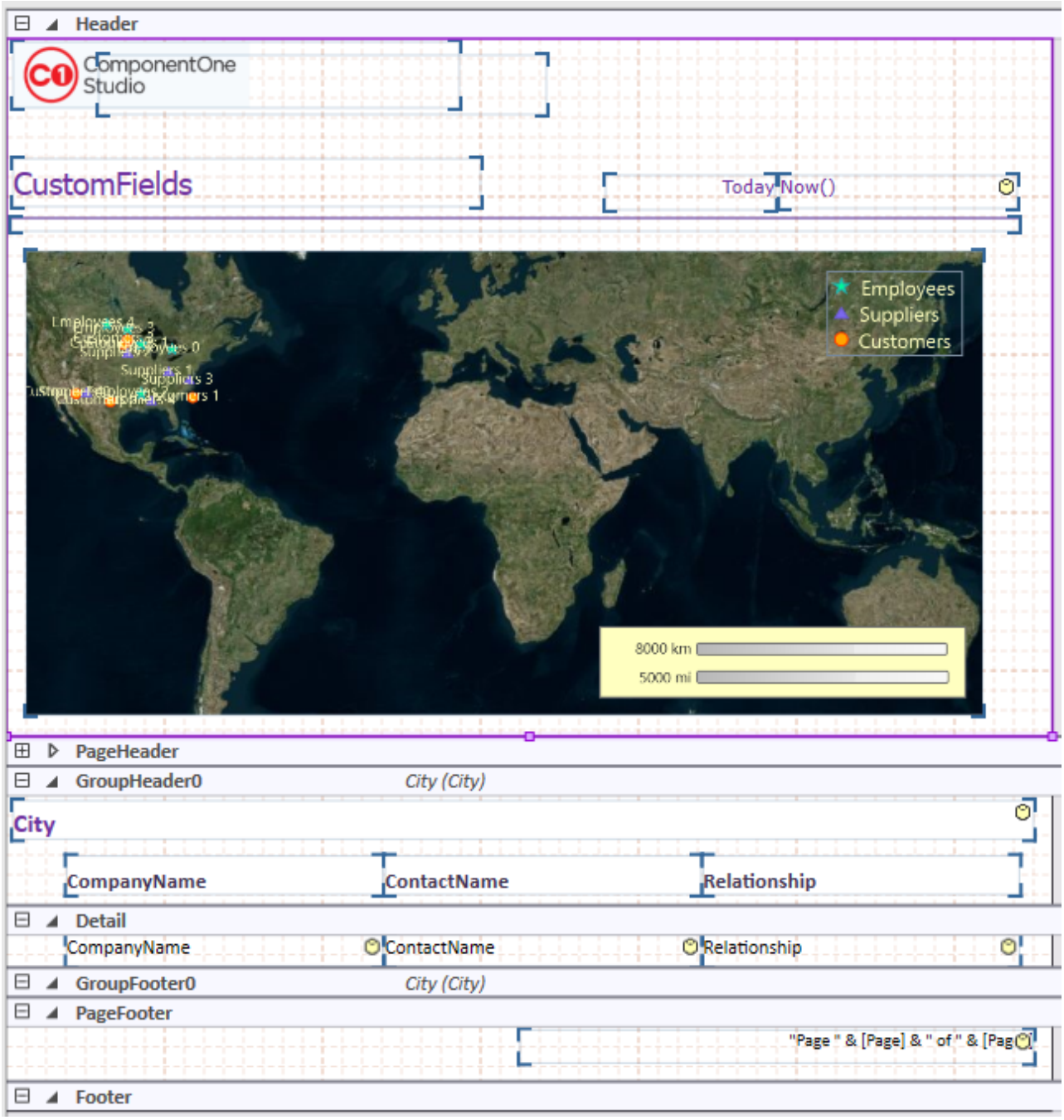
4. **Add Layers.**

   Click ellipsis button next to Layers collection to open **LayerBase Collection Editor**. Add Members 'Employees', Suppliers', and 'Customers', select the data source and set their marker style and map location properties.

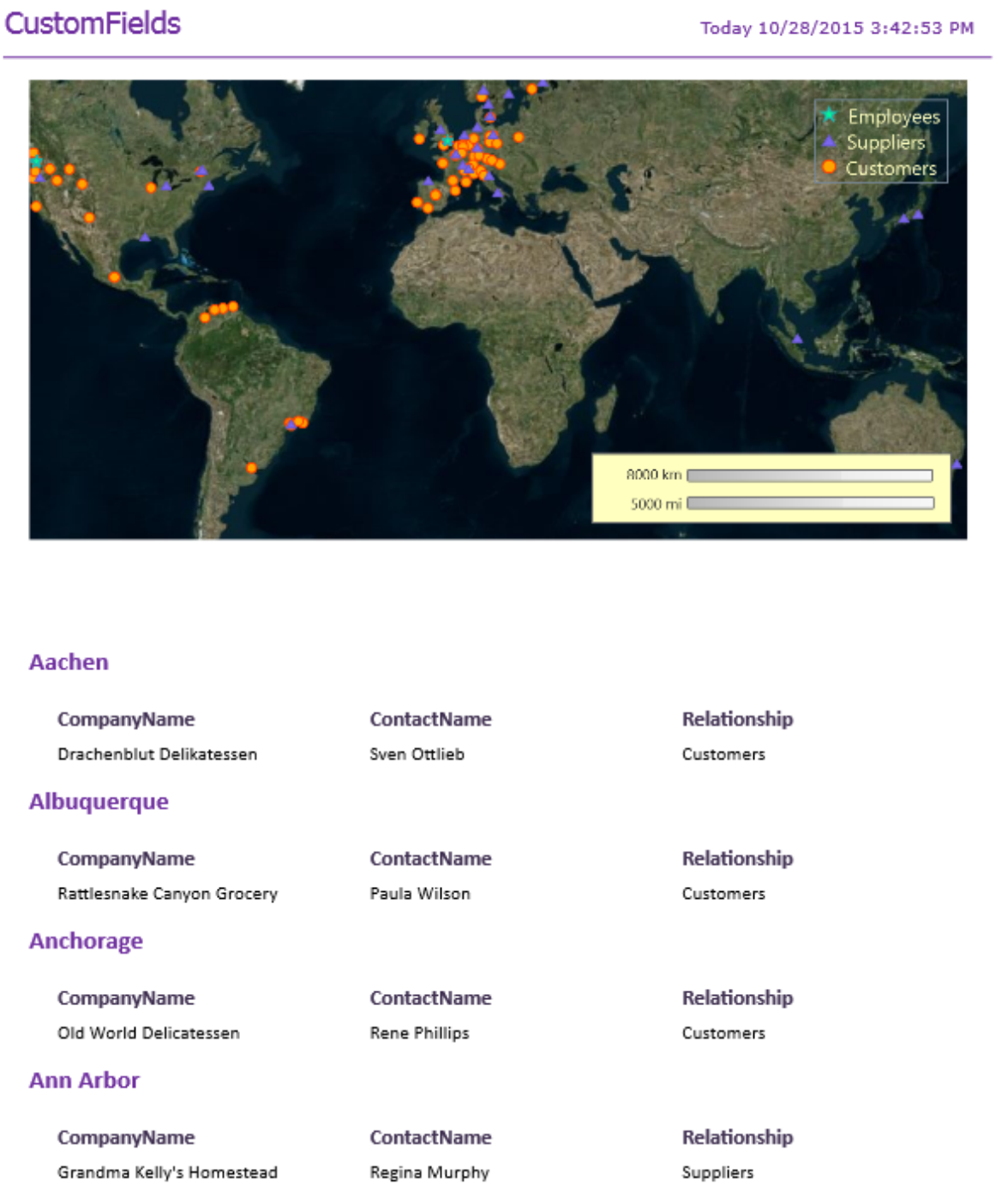


Note that you need to add a data source every time before you add a member to the **LayerBase Collection Editor** as each member uses different data source.

The design area looks like the following image:



5. Preview the report.



SuperLabel Custom Field

SuperLabel fields are used to insert HTML text in reports.

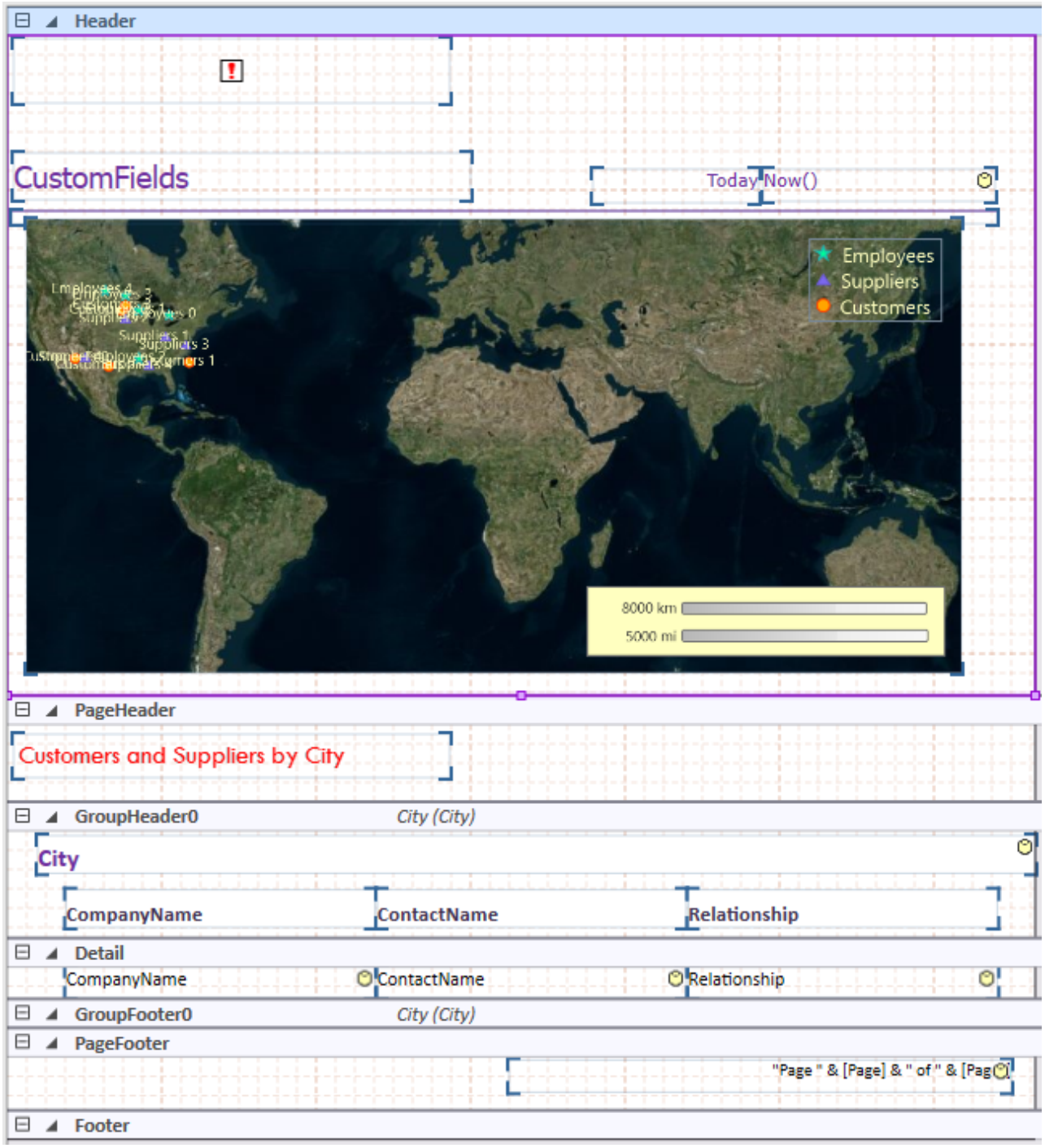
Let's add a SuperLabel field to the report created in Adding Map Custom Field.

1. Open the report.
2. Set PageHeader's **Visible** property to True.
3. Drop a **SuperLabel** field in the Page Header section of the report.

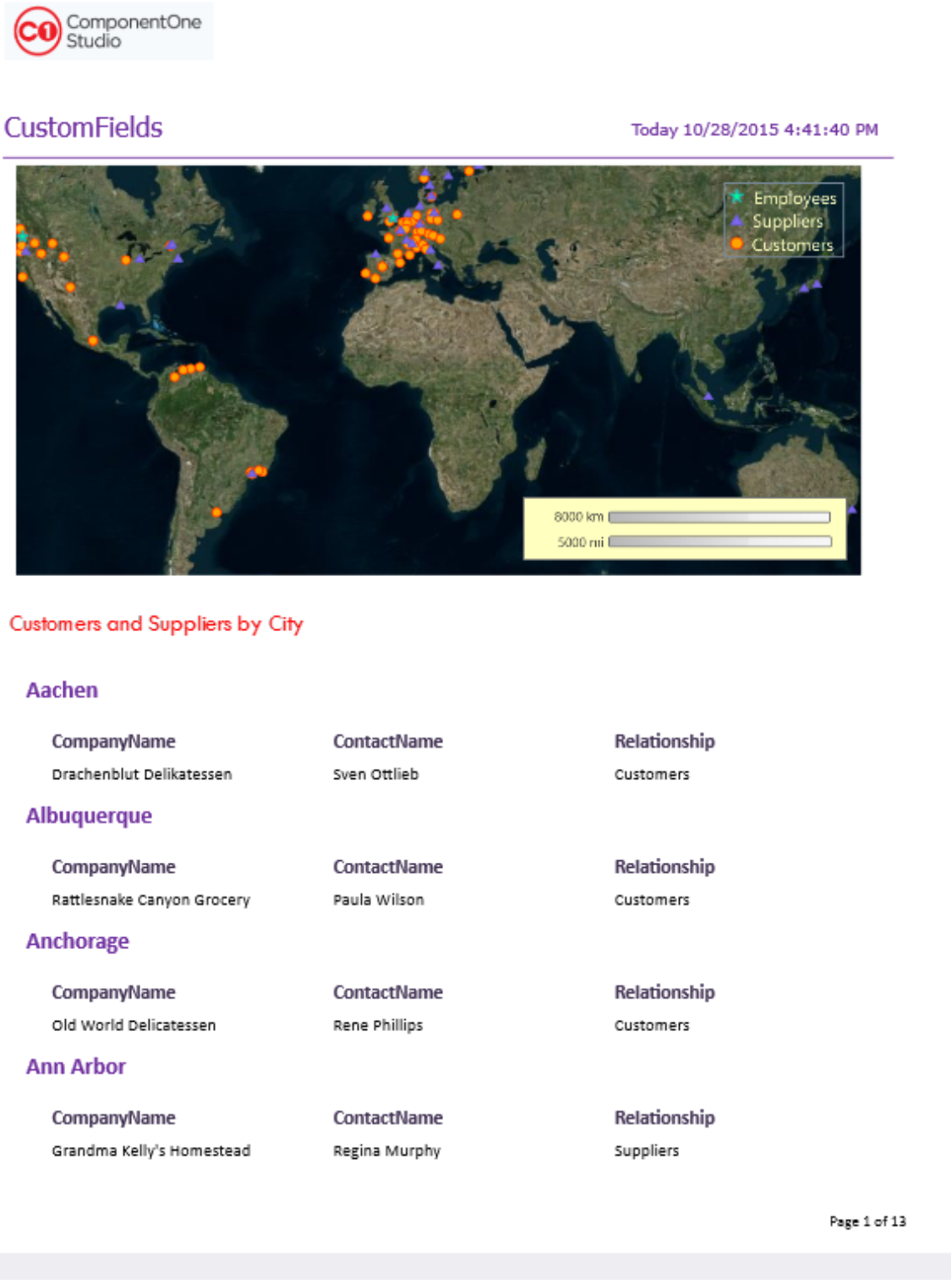
4. From the Properties window, set **Text** property to the following text:

SuperLabel.Text
`<html><body><font color="Red">Customers and Suppliers by City</font></body></html>`

5. In the designer, SuperLabel field should now look as follows:



6. Preview the report:



Working with Parameters

Using Parameters

Report parameters in FlexReport can be used to perform following tasks:

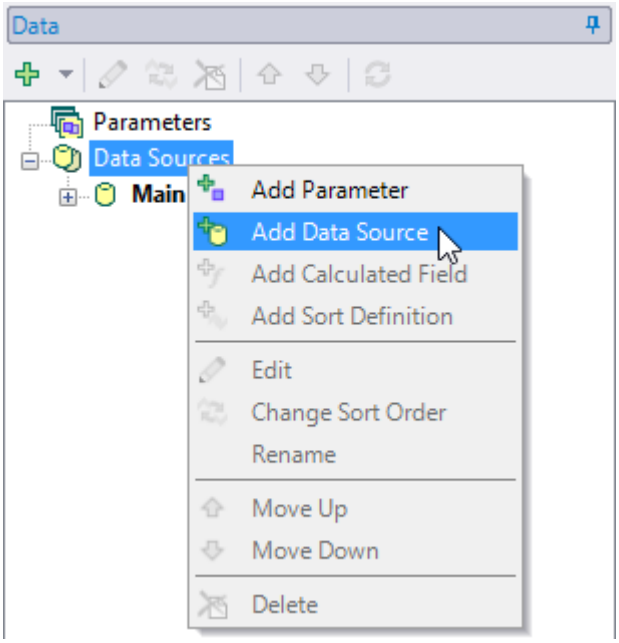
- Data binding
- Creating expressions for calculated fields

- Creating Subreports
- Passing multiple values to a field by adding multi-value parameters
- Silently passing values to a report
- Managing large amount of data using cascading parameters

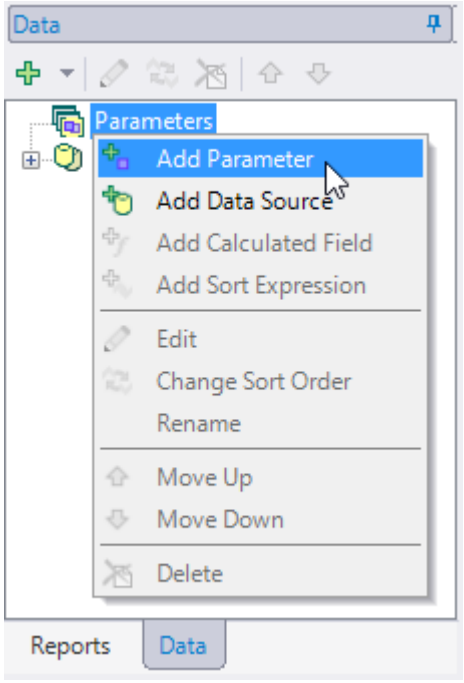
Adding parameters to a report

The steps to add parameter to a report are as follows:

1. Run **C1FlexReportDesigner.exe** application.
2. Create a new report. Bind it to a data source, which is the Main data source. The report opens in the Design mode.
3. Click the **Data** tab.
4. Right-click **Data Sources** and click **Add Data Source** to add a data source for specifying a parameter. Add as many data sources as the number of parameters that need to be added to the report. See Adding Multiple Data Sources for more information.



5. Right-click **Parameters** and select **Add Parameter** from the context menu.



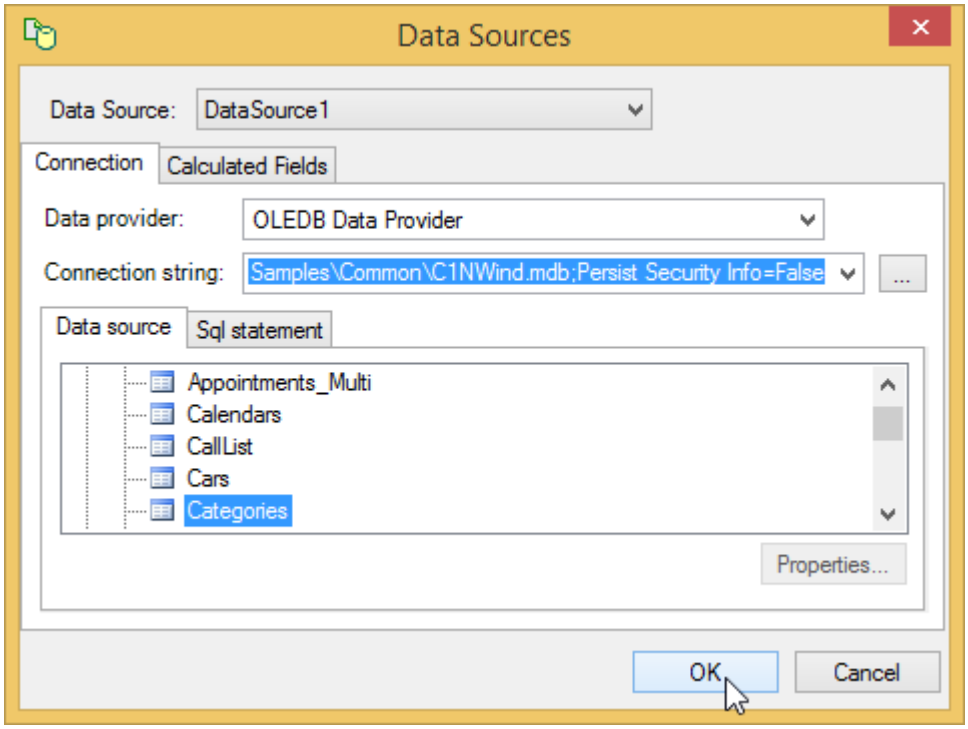
6. Set the properties of parameters from the Properties window depending upon the task that needs to be accomplished using parameters.

