# Reports for WinForms Task-Based Help

The task-based help assumes that you are familiar with programming in .NET, have a basic knowledge of reports, and know how to use controls in general. By following the steps outlined in the help, you will be able to create projects demonstrating a variety of **Reports for WinForms** features, and get a good sense of what the **Reports for** **WinForms** components can do.

## Reporting Task-Based Help

The task-based help assumes that you are familiar with programming in .NET, have a basic knowledge of reports, and know how to use controls in general. By following the steps outlined in the help, you will be able to create projects demonstrating a variety of C1Report features, and get a good sense of what the C1Report component can do.

Note that you should have the following namespace referenced to your project:

* + C1.C1Report

**CommonTasks.xml**

Most of this section's topics have pre-built reports that illustrate them. The pre-built reports can be found in the **CommonTasks.xml** report definition file, which if you have installed the Studio for WinForms samples, you can find in the Documents or MyDocuments folder in the ComponentOne Samples\Studio for WinForms\C1Report\C1Report\XML\CommonTasks directory.

### Adding Images to the Report

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

#### Creating Unbound Images

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**. For more information on how to access the **C1ReportDesigner**, see

[Accessing C1ReportDesigner from Visual Studio](#_bookmark102).

1. [Create a new report](#_bookmark104) or open an existing report. Once you have the report in the **C1ReportDesigner**, you can modify the report properties.
2. Click the **Design** button to begin editing the report.
3. In Design mode, click the **Add Unbound Picture** button  located in the **Fields** group of the **Design** tab.

The **Open** dialog box appears.

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

The following 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 in the **ComponentOne Samples** folder.

#### Creating Bound Images

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.

###### To add bound image fields to your report using the C1ReportDesigner:

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.

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

###### To add bound image fields to your report using 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

fEmployeePhoto.Picture = "Photo"

* + C#

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 in the **ComponentOne Samples** folder.

#### 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 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 PictureAlign and PictureShow properties.

**Sample Report Available**

For the complete report, see report "05: Watermark" in the **CommonTasks.xml** report definition file, which is available in the **ComponentOne Samples** folder.

### Creating Report Fields

With the report definition loaded into the component and a data source defined, you can add and edit report fields. The **Fields** group in the **C1ReportDesigner**'s **Design** tab allows you to easily add fields to your report. For button

descriptions, see [Enhancing the Report with Fields](#_bookmark106).



To add a field to your report, click any of these buttons and complete the following steps:

* + Drag the mouse over the report and the cursor changes into a cross-hair . Click and drag to define the rectangle that the new field will occupy, and then release the button to create the new field.

If you change your mind, hit the ESC key or click the **Undo** button to cancel the operation.

Note that C1Report only has one type of **Field** object. The buttons simply set some properties on the **Field**

object to make it look and act in a certain way.

OR

* You can also add fields by copying and pasting existing fields, or by holding down the CTRL key and dragging a field or group of fields to a new position to create a copy.

#### 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](#_bookmark102).

1. [Create a new report](#_bookmark104) or open an existing report. Once you have the report in the **C1ReportDesigner**, you can modify the report properties.
2. Click the **Close Print Preview** button to begin editing the report.
3. In the **Fields** group of the **Design** tab, click the **Add Chart Field** button .
4. Click in the Group Header section of your report and drag the field to resize the chart.
5. 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](#_bookmark107) topic.

**Sample Report Available**

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

#### Creating Custom Fields

You can create your own custom fields and add them to the Report Designer palette. To do that, you have to:

1. Create a custom field class that derives from **C1.Win.C1Report.Field**.
2. Register your custom field assembly in the Report Designer's settings file.

