ComponentOne

Reports for WPF

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

The ComponentOne product name is a trademark and ComponentOne is a registered trademark of GrapeCity, Inc. All other trademarks used herein are the properties of their respective owners.

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 $25 (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

While you are welcome to make backup copies of the software for your own use and protection, you are not permitted to make copies for the use of anyone else. We put a lot of time and effort into creating this product, and we appreciate your support in seeing that it is used by licensed users only.

Table of Contents

Reports for WPF Overview

Integrate reporting and document generating functionality into your Windows Presentation Foundation (WPF) applications with Reports for WPF. Using Reports for WPF, generate native WPF documents (FixedDocument objects and XPS files), strengthen your WPF applications with powerful report and document generating capabilities, and save time with the clear upgrade path for existing users moving from WinForms to WPF.

Reports for WPF includes the following tools:

- The C1DocumentViewer control, which hosts paginated fixed documents, including C1Report and C1PrintDocument FixedDocumentSequence representations. See C1DocumentViewer Overview for details.
- The C1Report component, which generates access-style, data-based banded reports that can be rendered into a FixedDocumentSequence, directly to a printer, or exported to various portable formats (including XPS, HTML, PDF, and text). For information about using C1Report, see the Reports for WinForms documentation.
- The C1PrintDocument component, which represents a document that can be previewed, printed or exported to an external format. For information about using C1PrintDocument, see the Reports for WinForms documentation.
- The C1ReportDesigner designer, a stand-alone application used to create report definitions without writing code. The designer allows you to quickly create and edit report definitions, or to import existing Microsoft Access and Crystal report definitions.
- The C1ReportsScheduler application, a stand-alone application used to schedule report creation to run in the background. Using the C1ReportsScheduler application, you can choose what reports to export or print, formats to export the report to, and the schedule and frequency for creating reports.

See Also

Help with WPF Edition

C1DocumentViewer Overview

C1Report and C1PrintDocument Overview

Reports and Preview .NET Versions

Differences between WPF and .NET 2.0 Versions

Help with WPF Edition

Getting Started

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

See Also

C1DocumentViewer Overview

C1DocumentViewer Overview

Reports for WPF Overview > C1DocumentViewer Overview

The **C1DocumentViewer** is a WPF control derived from the standard WPF System.Windows.Controls.DocumentViewer control. In addition to the features of the base control, it exposes two public dependency properties that simplify its use when previewing **C1Report** and **C1PrintDocument** documents:

- FileName property

Gets or sets the name of the report definition file, or of the **C1PrintDocument** (C1D or C1DX) file.

- ReportName property

Gets or sets the name of the report to show.

These properties work both at run time and at design time, but at design time the size of the report is limited to 4 pages (this does not apply to C1D/C1DX documents).

The **C1DocumentViewer** can be placed on the Visual Studio Toolbox and then dropped onto the WPF window or page. In addition to actually creating the document viewer on your window/page, this does two other important things:

- All the necessary references to system DLLs are automatically added to your application.
- The license for Reports for WPF is added to your application (and the licenses.licx file is created if it was not present).

See Also

C1Report and C1PrintDocument Overview

C1Report and C1PrintDocument Overview

Reports for WPF Overview > C1Report and C1PrintDocument Overview

Reports for WPF is based on the WinForms products, **Reports for .NET** and **Preview for .NET**, now both part of **Reports for WinForms**, and provides a report and document generating engine adapted to the WPF environment.

The following two components, similar to the WinForms components, are provided:

- **C1Report**
- **C1PrintDocument**

- Both of these components derive from System.ComponentModel.Component. As such, neither can be placed directly on the Visual Studio's Toolbox or on a WPF window or page. The C1DocumentViewer control provides a convenient way to quickly add **C1Report** and all necessary wiring (such as references to the required system DLLs and licenses) to your project. But it should be noted that a **C1Report** or a **C1PrintDocument** can be viewed in the standard **DocumentViewer** as well. And of course, all printing and exporting capabilities are accessible directly via **C1Report** and **C1PrintDocument** components.

  Existing users of the former **Preview for WinForms** and **Reports for WinForms** products will notice that while previously, **C1Preview** and **C1Report** were separate products shipped in different assemblies, in this release both components are provided by the same assembly, as does the new combined **Reports for WinForms**. For **C1Report** users, the main difference is that in this release, the **C1Report** namespace was changed from C1.Win.C1Report to C1.C1Report. Otherwise, the **C1Report** component provided in this release should be backwards compatible with previous WinForms versions of **C1Report**. The **C1PrintDocument** component included in **Reports for WPF** is almost (but not 100%) code compatible with the WinForms version (for details, see Differences between WPF and .NET 2.0 Versions).

See Also

Reports and Preview .NET Versions

Reports and Preview .NET Versions

The following table describes the available .NET versions of ComponentOne reporting and previewing products. Note that the list has been numbered to differentiate between versions (this product, **Reports for WPF**, is #7 below):

#	Name	.NET Framework	Assemblies	Controls
1	Preview for .NET	.NET 1.x	C1.C1PrintDocument.dll C1.Win.C1PrintPreview.dll	C1PrintDocument C1PrintPreview
2	Reports for .NET	.NET 1.x	C1 C1.Win.C1Report.dll	C1Report
3	Preview Classic for .NET	.NET 2.0	C1.C1PrintDocument.Classic.2.dll C1.Win.C1PrintPreview.Classic.2.dll	C1PrintDocument C1PrintPreview

4	Reports for .NET	.NET 2.0	C1.Win.C1Report.2.dll	C1Report
5	Preview for .NET	.NET 2.0	C1.C1Preview.2.dll C1.Win.C1Preview.2.dll	C1PrintDocument C1PreviewPane C1PrintPreviewControl C1PrintPreviewDialog C1PreviewThumbnailView C1PreviewOutlineView C1PreviewTextSearchPanel
6	Reports for WinForms	.NET 2.0	C1.C1Report.2.dll C1.Win.C1Report.2.dll	C1Report C1PrintDocument C1PreviewPane C1PrintPreviewControl C1PrintPreviewDialog C1PreviewThumbnailView C1PreviewOutlineView C1PreviewTextSearchPanel
7	Reports for WPF	.NET 3.0	C1.WPF.C1Report.dll C1.WPF.C1Report.Design.dll C1.WPF.C1Report.VisualStudio.Design.dll	C1DocumentViewer

Version Compatibility

While the products above provide reporting and previewing functionality and may include similar components, they are not all backwards compatible. Some considerations for upgrading versions are discussed below:

- **Preview Classic for .NET (.NET 2.0, #3 in the above table)**

  This is the "classic" version of preview controls. While the assembly names are different from #1, these are 100% backwards compatible, so upgrading from the .NET 1.x product (#1 above) does not require any changes except for changing the references in your project and licenses.licx files. Preview Classic for .NET is no longer actively developed and is in maintenance mode.

- **Reports for .NET (.NET 2.0, #4 above)**

  This is the old .NET 2.0 version of reporting controls. These reports can be shown by all versions of preview controls (#1, #3 or #5 above), by assigning the **C1Report.Document** property to the **Document** property of a preview control.

- **Preview for .NET (.NET 2.0, #5 above)**

  This is the newer previewing product (new compared to the classic version). This product has different code and object model from the previous versions (#1or #3 above). Automatic conversion from #1 or #3 to this product is **not** supported; in particular the **Convert2Report.exe** utility **cannot** convert those older projects. Converting from #1 or #3 to this preview requires rewriting user code, always. The scope of changes differs and may be trivial, but code **must** be updated by hand.

- **Reports for WinForms (.NET 2.0, #6 above)**

  The current .NET 2.0 combined reporting and previewing product. This includes both the "new" preview (#5 above) and reports (#4 above). Unlike in previous versions, to preview a **C1Report**, it itself (rather than its **Document** property) should be assigned to the **Document** property of the preview control. This build is backwards code compatible with #4 and #5, but assembly references and namespaces must be updated. The changes are always trivial and can be made manually or by using the **Convert2Report.exe** utility which can be downloaded from HelpCentral.

- **Reports for WPF (.NET 3.0, #7 above)**

  The WPF version of reporting and preview controls which includes **Preview for .NET 2.0** (#5) and **Reports for .NET 2.0** (#4). This product can be used in .NET 3.0 and 3.5 applications.

See Also

Differences between WPF and .NET 2.0 Versions

WPF and .NET 2.0 Versions

The new version of .NET framework includes some new classes for commonly used entities such as colors and brushes. In .NET 2.0 and earlier, colors were represented by the type System.Drawing.Color. In WPF, while that type can still be used, new types were added:

- System.Windows.Media.Color struct to define a color.
- System.Windows.Media.Colors class to represent predefined colors.

Similarly, while in .NET 2.0 the System.Drawing.Brush was used to paint objects, in WPF a new hierarchy of types was added to paint objects, based on System.Windows.Media.Brush.

To make programming against the **C1PrintDocument** object model easier in the WPF version, color and brush properties of the C1.C1Preview.Style class use the System.Windows.Media types. While some existing code may not be 100% backwards-compatible, the necessary changes should generally be trivial.

See Also

Key Features

Key Features

Benefit from using **Reports for WPF**, featuring:

- **Clear Upgrade Path From WinForms to WPF Reporting**

Save time with a clear upgrade path for existing ComponentOne users moving from **Reports for WinForms** and **Preview for WinForms** to WPF. Current users can easily upgrade reporting and document-generating code to the new WPF platform. With only minor modifications, your existing code will run and generate reports and documents in a WPF application.

- **Preview in C1DocumentViewer or WPF DocumentViewer**

Preview your reports and documents in the C1DocumentViewer control or the standard Microsoft WPF DocumentViewer control.

- **Save in XPS Format**
  Save in XPS (XML Paper Specification) format, offering numerous advantages including better printing, easier file sharing, and greater security.

- **Export to Common File formats**
  Render your reports directly to a printer or export your reports to various portable formats – Excel, PDF, HTML, text, and images.
- **Control conditional formatting, grouping, filtering, sorting, and more with VBScript expressions**
  Use VBScript expressions to retrieve, calculate, display, group, filter, sort, parameterize, and format the contents of a report, including extensions for aggregate expressions (sum, max, average, and so on).
- **Broad data source support**
  Flexibility to bind your report to the data source of your choice – specify a connection string and an SQL statement in your report or document definition, and **Reports for WPF** will load the data automatically for you. Optionally, use XML files, custom collections, and other data sources.

- **Interoperability between the WPF and WinForms versions**
  Both report definitions and C1DX/C1D documents are fully compatible with the WinForms versions of the Reports and Preview products.
- **Save and load report definition files**
  Save and load report definition files using custom fields for WPF. The custom fields that are supported in the report designer application are Chart, Superlabel, Gradient, Map, and QRCode.

Benefit from the features available in **C1Report**, including:

- **Fast, easy-to-use, and full-featured report generation**
  Use **C1Report** in many different scenarios, on the desktop and on the Web. Create a report definition, load data, and render the report without writing any code.

- **Compatible with Microsoft® Access reports**
  Supports features found in Microsoft® Access reports; the report designer can import Access and Crystal reports.

- **Organized report layout**
  Banded report model based on groups, sections, and fields.

- **Data-bound fields**
  Fields may be bound to simple and binary (object) database fields.

- **VBScript event handlers**
  Reports may contain embedded VBScript event handlers, making them self contained.

- **Sub-reports**
  Reports may contain nested reports to arbitrary levels (sub-reports).

- **Report parameters**
  Reports may contain parameterized queries, allowing users to customize the report by adding/limiting the data that should be included in the report before it is rendered.
- **Intelligent WYSIWYG report designer**
  Quickly create, edit, preview, load, and save report definition files without writing any code.

- **Powerful and easy-to-use report wizard**
  Effortlessly create a new report from start to finish with **C1Report**'s wizard walking you through the steps.

Benefit from the features available in C1PrintDocument, including:

- **Powerful document oriented object model**
  C1PrintDocument provides a flexible hierarchical document object model specifically oriented towards creation of paginated documents of arbitrary complexity. Powerful automatic layout, formatting and pagination control features alleviate the need to manually calculate the layout, insert page breaks and so on.

- **Rich text and graphics formatting options**
  Support for paragraphs of text with multiple fonts, text and background colors, text

positioning (subscript, superscript), inline images, various text alignment (including justified text), and more.

- **Hierarchical styles**
  The look of all document elements is controlled by hierarchical styles, with intelligent support for ambient and non-ambient style attributes. Some unique features include the ability to specify individual font attributes (such as boldness or font size), table grid lines, and so on.

- **Hyperlinks, link targets, TOC**
  Any document element can be a hyperlink, or a hyperlink jump target. You can also indicate that an element should be an item of the automatically generated table of contents (TOC).

- **Multiple page layouts**
  Several page layouts accommodating different paper sizes, page settings, number of columns, page headers and so on can be predefined, to be selected at runtime by setting a single property.

- **Powerful sizing and positioning of document elements**
  Sizes and positions of document elements can be specified as absolute values, relative to other elements' sizes and positions, or as simple expressions combining absolute and relative values. For instance, the width of an element can be specified as percentage of the parent element or of the current page width.

- **Powerful Excel-like tables**
  Tables support an Excel-like object model, with logically infinite number of columns and rows. Simply accessing a table element instantiates it, so you never have to worry about specifying the correct table size. Table elements can be grouped together in a number of ways. Styles can be applied to groups to modify the appearance of all elements in a group. Tables provide support for row and column headers, infinitely nested tables, automatic column widths and row heights, and more.

- **Horizontal pages**
  Wide elements can extend to multiple horizontal pages.

- **C1DX (C1D Open XML) File Format support**
  New OPC-based, file format for **C1PrintDocument** objects, C1DX complies with Microsoft Open Packaging Conventions, and is similar to the Office 2007 Open XML format. Due to built-in compression, resulting files are smaller in size. The C1D format is also fully supported for backwards compatibility.

See Also

Getting Started with Reports for WPF

Getting Started with Reports for WPF

The C1.C1Report.C1Report component in **Reports for WPF** provides a complete compatible replacement for the C1.Win.C1Report.C1Report component included in previous versions of **Reports for .NET**. The only difference, from the user code point of view, is the namespace change (C1.C1Report instead of C1.Win.C1Report). The C1.C1Report namespace also provides all other familiar public **C1Report** classes such as Field, Section, and so on. Again, the only expected difference affecting the user is the namespace change.

Internally, the new **C1Report** works differently from the old version. Whereas the old **C1Report** engine in the usual (preview/print) scenario generated metafile page images, the new **C1Report** builds a **C1PrintDocument** representing the report. That document is accessible via the new public read-only property on **C1Report**: C1PrintDocument C1Report.C1Document {get;}.

The document can then be used in any of the usual ways; for example, it can be exported to one of the formats supported by **C1PrintDocument**.

Export filters:

Along with other public members exposed by the old version, the new **C1Report** provides the familiar WinForms **C1Report's** export filters, so the following code is still completely valid:

C#	Copy Code

```
report.Load(...);
report.RenderToFile("MyReport.rtf",C1.C1Report.FileFormatEnum.RTF);
```

It is important to note that the file produced by the code above (in our example, an RTF file) will differ from the file produced by exporting the **C1PrintDocument** exposed by the report:

C#	Copy Code

```
report.Load(...);
report.C1Document.Export("MyReport.rtf");
```

Usually, the C1Report.RenderToFile method would yield better results, unless the target format is a fixed layout format (such as PDF), in which case the results should be identical. Also note that the C1Report.RenderToFile method does not support the new XPS format; the only way to generate an XPS file is to export the **C1PrintDocument** exposed by **C1Report**.

From 2015v2 onwards, please add C1.Win.4 and C1.Win.Barcode.4 dlls to the projects referencing C1.WPF.C1Report.CustomFields.4 dll.

See Also

Reports for WPF Quick Starts

Generating Reports (C1Report vs. C1PrintDocument)

Using the FixedDocumentSequence Property

Exporting to XML Paper Specification (XPS)

Using the C1DX (C1D Open XML) File Format

Reports for WPF Quick Starts

In the following three quick starts you'll familiarize yourself with the **C1DocumentViewer** control and the **C1Report** and **C1PrintDocument** components. The **C1DocumentViewer** control is very similar to the **DocumentViewer** control included in Visual Studio 2008 and if you've used the **Reports for WinForms** product, you should already be familiar with the **C1Report** and **C1PrintDocument** components.

See Also

C1DocumentViewer Quick Start

C1Report Quick Start

C1PrintDocument Quick Start

C1DocumentViewer Quick Start

In this quick start you'll create a code-free WPF application in Visual Studio that uses the C1DocumentViewer control to preview reports and documents.

Note: This quick start assumes that you have the **C1Report** sample **CommonTasks.xml** report definition file available on your system. The **CommonTasks.xml** is installed in the **ComponentOne Samples\C1WPFReport\C1Report\Samples\XML\CommonTasks** directory installed in your **MyDocuments** folder (**Documents** in Vista) by default.

Complete the following steps to create a basic application to preview reports and documents:

1. Create a new WPF project in Visual Studio.
2. Navigate to the Toolbox and double-click the **C1DocumentViewer** icon to add the control to the application window.
3. Resize the window and resize the **C1DocumentViewer** control to fill the window.

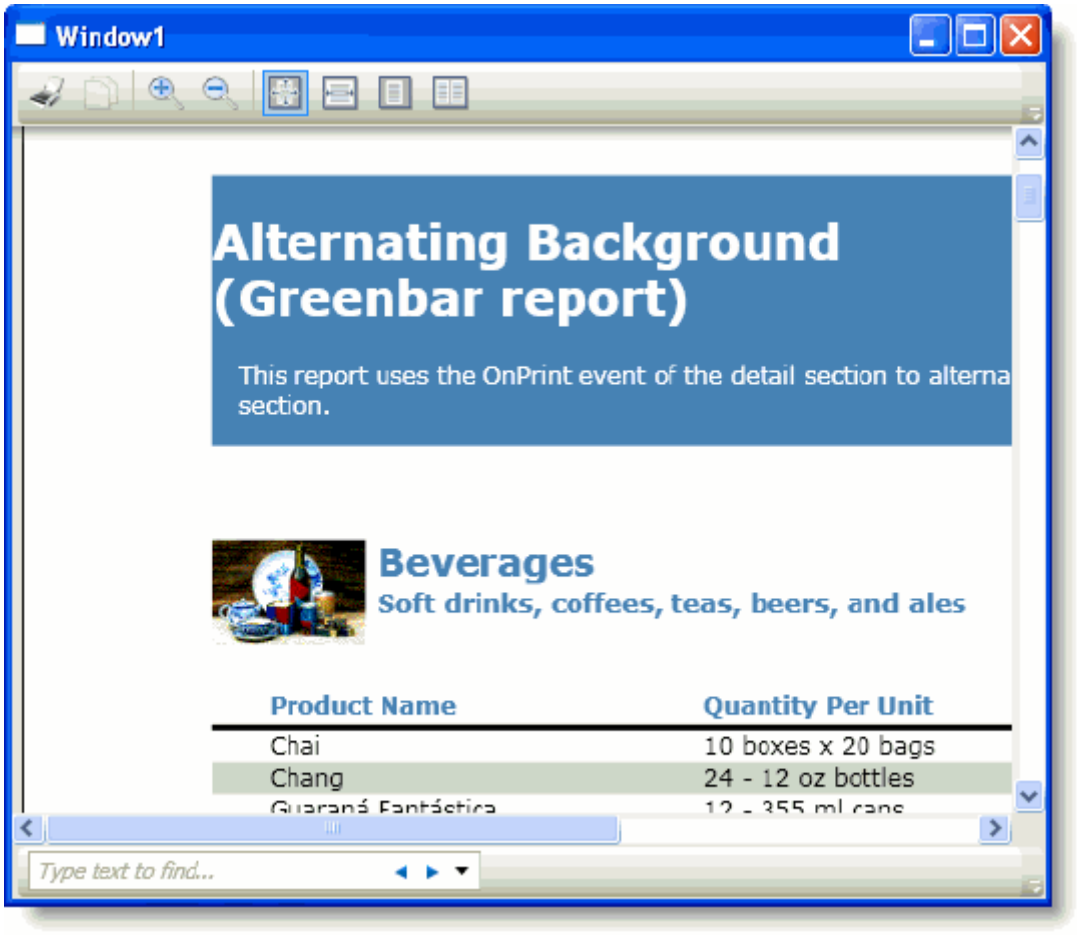
4. Click once on the **C1DocumentViewer** control to select it and navigate to the Properties window.
5. Expand the **C1Report** node and click on the ellipsis button (...) to the right of theC1DocumentViewer.FileName text box.
6. In the Open dialog box, locate and highlight a report definition file, for example select CommonTasks.xml (installed in the samples directory by default) and click **Open** to select the file in the editor.
7. Click the drop-down arrow next to the C1DocumentViewer.ReportName property and select a report; for example, select **01: Alternating Background (Greenbar report)**.

The report appears in the **C1DocumentViewer** window.

That's it; no code is necessary! You can now run your application and preview your report.

Run the application and observe:

Your simple application to view reports will look similar to the following:



See Also

C1Report Quick Start

C1Report Quick Start

In this quick start you'll create a basic WPF application in Visual Studio that references **Reports for WPF** for previewing reports and documents.

Note: This quick start assumes that you have the sample **CommonTasks.xml** report definition file available on your system. The **CommonTasks.xml** is installed in the samples directory by default.

Complete the following steps to create a basic form for previewing reports and documents:

1. Create a new WPF application in Visual Studio.
2. Add a reference to the C1.WPF.C1Report assembly to your project:
3. Select **Project | Add Reference** to open the **Add Reference** dialog box.
4. Select the ComponentOne Reports for WPF assembly from the list on the .NET tab or, on the **Browse** tab, browse to find the C1.WPF.C1Report.dll file and click **OK**.
5. Double-click the window caption area to switch to Code view. At the top of the file, add the following Imports statement (using in C#):

   Imports C1.C1Report

This makes the objects defined in the **Reports for WPF** assembly visible to the project.

