
ComponentOne

Menu and ContextMenu for Silverlight

Copyright © 1987-2011 ComponentOne LLC. All rights reserved.

Corporate Headquarters
ComponentOne LLC
201 South Highland Avenue
3rd 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 ComponentOne LLC. 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


Menu and ContextMenu for Silverlight Overview	1
Menu and ContextMenu for Silverlight Key Features	1
Menu and ContextMenu for Silverlight Quick Start	2
Step 1 of 5: Creating an Application with a C1Menu Control	2
Step 2 of 5: Adding Menu Items to the Control	3
Step 3 of 5: Adding Submenus to the Menu Items	4
Step 4 of 5: Adding a C1ContextMenu to the C1Menu Control	5
Step 5 of 5: Running the Project	6
Working with C1Menu and C1ContextMenu	8
C1Menu Elements	9
C1ContextMenu Elements	10
C1Menu and C1ContextMenu Features	11
Automatic Closing	11
Nested Submenus	11
Boundary Detection	11
Checkable Menu Items	13
Menu and ContextMenu for Silverlight Layout and Appearance	13
C1Menu and ContextMenu ClearStyle Properties	13
Menu and ContextMenu for Silverlight Appearance Properties	15
Text Properties	15
Content Positioning Properties	15
Color Properties	16
Border Properties	16
Size Properties	16
Templates	16
Item Templates	17
Menu Theming	18
Menu and ContextMenu for Silverlight Task-Based Help	21
Creating Menus	21
Creating a Top-Level Menu	21
Creating a Submenu	23
Creating a Context Menu	25
Working with Themes	26
Adding a Theme to the C1ContextMenu Control	26
Adding a Theme to the C1Menu Control	28
Working with Checkable Menu Items	29
Creating a Checkable Menu Item	29
Creating Mutually Exclusive Checkable Menu Items	30
Enabling Boundary Detection	31
Enabling Automatic Menu Closing	32
Adding a Separator Between Menu Items	32
Adding an Icon to a Menu Item	33

Menu and ContextMenu for Silverlight Overview

Add a complete menu system to your Silverlight app with **ComponentOne Menu™ for Silverlight** and **ComponentOne ContextMenu™ for Silverlight**. Use the C1Menu control to create menu and C1ContextMenu control to attach pop-up menu to your interface.

Menu for Silverlight includes the following controls:

- **C1ContextMenu**
The C1ContextMenu provides a pop-up menu that provides frequently used commands that are associated with the selected object.
- **C1Menu**
The C1Menu is a control that allows hierarchical organization of elements associated with event handlers.

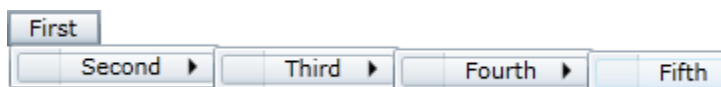
 **Getting Started**

- [Working with Menus](#) (page 8)
- [Quick Start](#) (page 2)
- [Task-Based Help](#) (page 21)

Menu and ContextMenu for Silverlight Key Features

ComponentOne Menu for Silverlight and **ComponentOne ContextMenu for Silverlight** allow you to create customized, rich applications. Make the most of **Menu for Silverlight** by taking advantage of the following key features:

- **Browser Boundaries Detection**
Menus are positioned automatically and always stay within the page bounds. See [Boundary Detection](#) (page 11) for more information.
- **Keyboard Navigation Support**
The C1Menu control supports keyboard navigation, making your Silverlight applications more accessible.
- **Deep Menus**
Nest menus to any depth. See [Nested Submenus](#) (page 11) for more information.

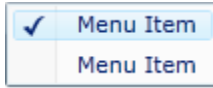


- **Icons for Menu Items**

Menu allows you to use an icon for each menu item along with the menu label.

- **Checkable Menu Items**

Declare menu items with checked/unchecked states. See [Checkable Menu Items](#) (page 13) for more information.



- **Context Menus**

The C1ContextMenu control enables you to provide pop-up menus that associate frequently used commands with selected objects. The C1ContextMenuService class exposes context menus as extender properties that can be attached to any **FrameworkElement** objects on the page, much like the **ToolTip** property provided by the **ToolTipService** class. See [C1ContextMenu Elements](#) (page 10) for more information.

- **Silverlight Toolkit Themes Support**

Add style to your UI with built-in support for the most popular Microsoft Silverlight Toolkit themes, including ExpressionDark, ExpressionLight, WhistlerBlue, RainierOrange, ShinyBlue, and BureauBlack. See [Menu Theming](#) (page 18) for further details.

Menu and ContextMenu for Silverlight Quick Start



The following quick start guide is intended to get you up and running with **Menu for Silverlight** and **ContextMenu for Silverlight**. In this quick start, you'll start in Expression Blend to create a new project with the C1Menu control. You will also add menu items, submenus, and a context menu to the control.

Step 1 of 5: Creating an Application with a C1Menu Control

In this step, you'll begin in Expression Blend to create a Silverlight application using the C1Menu control.

Complete the following steps:

1. In Expression Blend, select **File | New Project**.
2. In the **New Project** dialog box, select the Silverlight project type in the left pane and, in the right-pane, select **Silverlight Application + Website**.
3. Enter a **Name** and **Location** for your project, select a **Language** in the drop-down box, and click **OK**. Blend creates a new application, which opens with the **MainPage.xaml** file displayed in Design view.
4. Add the C1Menu control to your project by completing the following steps:
 - a. On the menu, select **Window | Assets** to open the **Assets** tab.
 - b. Under the **Assets** tab, enter "C1Menu" into the search bar.
 - c. The C1Menu control's icon appears.

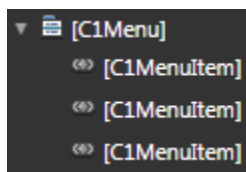
- d. Double-click the C1Menu icon to add the control to your project.
5. Under the **Objects and Timeline** tab, select **[C1Menu]** and then, under the **Properties** panel, set the following properties:
 - Locate the **Width** property and click its glyph  to set the width of the control to **Auto**.
 - Locate the **Height** property and click its glyph  to set the height of the control to **Auto**.

You have completed the first step of the **Menu and ContextMenu for Silverlight** quick start. In this step, you created a project and added a C1Menu control to it. In the next step, you will add top-level menu items to the control.





Step 2 of 5: Adding Menu Items to the Control



In the last step, you created a Silverlight project with a C1Menu control attached to it. In this step, you will add three menu items to the C1Menu control.

1. Add a C1MenuItem icon to the **Tools** panel by completing the following steps:
 - a. In the **Tools** panel, click the **Assets** button (the double-chevron button).
 - b. In the search bar, enter "C1MenuItem" to bring up the **C1MenuItem** icon.
 - c. Double-click the **C1MenuItem** icon to add it to the Tools panel.
2. Add a menu item to the menu by completing the following steps:
 - a. Under the **Objects and Timeline** tab, select **[C1Menu]**.
 - b. On the **Tools** panel, double-click the **C1MenuItem** icon (this is located beneath the Assets button) to add the **C1MenuItem** to the **C1Menu** control.
3. Repeat step 2 twice to add a total of three menu items to the C1Menu control. The hierarchy under the **Objects and Timeline** tab should appear as follows:



If the hierarchy doesn't appear as shown in the above image, you can use drag-and-drop operations to rearrange the C1MenuItems.

4. Under the **Objects and Timeline** tab, select the first **[C1MenuItem]** and then, under the **Properties** panel, set the following properties:
 - Locate the **Width** property and click its glyph  to set the width of the control to **Auto**.
 - Locate the **Height** property and click its glyph  to set the height of the control to **Auto**.
 - Locate the **Header** property and set it to "File".
5. Under the **Objects and Timeline** tab, select the second **[C1MenuItem]** and then, under the **Properties** panel, set the following properties:
 - Locate the **Width** property and click its glyph  to set the width of the control to **Auto**.
 - Locate the **Height** property and click its glyph  to set the height of the control to **Auto**.

- Locate the **Header** property and set it to "Edit".
6. Under the **Objects and Timeline** tab, select the third [**C1MenuItem**] and then, under the **Properties** panel, set the following properties:
 - Locate the **Width** property and click its glyph  to set the width of the control to **Auto**.
 - Locate the **Height** property and click its glyph  to set the height of the control to **Auto**.
 - Locate the **Header** property and set it to "Added Items".

In this step, you added top-level menu items to the C1Menu control in Expression Blend. In the next step, you will use XAML to add submenus to two of those items.

Step 3 of 5: Adding Submenus to the Menu Items

In the last step, you added the top-level menu items in Expression Blend. In this step, you will add submenus to two of the menu items using XAML.

1. Switch to XAML view.
2. Modify the first two `<c1:C1MenuItem>` tags so that they have both opening and closing tags. They should resemble the following:

```
<c1:C1MenuItem Header="File"></c1:C1MenuItem>
<c1:C1MenuItem Header="Edit"></c1:C1MenuItem>
```

Adding opening and closing tags to these items will allow you to nest other C1MenuItems between these tags in the next step. This step is necessary to create submenus in XAML.