The following sections explain the various tasks that can be achieved using parameters.

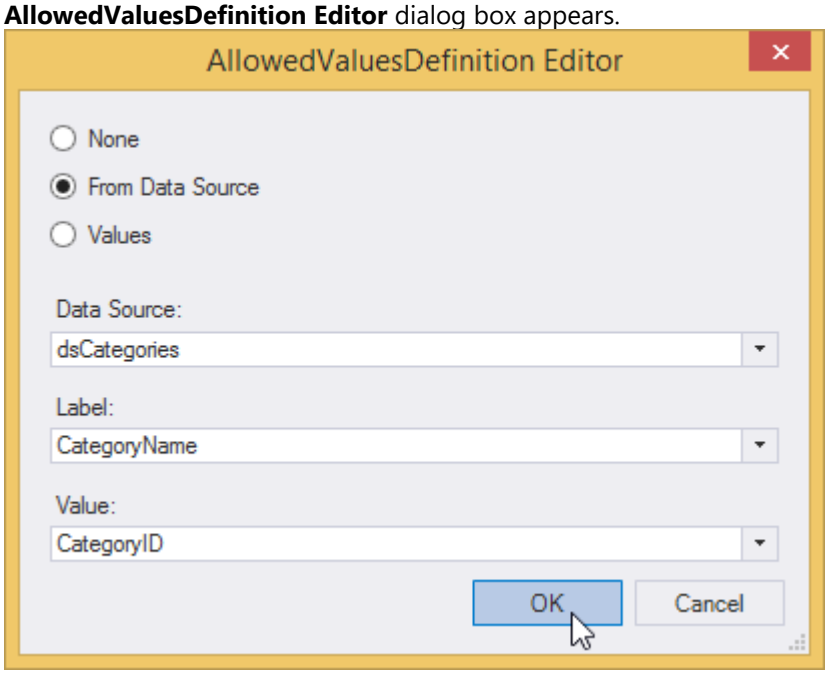
Data Binding

To perform data binding in report parameters, the steps are as follows:

1. Run **C1FlexReportDesigner.exe** application.
2. Create a new report. Bind it to a data source, which is the Main data source. The report opens in the Design mode.
3. Click the **Data** tab.
4. Right-click **Data Sources** and select **Add Data Source** The **Data Sources** dialog box appears.



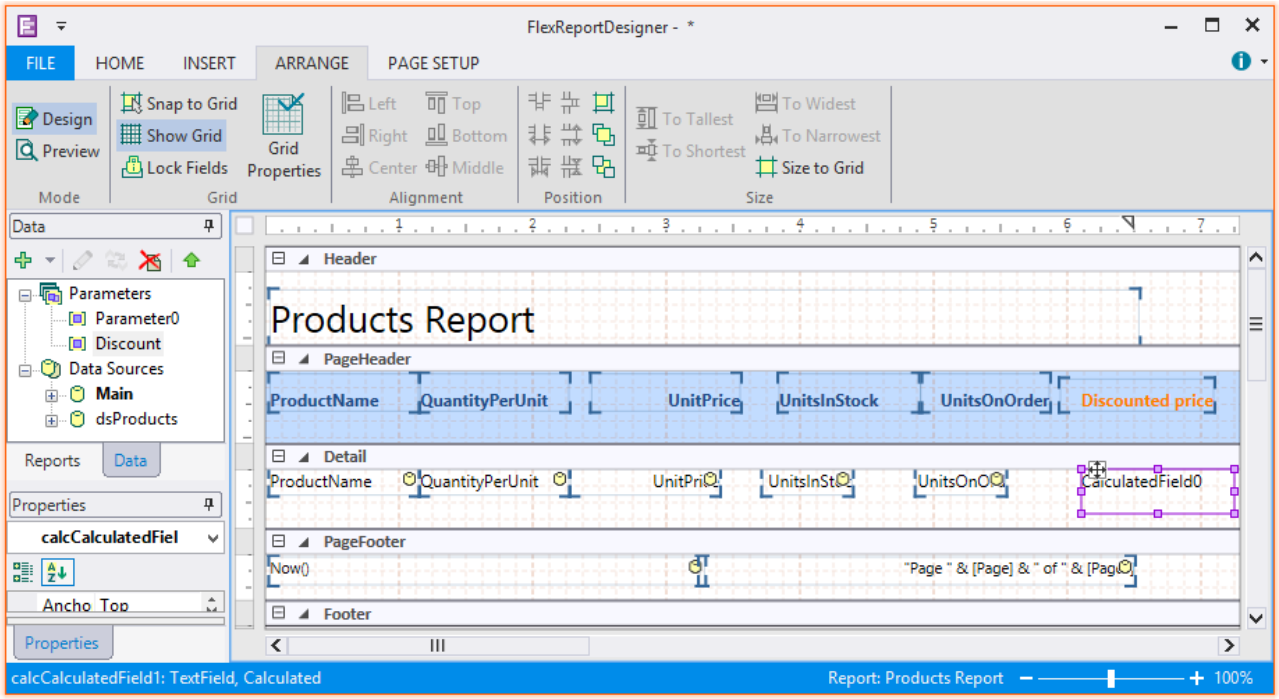
5. Select a **Data Source,** say **DataSource1**, to create a new data source and set Connection, Data provider, Connection string, etc.
6. You can rename DataSource1 from the Properties window by setting **Name** property. In our case, we have set Name to dsCategories.
7. Add a parameter by right-clicking **Parameters** and then selecting **Add Parameter.**
8. From the Properties window, click the ellipses button next to **AllowedValuesDefinition**. The **AllowedValuesDefinition Editor** dialog box appears.



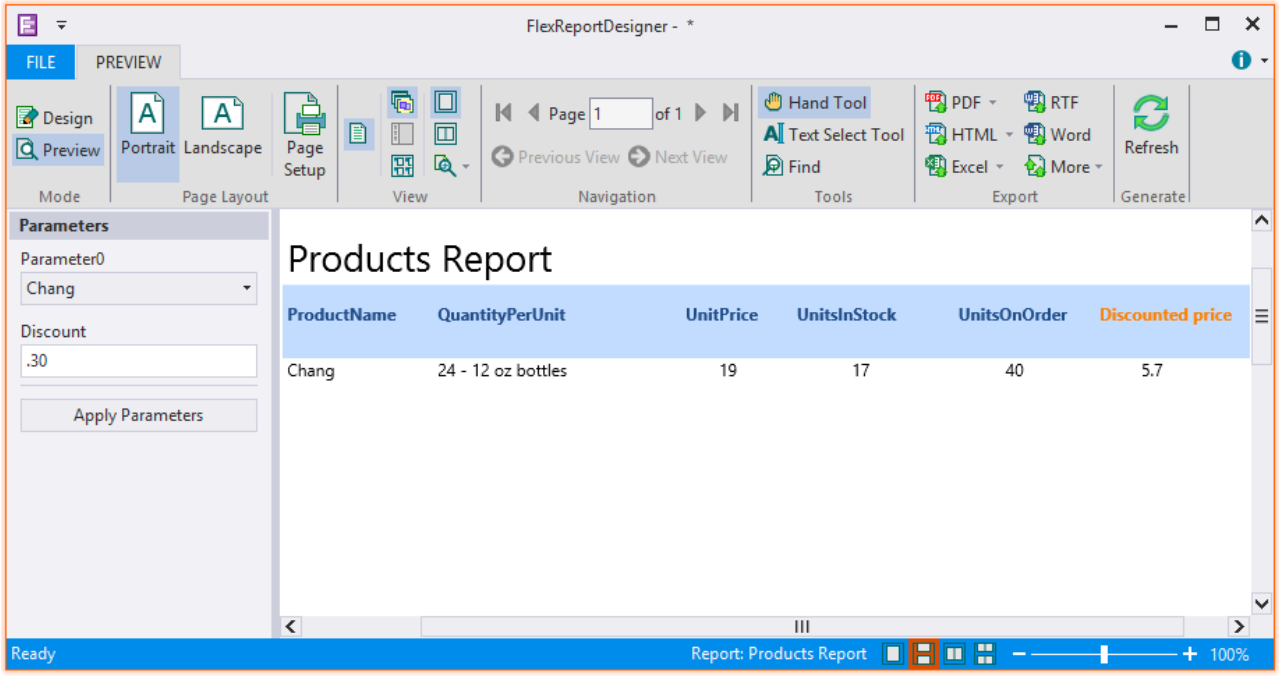
9. Select the **From Data Source** radio button and then select Data Source, Label, and Value for binding the data source to the parameter as shown. Click OK.
10. From the Properties window, set the **DataType** to a value that is same as the data type of **Value in the AllowedValuesDefinition Editor**. In our case, **DataType** is set to Integer since Value=CategoryID is integer.

Calculated Fields

1. Run **C1FlexReportDesigner.exe** application.
2. Create a new report. Bind it to Products table. The report opens in the Design mode.
3. Click the **Data** tab.
4. Right-click **Data Sources** and select **Add Data Source.** The **Data Sources** dialog box appears.
5. Add another Data source and bind it to Products table. This data source is added for the parameter. Name this data source as 'dsProducts'.
6. Add a parameter 'Parameter0' by right-clicking **Parameters** and then selecting **Add Parameter**.
7. From the Properties window, click the ellipses button next to **AllowedValuesDefinition**. The **AllowedValuesDefinition Editor** dialog box appears.
8. Select the **From Data Source** radio button and then select Data Source - dsProducts, Label - ProductName, and Value - ProductName. Click **OK**.
9. Add another parameter 'Discount'.
10. Edit the 'Main' data source. Write following Sql statement:
    ```
    Select * from Products where ProductName=Parameter0
    ```
11. Add a **Calculated** field from the Fields group in the **Insert** tab. The **VBScript Editor** opens.
12. Specify the following expression in the editor:
    ```
    UnitPrice*Discount
    ```
13. Drop the Calculated field in the Detail section. Also add a Text field to display 'Discounted Price' in the PageHeader section.



14. Preview the report. Select the product name from Parameter0 and enter the discount value in Discount parameter. The discounted price is calculated on clicking **Apply Parameters**.

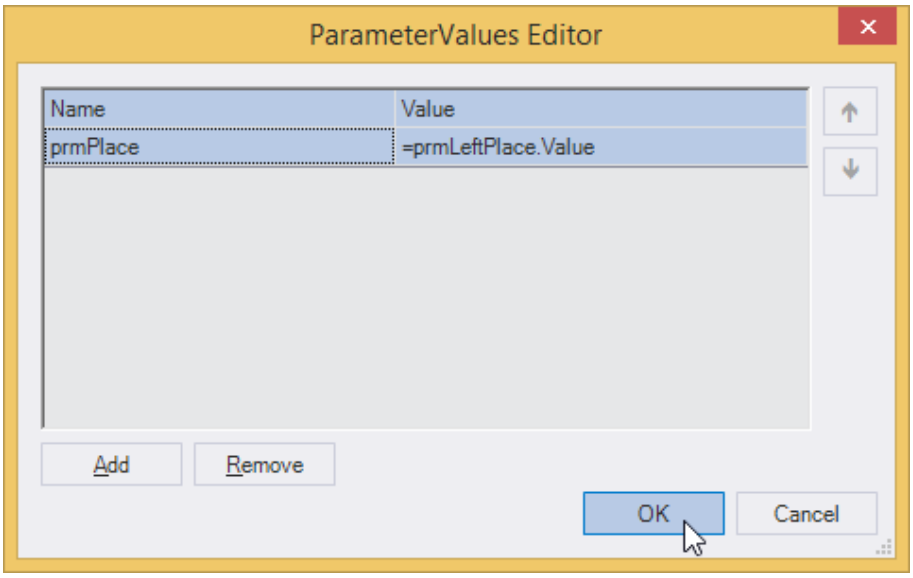


Subreports

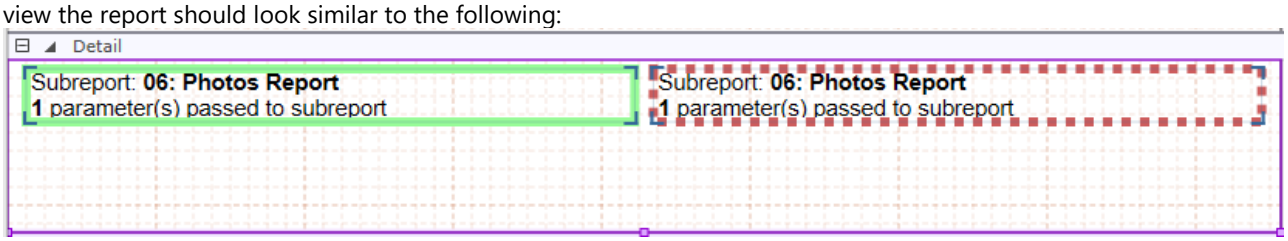
Subreports are useful in displaying additional information about the data present in the main report. Just as you pass parameters to a report, you can also pass parameters to subreports and create connection between the data values in subreports.

For instance, you want to fetch data in subreports that are rendered side by side on a main report. The steps to attain such a scenario are as follows.

1. Run **C1FlexReportDesigner.exe** application.
2. Create a new report definition, main report, in an unbound mode.
3. Create another report and bind it to Photos report available in the C1NWind.mdb.
4. From the **Insert** tab, click **Subreport** field. All the available reports will be displayed.
5. Drop two **Subreport** fields side by side.
6. Add two parameters that will be passed one for each subreport.
7. Set the name of parameters from **Name** property - 'prmLeftPlace' for parameter on left subreport and 'prmRightPlace' for parameter on right subreport.
8. Add Data Source for parameters - set data provider and connection string. Write a Sql Statement that will be used to pass values to the parameters. For example, to select 'Place' from the 'Photos' report, the statement should be:
   ```
   select distinct Place from Photos order by Place
   ```
9. Set **AllowedValuesDefinition** property for both the parameters -
   a. Click the ellipses button next to **AllowedValuesDefinition**.
   b. In the **AllowedValuesDefinition Editor** dialog box, select the **From Data Source** radio button.
   c. Select Data Source, Label, and Value for binding the data source to the parameter.
10. Set the prompt text from **Prompt** property for each of the parameter.
11. Select Subreport -Photos- click Data tab
12. Edit the data source and write the following Sql Statement:
    ```
    select distinct Place from Photos order by Place
    ```
13. Select the main report and click the Subreport field on left. Click ellipses next to **ParameterValues.ParameterValuesCollection Editor** appears. Set the Name and Value as shown.



14. Similarly, set the **Name** - prmPlace and **Value** - prmRightPlace.Value for Subreport field on the right. In the design view the report should look similar to the following:



15. Preview the report.

You can see two parameter prompts with a list of values (i.e. places), one for each subreport. Select the values and click **Apply Parameters**. Here, we have selected Chaumont for left subreport and Cote d'Azur for right subreport.

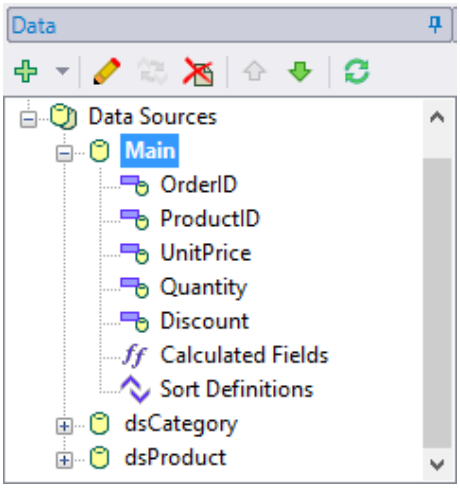
Cascading Parameters

Cascading parameters are used when a list of values in one parameter depends on the value selected for the other parameter. So you have two parameters, where one parameter alters the data source used to list data values for another

parameter.

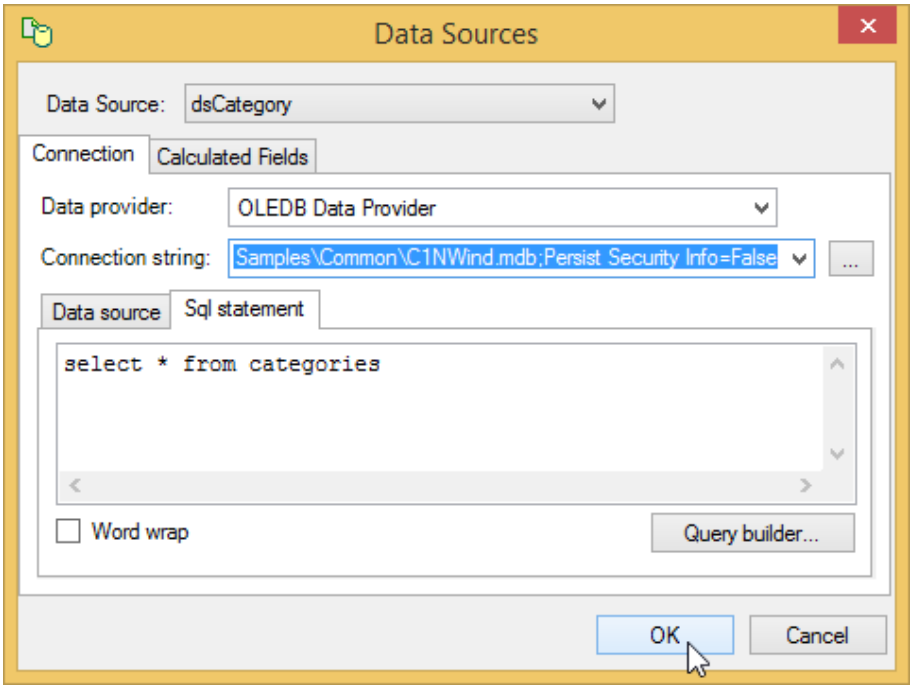
For instance, if you want to select a Product from a list of products that belong to the selected Category, then you need to do the following:

1. Create a new report. Bind the report to Order Details available in C1NWind.mdb.
2. Add two Data Sources, one for each parameter:
    - dsCategory to pass values to parameter prmCategory
    - dsProduct to pass values to parameter prmProduct



3. Right-click **dsCategories** and select **Edit** to edit the data source. Set Data provider, Connection string, and write following Sql Statement:

```
select * from categories
```



4. Repeat the previous step for **dsProducts**. Write the following Sql Statement:

```
select * from products where categoryid = prmCategory
```

5. Right-click **Main** data source, select **Edit** and write Sql Statement for Main data source:

```
select * from [order details] where productid = prmProduct
```

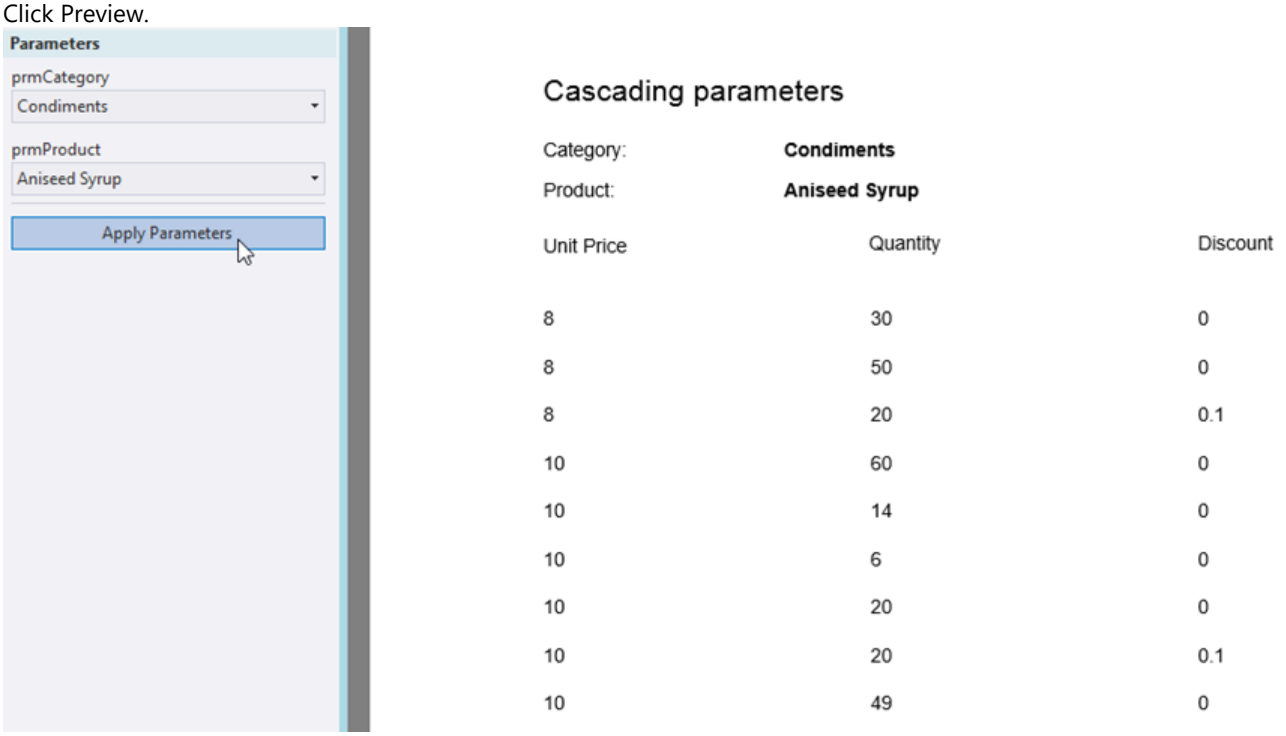
6. Add two parameters:

- prmCategory to select Category.
- prmProduct to select Product that belongs to the category selected in prmCategory parameter.