6. From the View menu, select Designer to return to Design view.
7. Navigate to the Toolbox and double-click the standard DocumentViewer and Button controls to add them to your window. Then resize the window and the control to fill the window.
8. Select the button control and in the Properties window set the **Button.Content** property to "View report".
9. Double-click the Button control to switch to Code view and create the **Button_Click** event handler, and add the following code to the **Button_Click** handler:

Visual Basic	Copy Code

```
Dim rep As New C1.C1Report.C1Report()
rep.Load("C:\Documents and Settings\<username>\My Documents\ComponentOne Samples\C1WPFReport\C1Report\Samples\XML\CommonTasks\CommonTasks.xml", "01: Alternating Background (Greenbar report)")
DocumentViewer1.Document = rep.FixedDocumentSequence
```

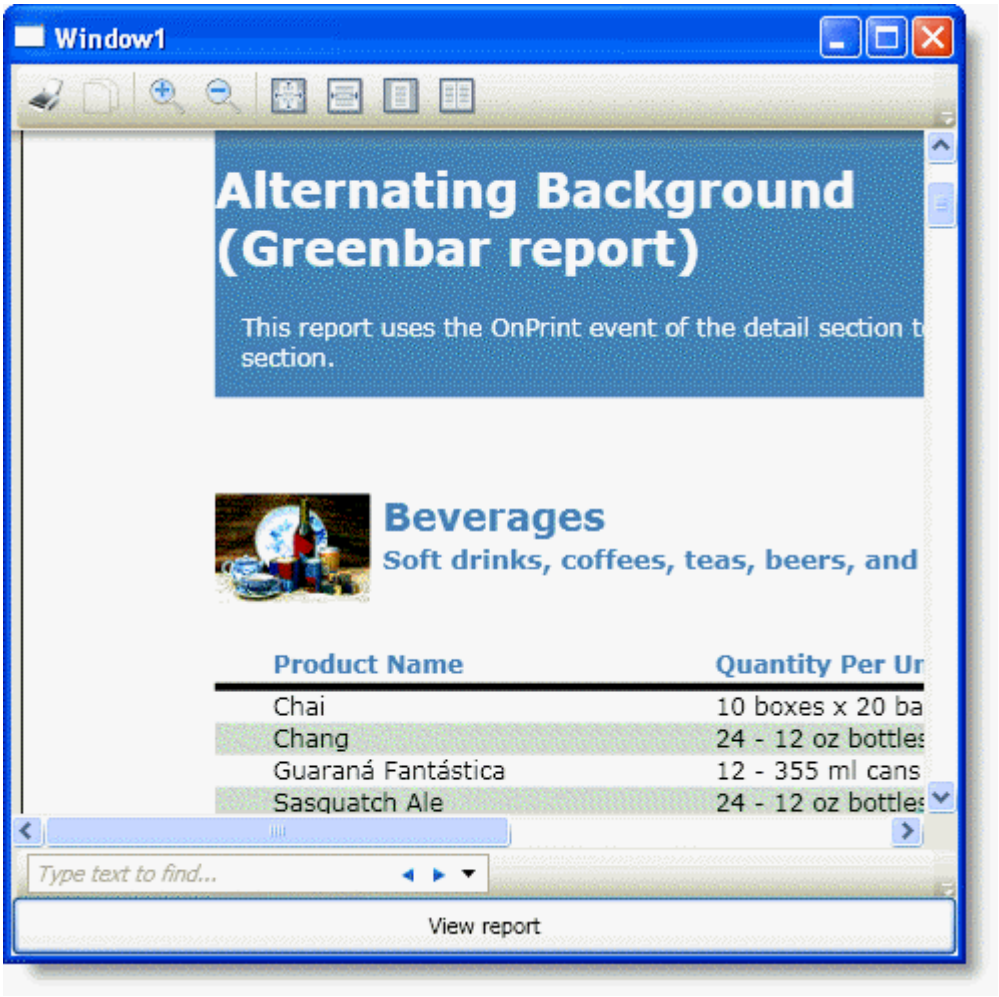
C#	Copy Code

```
C1.C1Report.C1Report rep =  new C1.C1Report.C1Report();
```

```
rep.Load(@"C:\\Documents and Settings\\<username>\\My Documents\\ComponentOne
Samples\\C1WPFReport\\C1Report\\Samples\\XML\\CommonTasks\\CommonTasks.xml",
@"01: Alternating Background (Greenbar report)");
documentViewer1.Document = rep.FixedDocumentSequence;
```

10. Run the application and click the **View report** button.

The report will take a few seconds to load, and will look similar to the following:



See Also

C1PrintDocument Quick Start

C1PrintDocument Quick Start

Getting Started with Reports for WPF > Reports for WPF Quick Starts > C1PrintDocument Quick Start

In this quick start you'll create a basic Visual Studio project that references **Reports for WPF** for previewing reports and documents.

To create a basic form for previewing reports and documents:

1. Create a new WPF application in Visual Studio.
2. Add a reference to the C1.WPF.C1Report assembly to your project:
3. Select the **Add Reference** option from the **Project** menu of your project.
4. Select the ComponentOne WPFReport assembly from the list on the .NET tab, or on the **Browse** tab, browse to find the C1.WPF.C1Report.dll file and click **OK**.
5. Double-click the window caption area to open the Code view. At the top of the file, add the following Imports statement (using in C#):

    Imports C1.C1Preview

This makes the objects defined in the **Reports for WPF** assembly visible to the project.

6. From the **View** menu, select **Designer** to return to Design view.
7. From the Toolbox, double-click the standard **DocumentViewer** and **Button** controls to add them to your window. Then resize the window and the control to fill the window.
8. Select the button control and from the Properties window set the **Button.Content** property to "View document".
9. Double-click the **Button** control to open the Code view, and add the following code to the **Button_Click** handler:

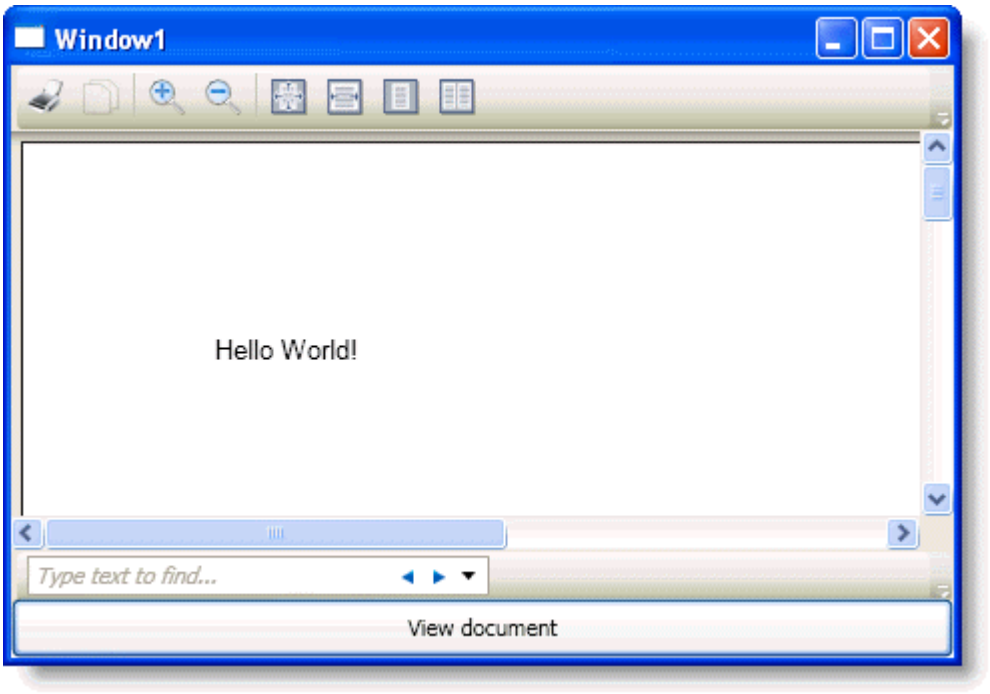
Visual Basic	Copy Code

```
Dim doc As New C1.C1Preview.C1PrintDocument()
doc.Body.Children.Add(New RenderText("Hello World!"))
DocumentViewer1.Document = doc.FixedDocumentSequence
```

C#	Copy Code

```
C1.C1Preview.C1PrintDocument doc = new C1.C1Preview.C1PrintDocument();
doc.Body.Children.Add(new RenderText("Hello World!"));
documentViewer1.Document = doc.FixedDocumentSequence;
```

10. Run the application and click the **View document** button. The report will take a few seconds to load, and should look similar to the following:



See Also

Generating Reports (C1Report vs. C1PrintDocument)

Generating Reports (C1Report vs. C1PrintDocument)

Getting Started with Reports for WPF > Generating Reports (C1Report vs. C1PrintDocument)

The **Reports for WPF** assembly now provides two distinctly different methods for generating reports:

- Using the C1PrintDocument.ImportC1Report method.
- Using the new **C1.C1Report.C1Report** component.

While many existing reports can be correctly generated using either method, there are some important differences between them including:

Data Binding

Import creates a data-bound C1PrintDocument which is then generated, and data is fetched at that time. The resulting document can be refreshed with data refreshing. When the C1Report component is used, the generated document already contains embedded data fetched during document generation, and the resulting document is not data-bound (but of course, the report can be rendered again, refreshing the data).

Document Structure

When a report is imported in the resulting **C1PrintDocument** structure, all fields are represented by **RenderField**, and all sections by **RenderSection** objects. If a report definition contains groups, each group is represented by a **RenderArea** object, with nested **RenderSection** objects for the group header and footer, and a RenderArea for the nested group.

When the **C1Report** component is used, each report section is rendered into a **RenderArea** object. The fields are rendered as follows:

- If **LineSlant** is specified, a **RenderLine** is generated;
- If image or barcode is specified, a **RenderImage** is generated;
- If **RTF** is True, a **RenderRichText** is generated;
- In all other cases, a **RenderText** is generated.

Page Size

When a report with unset (or set to zeroes) **Layout.CustomWidth** and **Layout.CustomHeight** properties is imported on a system without a printer, the default page size depends on the locale, as always in **C1PrintDocument** (for example, for US and Canada, the page size will be Letter, for Russia A4, and so on).

When such report is loaded into the **C1Report** component, the page size is always set to Letter (8.5 x 11 inches), which is the behavior of C1.Win.C1Report.C1Report.

Default Printer

On a system with one or more printers installed, the default page size for an imported report is determined using the same logic as for a regular **C1PrintDocument** (in particular, the C1PrintDocument.MeasurementPrinterName property is used to determine which printer to use). The main reason for this behavior is to avoid long wait times on systems with default network printers. When the **C1Report** component is used, the default page size is determined by the system default printer.

Import Limitations

The import method has some inherent limitations which are not likely to ever be lifted. Those include:

- When import is used, the C1Report object model is not available and report events are not invoked. Hence the main limitation of import: reports depending on C#/VB event handlers cannot be rendered correctly when imported.

- Scripting limitations:

- C1PrintDocument Object
  The **Font** property is read-only.

- Field Object
  - **Field.Section**, **Field.Font**, and **Field.Subreport** properties are read-only;
  - **LineSpacing**, **SubreportHasData**, and **LinkValue** properties are not supported; if the

Field.LinkTarget property contains an expression, it is not evaluated and will be inserted into the generated report as-is.

- Layout Object
  - The **ColumnLayout** property is not supported, columns always go top to bottom and left to right.
  - **LabelSpacingX**, **LabelSpacingY**, and **OverlayReplacements** are not supported.
  - **ForcePageBreak** cannot be used in **OnPrint** handler.

- The dialog box is not shown for parameterized reports. Instead, default values depending on the parameter type are used (0 for numbers, empty string for strings, the current date for dates and so on).
- **PageHeader**/**PageFooter** cannot reference database fields.
- In multi-column reports, the report header is printed only above the first column (in **C1Report**, the header spans all columns).
- Left to right orientation for columns is not supported – columns are always oriented from top to bottom.
- In **C1Report**, a Field containing an image, with **Field.CanGrow** set to **True** and **Field.PictureScale** set to **Clip**, has the width equal to that of the image. When imported, the width of such a field is scaled proportionally to the width of the image.

See Also

Deciding on Report Generation Method

Deciding on Report Generation Method

Getting Started with Reports for WPF > Generating Reports (C1Report vs. C1PrintDocument) > Deciding on Report Generation Method

Because two ways of generating reports are available (using the C1Report component vs. importing the report into a C1PrintDocument), you may ask, "Which method is preferable?" Our recommendations are as follows:

- If any of the limitations of import are critical to your application, use the **C1Report** component (see Generating Reports (C1Report vs. C1PrintDocument) for a list of limitations).
- If you are a user of a previous version of **C1Report** and are not familiar with the **C1PrintDocument** object model, you may still continue to use the **C1Report** component.
- If, on the other hand, you have some experience with **C1PrintDocument**, or are starting a new project, using import is the preferred approach, due to the following considerations:

- C1PrintDocument integration: when a report definition has been imported into a **C1PrintDocument**, the resulting document can be manipulated as any other **C1PrintDocument**. For example, user code can add content to the document body, modify document properties, and so on. Such changes will persist even when the document is refreshed.

- Some problems existing in **C1Report** are solved by import; specifically, in **C1Report** side-by-side objects cannot be correctly split between pages, and borders are not

rendered correctly on objects split between pages. Neither of these problems exists when a report is imported into **C1PrintDocument**.

- Import is slightly more efficient both memory- and speed-wise.
- Future enhancements: it is likely that some future enhancements will affect only import.

See Also

Using the FixedDocumentSequence Property

Using the FixedDocumentSequence Property

Getting Started with Reports for WPF > Using the FixedDocumentSequence Property

In this release, both C1Report and C1PrintDocument include a new read-only public property, FixedDocumentSequence FixedDocumentSequence {get;}, which provides access to an instance of **FixedDocumentSequence** representing the report/document.

The value of the **FixedDocumentSequence** property can be assigned to the **Document** property of the **C1DocumentViewer** or of the standard WPF **DocumentViewer** control to preview the document or report. It is important to note that accessing the value of the **FixedDocumentSequence** property can result in document/report (re)generation. Specifically, when **C1.C1Report.C1Report.FixedDocumentSequence** on a **C1Report** is accessed, the report is rendered if it has been changed since last generation. When **C1PrintDocument.FixedDocumentSequence** on a **C1PrintDocument** is accessed, the document is generated if currently its Pages collection is empty.

See Also

Exporting to XML Paper Specification (XPS)

Exporting to XML Paper Specification (XPS)

Getting Started with Reports for WPF > Exporting to XML Paper Specification (XPS)

A C1PrintDocument can now be exported to the new XML Paper Specification (XPS). As with other export formats supported by **C1PrintDocument**, there are several ways to export the document to XPS. The easiest way would be to use any of the **Export** methods' overloads, for example:

Visual Basic	Copy Code
`Dim doc As New C1PrintDocument()`	

```
doc.Export("MyDocument.xps")
```

C#	Copy Code

```
C1PrintDocument doc = new C1PrintDocument();
doc.Export("MyDocument.xps");
```

The Export method can be invoked on the **C1PrintDocument** exposed by the C1Report component:

Visual Basic	Copy Code

```
Dim rep As New C1Report()
rep.Load(…)
rep.C1Document.Export("MyReport.xps")
```

C#	Copy Code

```
C1Report rep = new C1Report();
rep.Load(…);
rep.C1Document.Export("MyReport.xps");
```

See Also

Using the C1DX (C1D Open XML) File Format

Using the C1DX (C1D Open XML) File Format

Getting Started with Reports for WPF > Using the C1DX (C1D Open XML) File Format

This release provides support for a new, OPC-based, file format for C1PrintDocument objects: C1DX. This new format complies with Microsoft Open Packaging Conventions, and is similar to the Office 2007 Open XML format. A C1DX file is a zip archive, with elements of that archive representing parts of the document. There are several advantages to the new format; the most obvious is that the resulting files are smaller in size due to built-in compression.

The following new types, properties, and methods have been added to support the new format:

- public enum C1DocumentFormatEnum
  C1DocumentFormatEnum defines the file format. Provides the following elements: C1D - the old C1D format (a single uncompressed XML file); C1DX – the new OPC-compliant format.

Specifically, new overloads have been added to C1PrintDocument.Save, C1PrintDocument.Load, C1PrintDocument.FromFile, and C1PrintDocument.FromStream methods on **C1PrintDocument**:

- public void Save(Stream stream, C1DocumentFormatEnum documentFormat)

Saves the current document to a stream using the specified format.

- public void Save(string fileName, C1DocumentFormatEnum documentFormat)

Saves the current document to a file using the specified format.

- public void Load(Stream stream, C1DocumentFormatEnum documentFormat)

Loads the current document from a stream using the specified format.

- public void Load(string fileName, C1DocumentFormatEnum documentFormat)

Loads the current document from a file using the specified format.

- public static C1PrintDocument FromFile(string fileName, C1DocumentFormatEnum documentFormat)
  Creates a new instance of **C1PrintDocument** class and loads it from a file using the specified format.

- public static C1PrintDocument FromStream(Stream stream, C1DocumentFormatEnum documentFormat)
  Creates a new instance of **C1PrintDocument** class and loads it from a stream using the specified format.

All methods that load a document from a stream, or save the document to a stream without specifying the format, currently use the old C1D format.

All methods that load a document from a file, or save the document to a file without specifying the format, try to determine the correct format by the file extension, and if that fails (for example, the file has no extension) assume the old C1D format.

Note: The default format may change in the future, so it is better to explicitly specify the format.

See Also

Design-Time Support

Exporting to XPS

Getting Started with Reports for WPF > Exporting to XPS

Reports for WPF supports exporting to XPS. In this version, XPS files created by the **C1PrintDocument** XPS module always contain a single fixed document sequence with a single

fixed document. The following **C1PrintDocument** objects are represented in the resulting XPS document as PNG images:

- RenderRichText (RTF text)
  RTF text in the resulting XPS document cannot be selected as text. This may affect the fidelity of the text, especially when printed on a high-resolution device. Note that this does not apply to **RenderText** or **RenderParagraph**, only to **RenderRichText**.
- RenderGraphics
  In the WinForms version, a **RenderGraphics** is represented by an enhanced Windows metafile, while in XPS it is represented by a PNG image. In some scenarios this may result in loss of fidelity.
- RenderText or parts of RenderParagraph objects
  Text using fonts for which embedding is not allowed.

Some image formats are not supported by the XPS standard, and if such images are present in the source document (**C1PrintDocument**), they are converted to the PNG image format. Specifically, only the following formats are preserved: JPEG, PNG, and TIFF. All other formats are converted to PNG.

Design-Time Support

Reports for WPF includes design-time support to enable you to create full-fledged reporting applications quickly and easily. Support includes a designer to make creating and customizing report definitions easy. The following topics detail the available design-time support in **Reports for WPF**.

See Also

C1DocumentViewer Context Menu

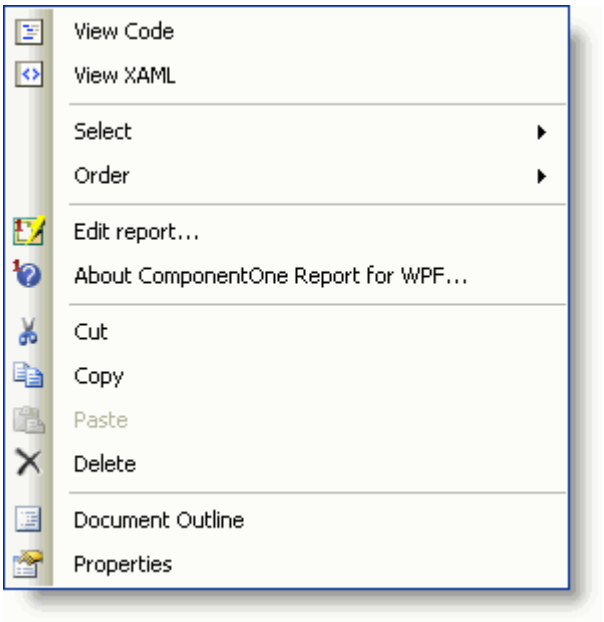
C1DocumentViewer Properties Window

C1ReportDesigner Application

C1DocumentViewer Context Menu

Design-Time Support > C1DocumentViewer Context Menu

The **C1DocumentViewer** control includes a context menu that can be accessed at design time in Visual Studio 2008 by right-clicking the control. The context menu appears similar to the following:



In addition to standard options, the context menu includes two options specific to the C1DocumentViewer control:

- Edit report

Clicking **Edit Report** opens the **C1Report Wizard** if you have not already created a report definition or the **C1ReportDesigner** if you have already created a report.

For more information on using the **C1Report Wizard**, see Creating a Basic Report Definition. For details on using the **C1ReportDesigner**, see Working with C1ReportDesigner.

Note: If the **Edit Report** command doesn't appear on the Tasks menu and Properties window, it is probably because the control could not find the **C1ReportDesigner** application. To fix this, simply run the **C1ReportDesigner** application once in stand-alone mode. The designer will save its location to the registry, and the control should be able to find it afterwards.

- About ComponentOne Report for WPF

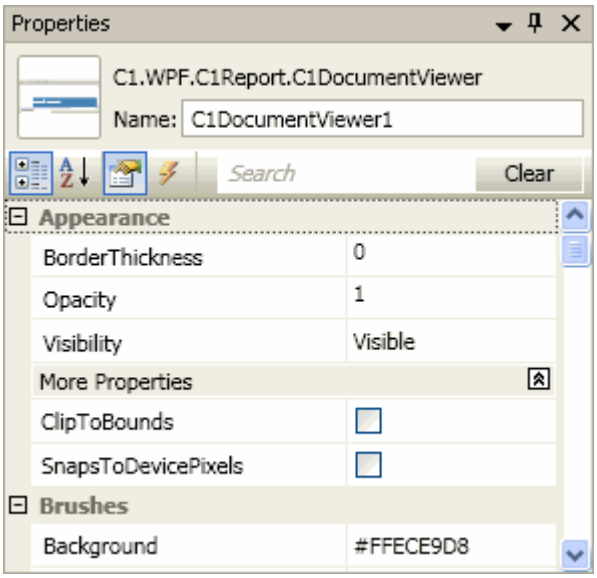
Clicking **About ComponentOne Report for WPF** displays the **About ComponentOne Report for WPF** dialog box, which is helpful in finding the version number of **Reports for WPF**, as well as online resources.

See Also

C1DocumentViewer Properties Window

C1DocumentViewer Properties Window

At design-time in Visual Studio 2008, you can access **C1DocumentViewer**'s properties and events through the Properties window. To open the Properties window, right-click on the C1DocumentViewer control and select **Properties** from the context menu. The Properties window will appear.



The Properties window enables easy access to all of the control's properties and events and allows you to set property values at design-time. The Properties window orders the properties either categorically or alphabetically; in the image above properties are ordered categorically. In order to allow the user access to objects and collections when ordered categorically, the Properties window supports a tree view structure where objects can be collapsed and expanded to hide and show constituent properties.

To view available events, click the lightning bolt **Events** button.

See Also

C1ReportDesigner Application

C1ReportDesigner Application

The **C1ReportDesigner** application can be access from the C1DocumentViewer context menu or Properties window and includes options for creating or customizing a report definition.

For more information about the **C1ReportDesigner** application, see Working with C1ReportDesigner.

See Also

Working with C1RdlReport

Working with C1RdlReport

The C1RdlReport class, a class that represents an RDL (Report Definition Language) report defined using the 2008 version of the RDL specification. The C1RdlReport class is similar to the C1Report class with the addition of RDL support.

RDL import in C1PrintDocument (provided by **ImportRdl** and **FromRdl** methods) is now obsolete. C1RdlReport should be used instead.

Important Notes for .NET 2.0/3.0 version users:

The C1.C1Report.2 assembly uses the **System.Windows.Forms.DataVisualization.dll** assembly that is only included in .NET Framework 3.5 and later. The assembly is installed on your system with when you install **Reports for WPF**. It MUST BE INCLUDED with other Reports assemblies when you deploy your application that using **Reports for WPF** to other systems.

Also, if you update the **Reports for WPF** DLLs manually, you MUST put **DataVisualization** where those assemblies are, and make sure that your project references it. This does not apply to .NET 4.0 users as **DataVisualization** is already included in the Framework.

See Also

Report Definition Language (RDL)

C1RdlReport Advantages

C1RdlReport Limitations

Loading an RDL file

Report Definition Language (RDL)

Report Definition Language (RDL) is a Microsoft-standard XML schema for representing reports. The goal of RDL is to promote the interoperability of reporting products by defining a common schema that allows interchange of report definitions. RDL is designed to be output format neutral. This means that reports defined using RDL should be able to be outputted to a variety of formats including Web and print-ready formats or data-focused formats like XML.

The C1RdlReport class was added to **Reports for WPF** in the 2010 v3 release to leverage the flexibility of RDL files. Using C1RdlReport, you can easily import RDL files into your reporting applications – adding flexibility and functionality through the use of the familiar format.