3. To add a submenu to the "File" menu item, add the following markup between the `<c1:C1MenuItem Header="File">` and `</c1:C1MenuItem>` tags:

```
<c1:C1MenuItem Height="Auto" Width="Auto" Header="New">
  <c1:C1MenuItem
    Height="Auto"
    Width="Auto"
    Header="Document"/>
  <c1:C1MenuItem Height="Auto" Width="Auto" Header="Project"/>
</c1:C1MenuItem>
<c1:C1MenuItem Height="Auto" Width="Auto" Header="Open">
  <c1:C1MenuItem
    Height="Auto"
    Width="Auto"
    Header="Document"/>
  <c1:C1MenuItem
    Height="Auto"
    Width="Auto"
    Header="Project"/>
  <c1:C1Separator/>
  <c1:C1MenuItem Header="Recent Document 1"
    Height="Auto"
    Width="Auto"
    GroupName="CheckedDocuments"
    IsCheckable="True"
    IsChecked="True">
</c1:C1MenuItem>
```

```

        <c1:C1MenuItem
            Header="Recent Document 2"
            Height="Auto"
            Width="Auto"
            GroupName="CheckedDocuments"
            IsCheckable="True">
        </c1:C1MenuItem>
</c1:C1MenuItem>
<c1:C1Separator/>
<c1:C1MenuItem
    Height="Auto"
    Width="Auto"
    Header="Close"/>
<c1:C1MenuItem
    Height="Auto"
    Width="Auto"
    Header="Close Solution"/>
<c1:C1Separator/>
<c1:C1MenuItem
    Height="Auto"
    Width="Auto"
    Header="Exit"/>

```

4. To add a submenu to the "Edit" menu item, following markup between the `<c1:C1MenuItem Header="Edit">` and `</c1:C1MenuItem>` tags:

```

<c1:C1MenuItem
    Height="Auto"
    Width="Auto"
    Header="Undo"/>
<c1:C1MenuItem
    Height="Auto"
    Width="Auto"
    Header="Redo"/>

```

In this step, you added submenus to two of the C1Menu control's menu items. In the next step, you will add a C1ContextMenu control to the C1Menu control.

Step 4 of 5: Adding a C1ContextMenu to the C1Menu Control

In the last step, you added submenus to two of the C1Menu control's menu items. In this step, you will add a C1ContextMenu control to the C1Menu control. This context menu will have one item that, when clicked, will add submenu items to the C1Menu control's top-level "Added Items" top-level menu item that you created in [Step 2 of 5: Adding Menu Items to the Control](#) (page 3).

Complete the following steps:

1. In XAML view, place the following XAML markup right before the `</c1:C1Menu>` tag:

```

<c1:C1ContextMenuService.ContextMenu>
    <c1:C1ContextMenu Width="Auto" Height="Auto">
        <c1:C1MenuItem
            Height="Auto"
            Width="Auto"
            Header="Add Item"

```

```

        Click="C1MenuItem_AddOnClick"/>
    </cl:C1ContextMenu>
</cl:C1ContextMenuService.ContextMenu>

```

The above markup adds a `C1ContextMenu` control to the `C1Menu` control using the **C1ContextMenuService** helper class. Note that the `C1ContextMenu` control contains one `C1MenuItem` that is attached to a **Click** event named "C1MenuItem_AddOnClick".

2. Add `x:Name="AddedItems"` to the `<cl:C1MenuItem Header="Added Items"/>` tag. This gives the item a unique identifier so that you can call it in code.
3. Open the **MainPage.xaml.cs** page and add the following **Click** event handler to the project:

- Visual Basic

```

Private Sub C1MenuItem_AddOnClick(ByVal sender As Object, ByVal
e As Cl.Silverlight.SourcedEventArgs)
    Dim C1MenuItemAdd As New Cl.Silverlight.C1MenuItem()
    C1MenuItemAdd.Header = "Added Item"
    AddedItems.Items.Add(C1MenuItemAdd)
End Sub

```

- C#

```

private void C1MenuItem_AddOnClick(object sender,
Cl.Silverlight.SourcedEventArgs e)
{
    Cl.Silverlight.C1MenuItem C1MenuItemAdd = new
Cl.Silverlight.C1MenuItem();
    C1MenuItemAdd.Header = "Added Item";
    AddedItems.Items.Add(C1MenuItemAdd);
}

```

When the **Click** event is raised, this code will create a new `C1MenuItem` and add it to the "Added Items" menu item.

4. Click the **Projects** tab and then double-click **Default.htm** to open the HTML document.

Note: If you are using Silverlight 4, you can skip step 5.

5. In **Default.htm**, add `<param name="windowless" value="true" />` between the `<Object>` and `</Object>` tags. This makes the application windowless so that users can right-click the `C1Menu` control to bring up the `C1ContextMenu` control.

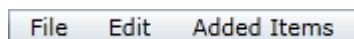
In this step, you added a `C1ContextMenu` control to the `C1Menu` control. In the next step, you will run the project and see the result of the **Menu for Silverlight** quick start.

Step 5 of 5: Running the Project

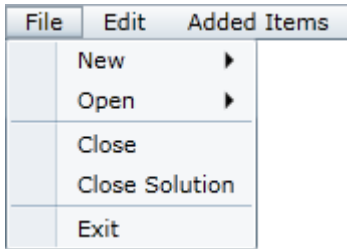
Now that you have created a Silverlight project with a `C1Menu` control and a `C1ContextMenu` control, the only thing left to do is run the project and observe the results of your work.

Complete the following steps:

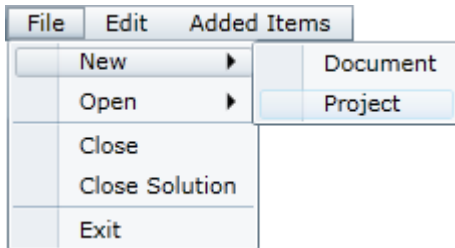
1. From the toolbar, select **Project | Run Project** to run the application. The application appears as follows:



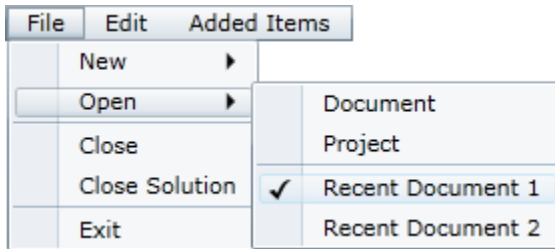
- Click **File** and observe that a submenu appears.



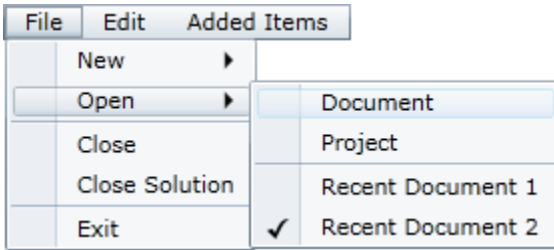
- Hover your cursor over **New** and observe that another submenu appears.



- Hover your cursor over **Open** and observe that another submenu appears. Note that **Recent Document 1** is checked.

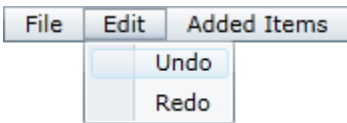


- Click **Recent Document 2**.
The **Open** submenu closes.
- Select **File | Open** and observe that **Recent Document 2** is checked instead of **Recent Document 1**.

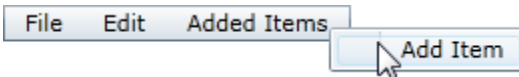


The two checkable items, **Recent Document 1** and **Recent Document 2**, are grouped together to form a list of mutually exclusive checkable items. You can read more about this by visiting the [Checkable Menu Items](#) (page 13) topic.

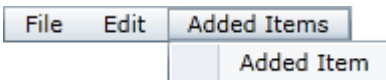
- Click **Edit** and observe that the **Edit** submenu appears.



- Click **Added Items** and observe that it has no submenu.
- Right-click the menu to bring up the context menu.
- On the context menu, click **Add Item**.



- Click **Added Items** and observe that it has a submenu consisting of one new item, which is named **Added Item**.



Congratulations! You have completed the **Menu for Silverlight** quick start. Now that you have learned some of the basics of the control, we recommend that you visit the [Working with C1Menu and C1ContextMenu](#) (page 8) section for an overview of the controls and their features.

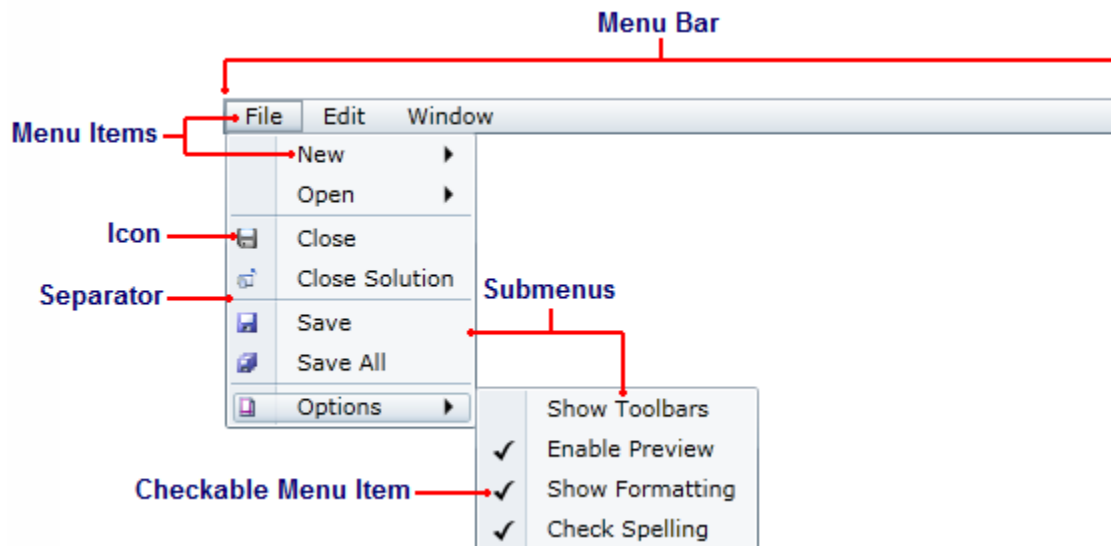
Working with C1Menu and C1ContextMenu

The following topics outline the basics of working with the C1Menu and C1ContextMenu controls. In this section, you will find outlines of the controls' elements and descriptions of some of the controls' most popular features.

C1Menu Elements

The C1Menu is a control that allows hierarchical organization of elements associated with event handlers. The menu can be nested to any depth that you desire, and you can add as many items to the menu as you need to add.

The following image diagrams the elements of the C1Menu control.



The elements of the C1Menu control can be described as follows:

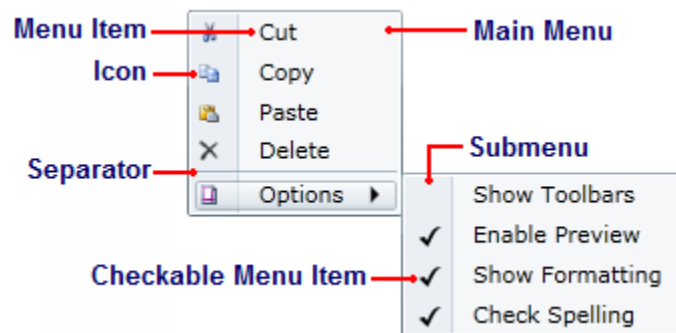
- **Menu Bar**
The main menu is a horizontal, top-level menu. It is comprised of first-level C1MenuItems and can hold any feature available to C1MenuItems.
- **Submenus**
Submenus are menus that can only be accessed from other menus. A submenu is created when you nest a C1MenuItem within another C1MenuItem.
- **Menu Items**
Menu items are represented by the C1MenuItem class. Menu items can have labels and icons, and they can also be specified as checkable menu items. Each menu item is associated with a click event handler.
- **Checkable Menu Item**
Checkable menu items are objects of the C1MenuItem class that can be selected or cleared by users. You can make checkable menu items standalone, or you can group them with other checkable menu items to create a list of mutually exclusive check boxes. Checkable menu items can't be used in the menu bar.
- **Icon**
Icons are created by adding an **Image** control to the Icon property. Icons can't be used in the menu bar.
- **Separator**
Separators are created through the C1Separator class. Icons can't be used in the menu bar.

C1ContextMenu Elements

The C1ContextMenu provides a pop-up menu that provides frequently used commands that are associated with the selected object. The C1ContextMenuService class exposes context menus as extender properties that can be attached to any **FrameworkElement** objects on the page, much like the **ToolTip** property provided by the **ToolTipService** class. Once the menu is attached to an object, the users can right-click that object to open the menu.

Note: The C1ContextMenu will only work in a windowless Silverlight application prior to Silverlight 4. For information about creating a windowless application, see [Creating a Context Menu](#) (page 25).

The following image diagrams the elements of the C1ContextMenu control.



The elements of the C1Menu control can be described as follows:

- **Main Menu**
The main menu is a horizontal bar comprised of first-level C1MenuItem. Menu items can be standalone items, or they can contain lists of submenu items.
- **Submenus**
Submenus are menus that can only be accessed from other menus. A submenu is created when you nest a C1MenuItem within another C1MenuItem.
- **Menu Items**
Menu items are represented by the C1MenuItem class. Menu items can have labels and icons, and they can also be specified as checkable menu items. Each menu item is associated with a click event handler.
- **Checkable Menu Item**
Checkable menu items are objects of the C1MenuItem class that can be selected or cleared by users. You can make checkable menu items standalone, or you can group them with other checkable menu items to create a list of mutually exclusive check boxes.
- **Icon**
Icons are created by adding an **Image** control to the Icon property.
- **Separator**
Separators are created through the C1Separator class. They can be used to create visual groups of menu items.

C1Menu and C1ContextMenu Features

The C1Menu control and the C1ContextMenu control share a common object model, and both controls take C1MenuItem items. This means that the C1Menu and C1ContextMenu controls possess the same features. This section outlines some popular features shared by the menu controls.

Automatic Closing

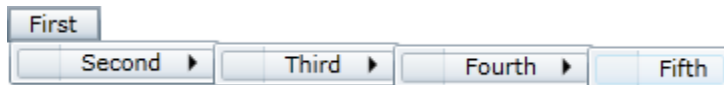
By default, the C1ContextMenu control, its submenus, and the submenus of a C1Menu control will remain open even when a user clicks outside of the menu. The only way users can close the menu is if they click the C1Menu control or the C1ContextMenu control. However, you can change this behavior by enabling the automatic closing feature, which will allow users to close menus by clicking outside of the menu control's boundaries. To turn on automatic closing, set the AutoClose property to **True**.

Nested Submenus

The C1Menu control and the C1ContextMenu control can hold submenus. These submenus are created when C1MenuItems are nested within the tags of other C1MenuItems. For example, placing the following XAML markup

```
<c1:C1MenuItem Header="First">
  <c1:C1MenuItem Header="Second">
    <c1:C1MenuItem Header="Third">
      <c1:C1MenuItem Header="Fourth">
        <c1:C1MenuItem Header="Fifth"/>
      </c1:C1MenuItem>
    </c1:C1MenuItem>
  </c1:C1MenuItem>
</c1:C1MenuItem>
```

between the opening and closing tags of a C1Menu would create the following:



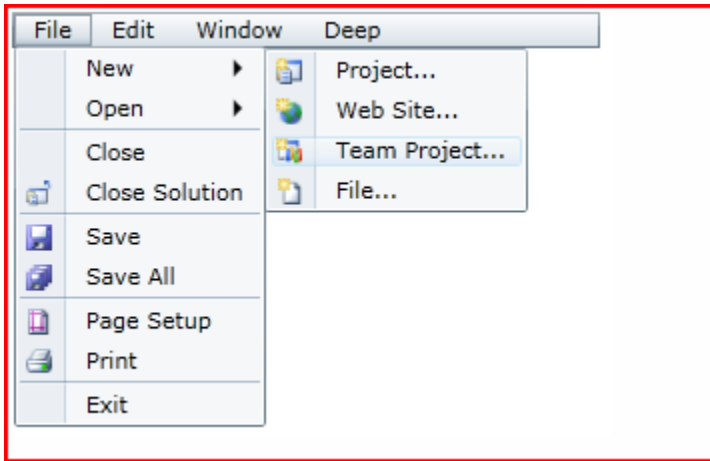
You can have as many nested menus as you want, although it's best not to have more than two or three submenus in a hierarchy for usability purposes.

For task-based help on creating a nested submenu for a C1Menu or a C1ContextMenu control, see [Creating a Submenu](#) (page 23).

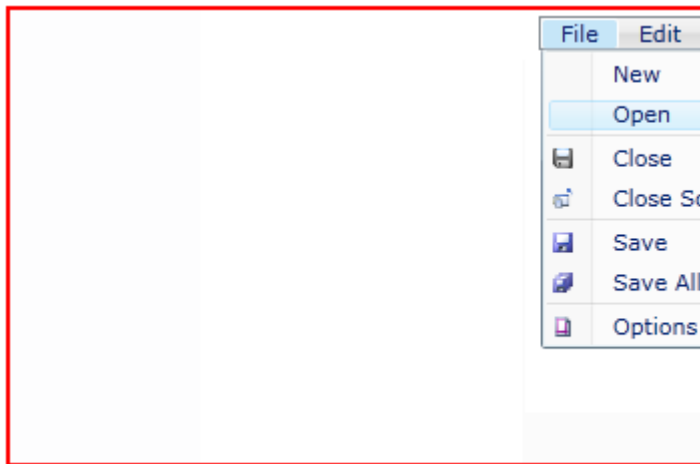
Boundary Detection

Boundary detection ensures that the menus a user opens will always stay within the page bounds. This is helpful if you have several nested submenus within your menu, as these menus can expand until they're beyond the scope of a project page.

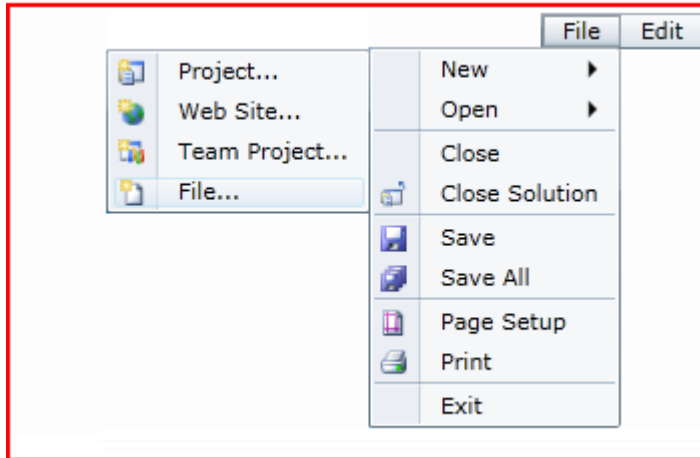
The image below illustrates a menu contained within a boundary large enough to allow for the expansion of two submenus.



The image below illustrates what that same menu would look like if the boundaries were shifted several centimeters to the left with the boundary detection feature disabled. Observe that the menu is cut off despite there being an extensive blank space to the left of the control.



The image below illustrates that same menu and same boundary, only this time boundary detection is enabled. Observe that the submenus shift to the left to avoid being cut off by the tight boundary on the right side of the page.



By default, boundary detection is disabled, but you can enable it by setting the `DetectBoundaries` property to **True**. For task-based help about enabling boundary detection, see the [Enabling Boundary Detection](#) (page 31) topic.

Checkable Menu Items

You can make any `C1MenuItem` a checkable menu item by setting its `IsCheckable` property to **True**. The value of the `IsChecked` property determines whether or not the menu item is checked or unchecked. By default, the `IsChecked` property of an item is **False**. When the item is clicked, the value of the `IsChecked` property sets to **True**.

You can create a group of mutually exclusive checkable items by setting the `GroupName` property of each item you wish to add to the group. For example, the XAML below will create a group of three mutually exclusive checkable items.

```
<c1:C1MenuItem Header="MenuItem1" IsCheckable="True"
GroupName="MutuallyExclusiveGroup" />
<c1:C1MenuItem Header="MenuItem2" IsCheckable="True"
GroupName="MutuallyExclusiveGroup" />
<c1:C1MenuItem Header="MenuItem3" IsCheckable="True"
GroupName="MutuallyExclusiveGroup" />
```

Menu and ContextMenu for Silverlight Layout and Appearance

The following topics detail how to customize the `C1Menu` and `C1ContextMenu` controls' 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 Silverlight's XAML-based styling. You can also use templates to format and layout the control and to customize the control's actions.

C1Menu and ContextMenu ClearStyle Properties

Menu and ContextMenu for Silverlight 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 **C1Menu** control:

Brush	Description
Background	Gets or sets the brush of the control's background.
HighlightedBackground	Gets or sets the System.Windows.Media.Brush used to highlight the menu items when they are hovered over.
OpenedBackground	Gets or sets the System.Windows.Media.Brush used to highlight the menu items when they are opened.

The following table outlines the brush properties of **C1ContextMenu** control:

ClearStyle Property	Description
Background	Gets or sets the brush of the control's background.
HighlightedBackground	Gets or sets the System.Windows.Media.Brush used to highlight the menu items when they are hovered over.
OpenedBackground	Gets or sets the System.Windows.Media.Brush used to highlight the menu items when they are opened.

The following table outlines the brushes of the **C1MenuItem** control:

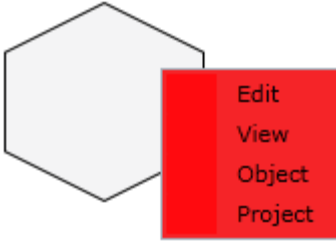
ClearStyle Property	Description
Background	Gets or sets the brush of the control's background.
HighlightedBackground	Gets or sets the System.Windows.Media.Brush used to highlight the menu item when it is hovered over.
OpenedBackground	Gets or sets the System.Windows.Media.Brush used to highlight the menu item when it is opened.
HeaderForeground	Gets or sets the foreground brush of the header.
HeaderBackground	Gets or sets the background brush of the header.

You can completely change the appearance of the **C1Menu** or **C1ContextMenu** controls by setting a few properties, such as the **C1Menu**'s **Background** property, which sets the background color of the menu. For example, if you set the **Background** property to "#FFE4005", the **C1Menu** control would appear similar to the following:



File Edit View Object Project

And if you set the **C1ContextMenu**'s **Background** property to "#FFE4005", it 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](#) topic.

Menu and ContextMenu for Silverlight Appearance Properties

ComponentOne Menu and ContextMenu for Silverlight 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 `C1Menu` control, the `C1ContextMenu` control, and the `C1MenuItem` items.

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.

Content Positioning Properties

The following properties let you customize the position of header and content area content in the `C1Menu` control, the `C1ContextMenu` control, and the `C1MenuItem` items.

Property	Description
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 C1Menu control, the C1ContextMenu control, and the C1MenuItem items.

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.

Border Properties

The following properties let you customize the borders of the C1Menu control, the C1ContextMenu control, and the C1MenuItem items.

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 C1Menu control, the C1ContextMenu control, and the C1MenuItem items.

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.

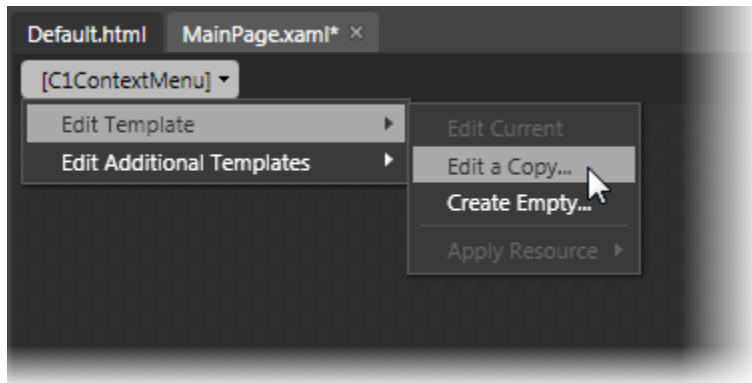
Templates

One of the main advantages to using a Silverlight 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 Silverlight applications, you can provide your own UI for data managed by **ComponentOne Menu for Silverlight**. 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 C1Menu or C1ContextMenu 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. The image below illustrates this using the C1ContextMenu control.



If you want to edit a C1MenuItem template, simply select the C1MenuItem 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.

Additional Menu Templates

In addition to the default templates, the C1Menu control and the C1ContextMenu control include a few additional templates. These additional templates can also be accessed in Microsoft Expression Blend – in Blend select the C1Menu or C1ContextMenu control and, in the menu, select **Edit Additional Templates**. Choose a template, and select **Create Empty**,

Additional Menu Item Templates

In addition to the default templates, the C1Menu control and the C1MenuItem control include a few additional templates. These additional templates can also be accessed in Microsoft Expression Blend – in Blend select the C1Menu or C1MenuItem control and, in the menu, select **Edit Additional Templates**. Choose a template, and select **Create Empty**.

Item Templates

ComponentOne Menu for Silverlight's menu and context controls are **ItemsControls** that serve as a container for other elements. As such, both controls include templates to customize items places within the menu. These templates include an **ItemTemplate** and an **ItemsPanel** template. You use the **ItemTemplate** to specify the visualization of the data objects and the **ItemsPanel** to define the panel that controls the layout of items.

Accessing Templates

You can access these templates in Microsoft Expression Blend by selecting the C1Menu or C1ContextMenu control and, in the menu, selecting **Edit Additional Templates**. Choose **Edit Generated Items (ItemTemplate)** or **Edit Layout of Items (ItemsPanel)** and select **Create Empty** to create a new blank template or **Edit a Copy**.

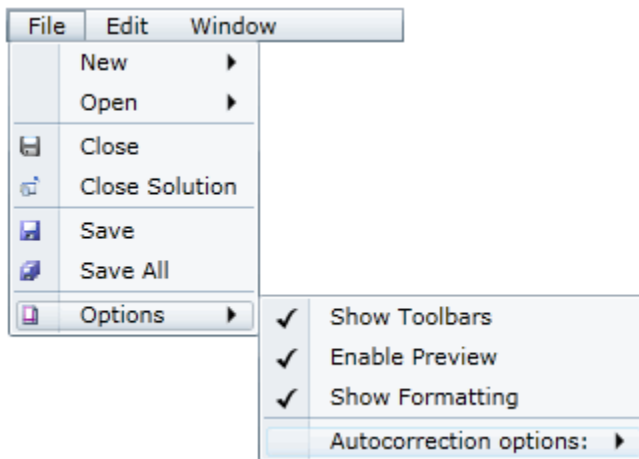
A dialog box will appear allowing you to name the template and determine where to define the template.

Menu Theming

Silverlight themes are a collection of image settings that define the look of a control or controls. The benefit of using themes is that you can apply the theme across several controls in the application, thus providing consistency without having to repeat styling tasks.

Note: This topic only illustrates themes on the C1Menu control. However, the themes for the C1ContextMenu control are identical to the theming of the C1Menu control's submenus.

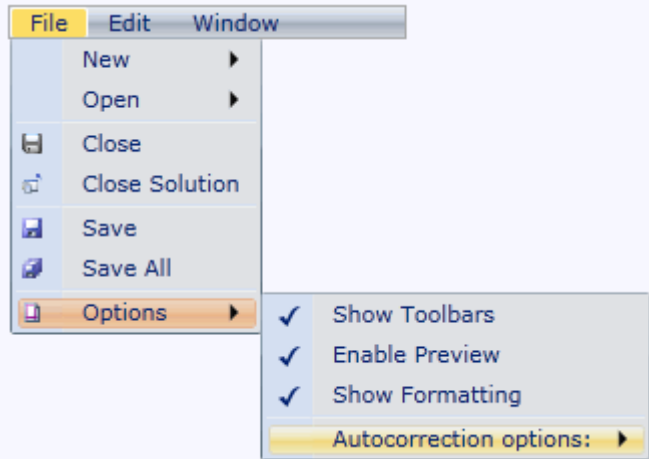
When you add the C1Menu control to your project, it appears with the default blue theme:



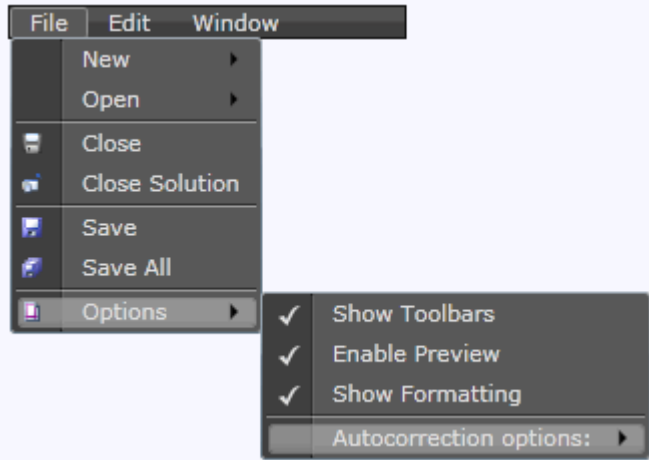
You can theme the C1Menu control with one of our six included Silverlight themes: BureauBlack, ExpressionDark, ExpressionLight, RainierOrange, ShinyBlue, and WhistlerBlue. The table below shows a sample of each theme:

Full Theme Name	Appearance
-----------------	------------

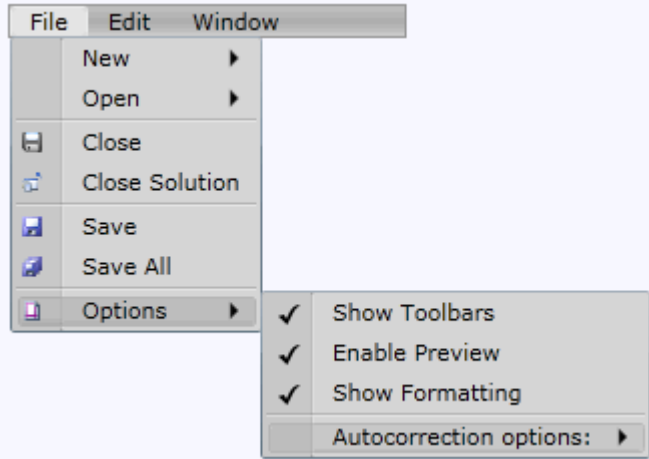
C1ThemeBureauBlack

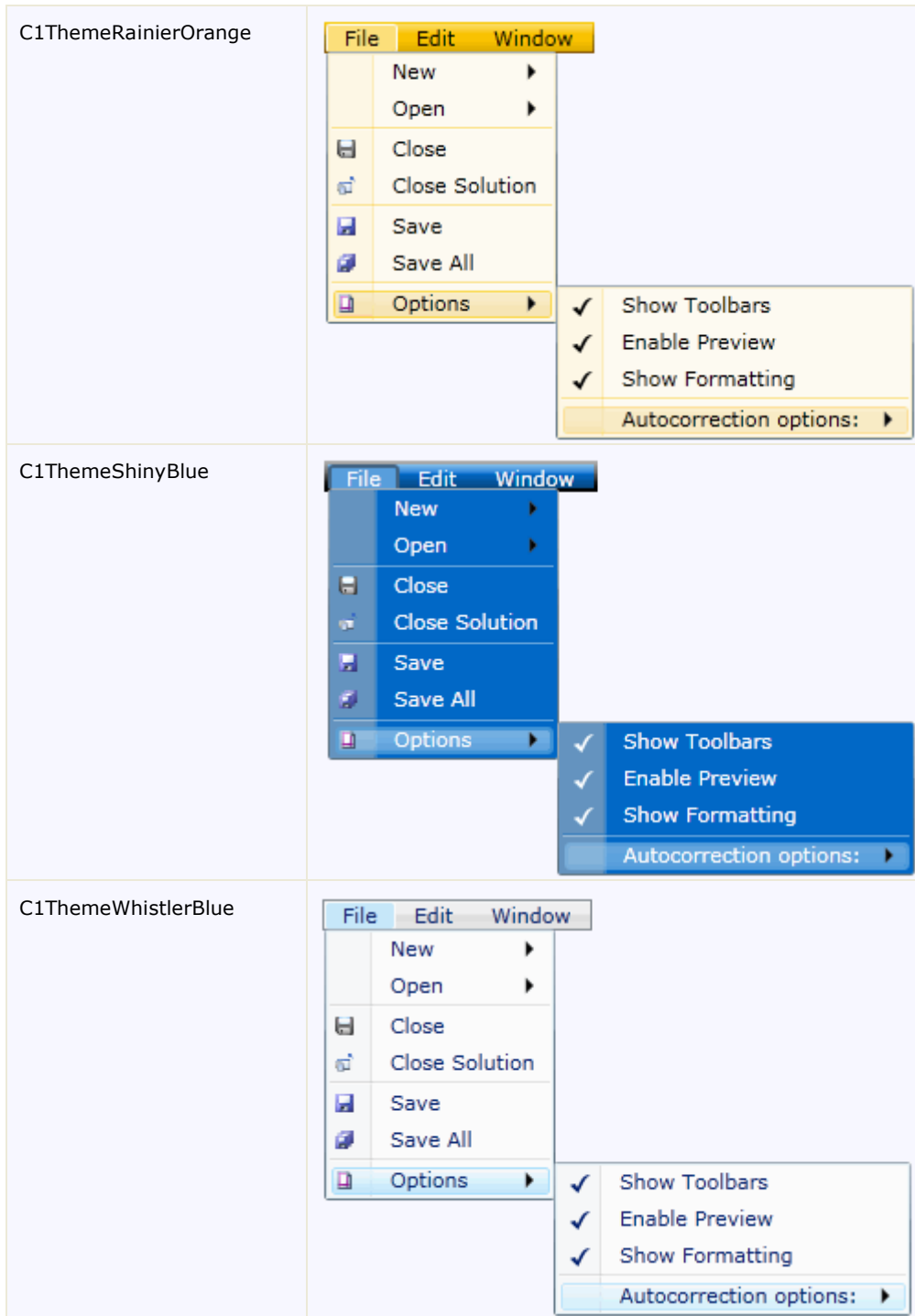


C1ThemeExpressionDark



C1ThemeExpressionLight





You can add any of these themes to the menu controls by declaring the theme around the control in markup. For task-based help about adding a theme to the menu controls, see [Working with Themes](#) (page 26).

Menu and ContextMenu for Silverlight Task-Based Help

The task-based help assumes that you are familiar with programming in Visual Studio .NET and know how to use the C1Menu control in general. If you are unfamiliar with the **ComponentOne Menu for Silverlight** and **ComponentOne ContextMenu for Silverlight** products, please see the **Menu and ContextMenu for Silverlight Quick Start** first.

Each topic in this section provides a solution for specific tasks using the **C1Menu** and **C1ContextMenu** controls. Each task-based help topic also assumes that you have created a new Silverlight project.

Creating Menus



The following topics detail how to create different types of menus.

Creating a Top-Level Menu

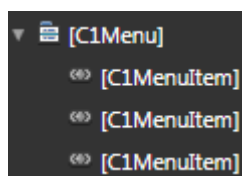
In this topic, you will learn how to create a top-level menu for the C1Menu and C1ContextMenu controls.

In Blend

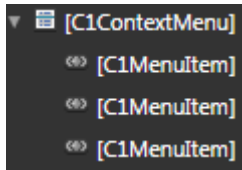
Complete the following steps:







1. Add a C1Menu control or a C1ContextMenu (see [Creating a Context Menu](#) (page 25)) control to your project.
2. Select the C1Menu or the C1ContextMenu control and set the following properties:
 - Locate the **Width** property and click its glyph  to set the width of the control to **Auto**.
 - Locate the **Height** property and click its glyph  to set the height of the control to **Auto**.
3. Add a C1MenuItem icon to the **Tools** panel by completing the following steps:
 - a. In the **Tools** panel, click the **Assets** button (the double-chevron button).
 - b. In the search bar, enter "C1MenuItem" to bring up the **C1MenuItem** icon.
 - c. Double-click the **C1MenuItem** icon to add it to the Tools panel.
4. Add a menu item to the menu by completing the following steps:
 - a. Under the **Objects and Timeline** tab, select **[C1Menu]** or **[C1ContextMenu]**.
 - b. On the **Tools** panel, double-click the **C1MenuItem** icon (this is located beneath the double-chevron button) to add the **C1MenuItem** to the control.
5. Repeat step 2 twice to add a total of three menu items to the C1Menu control. The hierarchy under the **Objects and Timeline** tab should appear as follows:

- For a C1Menu 



- For a C1ContextMenu 



6. Select the first **[C1MenuItem]** and set the following properties:
 - Locate the **Width** property and click its glyph  to set the width of the control to **Auto**.
 - Locate the **Height** property and click its glyph  to set the height of the control to **Auto**.
 - Locate the **Header** property and set it to "MenuItem1".
7. Select the second **[C1MenuItem]** and set the following properties:
 - Locate the **Width** property and click its glyph  to set the width of the control to **Auto**.
 - Locate the **Height** property and click its glyph  to set the height of the control to **Auto**.
 - Locate the **Header** property and set it to "MenuItem2".
8. Select the third **[C1MenuItem]** and set the following properties:
 - Locate the **Width** property and click its glyph  to set the width of the control to **Auto**.
 - Locate the **Height** property and click its glyph  to set the height of the control to **Auto**.
 - Locate the **Header** property and set it to "MenuItem3".

Note: If you are working with the C1ContextMenu control, you will have to attach it to another Framework element using the C1ContextMenuService class. For information about attaching C1ContextMenu to an element, see [Creating a Context Menu](#) (page 25).

9. Run the program and complete one of the following:
 - If you are using a C1Menu control, observe that a menu bar with three items appears.
OR
 - If you are using a C1ContextMenu control, right-click the element the context menu is attached to. Observe that a menu with three items appears.

In XAML

Complete the following steps:

1. Place the following XAML between the `<c1:C1Menu>` and `</c1:C1Menu>` tags or the `<c1:C1ContextMenu>` and `</c1:C1ContextMenu>` tags.

```
<c1:C1MenuItem Header="MenuItem1" Height="Auto" Width="Auto"/>
<c1:C1MenuItem Header="MenuItem2" Height="Auto" Width="Auto"/>
<c1:C1MenuItem Header="MenuItem3" Height="Auto" Width="Auto"/>
```

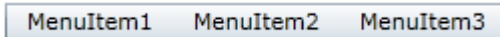
2. Add `Height="Auto"` and `Width="Auto"` to the `<c1:C1Menu>` or `<c1:C1ContextMenu>` tag.
3. Run the program and complete one of the following:

- If you are using a C1Menu control, observe that a menu bar with three items appears.
OR
- If you are using a C1ContextMenu control, right-click the element the context menu is attached to. Observe that a menu with three items appears.

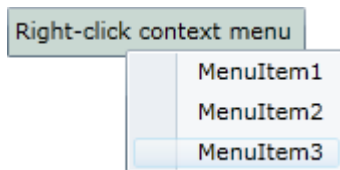
✔ **This Topic Illustrates the Following:**

This topic illustrates one of the following results:

- If you used a C1Menu to complete this task, the result will resemble the following:






- If you used a C1ContextMenu to complete this task, the result will resemble the following:

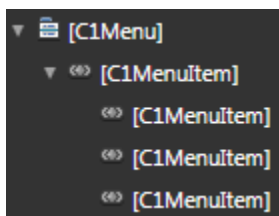


Creating a Submenu

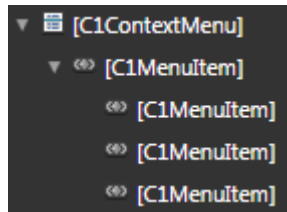
In this topic, you will create a submenu that's attached to one of a menu's items. This topic assumes that you have created a top-level menu (see [Creating a Top-Level Menu](#) (page 21)) with at least one menu item.





Complete the following steps:

1. Add a C1Menu control or a C1ContextMenu control to your project.
2. Select the C1Menu or the C1ContextMenu control and set the following properties:
 - Locate the **Width** property and click its glyph  to set the width of the control to **Auto**.
 - Locate the **Height** property and click its glyph  to set the height of the control to **Auto**.
3. Under the **Assets** tab, enter "C1MenuItem" in the search bar.
The C1MenuItem icon appears.
4. Double-click the C1MenuItem icon to add a menu item the menu control. Repeat this step three times to add a total of four menu items to your project.
5. Switch to the **Objects and Timeline** tab.
6. Use drag-and-drop operations to create the following hierarchy:
 - For a C1Menu 



- For a C1ContextMenu 



7. Select the first **[C1MenuItem]** and set the following properties:
 - Locate the **Width** property and click its glyph  to set the width of the control to **Auto**.
 - Locate the **Height** property and click its glyph  to set the height of the control to **Auto**.
 - Locate the **Header** property and set it to "Click here".
8. Set the properties of each submenu item as follows:
 - Locate the **Width** property and click its glyph  to set the width of the control to **Auto**.
 - Locate the **Height** property and click its glyph  to set the height of the control to **Auto**.
 - Locate the **Header** property and set it to "Submenu Item".

Note: If you are working with the C1ContextMenu control, you will have to attach it to another **Framework** element using the C1ContextMenuService class. For information about attaching C1ContextMenu to an element, see [Creating a Context Menu](#) (page 25).

9. Run the program and complete one of the following:
 - If you are using a C1Menu control, click **Click here**. Observe that a submenu with three items appears.
 - OR
 - If you are using a C1ContextMenu control, right-click the element the context menu is attached to. Click **Click Here** and observe that a submenu with three items appears.

In XAML

Complete the following steps:

1. Place the following XAML between the `<c1:C1Menu>` and `</c1:C1Menu>` tags or the `<c1:C1ContextMenu>` and `</c1:C1ContextMenu>` tags.

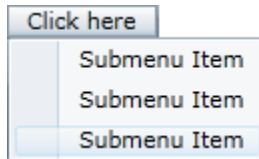
```
<c1:C1MenuItem Header="Click here" Height="Auto" Width="Auto">
  <c1:C1MenuItem Header="Submenu Item" Height="Auto" Width="Auto"/>
  <c1:C1MenuItem Header="Submenu Item" Height="Auto" Width="Auto"/>
  <c1:C1MenuItem Header="Submenu Item" Height="Auto" Width="Auto"/>
</c1:C1MenuItem>
```

2. Add `Height="Auto"` and `Width="Auto"` to the `<c1:C1Menu>` or `<c1:C1ContextMenu>` tag.
3. Run the program and complete one of the following:
 - If you are using a C1Menu control, click **Click here**. Observe that a submenu with three items appears.
 - OR
 - If you are using a C1ContextMenu control, right-click the element you attached it to in order to open the context menu. Click **Click Here** and observe that a submenu with three items appears.

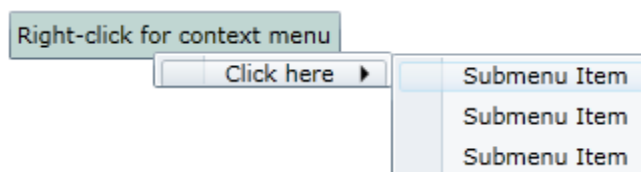
✔ **This Topic Illustrates the Following:**

This topic illustrates one of the following results:

- If you used a `C1Menu` to complete this task, the result will resemble the following:



- If you used a `C1ContextMenu` to complete this task, the result will resemble the following:



Creating a Context Menu

In this topic, you will use the `C1ContextMenuService` class to attach a context menu to a **TextBox** control.

Complete the following steps:

1. Add a reference to the **C1.Silverlight** assembly to your Silverlight project.
2. Open Source view (XAML view if you're using Blend).
3. Add the following markup to the `<UserControl>` tag to import the namespace of the assembly:

```
xmlns:c1="clr-namespace:C1.Silverlight;assembly=C1.Silverlight"
```

4. Replace the `<Grid>` tag with the following markup to add a **TextBox** control to the project:

```
<Grid x:Name="LayoutRoot" Background="White">  
    <TextBox Text="Right-click for context menu" Width="170" Height="25"  
Background="#FFC4D8D4">  
        </TextBox>  
    </Grid>
```

5. Add `C1ContextMenuService` to the **TextBox** by placing the following markup between the `<TextBox>` and `</TextBox>` tags:

```
<c1:C1ContextMenuService.ContextMenu>  
</c1:C1ContextMenuService.ContextMenu>
```

The `C1ContextMenuService` class exposes context menus as extender properties that can be attached to any **FrameworkElement** objects on the page.

- Place the following XAML between the `<c1:C1ContextMenuService.ContextMenu>` and `</c1:C1ContextMenuService.ContextMenu>` tags to create a context menu and a few context menu items:

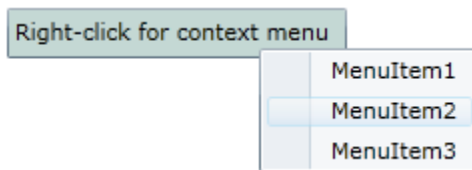
```
<c1:C1ContextMenu Width="Auto" Height="Auto">
  <c1:C1MenuItem Header="MenuItem1"></c1:C1MenuItem>
  <c1:C1MenuItem Header="MenuItem2"></c1:C1MenuItem>
  <c1:C1MenuItem Header="MenuItem3"></c1:C1MenuItem>
</c1:C1ContextMenu>
```

Note: If you are using Silverlight 4, you can skip step 7.

- In order to get the `C1ContextMenu` control to work properly, you will have to change your project to a windowless application. To accomplish this, open the `[ProjectName]TestPage.aspx` page (`Default.htm` if you're using Blend) and add the following markup between the `<Object>` and `</Object>` tags:

```
<param name="windowless" value="true" />
```

- Run the project and right-click the **TextBox** control. Observe that a context menu with three items appears.



Working with Themes

The following topics illustrate how to add Silverlight themes to the `C1ContextMenu` control and the `C1Menu` control.

Adding a Theme to the `C1ContextMenu` Control

Because the `C1ContextMenu` control is always attached to another control using a service, adding a theme to the `C1ContextMenu` control is going to be slightly different than adding a theme to another Silverlight control. You will have to wrap the theme around a grid or a panel and then use the implicit style manager to ensure that the theme will be passed down to the `C1ContextMenu` control.

In Blend

Complete the following steps:

- Click the **Assets** tab.
- In the search bar, enter "C1ThemeRainierOrange".
The **C1ThemeRainierOrange** icon appears.
- Double-click the **C1ThemeRainierOrange** icon to add it to your project.
- Navigate to the **Tools** panel and double-click the **Grid** icon to add a grid to your project.
- Click the **Objects and Timeline** tab.
- Select **[Grid]** and, using a drag-and-drop operation, place it underneath **[C1ThemeRainierOrange]**.

- With **[Grid]** still selected, return to the **Assets** tab.
- Add a control, such as a **TextBox** control or a **Border** control, to the grid and then add a **C1ContextMenu** control to that control (see [Creating a Context Menu](#) (page 25)).
- Switch to XAML view.
- Add the following namespace to the `<UserControl>` tag:

```
xmlns:c1Theming="clr-
namespace:C1.Silverlight.Theming;assembly=C1.Silverlight.Theming"
```

- Add `c1Theming:ImplicitStyleManager.ApplyMode="OneTime"` to the `<c1:C1ContextMenu>` tag that your markup resembles the following:

```
<c1:C1ContextMenu c1Theming:ImplicitStyleManager.ApplyMode="Auto">
```

- Run your project.

In Visual Studio

Complete the following steps:

- Open the `.xaml` page in Visual Studio.
- Place your cursor between the `<Grid></Grid>` tags.
- In the **Tools** panel, double-click the **C1ThemeRainierOrange** icon to declare the theme. Its tags will appear as follows:

```
<my:C1ThemeRainierOrange></my:C1ThemeRainierOrange>
```

- Place your cursor between the `<my:C1ThemeRainierOrange>` and `</my:C1ThemeRainierOrange>` tags and press **Enter**.
- In the **Tools** panel, double-click the **Grid** icon. The grid tags appear between the theme tags, as follows:

```
<my:C1ThemeRainierOrange>
  <Grid></Grid>
</my:C1ThemeRainierOrange>
```

- Add a **C1ContextMenu** between the `<Grid>` and `</Grid>` tags. For task-based help about adding a context menu to a Visual Studio project, see [Creating a Context Menu](#) (page 25).
- Add following namespace to the `<UserControl>` tag:

```
xmlns:c1Theming="clr-
namespace:C1.Silverlight.Theming;assembly=C1.Silverlight.Theming"
```

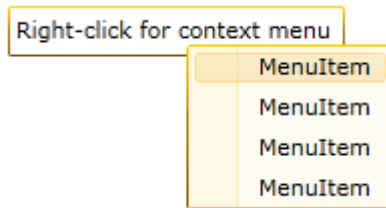
- Set the by **ApplyMode** property by adding `c1Theming:ImplicitStyleManager.ApplyMode="Auto"` to the `<c1ext:C1Accordion>` tag so that your markup resembles the following:

```
<c1:C1ContextMenu c1Theming:ImplicitStyleManager.ApplyMode="Auto">
```

9. Run your project.

✔ **This Topic Illustrates the Following:**

The following image depicts a C1ContextMenu control with the C1ThemeRainierOrange theme.



Adding a Theme to the C1Menu Control

The C1Menu control comes equipped with a light blue default theme, but you can also apply six themes (see [Menu Theming](#) (page 18)) to the control. In this topic, you will change the C1Menu control's theme to **C1ThemeRainierOrange**.

In Blend

Complete the following steps:

1. Click the **Assets** tab.
2. In the search bar, enter "C1ThemeRainierOrange".
The **C1ThemeRainierOrange** icon appears.
3. Double-click the **C1ThemeRainierOrange** icon to add it to your project.
4. In the search bar, enter "C1Menu" to search for the C1Menu control.
5. Double-click the C1Menu icon to add the C1Menu control to your project.
6. Under the **Objects and Timeline** tab, select [**C1Menu**] and use a drag-and-drop operation to place it under [**C1ThemeRainierOrange**].
7. Run the project.

In Visual Studio

Complete the following steps:

1. Open the **.xaml** page in Visual Studio.
2. Place your cursor between the `<Grid></Grid>` tags.
3. In the **Tools** panel, double-click the **C1ThemeRainierOrange** icon to declare the theme. Its tags will appear as follows:

```
<my:C1ThemeRainierOrange></my:C1ThemeRainierOrange>
```

4. Place your cursor between the `<my:C1ThemeRainierOrange>` and `</my:C1ThemeRainierOrange>` tags.
5. In the **Tools** panel, double-click the C1Menu icon to add the control to the project. Its tags will appear as children of the `<my:C1ThemeRainierOrange>` tags, causing the markup to resemble the following:

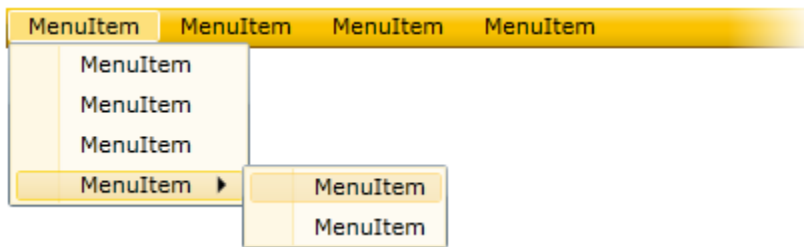
```
<my:C1ThemeRainierOrange>
```

```
<c1:C1Menu></c1:C1Menu>
</my:C1ThemeRainierOrange>
```

6. Run your project.

✔ **This Topic Illustrates the Following:**

The following image depicts a C1Menu control with the C1ThemeRainierOrange theme.



Working with Checkable Menu Items

In the following topics, you will learn how to create standalone and mutually exclusive checkable menu items.

Creating a Checkable Menu Item

In this topic, you will create a checkable menu item that can be selected or cleared by a user. In order to complete this topic, you must have a C1ContextMenu control that holds at least one item or a C1Menu control with at least one submenu.

In Blend

Complete the following steps:

1. In the **Objects and Timeline** tab, select the **[C1MenuItem]** that you wish to make checkable.
2. Under the **Properties** panel, select the **IsCheckable** check box to set the **IsCheckable** property to **True**. This makes the item checkable at run time.

Optional: If you want the item to be checked at run time, you can also select the **IsChecked** check box, but doing so isn't necessary to make an item checkable.

3. Run the project.

In XAML

Complete the following steps:

1. Locate the `<c1:C1MenuItem>` tag for the menu item you wish to make checkable and then add `IsCheckable="True"` to the tag so that the XAML resembles the following:

```
<c1:C1MenuItem Header="C1MenuItem" IsCheckable="True"/>
```

Optional: If you want the item to be checked at run time, you can also add `IsChecked="False"` to the tag.

2. Run the project.

In Code

Complete the following steps:

1. In Source view (XAML view if you're using Blend), locate the `<c1:C1MenuItem>` tag for the item you wish to make checkable and add `x:Name="CheckableMenuItem"` to it. This will give the item a unique identifier that you can use in code.
2. Enter Code view and add the following code beneath the **InitializeComponent()** method:

- Visual Basic
`CheckableMenuItem.IsCheckedable = True`
- C#
`CheckableMenuItem.IsCheckedable = true;`

Optional: If you want the item to be checked at run time, you can add the following code to the project:

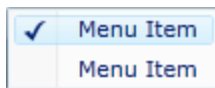
- Visual Basic
`CheckableMenuItem.IsChecked = True`
- C#
`CheckableMenuItem.IsChecked = true;`

3. Run the program

✔ This Topic Illustrates the Following:

Once the program is run, open the context menu or submenu. If you completed the optional step that created a checked item, a check mark will appear in the column to the left of the menu label; to clear the check mark, click the menu item. If you didn't complete the optional step, the check mark won't appear; to add a check mark, click the menu item.

The graphic below illustrates a checkable menu item.



Creating Mutually Exclusive Checkable Menu Items

In this topic, you learn how to create a list of checkable menu items that are grouped together so that only one item can be checked at a time.

In Blend

Complete the following steps:

1. Under the **Objects and Timeline** tab, select the first **[C1MenuItem]** you wish to make checkable and then set the following properties under the **Properties** panel:
 - Set the **IsCheckable** property to **True**.
 - Set the **GroupName** property to a string, such as "CheckableGroup".
2. Repeat step 1 for any other menu items you wish to add to the group of mutually exclusive checkable items.
3. Run the program and click the first item in the group. Observe that a check is added to the menu item. Now click the second item in the group and observe that the check is removed from the first item and then added to the second item.

In XAML

Complete the following steps:

1. Add `IsCheckable="True"` and `GroupName="CheckableGroup"` to the `<c1:C1MenuItem>` tag of each menu item you wish to add to the group of mutually exclusive checkable items.

2. Run the program and click the first item in the group. Observe that a check is added to the menu item. Now click the second item in the group and observe that the check is removed from the first item and then added to the second item.

In Code

Complete the following steps:

1. Set the **x>Name** property of each `C1MenuItem` you wish to add to the group of mutually exclusive checkable items.
2. Open the **MainPage.xaml.cs** page.
3. Set the `IsCheckable` and `GroupName` property of each menu item, replacing "ItemName" with the value of the menu item's **x>Name** property.
 - Visual Basic
`ItemName.IsCheckable = True`
 - C#
`ItemName.IsCheckable = true;`
4. Run the program and click the first item in the group. Observe that a check is added to the menu item. Now click the second item in the group and observe that the check is removed from the first item and then added to the second item.

Enabling Boundary Detection

Boundary detection ensures that the menus a user opens will always stay within page bounds. This is helpful if you have several nested submenus within your menu, as these menus can expand until they're beyond the scope of a project page. Boundary detection is disabled by default, but you can enable it by setting the `DetectBoundaries` property to **True**.

In Blend

Complete the following steps:

1. Select the `C1ContextMenu` control or the `C1Menu` control.
The control's properties appear underneath the **Properties** panel.
2. Select the `DetectBoundaries` check box.
3. Run the program.

In XAML

Complete the following steps:

1. Add `DetectBoundaries="True"` to the `<c1:C1Menu>` or the `<c1:C1ContextMenu>` tag.
2. Run the program.

In Code

Complete the following steps:

1. In Source view (XAML view, if you're using Blend), locate the `<c1:C1MenuItem>` tag for the item you wish to make checkable and add `x>Name="C1Menu1"` to it. This will give the item a unique identifier that you can use in code.
2. Enter Code view and add the following code beneath the **InitializeComponent()** method:
 - Visual Basic
`C1Menu1.DetectBoundaries = True`
 - C#

```
C1Menu1.DetectBoundaries = true;Run the program.
```

✔ This Topic Illustrates the Following:

When you run the project, expand all of your submenus and observe that they will shift position rather than being cut off at the edge of the Web page. For more information on boundary detection, see [Boundary Detection](#) (page 11).

Enabling Automatic Menu Closing

Automatic menu closing allows users to close menus by clicking outside of the menu area. To enable automatic menu closing, set the `AutoClose` property to **True**.

In Blend

Complete the following steps:

1. Select the `C1ContextMenu` control or the `C1Menu` control.
The control's properties appear underneath the **Properties** panel.
2. Select the `AutoClose` check box.
3. Run the program.

In XAML

Complete the following steps:

1. Add `AutoClose="True"` to the `<c1:C1Menu>` or the `<c1:C1ContextMenu>` tag.
2. Run the program.

In Code

Complete the following steps:

1. In Source view (XAML view, if you're using Blend), locate the `<c1:C1MenuItem>` tag for the item you wish to make checkable and add `x:Name="C1Menu1"` to it. This will give the item a unique identifier that you can use in code.
2. Enter Code view and add the following code beneath the `InitializeComponent()` method:
 - Visual Basic

```
C1Menu1.AutoClose = True
```
 - C#

```
C1Menu1.AutoClose = true;Run the program.
```

✔ This Topic Illustrates the Following:

After you've run the project, open the `C1ContextMenu` or one of the `C1Menu` control's submenus. With the submenu open, click outside of the submenu and observe that the submenu closes.

Adding a Separator Between Menu Items

In this topic, you will learn how to add separator bars between menu items.

In Blend

Complete the following steps:

1. Click the **Assets** tab and, in the search bar, enter "C1Separator".
2. Double-click the `C1Separator` icon to add the separator to your project.

3. Click the **Objects and Timeline** tab.
4. Select [**C1Separator**] and then use a drag-and-drop operation to place it in between two C1MenuItems.

In XAML

Complete one of the following:

To add a separator bar to a C1ContextMenu control, place `<c1:C1Separator />` between two `<c1:C1MenuItem>` tags. The result will resemble the following:

```
<c1:C1MenuItem Header="File" />
    <c1:C1Separator/>
<c1:C1MenuItem Header="Options" />
```

Adding an Icon to a Menu Item

In this step, you will learn how to add an icon to a C1MenuItem in XAML and in code. These steps will work in both Visual Studio and Blend.

In XAML

Complete the following steps:

1. Add an icon image to your Silverlight project. A 12x12 pixel image is best.
2. Add the following XAML markup between the `<c1:C1MenuItem>` and `</c1:C1MenuItem>` tags, replacing the value of the Source property with your image's name:

```
<c1:C1MenuItem.Icon>
    <Image Source="YourImage.png" Height="12" Width="12"
Margin="5,0,0,0"/>
</c1:C1MenuItem.Icon>
```

3. Run the project.

In Code

Complete the following steps:

1. Add an icon image to your Silverlight project. A 12x12 pixel image is best.
2. Add `x:Name="C1MenuItem1"` to the item you wish to add an icon to.
3. Import the following namespace:

- Visual Basic
`Imports System.Windows.Media.Imaging`

- C#
`using System.Windows.Media.Imaging;` Enter Code view and add the following code beneath the **InitializeComponent()** method:

```
• Visual Basic
'Create an image and assign it a source, margin, and width
Dim ItemIcon As New Image()
ItemIcon.Source = New BitmapImage(New Uri("02.png",
UriKind.RelativeOrAbsolute))
ItemIcon.Margin = New Thickness(5, 0, 0, 0)
ItemIcon.Height = 12
```

```
ItemIcon.Width = 12
//Set the C1MenuItem's icon property to the new image
C1MenuItem.Icon = ItemIcon
```

- C#

```
//Create an image and assign it a source, margin, and width
Image ItemIcon = new Image();
ItemIcon.Source = new BitmapImage(new Uri("02.png",
UriKind.RelativeOrAbsolute));
ItemIcon.Margin = new Thickness(5,0,0,0);
ItemIcon.Height = 12;
ItemIcon.Width = 12;
//Set the C1MenuItem's icon property to the new image
C1MenuItem1.Icon = ItemIcon;
```

5. Run the project.

✔ **This Topic Illustrates the Following:**

The following image depicts a C1MenuItem with a 12x12 pixel icon.