7. Set the **AllowedValuesDefinition** property for both the parameters as shown:

	prmCategory	prmProduct
Data Source	dsCategory	dsProduct
Label	CategoryName	ProductName
Value	CategoryID	ProductID

8. Click Preview.

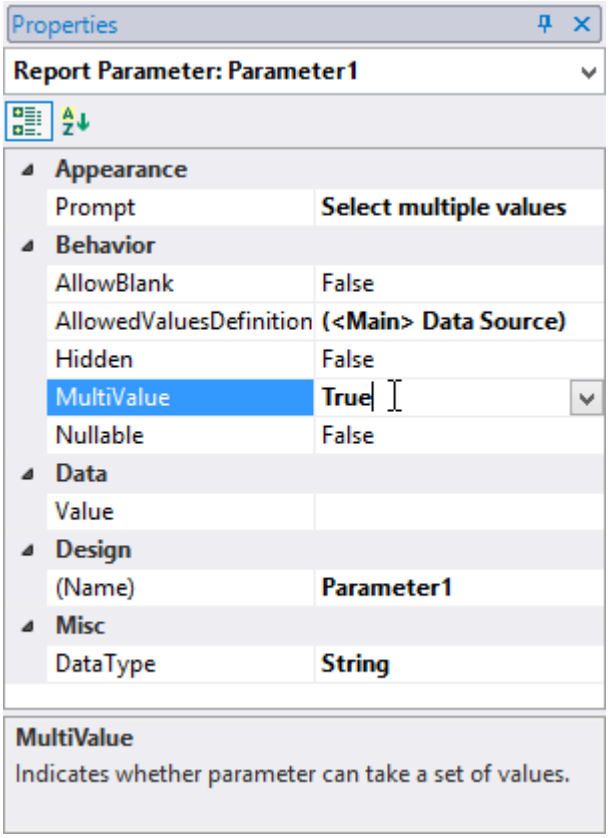


On selecting a category, a list of products available in that category are displayed in the other parameter. You can then select the product and then click **Apply Parameters**. Here, we have displayed data in Aniseed Syrup product that falls in the Condiments category.

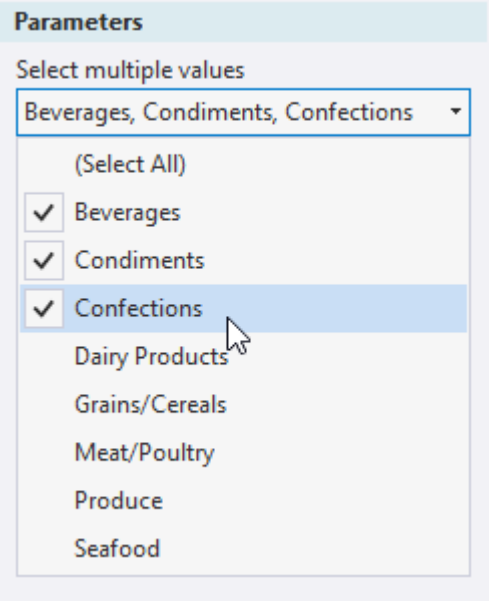
Multi-value Parameters

Multi-value parameters are parameter collection that allows you to pass multiple values for a parameter; that is, you can select multiple data from a list of data.

In FlexReportDesigner application, a parameter can be made multi-value by setting **Parameter.MultiValue** to True.



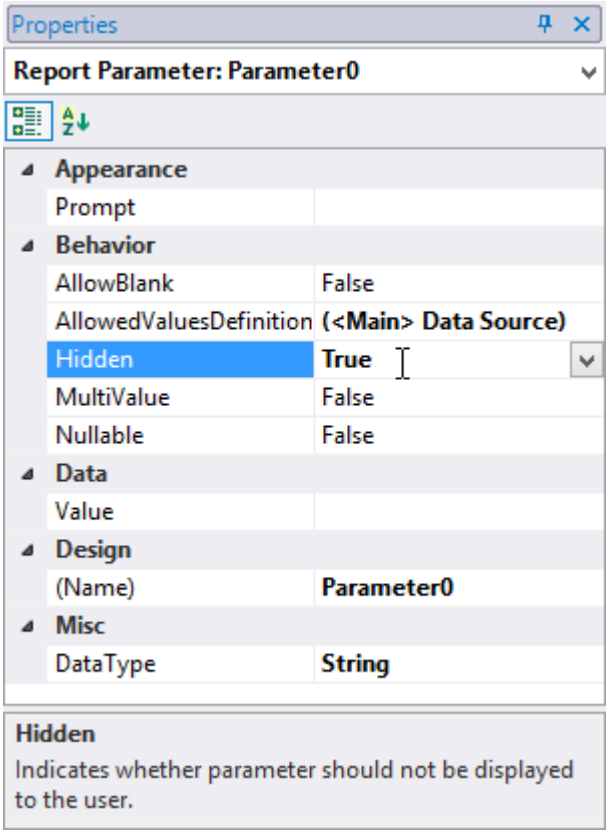
On previewing a report with multi-value parameter, a list of values bound to the parameter is displayed. You can then select the values that you want to be rendered in the report.



Pass Parameters Silently

Passing the parameters silently allows passing the values of the parameters without any user prompt (or user interaction), while rendering the report.

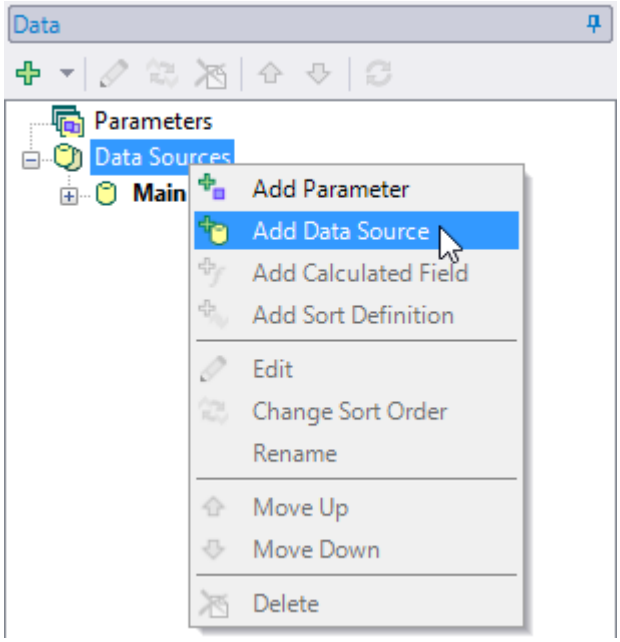
To pass the parameters silently, define parameter properties and set **Parameter.Hidden** property to True.



Adding Multiple Data Sources

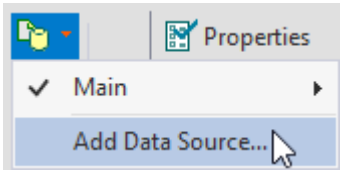
A FlexReport definition can include several data sources. Adding a new data source to a report using FlexReportDesigner is quiet easy. The steps to add data sources in an existing report bound to the Main data source are as follows:

1. Click the **Data** tab.
2. Right-click the **Data Sources** node.
3. Select **Add Data Source**.

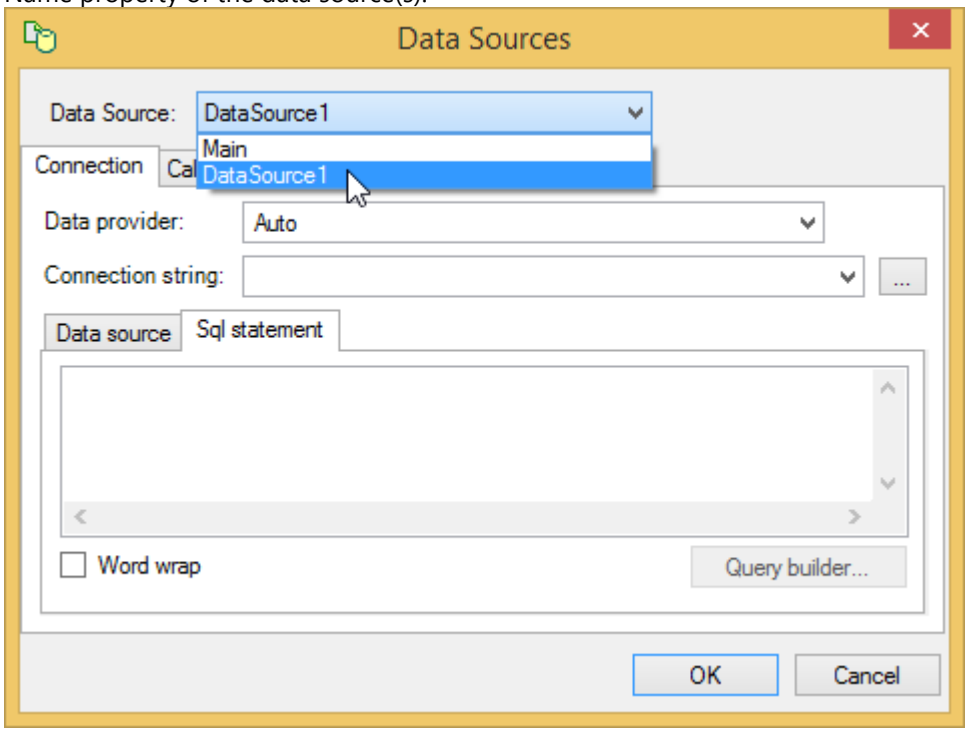


Or

1. Click the **Home** tab.
2. Click the drop-down on the **Data Sources** option.
3. Click **Add Data Source**.



4. **Data Sources** dialog box appears. From the drop-down next to Data Source option, select DataSource1, DataSource2, and so on for adding as many data sources. You can later on rename these data sources from the Name property of the data source(s).



Changing Data Source of FlexReport

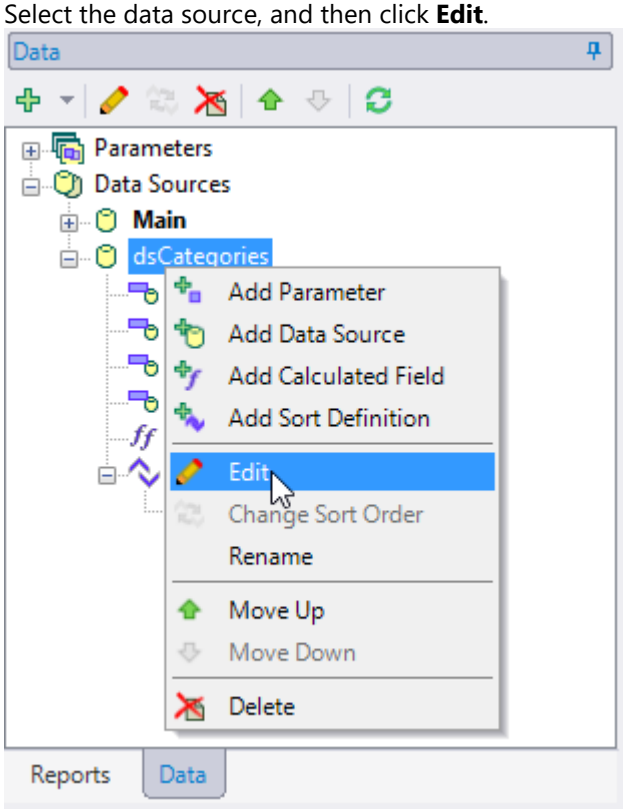
When you create a new data bound report, the first thing you do is select a data source. With the FlexReportDesigner application, you can easily edit all the data sources to which the report and its elements are bound.

The steps to change/edit a data source in a report using FlexReportDesigner are as follows:

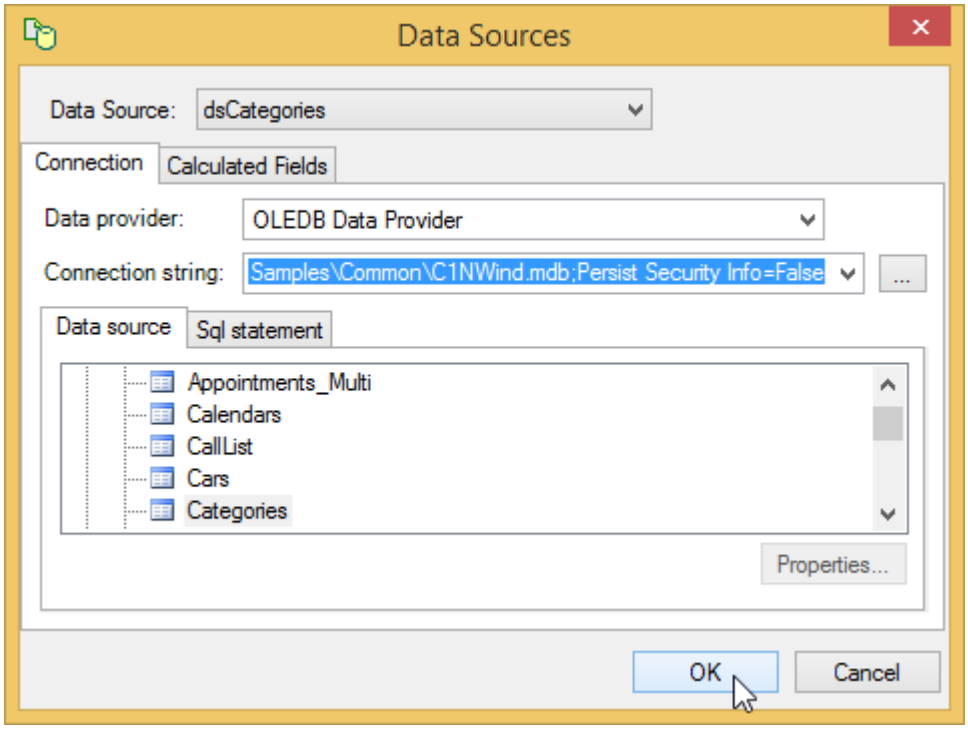
1. Click the **Data** tab.
2. Expand the **Data Sources** node.
3. Right-click the data source you want to edit and select **Edit**.

Or

1. Click the **Home.**
2. Click the drop-down on the **Data Sources** option.
3. Select the data source, and then click **Edit**.



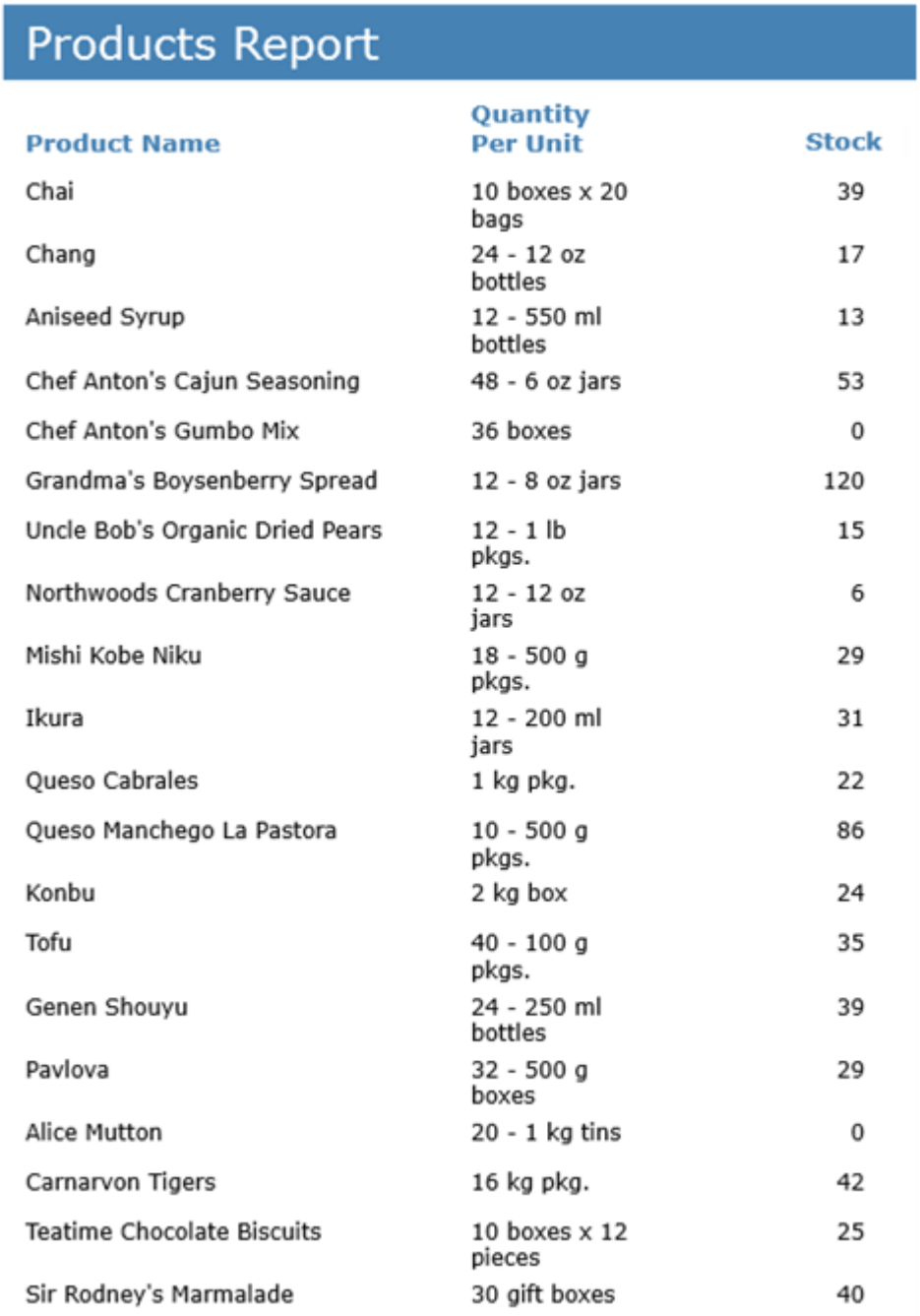
4. The **Data Sources** dialog box opens. Select the new data source, change the connection string, and edit the Sql statement, and you are done.