See Also

C1RdlReport Advantages

C1RdlReport Advantages

Working with C1RdlReport > C1RdlReport Advantages

The C1RdlReport control has several advantages over using the standard Microsoft Reporting Services. The major advantages provided by C1RdlReport include:

- Support for the current RDL 2008 specification.
- Programmatic access to the RDL object model (which follows the 2008 RDL specification) - this allows you to modify existing or create new RDL reports completely in code.
- Generation of RDL reports that can consume any data source (such as .mdb files).
- A self-contained RDL reporting solution without external dependencies such as the need to access a Microsoft Reporting Services server.
- Seamless integration with other **Reports for WPF** reporting engines including C1Report and C1PrintDocument.

See Also

C1RdlReport Limitations

C1RdlReport Limitations

Working with C1RdlReport > C1RdlReport Limitations

The C1RdlReport control supports most of the features of RDL. However, the initial release of C1RdlReport includes some limitations.

The following objects are not supported:

- Gauge objects are not supported.

The following RDL properties are not currently supported:

- Document.AutoRefresh
- Document.Width
- Document.Language
- Document.DataTransform
- Document.DataSchema
- Document.DataElementName
- Document.DataElementStyle
- DataSet.CaseSensitivity
- DataSet.Collation
- DataSet.AccentSensitivity
- DataSet.KanatypeSensitivity
- DataSet.WidthSensitivity
- DataSet.InterpretSubtotalsAsDetails
- TextBox.UserSort

- TextBox.ListStyle
- TextBox.ListLevel
- TextRun.ToolTip
- TextRun.MarkupType

Important Note for .NET 2.0/3.0 version users:

The C1.C1Report.2 assembly uses the **System.Windows.Forms.DataVisualization.dll** assembly that is only included in .NET Framework 3.5 and later. The assembly is installed on your system with when you install **Reports for WPF**. It MUST BE INCLUDED with other Reports assemblies when you deploy your application that using **Reports for WPF** to other systems.

Also, if you update the **Reports for WPF** DLLs manually, you MUST put **DataVisualization** where those assemblies are, and make sure that your project references it. This does not apply to .NET 4.0 users as **DataVisualization** is already included in the Framework.

See Also

Loading an RDL file

Loading an RDL file

To load an RDL file into the C1RdlReport class you can use the C1RdlReportBase.Load method. To remove an RDL file, you would use the C1RdlReport.Clear method. This method clears any RDL file previously loaded into the C1RdlReport control.

To load an RDL file into the C1RdlReport component you can use the C1RdlReportBase.Load method:

Visual Basic	Copy Code

```
C1RdlReport1.Load("C:/Report.rdl")
C1RdlReport1.Render()
```

C#	Copy Code

```
c1RdlReport1.Load(@"C:/Report.rdl");
c1RdlReport1.Render();
```

See Also

Working with C1ReportDesigner

Working with C1ReportDesigner

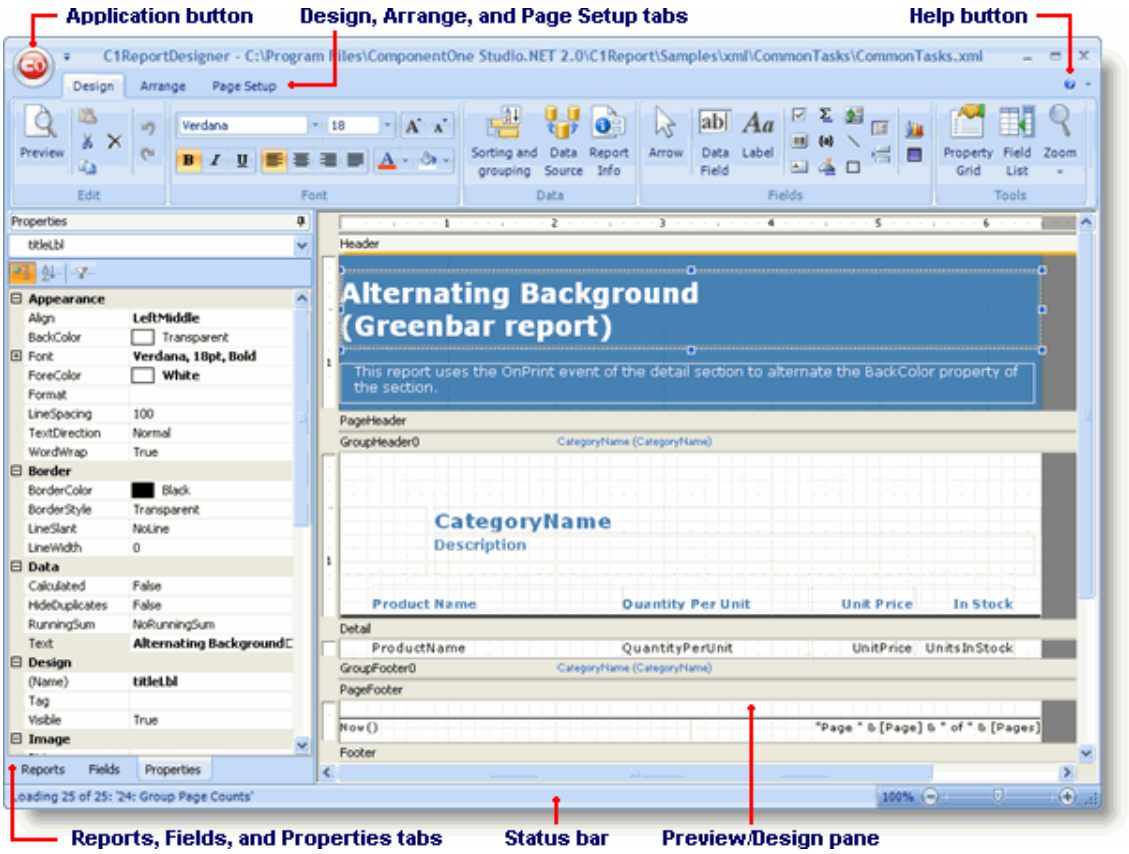
About C1ReportDesigner

Working with C1ReportDesigner > About C1ReportDesigner

The **C1ReportDesigner** application is a tool used for creating and editing C1.C1Report.C1Report report definition files. The Designer allows you to create, edit, load, and save XML files that can be read by the C1.C1Report.C1Report component. It also allows you to import report definitions from Microsoft Access files (MDB) and VSReport 1.0 (VSR).

To run the Designer, double-click the **C1ReportDesigner.exe** file by default installed in **C:\Program Files\ComponentOne\WPF Edition\C1WPFReport\Designer**. For steps on running the Designer from Visual Studio, see Accessing C1ReportDesigner from Visual Studio.

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



The main Designer window has the following components:

- **Application button:** Click the application button to open a menu to load and save report definition files and to import and export report definitions.
- **Design tab:** Provides shortcuts to the Edit, Font, Data, Fields, and Tools menu functions.
- **Arrange tab:** Provides shortcuts to the Edit, AutoFormat, Grid, Control Alignment, Position, and Size menu functions.

- **Page Setup tab:** Provides shortcuts to the Edit and Page Layout menu functions.

- **Report tab:** Lists all reports contained in the current report definition file. You can double-click a report name to preview or edit the report. You can also use the list to rename, copy, and delete reports.

- **Fields tab:**Lists all the fields contained in the current report.

- **Properties tab:** Allows you to edit properties for the objects that are selected in the Designer.

- **Preview/Design pane:** This is the main working area of the Designer. In preview mode, it displays the current report. In design mode, it shows the report's sections and fields and allows you to change the report definition.

- **Status bar:** The status bar displays information about what the Designer is working on (for example, loading, saving, printing, rendering, importing, and so on).

The following sections explain how you can use the **C1ReportDesigner** to create, edit, use, and save report definition files.

The main Designer window has the following components:

- **Application button:** Click the application button to open a menu to load and save report definition files and to import and export report definitions. See Application Button for more information.

- **Design tab:** Provides shortcuts to the Edit, Font, Data, Fields, and Tools menu functions. See Design Tab for more information.

- **Arrange tab:** Provides shortcuts to the Edit, AutoFormat, Grid, Control Alignment, Position, and Size menu functions. See Arrange Tab for more information.

- **Page Setup tab:** Provides shortcuts to the Edit and Page Layout menu functions. See Page Setup Tab for more information.

- **Preview tab:** Appears only when the report is being viewed in a print preview. See Preview Tab for more information.

- **Help button:** Provides options to open the online help file and view the About screen, which displays information about the application.

- **Reports tab:** Lists all reports contained in the current report definition file. You can double-click a report name to preview or edit the report. You can also use the list to rename, copy, and delete reports.

- **Fields tab:** Lists all the fields contained in the current report.

- **Properties tab:** Allows you to edit properties for the objects that are selected in the Designer.

- **Pages tab:** Only available when in preview mode, this tab includes thumbnails of all the pages in the document.
- **Properties tab:** Only available when in preview mode, this tab displays a text outline of the document.
- **Find tab:** Only available when in preview mode, this tab displays fine pane allowing you to search for text in the document.
- **Preview/Design pane:** This is the main working area of the Designer. In preview mode, it displays the current report. In design mode, it shows the report's sections and fields and allows you to change the report definition.
- **Status bar:** The status bar displays information about what the Designer is working on (for example, loading, saving, printing, rendering, importing, and so on). You can zoom in and out of a selected report by dragging the zoom slider at the right of the status bar.

The topics that follow explain how you can use the **C1ReportDesigner** application to create, edit, use, and save report definition files.

See Also

Application Button

Arrange Tab

Design Tab

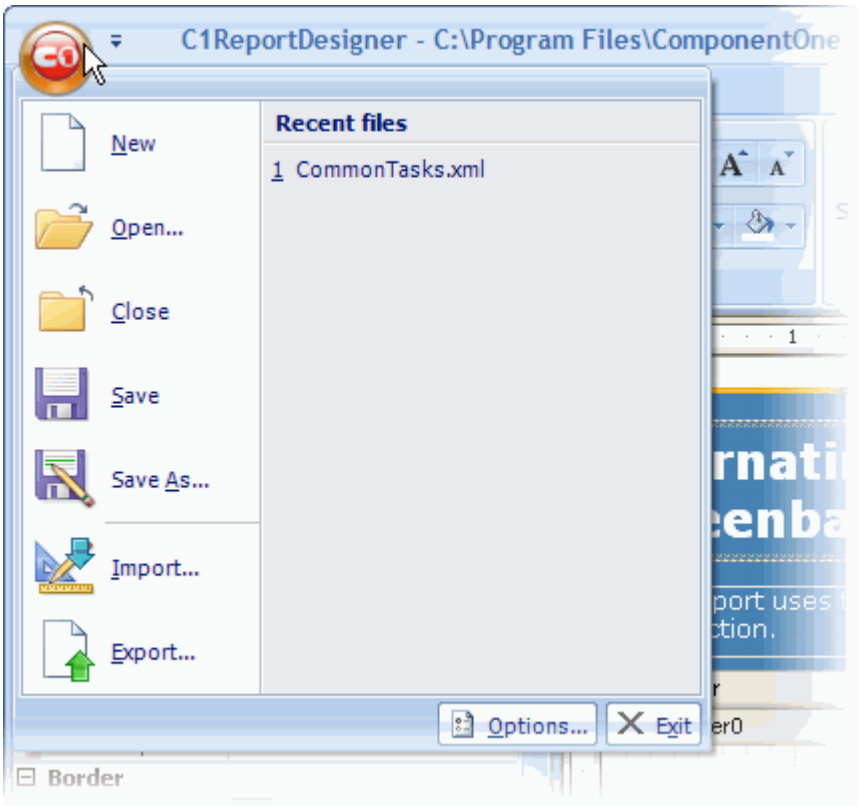
Page Setup Tab

Preview Tab

Application Button

The **Application** button provides a shortcut menu to load and save report definition files and to import and export report definitions. You can also access the **C1ReportDesigner** application's options through the **Application** menu.

Click the **Application** button to open the menu. It will appear similar to the following:



The menu includes the following options:

- **New:** Creates a new report definition file.
- **Open:** Brings up the Open Report Definition File dialog box, enabling you to select an existing file to open.
- **Close:** Closes the current report definition file.
- **Save:** Saves the report definition file, to the location previously saved.
- **Save As:** Opens the Save Report Definition dialog box allowing you to save your report definition as an XML file.
- **Import:** Opens the Import Report Definition dialog box enabling you to import Microsoft Access (.mdb and .adp) files and Crystal Reports (.rpt) files. See Importing Microsoft Access Reports and Importing Crystal Reports for more information.
- **Export:** Exports the current report file as an HTML, PDF, RTF, XLS, XLSX, TIF, TXT, or ZIP file.
- **Recent files:** Lists recently opened report definition files. To reopen a file, select it from the list.
- **Options:** Opens the C1ReportDesigner Options dialog box which allows you to customize the default appearance and behavior of the C1ReportDesigner application. See Setting C1ReportDesigner Options for more information.

- **Exit:** Closes the C1ReportDesigner application.

See Also

Arrange Tab

Arrange Tab

Working with C1ReportDesigner > About C1ReportDesigner > Arrange Tab

The **Arrange** tab is located in the **C1ReportDesigner**'s Ribbon menu and provides shortcuts to the Edit, AutoFormat, Grid, Control Alignment, Position, and Size menu functions. See Edit Group for information about that group, and the following topics for the other groups.

See Also

Edit Group

Font Group

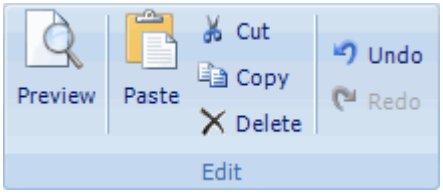
Grid Group

Tools Group

Edit Group

Working with C1ReportDesigner > About C1ReportDesigner > Arrange Tab > Edit Group

The **Edit** group on the **Design** tab appears similar to the following:



It consists of the following options:

- **Preview**: The Preview button opens a print preview view of the report. To exit the preview, click the **Close Print Preview** button. See Preview Tab and Previewing and Printing a Report for more information.

- **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.

- **Undo**: Undos the last change that was made to the report definition.

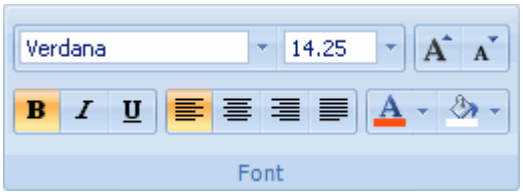
- **Redo**: Redos the last change that was made to the report definition.

See Also

Font Group

Font Group

The **Font** group on the **Design** tab appears similar to the following:



And consists of the following options:

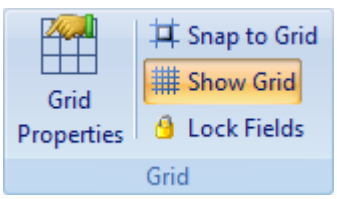
- **Font Name:** Displays the current font of the selected text and allows you to choose another font for the selected item (to do so click the drop-down arrow next to the font name).
- **Font Size:** Displays the current font size of the selected text and allows you to choose another font size. Type 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:** Bold the selected text (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).
- **Left:** Aligns text to the left.
- **Center**: Aligns text to the center.
- **Right:** Aligns text to the right.
- **Justify:** Justifies the selected text.
- **Text Color:** Allows you to select the color of the selected text.
- **Fill Color:** Allows you to select the background color of the selected text.

See Also

Grid Group

Grid Group

The **Grid** group on the **Arrange** tab of the Ribbon appears similar to the following image:



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

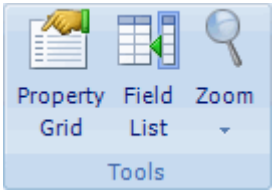
- **Grid Properties:** Opens the C1ReportDesigner Options dialog box. For details see Setting C1ReportDesigner Options.
- **Snap to Grid:** Snaps elements to the grid. When this item is selected elements cannot be placed in between lines of the grid.
- **Show Grid:** Shows a grid in the background of the report in the preview. The grid can help you place and align elements. By default, this option is selected.
- **Lock Fields:** After you've placed the fields in the desired positions, you can lock them to prevent inadvertently moving them with the mouse or keyboard. Use the Lock Fields button to lock and unlock the fields.

See Also

Tools Group

Tools Group

The **Tools** group on the **Design** tab appears similar to the following:



It consists of the following options:

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

See Also

Design Tab

Design Tab

The **Design** tab is located in the **C1ReportDesigner**'s Ribbon menu and provides shortcuts to the Edit, Font, Data, Fields, and Tools menu functions. For more information, see the following topics.

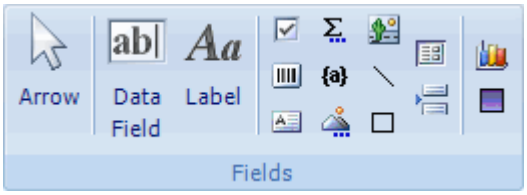
See Also

Field Group

Data group

Field Group

The **Fields** group of the **Design** tab in the **C1ReportDesigner** application provides tools for creating report fields. This toolbar is enabled only in design mode. The **Fields** group on the **Design** tab appears similar to the following:



Each field button creates a field and initializes its properties. The **Fields** group consists of the following options:

- **Arrow:** Returns the mouse cursor to an arrow cursor.
- **Data Field:** Creates a field that is bound to the source recordset. When you click this button, a menu appears and you can select the recordset field. Bound fields are not limited to displaying raw data from the database. You can edit their C1.C1Report.Field.Text property and use any VBScript expression.
- **Label:** Creates a field that displays static text.
- **Add 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 C1.C1Report.Field.Checkbox property after it has been created.
- **Add 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.
- **Add Rtf Field ():** Creates an RTF field. When you click this button, a menu appears where you can select other fields that are contained in the same report definition file to be displayed in RTF format.
- **Add Calculated Field ():** Creates a calculated field. When you click this button, the code editor dialog box appears so you can enter the VBScript expression whose value you want to display.
- **Add Common Calculated Field ():** Creates a field with a commonly used expression. When you click this button, a menu appears and you can select 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.
- **Add Unbound Picture ():** Creates a field that displays a static picture, such as a logo. When you click this button, a dialog box appears to prompt you for a picture file to insert in the report. A copy is made of the picture you select and placed in the same directory as the report file. You must distribute this file with the application unless you embed the report file in the application. When you embed a report file in your application, any unbound picture files are embedded too.
- **Add Bound Picture ():** Creates a field that displays a picture (or object) stored in the recordset. When you click this button, a menu appears so you can select a picture field in the source recordset (if there is one; not all recordsets contain this type of field).
- **Add Line ():** Creates a line. Lines are often used as separators.
- **Add Rectangle ():** Creates a rectangle. Rectangles are often used to highlight groups of fields or to create tables and grids.

- **Add SubReport ():** Creates a field that displays another report. When you click this button, a menu appears and you can select other reports that are contained in the same report definition file. See Creating a Master-Detail Report Using Subreports for more information.
- **Add Page Break ():** Creates a field that inserts a page break.
- **Add Chart Field ():** Creates a field that displays a chart. Unlike most bound fields, Chart fields display multiple values. To select the data you want to display, set the Chart field's Chart.DataX and Chart.DataY properties. To format the values along the X and Y axis, set the Chart.FormatX and Chart.FormatY properties. See Adding Chart Fields for more information.
- **Add Gradient Field ():** Creates a gradient field. Gradients are often used as a background feature to make other fields stand out. See Adding Gradient Fields for more information.

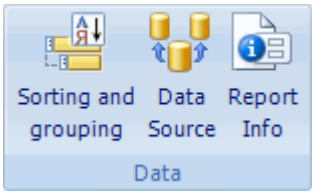
See Enhancing the Report with Fields for more information. For more information on adding fields to your report, see Creating Report Fields.

See Also

Data group

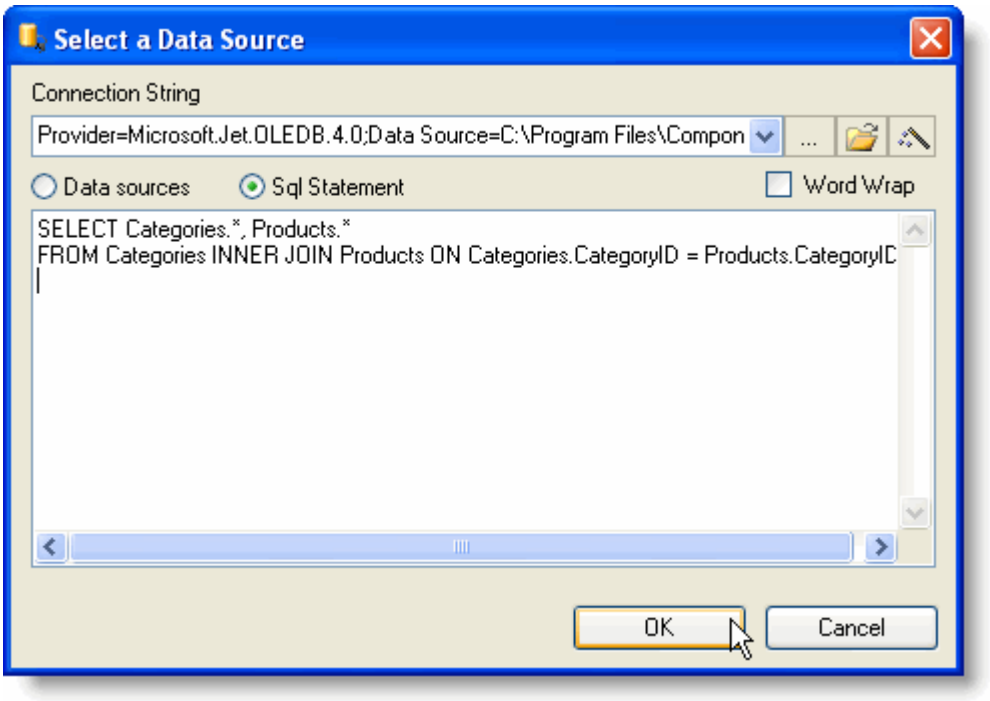
Data group

The **Data** group appears similar to the following image:



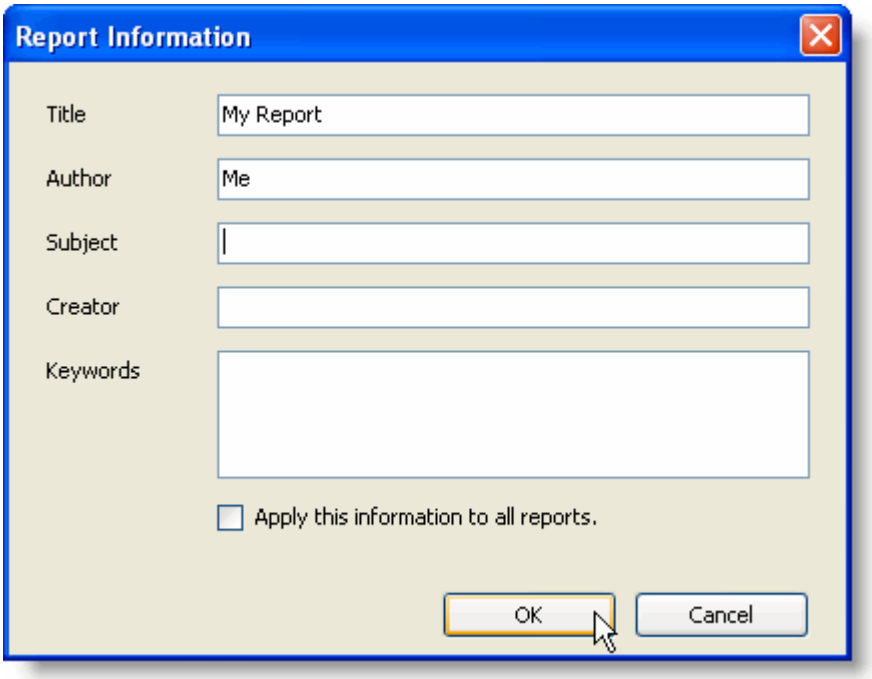
And consists of the following items:

- **Sorting and grouping:** Clicking this button opens the Sorting and Grouping dialog box where you can add and delete sorting and grouping criteria.
- **Data Source:** Clicking this button opens the Select a Data Source dialog box. The Select a Data Source dialog box allows you to choose a new data source, change the connection string, and edit the SQL statement. Clicking the Data sources radio button displays the tables and views in the current data source. Clicking the Sql statement radio button allows you to view the current SQL statement:



To select change the connection string, click the ellipses button. This will open the **Data Link Properties** dialog box. To open an XML Schema Definition file (XDS) click the **Open** button. To edit or change the SQL statement, click the **Build SQL Statement** button which will open the **Sql Builder** dialog box.