This is how the **Chart** and **Gradient** fields are implemented. The source code for these custom fields is available; you can use it as a starting point to create your own custom fields. The **Chart** and **Gradient** fields are registered in the **C1ReportDesigner** settings file with this entry:

<customfields>

<item value="C1.Win.C1Report.CustomFields;C1.Win.C1Report.CustomFields.Chart" />

<item value="C1.Win.C1Report.CustomFields;C1.Win.C1Report.CustomFields.Gradient"

/>

</customfields>

For example, to add a new field to the Designer palette, add your control to the <customfields> section in the "C1ReportDesigner.2.exe.settings" file:

<customfields>

<item

value="C1.Win.C1Report.CustomFields.2;C1.Win.C1Report.CustomFields.Chart"

/>

<item

value="C1.Win.C1Report.CustomFields.2;C1.Win.C1Report.CustomFields.Gradien t" />

<!-- THIS LINE WILL ADD A NEW FIELD TO THE DESIGNER -->

<item value="MyCustomFieldAssembly;MyCustomFieldAssembly.MyField" />

</customfields>

This assumes that your field is called "MyField" and it can be found in the assembly called "MyCustomFieldAssembly"

**Sample Report Available**

For the complete report, see report "12: Custom Fields" in the **CommonTasks.xml** report definition file, which is available in the **ComponentOne Samples** folder.

### Customizing the Page Headers

This section explains how to customize the behavior of page headers.

#### Adding a Continued Label to Headers on Page Breaks

Group Header sections are repeated across page breaks if their 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**. For more information on how to access the **C1ReportDesigner**, see

[Accessing C1ReportDesigner from Visual Studio](#_bookmark102).

1. [Create a new report](#_bookmark104) or open an existing report. Once you have the report in the **C1ReportDesigner**, 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 Detail.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: Detail.OnPrint

fContinued.Visible = true

1. Then select **GroupFooter** from the drop-down list above the Properties window.
2. Locate the GroupFooter.OnPrint property and click the empty field next to it, and then click the **ellipsis**

button.

1. 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 in the **ComponentOne Samples** folder.

#### 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 C1Report's PageHeader and 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 some script that shows or hides fields depending on the page being rendered. To do this, complete the following steps:

1. Open the **C1ReportDesigner**. For more information on how to access the **C1ReportDesigner**, see

[Accessing C1ReportDesigner from Visual Studio](#_bookmark102).

1. [Create a new report](#_bookmark104) or open an existing report. Once you have the report in the **C1ReportDesigner**, 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 OnFormat 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:

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 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 in the **ComponentOne Samples** folder.

### Customizing the Page Layout

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

#### Controlling Page Breaks

By default, 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.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.KeepTogether: Determines whether page breaks are allowed within sections. It has lower precedence than Group.KeepTogether.
* ForcePageBreak: Allows you to specify that page breaks should be inserted before, after, or before and after the section.
* 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.KeepTogether.

* ForcePageBreak: Allows you to specify that page breaks should be inserted before, after, or before and after the field.

Set these properties through the Properties grid of 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](#_bookmark102).

1. [Create a new report](#_bookmark104) or open an existing report. Once you have the report in **C1ReportDesigner**, 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 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:

cnt = cnt + 1 detail.forcepagebreak = "none" if cnt >= 10 then

cnt = 0

detail.forcepagebreak = "after" endif

**Sample Report Available**

For the complete report, see report "07: Force Page Breaks" in the **CommonTasks.xml** report definition file, which is available in the **ComponentOne Samples** folder.

#### 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 CanGrow and 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 in the **ComponentOne Samples** folder.

#### Creating a Gutter Margin

Gutter margins are extra space added to the margins next to the binding. They make it easier to bind the pages into folders, brochures, and so on.

To add a gutter margin to a report, you should increase the 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:

1. Open the **C1ReportDesigner**. For more information on how to access the **C1ReportDesigner**, see

[Accessing C1ReportDesigner from Visual Studio](#_bookmark102).

1. [Create a new report](#_bookmark104) or open an existing report. Once you have the report in the **C1ReportDesigner**, 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 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)

**Sample Report Available**

For the complete report, see report "10: Gutter" in the **CommonTasks.xml** report definition file, which is available in the **ComponentOne Samples** folder.

#### 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**. For more information on how to access the **C1ReportDesigner**, see

[Accessing C1ReportDesigner from Visual Studio](#_bookmark102).