Sorting Data using Designer

The data in a report can be easily sorted using FlexReportDesigner. The steps to sort data in a report are as follows:

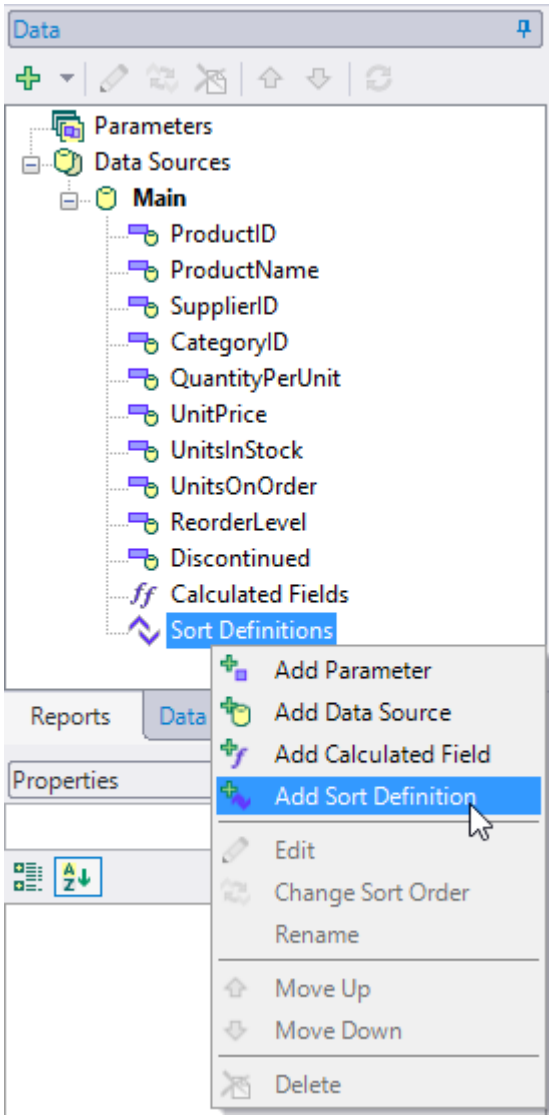
1. Create a new report - Products Report - bound to Main data source through C1NWind.mdb. Select Product Name, Quantity Per Unit, and Stock as the text and calculated fields, so that the report appears as shown:



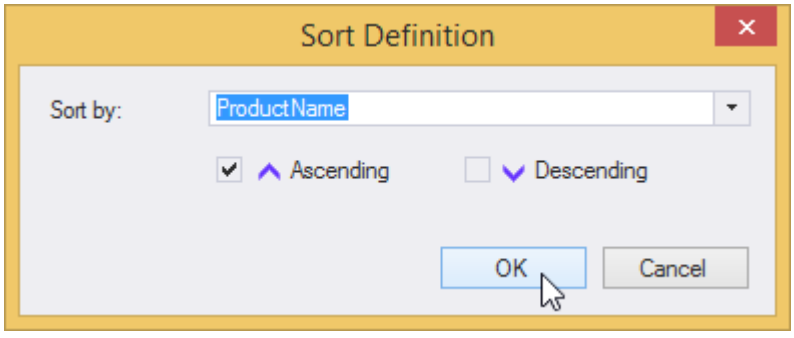
Products Report

Product Name	Quantity Per Unit	Stock
Chai	10 boxes x 20 bags	39
Chang	24 - 12 oz bottles	17
Aniseed Syrup	12 - 550 ml bottles	13
Chef Anton's Cajun Seasoning	48 - 6 oz jars	53
Chef Anton's Gumbo Mix	36 boxes	0
Grandma's Boysenberry Spread	12 - 8 oz jars	120
Uncle Bob's Organic Dried Pears	12 - 1 lb pkgs.	15
Northwoods Cranberry Sauce	12 - 12 oz jars	6
Mishi Kobe Niku	18 - 500 g pkgs.	29
Ikura	12 - 200 ml jars	31
Queso Cabrales	1 kg pkg.	22
Queso Manchego La Pastora	10 - 500 g pkgs.	86
Konbu	2 kg box	24
Tofu	40 - 100 g pkgs.	35
Genen Shouyu	24 - 250 ml bottles	39
Pavlova	32 - 500 g boxes	29
Alice Mutton	20 - 1 kg tins	0
Carnarvon Tigers	16 kg pkg.	42
Teatime Chocolate Biscuits	10 boxes x 12 pieces	25
Sir Rodney's Marmalade	30 gift boxes	40

2. To view the report where data is sorted by Product Name, go to **Design** mode.
3. Click **Data** tab.
4. Expand **Data Sources** node.
5. Expand the Main data source to which the report is bound.
6. Right-click **Sort Definitions**.



7. Select **Add Sort Definition**. **Sort Definition** dialog box appears.



8. Select **ProductName**, check the **Ascending** checkbox, and click **OK**.
9. Preview the report.

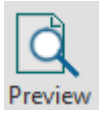
Products Report

Product Name	Quantity Per Unit	Stock
Alice Mutton	20 - 1 kg tins	0
Aniseed Syrup	12 - 550 ml bottles	13
Boston Crab Meat	24 - 4 oz tins	123
Camembert Pierrot	15 - 300 g rounds	19
Carnarvon Tigers	16 kg pkg.	42
Chai	10 boxes x 20 bags	39
Chang	24 - 12 oz bottles	17
Chartreuse verte	750 cc per bottle	69
Chef Anton's Cajun Seasoning	48 - 6 oz jars	53
Chef Anton's Gumbo Mix	36 boxes	0
Chocolade	10 pkgs.	15
Côte de Blaye	12 - 75 cl bottles	17
Escargots de Bourgogne	24 pieces	62
Filo Mix	16 - 2 kg boxes	38
Fløtemysost	10 - 500 g pkgs.	26
Geitost	500 g	112
Genen Shouyu	24 - 250 ml bottles	39
Gnocchi di nonna Alice	24 - 250 g pkgs.	21
Gorgonzola Telino	12 - 100 g pkgs	0
Grandma's Boysenberry Spread	12 - 8 oz jars	120

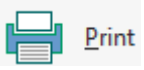
Previewing and Printing FlexReport

After designing the report, you would like to Preview the report and may also wish to take a Print of the report. Both the tasks can be accomplished by a few button clicks.

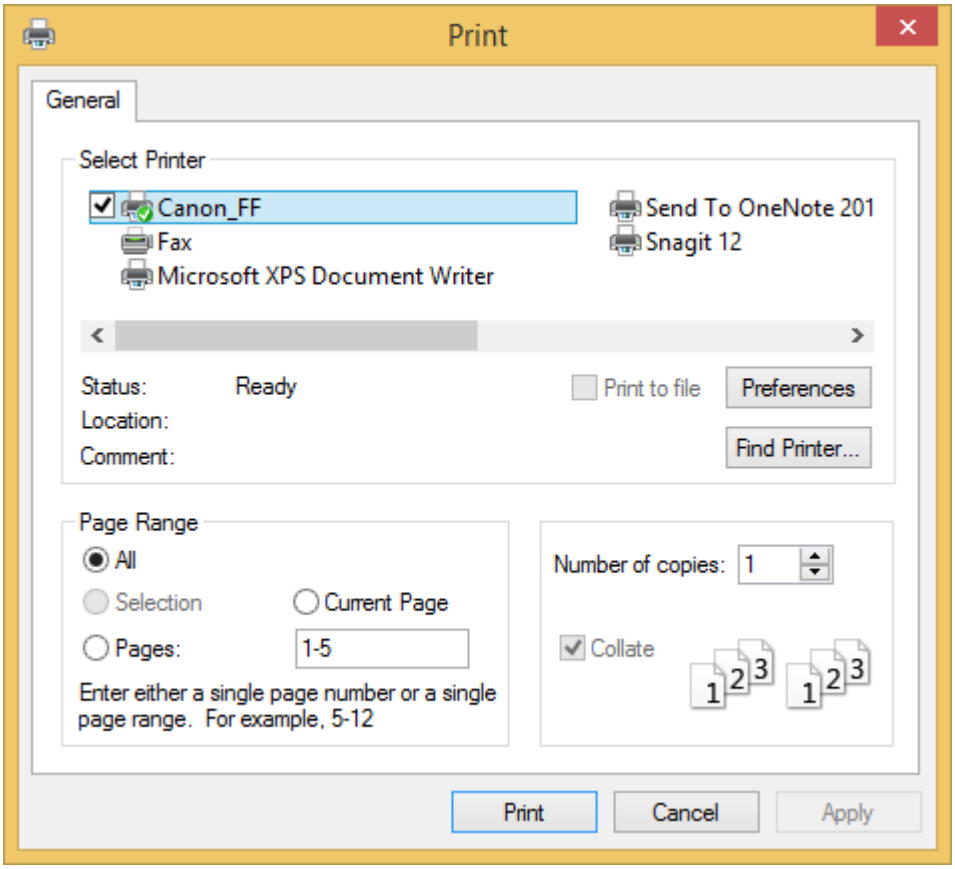
When you create a new report or open an existing report in the designer, the report is displayed in the Design mode.

For previewing the report, click Preview 

or press shortcut key F5.

For printing the report, go to **File** menu and click Print 

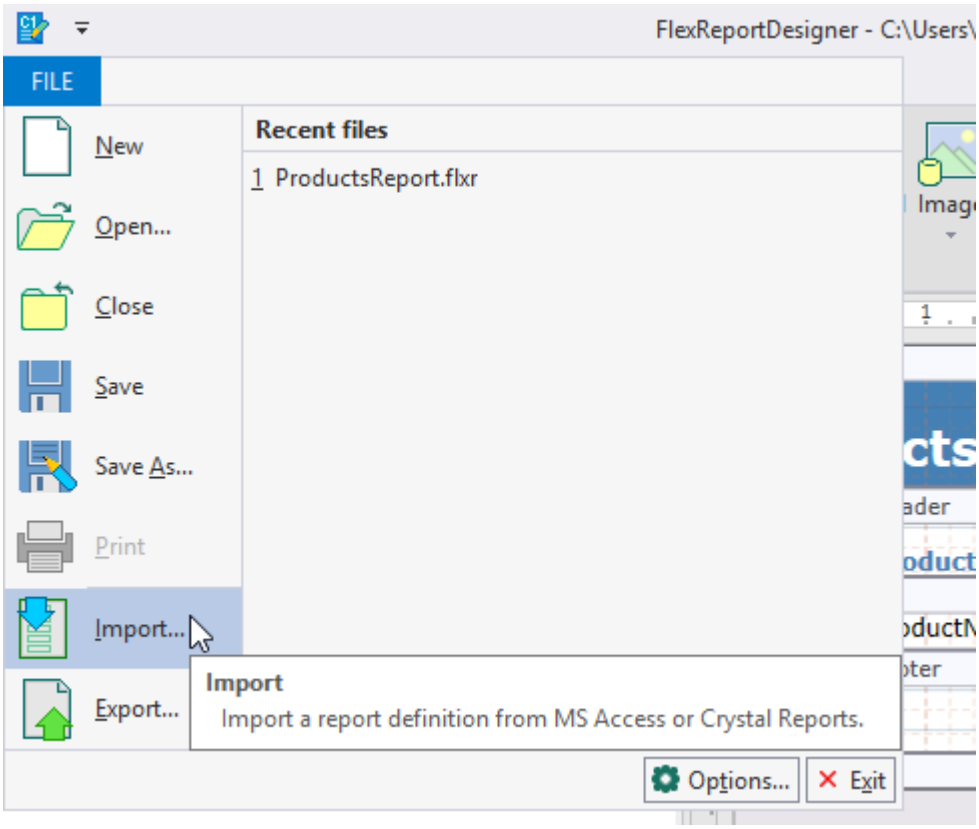
or press shortcut keys Ctrl+P. Note that the Print option is enabled only in **Preview** mode. The Print dialog box lets you specify the settings on the printer.



You can also edit the page settings through the options available in the Page Layout group

Importing Reports in FlexReportDesigner

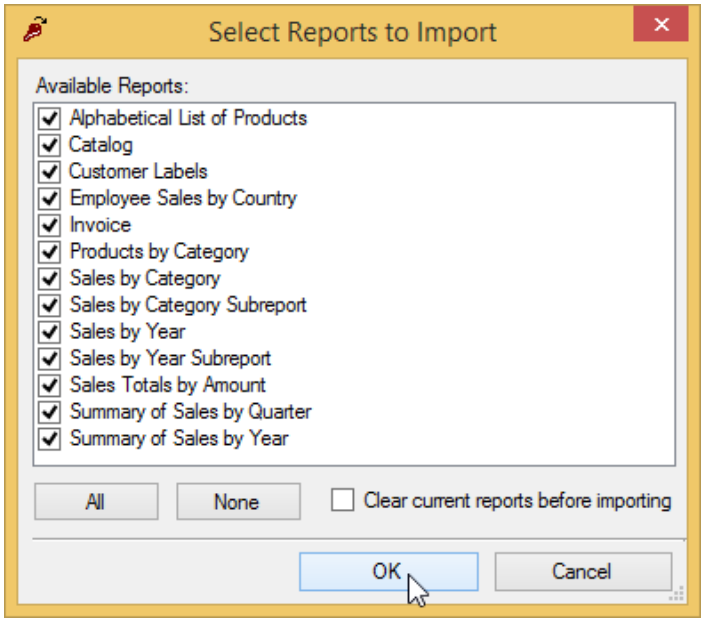
One of the most powerful features of **FlexReportDesigner** application is the ability to import reports created with Microsoft Access (.mdb) and Crystal Reports (.rpt).



To import reports, click the **File** menu and select **Import** from the menu. A dialog box prompts you for the name of the file you want to import.

Importing Microsoft Access Reports

To import a Microsoft Access file, select a Microsoft Access file (MDB or ADP) and the Designer scans the file and shows a dialog box where you can select which reports you would like to import:



The dialog box also allows you to specify if the Designer should clear all currently defined reports before starting the import process.

Note that importing Microsoft Access report requires Access to be installed on the computer. Once the report is imported into the Designer, Access is no longer required.

The import process handles most elements of the source reports, with a few exceptions:

- **Event handler code**

  Access reports can use VBA, macros and forms to format the report dynamically. **C1FlexReport** can do the same things, but it only uses VBScript. Because of this, all report code needs to be translated manually.

- **Form-oriented field types**

  Access reports may include certain fields that are not handled by the Designer's import procedure. The following field types are **not** supported: Chart, CommandButton, ToggleButton, OptionButton, OptionGroup, ComboBox, ListBox, TabCtl, and CustomControl.

- **Reports that use VBScript reserved words**

  Because Access does not use VBScript, you may have designed reports that use VBScript reserved words as identifiers for report objects or dataset field names. This causes problems when the VBScript engine tries to evaluate the expression, and prevents the report from rendering correctly.

  Reserved words you shouldn't use as identifiers include **Date**, **Day**, **Hour**, **Length**, **Minute**, **Month**, **Second**, **Time**, **TimeValue**, **Value**, **Weekday**, and **Year**. For a complete list, please refer to a VBScript reference.

These limitations affect a relatively small number of reports, but you should preview all reports after importing them, to make sure they still work correctly.

Importing the C1NWind.mdb File

To illustrate how the Designer fares in a real-life example, try importing the C1NWind.mdb file. It contains the following 13 reports. (The NWind.xml file that ships with C1FlexReport already contains all the following modifications.)

1. **Alphabetical List of Products**

   No action required.

2. **Catalog**

   No action required.

3. **Customer Labels**

   No action required.

4. **Employee Sales by Country**

   This report contains code which needs to be translated manually. The following code should be assigned to the Group 1 Header OnPrint property:

   To write code in Visual Basic

   Visual Basic

   ```
   If SalespersonTotal > 5000 Then
     ExceededGoalLabel.Visible = True
     SalespersonLine.Visible = True
   Else
     ExceededGoalLabel.Visible = False
     SalespersonLine.Visible = False
   End If
   ```

   To write code in C#

   C#

   ```
   if (SalespersonTotal > 5000)
   {
     ExceededGoalLabel.Visible = true;
     SalespersonLine.Visible = true;
   ```

```
} else
{
  ExceededGoalLabel.Visible = false;
  SalespersonLine.Visible = false;
}
```

5. **Invoice**

   No action required.

6. **Products by Category**

   No action required.

7. **Sales by Category**

   This report contains a Chart control that is not imported. To add a chart to your report, you can use the Chart field.

8. **Sales by Category Subreport**

   No action required.

9. **Sales by Year**

   This report contains code and references to a Form object which need to be translated manually. To replace the Form object in the Data panel, add a "ShowDetails" parameter. Set its DataType property to Boolean and Value property to False:

   Use the new parameter in the report's **OnOpen** event:

   To write code in Visual Basic

   Visual Basic

```
Dim script As String = _
  "bDetails = [Show Details]" & vbCrLf & _
  "Detail.Visible = bDetails" & vbCrLf & _
  "[Group 0 Footer].Visible = bDetails" & vbCrLf & _
  "DetailsLabel.Visible = bDetails" & vbCrLf & _
  "LineNumberLabel2.Visible = bDetails" & vbCrLf & _
  "Line15.Visible = bDetails" & vbCrLf & _
  "SalesLabel2.Visible = bDetails" & vbCrLf & _
  "OrdersShippedLabel2.Visible = bDetails" & vbCrLf & _
  "ShippedDateLabel2.Visible = bDetails" & vbCrLf & _
  "Line10.Visible = bDetails"
C1FlexReport1.Sections.Detail.OnPrint = script
```

   To write code in C#

   C#

```
string script = "bDetails = [Show Details]" +
  "Detail.Visible = bDetails\r\n" +
  "[Group 0 Footer].Visible = bDetails\r\n" +
  "DetailsLabel.Visible = bDetails\r\n" +
  "LineNumberLabel2.Visible = bDetails\r\n" +
  "Line15.Visible = bDetails\r\n" +
  "SalesLabel2.Visible = bDetails\r\n" +
  "OrdersShippedLabel2.Visible = bDetails\r\n" +
  "ShippedDateLabel2.Visible = bDetails\r\n" +
  "Line10.Visible = bDetails";
c1FlexReport1.Sections.Detail.OnPrint = script;
```

   Finally, two more lines of code need to be translated:

   To write code in Visual Basic

Visual Basic

```
Sections ("Detail").OnPrint = _
  "PageHeader.Visible = True"
Sections("Group 0 Footer).OnPrint = _
  "PageHeader.Visible = False"
```

To write code in C#

C#

```
Sections ("Detail").OnPrint =
  "PageHeader.Visible = true";
Sections("Group 0 Footer).OnPrint =
  "PageHeader.Visible = false";
```

10. **Sales by Year Subreport**

    No action required.

11. **Sales Totals by Amount**

    This report contains code that needs to be translated manually. The following code should be assigned to the Page Header OnPrint property:

    To write code in Visual Basic

    Visual Basic

    ```
    PageTotal = 0
    ```

    To write code in C#

    C#

    ```
    PageTotal = 0;
    ```

    The following code should be assigned to the Detail OnPrint property:

    To write code in Visual Basic

    Visual Basic

    ```
    PageTotal = PageTotal + SaleAmount
    HiddenPageBreak.Visible = (Counter = 10)
    ```

    To write code in C#

    C#

    ```
    PageTotal = PageTotal + SaleAmount;
    HiddenPageBreak.Visible = (Counter = 10);
    ```

12. **Summary of Sales by Quarter**

    No action required.

13. **Summary of Sales by Year**

    No action required.

Summing up the information on the table, out of the 13 reports imported from the NorthWind database: nine did not require any editing, three required some code translation, and one had a chart control that required adding a Chart field.

Importing Crystal Reports

To import reports from a Crystal report definition file (.rpt):

1. Click the **File** menu and select **Import**. The **Import Report Definition** dialog box opens and prompts you for the name of the file you want to import.
2. Select a Crystal report definition file (.rpt). The **FlexReportDesigner** application converts the report into the FlexReport format.
3. Save the report. Your Crystal Report is now converted to FlexReport.

Reports bound to internal or external data sources can be imported and run without any changes required to the original data source path.

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

Sections/SubSections

- Report Header
- Page Header
- Group Header
- Detail
- Group Footer
- Page Footer
- Report Footer

Fields

- Textbox
- Chart
- Box
- Line
- Picture
- Checkbox
- Subreport (needs manual correction of path)
- Image Fields - BLOBField (Picture) OLEObject (Picture)
- Database Field
- Cross-section box control

Special Fields

- Time Formats
- Page Numbers

Features

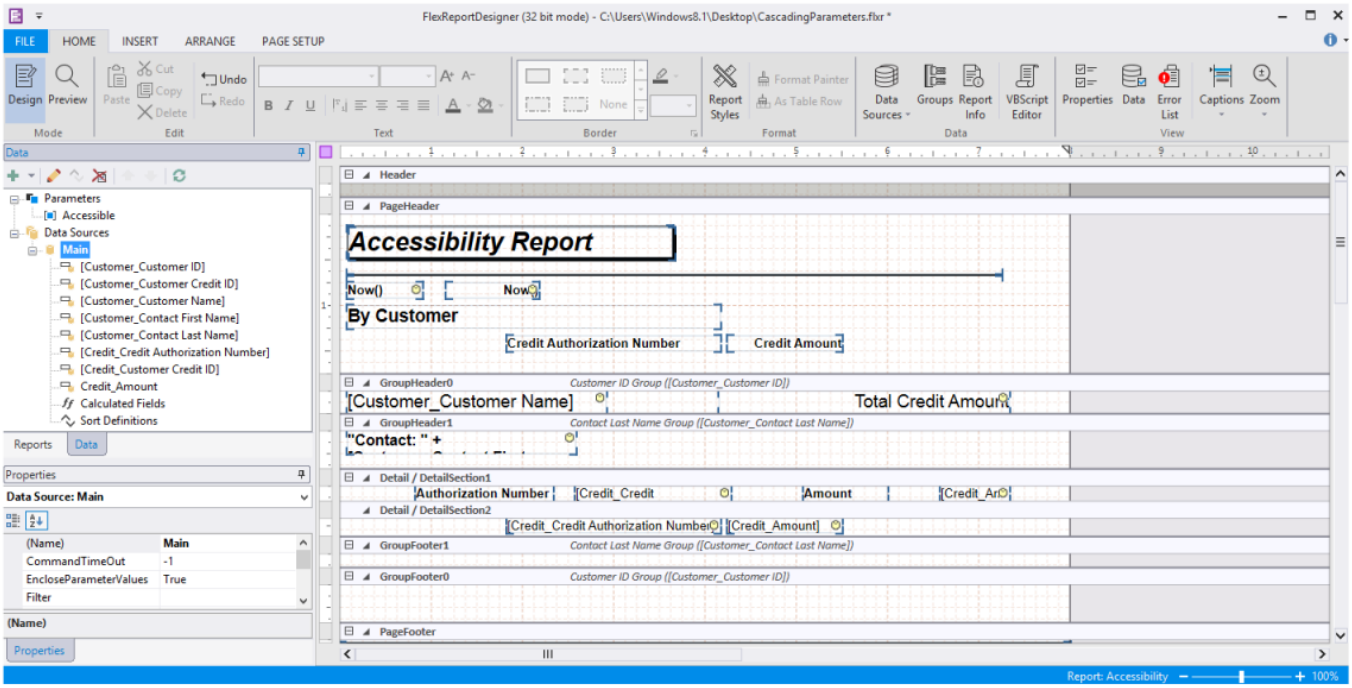
- Complex expressions combining Text+Database Fields (**"Text" + [Expression]**), and Aggregates
- Grouping
- Parameters
- Summary Fields
- Percentage Aggregate
- Textbox Formats (Currency, Date etc)
- KeepTogether (Fields, Sections, Subsections, Groups)
- Multiple join queries
- Hyperlinks
- Text Rotation
- Special order grouping

Formatting

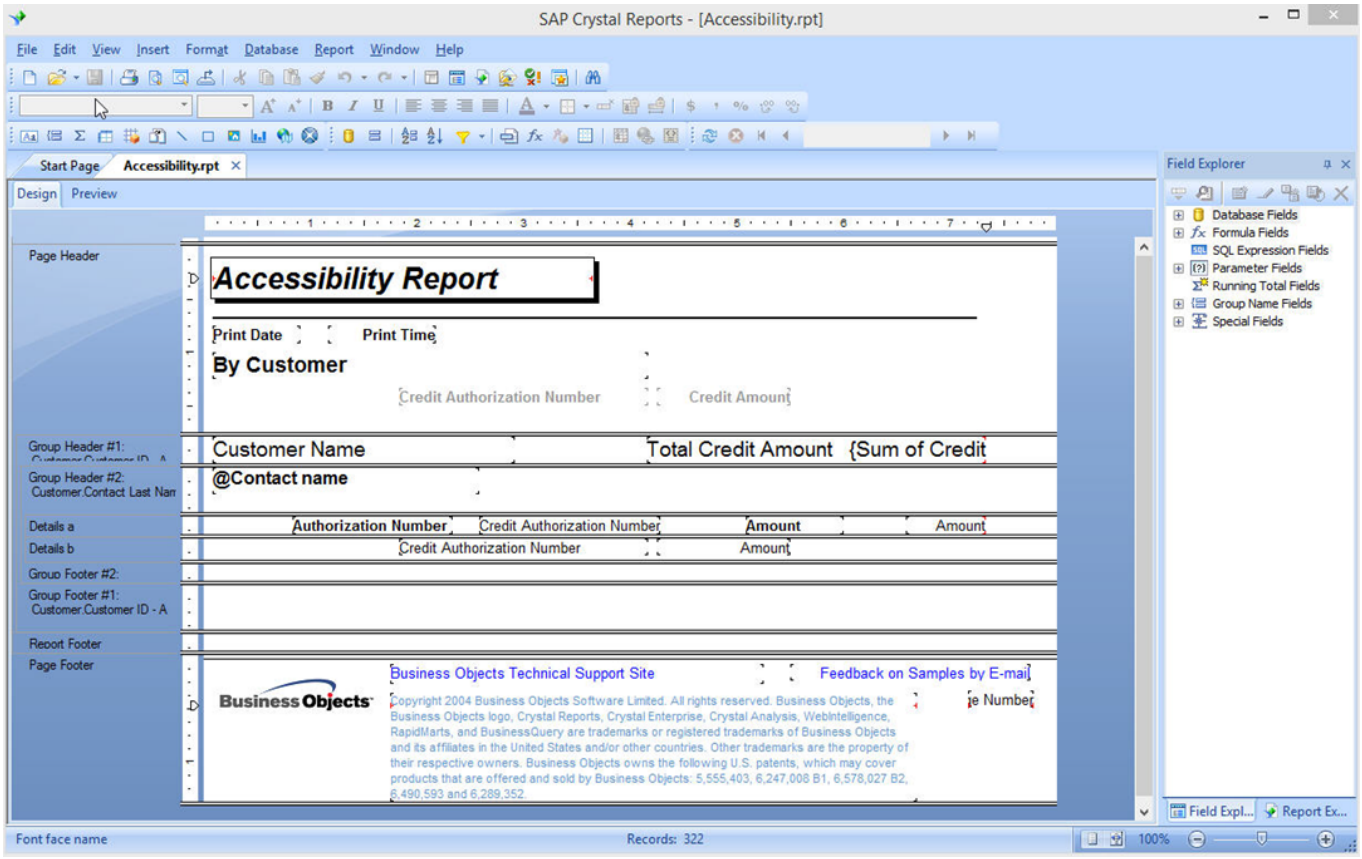
- Text Formatting

- Borders
- Backgrounds

The following image shows a Crystal Report imported in FlexReportDesigner:



The following image shows a crystal report imported in CRReportDesigner:



Note: Before you import the report, please ensure that you have Crystal Reports version 13.0.14.xxxx or later. This version of Crystal Reports should be compatible with Visual Studio installed on your system. Also note that

> if you have Crystal Reports 2013 version installed on the system, then on conversion of Crystal Report to C1FlexReport, the database path to xtreme.mdb will have to be changed manually in order to run the report.

The import process handles most elements of the source reports, with a few exceptions for elements that are not exposed by the Crystal object model or not supported by C1FlexReport. The conversions that are not yet supported in **C1FlexReportDesigner** application are as follows, however, these features are expected to be incorporated in upcoming 2016 releases.

- Fields: Table, CrossTab, OLAPGrid, Map, Flash, and Formula Fields.
- Scripts: Suppress property scripts, complex script expressions, and custom functions.
- Features: Alerts, complex expressions (text and parameter Fields), and hierarchical grouping.
- Properties: Drop Shadow.

Exporting and Publishing a Report

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

Format	Description
Paged HTML (*.html)	Creates one HTML file for each page in the report. The HTML pages contain links that let the user navigate the report.
Plain HTML (*.html)	Creates a single HTML file with no drill-down functionality.
PDF with non-embedded (linked) fonts (*.pdf)	Creates a PDF file that can be viewed on any computer equipped with Adobe's Acrobat viewer or browser plug-ins.
PDF/A with embedded fonts (*.pdf)	Creates a PDF file with embedded font information for extra portability.
RTF (*.rtf)	Creates an RTF file that can be opened by most popular word processors (for example, Microsoft Word, WordPad). It can be saved as Paged or Open XML document.
Microsoft Excel 97 (*.xls)	Creates an XLS file that can be opened by Microsoft Excel.
Microsoft Excel 2007/2010 Open XML (*.xlsx)	Creates an XLS file that can be opened by Microsoft Excel 2007 and later.
Open XML Word (*.docx)	Creates a DOCX file that can be opened by Microsoft Word 2007 and later.
Compressed Metafile (*.zip)	Creates a compressed metafile file, of the type EmfOnly, EmfPlusOnly,and EmfPlusDual.
TIFF (*.tiff), BMP, PNG, JPEG, GIF images	Create image file of type TIFF (Tag Image File Format), BMP (Bitmap Images), PNG(Portable Network Graphic), JPEG or GIF.

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

Export to PDF/A

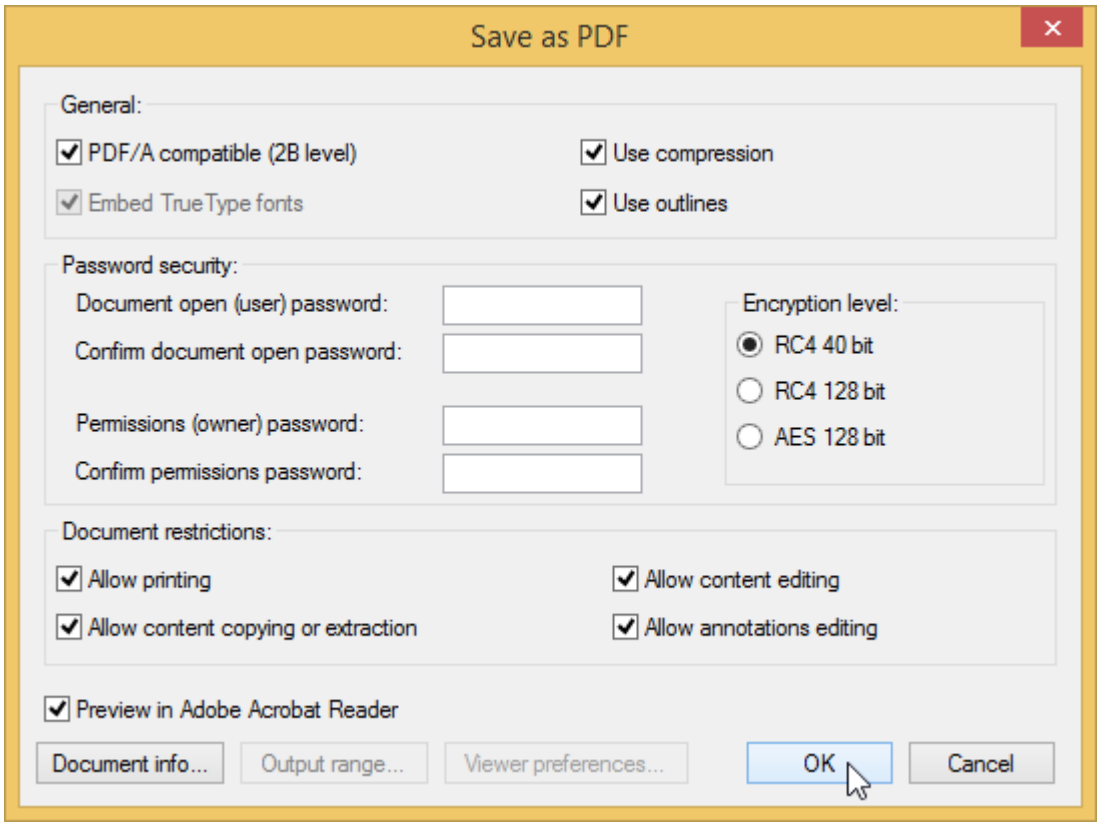
A FlexReport can be exported to a PDF format that is in compliance with the PDF/A standards.

PDF/A is commonly used for creating invoices, brochures, manuals or research reports, and storing the reports in

PDF/A formats. It enables export of JPEG2000 Images, provisions for digital signatures, and support for embedded fonts.

The steps to export to PDF/A format using FlexReportDesigner are as follows:

1. Run **C1FlexReportDeisgner** application.
2. Open the report.
3. Click **File|Export** OR Click **Preview** and from Export group, select **PDF/A** from the **PDF** icon's drop-down. **Export Report to File** dialog box opens.
4. Specify the File name. Save as type is by default PDF/A (*.pdf).
5. Click **Save**. The **Save as PDF** dialog box opens.



6. Fill the dialog box and then click **OK**. Your exported PDF/A document opens in the PDF reader installed on your computer.

You have successfully exported your report that is in compliance with the PDF/A standards!

Note: In 2015v3 release, only PDF/A-2B is supported.

Enhancing Look of FlexReports

A report generally requires some part of it to stand out from the rest. The visual properties such as Background and Border let you transform a plain report to a visually appealing one. To explore these features, see following sections.

Background

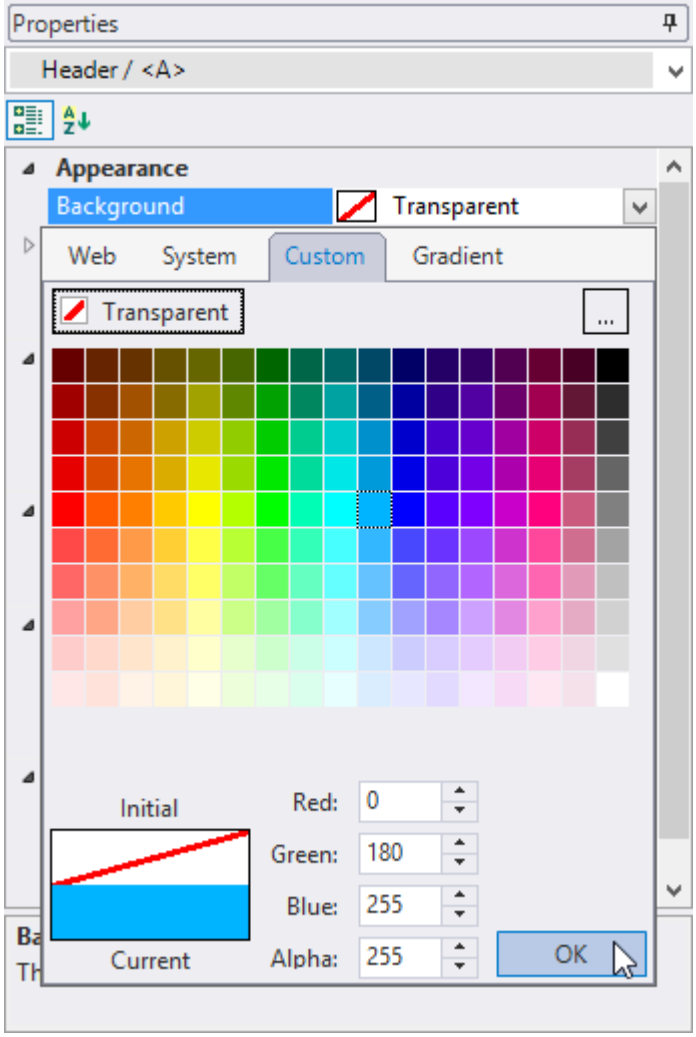
Setting a Background color using FlexReportDesigner is quiet easy. A background color in FlexReport can be set for

Fields, Sections, and Sub-sections. The background colors can be Solid or Gradient.

The following image shows a Report header without any background:

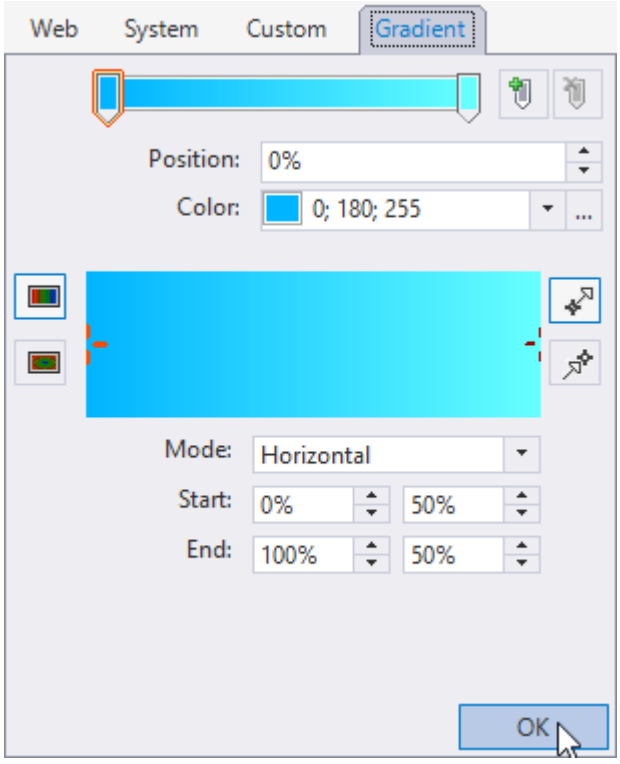
To set a background to the Report Header:

1. In the Design mode, select the **Header** section of the report.
2. From the Properties window, click the drop-down next to **Background** property. The color palette opens.



3. Select a color for the background. You can choose color from the tabs - Web, System, Custom, and Gradient.
4. Preview the report.

5. You can also set Gradient - Linear or Radial - for the background. For obtaining a Gradient, switch to Design mode.
6. Select the **Header** section of the report.
7. From the Properties window, click the drop-down next to **Background** property. Go to Gradient tab and edit the settings from the options available as shown:



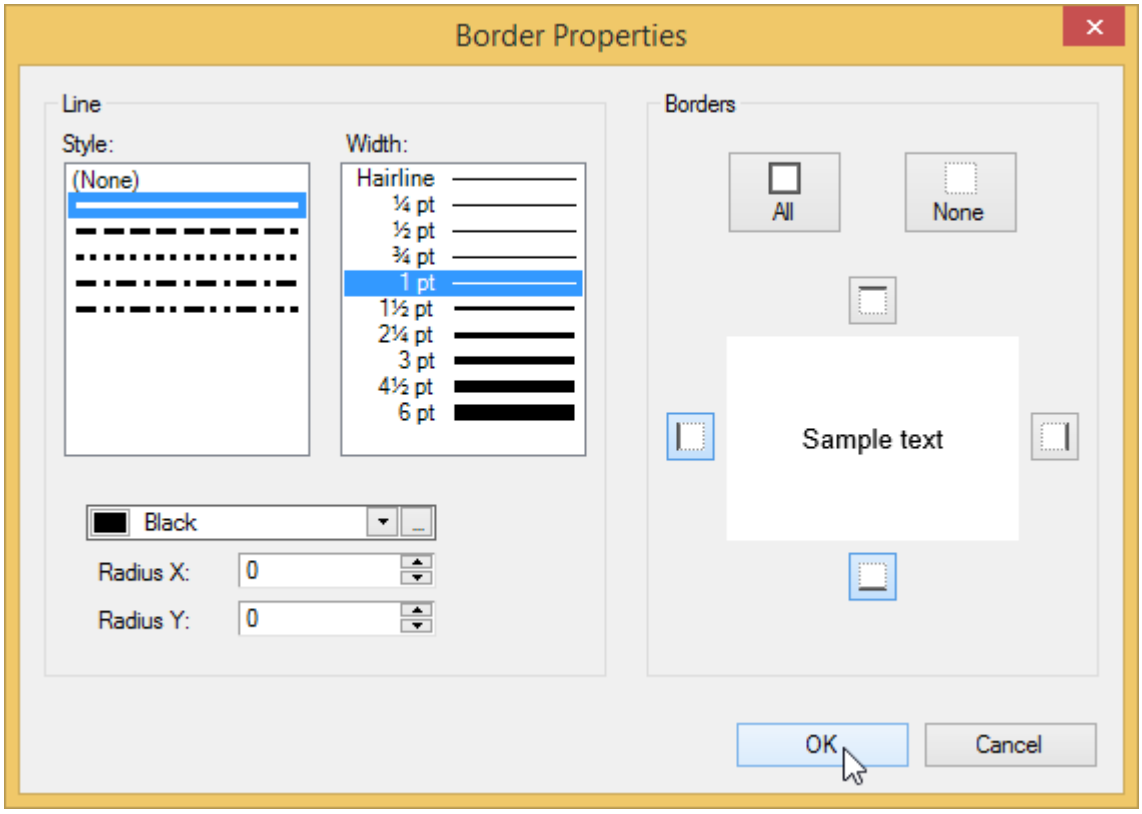
8. Specify the gradient as Linear. You can also drag the slider to change the start and stop for the gradient.
9. Preview the report. The Report Header now looks as follows:

Border

In FlexReportDesigner, you can easily set Borders for Fields, Sections, and Sub-sections.