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



Page Setup Tab

The **Page Setup** tab is located in the **C1ReportDesigner**'s Ribbon menu and provides shortcuts to the Edit and Page Layout menu functions. See Edit Group and Page Layout Group for more information.

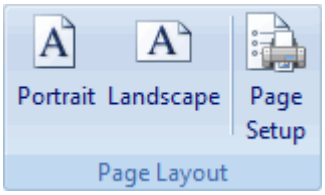
See Also

Page Layout Group

Auto Format Group

Page Layout Group

The **Page Layout** group on the **Page Setup** tab of the Ribbon appears similar to the following image:



The **Page Layout** group 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.

See Also

Auto Format Group

Auto Format Group

The **Arrange** tab is located in the **C1ReportDesigner**'s Ribbon menu and provides shortcuts to the Edit, AutoFormat, Grid, Control Alignment, Position, and Size menu functions. See Edit Group for information about that group, and the following topics for the other groups.

See Also

Preview Tab

Preview Tab

The **Preview** tab is located in the **C1ReportDesigner**'s Ribbon menu and provides shortcuts to the Print, Page Layout, Zoom, Navigation, Tools, Export, and Close Preview menu functions. To access the Preview tab and a print preview of your report, click the **Preview** option located in the **Edit** group of the Ribbon. See Edit Group for more information.

See Page Layout Group for information about that group, and the following topics for the other groups in the **Preview** tab.

See Also

Print Group

Zoom Group

Navigation Group

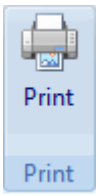
Tools Group

Export Group

Close Preview Group

Print Group

The **Print** group on the **Preview** tab of the Ribbon appears similar to the following image:



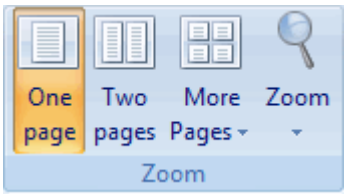
The **Print** group consists of the Print option. Clicking the **Print** option opens your printer options for printing the report.

See Also

Zoom Group

Zoom Group

The **Zoom** group on the **Preview** tab of the Ribbon controls how the print preview is zoomed and appears similar to the following image:



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

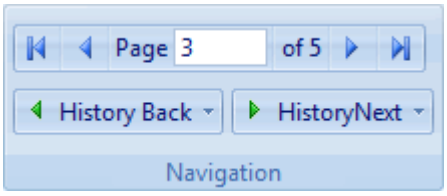
- **One page**: Allows you to preview one page at a time.
- **Two pages**: Allows you to preview two pages at a time.
- **More pages**: Clicking the drop-down arrow allows you to preview multiple pages at a time and includes the following options: **Four pages**, **Eight pages**, **Twelve pages**.
-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

See Also

Navigation Group

Navigation Group

The **Navigation** group on the **Preview** tab of the Ribbon controls how the print preview is navigated and appears similar to the following image:



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

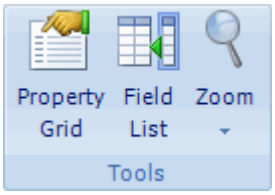
- **First page:** Navigates to the first page of the preview.
- **Previous page:** Navigates to the previous page of the preview.
- **Page:** Entering a number in this textbox navigates the preview to that page.
- **Next page:** Navigates to the next page of the preview.
- **Last page:** Navigates to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

See Also

Tools Group

Tools Group

The **Tools** group on the **Preview** tab of the Ribbon contains tools for selecting and locating items in the preview and appears similar to the following image:



The **Tools** group consists of the following buttons:

- **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:

You can click a search result to locate it in the document.

See Also

Export Group

Export Group

The **Export** group on the **Preview** tab of the Ribbon contains options for exporting the report to various formats and appears similar to the following image:

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

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

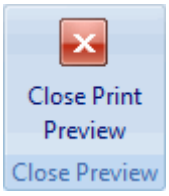
See Also

Close Preview Group

Close Preview Group

Working with C1ReportDesigner > About C1ReportDesigner > Preview Tab > Close Preview Group

The **Close Preview** group on the **Preview** tab of the Ribbon appears similar to the following image:



It includes just one button, **Close Print Preview**, enabling you to close the preview view and return to the design view. To return to the Review view again, click the **Preview** button in the **Edit** group.

See Also

Accessing C1ReportDesigner from Visual Studio

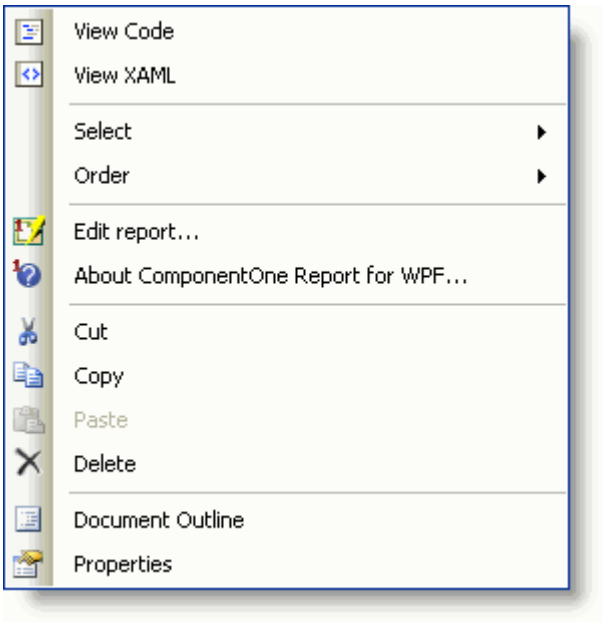
Accessing C1ReportDesigner from Visual Studio

Working with C1ReportDesigner > About C1ReportDesigner > Accessing C1ReportDesigner from Visual Studio

You can access the **C1ReportDesigner** application from Visual Studio through the **C1DocumentViewer**'s content menu or Properties window:

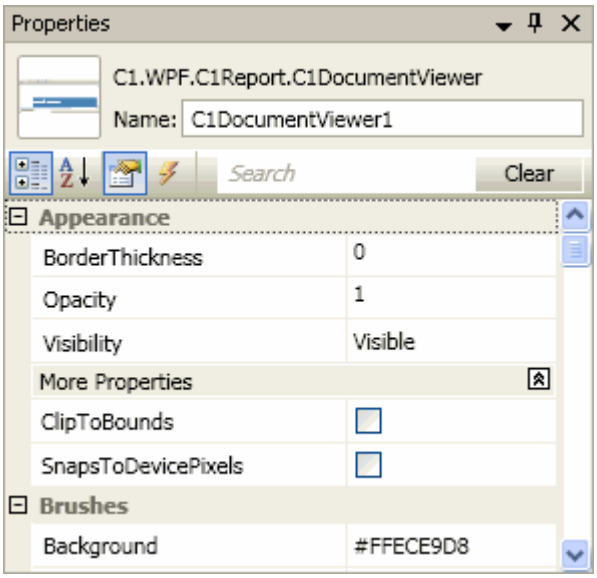
- **C1DocumentViewer Context Menu**

Right-click on the C1DocumentViewer control and select **Edit Report** from the context menu that appears.



- **C1DocumentViewer Properties Window**

Expand the **C1Report** node and click on the **Edit Report** link.



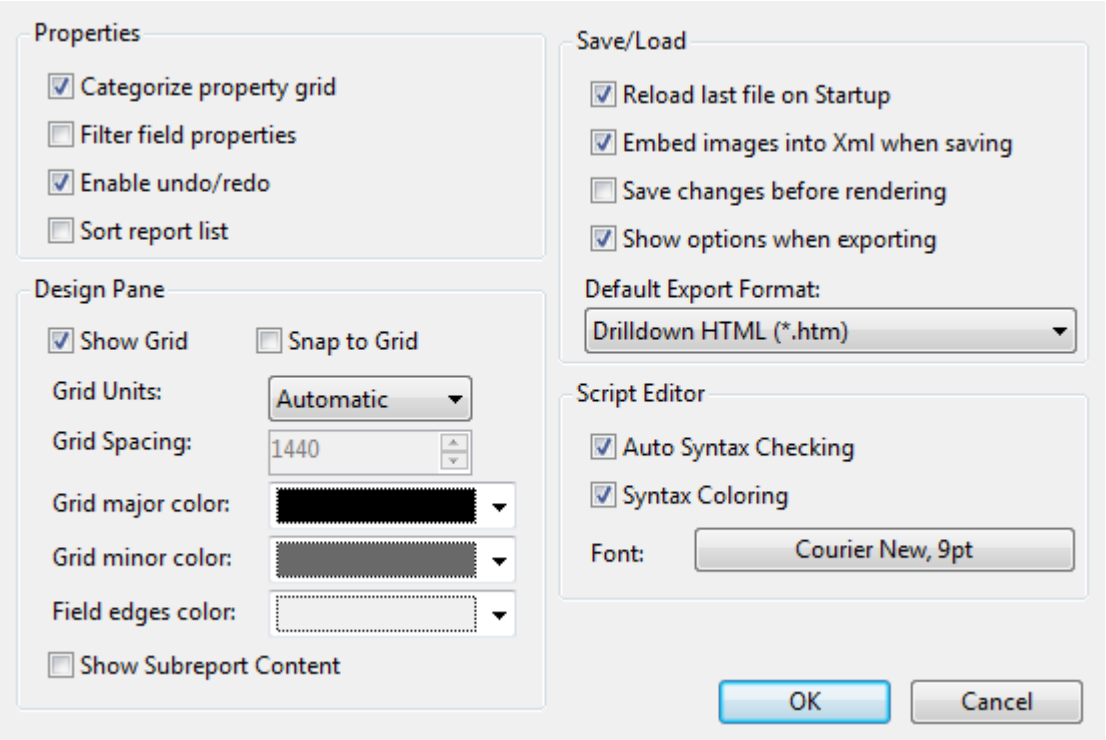
See Also

Setting C1ReportDesigner Options

Setting C1ReportDesigner Options

Working with C1ReportDesigner > About C1ReportDesigner > Setting C1ReportDesigner Options

To access the **C1ReportDesigner Options** dialog box, click the **Application** button and then **Options**. For more information about the **Application** button, see Application Button. The **C1ReportDesigner Options** dialog box appears similar to the following image:



The **C1ReportDesigner Options** dialog box includes options to control the appearance and behavior of the application. Options include:

- **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.
- **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.
- **Enable undo/redo:** Enables undo and redo in the application.
- **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.
- **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.
- **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.
- **Show Subreport Content:** Shows sub-report content in the report.
- **Reload last file on Startup:** If this option is checked, the last opened file will appear whenever the C1ReportDesigner application is opened.
- **Embed images into Xml when saving:** When the report is saved, images will be embedded into XML if this option is checked.
- **Save changes before rendering:** Checking this option saves the report before rendering.

- **Default Export Format:** Sets the default export format. For more information about exporting see Export Group.
- **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.
- **OK:** Click OK to save your changes.
- **Cancel:** Click Cancel to cancel any changes that you have made.

See Also

Creating a Basic Report Definition

Creating a Basic Report Definition

You can design your report to display your data in a variety of ways on the printed page. Using the **C1ReportDesigner** application, you can design comprehensive lists, summaries, or special subsets of data, like an invoice.

The easiest way to start a new report is to use the **C1Report Wizard**. In the **C1Report Wizard** click the **Application** button and select **New** from the menu to create a new report. To access the **C1Report Wizard**, click the **New Report** button from the **Reports** tab:

The **C1Report Wizard** appears, and will guide you through five easy steps:

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

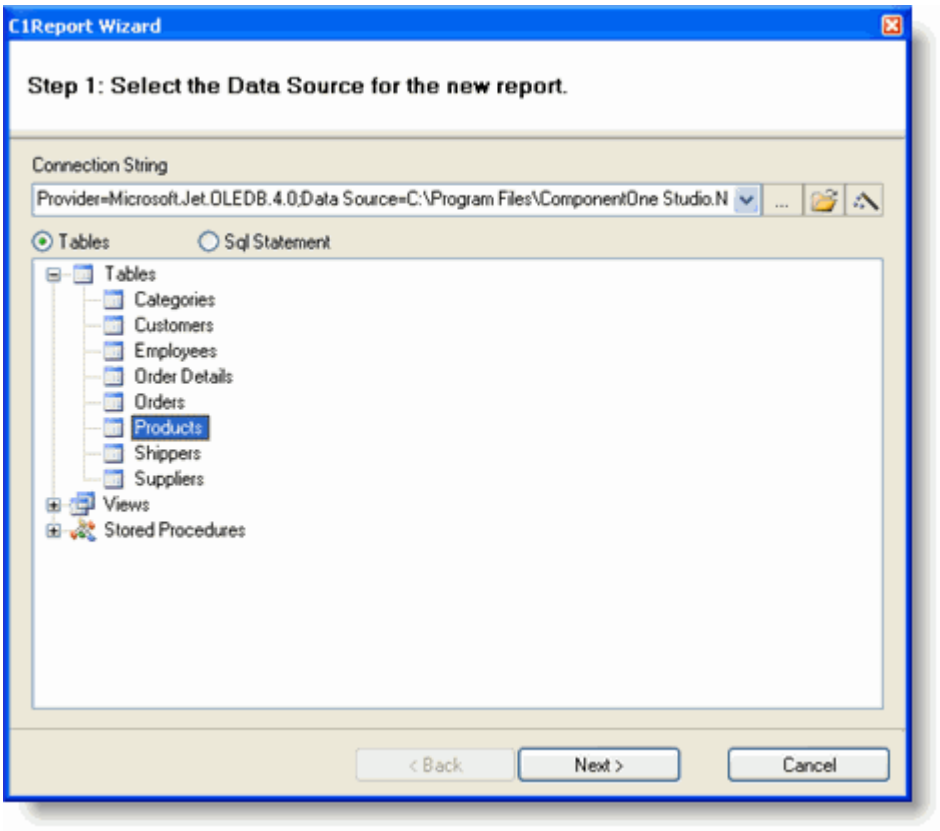
Use this page to select the C1.C1Report.DataSource.ConnectionString and C1.C1Report.DataSource.RecordSource that will be used to retrieve the data for the report.

You can specify the C1.C1Report.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 button with the ellipsis button (...) to bring up the standard connection string builder.

You can specify the C1.C1Report.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.

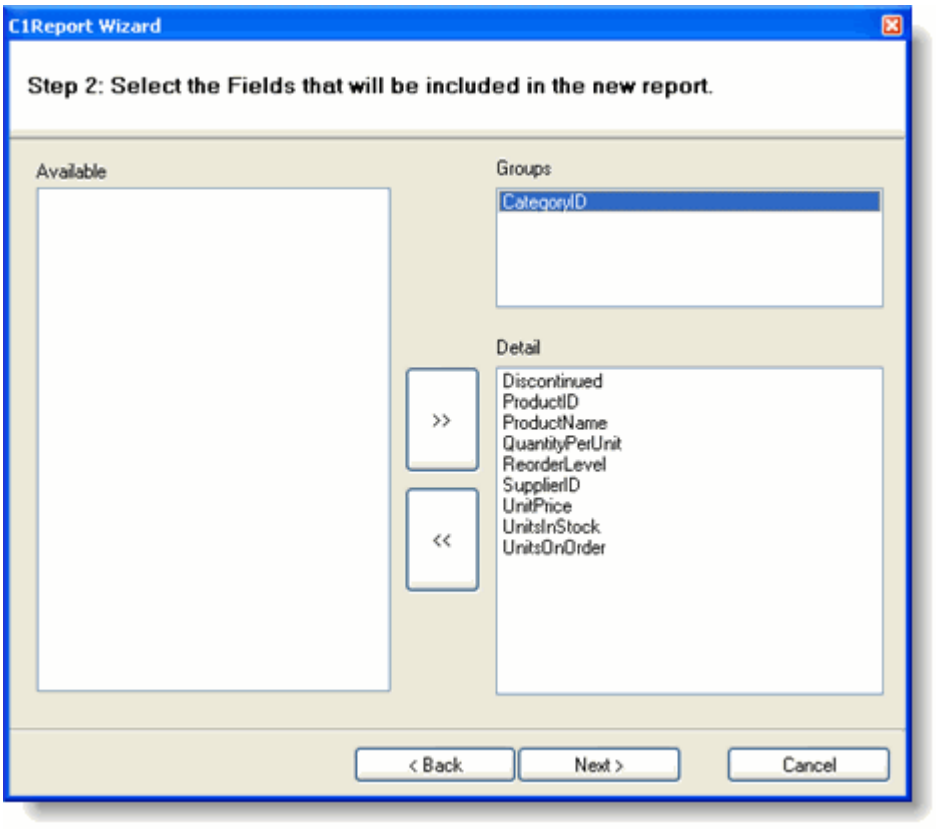


2. **Select the fields you want to include in the report.**

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 performing a drag-and-drop operation. 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.

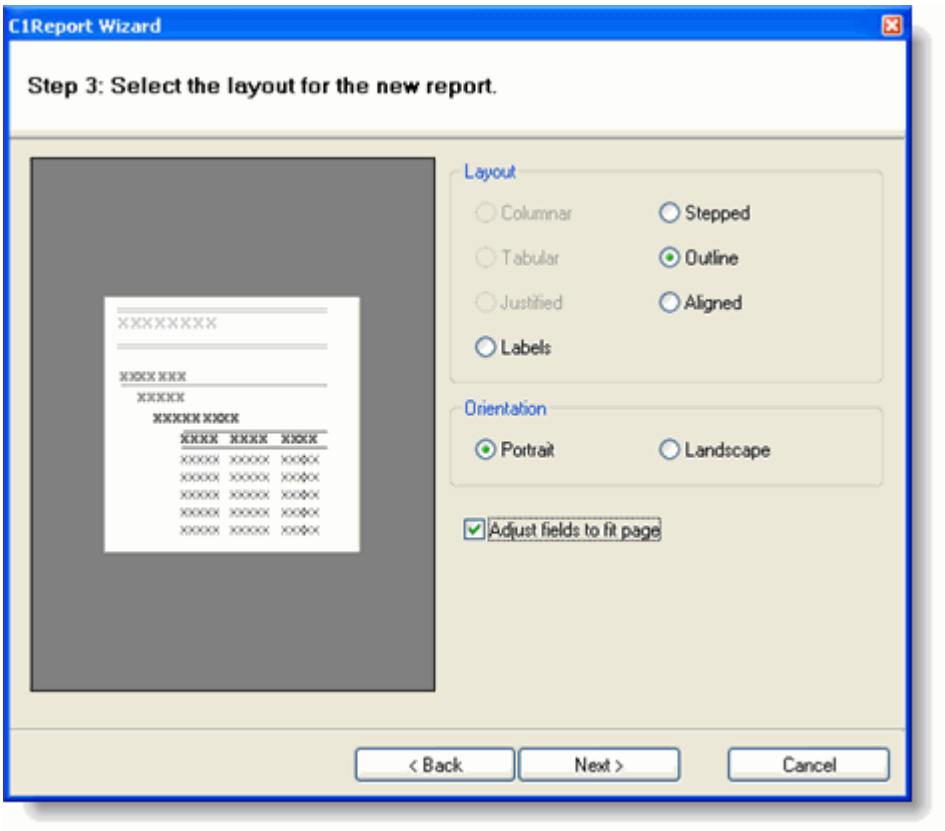
Use the **>>** button to add all fields in the **Available** list to the **Detail** list. Use the **<<** button to move all fields in the **Detail** list to the **Available** list.



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.

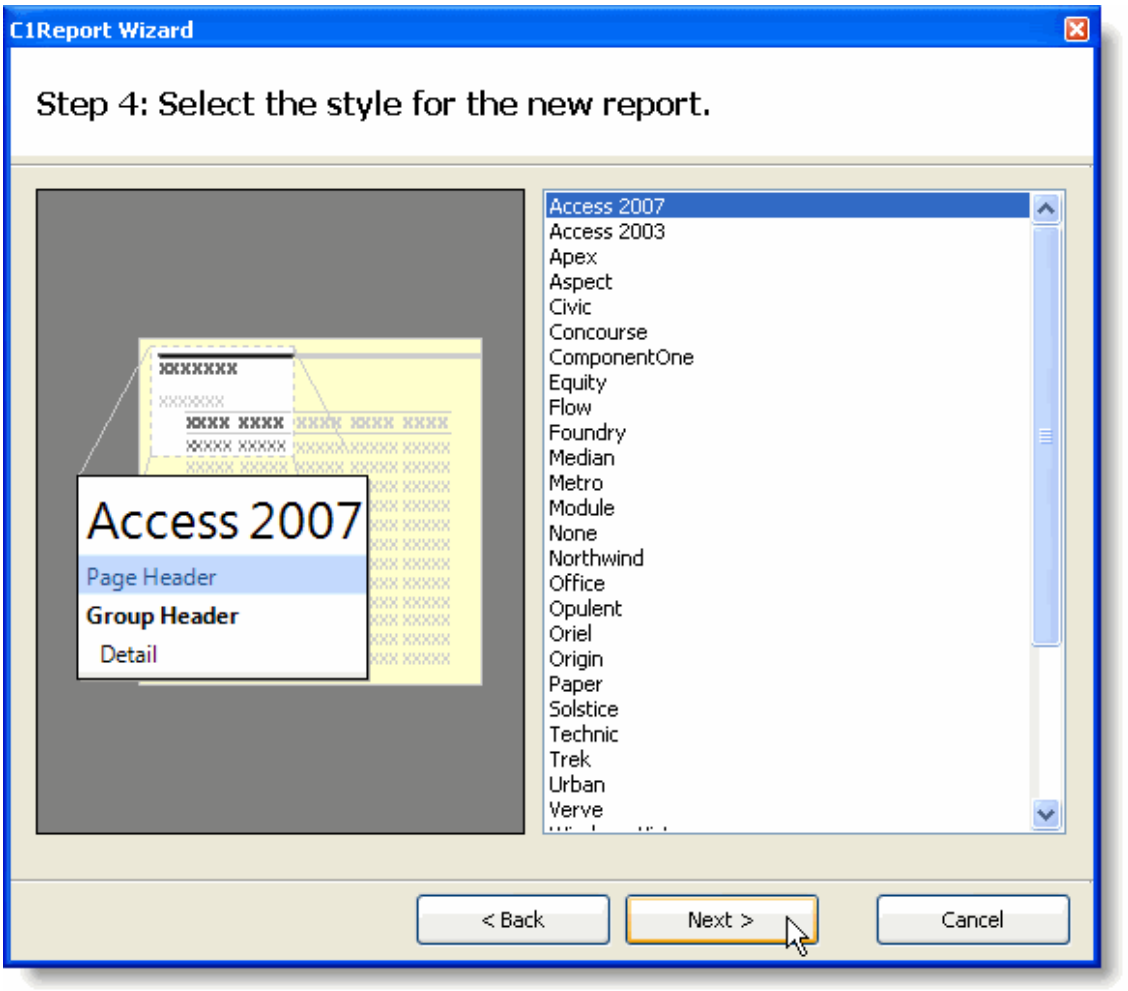


The **Labels** layout option is used to print Avery-style labels. If you select this option, you will see a page that prompts you for the type of label you want to print.