1. [Create a new report](#_bookmark104) or open an existing report. Once you have the report in the **C1ReportDesigner**, you can modify the report properties.
2. Click the **Close Print Preview** button to begin editing the report.
3. In the **Fields** group of the **Design** tab, click the **Add Label** button to add a field to your report.
4. Click on your report where you want the field placed and drag to resize the field.
5. Set the following properties for the field:
	* Field.Name = **linesPerPage**
	* Field.Text = **14**
	* Field.Visible = **False**
6. To control the number of detail lines per page, use script. Select **Detail** from the drop-down list above the Properties window.
7. Locate the OnPrint property and click the empty field next to it, and then click the **ellipsis** button.

The **VBScript Editor** appears.

1. 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 in the **ComponentOne Samples** folder.

#### Specifying Custom Paper Size

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

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

###### To specify a custom paper size of 25" x 11" for your report using the C1ReportDesigner:

1. Open the **C1ReportDesigner**. For more information on how to access the **C1ReportDesigner**, see

[Accessing C1ReportDesigner from Visual Studio](#_bookmark102).

1. [Create a new report](#_bookmark104) or open an existing report. Once you have the report in the **C1ReportDesigner**, you can modify the report properties.
2. Click the **Close Print Preview** button to begin editing the report.
3. In Design mode, select your report from the drop-down list above the Properties window.
4. Locate **Layout** and expand the property node to view all available properties.
5. Set the **Custom Height** property to **25"** or **25in**.

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

1. Set the **Custom Width** property to **11"** or **11in**.

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

1. Set the **PaperSize** property to **Custom**.

###### To specify a custom paper size of 25" x 11" for your report using code:

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

* + Visual Basic

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

* + C#

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

**Sample Report Available**

For the complete report, see report "02: Custom Paper Size" in the **CommonTasks.xml** report definition file, which is available in the **ComponentOne Samples** folder.

### 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.

#### Adding Alternating Background Color

To create a report with alternating background color, use the OnPrint property of the Detail section to change the BackColor property of the section. To do this, complete the following steps:

1. Open the **C1ReportDesigner**. For more information on how to access the **C1ReportDesigner**, see

[Accessing C1ReportDesigner from Visual Studio](#_bookmark102).

1. [Create a new report](#_bookmark104) or open an existing report. Once you have the report in the **C1ReportDesigner**, you can modify the report properties.
2. Click the **Close Print Preview** button to begin editing the report.
3. In Design mode, select the report from the drop-down list above the Properties window.
4. Locate the OnOpen property and enter **cnt = 0**. This initializes the cnt variable.
5. Next, select **Detail** from the drop-down list above the Properties window.
6. Locate the OnPrint property and click the empty field next to it, and then click the **ellipsis** button.
7. 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

1. Click the **Preview** button to preview the report with alternating background.

###### This topic illustrates the following:

This report illustrates the 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 in the **ComponentOne Samples** folder.

#### Adding Conditional Formatting

In some cases you may want to change a field's appearance depending on the data it represents. 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**. For more information on how to access the **C1ReportDesigner**, see

[Accessing C1ReportDesigner from Visual Studio](#_bookmark102).

1. [Create a new report](#_bookmark104) or open an existing report. Once you have the report in the **C1ReportDesigner**, 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 (since this section contains the fields to add conditional formatting to).
4. Locate the OnFormat 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: 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 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 in the **ComponentOne Samples** folder.

#### 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 OnFormat property.

To specify an expression for the OnFormat property, complete the following steps:

1. Open the **C1ReportDesigner**. For more information on how to access the **C1ReportDesigner**, see

[Accessing C1ReportDesigner from Visual Studio](#_bookmark102).

1. [Create a new report](#_bookmark104) or open an existing report. Once you have the report in the **C1ReportDesigner**, 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 Property window's drop-down list to view the available properties for the Detail section.
4. Locate the OnFormat property and click the **ellipsis** button next to the property.
5. 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.



#### Suppressing or Forcing the Display of Zeros

To suppress the display of fields with value zero, set their Format property to **#**. 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 in the **ComponentOne Samples** folder.