To set Border around Fields/Section

1. Select the **Field** around which you want borders to appear.
2. From the Properties window, select ellipses next to Borders property. The **Borders Properties** dialog box opens.



3. Select the Line **Style** and Line **Width**; select **Borders** to show - All or - any one or the combination of Left, Top, Right, or Bottom borders. Select the color and click **OK**.
4. Lets insert borders with rounded corners in the Report Header. Select the **Report Header**.
5. Set the Line Style, Line Width, Borders, and color as before.
6. Set the **Radius X** and **Radius Y** for the corners.
7. Preview the report.

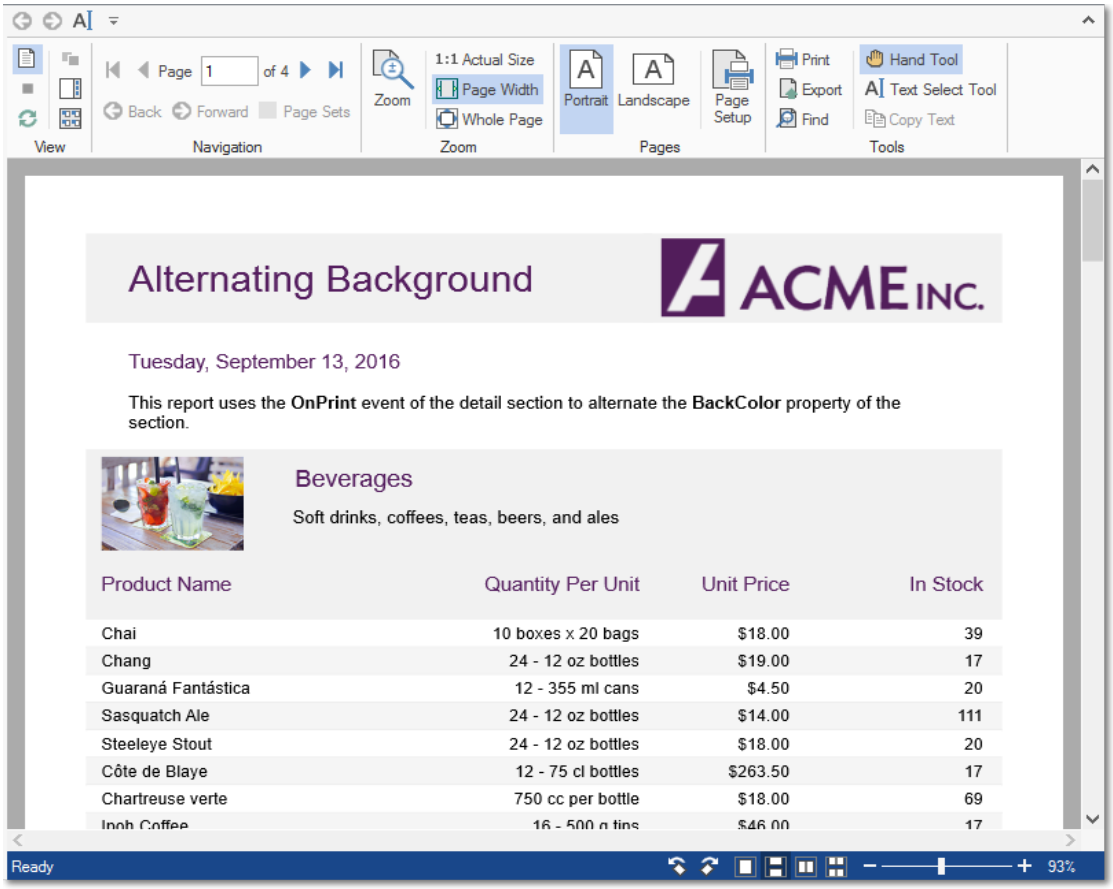
ProductName	ReorderLevel	UnitPrice	UnitsOnOrder
Chai	10	18	0
Chang	25	19	40
Aniseed Syrup	25	10	70
Chef Anton's Cajun Seasoning	0	22	0
Chef Anton's Gumbo Mix	0	21.35	0
Grandma's Boysenberry Spread	25	25	0
Uncle Bob's Organic Dried Pears	10	30	0
Northwoods Cranberry Sauce	0	40	0

For inserting rounded corners note that:

- Range of Radius X is from 0 to field's width.
- Range of Radius Y is from 0 to field's height.

Report and Document Viewer (FlexViewer Control)

The FlexViewer control is a previewing control that can be used to preview different document types such as C1Report, C1FlexReport, SSRS, and PDF document. With an interactive and user friendly UI, you can view parameters, reset them, refresh report rendering, view hierarchy in the outlines, and use bookmarks to jump to report locations. See following topics for more information.



FlexViewer Key Features

The key features of FlexViewer are as follows:

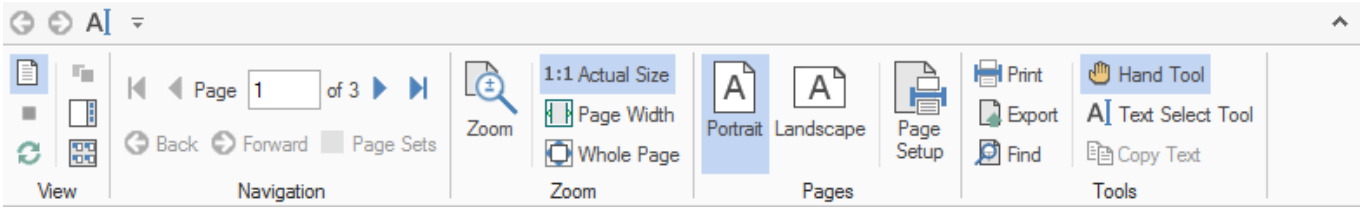
- **User-friendly UI**
  FlexViewer has an interactive and user friendly full-featured modern Ribbon-based UI that helps preview different document types such as FlexReport, SSRS, and PDF document. The ribbon contains command buttons clustered in groups.
- **Use/Reset Parameters**
  With FlexViewer, you can view parameters and reset them as well. You can in enter data parameters of your choice to display a report. FlexViewer supports String, Boolean, Date, Integer, and Float type parameters.
- **Refresh and Stop Rendering**
  FlexViewer provides you options to Refresh and Cancel Report rendering through Refresh and Stop button.
- **View Thumbnails and Hierarchy**
  FlexViewer allows you to view the thumbnails of the report pages and the hierarchy in the outlines that allows you to jump to a required location.
- **Page Navigation**
  Page navigation is available in the Navigation group of the FlexViewer control, which contains a set of command buttons allowing you can navigate through the report pages and if you want to jump to a specific

page number then, you can type the page number in the page number textbox.

- **Zoom in/out with Ease**
  FlexViewer allows you to access Zoom dialog box using Zoom button. It gives you the following options in the dialog box:
  - Actual Size – Shows the pages in their actual size.
  - Page Width - Fits the page to the width of the preview window.
  - Whole Page - Fits the whole page in the preview window.
  - Custom - Sets the custom zoom % for the page.
  - Zoom % - Shows the percentage you set to zoom the page.
  - Wide Layout by Default - Indicates whether a multi-column layout is applied to wide reports by default.
  - Facing Pages, Cover on Right - Shows report pages side by side.
  - Continuous - Shows the pages in continuity.
  - Rotate View - Allows you to choose the angle of rotation of the page.
  - Page Columns - Allows you to choose the number of page columns to show in the preview window.
  - Page Rows - Allows you to choose the number of page rows to show in the preview window.
- **Print**
  You can also use the FlexViewer control to print your reports using the Print button which provides you with a Print Dialog box containing standard printer settings.
- **Export**
  FlexViewer allows you to export your reports and documents to the various formats, such as DOCX, RTF, Open XML Word, Open XML Excel, HTML, PDF, GIF, JPEG, PNG, BMP, and TIFF.
- **Tools**
  FlexViewer provides you Hand tool, Text Select tool, and copy text tool that can be used to scroll in the window, select text to copy, and copy the selected text.

FlexViewer Toolbar

The FlexViewer toolbar appears on the top of the control as shown in the image below:

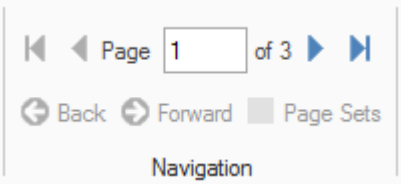
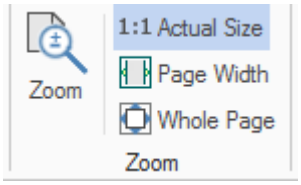
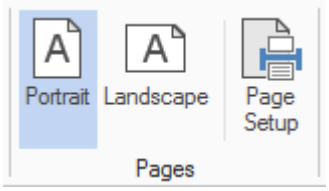
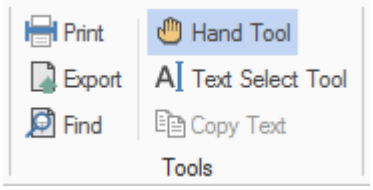


It consists of the following shortcut command buttons at the top of the toolbar:

Command Button	Command Button Name	Description
	Go back	Allows you to go back to the previously opened page
	Go forward	Allows you to go forward to the next page
	Select text to copy	Allows you to select a text to copy

All the command buttons are divided in groups in FlexViewer toolbar as listed below:

Group	Command Buttons inside Group	Description
View	Print Layout, Stop, Refresh, Parameters, Outline, Thumbnails	Allows you to view the report pages in print layout, refresh the view and stop the preview. It also allows you to view the parameters, outlines and thumbnails of the report pages

	Go to the first page of the document, Go to the previous page of the document, The current page number, Number of pages in the current document, Go to the next page of the document, Go to the last page of the document, Back, Forward, Page Sets	Allows you to navigate to the first, last, previous, next page of the report. It shows you the current page number and lets you move back and forward. The Page Sets button allows you to navigate to the page sets instead of pages
	Zoom, Actual Size, Page Width,Whole Page	Allows you to open Zoom dialog box using Zoom button. Actual Size allows you to zoom the page to 100% of the normal size, Page Width allows you to zoom pages to fill the window width, and Whole page allows you to zoom pages to fit within the window
	Portrait, Landscape, Page Setup	Allows you to change the orientation of the report pages to portrait or landscape mode. Page Setup changes the page settings of the current report.
	Print, Export, Find, HandTool, TextSelect Tool, Copy Text	Allows you to print and export reports. Find button allows you to find text in the report, Hand Tool allows you to scroll in window using the mouse, Text Select Tool selects text to copy, and Copy Text button allows you to copy the selected text

Rotate View of Reports

FlexViewer provides you the flexibility to rotate the view of reports to different angles according to your requirements. To rotate view of a report to various degrees of rotation, you can set the RotateView property of C1FlexViewer class. The RotateView property accepts the following values from the FlexViewerRotateView enum describing the rotation angle of the view:

- **NoRotation:** Rotation is not applied to the view.
- **Rotation180:** Allows rotation of the view by 180 degrees.
- **Rotation90Clockwise:** Allows rotation of the view by 90 degrees in clockwise direction.
- **Rotation90CounterClockwise:** Allows rotation of the view by 90 degrees in counter-clockwise direction.

Rotate View of Report at Design Time

You can rotate the view of reports in FlexViewer at design time by performing the following steps:

1. Right-click on the **FlexViewer** control and select properties.
2. In **Properties** window, select a value for the RotateView property from the provided list to rotate the report at different angles.

Rotate View of Report Programmatically

To rotate view of a report, at first you need to create and load a report and then preview it in the **FlexViewer** control. After doing so, you can use FlexViewerRotateView enum to rotate view of a report. The following code illustrates the

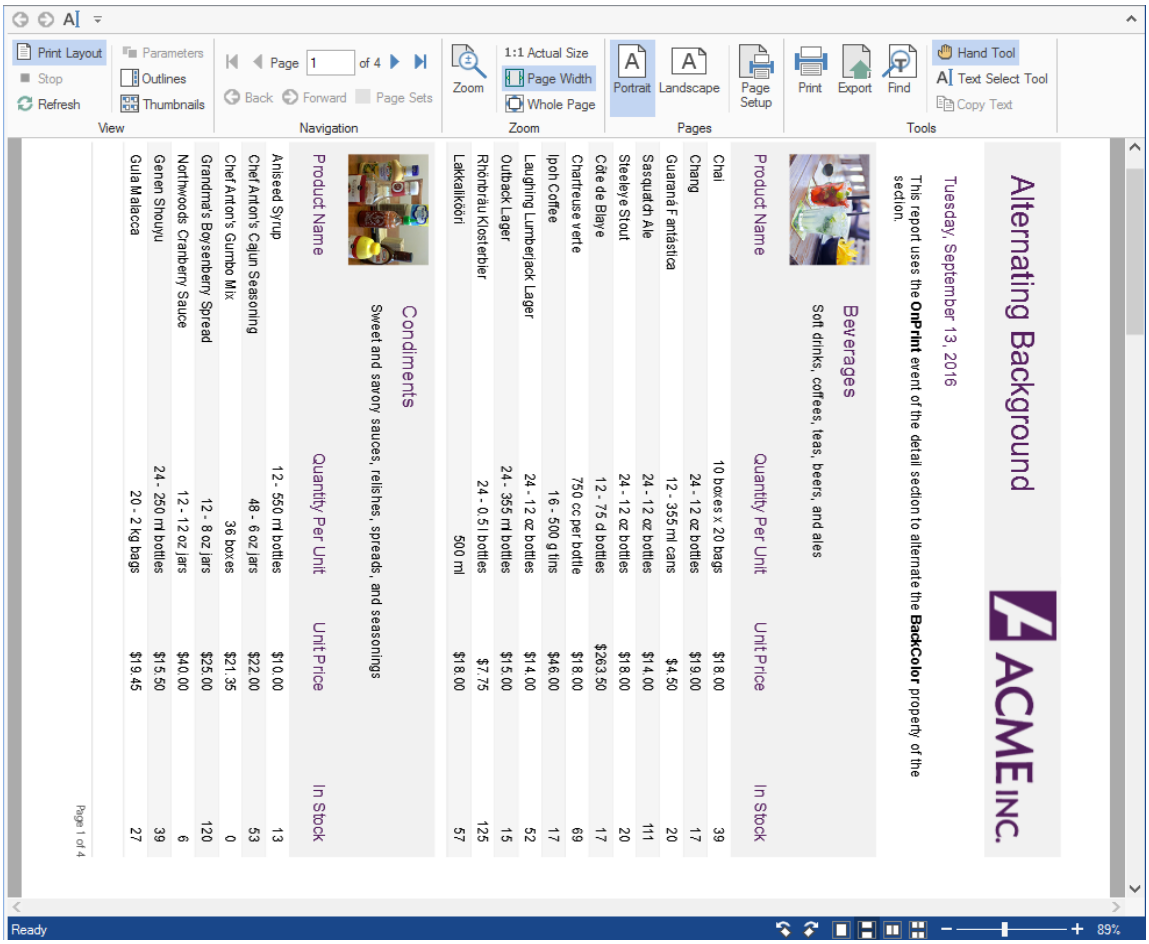
use of **FlexViewerRotateView** enum:

Visual Basic

```
c1FlexViewer1.RotateView = FlexViewerRotateView.Rotation90Clockwise
```

- **C#**

```
c1FlexViewer1.RotateView = FlexViewerRotateView.Rotation90Clockwise;
```



Binding FlexReport with FlexViewer

To render a report, you need to bind the report to the **FlexViewer** control. Once the report definition has been created, a data source defined, and loaded into the C1FlexReport component, you can render the report to the printer, to the preview control - **FlexViewer**, or export to different file formats.

To preview the report in the **FlexViewer** control, the steps are as follows:

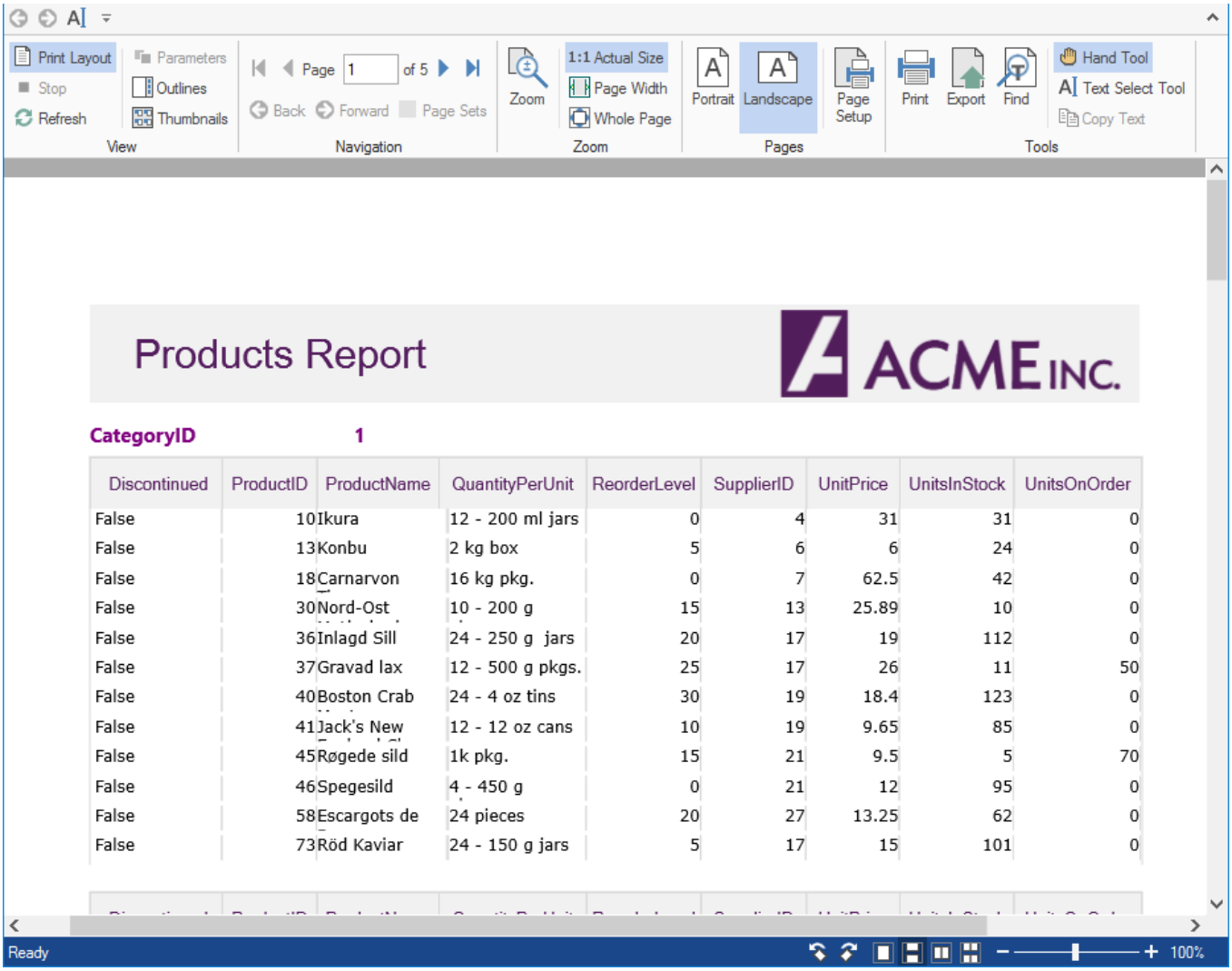
1. From the Toolbox, double-click the **FlexViewer** control to add it to your project.
2. From the Properties window, set the **C1FlexViewer.Dock** property to Fill.
3. Select the Windows Form with your mouse and drag to resize it. For this example, we resized the Form to 600x500 so it better reveals the preview panel.
4. Double-click the form and enter the following code in the Form_Load event handler:

Visual Basic

```
'load report definition
c1FlexReport1.Load("..\..\Products Report.flxr", "Products Report")
'preview the report
c1FlexViewer1.DocumentSource = c1FlexReport1
```

- **C#**

```
//load report definition
c1FlexReport1.Load(@"..\..\Products Report.flxr", "Products Report");
//preview the report
c1FlexViewer1.DocumentSource = c1FlexReport1;
```



FlexReport Samples

With the **C1Studio** installer, you get FlexReport samples that help you understand the implementation of the product. The C# and VB samples are available at the default installation folder - \Documents\ComponentOneSamples\Winforms\C1FlexReport.

The list of available C# samples is as follows:

Sample	Description
AddScriptObject	This sample demonstrates how to add custom objects to C1FlexReport's script engine.
AdHocSorting	This sample demonstrates how to select the sorting criteria before rendering the report.
CustomFields	This sample demonstrates how to load all CustomFields in C1FlexReport.
ExternalDataSource	This sample demostrates how to implement the IC1FlexReportRecordset and IC1FlexReportExternalRecordset interfaces. These interfaces allow an assembly to be used as a custom record source for C1FlexReport.
FlexCommonTasks	This sample demonstrates how to use parameters, charts, subreports, and many other features in FlexReport. It uses FlexCommonTasks.flxr file.
FlexReport Explorer	This sample loads a categorized list of reports, with categories such as enterprise, financial, medical and so on. Some reports demonstrate specific features of FlexReport, such as conditional formatting or watermarks, while others combine various features to produce reports that might be used in different real-life applications.
FlexReportViewer	This sample demonstrates various features of FlexViewer.
MapReports	The sample renders and previews several sample reports demonstrating the Map custom field.
ODataRecordset	This sample demonstrates how to use data provided by the OData service in the C1FlexReport.
PdfViewer	This sample shows how the C1PdfDocumentSource component can be used with C1FlexViewer to view PDF document.
SsrsViewer	This sample demonstrates how C1SSRSDocumentSource component can be used with C1FlexViewer to browse the reports tree available on a SSRS server, and to preview individual reports.
SubReportDataSource	This sample demonstrates how to use custom data sources with subreports in the C1FlexReport component.
Xml2FlxrConverter	This sample allows the user to select one or more .XML files containing C1Report report definitions, and to convert them to the new FlexReport's .FLXR file format.
ZipReport	This sample demonstrates how to compress and encrypt report definition files using the C1FlexReport and C1Zip components.

The list of available VB samples is as follows:

Sample	Description
ExternalDataSource	This sample demonstrates how to implement the IC1FlexReportRecordset and IC1FlexReportExternalRecordset interfaces. These interfaces allow an assembly to be used as a custom record source for C1FlexReport.

FlexCommonTasks	This sample demonstrates how to use parameters, charts, subreports, and many other features in FlexReport. It uses FlexCommonTasks.flxr file.
FlexReport Explorer	This sample loads a categorized list of reports, with categories such as enterprise, financial, medical and so on. Some reports demonstrate specific features of FlexReport, such as conditional formatting or watermarks, while others combine various features to produce reports that might be used in different real-life applications.
FlexReportViewer	This sample demonstrates various features of FlexViewer.
SsrsViewer	This sample demonstrates how C1SSRSDocumentSource component can be used with C1FlexViewer to browse the reports tree available on a SSRS server, and to preview individual reports.

Task Based Help

The task-based help assumes that you are familiar with programming in .NET, have a basic knowledge of reports, and know how to use controls in general. By following the steps outlined in the help, you will be able to create projects demonstrating a variety of FlexReport features, and get a good sense of what C1FlexReportDesigner can do.

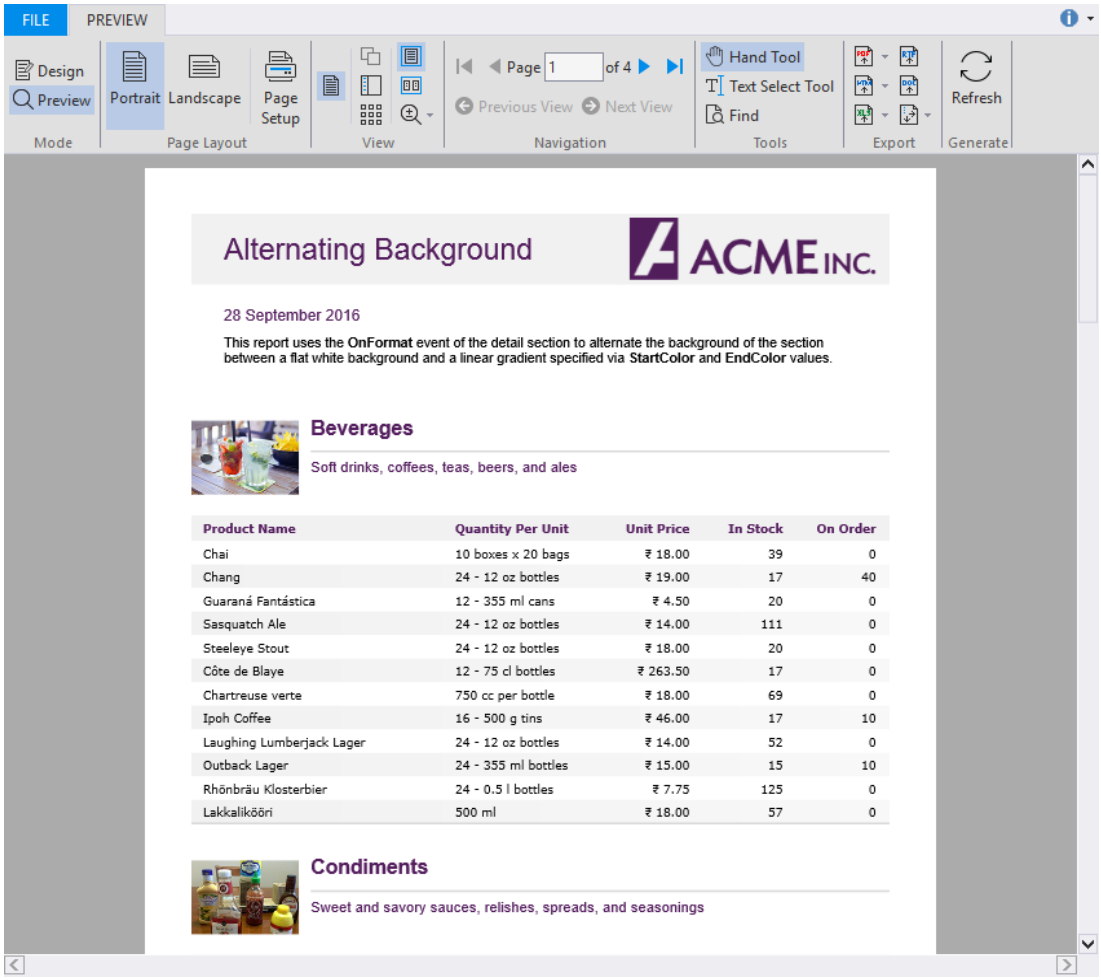
This section's topics have pre-built reports that illustrate them. The pre-built reports can be found in the FlexReportCommonTasks.flxr report definition file, which if you have installed the WinForms Edition, can be found in the Documents or MyDocuments folder in the **ComponentOne Samples\Winforms\C1FlexReport\CS\FlexCommonTasks** folder.

Adding Alternating Background

To create a report with alternating background color, the **OnFormat** property of the **Detail** section is used.

Complete the following steps to create a report with alternating background:

1. Open the **C1FlexReportDesigner**.
2. Create a new report or open an existing report. Once you have the report in the **C1FlexReportDesigner**, you can modify the report properties.
3. Switch to **Design** mode to begin editing the report.
4. In Design mode, select the report from the drop-down list above the Properties window.
5. Go to the **OnOpen** property of the report and enter **cnt = 0**. This initializes the 'cnt' variable.
6. Select **Detail** section from the drop-down list above the Properties window or from the design area.
7. Go to the **OnFormat** property and then click the ellipses button. The **VBScript Editor** appears. Enter the following VBScript expression in the editor and click **Done**:
```
cnt = cnt + 1
if cnt mod 2 = 0 then
  Detail.Background.StartColor = Rgb(238, 214, 200)
  Detail.Background.EndColor = Rgb(238, 200, 177)
else
  Detail.Background.StartColor = vbWhite
  Detail.Background.EndColor = vbWhite
endif
```
8. Click the **Preview** button to preview the report with alternating background.



Note: For the complete report, see report 'Alternating Background' in the **FlexCommonTasks.flxr** report definition file, which is available in the **ComponentOne Samples\Winforms\C1FlexReport\CS\FlexCommonTasks** folder. The data base used is **C1NWind.mdb** which is also available in the **ComponentOne Samples** folder.

Adding Conditional Formatting

In some cases you may want to change a field's appearance depending on the data it represents. This can be done using parameters.

For example, you may want to highlight some data fields depending on a condition, then, you can define two parameters - one to define the condition and other to highlight the data fields depending on the condition.

Complete the following steps to create a report with conditional formatting:

1. Open the **C1FlexReportDesigner**.
2. Create a new report or open an existing report.
3. Switch to **Design** mode to begin editing the report.
4. Add two Parameters - 'pCondition' and 'pHighlightColor'.
5. Set **DataType** of 'pCondition' as Integer and 'pHighlightColor' as String.
6. Specify the properties for each parameters in **AllowedValuesDefinition** property as follows:

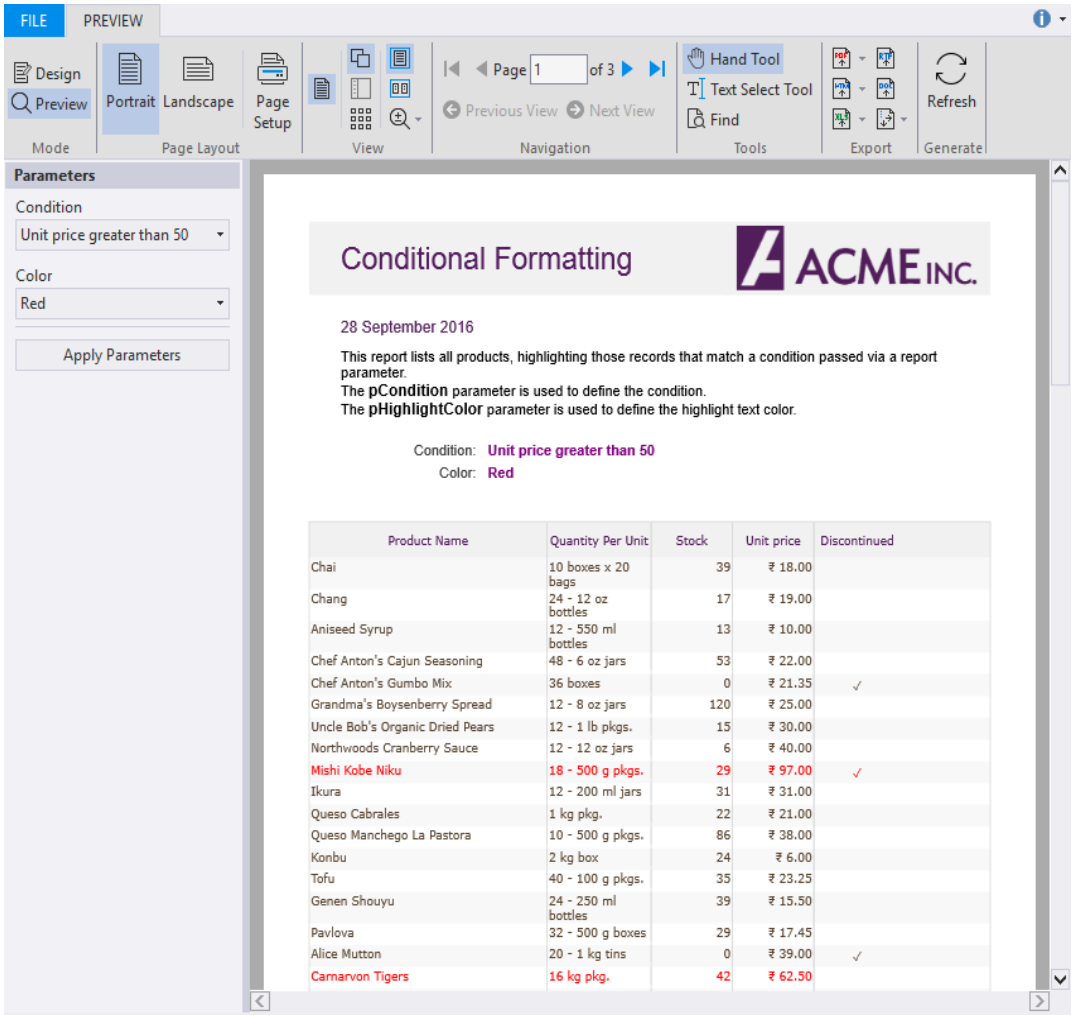
	pCondition	pHighlightColor
Values	Label -Unit price greater than 50, Value - CInt(1)	Label - Red, Value - Red

	Label -Unit price greater than 100, Value - CInt(2)	Label - Green, Value - Green

7. In Design mode, select **Detail** from the drop-down list above the Properties window (since this section contains the fields to add conditional formatting to).
8. Go to the **OnFormat** property and click the ellipses next to it.
9. The **VBScript Editor** appears. Enter the following VBScript expression in the editor:

```
row = row + 1
if row mod 2 = 1 Then
  rowColor = Rgb(238, 214, 200)
Else
  rowColor = "Transparent"
EndIf
doHighlight = False
if pCondition = 1 Then
  doHighlight = UnitPrice > 50
ElseIf pCondition = 2 Then
  doHighlight = UnitPrice > 100
ElseIf pCondition = 3 Then
  doHighlight = UnitsInStock > 50
EndIf
if doHighlight Then
  textColor = pHighlightColor
Else
  textColor = Rgb(90, 70, 50)
EndIf
Detail.BackColor = rowColor
for i = 0 to Detail.Fields.Count - 1
  Detail.Fields(i).BackColor = rowColor
  Detail.Fields(i).ForeColor = textColor
Next
```

10. Preview the report.



Note: For the complete report, see report 'Conditional Formatting' in the **FlexCommonTasks.flxr** report definition file, which is available in the **ComponentOne Samples\Winforms\C1FlexReport\CS\FlexCommonTasks** folder. The data base used is **C1NWind.mdb** which is also available in the **ComponentOne Samples** folder.

Specifying Custom Paper Size

By default, C1FlexReport creates reports using the default paper size on the default printer.

You can specify the paper size and orientation using the PaperSize and Orientation properties. However, C1FlexReport checks that the selected paper size is available on the current printer before rendering, and changes to the default paper size if the selected setting is not available.

If you want to specify a certain paper size and use it regardless of the printers available, set the PaperSize property to **Custom**, and set the Layout.CustomWidth and Layout.CustomHeight properties to the page dimensions (in *twips*).

To specify a custom paper size of 8.5" x 25" for your report using FlexReportDesigner:

1. Open the **C1FlexReportDesigner**.
2. Create a new report or open an existing report. Once you have the report in the **C1FlexReportDesigner**, you can modify the report properties.
3. In the Design mode, select your report from the drop-down list above the Properties window.

4. Go to **Layout** and expand the property node to view all available properties.
5. Set the following properties:

- **CustomHeight**=36000
- **CustomWidth**=12472
- **PaperSize**=Custom

When specified this way, the custom paper size is used regardless of what printers are installed and what paper sizes are actually available.

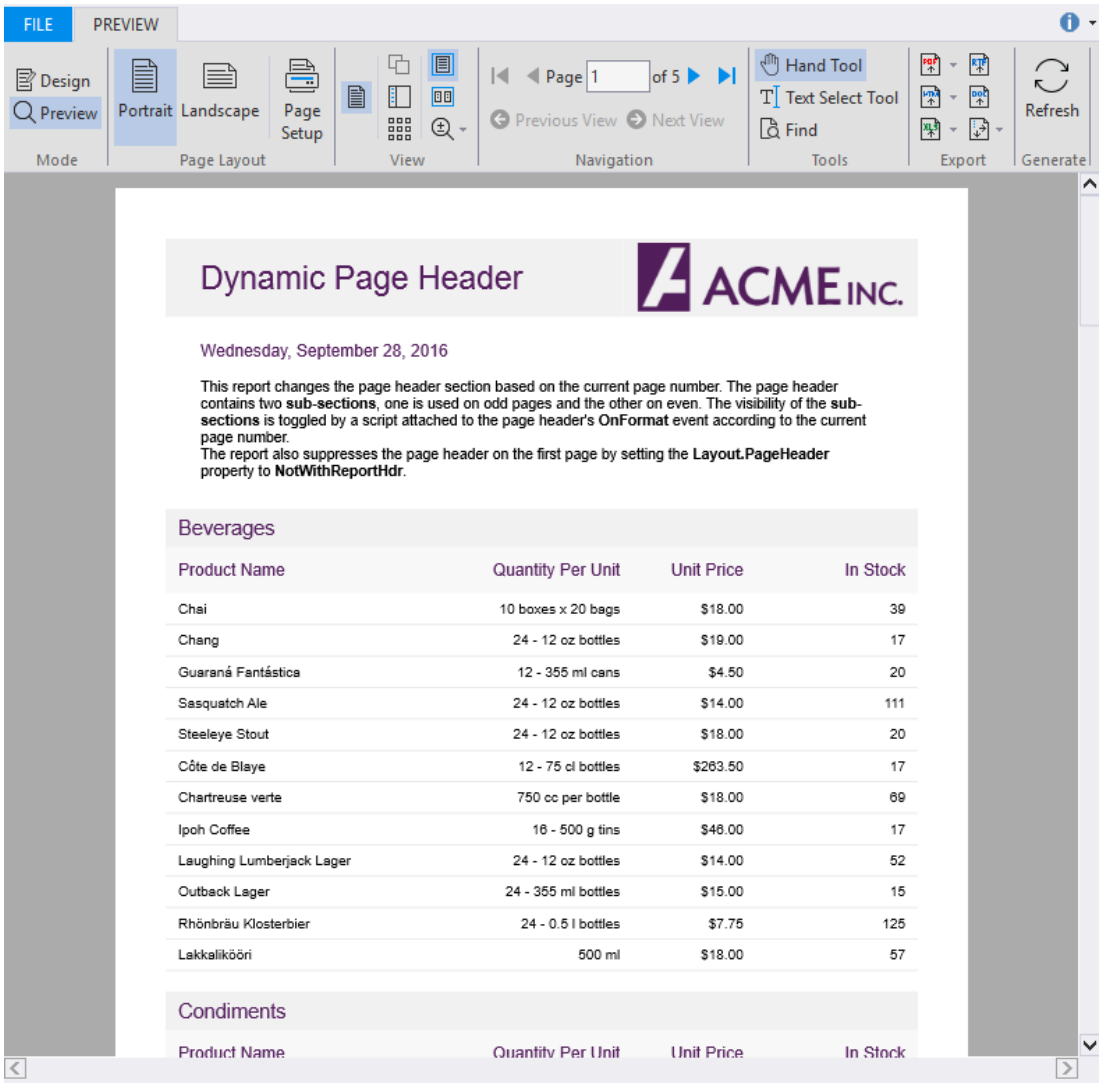
> **Note:** For the complete report, see report 'Custom Paper Size' in the **FlexCommonTasks.flxr** report definition file, which is available in the **ComponentOne Samples\Winforms\C1FlexReport\CS\FlexCommonTasks** folder. The data base used is **C1NWind.mdb** which is also available in the **ComponentOne Samples** folder.

Adding Dynamic Page Header

You may want to generate dynamic page header, that is different page headers on different pages of a report, depending on a condition. This can be done using **OnFormat** property of **PageHeader**.

For example, you want to display different report headers on even and odd pages of a report. Complete the following steps to create a report with dynamic page header:

1. Open the **C1FlexReportDesigner**.
2. Create a new report or open an existing report.
3. Switch to **Design** mode to begin editing the report.
4. Add two subsections in the PageHeader section, say OddPageHeader and EvenPageHeader. OddPageHeader section will be visible for odd pages and EvenPageHeader will be visible for even pages.
5. Add text and/or paragraph fields in each of the above subsections that contain expressions for displaying the headers on odd and even pages.
6. Go to the **OnFormat** property of **PageHeader** and click the ellipses next to it.
7. The **VBScript Editor** appears. Enter the following VBScript expression in the editor:
```
odd = (page mod 2 <> 0)
PageHeader.SubSections(0).Visible = odd
PageHeader.SubSections(1).Visible = not odd
```
8. To suppress the page header on first page of a report, set the Layout.PageHeader property of the report to NotWithReportHdr.
9. Preview the report.



Note: For the complete report, see report 'Dynamic Page Header' in the **FlexCommonTasks.flxr** report definition file, which is available in the **ComponentOne Samples\Winforms\C1FlexReport\CS\FlexCommonTasks** folder. The data base used is **C1NWind.mdb** which is also available in the **ComponentOne Samples** folder.

Creating a Gutter Margin

Gutter margins are extra space added to the margins next to the binding. They make it easier to bind the pages into folders, brochures, and so on.

To add a gutter margin to a report, you should increase the left margin on odd pages. This can be done by increasing the value of Layout.MarginLeft on odd pages and using the default value on even pages.

To create gutter margin in a report, complete the following steps:

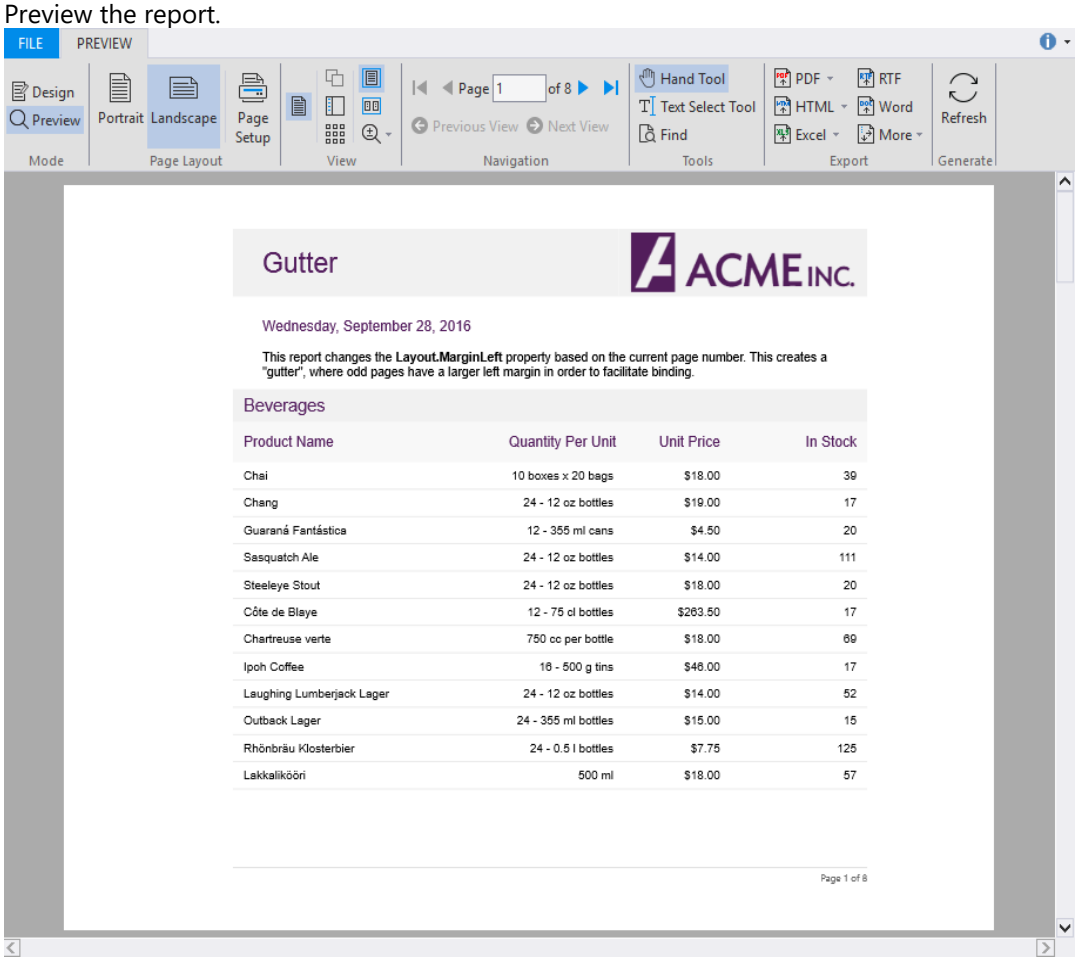
1. Open the **C1FlexReportDesigner**.
2. Create a new report or open an existing report.
3. In Design mode, select **Detail** from the Properties window drop-down list. This reveals the section's available properties.
4. Go to the report's **OnOpen** property and then click the ellipsis button. The **VBScript Editor** appears. Enter the following VBScript expression in the editor:

```
cnt = 0
gutter = report.layout.marginleft
marginRight  = report.layout.marginRight
```

5. Go to the report's **OnPage** property and then click the ellipsis button. The **VBScript Editor** appears. Enter the following VBScript expression in the editor:

```
if page mod 2 = 1 then
  Report.layout.marginleft = gutter
  Report.Layout.marginright = marginRight
else
  Report.layout.marginleft = gutter - 1440
  Report.Layout.marginright = marginRight + 1440
EndIf
```

6. Preview the report.



Note: For the complete report, see report 'Gutter' in the **FlexCommonTasks.flxr** report definition file, which is available in the **ComponentOne Samples\Winforms\C1FlexReport\CS\FlexCommonTasks** folder. The data base used is **C1NWind.mdb** which is also available in the **ComponentOne Samples** folder.

Grouping and Sorting

This topic uses "Groups and Sorting" report to demonstrate how grouping, group total, subtotal, and sorting are performed in a report.

Grouping

Grouping allows you to create groups of records based on common attributes of the records. For example, in this report, all the records having same EmployeeID are grouped together to form EmployeeGroup. For more information on grouping feature in FlexReport, refer Grouping Data.

This report contains two groups, namely EmployeeGroup and OrderGroup with EmployeeID and OrderID as their respective grouping criterion. Here, EmployeeGroup is added first and then the OrderGroup, so the data is first grouped on the basis of Employee ID and then the subgroups are formed within these groups on the basis of Order ID.

The report also calculates the group totals and subtotals. You can calculate the total or subtotal using **=Sum(OrderSum)** expression in a TextField. When a TextField with this expression is dropped in the EmployeeGroup header section, it calculates the total sales for all the orders by an employee, which is the group total. Similarly, when a TextField with the same expression is dropped in the OrderGroup footer section, it calculates the group subtotal.

Sorting

Sorting allows you to organize data in ascending or descending order. In FlexReport, the groups are sorted using group expressions. However, you can change the manner in which the groups are sorted using Group.SortExpression property. The expression may contain aggregate functions, for example in this report the groups are sorted using Sum(OrderSum) expression which contains an aggregate function. For more information on sorting, refer Sorting Data.

In this report, the EmployeeGroup is sorted on the basis of total sales in descending order. However, you can also sort the group alphabetically. In addition, you can provide the sorting options in **Parameters** Panel too. Please refer the steps below to see how this can be done.

To create the report in FlexReportDesigner

- **Step 1: Create a report from scratch**
- **Step 2: Connect the report to a data source**
- **Step 3: Add calculated fields and parameters**
- **Step 4: Group and sort data**
- **Step 5: View the report**

Step 1: Create a report from scratch

1. Open the **C1FlexReportDesigner** and go to **FILE|New**.
2. Click **New Report** drop down from the **Reports** tab and select **Empty Report** to create a report.

Step 2: Connect the report to a data source

1. Switch to the **Data** tab, right-click the Main data source and choose **Edit** to open and the **Data Sources** Wizard and start editing.
2. Select **OLEDB Data Provider** from the **Data provider** drop-down and click the ellipsis button next to the **Connection string** textbox to select the C1NWind.mdb file.
3. Specify the following Sql statement in the **Sql statement** tab:

```
SELECT Employees.EmployeeID as EmployeeID,
       Employees.FirstName as FirstName,
       Employees.LastName as LastName,
       Orders.OrderID as OrderID,
       Orders.OrderDate as OrderDate,
       Orders.ShippedDate as ShippedDate,
       Products.ProductName as ProductName,
       od.UnitPrice as UnitPrice,
       od.Quantity as Quantity
  FROM (([Order Details] od INNER JOIN Orders ON od.OrderID = Orders.OrderID)
       INNER JOIN Products ON od.ProductID = Products.ProductID)
       INNER JOIN Employees ON Orders.EmployeeID = Employees.EmployeeID
```

Step 3: Add calculated fields and parameters

1. Switch to the **Calculated Fields** tab and add two calculated fields with the following properties:

Name	FullName	OrderSum
Expression	FirstName & " " & LastName	UnitPrice * Quantity

2. In the **Data** tab, right-click **Parameters** and select **Add Parameters** to add three parameters, pEmployeeSortExpression, pEmployeeSortOrder, and pRepeatEmployeeHeader.
3. Set the following properties for the parameters from the **Properties** window.

	pEmployeeSortExpression	**pEmployeeSortOrder**	**pRepeatEmployeeHeader**
Prompt	Employees Sort Expression	Employees Sort Order	Repeat Employee Header

Value	Sum(OrderSum)	Descending	False
DataType	String	String	Boolean

4. Click the ellipsis button next to the **AllowedValuesDefinition** property, select **Values** radio button, click **Add** button to add the following values for **pEmployeeSortExpression** and **pEmployeeSortOrder** parameters:

pEmployeeSortExpression		
Label	Alphabetical	Sales
Value	FullName	Sum(OrderSum)

pEmployeeSortOrder			
Label	NoSort	Ascending	Descending
Value	NoSort	Ascending	Descending

Note that **Add** button will add blank values. To edit them, double click below the **Label** to add the required labels and **Values** to add the required values.

Step 4: Group and sort data

1. Switch to the **Design** mode to start editing the report.
2. Click **Groups** in **Home** tab and add **EmployeeGroup** and then add **OrderGroup**.
   Note that the groups are nested in the order in which they appear in the group wizard. Therefore, as you add the OrderGroup after the EmployeeGroup, it is added as the subgroup of the EmployeeGroup.
3. In the **Groups** wizard, set the following properties for the newly created groups:

	EmployeeGroup	**OrderGroup**
GroupBy	EmployeeID	OrderID
Keep Together	KeepFirstDetail	KeepWholeGroup
Sort	Ascending	No Sort
ShowGroupFooter	True	True
ShowGroupHeader	True	True
OutlineLabel	=FullName	-

4. Click **OK** to close the **Groups** wizard.
5. In the **EmployeeGroup_Header**, add three TextFields from the **INSERT** tab and set the following properties:

TextField1.Text	=FullName
TextField2.Text	Total Sales:
TextField3.Text	=Sum(OrderSum)

The **Sum(OrderSum)** calculates the group total, i.e., total sales for all the orders by an Employee.

6. Add a **ParagraphField** in the **EmployeeGroup_Footer** and display the group total using the following expression:
   Total Sales for {FullName}:{Sum(OrderSum)}
7. Right-Click the **OrderGroup_Header** and select **Add SubSection** from the context menu to divide the header into two parts.
8. In **OrderGroup_Header/ <A>**, add six more TextFields and set the following properties:

TextField4.Text	Order ID
TextField5.Text	=OrderID
TextField6.Text	Order Date
TextField7.Text	=OrderDate
TextField8.Text	Shipped Date
TextField9.Text	=ShippedDate

9. In **OrderGroup_Header/ <B>**, add three more TextFields, Product Name, Unit Price, and Quantity.
10. In the **Details** section, add three more TextFields corresponding to the TextFields in **OrderGroup_Header/ <B>**, and set the

following properties:

TextField13.Text	=ProductName
TextField14.Text	=UnitPrice
TextField15.Text	=Quantity

11. In the **OrderGroup_Footer**, display the subtotal of the order group by using a **TextField** with its **Text** property set to **=Sum(OrderSum).**
12. Select the report name from the drop-down situated above the list of properties in the **Properties** window.
13. Go to **GlobalScripts** property of the report and write following expression in the VBScriptEditor.

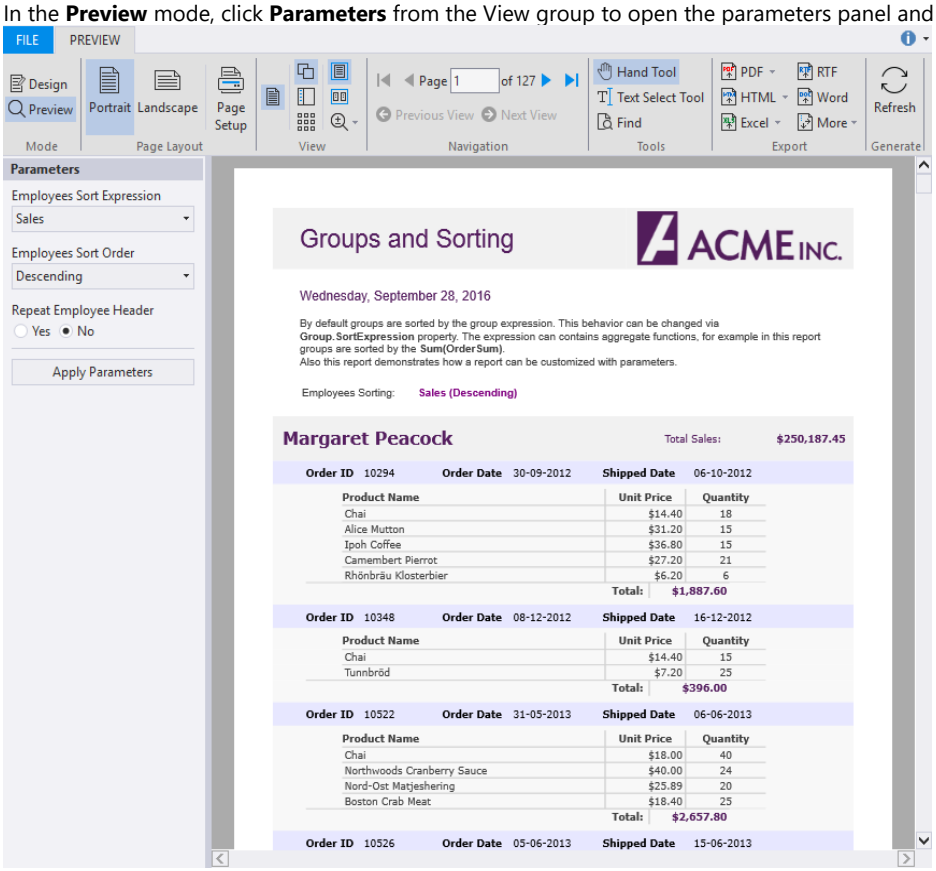
```
EmployeeGroup.SortExpression = pEmployeeSortExpression.Value
EmployeeGroup.Sort = pEmployeeSortOrder.Value
```

14. Go to **OnOpen** property of the report and write the following expression in the VBScript Editor.

```
If pEmployeeSortOrder = "NoSort" Then
  fldSortDesc = "No Sorting"
Else
  fldSortDesc = pEmployeeSortExpression.DisplayText & " (" & pEmployeeSortOrder.DisplayText & ")"
EndIf

EmployeeGroup_Header.Repeat = pRepeatEmployeeHeader.Value
```

Step 5: View the report

1. Preview the report.
2. In the **Preview** mode, click **Parameters** from the View group to open the parameters panel and apply parameters.



Note: For the complete report, see report 'Groups and Sorting' in the **FlexCommonTasks.flxr** report definition file, which is available in the **ComponentOne Samples\Winforms\C1FlexReport\CS\FlexCommonTasks** folder. The data base used is **C1NWind.mdb** which is also available in the **ComponentOne Samples** folder.

Cascading Parameters

In order to manage large amount of data in reports, you need to use cascading parameters. With cascading parameters, set of related parameters can be defined so that the list of values for one parameter depends on the value selected in another parameter.

Here, we will create a report that contains two parameters pCountry and pCustomers. After selecting a country, you can select one or more customers from the selected country and list all orders for those customers. The ReportParameter.DisplayText property is used to display parameters in this report.

Let us create a report in which cascading parameters is used.

1. Create a new report and bind it to the Main data source using the following Sql Statement:
```
Select orderid, orders.customerid, companyname, employees.firstname,
employees.lastname, orderdate, RequiredDate, shippeddate, Freight
from (orders inner join customers on orders.customerid = customers.customerid)
inner join employees on orders.employeeid = employees.employeeid
where orders.CustomerID in pCustomers
```

2. Switch to the Calculated Fields tab and add a field named Salesperson with the following expression:
```
FirstName & " " & lastname
```

3. Add a new data source, dsCountries, and bind the report to the data source using the following Sql Statement:
```
Select Country, Count(*) as CustomerCount
from Customers group by Country order by Country
```

4. Switch to the Calculated Fields tab and add a field named CountryDesc with the following expression:
```
Country & " (" & CustomerCount & " customers)"
```

5. Add another data source named dsCustomers and bind the report to the data source using the following Sql Statement:
```
Select CustomerID, CompanyName from Customers where Country = pCountry
```

6. Add a parameter, pCountry, and set the following properties from the Properties window.

DataType	String
Prompt	Country
Value	Germany

7. Click the ellipsis button next to the **AllowedValuesDefinition** property, select **From Data Source** radio button, and set the following properties:

Data Source	dsCountries
Label	CountryDesc
Value	Country

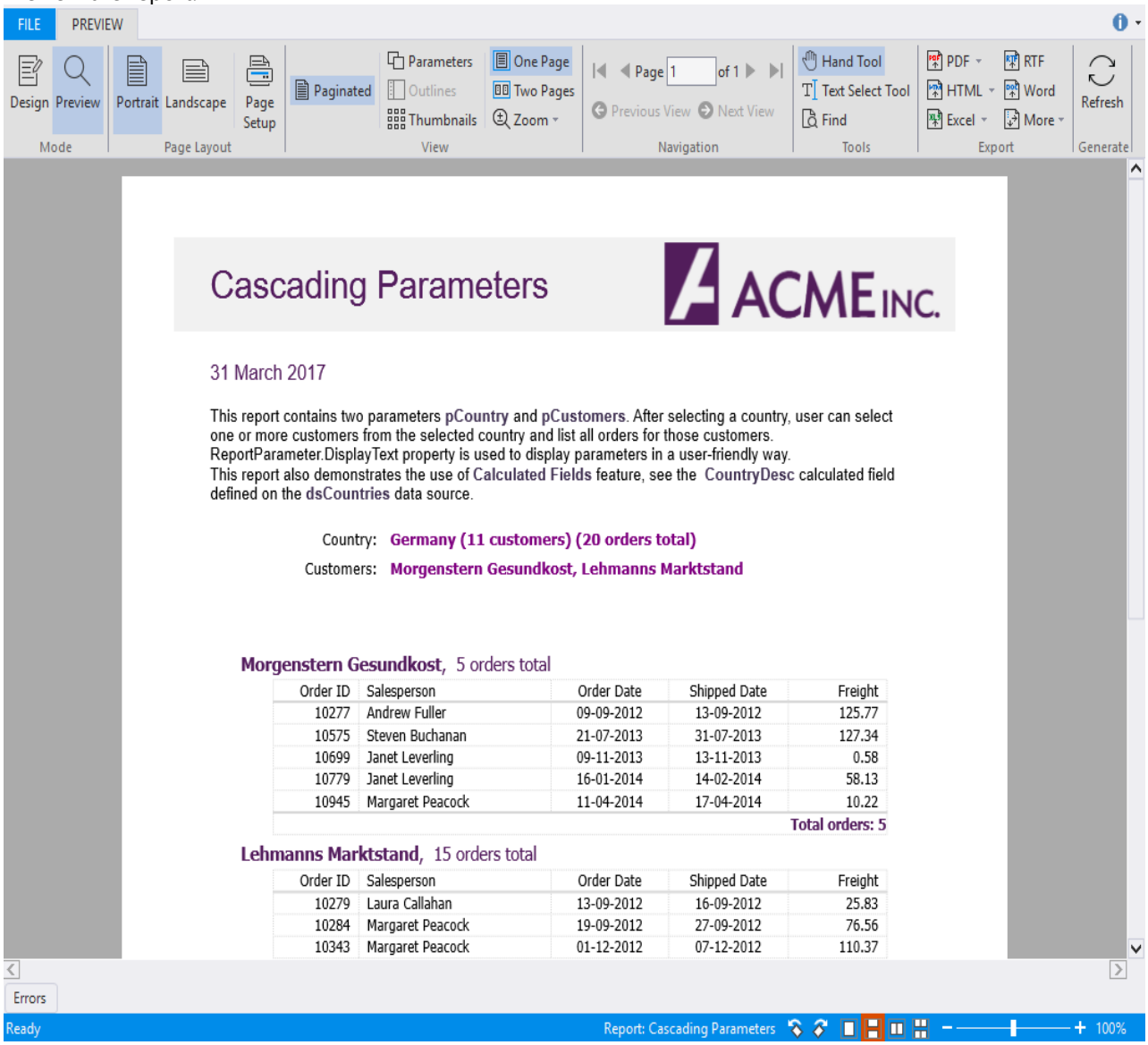
8. Add a parameter, pCustomers, and set the following properties from the Properties window.

DataType	String
MultiValue	True
Prompt	Customers
Value	[MORGK, LEHMS]

9. Click the ellipsis button next to the **AllowedValuesDefinition** property and select **From Data Source** radio button, and set the following properties:

Data Source	dsCustomers
Label	CompanyName
Value	CustomerID

10. Preview the report.



Note: For the complete report, see report 'Cascading Parameters' in the **FlexCommonTasks.flxr** report definition file, which is available in the **ComponentOne Samples\Winforms\C1FlexReport\CS\FlexCommonTasks** folder. The data base used is **C1NWind.mdb** which is also available in the **ComponentOne Samples** folder.