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

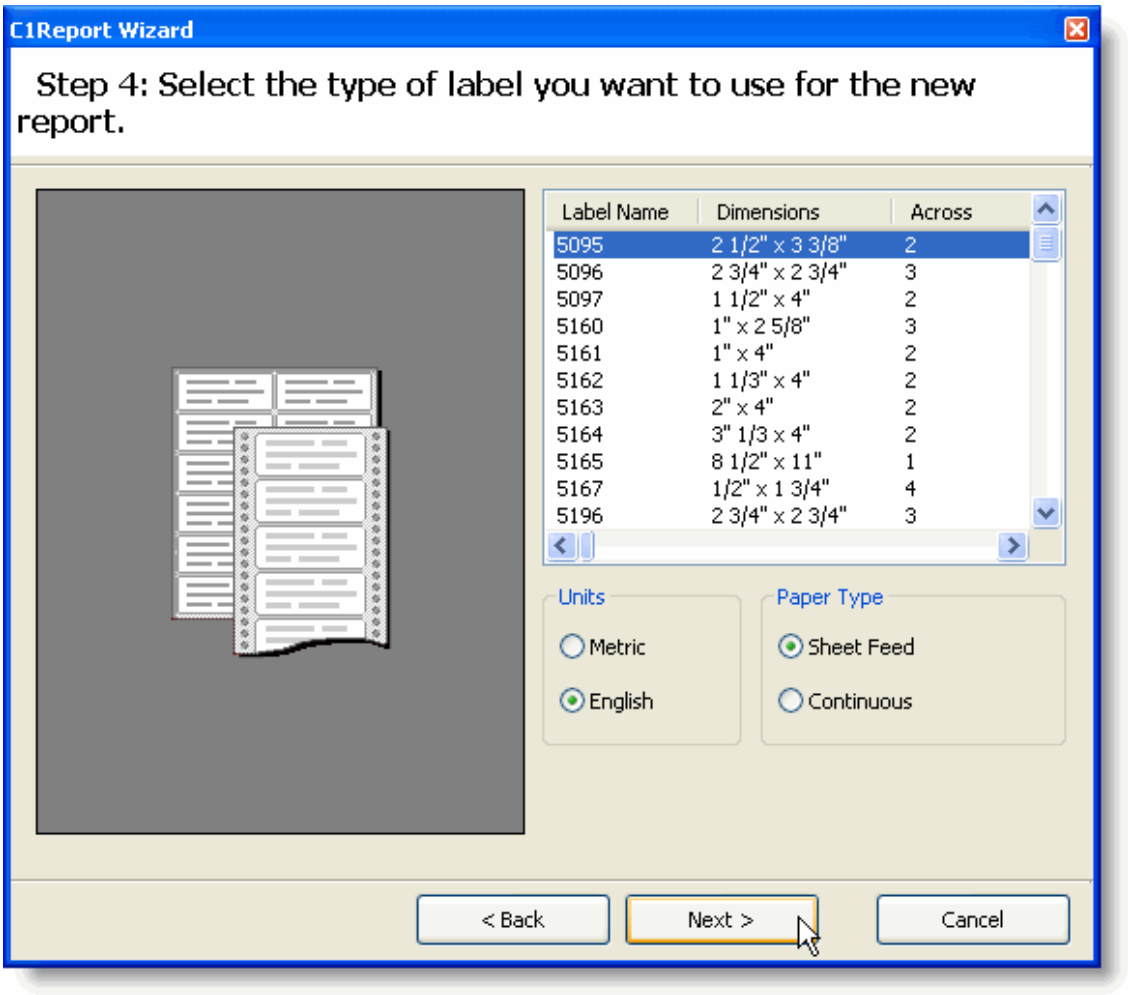
- **Style Layouts**

This page allows you to select the style 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).



- **Label Layout Only**

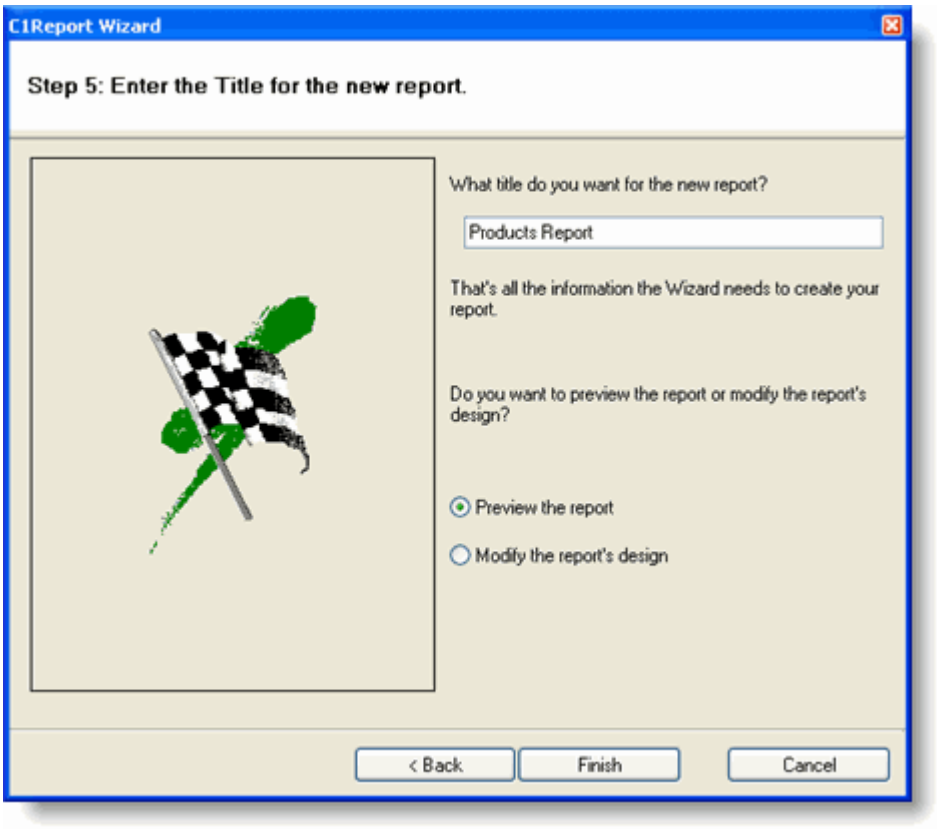
This page allows you to select the type of label you want to create. The Designer has over 170 predefined label types for you to choose from. The labels are divided into four groups, depending on whether they use metric or English measurements, and on the type of paper they use (sheets or continuous forms).



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.

If you choose to preview the report and click finish, you will immediately see the report in the preview pane of the **Designer**. For example, it might look like the following image:



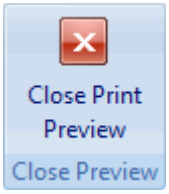
See Also

Modifying the Report Layout

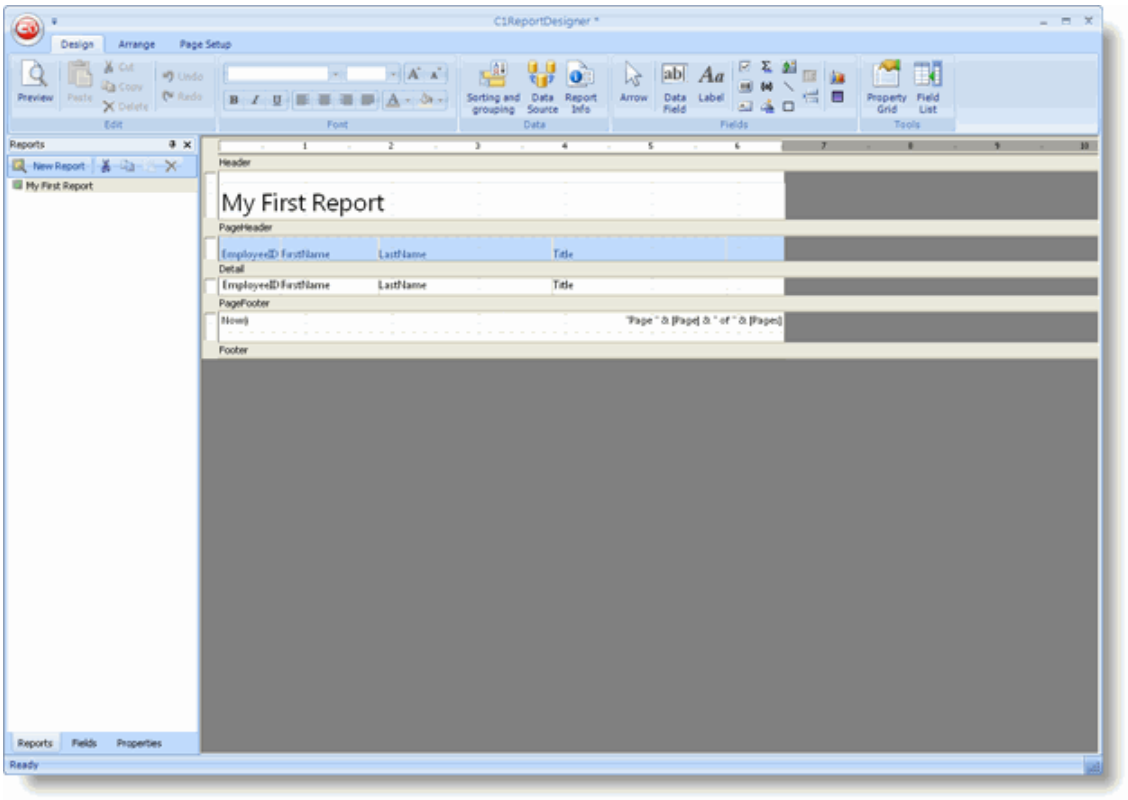
Modifying the Report Layout

The report generated for you by the wizard is a good starting point, but you'll usually need to adjust and enhance it to get exactly what you want. You can customize your report with the **C1ReportDesigner**.

To start using the Designer, click the **Close Print Preview** button in the **Close Preview** group of the **Preview** tab:



The right pane of the main window switches from Review mode into Design mode, and it shows the controls and fields that make up the report:



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

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

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

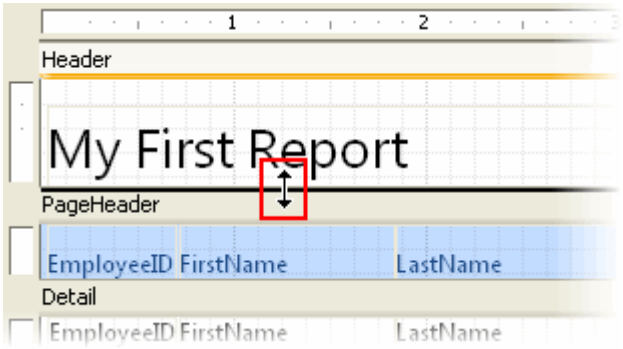
Note that you cannot add and delete sections directly. The number of groups determines the number of sections in a report. Every report has exactly five fixed sections (Report Header/Footer, Page Header/Footer, and Detail) plus two sections per group (a Header and Footer). You can hide sections that you don't want to display by setting their C1.C1Report.Section.Visible property to **False**.

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

Resizing a Section

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

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



Release the mouse button and the section is resized.

See Also

Enhancing the Report with Fields

Enhancing the Report with Fields

Working with C1ReportDesigner > About C1ReportDesigner > Enhancing the Report with Fields

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

Report Fields

The **Fields** group of the **Design** tab in the **C1ReportDesigner** application provides tools for creating report fields. This toolbar is enabled only in design mode. Each button creates a field and initializes its properties. For more information about the **Fields** group, see Fields Group. For more information on adding fields to your report, see Creating Report Fields.

See Also

Adding Chart Fields

Adding Gradient Fields

Selecting, Moving, and Copying Fields

Changing the Data Source

Adding Chart Fields

Chart fields are implemented using the **C1Chart** control.

Note: You must deploy the **C1Chart** assembly with your application if you use charts.

Note: You must deploy the **C1Chart** assembly with your application if you use charts.

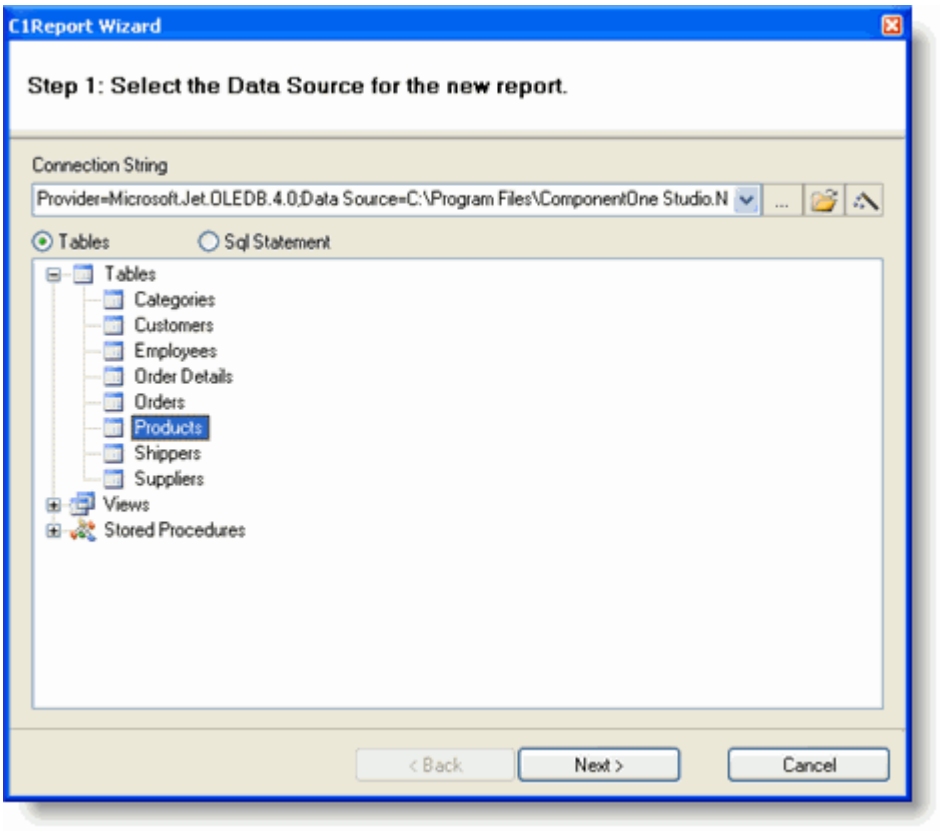
To add a Chart field to your report, complete the following steps:

1. Open the report in the C1ReportDesigner application.
2. Click the Add Chart Field button in the toolbar, and mark the area in the report where the Chart should be displayed.
3. Then set the field properties as usual.

The only unusual aspect of Chart fields is that unlike most bound fields, they display multiple values. To select the data you want to display, set the Chart field's **Chart.DataX** and **Chart.DataY** properties. To format the values along the X and Y axis, set the **Chart.FormatX** and **Chart.FormatY** properties. You can also customize the chart appearance by setting other properties such as **Chart.ChartType**, **Chart.Palette**, and so on.

To create a new report with an embedded chart, use the **C1Report Wizard**. Complete the following steps:

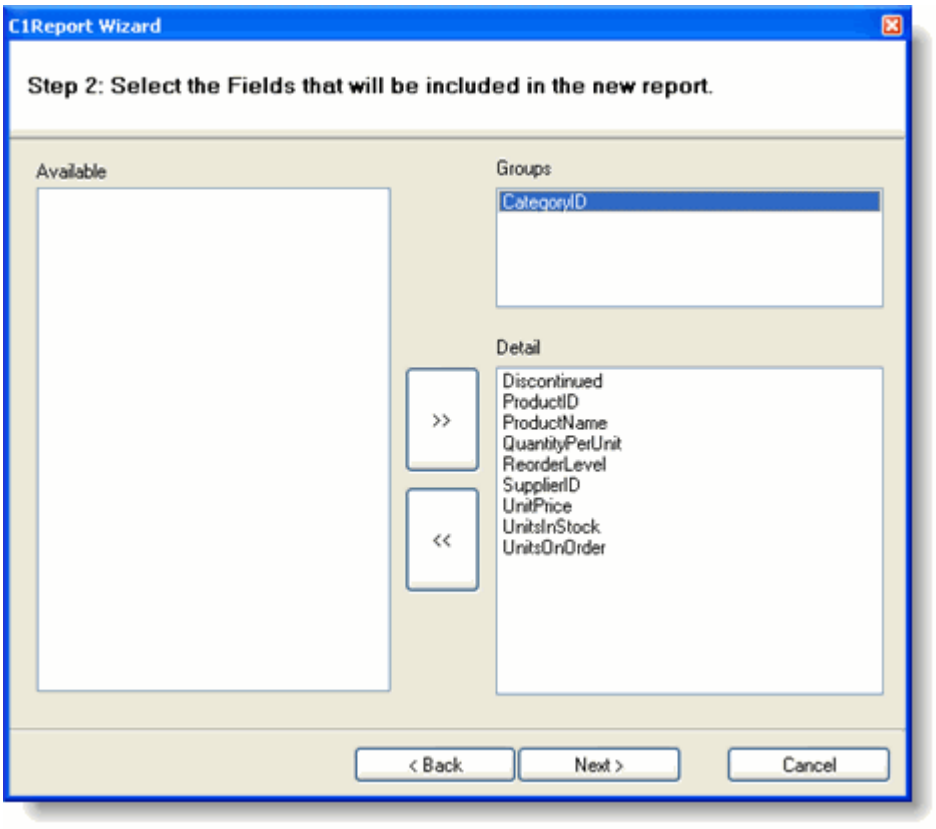
1. **Select the data source for the new report.**
    1. Click the Build a connection string ellipses button (...). The Data Link Properties dialog box appears.
    2. Select the Provider tab and then select Microsoft Jet 4.0 OLE DB Provider from the list.
    3. Click Next.
    4. In the Connection page, click the ellipsis button to browse for the Nwind.mdb database.
    5. This is the standard Visual Studio Northwind database. By default, the database is installed in the samples directory.
    6. Select the Tables radio button, and then select the Sales by Category view. The following image shows this step:



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

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

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



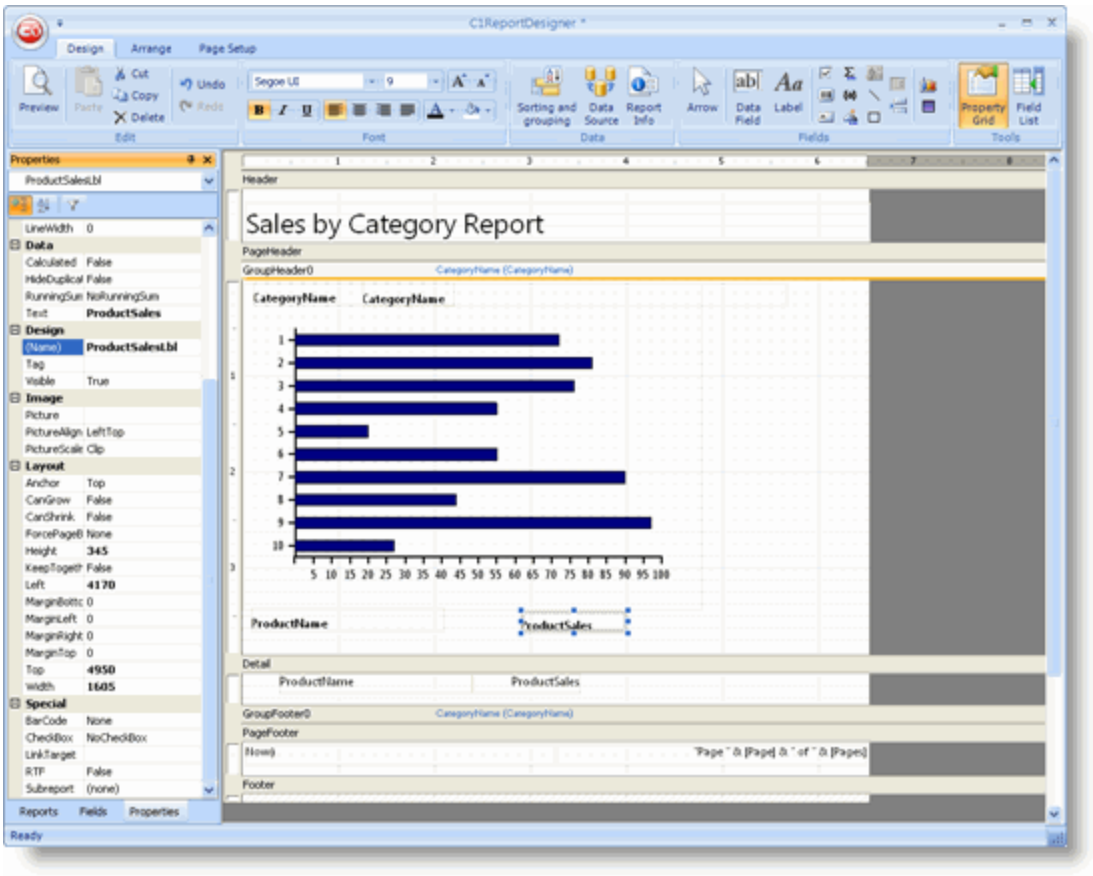
3. **Add the Chart to the Group Header section of the report.**

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

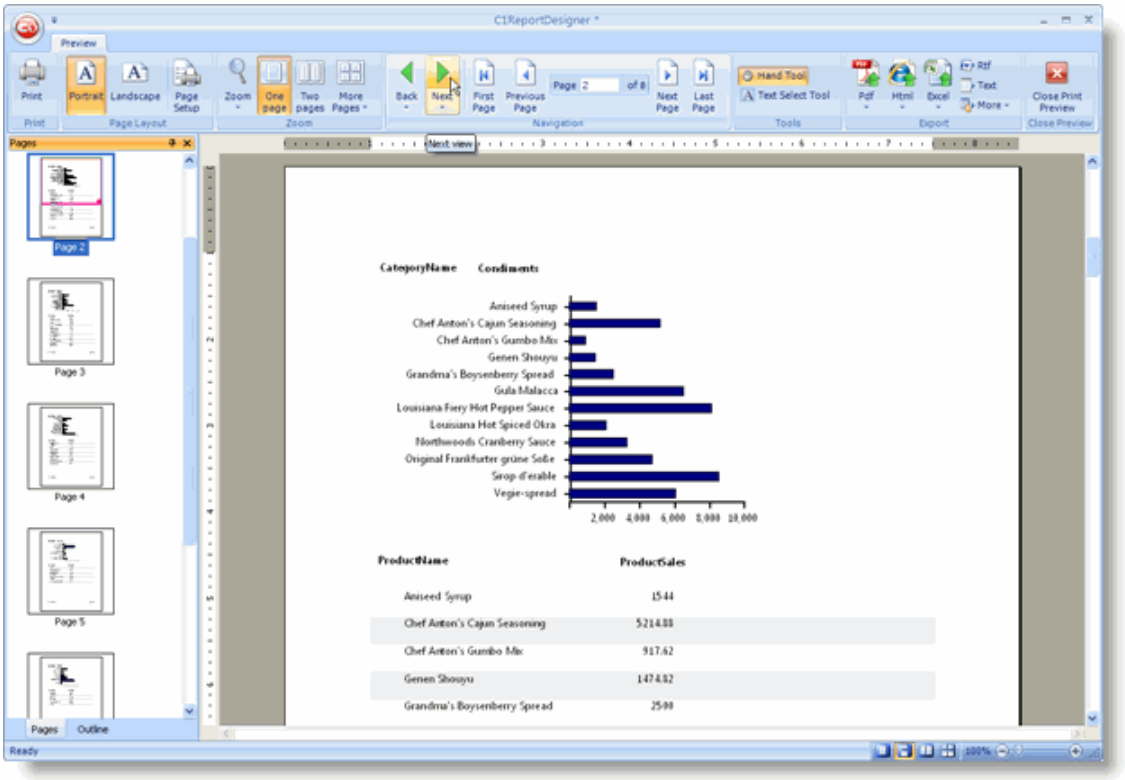
1. Click the Close Print Preview button in the Close Preview group to switch to Design mode to begin editing the report.
2. Expand the Group Header Section by performing a drag-and-drop operation with the section's borders.
3. Then click the Add Chart Field button in the Fields group of the Design tab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

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

Your report should look similar to the following report:



Click the **Preview** button to see the report and click the **Next page** button to scroll through the report to view the Chart field for each group. The sample report should look like the following image:



Note that the Report field is sensitive to its position in the report. Because it is in a Group Header section, it only includes the data within that group. If you place the Chart field in a Detail section, it will include all the data for the entire report. This is not useful because there will be one chart in each Detail section and they will all look the same. If you need more control over what data should be displayed in the chart, you can use the **DataSource** property in the chart field itself.