### Loading Report Definitions

C1Report works by combining a report definition with raw data to create reports. In order to create reports, you need to load the report definition into C1Report. The following topics explain several ways to load report definitions.

#### Loading a Report Definition from a File

You can use the **C1ReportDesigner** to create report definition files (XML files that may contain one or more report definitions). For details on how to use the **C1ReportDesigner**, see the [Working with C1ReportDesigner](#_bookmark90) section of the documentation.

###### To load a report definition from a file at design time:

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

* + Right-click the **C1Report** component and select the **Load Report** menu option.

OR

* + Click the smart tag ( ) above the **C1Report** component and select **Load Report** from the **C1Report Tasks**

menu.

Using the **Select a report** dialog box to select the report you want, complete the following tasks:

1. Click the **ellipsis** button. The **Open** dialog box appears and you can select the XML file.
2. The available report definitions are listed in the **Report** drop-down box. Select the report definition to load.
3. Click **Load** and **OK** to close the dialog box. This is what the report selector dialog box looks like:



###### To load a report definition from a file using code:

To load a report definition from a file, use the Load method. It takes as parameters the name of the report definition file and the name of the report you want to load. If you want to list the reports contained in a report definition file, use the GetReportInfo method. It returns a list of the reports in the file.

For example:

* + Visual Basic

' Get list of reports in a report definition file

Dim reports As String() = c1r.GetReportInfo(reportFile)

' Load first report into C1Report component

c1r.Load(reportFile, reports(0))

* + C#

// Get list of reports in a report definition file

string[] reports = c1r.GetReportInfo(reportFile);

// Load first report into C1Report component

c1r.Load(reportFile, reports[0]);

#### Loading a Report Definition from a String

C1Report has a ReportDefinition property that allows you to get or set the entire report definition as a string. This is a convenient way to store and retrieve report definitions in databases or in data structures within your application.

The **ReportDefinition** string contains the exact same XML that would be stored in the report definition file, for example:

* Visual Basic

' Load report definition into C1Report component

c1r.Load(reportFile, reportName)

' Copy report definition to the clipboard Dim repDef As String = c1r.ReportDefinition Clipboard.SetDataObject(repDef)

' Copy report definition to c1r2 component

c1r2.ReportDefinition = repDef

* C#

// Load report definition into C1Report component

c1r.Load(reportFile, reportName);

// Copy report definition to the clipboard string repDef = c1r.ReportDefinition; Clipboard.SetDataObject(repDef);

// Copy report definition to c1r2 component

c1r2.ReportDefinition = repDef;

### Modifying Subreports

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 Subreport property in the reference section.

#### Adding Page Headers to Subreports

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 would like 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 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 in the **ComponentOne Samples** folder.

#### 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 Tag property, then have the main report read those values.

In this sample, 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 in the **ComponentOne Samples** folder.

### Rendering Reports (Previewing, Printing, and Exporting)

Once the report definition has been loaded into the component and a data source has been defined, you can render the report to the printer, into preview controls, or to report files.

#### Displaying a Progress Indicator While the Report Renders

Most preview applications have progress indicators that show which page is being rendered and have a button that allows the user to cancel the report while it is being generated. The .NET print preview controls provide this automatically for you. If you are printing the report directly to the printer or exporting it to a file, however, there is no built-in progress report UI.

Use C1Report events to create a progress report dialog, or to update a status bar while the report is being rendered. The StartPage and EndReport events are sufficient to provide feedback on which page is being printed and when the report is finished. For example, this code uses the StartPage event to provide feedback through a status bar (StatusStrip1):

* Visual Basic

Private Sub c1r\_StartPage(ByVal sender As System.Object, ByVal e As C1.Win.C1Report.ReportEventArgs) Handles c1r.StartPage

StatusStrip1.Text = String.Format("Rendering page {0} of '{1}'...", c1r.Page, c1r.ReportName)

End Sub

* C#

private void c1r\_StartPage(object sender, ReportEventArgs e)

{

statusStrip1.Text = string.Format("Rendering page {0} of '{1}'...", c1r.Page, c1r.ReportName);

}

To cancel the report before it is finished, add a **Cancel** button to your application and use it to set the

**C1Report**.Cancel property to **True**. For example:

* Visual Basic

Private Sub \_btnCancel\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles btnCancel.Click

c1r.Cancel = True Close()

End Sub

* C#

private void \_btnCancel\_Click(object sender, System.EventArgs e)

{

c1r.Cancel = true;

Close();

}

Note that you may also want to provide progress bars and "page n of m" indicators, but that is generally difficult to do because the page count is not known until the report has been rendered.

**Sample Project Available**

For a complete sample using a progress indicator, see the **ProgressIndicator** sample installed in the

**ComponentOne Samples** folder.

#### Previewing Reports

To preview the report, use the use the **C1Report**.Document property. Assign it to the **Document** property in the **Reports for WinForms** preview control or to the .NET PrintPreview or PrintPreviewDialog controls and the preview controls will display the report and allow the user to browse, zoom, or print it. For example:

* Visual Basic

' Load report definition

c1r.Load(reportFile, reportName)

' Preview the document

c1preview1.Document = c1r.Document

* C#

// Load report definition

c1r.Load(reportFile, reportName);

// Preview the document

c1preview1.Document = c1r.Document;

**Note:** C1Report works with the .NET preview components, but it is optimized to work with the included **Reports for WinForms** preview controls. When used with the included controls, you can see each report page as it is generated. With the standard controls, you have to wait until the entire report is ready before the first page is displayed.

#### Printing Reports

To print a report directly to the printer, use the **C1Report**.Document property. This property returns a standard

**PrintDocument** object that has a **Print** method and exposes printer and page settings.

For example, the following code shows a print dialog and prints the report:

* Visual Basic

' Load report definition

c1r.Load(reportFile, reportName)

' Get PrintDocument object

PrintDocument doc = c1r.Document

' Show a PrintDialog so user can customize the printing

Dim pd As PrintDialog = New PrintDialog()

' Use PrinterSettings in report document

pd.PrinterSettings = doc.PrinterSettings

' Show the dialog and print the report

If pd.ShowDialog() = DialogResult.OK Then doc.Print()

End If

' Cleanup and release PrintDialog resources

pd.Dispose()

* C#

// Load report definition

c1r.Load(reportFile, reportName);

// Get PrintDocument object

PrintDocument doc = c1r.Document;

// Show a PrintDialog so user can customize the printing

PrintDialog pd = new PrintDialog();

// Use PrinterSettings in report document

pd.PrinterSettings = doc.PrinterSettings;

// Show the dialog and print the report if (pd.ShowDialog() == DialogResult.OK) doc.Print();

// Cleanup and release PrintDialog resources

pd.Dispose();

#### Exporting the Report

###### Exporting the report to common file formats

C1Report has a RenderToFile method that allows you to export your report to several file formats, including HTML, RTF, PDF, TIFF, Text, and XLS. For example, the following code creates PDF and XLS versions of a report:

* + Visual Basic

' Load report definition

c1r.Load(reportFile, reportName)

' Export to PDF

c1r.RenderToFile(outFile + ".pdf", FileFormatEnum.PDF) c1r.RenderToFile(outFile + ".xls", FileFormatEnum.Excel)

* + C#

// Load report definition

c1r.Load(reportFile, reportName);

// Export to PDF

c1r.RenderToFile(outFile + ".pdf", FileFormatEnum.PDF);

c1r.RenderToFile(outFile + ".xls", FileFormatEnum.Excel);

**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.

###### Exporting the report to custom formats

If you want to export the report to a format that is not supported by C1Report, you can write your own export filter class and use the **C1Report**.RenderToFilter method to render the report into your custom filter.

Custom filters are classes that derive from the **C1.Win.C1Report.**ExportFilter class and override a few simple methods like StartReport, StartSection, RenderField, EndSection, and EndReport.

Writing a custom export filter is not difficult. It can be used, for example, to create custom XML representations of your reports for later processing by other applications.

### Saving a Report Definition

When you are done creating and viewing your report in the **C1ReportDesigner** application, 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 C1Report component.