ComponentOne

Sizer for WinForms

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

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

Trademarks

Warranty

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

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

Copying and Distribution

Table of Contents

Sizer for WinForms Overview

Create resolution-independent applications without having to write any code with **Sizer for WinForms**. The powerful **C1Sizer** and **C1SizerLight** components, provided with **Sizer for WinForms**, make this possible. Now you can resize all contained controls proportionally, so your Windows Form retains its appearance at any resolution.

C1Sizer is a container control with a powerful grid layout manager that extends the basic layout capabilities provided by the .NET Framework (Dock and Anchor properties). The **C1Sizer** control allows you to define a grid made up of bands and then add controls that snap to these bands. When the **C1Sizer** control is resized, the bands are automatically recalculated, and contained controls move automatically to their new positions. You can set up bands at design time and configure them to act as splitters or to keep their size constant when the control is resized.

Help with WinForms Edition

Getting Started

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

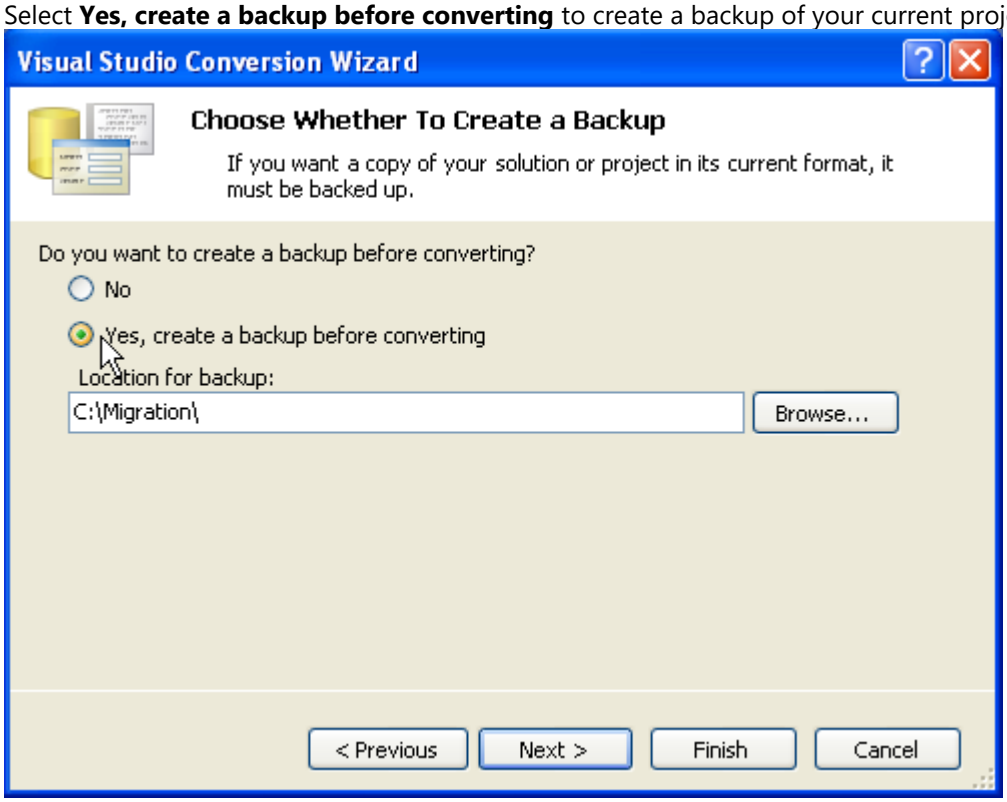
Migrating a C1Sizer Project to Visual Studio 2005

To migrate a project using ComponentOne components to Visual Studio 2005, there are two main steps that must be performed. First, you must convert your project to Visual Studio 2005, which includes removing any references to a previous assembly and adding a reference to the new assembly. Secondly, the .licx file, or licensing file, must be updated in order for the project to run correctly.

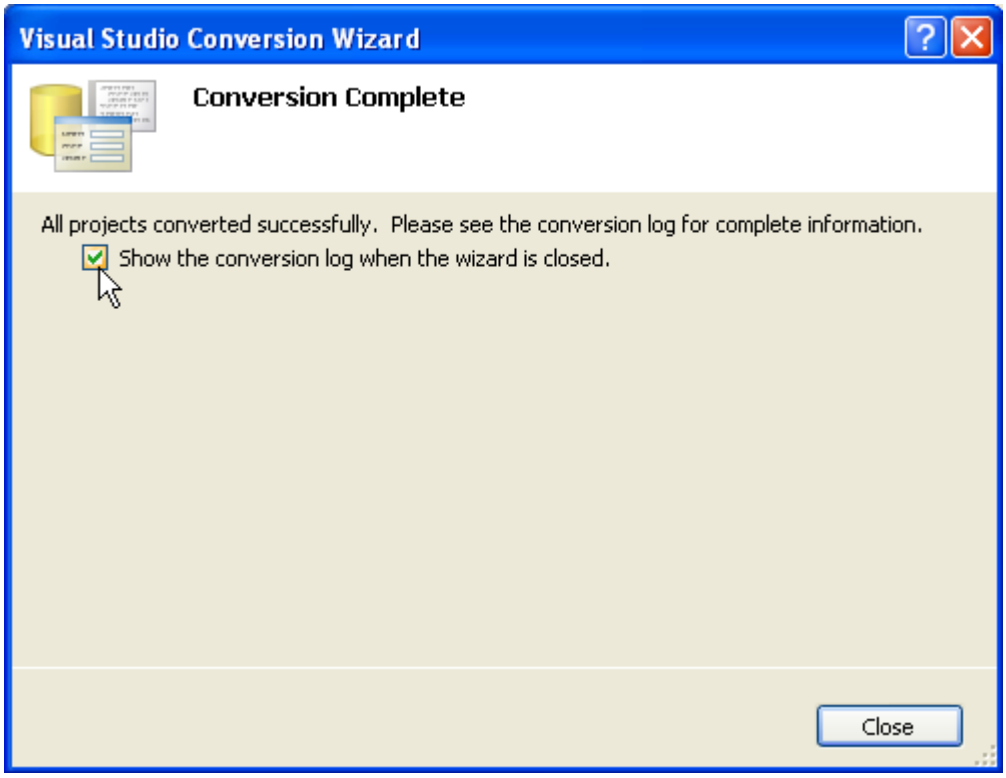
To convert the project:

1. Open Visual Studio 2005 and select **File, Open Project**.
2. Locate the **.sln** file for the project that you wish to convert to Visual Studio 2005. Select it and click **Open**. The **Visual Studio Conversion Wizard** appears.

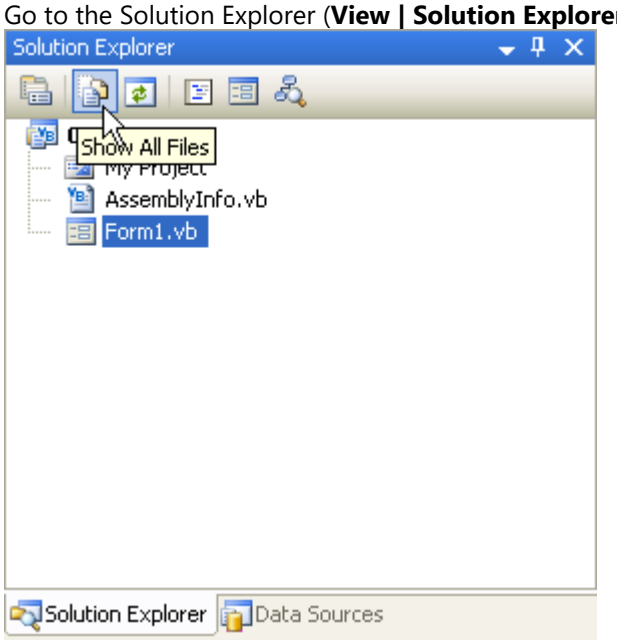
3. Click **Next**.
4. Select **Yes, create a backup before converting** to create a backup of your current project and click **Next**.



5. Click **Finish** to convert your project to Visual Studio 2005. The **Conversion Complete** window appears.
6. Click **Show the conversion log when the wizard is closed** if you want to view the conversion log.

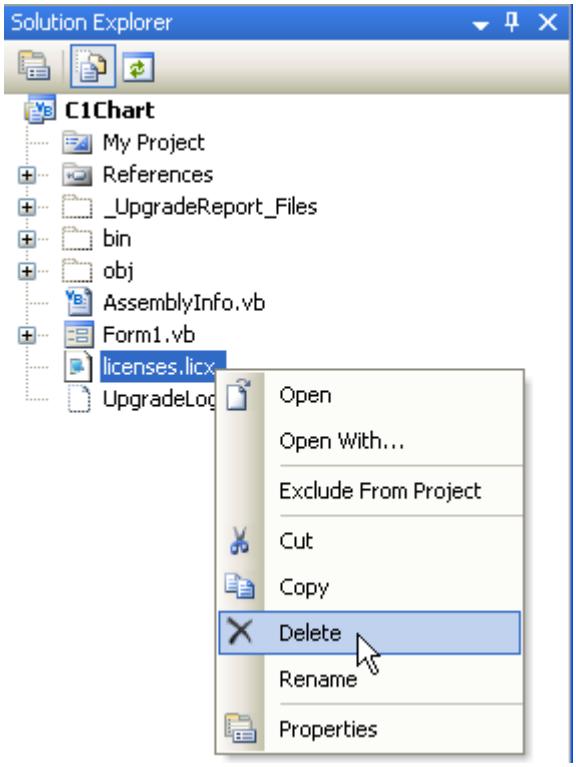


7. Click **Close**. The project opens. Now you must remove references to any of the previous ComponentOne .dlls and add references to the new ones.
8. Go to the Solution Explorer (**View | Solution Explorer**), select the project, and click the **Show All Files** button.

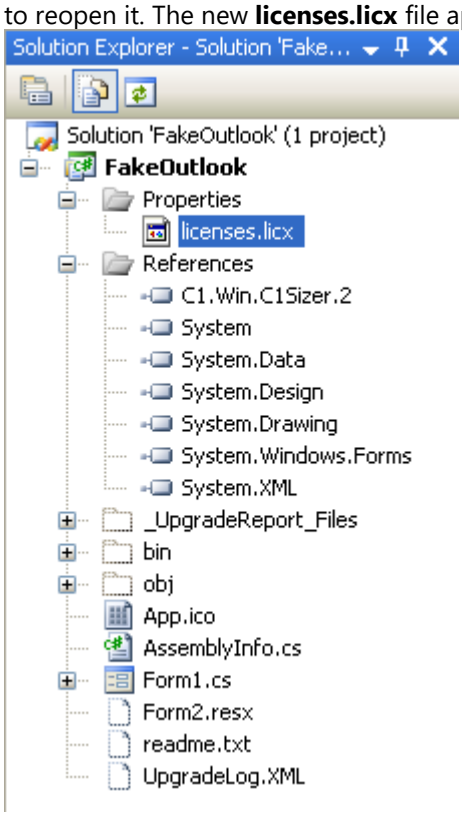


> **Note**: The Show All Files button does not appear in the Solution Explorer toolbar if the Solution project node is selected.

9. Expand the **References** node, right-click C1.Win.C1Sizer and select **Remove**.
10. Right-click the **References** node and select **Add Reference**.
11. Locate and select **C1.Win.C1Sizer.2.dll**. Click **OK** to add it to the project.
12. To update the .licx file:
13. In the Solution Explorer, right-click the **licenses.licx** file and select **Delete**.



14. Click **OK** to permanently delete **licenses.licx**. The project must be rebuilt to create a new, updated version of the .licx file.
15. Click the **Start Debugging** button to compile and run the project. The new .licx file may not be visible in the Solution Explorer.
16. Select **File | Close** to close the form and then double-click the **Form.vb** or **Form.cs** file in the Solution Explorer to reopen it. The new **licenses.licx** file appears in the list of files.



The migration process is complete.

Key Features

Sizer for WinForms includes the following features:

- **Create a Gradient Background**
  With the easy-to-use C1Sizer Gradient Editor that is accessible from the "Edit Gradient" Smart Tag, you can quickly add a gradient to the sizer panel and customize the gradient settings.
- **Update your Form with Rounded Corners**
  The new Border property gives you the flexibility to add rounded corners to the sizer and modify the color, thickness, and more.
- **Flexibility when Adding Images to the Background**
  With the new image properties you have more power and flexibility for using images as part of the background. You can now control the alignment and scale of the images.
- **Arrange Controls on the Form with Ease**
  The C1Sizer control act like a piece of graph paper where each control will occupy one or more of the grid boxes on the form. The graph allows you to neatly layout the controls on the form. You have the flexibility to set up the grid before or after you create the child controls.
- **Resize all Contained Controls Proportionally**
  The C1SizerLight component is a non-visual component that keeps track of a form's size and position When the form is resized, the C1SizerLight component resizes all contained controls proportionally, so the form retains its appearance at any resolution. C1SizerLight can also resize the fonts on all or some of the contained controls.
- **Easy-to-use Grid Editor**
  Efficiently set up your form with the Grid Editor. Simply add one or more C1Sizer controls to the form, and set their Dock property to fill the form. Then use the Grid Editor to set up a grid layout, and add controls which will snap to the grid. You can also set the number and dimension of the grid's bands (rows and columns), and also specify which bands should act as splitters, and which should have fixed dimensions.

Design-Time Support

C1Sizer provides customized context menus, smart tags, and a designer that offers rich design-time support and simplifies working with the object model.

The following sections describe how to use **C1Sizer's** design-time environment to configure **C1Sizer's** controls/components.

C1Sizer Smart Tag

In Visual Studio, the **C1Sizer** control includes a smart tag. A smart tag represents a short-cut tasks menu that provides the most commonly used properties in each component/command.

The **C1Sizer** control provides quick and easy access to the **C1Sizer Grid Editor** and common properties through its smart tag.

To access the **C1Sizer Tasks** menu, click on the smart tag (▶) in the upper-right corner of the **C1Sizer** control. This will open the **C1Sizer Tasks** menu.

The **C1Sizer Tasks** menu operates as follows:

About C1Sizer

Clicking on the **About** item displays the **About C1Sizer** dialog box, which is helpful in finding the version number of **C1Sizer** and online resources.

Edit Gradient

Clicking on the **Edit Gradient** item opens the **C1Sizer Grid Editor**. For more information about the C1Sizer Gradient Editor's elements, see C1Sizer Gradient Editor.

Edit Grid

Clicking on the **Edit Grid** item opens the **C1Sizer Grid Editor**. For more information about the C1Sizer Grid Editor's elements, see C1Sizer Grid Editor.

Auto Grid

Clicking on the **Auto Grid** item clears all unused bands in the grid that don't have any controls attached to them.

Clear Grid

Clicking on the **Clear Grid** item removes the bands from the grid.

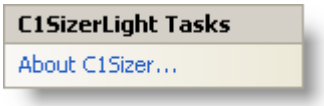
Dock in parent container

Clicking on the **Dock in parent container** link docks the **C1Sizer** control inside its parent container.

C1SizerLight Smart Tag

The **C1SizerLight** component provides quick and easy access to the **About C1Sizer** dialog box through its smart tag.

To access the **C1SizerLight Tasks** menu, click on the smart tag (▶) in the upper right corner of the **C1SizerLight** component. This will open the **C1SizerLight Tasks** menu.



The **C1SizerLight Tasks** menu operates as follows:

About C1Sizer

Clicking on the **About** item displays the **About C1Sizer** dialog box, which is helpful in finding the version number of **C1Sizer** and online resources.

C1Sizer Context Menus

C1Sizer has additional commands with each context menu that Visual Studio provides for all .NET controls.

C1Sizer Context Menu

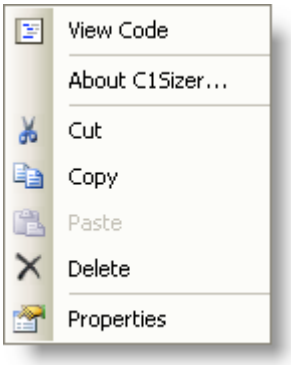
Right-click anywhere on the list to display the **C1Sizer** context menu.

The following table provides a brief description of the custom commands added by **C1Sizer**:

Commands	Description
About C1Sizer	Displays the **About C1Sizer** dialog box, which is helpful in finding the version number of **C1Sizer** and online resources.
Edit Grid	Opens the **C1Sizer Grid Editor**.
Edit Gradient	Opens the C1Sizer Gradient Editor.
Auto Grid	Clears all unused bands in the grid that don't have any controls attached to them.
Clear Grid	Removes the bands from the grid.

C1SizerLight Context Menu

Right-click anywhere on the **C1SizerLight** component to display the **C1SizerLight** context menu.



The following table provides a brief description of the custom commands added by **C1SizerLight**:

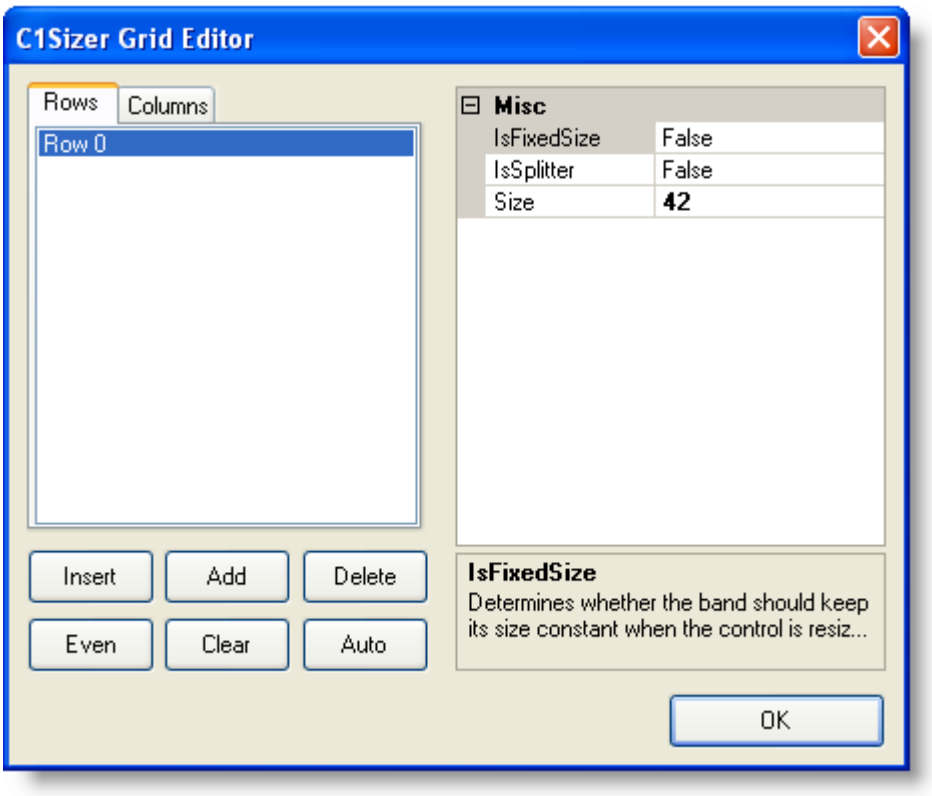
Commands	Description
About C1Sizer	Displays the **About C1Sizer** dialog box, which is helpful in finding the version number of **C1Sizer** and online resources.

C1Sizer Grid Editor

C1Sizer has a **Grid Editor** to edit or add bands to rows and columns of the grid at design time.

To access the C1Sizer Grid Editor:

Open the **C1Sizer Tasks** menu and click on the **Edit Grid** item or right-click the **C1Sizer** control and select **Edit Grid** from its context menu.



Tabs

C1Sizer Grid Editor consist of two important tabs: **Rows** and **Columns**. Both tabs share the same command buttons and properties. Click the **Columns** tab to create or modify the bands in the *Columns* of the grid. Click the **Rows** tab to create or modify the bands in the *Rows* of the grid.

Command Buttons

The following command buttons are available in both tabs:

Command	Description
Insert	Inserts a new band at the selection.
Add	Adds a new band to the collection.
Delete	Deletes the selected bands.
Even	Makes all bands the same size.
Clear	Removes all bands.
Auto	Creates bands based on child control.
OK	Updates the changes and closes the C1Sizer Grid Editor.

Properties

The following properties are available in both tabs:

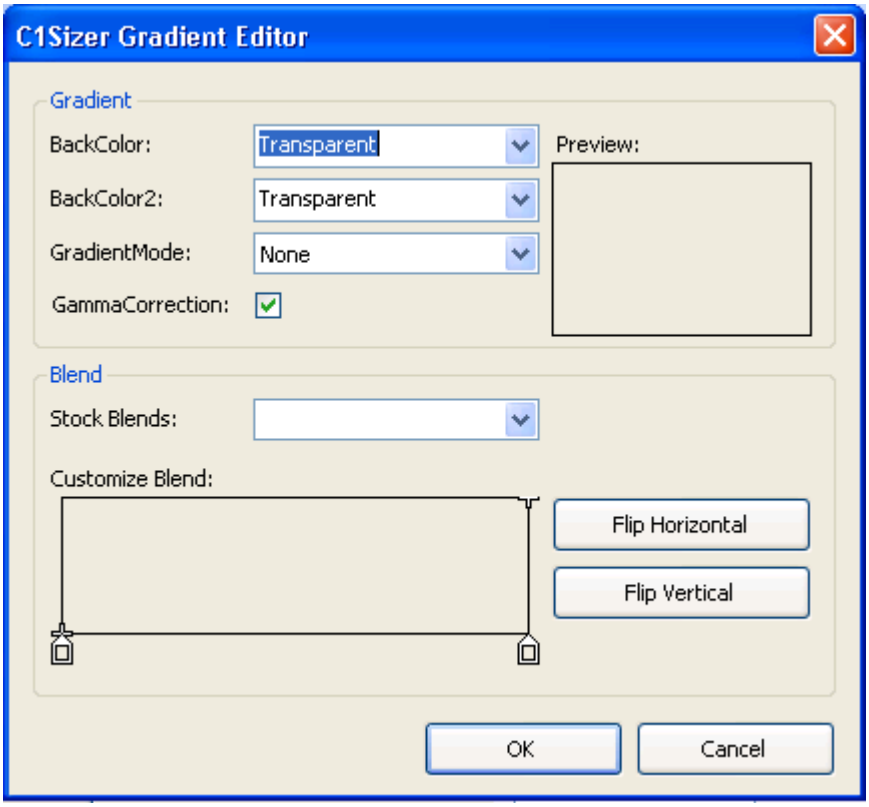
Property	Description
IsFixedSize	Determines whether the band should keep its size constant when the control is resized. The default value is False.
IsSplitter	Determines whether the band acts like a splitter (can be resized with the mouse at run time). The default value is False.
Size	Gets or sets the row height or column width in pixels.

C1Sizer Gradient Editor

C1Sizer has a **Gradient Editor** to add gradients to the C1Sizer grid at design time.

To access the C1Sizer Gradient Editor:

Open the **C1Sizer Tasks** menu and click on the **Edit Gradient** item or right-click the **C1Sizer** control and select **Edit Gradient** from its context menu.



Gradient Group

C1Sizer Gradient Editor consist of two important groups: **Gradient** and **Blend**. The Gradient group consist of BackColor and Gradient properties and a Preview window so you can preview the gradient updates you make to the C1Sizer control.

The Gradient group consists of the following properties:

Property	Description
C1Sizer.BackColor	Gets or sets the Drawing.Color used to paint the background.
Gradient.BackColor2	Gets or sets the secondary color used to build the background gradient.
GradientMode	Specifies the background gradient mode.
Gradient.GammaCorrection	Gets or sets whether to apply gamma correction to the background gradient.

Blend Group

The Blend group enables you to choose from a stock blend and customize the blend by flipping the blend colors horizontally or vertically.

Using the C1SizerLight Component

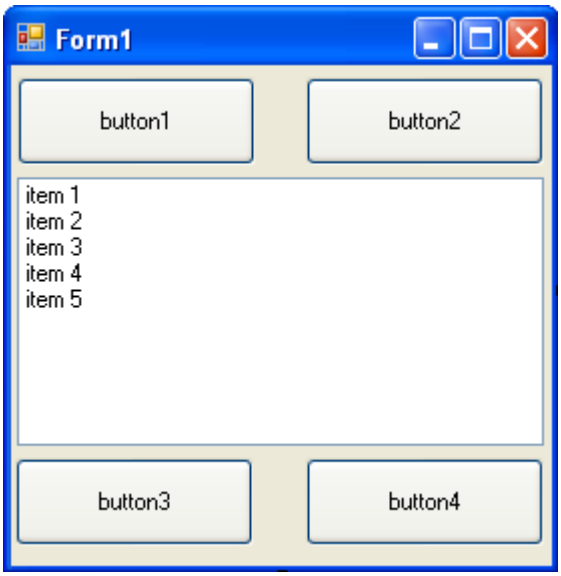
C1SizerLight is a non-visual component. When you add it to a form, it keeps track of the form's size and position. When the form is resized, **C1SizerLight** resizes all contained controls proportionally, so the form retains its appearance at any resolution. **C1SizerLight** can also resize the fonts on all or some of the contained controls.

Using the **C1SizerLight** component is extremely easy. Start by creating a Windows Form as usual (or open an existing one), and simply add a **C1SizerLight** component to it. Then run the project and resize the form. All controls on the form will be automatically resized with the form (including their fonts).

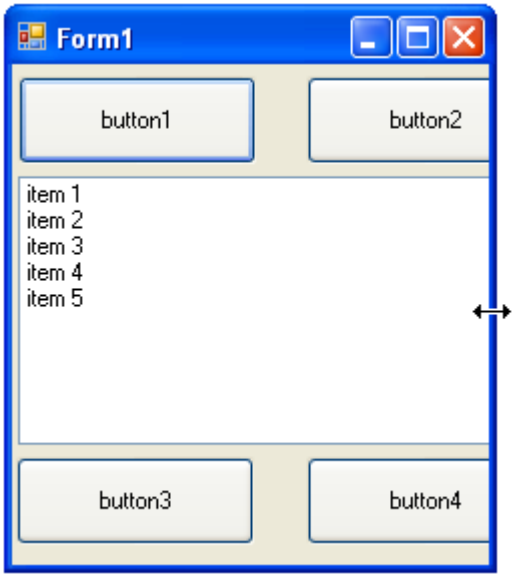
Quick Start Tutorial

This Quick Start tutorial shows how the C1SizerLight component functions. It also demonstrates how the C1SizerLight.ResizeFonts property and C1SizerLight.ResizingFont event work.

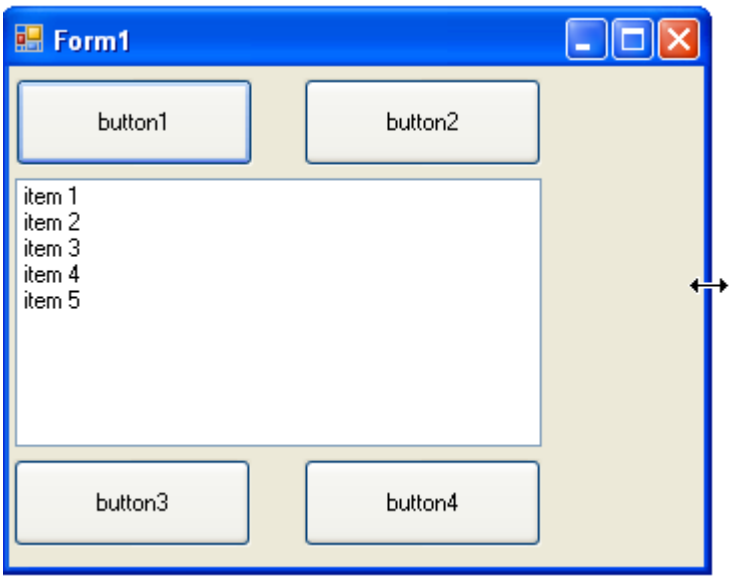
Open Visual Studio and add buttons and list box controls on the form, so it looks like the following:



Now run the project and resize the form. If you make it smaller, parts of the controls will be clipped.

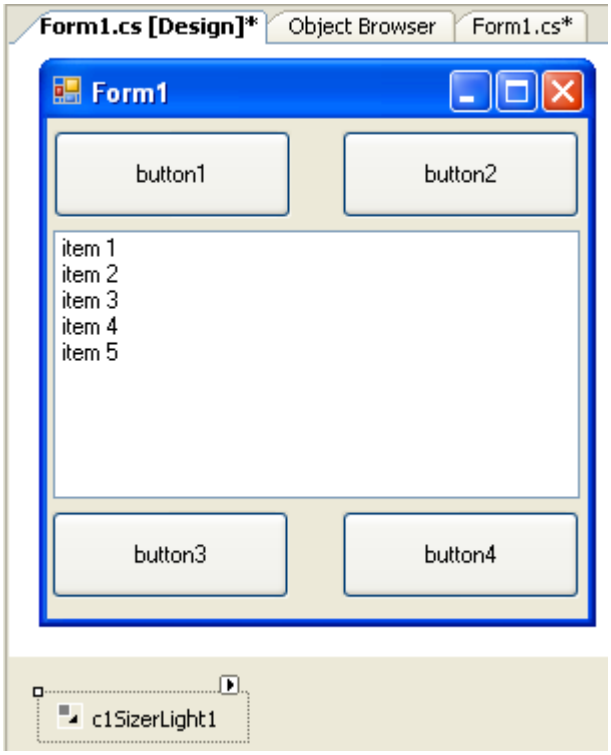


If you make it bigger, you will see an empty gray area.

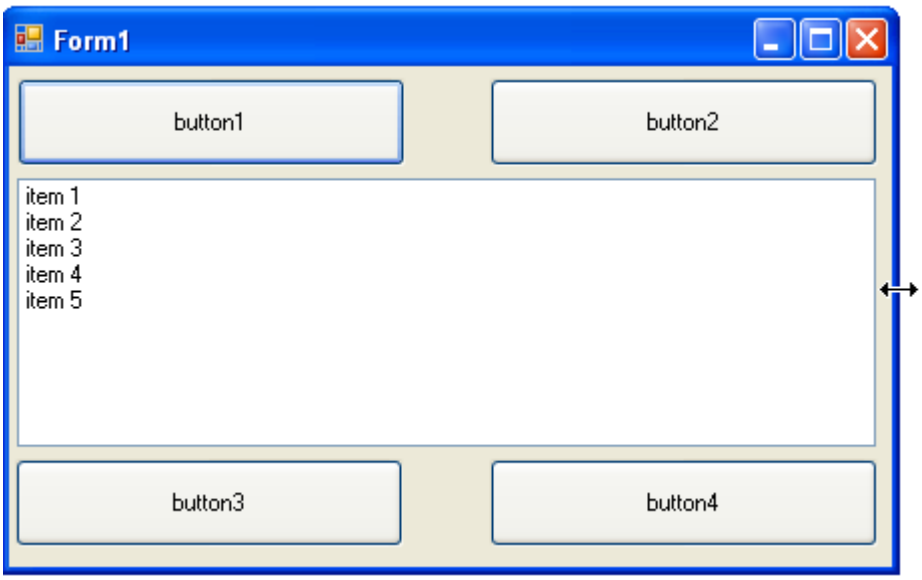


You could improve the controls' appearance on the form using the **Dock** and **Align** properties on the controls, but there is no combination of these two properties that will prevent the controls from overlapping each other or leaving blank areas on the form. Also, depending on your monitor's resolution, the fonts may appear very small when you enlarge the form.

Instead of making the form size fixed and giving up the resize feature, add a C1SizerLight component to the form. C1SizerLight is a non-visual component, so it will appear in the tray area below the form.



You do not have to set any properties or write any code, just run the project again and resize the form. All contained controls will automatically resize and the form will retain its aspect:

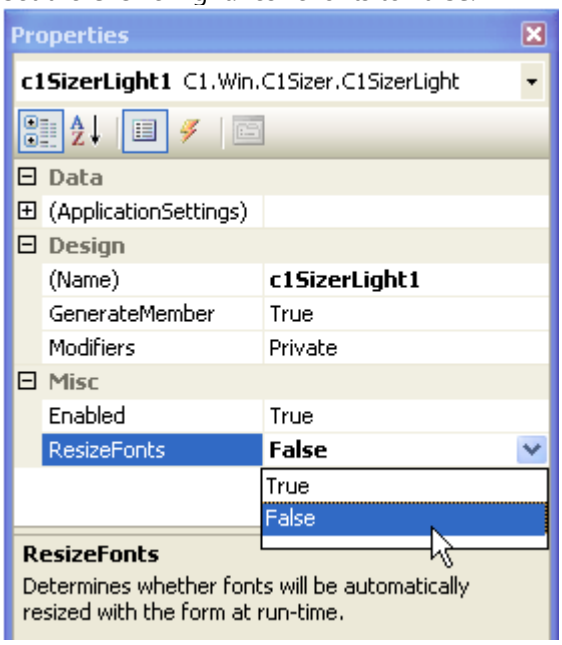


Resizing Fonts

Notice that when you resized the form, the fonts in the button controls were resized to accommodate the new size. This is the default behavior provided by C1SizerLight.In some cases, you may want to prevent that and keep all the original font sizes. You can do this using the C1SizerLight.ResizeFonts property. Set C1SizerLight.ResizeFonts to **False** and the fonts will not resize.

To set the C1SizerLight.ResizeFonts to **False** at design-time use the following steps:

1. Select **C1SizerLight** in the **Properties** sheet.
2. Set the C1SizerLight.ResizeFonts to **False**.



You may also want to prevent font resizing in specific controls. For example, it would probably be a good idea to keep the font size constant for the list control on our sample form, and update the fonts on the buttons only. (In general, controls that can scroll their contents do not need the font resizing feature). To do this, you need to handle the C1SizerLight.ResizingFont event and cancel it for some controls. For example, the following event handler will cause the component to resize fonts in button controls only:

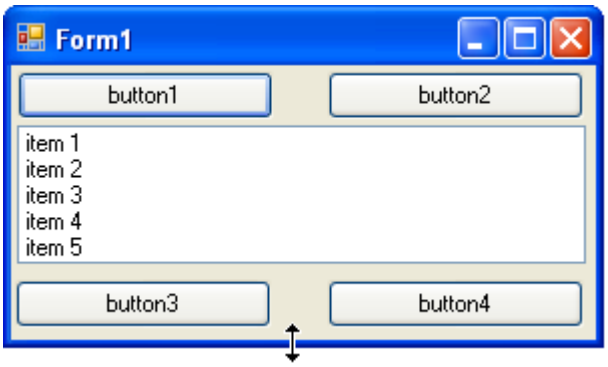
Visual Basic

```
Private Sub C1SizerLight1_ResizingFont(ByVal sender As Object, _
    ByVal e As C1.Win.C1Sizer.C1SizerLightEventArgs) Handles _
    C1SizerLight1.ResizingFont
    If Not (e.Control.GetType() Is GetType(Button)) Then
    e.Cancel = True
    End If
End Sub
```

C#

```
private void c1SizerLight1_ResizingFont(object sender,
    C1.Win.C1Sizer.C1SizerLightEventArgs e)
    {
    if (!(e.Control is Button))
    e.Cancel = true;
}
```

If you run the project again, you will notice that the **ListBox** control retains the original font when the form is resized.



Using C1SizerLight with C1Sizer

Note that if you use C1SizerLight along with C1Sizer in your application, the C1SizerLight does not resize the fonts of the controls whose parent is C1Sizer. This is a limitation of using C1Sizer with C1SizerLight. So, to resize the fonts of controls in a C1Sizer when the form is resized, use the following code snippet to change the Font property on the form.

Visual Basic

```
Private _cs As Size
    Private _fs As Single
    Public Sub New()
        InitializeComponent()
        _fs = Font.Size
        _cs = ClientSize
        AutoScaleMode = AutoScaleMode.None
    End Sub
    Protected Overrides Sub OnResizeEnd(e As EventArgs)
        Dim sz = ClientSize
        Dim factor = Math.Min(sz.Width / CSng(_cs.Width), sz.Height /
CSng(_cs.Height))
        Font = New Font(Font.Name, CSng(_fs * factor))
        MyBase.OnResizeEnd(e)
```

```
    End Sub
```

C#

```
private Size _cs;
private float _fs;

public Form1()
{
  InitializeComponent();
  _fs = Font.Size;
  _cs = ClientSize;
  AutoScaleMode = System.Windows.Forms.AutoScaleMode.None;
}
protected override void OnResizeEnd(EventArgs e)
{
        var sz = ClientSize;
        var factor = Math.Min(sz.Width / (float)_cs.Width, sz.Height /
(float)_cs.Height);
        Font = new Font(Font.Name, (float)(_fs * factor));
        base.OnResizeEnd(e);
        }
```

However, this will affect all those controls on the form that do not specify their own fonts. If a control does specify its own font (as the labels do), then those would not be affected.

Docked and Nested Controls

Although C1SizerLight provides easy-to-use power, you may still want to use the native layout capabilities in the .NET framework. For example, if you add a **ToolBar** or **StatusBar** control to the form, you probably want those controls docked to the top or bottom of the form, and **C1SizerLight** should not change their dimensions.

That is why **C1SizerLight** will not resize any docked controls. It will take into account the fact that they are docked and will reduce the form's client area accordingly, so all non-docked controls will still be resized correctly.

This makes it easy to use **C1SizerLight** and still takes advantage of the layout capabilities built into the .NET framework.

MDI Forms

C1SizerLight also works with MDI child forms. Simply add a **C1SizerLight** component to each child form and they will be resized proportionally, just like any regular form.

Using the C1Sizer Control

C1Sizer is a container control with a powerful grid layout manager that extends the basic layout capabilities provided by the .NET framework (**Dock** and **Anchor** properties). The **C1Sizer** control allows you to define a grid made up of bands, then add controls that snap to these bands. When the **C1Sizer** control is resized, the bands are automatically recalculated, and contained controls move automatically to their new positions. You can set up bands at design time and configure them to act as splitters, or to keep their size constant when the control is resized.

Think of **C1Sizer** as a piece of graph paper. Each component will occupy one or more of the little boxes on the paper. This is similar to tables in word processors such as Microsoft Word, where you can merge adjacent cells, and to Java's Grid Bag Layout, which works in a similar way.

There are two basic approaches to using the **C1Sizer** control in your projects: You can set up the grid before or after you create the child controls. For more information how to set up the grid before you create the child controls, see Tutorial 1: Set up the grid, then add the controls or how to set up the grid after you create the controls, see Tutorial 2: Add the controls, then set up the grid.

The following table lists the controls that require an additional property other than the Band.Bounds property:

Control	Property Setting
TextBox	The TextBox control allows you to set its height only if MultiLine = True. Otherwise the height is set automatically based on the Font.
ComboBox	The ComboBox control allows you to set its height only if DrawMode = OwnerDraw.
ListBox	The ListBox control allows you to set its height (exactly) only if IntegralHeight = False.

The next section includes two tutorials that that illustrate some of the main features in the C1Sizer control. The tutorials walk you through the creation of several simple projects, describing each step in detail. The distribution CD contains more sophisticated samples that can be used as a reference.

Here is a brief overview of the tutorials that follow:

Name	Description
Tutorial 1	This tutorial provides a thorough introduction to the primary features of the **C1Sizer** control. It shows how to set up the grid by inserting columns and rows, adding labels and data entry controls to the grid, cleaning up unused bands, setting up the splitters, and creating fix-sized bands.
Tutorial 2	This tutorial also provides an introduction to the typical usage of the **C1Sizer** control. It shows how to add the controls and then set up the grid.

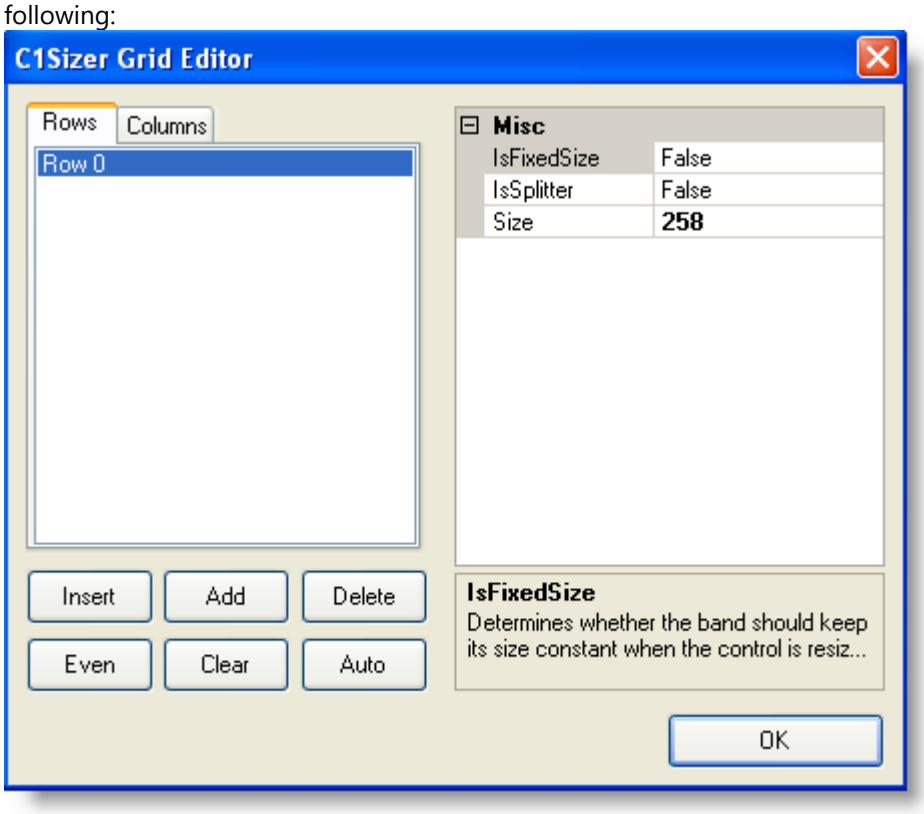
Tutorial 1: Set up the grid, then add the controls

In this tutorial you'll learn how to use elements and properties in the **C1Sizer Grid Editor** to set up a grid. This tutorial demonstrates the following **C1Sizer** features:

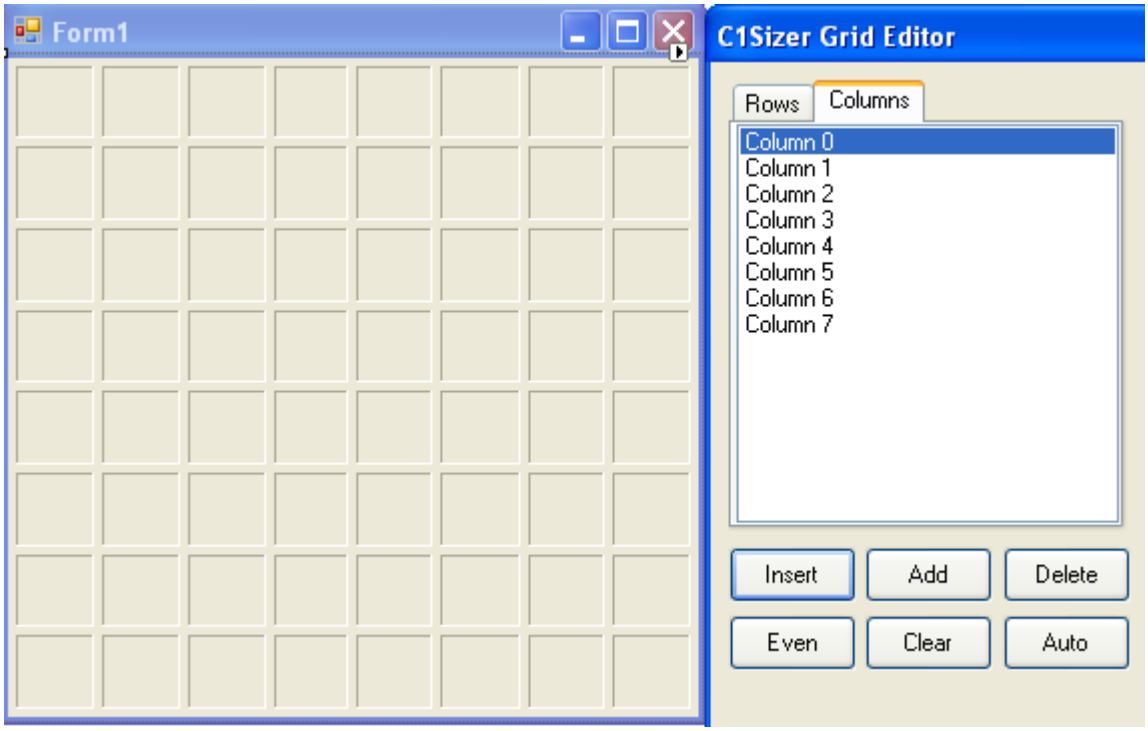
- Inserting/deleting rows and columns
- Arranging controls on the grid
- Settingthe C1Sizer.SplitterWidth and C1Sizer.BorderWidth properties
- Applying AutoGrid to clean up unused bands
- Setting the Band.IsFixedSize and Band.IsSplitter properties

To set up the grid for **C1Sizer**, complete the following steps:

1. Open Visual Studio and create a new project, then add a **C1Sizer** control to the form and set its **Dock** property to **DockStyle.Fill** so it takes up the whole form. At this point, the **C1Sizer** does not have a grid layout set up yet, or any child controls, so it will display a message with brief instructions on how to use the control, but this can be ignored for now.
2. Now, right-click the **C1Sizer** control and select the **Edit Grid** option from the menu. This will bring up the **C1Sizer Grid Editor** dialog box so you can set up the grid layout. The **C1Sizer Grid Editor** appears like the following:

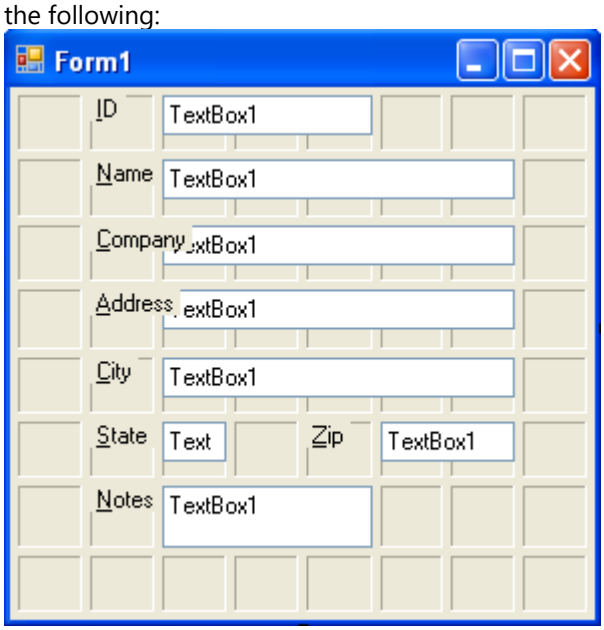


3. Position the editor next to the grid so you can see the effect of the changes on the control. Next, click the **Insert** button until the grid has eight rows. Then, click the **Columns** tab and click **Insert** again until the grid has eight columns. This is what the control will look like at design time:



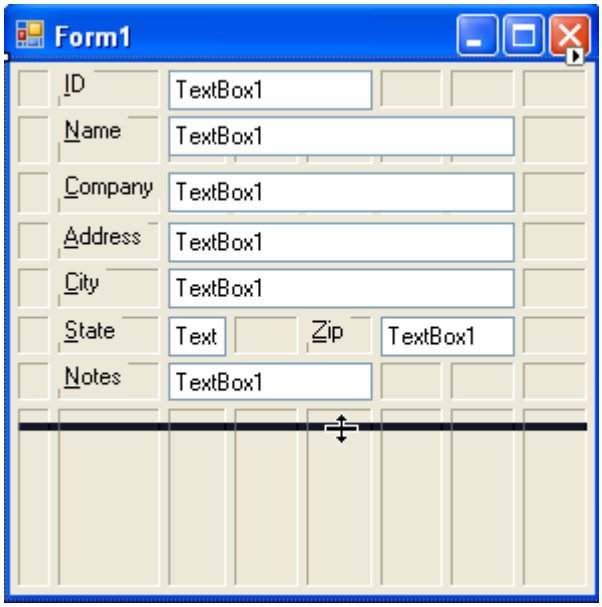
The grid is displayed at design time as a series of indented rectangles. Click **OK** to dismiss the editor, and add some controls to the form. Notice how the controls snap to the grid layout. You can resize any control and make it span any number of rows and columns.

4. To follow the tutorial, add eight text boxes and eight label controls, and arrange them so the form looks like the following:

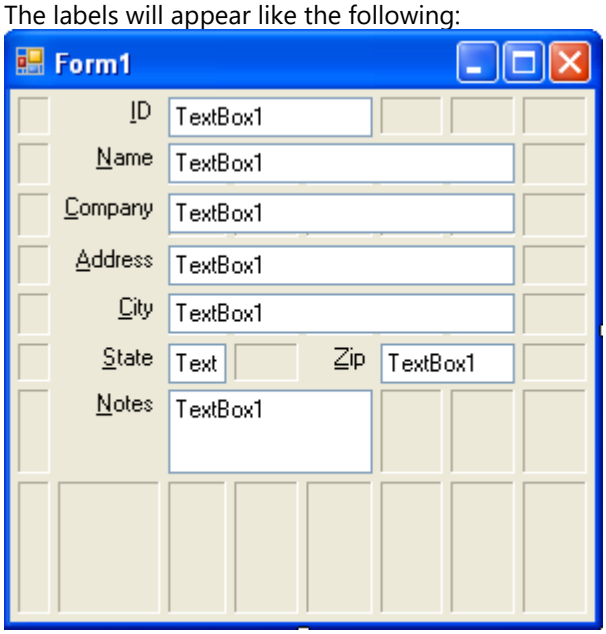


Note: Remember to set the **MultiLine** property of the textbox controls to True, or they will not resize vertically. Also, when you place a text box control on the form it automatically expands to three columns.

5. Select the cursor over the second column and drag it to the left so there is enough space for the labels, then select the cursor over each row and drag it upward to make the height of each row smaller. The following picture illustrates this effect:

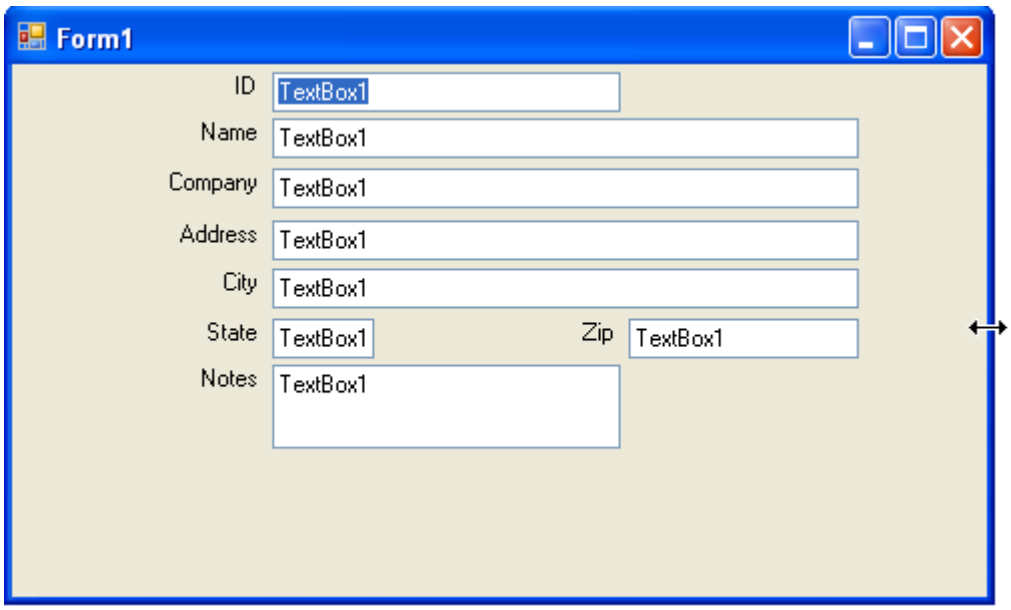


6. Change each label's **AutoSize** property to **False**, then change each label's **TextAlign** property to **TopRight**. The labels will appear like the following:



If you think the controls are too close to each other or too far apart, change the C1Sizer.SplitterWidth and C1Sizer.BorderWidth properties to achieve the effect you want.

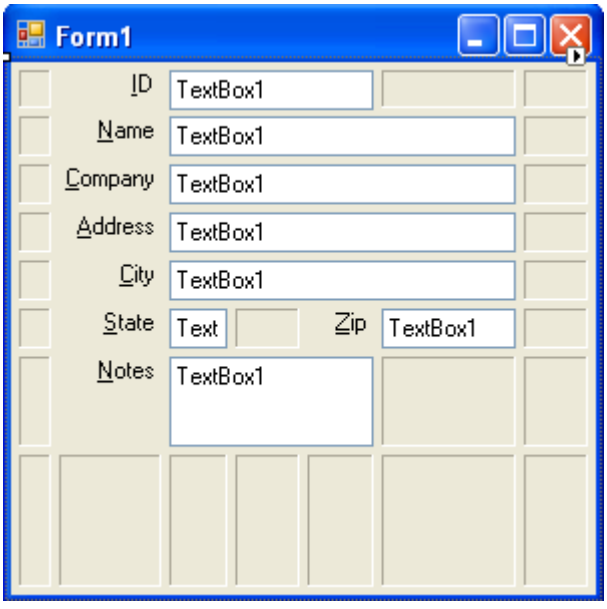
If you run the project now, you will see that when you resize the form, the text boxes and labels are automatically resized.



Cleaning up the grid

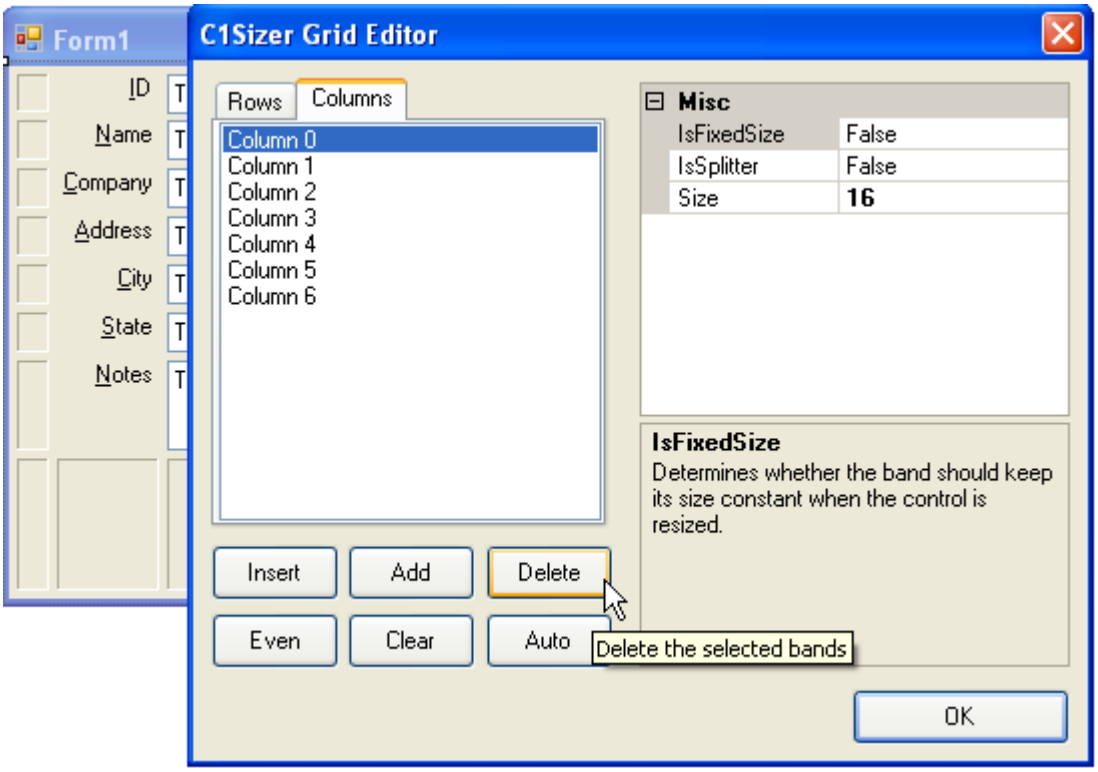
Right now the grid has eight rows and eight columns. Most of these bands are being used (controls are attached to them), but not all. To clean up the layout and get rid of unused bands, right click the **C1Sizer** control and select the **AutoGrid** option from the menu. This will clear any bands that have no controls attached to them.

The **C1Sizer** appears like the following:

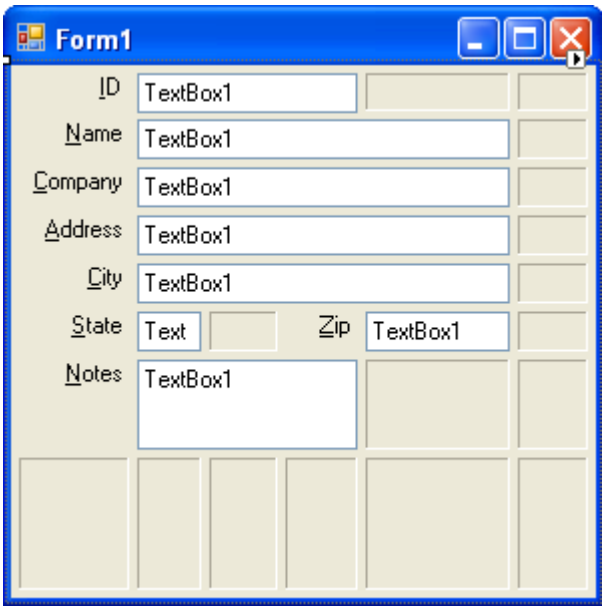


Deleting columns and rows

To remove the first column, **Column 0**, from the **C1Sizer Grid Editor**, right-click the **C1Sizer** control and select the **Edit Grid** option from the menu. The **C1Sizer Grid Editor** appears. Select the **Columns** tab and click on the **Delete** button and click **OK**.

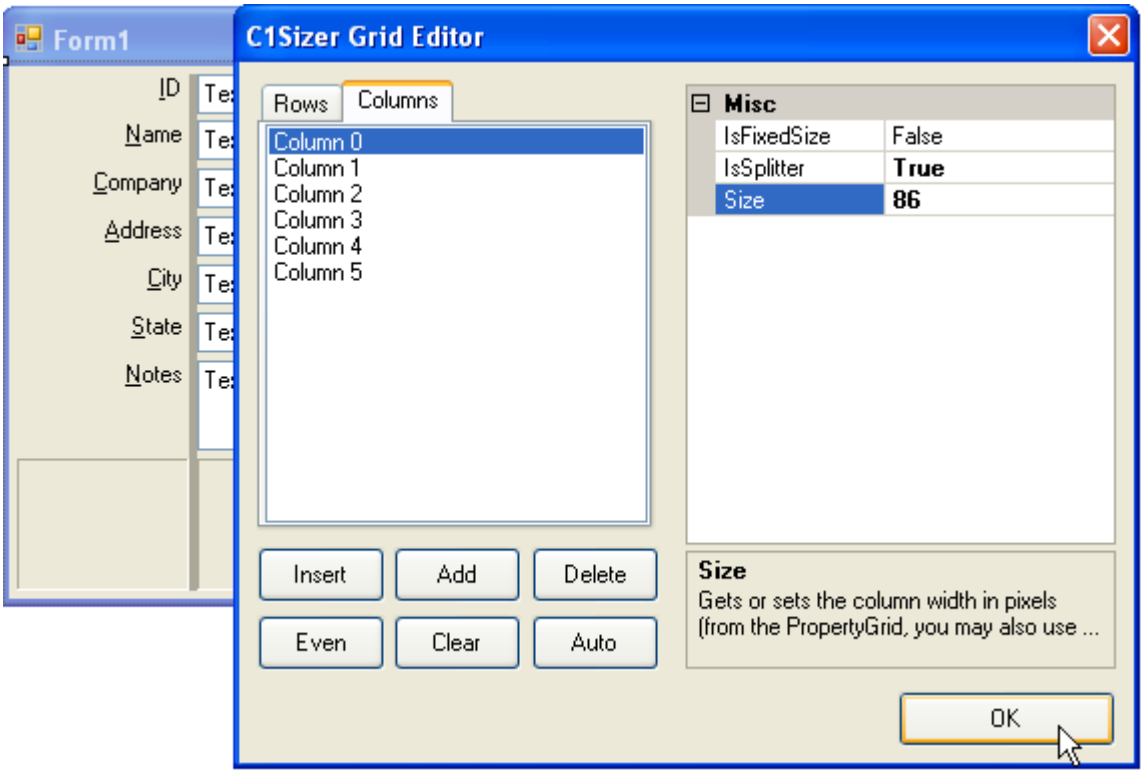


The **C1Sizer** grid appears like the following image at design time.



Setting up a splitter

Each band (row or column) on the grid can be configured to act as a splitter. For example, if you wanted to allow users to control the size of the label area on the left, you would bring up the grid editor, select the first column and set the Band.IsSplitter property to **True**, as shown below:



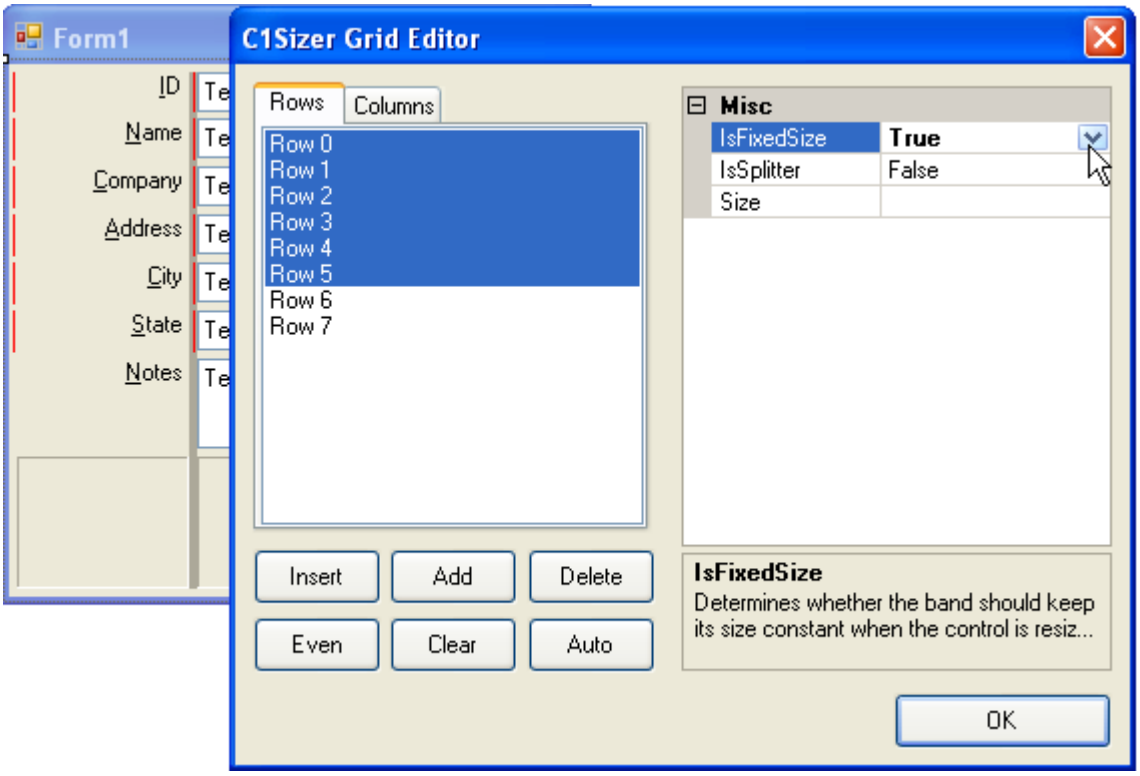
Note: A splitter can also be added by right-clicking the band and selecting **Splitter** from the context menu.

Notice the dark gray bar to the right of the first column. This indicates that the user will be able to drag that splitter at run time, automatically resizing the labels and text boxes on the first two columns of the grid.

Setting up a fixed-size band

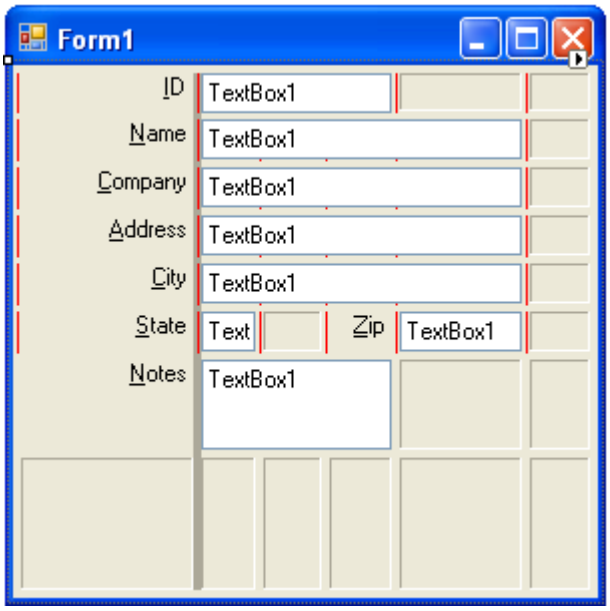
When the control is resized, the grid layout is adjusted to keep the proportion between the bands. You can designate certain bands as fixed size, and they will not be resized.

For example, we might want to allow only the bottom text box to be resized vertically, and keep the height of all others constant. To do this, bring up the grid editor, select the first six rows and set the Band.IsFixedSize property to **True**.



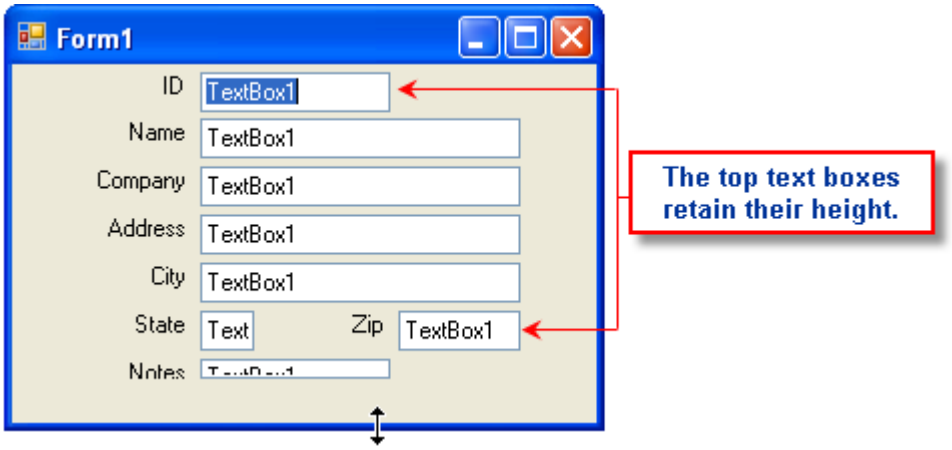
Note: A fixed-sized band can also be configured by right-clicking the band and selecting **Fixed Size** from the context menu.

Click the **OK** button. The **C1Sizer** appears with red markers on the first six rows. This indicates that those rows will not be resized with the form (the last two will).



Run the program and observe the following:

Use the splitter to adjust the size of the labels and text boxes, then resize the form and notice how the top text boxes retain their height.

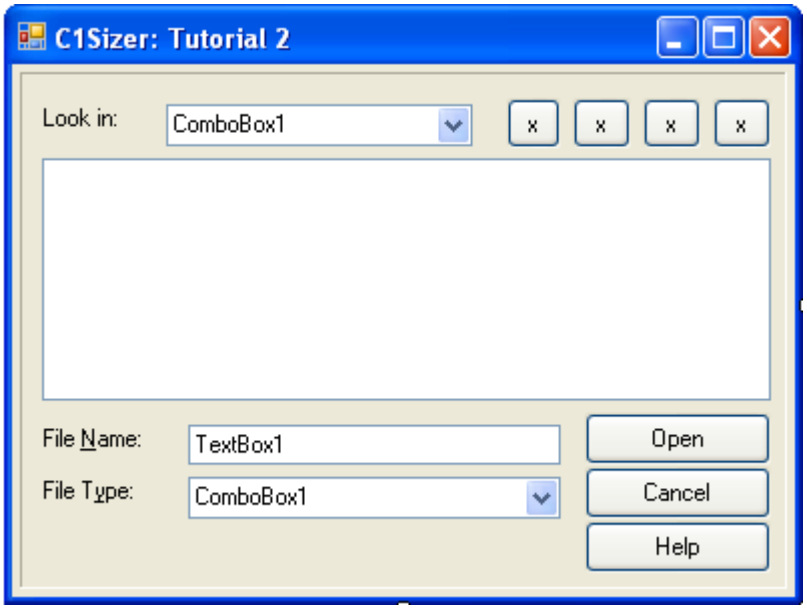


This concludes Tutorial 1. The following tutorial provides more detail about adding controls to the form and setting up the C1Sizer grid.

Tutorial 2: Add the controls, then set up the grid

In the previous tutorial, we started with the grid and then added controls. You may prefer to work the other way around. Forget about the grid initially, and design your form the normal way. Then, when you are done, let **C1Sizer** create the grid for you automatically. (Of course, once the grid is created you may still modify it with the design time commands we used earlier to tweak the grid's appearance and behavior.)

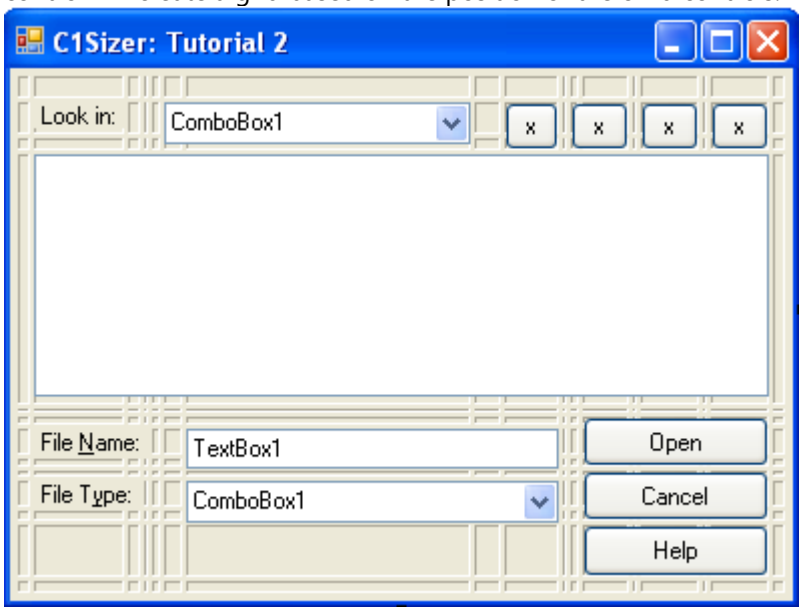
1. Open Visual Studio and create a new project, then add a **C1Sizer** control to the form and set its **Dock** property to **DockStyle.Fill** so it takes up the whole form.
2. Now add the following controls so the form looks like a FileOpen dialog box:
   - 3 Labels
   - 2 ComboBoxes
   - 1 TextBox
   - 7 Buttons
   - 1 ListBox



   You can use all the layout commands available in Visual Studio to set up this new form. There is nothing new so far.
3. Next, change the C1Sizer.SplitterWidth property on the **C1Sizer** control to a small value, one or two pixels. This is recommended because the form has some buttons that are very close together, and we want them to stay

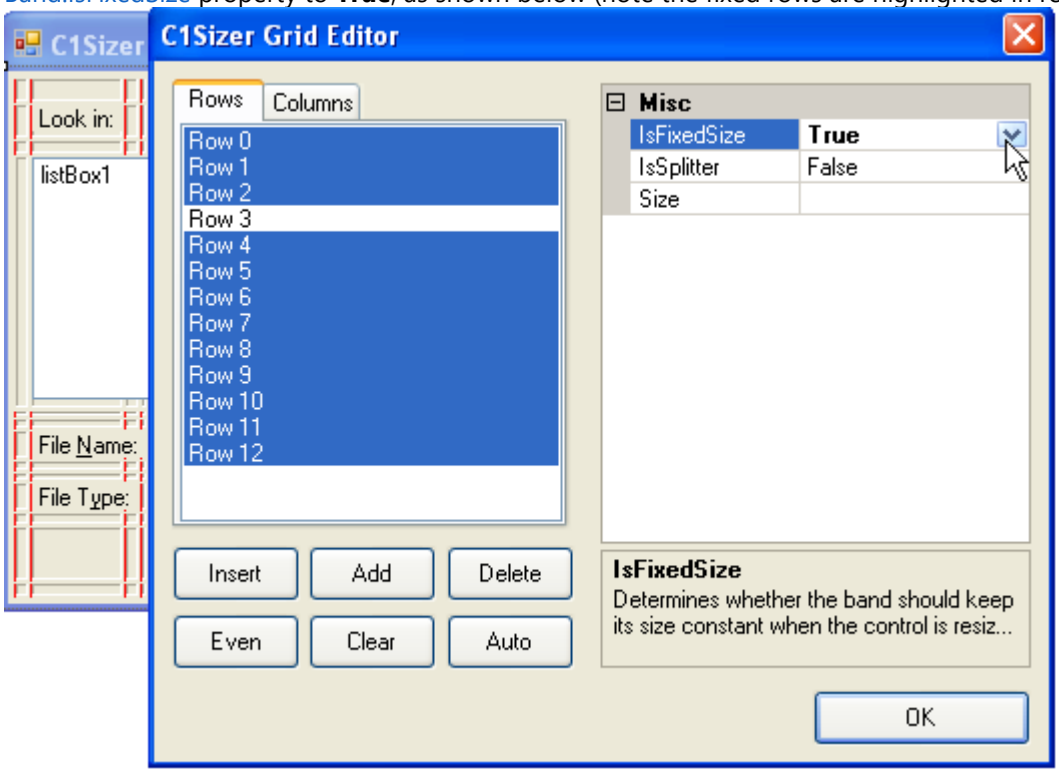
that way when we create the grid.

4. To create the grid, right-click the **C1Sizer** control and select the **AutoGrid** option from the context menu. The control will create a grid based on the position of the child controls. The grid will look like this:



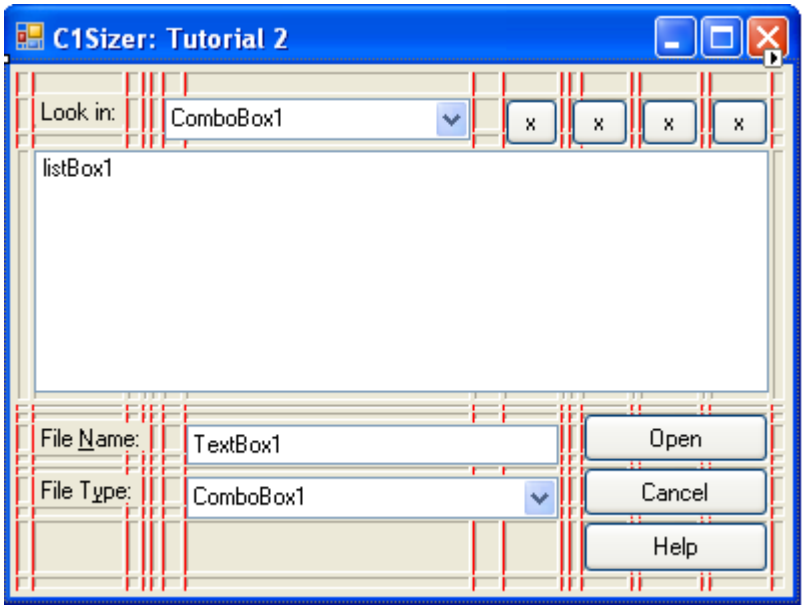
If you run the project now, you will see that when you resize the form, the contained controls are resized proportionally.

5. In this case, however, you probably do not want the buttons and text boxes stretched vertically, only the list box. In this case, bring up the grid editor, select all rows except the one that contains the ListBox, and set the Band.IsFixedSize property to **True**, as shown below (note the fixed rows are highlighted in red in the designer):



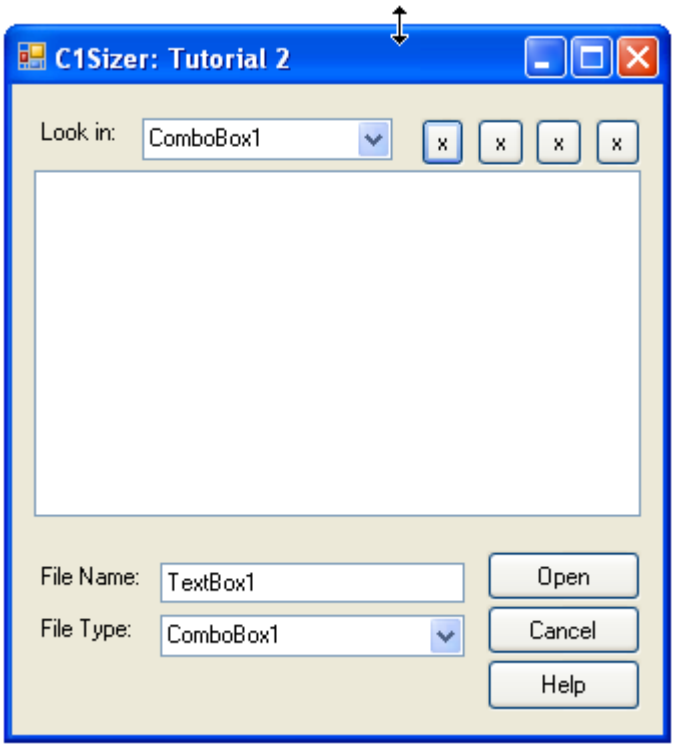
6. Select **OK**.

The designer highlights the fixed rows in red.



Run the program and observe the following:

Resize the form to see how the controls snap to the grid and adjust the layout automatically.



Sizer for WinForms Samples

Please be advised that this ComponentOne software tool is accompanied by various sample projects and/or demos which may make use of other development tools included with ComponentOne Studio.

Please refer to the pre-installed product samples through the following path:

Documents\ComponentOne Samples\WinForms

The following examples are included for the **C1Sizer** control:

Visual Basic Samples

C1Sizer includes the following Visual Basic samples:

Sample	Description
SizerTutorial1	Demonstrates how to set up the grid and add controls. This sample uses the **C1Sizer** control.
SizerTutorial2	Demonstrates how to add controls and set up the grid. This sample uses the **C1Sizer** control.

C# Samples

C1Sizer includes the following C# samples:

Sample	Description
AddControls	Shows how to add child controls to a **C1Sizer** using code. This sample uses the **C1Sizer** control.
CustomSplitters	Use the **OnPaint** event to customize the appearance of the splitter bars. This sample uses the **C1Sizer** control.
FakeOutlook	Create a user interface that looks like OutlookExpress. This sample uses the **C1Sizer** control.
FakeStudio	Create a user interface that looks like Visual Studio. This sample uses the **C1Sizer** control.
FindCell	Shows how to determine which cell is at a given point. This sample uses the **C1Sizer** control.
FindControl	Shows how to determine which control is at a given grid cell. This sample uses the **C1Sizer** control.
Light_MDI	Use the **C1SizerLight** with MDI forms. This sample uses the **C1SizerLight** component.
Light_Nested	Use **C1SizerLight** in forms with docked and nested controls. This sample uses the **C1SizerLight** component.
Light_Runtime	Control the **C1SizerLight** at run time. This sample uses the **C1SizerLight** component.
Light_Toolbar	Automatically resize controls with a wrapping toolbar. The samples shows how the **C1SizerLight** component works when the form has a wrapping toolbar docked to the top. This sample uses the **C1SizerLight** component.
RowHeaders	Demonstrates how to combine the .NET **Dock** property and panel controls to provide areas with headers. This sample uses the **C1Sizer** control.
SizerDesigner	Shows how to display the **C1Sizer** grid and implement the drag-and-drop operations between grid cells. This sample uses the **C1Sizer** control.

Sizer for WinForms Task-Based Help

The task-based help section assumes that you are familiar with programming in the Visual Studio environment, and know how to use the C1Sizer and C1SizerLight controls/components in general. If you are a novice to the **C1Sizer for .NET** product, please see Using the C1Sizer Control first.

Each topic provides a solution for specific tasks using the Sizer for .NET product.

Each task-based help topic also assumes that you have created a new .NET project. For additional information on this topic, see Creating a .NET 2.0 Project.

Add a C1SizerLight Component to a Form in code

To programmatically add a C1SizerLight component to a form, use the following code:

To write code in Visual Basic

Visual Basic

```
If Me.components Is Nothing Then
Me.components = New System.ComponentModel.Container
End If
Dim szl As C1SizerLight = New C1SizerLight(components)
szl.SetAutoResize(Me, True)
```

To write code in C#

C#

```
if (this.components == null)
this.components = new System.ComponentModel.Container();
C1SizerLight szl = new C1SizerLight(components);
szl.SetAutoResize(this, true);
```

To programmatically create a generic form start-up procedure which every form will call, use the following code:

To write code in Visual Basic

Visual Basic

```
Dim c As Container = New System.ComponentModel.Container
     For Each f As Form In myFormList
     Dim szl As C1SizerLight = New C1SizerLight(c)
     szl.SetAutoResize(f, True)
Next
```

To write code in C#

C#

```
Container c = new System.ComponentModel.Container();
    foreach (Form f in myFormList)
    {
        C1SizerLight szl = new C1SizerLight(c);
        szl.SetAutoResize(f, true);
```

```
    }
```

Position Controls on the C1Sizer Grid at Run Time

To position controls on the C1Sizer grid at run time, simply move the control to the area where you want it to be located. The C1Sizer will snap the control to the nearest position on the grid, and will automatically resize the control when the form is resized. The control can be positioned over a single grid cell or it may span multiple cells.

1. Add the C1Sizer control to your form.
2. Right-click on the C1Sizer control and select **Edit Grid**.
3. In the **C1Sizer Grid Editor** add three rows and three columns and then click **OK**.

To use the **Band.Bounds** property to determine the position of a grid cell, and the **Control.Bounds** property to move the control, use the following code:

To write code in Visual Basic

Visual Basic

```
 Private Sub Form1_Load(ByVal sender As Object, ByVal e As System.EventArgs)
  Dim b1 As Button = New Button
  c1Sizer1.Controls.Add(b1)
  b1.Visible = True
  b1.Bounds = GetCellRectangle(0, 0)
  Dim rc As Rectangle = GetCellRectangle(1, 1)
  rc = Rectangle.Union(rc, GetCellRectangle(2, 2))
  Dim b2 As Button = New Button
  c1Sizer1.Controls.Add(b2)
  b2.Visible = True
  b2.Bounds = rc
 End Sub

Private Function GetCellRectangle(ByVal row As Integer, ByVal col As Integer) As
Rectangle
   Return Rectangle.Intersect(c1Sizer1.Grid.Rows(row).Bounds,
c1Sizer1.Grid.Columns(col).Bounds)
End Function
```

To write code in C#

C#

```
 private void Form1_Load(object sender, System.EventArgs e)
 {
     // Create a button and position it over cell (0,0)
     Button b1 = new Button();
     c1Sizer1.Controls.Add(b1);
     b1.Visible = true;
     b1.Bounds = GetCellRectangle(0,0);
    // Calculate position of range (1,1)-(2,2)
     Rectangle rc = GetCellRectangle(1,1);
     rc = Rectangle.Union(rc, GetCellRectangle(2,2));
    // Create a button and position it over cell range (1,1)-(2,2)
     Button b2 = new Button();
```

```
    c1Sizer1.Controls.Add(b2);
    b2.Visible = true;
    b2.Bounds = rc;
}
private Rectangle GetCellRectangle(int row, int col)
{
        return Rectangle.Intersect(
        c1Sizer1.Grid.Rows[row].Bounds,
        c1Sizer1.Grid.Columns[col].Bounds);
}
```

Create a Three Dimensional Border for the Rows and Columns

The following steps show you how to create a styled border for the rows and columns of the C1Sizer control by using the **C1Sizer.Paint** event to repaint the control's rows and columns and the **DrawBorder3D** method to draw an etched three dimensional border around the rows and columns of the C1Sizer control:

1. Add the C1Sizer control to your form and set its **Dock** property to **DockStyle.Fill** so it takes up the whole form.
2. Add some controls to **C1Sizer**'s panel, for example two button controls.
3. Right-click the panel and select **Auto Grid** to create and activate the grid layout.
4. Add the following code to the **C1Sizer1_Paint** event to repaint its rows and columns with a three dimensional border style:

To write code in Visual Basic

Visual Basic

```
Private Sub C1Sizer1_Paint(ByVal sender As Object, ByVal e As PaintEventArgs)
 For Each row As Band In Me.c1Sizer1.Grid.Rows
   Dim rcrow As Rectangle = row.Bounds
   For Each col As Band In Me.c1Sizer1.Grid.Columns
     Dim rccol As Rectangle = col.Bounds
     Dim rccel As Rectangle = Rectangle.Intersect(rcrow, rccol)
     ControlPaint.DrawBorder3D(e.Graphics, rccel, Border3DStyle.Etched)
   Next
 Next
End Sub
```

To write code in C#

C#

```
        private void c1Sizer1_Paint(object sender, PaintEventArgs e)
        {
           // Paint sizer grid
            foreach (Band row in this.c1Sizer1.Grid.Rows)
            {
                Rectangle rcrow = row.Bounds;
                foreach (Band col in this.c1Sizer1.Grid.Columns)
                {
                    Rectangle rccol = col.Bounds;
```

```
                    Rectangle rccel = Rectangle.Intersect(rcrow, rccol);
                    ControlPaint.DrawBorder3D(e.Graphics, rccel,
                        Border3DStyle.Etched);
                }
            }
        }
```

Store Layout Information for the C1Sizer Control

To store layout information for the C1Sizer control, use the GridDefinition property that gets or sets a string containing the grid information. This property is non-browsable, but can be used as any other property.

This example assumes that you have imported the C1.Win.C1Sizer and System.Collections namespaces to your source code. To implement two buttons that save and restore the grid definition and control positions, use the following code:

To write code in Visual Basic

Visual Basic

```
Private _gridDefinition As String
Private _ctlPositions As ArrayList
Private Sub btnSaveGrid_Click(ByVal sender As Object, ByVal e As System.EventArgs)
 _gridDefinition = C1Sizer1.GridDefinition
 _ctlPositions = New ArrayList
 For Each ctl As Control In C1Sizer1.Controls
   _ctlPositions.Add(ctl.Bounds)
 Next
End Sub
Private Sub btnRestoreGrid_Click(ByVal sender As Object, ByVal e As System.EventArgs)
 If Not (_gridDefinition Is Nothing) Then
   c1Sizer1.GridDefinition = _gridDefinition
   Dim i As Integer = 0
   While i < c1Sizer1.Controls.Count
    c1Sizer1.Controls(i).Bounds = CType(_ctlPositions(i), Rectangle)
    System.Math.Min(System.Threading.Interlocked.Increment(i),i-1)
   End While
 End If
End Sub
```

To write code in C#

C#

```
        string _gridDefinition;
        ArrayList _ctlPositions;
        private void btnSaveGrid_Click(object sender, System.EventArgs e)
        {
            // Save grid definition
            _gridDefinition = c1Sizer1.GridDefinition;
            // Save control positions
            _ctlPositions = new ArrayList();
```

```
        foreach (Control ctl in c1Sizer1.Controls)
        {
            _ctlPositions.Add(ctl.Bounds);
        }
    }
    private void btnRestoreGrid_Click(object sender, System.EventArgs e)
    {
        if (_gridDefinition != null)
        {
            // Restore grid definition
            c1Sizer1.GridDefinition = _gridDefinition;
            // Restore control positions
            for (int i = 0; i < c1Sizer1.Controls.Count; i++)
            {
                c1Sizer1.Controls[i].Bounds = (Rectangle)_ctlPositions[i];
            }
        }
    }
```