You can now save the report and use it in your WinForms and ASP.NET applications.

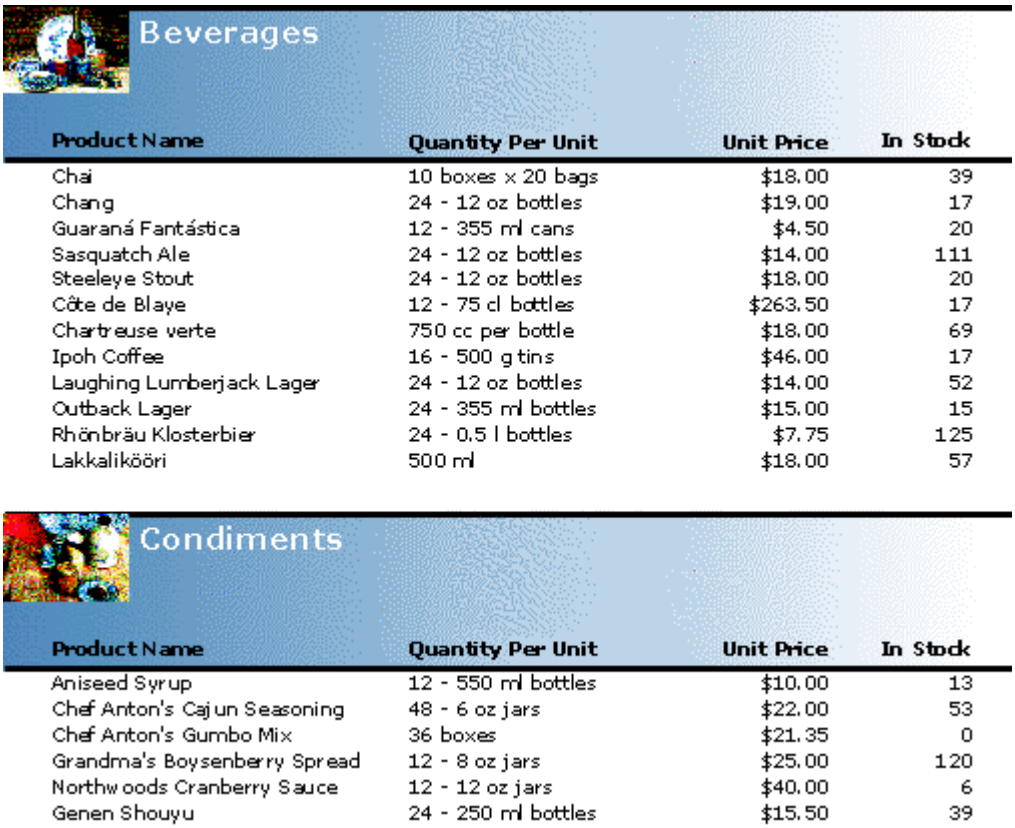
See Also

Adding Gradient Fields

Adding Gradient Fields

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

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



Beverages

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

Condiments

Product Name	Quantity Per Unit	Unit Price	In Stock
Aniseed Syrup	12 - 550 ml bottles	$10.00	13
Chef Anton's Cajun Seasoning	48 - 6 oz jars	$22.00	53
Chef Anton's Gumbo Mix	36 boxes	$21.35	0
Grandma's Boysenberry Spread	12 - 8 oz jars	$25.00	120
Northwoods Cranberry Sauce	12 - 12 oz jars	$40.00	6
Genen Shouyu	24 - 250 ml bottles	$15.50	39

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

1. In Design mode of the Designer, select the Add Gradient Field button from the Fields group in the Design tab.

2. Move your mouse cursor (which has changed to a cross-hair) over the labels in the Group Header section and drag the field to the desired size.
3. To ensure that the field is behind the labels, right-click the gradient field and select Send To Back.
4. Then set the Gradient.ColorFrom and Gradient.ColorTo properties to SteelBlue and White, respectively.

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

See Also

Selecting, Moving, and Copying Fields

Selecting, Moving, and Copying Fields

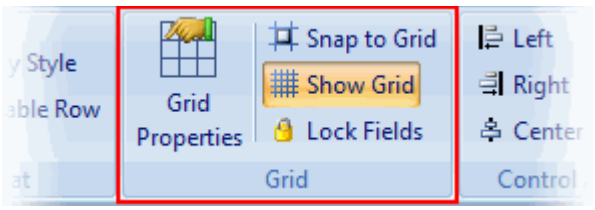
You can use the mouse to select fields in the **C1ReportDesigner** as usual:

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

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

Show a grid

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



Lock fields

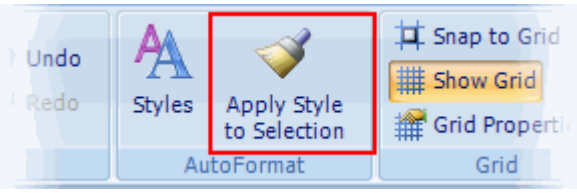
After you have placed fields in the desired positions, you can lock them to prevent inadvertently moving them with the mouse or keyboard. Use the **Lock Fields** button to lock and unlock the fields.

Format fields

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

Apply styles

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



Determine order for overlapping fields

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

Move fields using the keyboard

The **C1ReportDesigner** application also allows you to select and move fields using the keyboard:

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

See Also

Changing Field, Section, and Report Properties

Changing Field, Section, and Report Properties

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

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

To see how this works, click the label in the Header section and change its C1.C1Report.Field.Font and C1.C1Report.Field.ForeColor properties. You can also change a field's position and dimensions by typing new values for the C1.C1Report.Field.Left, C1.C1Report.Field.Top, C1.C1Report.Field.Width, and C1.C1Report.Field.Height properties.

The Properties window expresses all measurements in *twips* (the native unit used by **C1.C1Report.C1Report**), but you can type in values in other units (in, cm, mm, pix, pt) and they will be automatically converted into *twips*. For example, if you set the field's **C1.C1Report.Field.Height** property to **0.5in**, the Properties window will convert it into 720 *twips*.

See Also

Changing the Data Source

Changing the Data Source

The data source is defined by the C1.C1Report.DataSource.ConnectionString, C1.C1Report.DataSource.RecordSource, and C1.C1Report.DataSource.Filter properties. These are regular Report properties and may be set one of the following ways:

- From the Properties window, select the ellipsis button next to the DataSource property (if you click the gray area in the background, the Report properties are displayed).

OR

- Click the DataSource button in the Data group of the Design tab to open the Select a Data Source dialog box that allows you to set the C1.C1Report.DataSource.ConnectionString and C1.C1Report.DataSource.RecordSource properties directly.

See Also

Creating a Master-Detail Report Using Subreports

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

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

Beverages

Soft drinks, coffees, teas, beers, and ales

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

Condiments

Sweet and savory sauces, relishes, spreads, and seasonings

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

Page 1 of 5

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

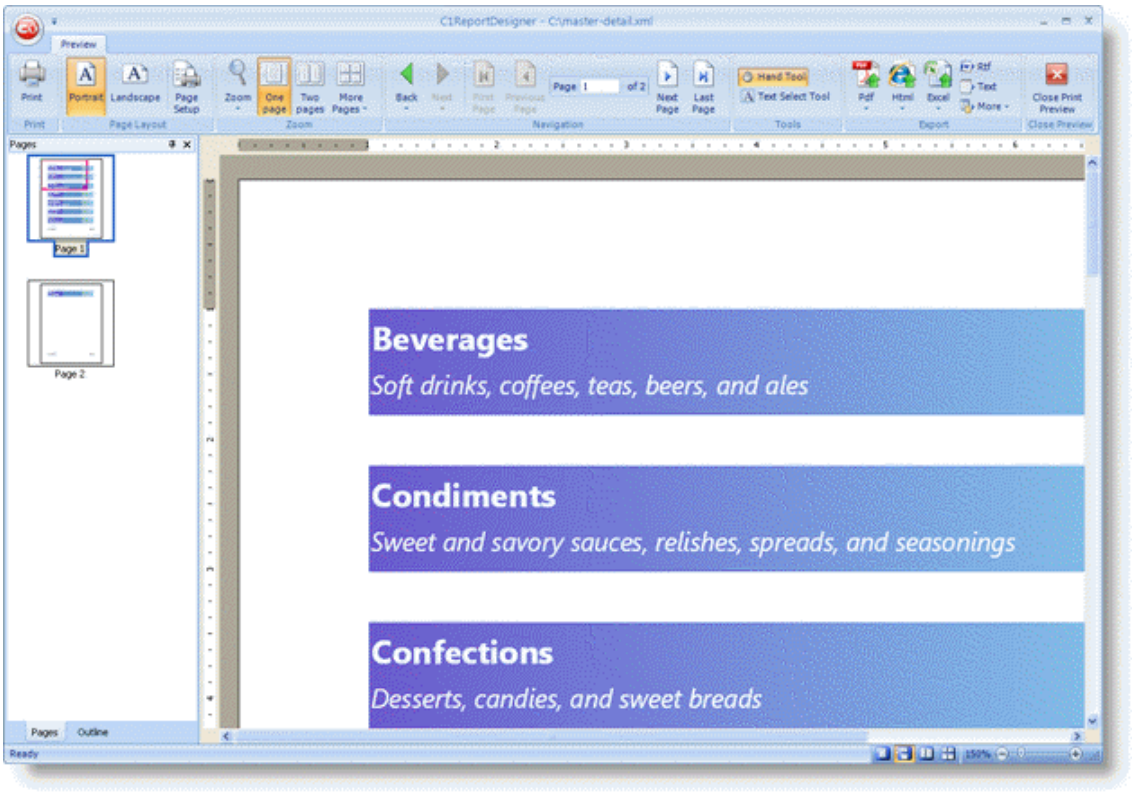
Step 1: Create the master report

Complete the following steps to create the master report:

1. Create a basic report definition using the C1Report Wizard.
    1. Select the Categories table from the Northwind database (Nwind.mdb located in the ComponentOne Samples\Common folder).
    2. Include the CategoryName and Description fields in the report.

2. In the C1ReportDesigner application, click the Close Print Preview 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 (Font and ForeColor). Note that for this example, a Gradient field was added to the Detail Section. For information on Gradient fields, see Adding Gradient Fields.

Select the **Preview** button, the Categories report should now look similar to the following image:



Step 2: Create the detail report

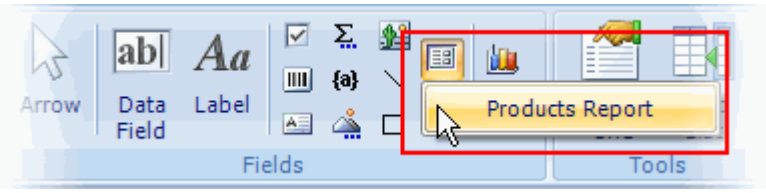
Complete the following steps to create the detail report:

1. In the C1ReportDesigner, click the New Report button to create a basic report definition using the C1Report Wizard.
    1. Select the Products table from the Northwind database.
    2. Include the following fields in the report: ProductName, QuantityPerUnit, UnitPrice, UnitsInStock, and UnitsOnOrder.
2. In the Report Designer, click the Close Print Preview 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.

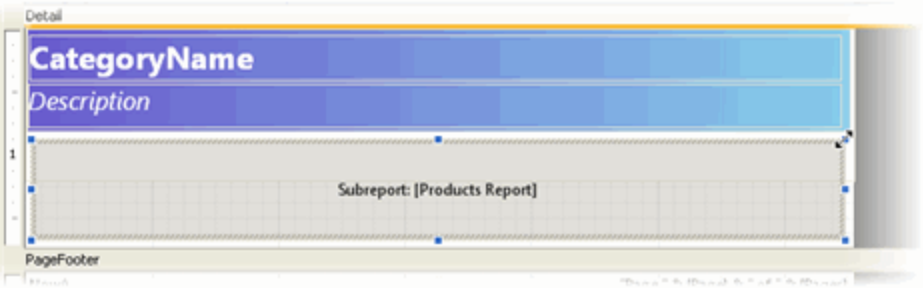
Step 3: Create the subreport field

The **C1ReportDesigner** application now has two separate reports**, Categories Report** and **Products Report**. Complete the following steps to create a subreport:

1. From the Reports list in the Designer, select Categories Report (master report).
2. In design mode, from the click the Add Subreport button in the Fields group of the Design tab 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:

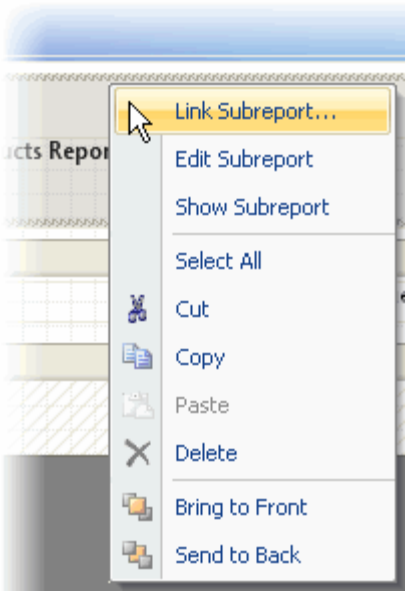


Step 4: Link the subreport to the master report

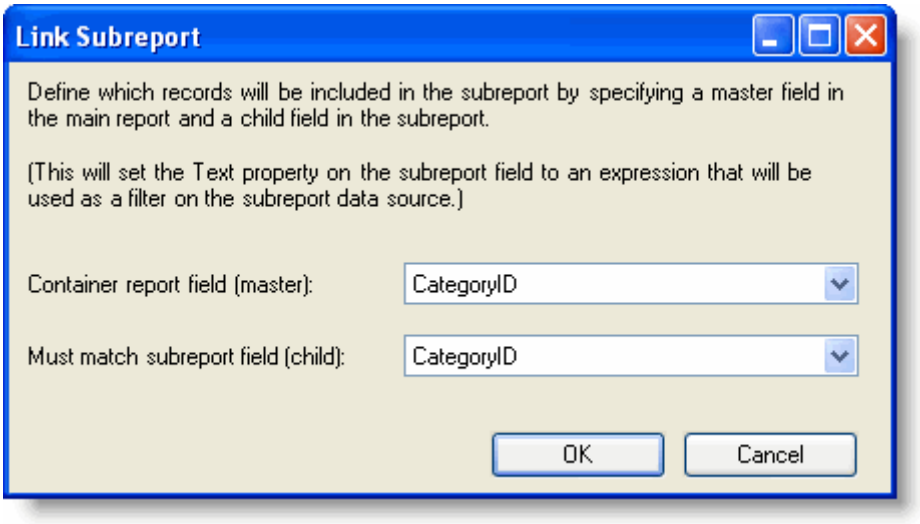
The master-detail relationship is controlled by the C1.C1Report.Field.Text property of the subreport field. This property should contain an expression that evaluates into a filter condition that can be applied to the subreport data source.

C1ReportDesigner can build this expression automatically for you. Complete the following steps:

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



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

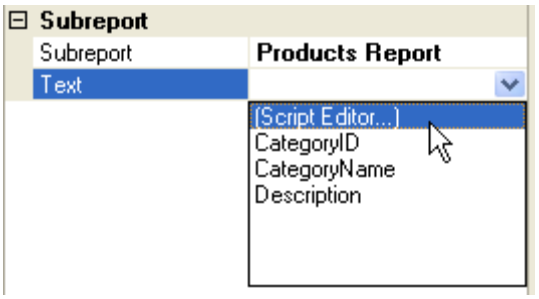


3. Once you make a selection and click OK, the Report Designer builds the link expression and assigns it to the C1.C1Report.Field.Text property of the subreport field. In this case, the expression is:

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

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

1. From the Properties window, click on the C1.C1Report.Field.Text property of the subreport field and select Script Editor from the drop-down list.



2. Enter the following expression in the VBScript Editor:

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

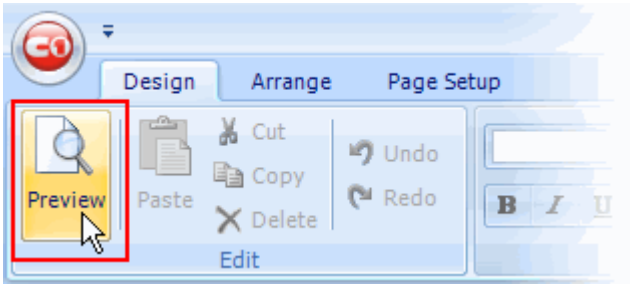
3. Click OK to close the VBScript Editor and build the expression.

See Also

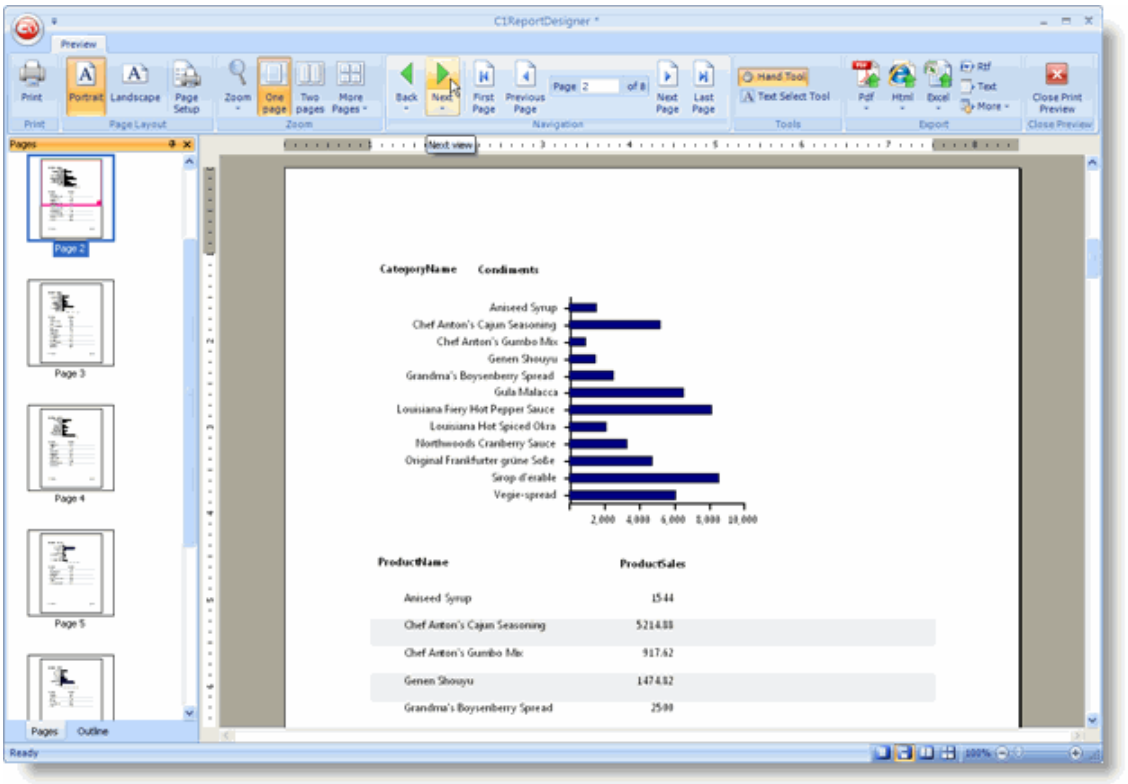
Previewing and Printing a Report

Previewing and Printing a Report

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

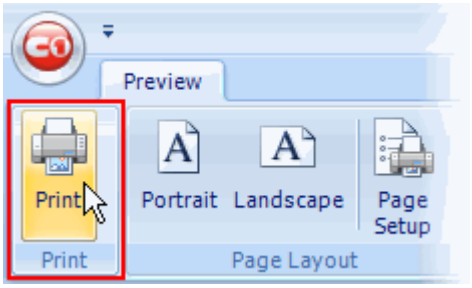


Alternatively, you can select **View | Preview** from the menu. The report is displayed on the right pane, as shown in the following screen shot:



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

At this point, you can print the report by clicking the **Print** button:



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 HMTL (*.htm)	Creates one HTML file for each page in the report. The HTML pages contain links that let the user navigate the report.
Drill-Down HTML (*.htm)	Creates a single HTML file with sections that can be collapsed and expanded by the user by clicking on them.
Plain HMTL (*.htm)	Creates a single HTML file with no drill-down functionality.
PDF with system fonts (*.pdf)	Creates a PDF file that can be viewed on any computer equipped with Adobe's Acrobat viewer or browser plug-ins.
PDF with embedded fonts	Creates a PDF file with embedded font information for extra portability.

(*.pdf)	This option significantly increases the size of the PDF file.
RTF (*.rtf)	Creates an RTF file that can be opened by most popular word processors (for example, Microsoft Word, WordPad).
RTF with fixed positioning (*.rtf)	Creates an RTF file with fixed positioning that can be opened by most popular word processors (for example, Microsoft Word, WordPad).
Microsoft Excel 97 (*.xls)	Creates an XLS file that can be opened by Microsoft Excel.
Microsoft Excel 2007/2010 Open XML (*.xlsx)	Creates an XLS file that can be opened by Microsoft Excel 2007 and later.
TIFF (*.tif)	Creates a multi-page TIFF (Tag Image File Format) file.
Text (*.txt)	Creates a plain text file.
Single Page Text (*.txt)	Creates a single-page plain text file.
Compressed Metafile (*.txt)	Creates a compressed metafile text file.

