
ComponentOne

Accordion for WPF

Copyright © 1987-2012 GrapeCity, Inc. All rights reserved.

ComponentOne, a division of GrapeCity

201 South Highland Avenue, Third Floor

Pittsburgh, PA 15206 • USA

Internet: info@ComponentOne.com

Web site: <http://www.componentone.com>

Sales

E-mail: 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 original CD (or diskettes) are 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 CD (or disk) 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 a defective CD (or disk) by sending it and a check for \$25 (to cover postage and handling) to ComponentOne.

Except for the express warranty of the original CD (or disks) 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. We are 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.

This manual was produced using [ComponentOne Doc-To-Help™](#).

Table of Contents

ComponentOne Accordion for WPF	1
Help with ComponentOne Studio for WPF	1
Key Features	3
Accordion for WPF Quick Start	3
Step 1 of 3: Creating an Application with a C1Accordion Control	3
Step 2 of 3: Adding Accordion Panes.....	4
Step 3 of 3: Running the Project.....	4
Using C1Accordion.....	6
C1Accordion Elements	7
Accordion Pane Header	7
Accordion Pane Content Area	8
Expanding and Collapsing Accordion Panes.....	8
Accordion for WPF Layout and Appearance.....	11
ComponentOne ClearStyle Technology	12
How ClearStyle Works.....	12
C1Accordion and C1AccordionItem ClearStyle Properties.....	12
Accordion for WPF Appearance Properties	13
Text Properties	13
Content Positioning Properties.....	14
Color Properties.....	14
Border Properties.....	15
Size Properties	15
Templates.....	15
C1Accordion Themes	16
Accordion for WPF Samples.....	20
Accordion for WPF Task-Based Help	23
Adding Accordion Panes to the C1Accordion Control	23
Adding Content to Header Elements.....	24
Adding Text to the Header.....	24

Adding a Control to the Header	24
Adding Content to Content Areas	25
Adding Text to the Content Area	26
Adding a Control to the Content Area	26
Adding Multiple Controls to the Content Area.....	28
Changing the Expand Direction	28
Filling Out the Accordion's Height	29

ComponentOne Accordion for WPF

Display a list of expandable items with **ComponentOne Accordion™ for WPF**. Select an item to expand it and collapse all others, automatically organizing your UI and optimizing the use of screen real estate.



Getting Started

- [Using C1Accordion](#) (page 6)
- [Quick Start](#) (page 3)
- [Task-Based Help](#) (page 23)

Help with ComponentOne Studio for WPF

Getting Started

For information on installing ComponentOne Studio for WPF, licensing, technical support, namespaces and creating a project with the control, please visit [Getting Started with Studio for WPF](#).

What's New

For a list of the latest features added to **ComponentOne Studio for WPF**, visit [What's New in Studio for WPF](#).

Key Features

ComponentOne Accordion for WPF allows you to create customized, rich applications. Make the most of **Accordion for WPF** by taking advantage of the following key features:

- **Expand Direction**

The **C1Accordion** control has the ability to expand in four different directions. The **ExpandDirection** property indicates which direction the control expands and can be set to **Top**, **Right**, **Bottom**, or **Left**. For more information, see the **ExpandDirection** topic.

- **Custom Header**

An accordion pane's header can be customized with both text and controls. For more information on the customizable header element, see [Accordion Pane Header](#) (page 7).

- **Configure Items in an Organized Pattern**

Accordion is designed to maximize space. Configure the size and position of **C1Accordion** to hide items until needed.

- **Add Objects of any Data Type**

Because the **C1Accordion** control inherits from **ItemsControl**, you can add objects of any data type to its **Items** collection and use a **DataTemplate** to create a visual representation of the items.

Accordion for WPF Quick Start

The following quick start guide is intended to get you up and running with **Accordion for WPF**. In this quick start, you'll start in Visual Studio to create a new project with a **C1Accordion** control. You will also customize the accordion, add accordion panes filled with content to it, and then observe some of the run-time features of the control.

Step 1 of 3: Creating an Application with a C1Accordion Control

In this step, you'll begin in Visual Studio to create a WPF application using **Accordion for WPF**.

Complete the following steps:

1. In Visual Studio 2008, select **File | New | Project**.
2. In the **New Project** dialog box, select a language in the left pane, and in the templates list select **WPF Application**.
3. Enter a **Name** for your project and click **OK**.
4. Navigate to the Toolbox and double-click the **C1Accordion** icon to add the control to the project.
5. Right click the **C1Accordion** control to open its context menu and then select **Properties**.

The **Properties** window opens with the **C1Accordion** control's properties in focus.

6. Set the following properties.
 - Set the **Height** property to "250" to set the height of the control.
 - Set the **Width** property to "400" to set the width of the control.
 - Set the **ExpandDirection** property to **Left** so that the **C1Accordion** control will expand from the bottom rather than expanding from the top, which is its default.

- Set the **Fill** property to **True** by selecting the **Fill** check box. This means that each pane will expand to fill the specified width of the **C1Accordion** control.
- Set **AllowCollapseAll** to **False** by clearing the **AllowCollapseAll** check box. This will prevent users from collapsing all panes at the same time.

You've successfully created a WPF application containing a **C1Accordion** control. In the next step, you will customize the appearance and behavior of the **C1Accordion** control.

Step 2 of 3: Adding Accordion Panes

In the last step, you created a project with a customized **C1Accordion** control. In the next step, you will add accordion panes, which you will customize and add content to.

Complete the following steps:

1. Click the **C1Accordion** control once to select it.
2. In the **Properties** window, click the **Items** ellipsis button.
The **Collection Editor: Items** dialog box opens.
3. Click the **Add** button three times to add three **C1AccordionItem** items to the **C1Accordion** control.
4. Select the first **C1AccordionItem** and set the following properties:
 - Set the **Background** property to **Aqua** to set the background color of the accordion pane.
 - Set the **Content** property to "This is text content" to add text content to the accordion pane.
5. Select the second **C1AccordionItem** and set the following properties:
 - Set the **Background** property to **AliceBlue** to set the background color of the accordion panel.
 - Set the **IsExpanded** property to **True** so that this pane will be expanded at run time.
6. Select the third **C1AccordionItem** and set the **Background** property to **LawnGreen** to set the background color of the accordion pane.
7. Click **OK** to close the **Collection Editor: Items** dialog box.
8. Switch to XAML view and complete the following:
 - Add `Header="Pane 1"` to the first `<c1:C1AccordionItem>` tag.
 - Add `Header="Pane 2"` to the second `<c1:C1AccordionItem>` tag.
 - Add `Header="Pane 3"` to the third `<c1:C1AccordionItem>` tag.
9. Switch to Design view and add a control to the second accordion pane by completing the following steps:
 - a. In the Designer, click the second accordion pane to select it.
 - b. Navigate to the Toolbox and double-click the **Calendar** icon to add the control to the accordion pane.

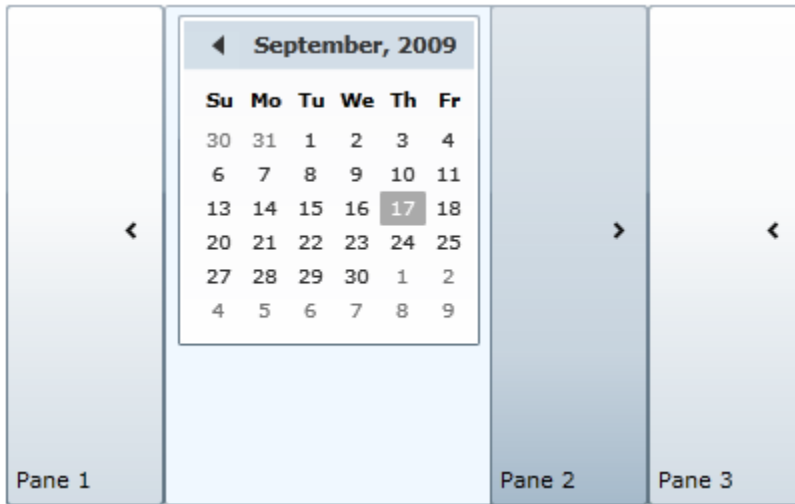
In this step, you added three accordion panes to the **C1Accordion** control and then added content to two of the accordion panes. In the next step, you will run the project and observe the run-time features of the control.

Step 3 of 3: Running the Project

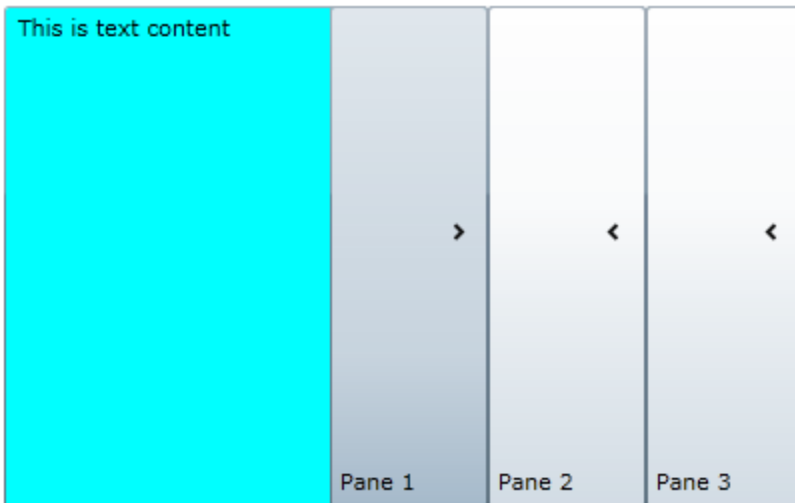
In the last step, you added accordion panes and content to the **C1Accordion** control. In this step, you will run the project and observe some of the run-time features of the **C1Accordion** control.

Complete the following steps:

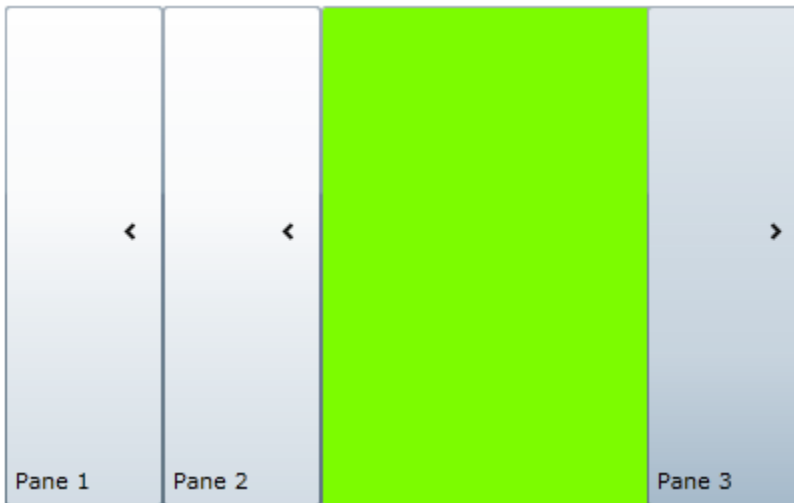
1. From the **Debug** menu, select **Start Debugging** to view how your application will appear at run time. Observe that the second pane, which holds the **Calendar** control, is expanded.



2. Click **Pane 1** and observe that the first pane expands to reveal its content.



3. Click **Pane 3** and observe that the last pane expands. Note that the third pane expands to the same width as the other panes despite the fact that it has no content.



4. Click **Pane 3** and observe that you can't close the pane. This is because you set the `AllowCollapseAll` property to **False**, which means that one accordion pane must be expanded at all times.

Congratulations! You have completed the **Accordion for WPF** quick start tutorial. In this tutorial, you created a WPF project containing a `C1Accordion` control, modified the appearance and behavior of the control, added accordion panes and accordion pane content to the control, and then observed some of the run-time features of the control.

Using C1Accordion

The `C1Accordion` control is a container that can hold a series of expandable and collapsible panes for storing text, images, and controls. The `C1Accordion` control is an **ItemsControl**, which means that the control is designed to host a series of objects. The `C1AccordionItem` class represents the items, or accordion panes, that can be hosted by the `C1Accordion` control.

When you add the `C1Accordion` control to a project, it exists as nothing more than a container. But once the control is added to your project, you can easily add multiple accordion panes to it in Design view, in XAML, or in code. The following image depicts a `C1Accordion` control with three accordion pane items, the first of which is expanded.



The image above shows the accordion panes sans header text or content, but customizing the header and adding content to the pane is as simple as setting a few properties. You can also modify behaviors, such as the expandability and the direction, of the control.

The following topics provide an overview of the `C1Accordion` control's elements and features.

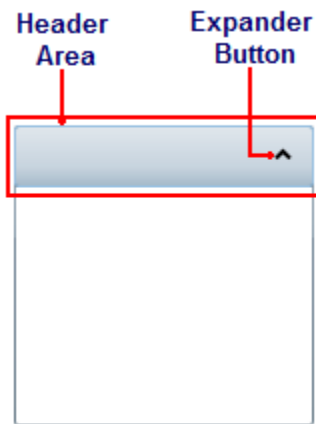
C1Accordion Elements

This section provides a visual and descriptive overview of the elements that comprise the C1Accordion control. The control is comprised of two elements – the header and the content area – which combine to make the complete C1Accordion control.

Accordion Pane Header

By default, the header element of an accordion pane appears at the top of the control and the expander button appears on the right side of the header. When the C1AccordionItem item (accordion pane) is first placed on the page, the header element contains no text.

The following image labels the header area of an accordion pane.



To add text to the header element, simply set the **Header** property to a string. Once the text is added, you can style it using several font properties (see [Text Properties](#) (page 13)). You can also add WPF controls to the header.

The placement of the header element and expander button will change depending on the expand direction of the control. For more information on expand directions, see the [Expand Direction](#) (page 9) topic.

Attribute Syntax versus Property Element Syntax

When you want to add something simple to the header, such as an unformatted string, you can simply use the common XML attributes in your XAML markup, such as in the following:

```
<c1ext:C1AccordionItem Header="Hello World"/>
```

However, there may be times where you want to add more complex elements, such as grids or panels, to the content area. In this case you would use property element syntax, such as in the following:

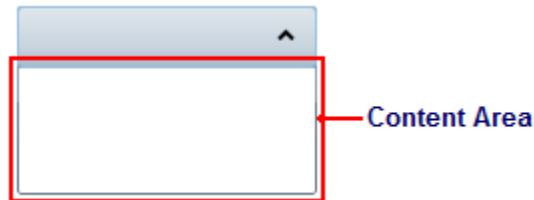
```
<c1ext:C1AccordionItem Width="150" Height="55" Name="C1AccordionItem1">
  <c1ext:C1AccordionItem.Header>
    <Grid HorizontalAlignment="Stretch">
      <Grid.ColumnDefinitions>
        <ColumnDefinition Width="Auto" />
        <ColumnDefinition Width="*" />
      </Grid.ColumnDefinitions>
      <TextBlock Text="C1AccordionItem" />
    </Grid>
  </c1ext:C1AccordionItem.Header>
</c1ext:C1AccordionItem>
```

```
</c1ext:C1AccordionItem.Header>  
</c1ext:C1AccordionItem>
```

Accordion Pane Content Area

An accordion pane's content area initially consists of an empty space. In the content area, you can add grids, text, images, and arbitrary controls. When working in Blend or Visual Studio's Design view, elements in the content area of the control can be added and moved on the control through a simple drag-and-drop operation.

The following image labels the content area of an accordion pane.



You can add text to the content area by setting the item's **Content** property or by adding a **TextBox** element to the content area. Adding WPF elements to the content area at run time is simple: You can either use simple drag-and-drop operations or XAML in Visual Studio or Blend. If you'd prefer to add a control at run time, you can use C# or Visual Basic code.

A `C1AccordionItem` item can only accept one child element at a time. However, you can circumvent this issue by adding a panel-based control as its child element. Panel-based controls, such as a **StackPanel** control, are able to hold multiple elements. The panel-based control meets the one control limitation of the `C1AccordionItem` item, but its ability to hold multiple elements will allow you to show several controls in the content area of the accordion pane.

Attribute Syntax versus Property Element Syntax

When you want to add something simple to the content area, such as an unformatted string or a single control, you can simply use the common XML attributes in your XAML markup, such as in the following:

```
<c1ext:C1AccordionItem Content="Hello World"/>
```

However, there may be times where you want to add more complex elements, such as grids or panels, to the content area. In this case you can use property element syntax, such as in the following:

```
<c1ext:C1AccordionItem Width="150" Height="55" Name="C1AccordionItem1">  
  <c1ext:C1AccordionItem.Content>  
    <StackPanel>  
      <TextBlock Text="Hello"/>  
      <TextBlock Text="World"/>  
    </StackPanel>  
  </c1ext:C1AccordionItem.Content>  
</c1ext:C1AccordionItem>
```

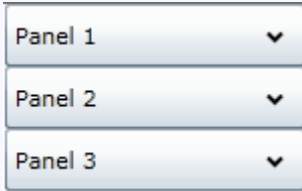
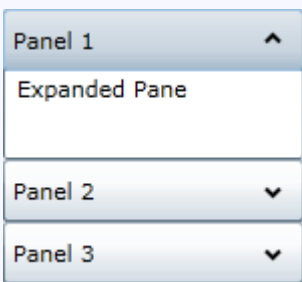
Expanding and Collapsing Accordion Panes

This section details the options for customizing the way that the accordion panes expand and collapse.

Accordion Pane Initial Expand State

By default, the **IsExpanded** property of each accordion pane is set to **False**, which means that the pane appears in its collapsed state when the page is loaded. If you want the pane to be expanded upon page load, you can set the **IsExpanded** property to **True**.

The following table illustrates the difference between the two expand states.

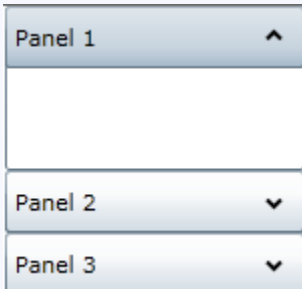
IsExpanded	Result
IsExpanded=False	
IsExpanded=True	

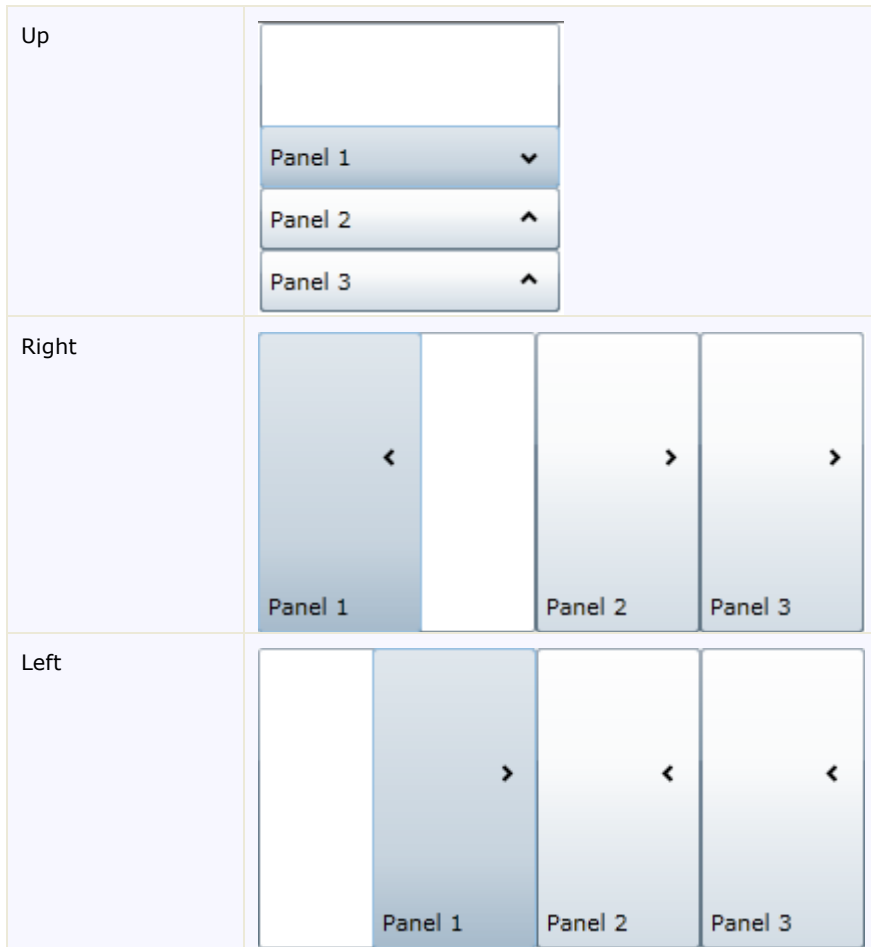
You can set the expand states in Design view, in XAML, or in code.

Expand Direction

The `CIAccordion` control includes the option to specify the expand direction using the `ExpandDirection` property. In addition to setting the direction that the control expands, changing the `ExpandDirection` also changes the header's orientation to the content area of the control. By default the `ExpandDirection` property is set to **Down** and the control expands from top to bottom.

The following table illustrates each `ExpandDirection` setting.

ExpandDirection	Result
Down	



You can set the collapsing and expanding direction in Design view, in XAML, or in code.

Collapsing Panes

By default, the `C1Accordion` control's `AllowCollapseAll` property is set to **True**, meaning that all panes of an accordion can be collapsed by the user. This is useful in cases where you want to conserve screen real estate so that the user is free of unnecessary distractions. Think of it like the musical instrument accordion, which can be tightly compressed for storage and then decompressed when a player wants to use it.

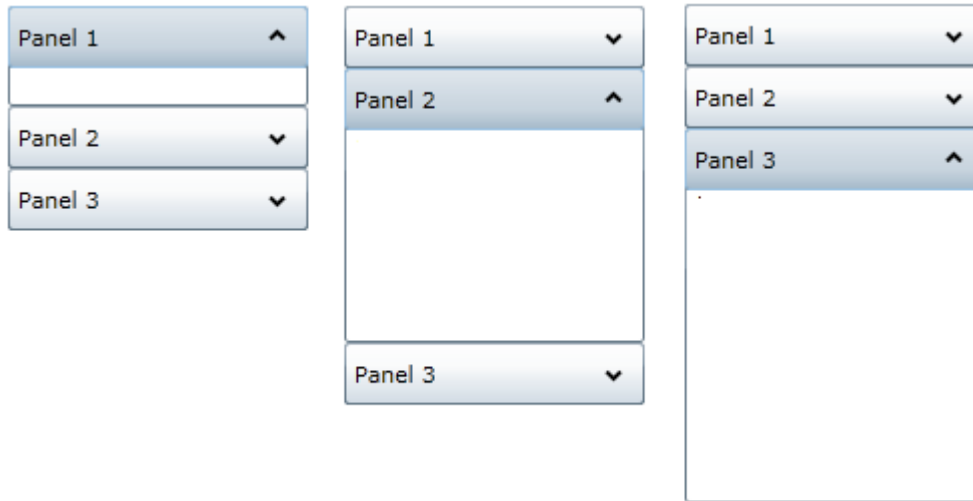
But there are times when this won't fit into the design of your user interface. For example, you may want to use the accordion as a menu element. In that case, you'd want the `C1Accordion` control to fill to a specific height or width at all times, and the control won't fulfill that need if all of its panes can be collapsed. Therefore, you would set the `AllowCollapseAll` property to **False** so that one pane has to remain open at all times. You would then set the height or width property as desired and then set the `Fill` property to **True** so that the `C1Accordion` control and its panes always fill to your specified height or width.

Note: The `C1Accordion` control is only capable of expanding one pane at a time.

Expansion Fill

The default behavior of an accordion pane (`C1AccordionItem`) is to fill only to the height (for accordions that expand up or down) or width (for accordions that expand right or left) of an accordion pane's contents. This will

cause the accordion to grow to different heights or widths depending on the height or width of the content for a given pane. This is illustrated in the example below.



To avoid this, set the height or width of the C1Accordion control and then set its Fill property to **True**. This will cause each panel to expand to fill the width or height that you specified, thus providing a uniformed width or height for accordion pane. You can observe the difference in the example below.



Accordion for WPF Layout and Appearance

The following topics detail how to customize the C1Accordion control's layout and appearance. You can use built-in layout options to lay your controls out in panels such as Grids or Canvases. Themes allow you to customize the appearance of the grid and take advantage of WPF's XAML-based styling. You can also use templates to format and layout the control and to customize the control's actions.

ComponentOne ClearStyle Technology

ComponentOne ClearStyle™ technology is a new, quick and easy approach to providing Silverlight and WPF control styling. ClearStyle allows you to create a custom style for a control without having to deal with the hassle of XAML templates and style resources.

Currently, to add a theme to all standard WPF controls, you must create a style resource template. In Microsoft Visual Studio, this process can be difficult; this is why Microsoft introduced Expression Blend to make the task a bit easier. Having to jump between two environments can be a bit challenging to developers who are not familiar with Blend or do not have the time to learn it. You could hire a designer, but that can complicate things when your designer and your developers are sharing XAML files.

That's where ClearStyle comes in. With ClearStyle the styling capabilities are brought to you in Visual Studio in the most intuitive manner possible. In most situations you just want to make simple styling changes to the controls in your application so this process should be simple. For example, if you just want to change the row color of your data grid this should be as simple as setting one property. You shouldn't have to create a full and complicated-looking template just to simply change a few colors.

How ClearStyle Works

Each key piece of the control's style is surfaced as a simple color property. This leads to a unique set of style properties for each control. For example, a **Gauge** has **PointerFill** and **PointerStroke** properties, whereas a **DataGrid** has **SelectedBrush** and **MouseOverBrush** for rows.

Let's say you have a control on your form that does not support ClearStyle. You can take the XAML resource created by ClearStyle and use it to help mold other controls on your form to match (such as grabbing exact colors). Or let's say you'd like to override part of a style set with ClearStyle (such as your own custom scrollbar). This is also possible because ClearStyle can be extended and you can override the style where desired.

ClearStyle is intended to be a solution to quick and easy style modification but you're still free to do it the old fashioned way with ComponentOne's controls to get the exact style needed. ClearStyle does not interfere with those less common situations where a full custom design is required.

C1Accordion and C1AccordionItem ClearStyle Properties

Accordion for WPF supports ComponentOne's new ClearStyle technology that allows you to easily change control colors without having to change control templates. By just setting a few color properties you can quickly style the entire grid.

The following table outlines the brush properties of the **C1Accordion** control:

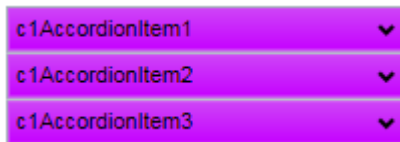
Brush	Description
Background	Gets or sets the brush of the control's background.
ExpandedBackground	Gets or sets the brush of the header background that appears when items of the C1Accordion control are expanded.
MouseOverBrush	Gets or sets the System.Windows.Media.Brush used to highlight the control's C1AccordionItems when it is moused over.
HeaderBackground	Gets or sets the background brush of all headers of all C1AccordionItem headers contained within the C1Accordion control.

The following table outlines the brush properties of the **C1AccordionItem** control:

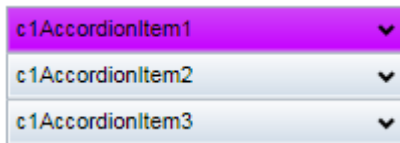
Brush	Description
-------	-------------

Background	Gets or sets the brush of the control's background.
ExpandedBackground	Gets or sets the brush of the header background when the C1AccordionItem is expanded.
MouseOverBrush	Gets or sets the System.Windows.Media.Brush used to highlight the control when it is moused over.
HeaderBackground	Gets or sets the background brush of the C1AccordionItem header.

You can completely change the appearance of the **C1Accordion** and **C1AccordionItem** controls by setting a few properties, such as the **C1Accordion** control's **HeaderBackground** property, which sets the background color for every accordion item in a **C1Accordion** control. For example, if you set the **C1Accordion** control's **HeaderBackground** property to "#FFC500FF", each header in the **C1Accordion** control would appear similar to the following:



You can also choose to modify the accordion items independently by setting a **C1AccordionItem** control's **HeaderBackground** property. For example, if you set a the **HeaderBackground** property of the first accordion item to "#FFC500FF", the control would appear similar to the following:



It's that simple with ComponentOne's ClearStyle technology. For more information on ClearStyle, see the [ComponentOne ClearStyle Technology](#) (page 12) topic.

Accordion for WPF Appearance Properties

ComponentOne Accordion for WPF includes several properties that allow you to customize the appearance of the control. You can change the appearance of the text displayed in the control and customize graphic elements of the control. The following topics describe some of these appearance properties.

Text Properties

The following properties let you customize the appearance of text in the accordion pane.

Property	Description
FontFamily	Gets or sets the font family of the control. This is a dependency property.
FontSize	Gets or sets the font size. This is a dependency property.

FontStretch	Gets or sets the degree to which a font is condensed or expanded on the screen. This is a dependency property.
FontStyle	Gets or sets the font style. This is a dependency property.
FontWeight	Gets or sets the weight or thickness of the specified font. This is a dependency property.
TextAlignment	Gets or sets how the text should be aligned in the accordion pane content area. This is a dependency property.
Header	Gets or sets the header of an accordion item.
HeaderFontFamily	Gets or sets the font family of the header.
HeaderFontStretch	Gets or sets the font stretch of the header.
HeaderFontStyle	Gets or sets the font style of the header.
HeaderFontWeight	Gets or sets the font weight of the header.

Content Positioning Properties

The following properties let you customize the position of header and content area content in the C1Accordion control.

Property	Description
HeaderPadding	Gets or sets the padding of the header.
HeaderHorizontalContentAlignment	HorizontalContentAlignment of the header.
HeaderVerticalContentAlignment	Gets or sets the vertical content alignment of the header.
HorizontalContentAlignment	Gets or sets the horizontal alignment of the control's content. This is a dependency property.
VerticalContentAlignment	Gets or sets the vertical alignment of the control's content. This is a dependency property.

Color Properties

The following properties let you customize the colors used in the control itself.

Property	Description
Background	Gets or sets a brush that describes the background of a control. This is a dependency property.
Foreground	Gets or sets a brush that describes the foreground color. This is a dependency property.
HeaderBackground	
HeaderForeground	Gets or sets the foreground brush of the

	header.
--	---------

Border Properties

The following properties let you customize the control's border.

Property	Description
BorderBrush	Gets or sets a brush that describes the border background of a control. This is a dependency property.
BorderThickness	Gets or sets the border thickness of a control. This is a dependency property.

Size Properties

The following properties let you customize the size of the **C1Accordion** control.

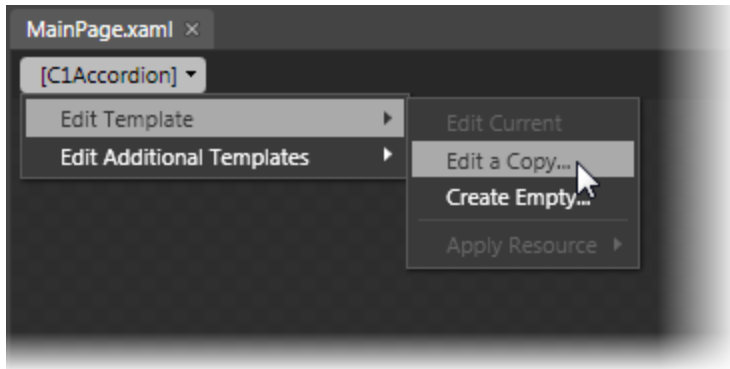
Property	Description
Height	Gets or sets the suggested height of the element. This is a dependency property.
MaxHeight	Gets or sets the maximum height constraint of the element. This is a dependency property.
MaxWidth	Gets or sets the maximum width constraint of the element. This is a dependency property.
MinHeight	Gets or sets the minimum height constraint of the element. This is a dependency property.
MinWidth	Gets or sets the minimum width constraint of the element. This is a dependency property.
Width	Gets or sets the width of the element. This is a dependency property.

Templates

One of the main advantages to using a WPF control is that controls are "lookless" with a fully customizable user interface. Just as you design your own user interface (UI), or look and feel, for WPF applications, you can provide your own UI for data managed by **ComponentOne Accordion for WPF**. Extensible Application Markup Language (XAML; pronounced "Zammel"), an XML-based declarative language, offers a simple approach to designing your UI without having to write code.

Accessing Templates

You can access templates in Microsoft Expression Blend by selecting the C1Accordion control and, in the menu, selecting **Edit Template**. Select **Edit a Copy** to create an editable copy of the current template or select **Create Empty** to create a new blank template.



If you want to edit the C1AccordionItem template, simply select the C1AccordionItem control and, in the menu, select **Edit Template**. Select **Edit a Copy** to create an editable copy of the current template or **Create Empty**, to create a new blank template.

Note: If you create a new template through the menu, the template will automatically be linked to that template's property. If you manually create a template in XAML you will have to link the appropriate template property to the template you've created.

Note that you can use the [Template](#) property to customize the template.

C1Accordion Themes

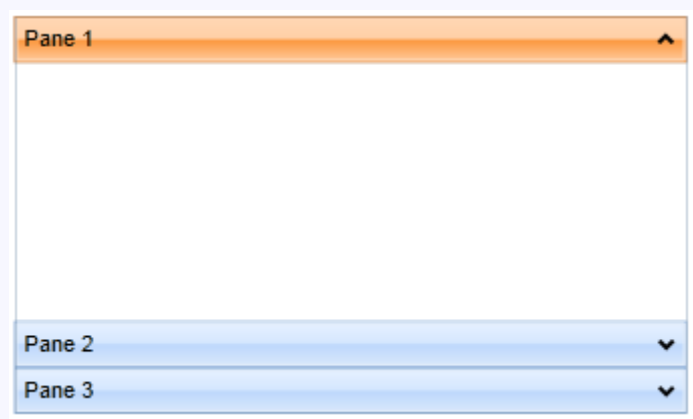
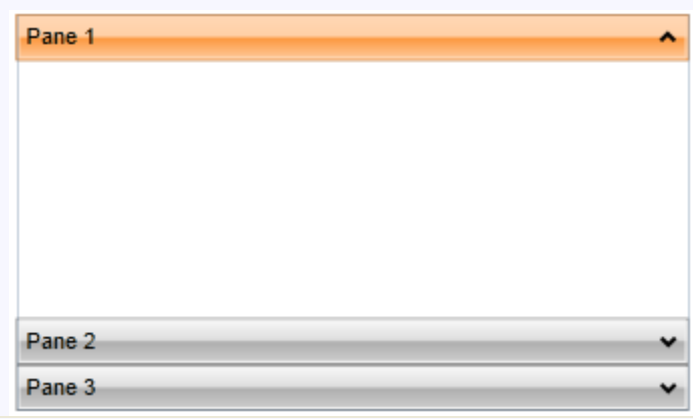
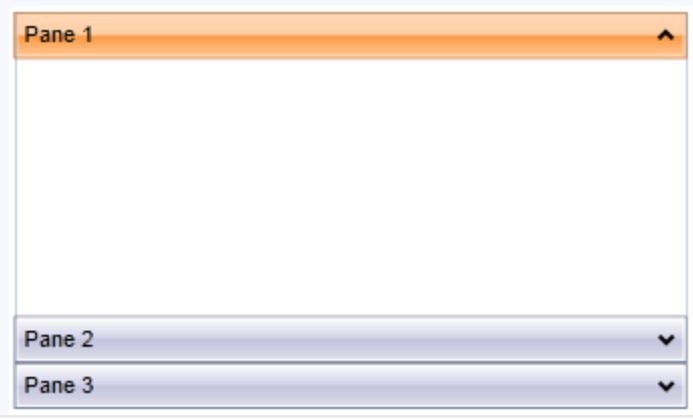
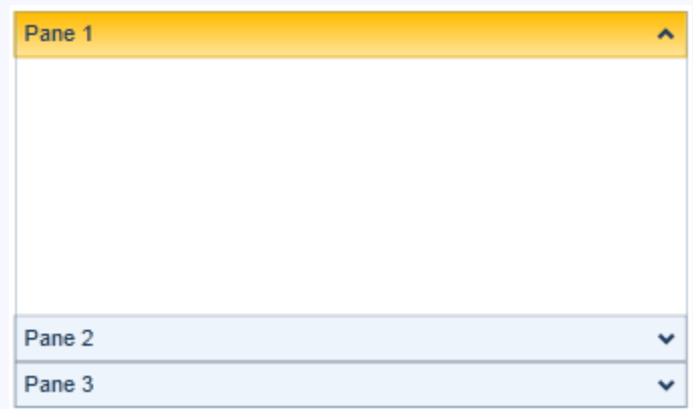
ComponentOne Accordion for WPF incorporates several themes that allow you to customize the appearance of your grid. When you first add a C1Accordion control to the page, it appears similar to the following image:

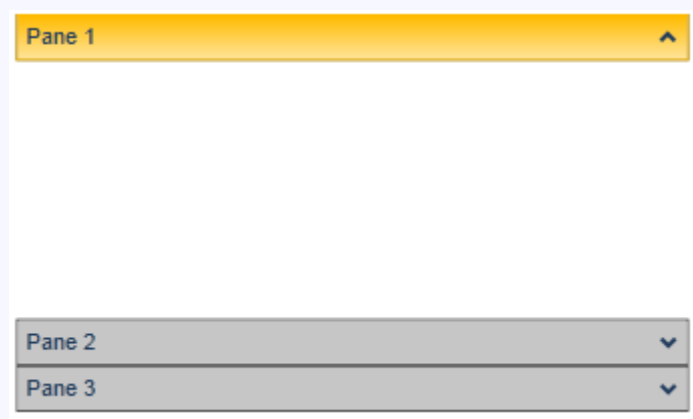
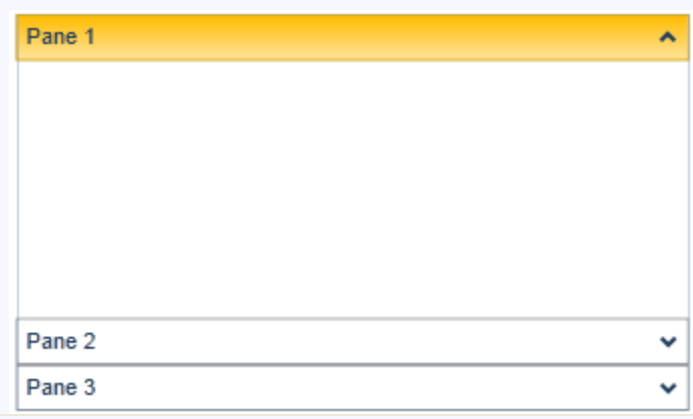

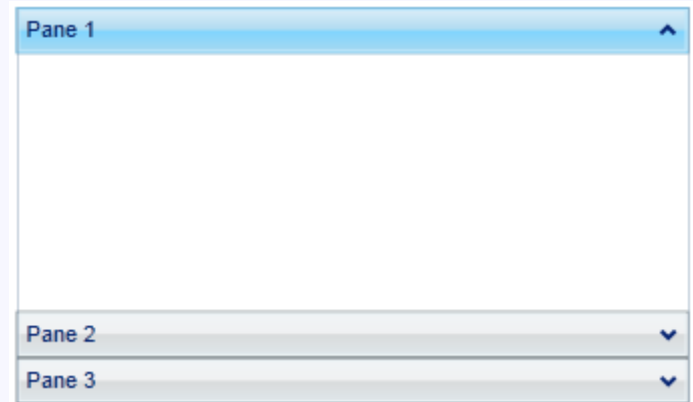


This is the control's default appearance. You can change this appearance by using one of the built-in themes or by creating your own custom theme. All of the built-in themes are based on WPF Toolkit themes. The built-in themes are described and pictured below; note that in the images below, a row has been selected to show selected styles:

Theme Name	Theme Preview
------------	---------------

C1Blue	
C1ThemeBureauBlack	
C1ThemeExpressionDark	
C1ThemeExpressionLight	

C1ThemeOffice2007Blue	 <p>Office 2007 Blue theme layout showing three panes. Pane 1 is at the top with an orange header and an upward arrow. Pane 2 is in the middle with a light blue header and a downward arrow. Pane 3 is at the bottom with a light blue header and a downward arrow.</p>
C1ThemeOffice2007Black	 <p>Office 2007 Black theme layout showing three panes. Pane 1 is at the top with an orange header and an upward arrow. Pane 2 is in the middle with a grey header and a downward arrow. Pane 3 is at the bottom with a grey header and a downward arrow.</p>
C1ThemeOffice2007Silver	 <p>Office 2007 Silver theme layout showing three panes. Pane 1 is at the top with an orange header and an upward arrow. Pane 2 is in the middle with a light purple header and a downward arrow. Pane 3 is at the bottom with a light purple header and a downward arrow.</p>
C1ThemeOffice2010Blue	 <p>Office 2010 Blue theme layout showing three panes. Pane 1 is at the top with a yellow header and an upward arrow. Pane 2 is in the middle with a light blue header and a downward arrow. Pane 3 is at the bottom with a light blue header and a downward arrow.</p>

C1ThemeOffice2010Black	 <p>Pane 1</p> <p>Pane 2</p> <p>Pane 3</p>
C1ThemeOffice2010Silver	 <p>Pane 1</p> <p>Pane 2</p> <p>Pane 3</p>
C1ThemeShinyBlue	 <p>Pane 1</p> <p>Pane 2</p> <p>Pane 3</p>
C1ThemeWhistlerBlue	 <p>Pane 1</p> <p>Pane 2</p> <p>Pane 3</p>

To set an element's theme, use the **ApplyTheme** method. First add a reference to the theme assembly to your project, and then set the theme in code, like this:

- Visual Basic

```
Private Sub Window_Loaded(sender As System.Object, e As
System.Windows.RoutedEventArgs) Handles MyBase.Loaded
    Dim theme As New C1ThemeExpressionDark

    ' Using ApplyTheme
    C1Theme.ApplyTheme(LayoutRoot, theme)
```

- C#

```
private void Window_Loaded(object sender, RoutedEventArgs e)
{
    C1ThemeExpressionDark theme = new C1ThemeExpressionDark();

    //Using ApplyTheme
    C1Theme.ApplyTheme(LayoutRoot, theme);
}
```

To apply a theme to the entire application, use the **System.Windows.ResourceDictionary.MergedDictionaries** property. First add a reference to the theme assembly to your project, and then set the theme in code, like this:

- Visual Basic

```
Private Sub Window_Loaded(sender As System.Object, e As
System.Windows.RoutedEventArgs) Handles MyBase.Loaded
    Dim theme As New C1ThemeExpressionDark

    ' Using Merged Dictionaries
    Application.Current.Resources.MergedDictionaries.Add(C1Theme.GetCurrentThem
eResources(theme))
End Sub
```

- C#

```
private void Window_Loaded(object sender, RoutedEventArgs e)
{
    C1ThemeExpressionDark theme = new C1ThemeExpressionDark();

    //Using Merged Dictionaries
    Application.Current.Resources.MergedDictionaries.Add(C1Theme.GetCurrentThem
eResources(theme));
}
```

Note that this method works only when you apply a theme for the first time. If you want to switch to another ComponentOne theme, first remove the previous theme from **Application.Current.Resources.MergedDictionaries**.

Accordion for WPF Samples

Please be advised that these ComponentOne software tools are accompanied by various sample projects and/or demos which may make use of other development tools included with the ComponentOne Studios.

Samples can be accessed from the **ComponentOne Control Explorer**. To view samples, on your desktop, click the **Start** button and then click **ComponentOne | Studio for WPF | Samples | WPF ControlExplorer**.

The following pages within the **ControlExplorer** detail the C1Accordion control:

Sample	Description
Accordion	Illustrates the functionality of the C1Accordion control.

Accordion for WPF Task-Based Help

The task-based help assumes that you are familiar with programming in Visual Studio .NET and know how to use the C1Accordion control in general. If you are unfamiliar with the **ComponentOne Accordion for WPF** product, please see the **Accordion for WPF** Quick Start first.

Each topic in this section provides a solution for specific tasks using the **ComponentOne Accordion for WPF** product.

Each task-based help topic also assumes that you have created a new WPF project.

Adding Accordion Panes to the C1Accordion Control

In this topic, you will add an accordion pane to a C1Accordion control in Design view, in XAML, and in code.

At Design Time in Design view

To add a pane to the C1Accordion control, complete the following steps:

1. Click the C1Accordion control once to select it.
2. In the **Properties** window, click the **Items** ellipsis button.
3. The Collection Editor: Items dialog box opens.
4. Click the **Add** button once to add one C1AccordionItem item to the C1Accordion control.
5. In the **Properties** grid, set the **Width** property to "150".

In XAML

To add a pane to the C1Accordion control, place the following markup between the `<c1ext:C1Accordion>` and `</c1ext:C1Accordion>` tags:

```
<c1ext:C1AccordionItem Name="C1AccordionItem" Width="150">
</c1ext:C1AccordionItem>
```

In Code

To add accordion panes in code, complete the following steps:

1. Enter Code view and import the following namespace:
 - Visual Basic
2. Add the following code beneath the **InitializeComponent()** method:
 - C#

```
Imports C1.WPF.Extended
```

```
C#
using C1.WPF.Extended;
```

```
Visual Basic
'Create the accordion pane and add content
Dim C1AccordionItem1 As New C1AccordionItem()
C1AccordionItem1.Content = "C1AccordionItem1"
'Add the accordion pane to the C1Accordion control
C1Accordion1.Items.Add(C1AccordionItem1)
```

```
C#
//Create the accordion pane and add content
```

```
C1AccordionItem c1AccordionItem1 = new C1AccordionItem();
C1AccordionItem1.Content = "c1AccordionItem1";
//Add the accordion pane to the C1Accordion control
c1Accordion1.Items.Add(c1AccordionItem1);
```

3. Run the program.

Adding Content to Header Elements

You can easily add both simple text and WPF controls to an accordion pane's header. The topics in this section will provide step-by-step instructions about adding text content and controls to the header.

For more information on the header element, you can also visit the [Accordion Pane Header](#) (page 7) topic.

Adding Text to the Header

By default, an accordion pane's header is empty. You can add text to the control's header by setting the **Header** property to a string in Design view, in XAML, or in code.

This topic assumes that you have added a `C1Accordion` control with at least one `C1AccordionItem` item to your project.

In XAML

To set the **Header** property in XAML, add `Header="Hello World"` to the `<c1ext:C1AccordionItem>` tag so that it appears similar to the following:

```
<c1ext:C1AccordionItem Name="C1AccordionItem1" Header="Hello World"
Width="150" Height="55">
```

In Code

To set the **Header** property in code, complete the following steps:

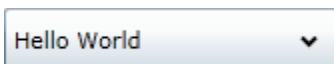
1. Enter Code view and add the following code beneath the **InitializeComponent()** method:
 - Visual Basic

```
C1AccordionItem1.Header = "Hello World"
```
 - C#

```
c1AccordionItem1.Header = "Hello World";
```
2. Run the program.

✔ This Topic Illustrates the Following:

The header of the accordion pane now reads "Hello World". The result of this topic should resemble the following:



Adding a Control to the Header

Any accordion pane header element is able to accept a WPF control. In this topic, you will add a **Button** control to the header in XAML and in code.

This topic assumes that you have added a `C1Accordion` control with at least one `C1AccordionItem` item to your project.

In XAML

To add a **Button** control to the header in XAML, place the following XAML markup between the `<clext:C1AccordionItem>` and `</clext:C1AccordionItem>` tags:

```
<clext:C1AccordionItem.Header>
  <Button Content="Button" Height="Auto" Width="50"/>
</clext:C1AccordionItem.Header>
```

In Code

To add a **Button** control to the header in code, complete the following steps:

1. Enter Code view and add the following code beneath the **InitializeComponent()** method:

- Visual Basic

```
'Create the Button control
Dim NewButton As New Button()
NewButton.Content = "Button"
'Set the Button Control's Width and Height properties
NewButton.Width = 50
NewButton.Height = Double.NaN
'Add the Button to the header
C1AccordionItem1.Header = (NewButton)
```

- C#

```
InitializeComponent();
//Create the Button control
Button NewButton = new Button();
NewButton.Content = "Button";
//Set the Button Control's Width and Height properties
NewButton.Width = 50;
NewButton.Height = Double.NaN;
//Add the Button to the header
c1AccordionItem1.Header = (NewButton);
```

2. Run the program.

✔ This Topic Illustrates the Following:

As a result of this topic, the control will appear in the header. The final result will resemble the following image:



Adding Content to Content Areas

You can easily add both simple text and WPF controls to an accordion pane's header. The topics in this section will provide step-by-step instructions about how to add text content and controls to the header.

For more information on the header element, you can also visit the [Accordion pane Content Area](#) (page 8) topic.

Adding Text to the Content Area

You can easily add a simple line of text to the content area of an accordion pane by setting the **Content** property to a string in Design view, in XAML, or in code.

This topic assumes that you have added a `C1Accordion` control with at least one `C1AccordionItem` item to your project.

Note: You can also add text to the content area by adding a **TextBox** control to the content area and then setting the **TextBox** control's **Text** property. To learn how to add a control to the content area, see [Adding a Control to the Content Area](#) (page 26).

At Design Time in Design view

To set the **Content** property in Design view, complete the following steps:

1. Click the `C1AccordionItem` item once to select it.
2. In the **Properties** window, set the **Content** property to a string (for example, "Hello World").
3. Run the program and expand the accordion pane.

In XAML

To set the **Content** property in XAML, complete the following:

1. Add `Content="Hello World"` to the `<clext:C1AccordionItem>` tag so that it appears similar to the following:

```
<clext:C1AccordionItem Name="C1AccordionItem1" Content="Hello World" Width="150" Height="55">
```

2. Run the program and expand the accordion pane.

In Code

To set the **Content** property in code, complete the following steps:

1. Enter Code view and add the following code beneath the **InitializeComponent()** method:

- Visual Basic

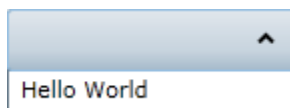
```
C1AccordionItem1.Content = "Hello World"
```
- C#

```
c1AccordionItem1.Content = "Hello World";
```

2. Run the program and expand the accordion pane.

✔ This Topic Illustrates the Following:

The content of your accordion pane now reads "Hello World". The result of this topic should resemble the following:



Adding a Control to the Content Area

Each accordion pane (`C1AccordionItem`) will accept one child control in its content area. In this topic, you will add a WPF button control in Design view, in XAML, and in code.

This topic assumes that you have added a `C1Accordion` control with at least one `C1AccordionItem` item to your project.

At Design Time in Design view

To add a **Button** control to the content area, complete the following steps:

1. Select the accordion pane you wish to add the control to.
2. Double click the **Button** icon to add it to the accordion pane's content area.
3. In the designer, select the **Button** control so that its properties take focus in the **Properties** window.
4. Set the **Width** property to "Auto".
5. Set the **Height** property to "Auto".
6. Run the program and expand the accordion pane to reveal the button control.

In XAML

Complete the following steps:

1. To add a **Button** control to the content area in XAML, complete the following:
2. Place the following markup between the `<c1ext:C1AccordionItem>` and `</c1ext:C1AccordionItem>` tags:

```
<Button Content="Button" Height="Auto" Width="Auto"/>
```

3. Run the program and expand the accordion pane to reveal the button control.

In Code

To add a **Button** control to the content area in code, complete the following:

1. Enter Code view and add the following code beneath the **InitializeComponent()** method:

- Visual Basic

```
'Create the Button control
Dim NewButton As New Button()
NewButton.Content = "Button"
'Set the Button Control's Width and Height properties
NewButton.Width = Double.NaN
NewButton.Height = Double.NaN
'Add the Button to the content area
C1AccordionItem1.Content = (NewButton)
```

- C#

```
//Create the Button control
Button NewButton = new Button();
NewButton.Content = "Button";
//Set the Button Control's Width and Height properties
NewButton.Width = double.NaN;
NewButton.Height = double.NaN;
```

```
//Add the Button to the content area
c1AccordionItem1.Content = (NewButton);
```

2. Run the program and expand the accordion pane to reveal the button control.

✔ This Topic Illustrates the Following:

When accordion pane is expanded, the button control will appear in its content area, resembling the following image:



Adding Multiple Controls to the Content Area

You cannot set an accordion pane's (`C1AccordionItem`) **Content** property to more than one control at a time. However, you can circumvent this issue by adding a panel-based control that can accept more than one control, such as a **StackPanel** control, to the content area of the accordion pane. When you add multiple controls to the panel-based control, each one will appear within the accordion pane's content area.

This topic assumes that you have added a `C1Accordion` control with at least one `C1AccordionItem` item to your project.

To add multiple controls to the content area, complete these steps:

1. Place the following XAML markup between the `<c1ext:C1AccordionItem>` and `</c1ext:C1AccordionItem>` tags:

```
<c1ext:C1AccordionItem.Content>
  <StackPanel>
    <TextBlock Text="1st TextBlock"/>
    <TextBlock Text="2nd TextBlock"/>
    <TextBlock Text="3rd TextBlock"/>
  </StackPanel>
</c1ext:C1AccordionItem.Content>
```

2. Run the program.
3. Expand the accordion pane and observe that each of the three **TextBlock** controls appear in the content area. The result will resemble the following:



Changing the Expand Direction

By default, the `C1Accordion` control's accordion panes expand from top-to-bottom because the `ExpandDirection` property is set to **Down**. You can easily change the expand direction by setting the `ExpandDirection` property to **Up**, **Right**, or **Left** in Design view, in XAML, or in code.

This topic assumes that you have added a `C1Accordion` control with at least one `C1AccordionItem` to your project.

At Design Time in Design view

To set the `ExpandDirection` property in Design view, complete the following steps:

1. Click the `C1Accordion` control once to select it.
2. In the **Properties** window, click the `ExpandDirection` drop-down arrow and select one of the options from the list. For this example, select **Right**.

In XAML

To set the `ExpandDirection` property to **Right** in XAML, add `ExpandDirection="Right"` to the `<clext:C1Accordion>` tag so that it appears similar to the following:

```
<clext:C1Accordion Name="C1Accordion1" ExpandDirection=Right  
Width="150" Height="55">
```

In Code

To set the `ExpandDirection` property in code, complete the following steps:

1. Enter Code view and add the following code beneath the **InitializeComponent()** method:
 - Visual Basic
`C1Accordion1.ExpandDirection = C1.WPF.Extended.ExpandDirection.Right`
 - C#
`c1Accordion1.ExpandDirection = C1.WPF.Extended.ExpandDirection.Right;`
2. Run the program.

✔ This Topic Illustrates the Following:

By following the instructions in this topic, you have learned how to set the `ExpandDirection` property. In this topic, you set the `ExpandDirection` property to **Right**, which will make the `C1Accordion` control resemble the following:



Filling Out the Accordion's Height

The default behavior of an accordion pane (`C1AccordionItem`) is to fill only to the height (for accordions that expand up or down) or width (for accordions that expand right or left) of its contents. If you want the `C1Accordion` control to fill to a specific height or width, you will have to set the height or width of the control and then set the control's `Fill` property to **True**.

At Design Time in Design view

To set the `Fill` property in Design view, complete the following steps:

1. Add three accordion panes to your `C1Accordion` control (see [Adding Accordion Panes to the C1Accordion Control](#) (page 23)).
2. Click the `C1Accordion` control once to select it.
3. In the **Properties** window, complete the following:

- Select the **Fill** check box.
- Set the **Height** property to "200".

In XAML

To set the Fill property to **True** in XAML, complete the following:

1. Add `Fill="True"` to the `<clext:C1Accordion>` tag so that it appears similar to the following:

```
<clext:C1Accordion Name="C1Accordion1" Fill="True">
```

2. Add `Height="200"` to the `<clext:C1Accordion>` tag so that it appears similar to the following:

```
<clext:C1Accordion Name="C1Accordion1" Fill="True" Height="200">
```

In Code

To set the Fill property in code, complete the following steps:

1. Add and `Height="200"` to the `<clext:C1Accordion>` tag. This ensures that each accordion pane expands to fill a height of 200 pixels.
2. Enter Code view and add the following code beneath the **InitializeComponent()** method:

- Visual Basic

```
C1Accordion1.Fill = True
```

- C#

```
c1Accordion1.Fill = true;
```

3. Run the program.