To create an export file, select **File | Export** from the menu and use the **File Save** dialog box to select the type of file you want to create, specifying its name and location.

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

See Also

Managing Report Definition Files

Managing Report Definition Files

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

To move a report from one file to another, open two instances of the **C1ReportDesigner** application and drag the report from one instance to the other. If you hold down the CTRL key while doing this, the report will be copied. Otherwise, it will be moved.

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

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

See Also

Importing Microsoft Access Reports

Importing Microsoft Access Reports

One of the most powerful features of the **C1ReportDesigner** application is the ability to import reports created with Microsoft Access. **This feature requires Access to be installed on the computer.** Once the report is imported into the Designer, Access is no longer required.

To import reports from an Access file, click the **Application** button and select **Import** from the menu. A dialog box prompts you for the name of the file you want to import.

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.

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. C1.C1Report.C1Report 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.

- **Reports that sort dates by quarter (or weekday, month of the year, and so on)**

C1.C1Report.C1Report uses the ADO.NET dataset C1.C1Report.Group.Sort property to sort groups. This property sorts datasets according to field values only and does not take expressions. (Note that you can group according to an arbitrary expression, but you can't sort.) An Access report that sorts groups by quarter will sort them by date after it is imported. To fix this, you have two options: create a field that contains the value for the expression you want to sort on or change the SQL statement that creates the dataset and perform the sorting that way.

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 Nwind.mdb File

To illustrate how the Designer fares in a real-life example, try importing the Nwind.mdb file. It contains the following 13 reports. (The Nwind.xml file that ships with C1.C1Report.C1Report 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 C1.C1Report.Section.OnPrint property:

Visual Basic	Copy Code

```
If SalespersonTotal > 5000 Then
  ExceededGoalLabel.Visible = True
  SalespersonLine.Visible = True
Else
  ExceededGoalLabel.Visible = False
  SalespersonLine.Visible = False
End If
```

C#	Copy Code

```
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 could use an unbound picture field, then write a VB event handler that would create the chart and assign it to the field as a picture.

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, edit the C1.C1Report.DataSource.RecordSource property to add a [Show Details] parameter:

Visual Basic	Copy Code

```
PARAMETERS (Beginning Date) DateTime 1/1/1994,
  (Ending Date) DateTime 1/1/2001,
  (Show Details) Boolean False; ...
```

C#	Copy Code

```
PARAMETERS [Beginning Date] DateTime 1/1/1994,
  [Ending Date] DateTime 1/1/2001,
  [Show Details] Boolean False; ...
```

Use the new parameter in the report's **OnOpen** event:

Visual Basic	Copy Code

```
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"
c1r.Sections.Detail.OnPrint = script
```

C#	Copy Code

```
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";
c1r.Sections.Detail.OnPrint = script;
```

Finally, two more lines of code need to be translated:

Visual Basic	Copy Code

```
Sections ("Detail").OnPrint = _
  "PageHeader.Visible = True"
Sections("Group 0 Footer).OnPrint = _
  "PageHeader.Visible = False"
```

C#	Copy Code

```
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 C1.C1Report.Section.OnPrint property:

Visual Basic	Copy Code
`PageTotal = 0`	

C#	Copy Code
`PageTotal = 0;`	

The following code should be assigned to the Detail C1.C1Report.Section.OnPrint property:

Visual Basic	Copy Code
`PageTotal = PageTotal + SaleAmount` `HiddenPageBreak.Visible = (Counter = 10)`	

C#	Copy Code
`PageTotal = PageTotal + SaleAmount;` `HiddenPageBreak.Visible = (Counter = 10);`	

12. **Summary of Sales by Quarter**

This report has a group that is sorted by quarter (see item 4 above). To fix this, add a field to the source dataset that contains the value of the **ShippedDate** quarter, by changing the C1.C1Report.DataSource.RecordSource property as follows:

```
SELECT DISTINCTROW Orders.ShippedDate,
  Orders.OrderID,
  [Order Subtotals].Subtotal,
  DatePart("q",Orders.ShippedDate) As ShippedQuarter
```

FROM Orders INNER JOIN [Order Subtotals]

ON Orders.OrderID = [Order Subtotals].OrderID

WHERE (((Orders.ShippedDate) Is Not Null));

Change the group's C1.C1Report.Group.GroupBy property to use the new field, **ShippedQuarter**.

13. **Summary of Sales by Year**

No action required.

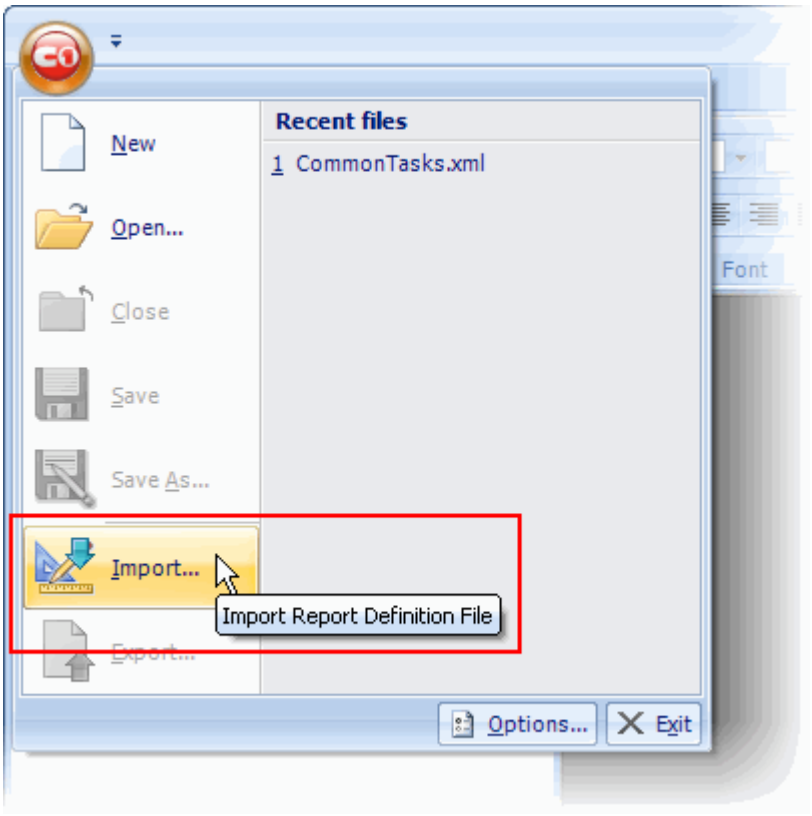
Summing up the information on the table, out of the 13 reports imported from the NorthWind database: eight did not require any editing, three required some code translation, one required changes to the SQL statement, and one had a chart control that was not replaced.

See Also

Importing Crystal Reports

Importing Crystal Reports

The **C1ReportDesigner** application can also import Crystal report definition files (.rpt files).



To import reports from a Crystal report definition file:

1. Click the Application button and select Import from the menu.

A dialog box prompts you for the name of the file you want to import.

2. Select a Crystal report definition file (RPT) and the Designer will convert the report into the C1.C1Report.C1Report format.

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 C1.C1Report.C1Report. The exceptions include image fields, charts, and cross-tab fields.

See Also

Task Based Help

Task Based Help

The task-based help assumes that you are familiar with programming in Visual Studio, and know how to use the **Reports for WPF** product in general. If you are unfamiliar with the **Reports for WPF** product, please see Reports for WPF Quick Starts first.

Each topic in this section provides a solution for specific tasks using the Reports for WPF product. Each task-based help topic also assumes that you have created a new WPF project.

Note: For information about using C1Report and C1PrintDocument, see the Reports for WinForms documentation.

See Also

Add image field

Creating Field Reports

Customize page header

Customize Page Layout

Formatting Reports

Modifying Subreport

Saving a Report Definition

Loading a Report into the C1DocumentViewer Control

Add image field

Task Based Help > Add image field

Using the **C1ReportDesigner** application, you can add unbound or bound images and create watermarks.

See Also

Creating unbound image

Creating bound image

creating a report that contains a watermark

Creating unbound image

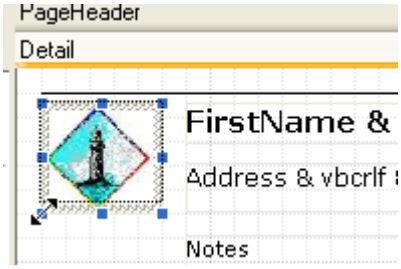
Unbound images are static images such as logos and watermarks that are not stored in the database. To add unbound image fields to your report, complete the following tasks:

1. Open the C1ReportDesigner application. For more information on how to access C1ReportDesigner, see Accessing C1ReportDesigner from Visual Studio.
2. Create a new report or open an existing report. Once you have the report in the C1ReportDesigner, you can modify the report properties.
3. Click the Design button to begin editing the report.
4. In Design mode, click the Add Unbound Picture button located in the Fields group of the Design tab.

The **Open** dialog box appears.

5. Select the image file you want to include in the report, and click Open.
6. Click on your report where you would like to place the image, and then resize the field to show the image.

In the following image an unbound image has been added to the report and is being resized:



Note that the image file can be embedded in the report definition, or it can be a reference to an external file. To choose the option you prefer, in the Designer select the **Application** button and in the menu that appears select **Options**. The **C1ReportDesigner Options** dialog box appears where you can choose the **Embed images into Xml when saving** option.

Sample Report Available

For the complete report, see report "03: Unbound Images" in the CommonTasks.xml report

definition file, which is available for download from the CommonTasks sample on the ComponentOne HelpCentral Sample page.

See Also

Creating bound image

Creating bound image

Bound images are images stored in database fields. To display these images in your reports, add a field to the report and set its **Picture** property to a string containing the name of the column where the image is stored.

Using the C1ReportDesigner

To add bound image fields to your report using the **C1ReportDesigner**, complete the following:

1. In Design mode of the C1ReportDesigner, click the Add Bound Picture button located in the Fields group of the Design tab.

This shows a menu with all binary fields in the current data source.

2. Select the field you want to add to the report.

In Code

If the field "Photo" in the database contains embedded OLE objects or raw image streams, and the report contains a field called "fEmployeePhoto", then the following code would display the employee photo in the field:

Visual Basic	Copy Code

```
fEmployeePhoto.Picture = "Photo"
```

C#	Copy Code

```
fEmployeePhoto.Picture = "Photo";
```

Sample Report Available

For the complete report, see report "04: Bound Images" in the CommonTasks.xml report

definition file, which is available for download from the CommonTasks sample on the ComponentOne HelpCentral Sample page.

See Also

creating a report that contains a watermark

Creating a Watermark

Watermarks are images displayed behind the report content. The images are often washed out to prevent them from interfering with the actual report content.

To display an image as a watermark, set the C1.C1Report.Layout.Picture property to a file that contains the image. You can also control the way the watermark is scaled and the pages on which it should appear using the C1.C1Report.Layout.PictureAlign and C1.C1Report.Layout.PictureShow properties.

Sample Report Available:

For the complete report, see report "05: Watermark" in the CommonTasks.xml report definition file, which is available for download from the CommonTasks sample on the ComponentOne HelpCentral Sample page.

See Also

Creating Field Reports

Working with Report Fields

Loading Custom fields

Reports for WPF allow users to load reports that contain custom fields by using the **CustomFields** assembly. Reports created by the **C1ReportDesigner** application offers following custom fields:

- Chart
- SuperLabel
- Gradient
- QRCode
- Map

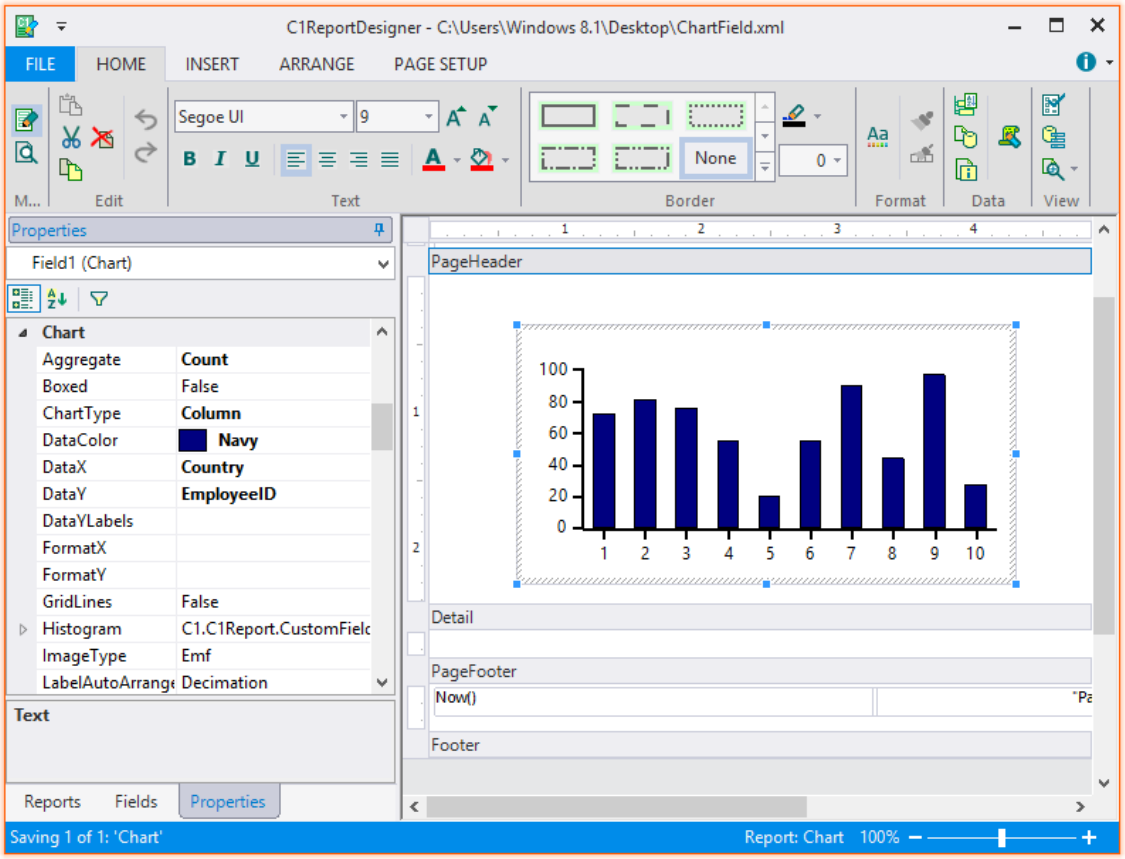
Note that some custom fields, such as those listed below, need additional ComponentOne assemblies to be referenced:

- Chart: C1.Win.C1Chart
- SuperLabel: C1.Win.C1SuperTooltip
- QRCode: C1.Win, C1.Win.BarCode
- Map: C1.WPF.Maps

The following sample creates the report definition with Chart custom field, loads, and renders it to **C1DocumentViewer** in WPF application; the steps are illustrated as follows:

1. Create the report definition:
    1. Open the **C1ReportDesigner** application. For more information on how to access the C1ReportDesigner, see Accessing C1ReportDesigner from Visual Studio.
    2. Create a new report by using **C1Report Wizard**. Set the Employees table in **C1NWind.mdb** as the data source, select the fields, and set the report name as "Chart".
    3. Click the **Design** button to begin editing the report.
    4. In the **Custom Fields** group of the **Insert** tab, click the **Chart** button.
    5. Place the **Chart** field in your report and resize the field to show the chart.
    6. From the Properties window, set the properties of the **Chart** field as following:
        - DataX: Country
        - DataY: EmployeeID
        - Aggregate: Count
        - ChartType: Column
        - YMin: 0
    7. Preview the report by pressing **F5**.

8. Save the report definition file with the file name "ChartField.xml". The design view of the report definition is as shown:



2. Create the WPF application:
    1. Create a new WPF application. In order to access **C1NWind.mdb**, set the project's platform target to **x86**.
    2. Add following assemblies to the **References** in the project:
        - C1.WPF.C1Report.CustomFields
        - C1.WPF.C1Report
        - C1.WPF C1.Win.C1Chart
    3. Add the report definition file **ChartField.xml** into the project folder. Set its 'Copy to Output Directory' property to 'Copy if newer'.
    4. Add **C1DocumentViewer** control into the MainWindow.
    5. In the code view, add the following code:

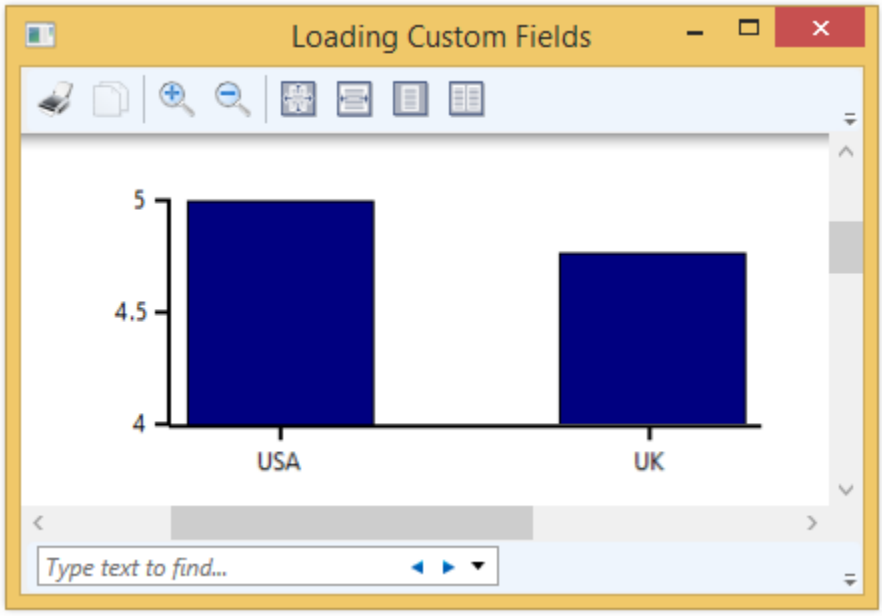
C#	Copy Code

```
public MainWindow()
        {
            InitializeComponent();
            var report = new C1.C1Report.C1Report();
            report.Load("ChartField.xml", "Chart");
            this.c1DocumentViewer.Document =
report.C1Document.FixedDocumentSequence;
        }
```

VB	Copy Code

```
Public Sub New()
    InitializeComponent()
    Dim report = New C1.C1Report.C1Report()
    report.Load("ChartField.xml", "Chart")
    Me.c1DocumentViewer.Document = report.C1Document.FixedDocumentSequence
End Sub
```

3. Run the application. You see the report with custom field rendered in the **C1DocumentViewer** as shown:



For more information on creating chart custom field, see the Adding Chart Fields topic. The report definition CustomFields.xml available in the default location C:\Users\Windows 8.1\Documents\ComponentOne Samples\Studio for WPF\C1.WPF.C1Report\C1WPFReport\XML\CommonTasks\ contains all the custom fields.

From 2015v2 onwards, for WPF applications referencing C1.WPF.C1Report.CustomFields.4 dll, please add C1.Win.4 and C1.Win.Barcode.4 dlls to your projects.

See Also

Adding Chart Fields

Creating Charts

In the initial versions of C1Report, adding charts to reports required handling the **StartSection** event, generating the chart, and assigning the chart image to a field's Picture property. This is not hard to do, and continues to be the most flexible way to add dynamic images to reports.

However, this approach has two drawbacks:

- It requires you to write code outside the report definition, which means only your application will be capable of showing the report the way it is meant to be shown.
- It requires you to write code for generating the chart, which can be tedious.

The current C1Report supports custom report fields, including a chart field that is based on the **C1Chart** control.

To add a chart field to a Group Header section in your report, complete the following steps:

1. Open the **C1ReportDesigner**. For more information on how to access the **C1ReportDesigner**, see Accessing C1ReportDesigner from Visual Studio.
2. Create a new report or open an existing report. Once you have the report in the **C1ReportDesigner**, you can modify the report properties.
3. Click the **Design** button to begin editing the report.
4. In the **Custom Fields** group of the **Insert** tab, click the **Chart Field** button.
5. Click in the Group Header section of your report and drag the field to resize the chart.
6. From the Properties window, set the chart field's **Chart.DataX** and **Chart.DataY** properties to the values you want to display on the chart. You can show several series by setting the **Chart.DataY** property to a list of fields delimited by semicolons (for example, "UnitsInStock;ReorderLevel").

The chart data is automatically scoped to the current report group. For example, when rendering the "Beverages" section, only data for that category will be charted. You can customize the chart using many other properties such as **Chart.ChartType**, **Chart.GridLines**, **Chart.Use3D**, and **Chart.Palette** properties.

For more information on creating chart fields, see the Adding Chart Fields topic.

Note: For the complete report, see report "11: Charts" in the **CommonTasks.xml** report definition file, which is available in the **ComponentOne Samples** folder.

Customize Page Header

Adding a Continued lables/Character string on page headers

Group Header sections are repeated across page breaks if their C1.C1Report.Section.Repeat property is set to **True**. This makes the report easier to read, but it can be hard to tell if a group header on a page marks the beginning of a group or is just a continuation.

One way to address this is to add a field with a "Continued" label named, **fContinued**, for example to the group header and control its visibility with script. To do this, complete the following steps:

1. Open the C1ReportDesigner application. For more information on how to access C1ReportDesigner, see Accessing C1ReportDesigner from Visual Studio.
2. Create a new report or open an existing report. Once you have the report in the C1ReportDesigner application, you can modify the report properties.
3. Click the Close Print Preview button to begin editing the report.
4. In Design mode, select Detail from the drop-down list above the Properties window.
5. Locate the Detail.C1.C1Report.Section.OnPrint property and click the empty field next to it, and then click the ellipsis button.
6. The VBScript Editor appears. Enter the following VBScript expression in the code editor:

```
' VBScript: Detail.OnPrint

fContinued.Visible = true
```

7. Then select GroupFooter from the drop-down list above the Properties window.
8. Locate the GroupFooter.C1.C1Report.Section.OnPrint property and click the empty field next to it, and then click the ellipsis button.
9. The VBScript Editor appears. Enter the following VBScript expression in the code editor:

```
' VBScript: GroupFooter.OnPrint

fContinued.Visible = false
```

If the **fContinued** field is initially invisible, then the script will show the label only on continued page headers. This script ensures that the **fContinued** field is visible within the group. Any page breaks created after the group footer and before the next Detail section will not show the label.

Sample Report Available:

For the complete report, see report "18: Continued Headers" in the CommonTasks.xml report definition file, which is available for download from the CommonTasks sample on the ComponentOne HelpCentral Sample page.

See Also

Dynamically changing of Page Header

Changing Page Headers Dynamically

To specify whether Page Header and Page Footer sections should appear on all pages, or be suppressed on the pages that contain the report Header and report Footer sections use **C1.C1Report.C1Report**'s C1.C1Report.Layout.PageHeader and C1.C1Report.Layout.PageFooter properties.

Sometimes you may want to further customize this behavior. For example, you may want to render different headers on odd and even pages. This can be done with script that shows or hides fields depending on the page being rendered. To do this, complete the following steps:

1. Open the C1ReportDesigner application. For more information on how to access C1ReportDesigner, see Accessing C1ReportDesigner from Visual Studio.
2. Create a new report or open an existing report. Once you have the report in the C1ReportDesigner application, you can modify the report properties.
3. Click the Close Print Preview button to begin editing the report.
4. In Design mode, select Detail from the drop-down list above the Properties window.
5. Locate the C1.C1Report.Section.OnFormat property and click the empty field next to it, and then click the ellipsis button.
6. The VBScript Editor appears. Enter the following VBScript expression in the code editor:

```
odd = (page mod 2 <> 0)

h1odd.Visible = odd

h2odd.Visible = odd

h1even.Visible = not odd

h2even.Visible = not odd
```

This script will show or hide fields for odd and even pages if a report header contains several fields named "h<x>odd" and "h<x>even".

Note that to prevent the page header from showing blank spaces, all the fields should have the C1.C1Report.Field.CanShrink property set to **True**.

Sample Report Available:

For the complete report, see report "09: Dynamic Page Header" in the CommonTasks.xml report definition file, which is available for download from the CommonTasks sample on the ComponentOne HelpCentral Sample page.

See Also

Customize Page Layout

Customize Page Layout

The following topics explain how you can customize the layout of your report.

See Also

Control Page break

Automatic adjustment of Field Size

Creating CanGrow/CanShrink Fields

Creating a Gutter Margin

Specifying Custom Paper Size

Defining and Using Global Constants

Control Page break

By default, C1.C1Report.C1Report fills each page until the bottom, inserts a page break, and continues rendering in the next page. You can override this behavior using several properties:

- Group.C1.C1Report.Group.KeepTogether: Determines whether Group Header sections are allowed to render on a page by themselves, if they must be rendered with at least one Detail section, or if the entire group should be kept together on a page.
- Section.C1.C1Report.Section.KeepTogether: Determines whether page breaks are allowed within sections. It has lower precedence than Group.C1.C1Report.Group.KeepTogether.
- C1.C1Report.Section.ForcePageBreak: Allows you to specify that page breaks should be inserted before, after, or before and after the section.
- Field.C1.C1Report.Field.KeepTogether: Determines whether page breaks are allowed within fields. This allows long Text fields to span multiple pages. It has lower precedence than Section.C1.C1Report.Section.KeepTogether.
- C1.C1Report.Field.ForcePageBreak: Allows you to specify that page breaks should be inserted before, after, or before and after the field.

You can set these properties through the Property grid in the **C1ReportDesigner**.

You can use script to change the properties while the report is being rendered. For example, to cause page breaks after each 10 Detail sections, complete the following steps:

1. Open the C1ReportDesigner application. For more information on how to access C1ReportDesigner, see Accessing C1ReportDesigner from Visual Studio.
2. Create a new report or open an existing report. Once you have the report in the C1ReportDesigner application, you can modify the report properties.
3. Click the Close Print Preview button to begin editing the report.
4. In Design mode, select Detail from the drop-down list above the Properties window.
5. Locate the C1.C1Report.Section.OnPrint property and click the empty field next to it, and then click the ellipsis button.
6. The VBScript Editor appears. Enter the following VBScript expression in the code editor:

```
cnt = cnt + 1

detail.forcepagebreak = "none"

if cnt >= 10 then

  cnt = 0

  detail.forcepagebreak = "after"

endif
```

See Also

Automatic adjustment of Field Size

Automatic adjustment of Field Size

For the complete report, see report "07: Force Page Breaks" in the CommonTasks.xml report definition file, which is available for download from the CommonTasks sample on the ComponentOne HelpCentral Sample page.

See Also

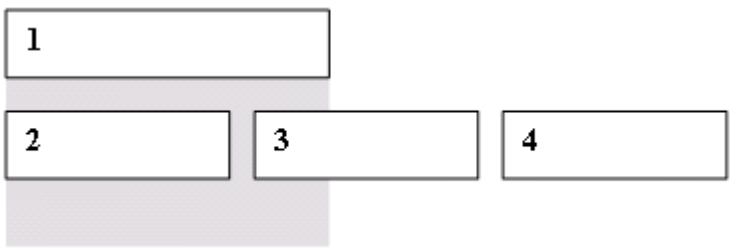
Creating CanGrow/CanShrink Fields

Creating CanGrow/CanShrink Fields

It is common for report fields to have content that may span multiple lines or collapse to no lines at all. In some cases, you may want to allow these fields to grow or shrink to fit their content rather than clip the excess or leave white spaces in the report.

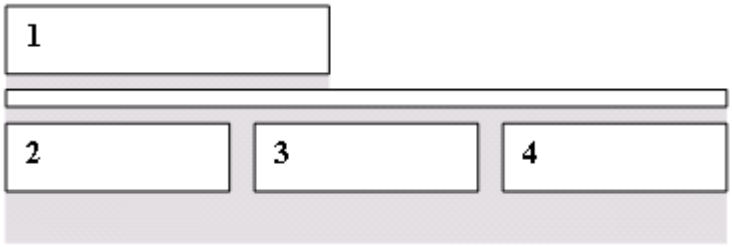
To do this, in Design mode of the **C1ReportDesigner**, set the **Field** object's C1.C1Report.Field.CanGrow and C1.C1Report.Field.CanShrink properties to **True**.

Fields that grow push down the fields below them. Likewise, fields that can shrink push up the fields below them. Below in this case means "strictly" below, as shown in the following diagram:



Field 1 will push or pull fields 2 and 3 when it grows or shrinks. Field 4 will not be affected because it is not directly below field 1. The shaded area in the diagram shows the region affected by field 1.

If you want field 4 to remain aligned with fields 2 and 3, add an extra field spanning the whole area above fields 2 and 3. The new field will be pushed down by field 1 and will in turn push fields 2, 3, and 4. The following diagram shows this new layout:



Sample Report Available:

For the complete report, see report "06: CanGrow CanShrink" in the CommonTasks.xml report definition file, which is available for download from the CommonTasks sample on the ComponentOne HelpCentral Sample page.

See Also

Creating a Gutter Margin

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 C1.C1Report.Layout.MarginLeft property on odd pages and use the default value on even pages. This can be done with script. To add script that changes the margins based on the page being rendered, complete the following steps:

Open the **C1ReportDesigner** application. For more information on how to access **C1ReportDesigner**, see Accessing C1ReportDesigner from Visual Studio.

1. Create a new report or open an existing report. Once you have the report in the C1ReportDesigner application, you can modify the report properties.
2. Click the Close Print Preview button to begin editing the report.
3. In Design mode, select Detail from the drop-down list above the Properties window.
4. Locate the C1.C1Report.Section.OnPrint property and click the empty field next to it, and then click the ellipsis button.
5. The VBScript Editor appears. Enter the following VBScript expression in the code editor:

```
' VBScript: Report.OnOpen

gutter = report.layout.marginleft ' initialize variable

' VBScript: Report.OnPage

report.layout.marginleft = _

  Iif(page mod 2 = 1, gutter, gutter - 1440)
```

Note: For the complete report, see report "10: Gutter" in the CommonTasks.xml report definition file, which is available for download from the CommonTasks sample on the ComponentOne HelpCentral Sample page.

See Also

Specifying Custom Paper Size

Specifying Custom Paper Size

By default, C1.C1Report.C1Report creates reports using the default paper size on the default printer.

You can specify the paper size and orientation using the C1.C1Report.Layout.PaperSize and C1.C1Report.Layout.Orientation properties. However, C1.C1Report.C1Report 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 C1.C1Report.Layout.PaperSize property to **Custom**, and set the C1.C1Report.Layout.CustomWidth and C1.C1Report.Layout.CustomHeight properties to the page dimensions (in *twips*).

Using the C1ReportDesigner

Complete the following steps to specify a custom paper size of 25" x 11" for your report using the **C1ReportDesigner** application:

1. Open the C1ReportDesigner. For more information on how to access the C1ReportDesigner, see Accessing C1ReportDesigner from Visual Studio.
2. Create a new report or open an existing report. Once you have the report in the C1ReportDesigner, you can modify the report properties.
3. Click the Close Print Preview button to begin editing the report.
4. In Design mode, select your report from the drop-down list above the Properties window.
5. Locate Layout and expand the property node to view all available properties.
6. Set the Custom Height property to 25" or 25in.

Notice that the measurement is converted into *twips* automatically. The Properties window displays the measurement in *twips* (36000).

7. Set the Custom Width property to 11" or 11in.

The Properties window displays the measurement in *twips* (15840).

8. Set the PaperSize property to Custom.

In Code

Regardless of what is available on the printer, the following code sets the report paper to 25" x 11":

Visual Basic	Copy Code

```
c1r.Layout.PaperSize = PaperKind.Custom
c1r.Layout.CustomHeight = 25 * 1440 ' in twips
c1r.Layout.CustomWidth = 11 * 1440
```

C#	Copy Code

```
c1r.Layout.PaperSize = PaperKind.Custom;
c1r.Layout.CustomHeight = 25 * 1440; // in twips
c1r.Layout.CustomWidth = 11 * 1440;
```

See Also

Defining and Using Global Constants

Defining and Using Global Constants

There is no special mechanism for defining and using global constants in a report, but you can add hidden fields to the report and use their values as global parameters. To do this, complete the following steps:

1. Open the C1ReportDesigner application. For more information on how to access C1ReportDesigner, see Accessing C1ReportDesigner from Visual Studio.
2. Create a new report or open an existing report. Once you have the report in the C1ReportDesigner application, you can modify the report properties.
3. Click the Close Print Preview button to begin editing the report.

4. In the Fields group of the Design tab, click the Add Label button to add a field to your report.
5. Click on your report where you want the field placed and drag to resize the field.
6. Set the following properties for the field:

Property	Setting
Field.Name	linesPerPage
Field.Text	14
Field.Visible	False

7. To control the number of detail lines per page, use script. Select Detail from the drop-down list above the Properties window.
8. Locate the C1.C1Report.Section.OnPrint property and click the empty field next to it, and then click the ellipsis button.

The **VBScript Editor** appears.

9. Enter the following VBScript expression in the code editor:

```
cnt = cnt + 1

detail.forcepagebreak = "none"

if cnt >= linesPerPage then

  cnt = 0

  detail.forcepagebreak = "after"

endif
```

Note that the value in the **linesPerPage** field can be set prior to rendering the report, by changing the field's **Text** property.

Sample Report Available:

For the complete report, see report "08: Global Constant" in the CommonTasks.xml report definition file, which is available for download from the CommonTasks sample on the ComponentOne HelpCentral Sample page.

See Also

Formatting Reports

Formatting Reports

The following topics show how to apply formatting to your report. By simply modifying properties in the Properties window or adding a few lines of script to your VBScript expression, you can visually enhance your report.

See Also

Adding Alternating Background Color

Conditional Changing of Blank Form

Editing the Field's Format Based on Value

Suppressing or Forcing the Display of Zeros

Avoid repetition of value

Adding Alternating Background Color

To create a report with alternating background color, use the C1.C1Report.Section.OnPrint property of the Detail section to change the C1.C1Report.Section.BackColor property of the section.

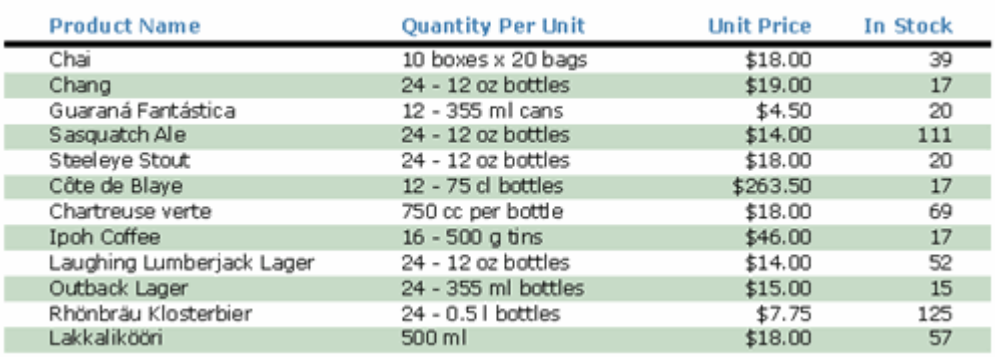
To do this, complete the following steps:

1. Open the C1ReportDesigner application. For more information on how to access C1ReportDesigner, see Accessing C1ReportDesigner from Visual Studio.
2. Create a new report or open an existing report. Once you have the report in the C1ReportDesigner application, you can modify the report properties.
3. Click the Close Print Preview button to begin editing the report.
4. In Design mode, select the report from the drop-down list above the Properties window.
5. Locate the C1.C1Report.C1Report.OnOpen property and enter cnt = 0. This initializes the cnt variable.
6. Next, select Detail from the drop-down list above the Properties window.
7. Locate the C1.C1Report.Section.OnPrint property and click the empty field next to it, and then click the ellipsis button.
8. The VBScrpit Editor appears. Enter the following VBScript expression in the code editor:

```
cnt = cnt + 1

if cnt mod 2 = 0 then

  detail.backcolor = rgb(200,220,200)

else

  detail.backcolor = rgb(255,255,255)

endif
```

9. Click the Preview button to preview the report with alternating background.

This topic illustrates the following:



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

The report will appear with an alternating background color.

Whenever a Detail section is rendered, the counter is incremented and the **BackColor** property of the Detail section is toggled.

Sample Report Available:

For the complete report, see report "01: Alternating Background (Greenbar report)" in the CommonTasks.xml report definition file, which is available for download from the CommonTasks sample on the ComponentOne HelpCentral Sample page.

See Also

Conditional Changing of Blank Form

Conditional Changing of Blank Form

In some cases you may want to change a field's appearance depending on the data it represents. For example, you may want to highlight items that are expensive, or low in stock. This can be done with script.

To do this, complete the following steps:

1. Open the C1ReportDesigner application. For more information on how to access C1ReportDesigner, see Accessing C1ReportDesigner from Visual Studio.
2. Create a new report or open an existing report. Once you have the report in the C1ReportDesigner application, you can modify the report properties.
3. Click the Close Print Preview button to begin editing the report.
4. 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).
5. Locate the C1.C1Report.Section.OnFormat property and click the empty field next to it, and then click the ellipsis button.

6. The VBScript Editor appears. Enter the following VBScript expression in the code editor:

```
' VBScript: Detail.OnFormat

If UnitsInStock + UnitsOnOrder < ReorderLevel And _

  Discontinued = False Then

 Detail.BackColor = rgb(255,190,190)

Else

 Detail.BackColor = vbWhite

Endif
```

The script changes the Detail section's C1.C1Report.Section.BackColor property depending on the value of the fields **UnitsInStock**, **UnitsOnOrder**, **ReorderLevel**, and **Discontinued**.

Sample Report Available:

For the complete report, see report "16: Conditional Formatting" in the CommonTasks.xml report definition file, which is available for download from the CommonTasks sample on the ComponentOne HelpCentral Sample page.

See Also

Editing the Field's Format Based on Value

Editing the Field's Format Based on Value

You can change a report field's format based on its value by specifying an expression for the Detail section's C1.C1Report.Section.OnFormat property.

To specify an expression for the C1.C1Report.Section.OnFormat property, complete the following steps:

1. Open the C1ReportDesigner application. For more information on how to access C1ReportDesigner, see Accessing C1ReportDesigner from Visual Studio.
2. Create a new report or open an existing report. Once you have the report in the C1ReportDesigner application, you can modify the report properties.
3. Click the Close Print Preview button to begin editing the report.
4. In Design mode, select Detail from the Properties window's drop-down list to view the available properties for the Detail section.
5. Locate the C1.C1Report.Section.OnFormat property and click the ellipsis button next to the property.
6. The VBScript Editor appears where you can specify an expression.

The following expression changes the *UnitsInStock* field's **ForeColor** to red if the sum of the UnitsInStock value and the UnitsOnOrder value is less than the ReorderLevel value. There are several ways to write this expression.

Option 1:

```
UnitsInStockCtl.Forecolor = Iif(UnitsInStock + UnitsOnOrder < ReorderLevel, vbRed, vbBlack)
```

Option 2:

```
lowStock = UnitsInStock + UnitsOnOrder < ReorderLevel

UnitsInStockCtl.Forecolor = Iif(lowStock, vbRed, vbBlack)
```

Option 3:

```
If UnitsInStock + UnitsOnOrder < ReorderLevel Then
   UnitsInStockCtl.Forecolor = vbRed
Else
   UnitsInStockCtl.Forecolor = vbBlack
End If
```

Option 4:

```
color = Iif(UnitsInStock + UnitsOnOrder < ReorderLevel, vbred, vbblack)

UnitsInStockCtl.Forecolor = color
```

This topic illustrates the following:

Notice that the Outback Lager's UnitsInStock value is formatted in red since the sum of the UnitsInStock and UnitsOnOrder is less than the ReorderLevel.

ProductName	QuantityPerUnit	UnitPrice	UnitsInStock	UnitsOnOrder	ReorderLevel
Outback Lager	24 - 355 ml bottles	$15.00	15	10	30
Fløtemysost	10 - 500 g pkgs.	$21.50	26	0	0
Mozzarella di Giovanni	24 - 200 g pkgs.	$34.80	14	0	0
Röd Kaviar	24 - 150 g jars	$15.00	101	0	5
Longlife Tofu	5 kg pkg.	$10.00	4	20	5
Rhönbräu Klosterbier	24 - 0.5 l bottles	$7.75	125	0	25
Lakkalikööri	500 ml	$18.00	57	0	20
Original Frankfurter grüne Soße	12 boxes	$13.00	32	0	15

See Also

Suppressing or Forcing the Display of Zeros

Suppressing or Forcing the Display of Zeros

To suppress the display of fields with value zero, set their C1.C1Report.Field.Format property to "#" (pound sign). The pound sign is a formatting symbol that displays only significant digits (no leading or trailing zeros).

To force the display of a certain number of digits, use a format like "0000". The zero forces the display of a digit, including leading and trailing zeros.

Each format string can have up to three sections separated by semi-colons. If two sections are provided, the first is used for positive values and zero, the second for negative values. If three sections are provided, the first is used for positive values, the second for negative values, and the third for zero. For example: "#;(#);ZERO".

Sample Report Available:

For the complete report, see report "21: Suppress or Force Zeros" in the CommonTasks.xml report definition file, which is available for download from the CommonTasks sample on the ComponentOne HelpCentral Sample page.

See Also

Avoid repetition of value

Modifying Subreport

This section shows how to modify subreports. Subreports are regular reports contained in a field in another report – the main report – that are usually designed to display detail information based on a current value in the main report, in a master-detail scenario. For more information on subreports, see the C1.C1Report.Field.Subreport property in the reference section.

See Also

Showing Subreport's header

Retrieving Values from Subreports

Showing Subreport's Header

C1.C1Report.C1Report ignores Page Header and Page Footer sections in subreports. Instead, it uses the Page Header and Page Footer sections defined in the main report. This is the same behavior as in Microsoft Access.

In many cases, however, you may want your subreports to include header information across page breaks. To do this, place the headers in a Group Header section and set the section's C1.C1Report.Section.Repeat property to **True**. If your subreport doesn't have any groups, add an empty one.

Sample Report Available:

For the complete report, see report "14: Page Headers in Subreports" in the CommonTasks.xml report definition file, which is available for download from the CommonTasks sample on the ComponentOne HelpCentral Sample page.

See Also

Retrieving Values from Subreports

Retrieving Values from Subreports

In some cases you may want to pass data from the subreport back to the main report. Script variables can't be used for this because each report has its own script scope (this avoids the possibility of conflicting variable names).

To pass data from a subreport back to the main report, you have to store values in subreport fields or in the subreport's C1.C1Report.Field.Tag property, then have the main report read those values.

In the "15: Retrieve Values from Subreports" sample report, the subreport calculates the average unit price per product category and stores that value in its **Tag** property. The main report retrieves and displays that value.

Sample Report Available:

For the complete report, see report "15: Retrieve Values from Subreports" in the CommonTasks.xml report definition file, which is available for download from the CommonTasks sample on the ComponentOne HelpCentral Sample page.

See Also

Saving a Report Definition

When you are done creating and viewing your report, select the **Application** button and select **Save** to save the report definition file. The Designer saves the report definition in XML format, which can be read back into the Designer or directly into a C1.C1Report.C1Report component.

See Also

Loading a Report into the C1DocumentViewer Control

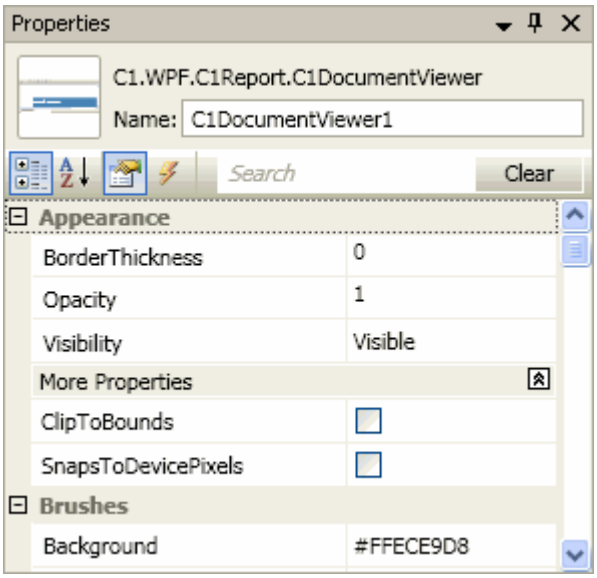
Loading a Report into the C1DocumentViewer Control

To load a report into the C1DocumentViewer control, just set the C1DocumentViewer.FileName and C1DocumentViewer.ReportName properties. You can do so in the designer, in code or in XAML:

In the Designer

To load a report in the designer, complete the following steps:

1. Select the C1DocumentViewer control and navigate to the Properties window.
2. Select the ellipsis button next to the C1DocumentViewer.FileName property to open the Open dialog box.



3. Locate and open a report definition file. For example, open the CommonTasks.xml file installed by default in the samples installation directory.

The reports available in the reports file should now appear in the C1DocumentViewer.ReportName drop-down box.

4. Click the C1DocumentViewer.ReportName drop-down box and select a report. For example, select 01: Alternating Background (Greenbar report).

In Code

Set theC1DocumentViewer.FileName and C1DocumentViewer.ReportName properties to the appropriate report definition location and report name in code. For example, add the following code to the **Page_Load** event to load the **CommonTasks.xml** report definition file into the C1DocumentViewer control:

Visual Basic	Copy Code

```
C1DocumentViewer1.FileName = "C:\Documents and Settings\<username>\My Documents\ComponentOne Samples\C1WPFReport\C1Report\Samples\XML\CommonTasks\CommonTasks.xml"
C1DocumentViewer1.ReportName = "01: Alternating Background (Greenbar report)"
```

C#	Copy Code

```
C1DocumentViewer1.FileName = "@C:\Documents and Settings\<username>\My Documents\ComponentOne Samples\C1WPFReport\C1Report\Samples\XML\CommonTasks\CommonTasks.xml";
C1DocumentViewer1.ReportName = "01: Alternating Background (Greenbar report)";
```

Note: Update the directory in the code above with the correct samples installation directory on your machine.

In XAML

Set theC1DocumentViewer.FileName and C1DocumentViewer.ReportName properties to the appropriate report definition location and report name in the <my:C1DocumentViewer> tag. For example, use the following XAML to load the **CommonTasks.xml** report definition file into the C1DocumentViewer control:

```
<my:C1DocumentViewer Name="C1DocumentViewer1" FileName="C:\Documents and
Settings\<username>\My Documents\ComponentOne
Samples\C1WPFReport\C1Report\Samples\XML\CommonTasks\CommonTasks.xml"
ReportName="01: Alternating Background (Greenbar report)"></my:C1DocumentViewer>
```

Note: Update the directory in the code above with the correct samples installation directory on your machine.

Run your application and observe:

The 01: Alternating Background (Greenbar report) report has been loaded in the C1DocumentViewer control: