ComponentOne

Toolbar for Silverlight

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)

This manual was produced using ComponentOne Doc-To-Help™.

Table of Contents

ComponentOne ToolBar for Silverlight Overview

Create custom toolbars to provide additional navigation in your website with **ComponentOne Toolbar™ for Silverlight**. **Toolbar for Silverlight** includes one control, **C1Toolbar**, which includes items (such as links, custom content, and separators) and groups, giving you the flexibility to place almost any control in the toolbar.

For a list of the latest features added to **ComponentOne Studio for Silverlight**, visit What's New in Studio for Silverlight.

Getting Started

Get started with the following topics:

- Key Features (page 3)

- Toolbar for Silverlight Layout and Appearance (page 15)

- C1Toolbar Templates (page 19)

Help with ComponentOne Studio for Silverlight

Getting Started

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

What's New

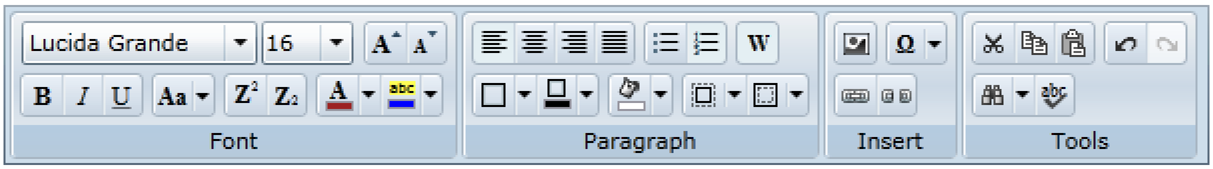
For a list of the latest features added to **ComponentOne Studio for Silverlight**, visit What's New in Studio for Silverlight.

Key Features

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

- **Ribbon-like Toolbar**

  Create an advanced Microsoft Ribbon-like toolbar.



- **Lightweight ToolbarStrip**

  Create a lightweight C1ToolbarStrip that can be used separately, for simple scenarios.

- **ToolBar Group**

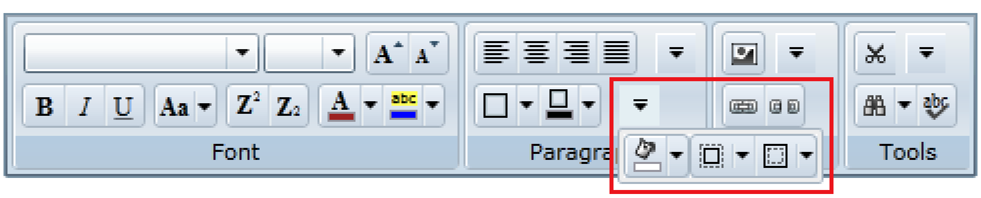
  The C1ToolBar control is a container the supports any UIElement including a C1ToolBarStrip. This is used to group similar toolbar buttons such as the “Font” group in Microsoft Word. The C1ToolbarGroup has a Header portion which is displayed as a band at the bottom with text.

  For more information see Toolbar Elements (page 8).

- **OverFlow Support**

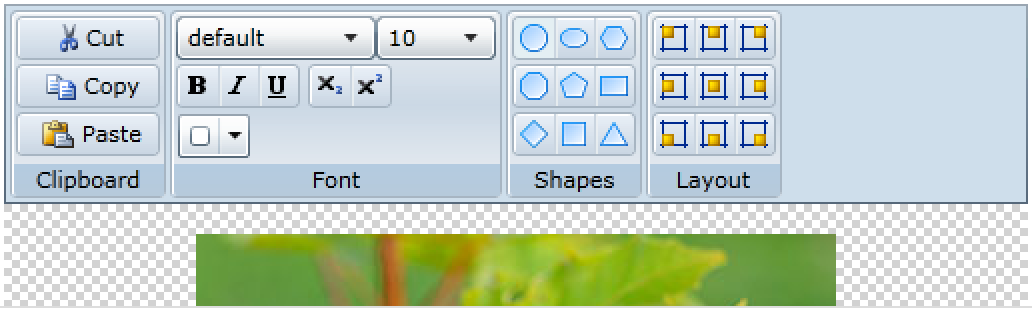
  Depending on the available space in the Strip panel, the items will jump between the Strip panel and drop-down Overflow panel. This occurs automatically by default, but can be set to occur never, always, or as needed.

  For more information see Toolbar Strip (page 12).



- **Personalize Your Toolbar**

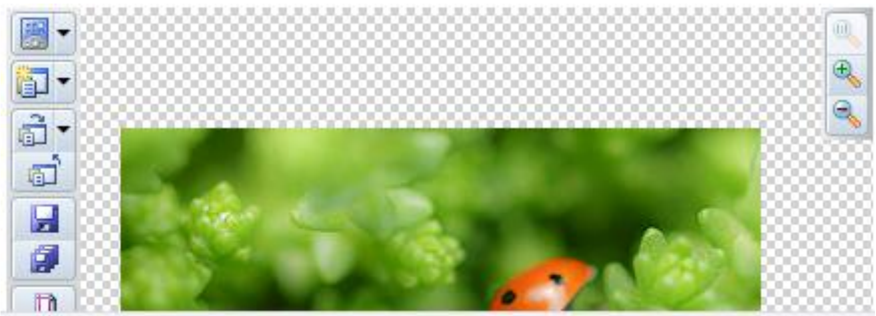
  Place a toolbar button, toggle button, drop-down button, or drop-down inside a C1ToolBarStrip. Attach a behavior to each toolbar item adding simple event handlers to them.



- **Change the Orientation**

  Select from horizontal (default) or vertical orientations for your toolbar.

  For more information see Toolbar Layout (page 16).



- **Add Separators**

  Draw a line, horizontal or vertical, between items in the toolbar. The separator helps to organize your toolbar.

- **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, RainerOrange, ShinyBlue, and BureauBlack.

- **Supports ClearStyle Technology**

  Toolbar for Silverlight supports ComponentOne's ClearStyle technology, which allows you to easily change control colors without having to change control templates. By setting a few color properties, you can quickly style the **Toolbar** elements. See Toolbar ClearStyle Properties (page 19) for more information on the ComponentOne ClearStyle technology.

Toolbar for Silverlight Quickstart

The following quick start guide is intended to get you up and running with **Toolbar for Silverlight**. In this quick start you'll start in Visual Studio and create a new project, add **Toolbar for Silverlight** to your application, and add C1Toolbar items such as C1ToolbarGroup, C1ToolbarStrip, C1ToolbarButton, and C1ToolbarToggleButton to your C1Toolbar. For more information see Toolbar Elements (page 8).

Step 1 of 3: Adding Toolbar for Silverlight to your Project

To set up your project and add a **C1Toolbar** control to your application, complete the following steps

1. Create a new Silverlight project in Visual Studio.
2. Add a reference to the C1.Silverlight and the **C1.Silverlight.C1Toolbar** assemblies. In the Solution Explorer right-click on **References** and select **Add Reference**. In the **Add Reference** dialog box select the Browse tab. Browse for the C1.Silverlight.C1Toolbar.dll and the C1.Silverlight and select **OK**.
3. Define the System and the **C1.WPF.C1Toolbar** prefixes.

```
xmlns:System="clr-namespace:System;assembly=mscorlib"


        xmlns:c1toolbar="clr-
namespace:C1.Silverlight.C1Toolbar;assembly=C1.Silverlight.C1Toolbar"
```

4. Add 2 Rows to the LayoutRoot Grid and set the **Height** of the first row to **Auto**.

```
<Grid x:Name="LayoutRoot" Background="White">
    <Grid.RowDefinitions>
        <RowDefinition Height="Auto"/>
        <RowDefinition Height="*"/>
    </Grid.RowDefinitions>
</Grid>
```

5. Drop a C1Toolbar onto the page within the first row and Remove the default properties: Height="100" HorizontalAlignment="Left" Margin="174,34,0,0". Your XAML should now look like the following:

```
<Grid.RowDefinitions>
   <RowDefinition Height="Auto"/>
   <RowDefinition Height="*"/>
</Grid.RowDefinitions>
   <c1:C1Toolbar Grid.Row="1" Name="c1Toolbar1">
   </c1:C1Toolbar>
```

Step 2 of 3: Adding C1ToolbarGroups to C1Toolbar

In this step you'll continue in Visual Studio by adding C1ToolbarGroups to your C1Toolbar. You'll then add C1ToolbarButtons to the C1ToolbarGroups.

1. Right-click on the **C1Toolbar** control and select **Properties** to open its Properties window. Navigate to the **ToolbarItems** property and click on the ellipsis button to open up the **Collection Editor: ToolbarItems**.
2. Click on the **Add** button twice to add two **C1ToolbarGroups** and click **OK**. Your XAML should now look like the following:

```
<c1:C1Toolbar Name="c1Toolbar1">
    <c1:C1ToolbarGroup />
    <c1:C1ToolbarGroup />
</c1:C1Toolbar>
```

3. In XAML, set the **Header** property of each **C1ToolbarGroup** to the following: **Clipboard** and **Font**. Your XAML should now look like the following:

```
<c1:C1Toolbar Grid.Row="1" Name="c1Toolbar1">
   <c1:C1ToolbarGroup Header="Clipboard"/>
   <c1:C1ToolbarGroup Header="Font"/>
</c1:C1Toolbar>
```

4. Select the **Clipboard C1ToolbarGroup** and click on the ellipsis button next to the **Items** collection editor.
5. From the **Select item** drop-down, click **Add** three times to add three **C1ToolbarButtons** to the collection.
6. Select the first **C1ToolbarButton** and expand the **Other** node in its properties window to set the **LabelTitle** property to **Paste** and set the other two to **Cut** and **Copy**, repectively.
7. As an optional step you can set the **LargeImageSource** and/or **SmallImageSource** properties to resources found within your project or add new images.
8. Click **OK** to close the **Items Collection** editor. Your XAML should now look like this:

```
<c1:C1Toolbar Name="c1Toolbar1">
    <c1:C1ToolbarGroup Header="Clipboard">
            <c1:C1ToolbarButton LabelTitle="Paste"
LargeImageSource="/ToolbarQuickstart;component/Images/Paste.png" />
            <c1:C1ToolbarButton LabelTitle="Cut"
SmallImageSource="/ToolbarQuickstart;component/Images/Cut.png" />
            <c1:C1ToolbarButton LabelTitle="Copy"
SmallImageSource="/ToolbarQuickstart;component/Images/Copy.png" />
        </c1:C1ToolbarGroup>
        <c1:C1ToolbarGroup Header="Font"/>
</c1:C1Toolbar>
```

Step 3 of 3: Adding a C1ToolbarStrip and C1ToolbarToggleButtons to C1ToolbarGroup

In this step you'll continue in Visual Studio by adding a C1ToolbarStrip to your 'Font' C1ToolbarGroup and then you will add C1ToolbarToggleButtons to the C1ToolbarGroup.

1. Select the **Font C1ToolbarGroup** and and add a **C1ToolbarStrip** in XAML.

```
<c1:C1ToolbarGroup Header="Font">
    <c1:C1ToolbarStrip />
</c1:C1ToolbarGroup>
```

   The rest of your XAML should appear like the following:

```
<c1:C1Toolbar Grid.Row="1" Name="c1Toolbar1"
    <c1:C1ToolbarGroup Header="Clipboard">
        <c1:C1ToolbarButton LabelTitle="Paste" />
        <c1:C1ToolbarButton LabelTitle="Cut" />
        <c1:C1ToolbarButton LabelTitle="Copy" />
    </c1:C1ToolbarGroup>
```

```
        <c1:C1ToolbarGroup Header="Font">
        <c1:C1ToolbarStrip />
        </c1:C1ToolbarGroup>
    </c1:C1Toolbar>
```

2. Select the C1ToolbarStrip and open its **Items** collection editor
3. Select the **C1ToolbarToggleButton** from the **Select item** dropdown and click **Add** three three times to add three C1ToolbarToggleButtons; set each **LabelTitle** property to: Bold, Italic, and Underline. Click OK to close the **Items** collection editor.

```
    <c1:C1ToolbarGroup Header="Font">
        <c1:C1ToolbarStrip>
            <c1:C1ToolbarToggleButton LabelTitle="Bold" />
            <c1:C1ToolbarToggleButton LabelTitle="Italic" />
            <c1:C1ToolbarToggleButton LabelTitle="Underline" />
        </c1:C1ToolbarStrip>
    </c1:C1ToolbarGroup>
```

Congratulations! You've now completed creating a toolbar UI using **Toolbar for WPF**.

Further topics:

- Adding Logic Behind the ToolbarButton Click Event (page 21) – This topic shows you how to use button click events to add logic behind buttons
- Commanding with C1Toolbar for Silverlight (page 23) – This tutorial demonstrates how to use C1Toolbar with commands in a Silverlight application
- Button Size and Text Position in C1ToolbarGroup (page 16) – This topic shows you how to use GroupSizeDefinitions of the C1ToolbarGroup.
- Toolbar Tab Item (page 13) – This topic shows you how to add the C1ToolbarTabItem control.

XAML Quick Reference

This section provides an example that show how to use the Silverlight Toolbar control with only XAML code.

EX: Add Items to the C1Toolbar

The following XAML code shows you how to add a C1ToolbarGroup, C1ToolbarStrip, and C1ToolbarButton to the C1Toolbar control:

```
<c1:C1Toolbar Name="c1Toolbar1">
 <c1:C1ToolbarTabControl>
  <c1:C1ToolbarTabItem Header="Home">
   <c1:C1ToolbarGroup Header="Clipboard">
```

```
      <c1:C1ToolbarButton LabelTitle="Paste"
LargeImageSource="/Images/Paste.png" />
      <c1:C1ToolbarButton LabelTitle="Cut"
SmallImageSource="/Images/Cut.png" />
      <c1:C1ToolbarButton LabelTitle="Copy"
SmallImageSource="/Images/Copy.png" />
   </c1:C1ToolbarGroup>
   <c1:C1ToolbarGroup Header="Font">
    <c1:C1ToolbarStrip>
       <c1:C1ToolbarToggleButton LabelTitle="Bold" />
       <c1:C1ToolbarToggleButton LabelTitle="Italic" />
       <c1:C1ToolbarToggleButton LabelTitle="Underline" />
    </c1:C1ToolbarStrip>
   </c1:C1ToolbarGroup>
  </c1:C1ToolbarTabItem>
 </c1:C1ToolbarTabControl>
</c1:C1Toolbar>
```

The following image shows how the C1Toolbar control will appear after adding the above XAML code:

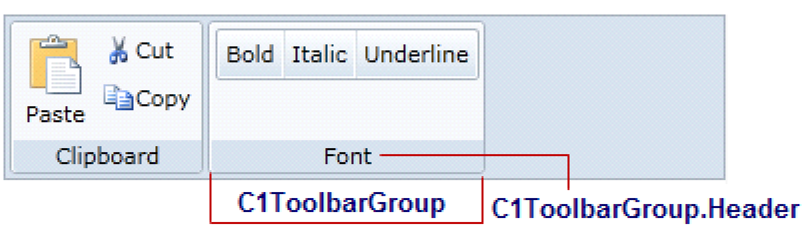
Toolbar Elements

C1Toolbar is a Silverlight control that is used as a container to hold other controls such as buttons, check buttons, text boxes, drop-down lists, split-buttons, and separators. C1ToolBar provides seven different objects to support the various types of controls used in the C1ToolBar: C1ToolbarButton, C1ToolbarDropDown, C1ToolbarGroup, C1ToolbarSplitButton, C1ToolbarStrip, C1ToolbarTabItem, C1ToolbarToggleButton,

Toolbar Group

The C1ToolBarGroup object defines a group of toolbar elements. The C1ToolbarGroup can hold the following toolbar controls: C1ToolbarButton, C1ToolbarToggleButton, C1ToolbarDropDown, and C1ToolbarSplitButton.

When you add toolbar items to the C1ToolbarGroup the child toolbar items appear grouped in a rectangular box like the following:



C1ToolbarGroups are typically used when a group of functions are mutually exclusive. That is, only one of the functions represented by the group of buttons can be on at a time.

The images of the controls (C1ToolbarButons and C1ToolbarToggleButtons) inside the C1ToolbarGroup can be all be sized to large, medium, or small using the GroupSizeDefinitions class.

The **GroupSizeDefinition** defines the control sizes as small, medium, or large and the position of the button text in a C1ToolbarGroup. If "large" is used then all the controls (C1ToolbarButons and C1ToolbarToggleButtons) in the group gets the image size from the LargeImageResource property and positions the text value of the LabelTitle property below the image. If "medium" is used then all the controls (C1ToolbarButons and C1ToolbarToggleButtons) in the group gets the image size from the SmallImageResource property and positions the text value of the LabelTitle property to the right of the image. If "small" is used then all the controls (C1ToolbarButons and C1ToolbarToggleButtons) in the group gets the image size from the SmallImageResource property and does not show the text.

EX: C1ToolbarGroup with a C1ToolbarStrip containing C1ToolbarButtons

```
</c1:C1ToolbarGroup>
<c1:C1ToolbarGroup Header="Font">
 <c1:C1ToolbarStrip>
    <c1:C1ToolbarToggleButton LabelTitle="Bold" />
    <c1:C1ToolbarToggleButton LabelTitle="Italic" />
    <c1:C1ToolbarToggleButton LabelTitle="Underline" />
 </c1:C1ToolbarStrip>
</c1:C1ToolbarGroup>
```

The C1ToolBarGroup includes the following properties:

Property	Definition
FocusBrush	Gets or sets the item that is located at a specified index of the collection.
GroupSizeDefinitions	Gets or sets the mode of selection for the items in the control.
Header	Gets or sets the header of the toolbar group.
MouseOverBrush	Gets or sets the Brush used to highlight the control when it has the mouse over.
PressedBrush	Gets or sets the Brush used to paint a button when it is pressed.
ShowDialogLauncher	Gets or sets the dialog launcher visibility.

Toolbar Button

The C1ToolBarButton object defines a toolbar button. A button can include text, image, or text and an image. Set text with the LabelTitle property, and an image with the **LargeImageSource** or **SmallImageSource** property for each C1ToolBarButton object.

The buttons can be added at design time using the **C1ToolbarItems** collection editor, programmatically using **C1ToolBarItemCollection.Add** or **C1ToolBarItemCollection.Remove** methods from the **C1ToolBarItemCollection** collection, or through XAML code..

The C1ToolBarButton includes the following unique properties:

Property	Definition
LabelTitle	Gets or sets the label title of control.
LargeImageSource	Gets or sets the large image source of the control.

MouseOverBrush	Gets or sets the Brush used to highlight the control when it has the mouse over
PressedBrush	Gets or sets the Brush used to paint a button when it is pressed.
SmallImageSource	Gets or sets the large image source of the control.

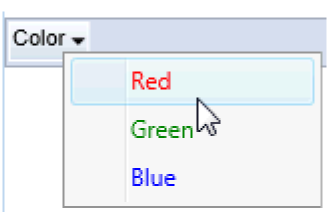
Toolbar DropDown

The C1ToolbarDropDown control represents a drop-down button on the C1ToolbarStrip. When clicking it displays popup panel with Content property or context menu set by Menu property. A drop-down button provides users with a list of options. When you click on a **C1ToolBarDropDown** button the **Click** event always fires and the drop down list appears.

Example 1: Drop-down with popup stack panel with buttons

```
<c1:C1ToolbarDropDown Padding="2" Header="Color">
   <c1:C1ToolbarDropDown.Content>
     <StackPanel Margin="2" Orientation="Horizontal">
       <Button Margin="2" Content="Red" Foreground="Red" />
       <Button Margin="2" Content="Green" Foreground="Green" />
       <Button Margin="2" Content="Blue" Foreground="Blue" />
     </StackPanel>
   </c1:C1ToolbarDropDown.Content>
</c1:C1ToolbarDropDown>
```

Example 2: Dropdown with popup menu




```
<c1:C1Toolbar Name="c1Toolbar1" Margin="0,127,0,160">
   <c1:C1ToolbarDropDown Padding="2" Header="Color">
     <c1:C1ToolbarDropDown.ContextMenu>
        <ContextMenu>
          <MenuItem Foreground="Red" Header="Red" IsCheckable="True" />
          <MenuItem Foreground="Green" Header="Green"
IsCheckable="True" />
          <MenuItem Foreground="Blue" Header="Blue" IsCheckable="True"
/>
         </ContextMenu>
     </c1:C1ToolbarDropDown.ContextMenu>
   </c1:C1ToolbarDropDown>
```

```
</c1:C1Toolbar></c1:C1ToolbarDropDown>
```

The C1ToolBarDropDown includes the following unique properties:

Property	Definition
ContentBackground	Gets or sets the content background.
Menu	Gets or sets the context menu which is displayed when the control is clicked.
MouseOverBrush	Gets or sets the Brush used to highlight the control when it has the mouse over.
PressedBrush	Gets or sets the Brush used to paint a button when it is pressed.

Toolbar SplitButton

C1ToolbarSplitButton control represents a drop-down split button on the C1ToolbarStrip.

It's similar to C1ToolbarDropDown but contains two clickable areas: the button area and the downward-pointing arrow.

When clicking on the rightmost part of the downward-pointing rectangle it displays a popup panel with the **Content** property or context menu set by the **ContextMenu** property. Clicking on the left part of button fires the **Click** event as in the standard button. Usually the **Click** event is used to perform the default or last action while the popup allows to select alternative options.

A solid vertical line dividing the image from the drop down arrow appears when you hover the cursor over the button like in the following image:

EX: Split button with popup menu

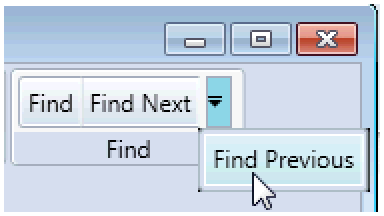
```
<c1:C1ToolbarSplitButton Padding="2" Header="Defaut"
   Click="SetDefaultStyle">
   <c1:C1ToolbarDropDown.Menu>
     <c1:C1ContextMenu>
       <c1:C1MenuItem Header="Heading 1" FontSize="14" />
       <c1:C1MenuItem Header="Heading 2" FontSize="12" />
       <c1:C1MenuItem Header="Title" FontWeight="Bold" />
       <c1:C1MenuItem Header="Subtitle" FontWeight="SemiBold"
         FontStyle="Italic"  />
       <c1:C1MenuItem Header="Quote" FontStyle="Italic" />
     </c1:C1ContextMenu>
   </c1:C1ToolbarDropDown.Menu>
</c1:C1ToolbarSplitButton>
```

Toolbar Strip

Represents a container for the toolbar controls and other C1 WPF controls. The C1ToolbarStrip is capable of hosting C1ToolbarButton, C1ToolbarToggleButton, C1ToolbarDropDown, C1ToolbarSplitButton, C1Separator, C1ComboBox, and C1TextBox controls.

The C1ToolbarStrip supports overflow depending on the available space in the Strip panel. The items will jump between the Strip panel and drop-down Overflow panel. This occurs automatically by default, but can be set to occur never, always, or as needed

The following imageIllustrates the overflow support the C1ToolbarStrip provides:



The C1ToolbarStrip can be added using XAML or programatically.

Ex: Toolbar strip with toggle buttons

```
<c1:C1ToolbarGroup Header="Font">
    <c1:C1ToolbarStrip>
        <c1:C1ToolbarToggleButton LabelTitle="Bold" />
        <c1:C1ToolbarToggleButton LabelTitle="Italic" />
        <c1:C1ToolbarToggleButton LabelTitle="Underline" />
    </c1:C1ToolbarStrip>
</c1:C1ToolbarGroup>
```

The C1ToolBarStrip includes the following unique properties:

Property	**Definition**
ButtonBackground	Gets or sets the Brush that will be assigned to the Background of the buttons inside the control.
ButtonForeground	Gets or sets the Brush that will be assigned to the Foreground of the buttons inside the control.
FocusBrush	Gets or sets the Brush used to highlight the focused control.
MouseOverBrush	Gets or sets the Brush used to highlight the control when it has the mouse over.
Orientation	Gets or sets the orientation of the toolbar strip.
Overflow	Gets or set the value that indicates how to handle the items which do not fit to the available space.
OverflowMenuItems	Gets the collection that contains elements of overflow menu.
OverflowPanel	Gets or sets the template for overflow panel.
PressedBrush	Gets or sets the Brush used to paint a button when it is pressed.

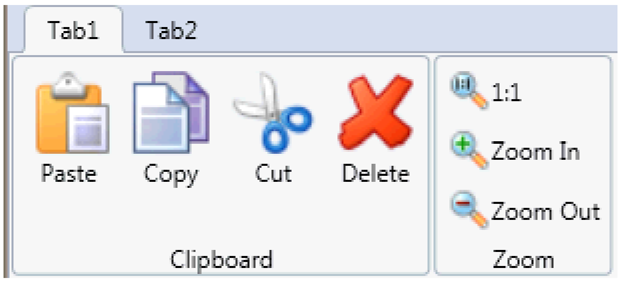
Toolbar Tab Item

The **C1ToolbarTablItem** represents a tab item on the C1Toolbar. The **C1ToolbarTabItem**s are beneficial to use when you have substantial amounts of information to display on minimum window space.

The C1ToolbarTabItem divides the controls onto a separate page and shows one tab at a time.

A **C1ToolbarTabItem** can hold a collection of C1ToolbarGroups which can contain C1ToolbarButtons, C1ToolbarToggleButtons, C1ToolbarDropDowns, and C1ToolbarSplitButtons.

The following image displays two **C1ToolbarTabItems**:



A **C1ToolbarTabItem** can be added to **C1Toolbar** via the **ToolbarItems** collection editor, through XAML, or programatically.

To add a **C1ToolbarTabItem** using the designer, complete the following:

1. Add a **C1Toolbar** control to your page.
2. Select the **C1Toolbar** control and click on the ellipsis button next to the **ToolbarItems** property in the **C1Toolbar** properties window.

   The **ToolbarItems** collection editor appears.
3. Select the **C1ToolbarTabItem** from the select item dropdown list and click **Add**.

   The **C1ToolbarTabItem** is added to the **C1Toolbar** control.
4. Set the **Header** property to **Tab 1**.

To add a **C1ToolbarTabItem** using XAML code, add the following:

```
<c1:C1Toolbar Grid.Row="1" Name="c1Toolbar1">
            <c1:C1ToolbarTabItem Header="Tab 1">
                <c1:C1ToolbarTabItem.Content>
                    <c1:C1ToolbarPanel />
                </c1:C1ToolbarTabItem.Content>
            </c1:C1ToolbarTabItem>
        </c1:C1Toolbar>
```

C1ToolBarGroups can be added to the **C1ToolbarTabItem** via the **Groups** collection editor, through XAML, or programatically using the Groups property.

To add a **C1ToolbarGroup** to the **C1ToolbarTabItem** using the designer, complete the following:

1. Add a **C1Toolbar** control to your page.
2. Right-click on the **C1ToolbarTabItem** and select **Properties**. Click on the **ellipsis** button next to the **Groups** property in the **C1ToolbarTabItem** properties window.

The **Groups** collection editor appears.

3. Click **Add** to add a **C1ToolbarGroup** to the **C1ToolbarTabItem**.

   The **C1ToolbarGroup** is added to the C1ToolbarTabItem.

4. Set the **Header** property to **Tab 1**.

To add a **C1ToolbarGroup** to a **C1ToolbarTabItem** using XAML code, add the following:

```
<c1:C1Toolbar Grid.Row="1" Name="c1Toolbar1">
            <c1:C1ToolbarTabItem Header="Tab 1">
                <c1:C1ToolbarTabItem.Content>
                    <c1:C1ToolbarPanel />
                </c1:C1ToolbarTabItem.Content>
                <c1:C1ToolbarGroup/>
            </c1:C1ToolbarTabItem>
        </c1:C1Toolbar>
```

The C1ToolBarTabItem includes the following unique property:

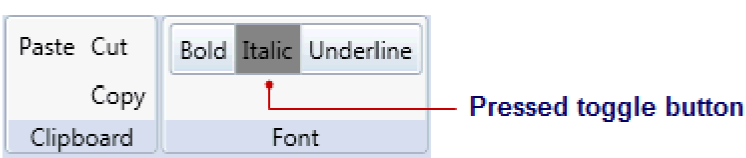
Property	Definition
Groups	Gets the collection of toolbar groups.

Toolbar ToggleButton

The C1ToolbarToggleButton represents a toggle button on the **C1ToolbarStrip**. It is a stateful button that enables users to toggle between on and off states. When clicking on the toggle button it remains activated, or pressed, until it is clicked again.

Toggle buttons display graphic or text like command buttons, but when it is pressed or activated the state appears in an on state. To indicate an on state you can set a color for the PressedBrush property.

The following image illustrates three C1ToolbarToggleButtons with the **Italic** toggle button pressed. The second toggle button, **Italic**, has the PressedBrush set to **Gray**.



C1ToolbarToggleButtons can be added to the C1ToolbarStrip via the **Items** collection editor, programmatically, or through XAML.

EX: Three C1TolbarToggleButtons with one of them having an on state

```
<c1:C1Toolbar Grid.Row="1" Name="c1Toolbar1">
  <c1:C1ToolbarGroup Header="Clipboard">
    <c1:C1ToolbarButton LabelTitle="Paste" />
    <c1:C1ToolbarButton LabelTitle="Cut" />
```

```
    <c1:C1ToolbarButton LabelTitle="Copy" />
  </c1:C1ToolbarGroup>
  <c1:C1ToolbarGroup Header="Font">
  <c1:C1ToolbarStrip>
     <c1:C1ToolbarToggleButton LabelTitle="Bold" />
     <c1:C1ToolbarToggleButton PressedBrush ="Gray" LabelTitle="Italic"
/>
     <c1:C1ToolbarToggleButton LabelTitle="Underline" />
     </c1:C1ToolbarStrip>
  </c1:C1ToolbarGroup>
</c1:C1Toolbar>
```

The C1ToolBarToggleButton includes the following properties:

Property	Definition
GroupName	Gets or sets the name that specifies which C1ToolbarToggleButton controls are mutually exclusive.
LabelTitle	Gets or sets the label title of control.
LargeImageSource	Gets or sets the large image source of the control.
MouseOverBrush	Gets or sets the Brush used to highlight the control when it has the mouse over.
PressedBrush	Gets or sets the Brush used to paint a button when it is pressed.
SmallImageSource	Gets or sets the small image source of the control.

Toolbar for Silverlight Layout and Appearance

The following topics detail how to customize the C1Toolbar control's layout and appearance. You can use built-in layout options to lay your controls out in panels such as Grids or Canvases.

You can customize the appearance of your toolbar items by using the stack panel. Inside the stack panel you can determine the orientation of your toolbar item. Additionally you can add an image and some text. The text's alignment, font style, and color can be modified.

```
<c1tb:C1Toolbar>
 <c1tb:C1ToolbarGroup Header="Clipboard">
   <c1tb:C1ToolbarStrip Padding="0">
     <c1tb:C1ToolbarButton Width="60"
HorizontalContentAlignment="Left">
        <StackPanel Orientation="Horizontal" >
         <Image Grid.Column="1" Source="Resources/cut.png" Margin="7 0
7 0"/>
         <TextBlock Text="Cut" Foreground= "DarkOrange"
VerticalAlignment="Center" TextAlignment="Center" />
           </StackPanel>
    </c1tb:C1ToolbarButton>
   </c1tb:C1ToolbarStrip>
```

```
    <c1tb:C1ToolbarStrip Padding="0">
       <c1tb:C1ToolbarButton Width="60"
HorizontalContentAlignment="Left" >
         <StackPanel Orientation="Horizontal">
       <Image Source="Resources/copy.png"  Margin="4 0 4 0"/>
    <TextBlock Text="Copy" VerticalAlignment="Center"
TextAlignment="Center"/>
         </StackPanel>
       </c1tb:C1ToolbarButton>
     </c1tb:C1ToolbarStrip>
   <c1tb:C1ToolbarStrip Padding="0">
      <c1tb:C1ToolbarButton Width="60"
HorizontalContentAlignment="Left">
           <StackPanel Orientation="Horizontal">
             <Image Source="Resources/paste.png"  Margin="4 0 4 0"/>
            <TextBlock Text="Paste" VerticalAlignment="Center"
TextAlignment="Center"/>
         </StackPanel>
       </c1tb:C1ToolbarButton>
     </c1tb:C1ToolbarStrip>
   </c1tb:C1ToolbarGroup>
</c1tb:C1Toolbar>
```

Toolbar Layout

Select from horizontal (default) or vertical orientations for your toolbar when you use the C1ToolbarStrip.

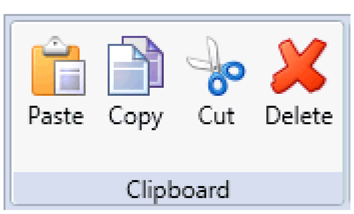
The strip panel in the C1Toolbar control can be rendered horizontally or vertically by setting its Orientation property. The layout for the individual child elements in the C1Toolbar can also be controlled using the **StackPanel.Orientation** property.

Button Size and Text Position in C1ToolbarGroup

You can define the same size (small, medium, or large) for all C1ToolbarButtons and C1ToolbarToggleButtons in a C1ToolbarGroup using the **GroupSizeDefinition**.

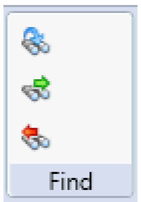
The **GroupSizeDefinition** defines the button sizes as small, medium, or large and the position of the button text in a C1ToolbarGroup. If "large" is used then all the buttons in the group gets the image size from the LargeImageResource property and positions the text value of the LabelTitle property below the image. If "medium" is used then all buttons in the group gets the image size from the SmallImageResource property and positions the text value of the LabelTitle property to the right of the image. If "small" is used then all the buttons in the group gets the image size from the SmallImageResource property and does not show the text.

The following image displays the buttons as Large with the **LargeImageResource** property used and the text for the **LabelTitle** property shown below the image:



The following image displays the buttons as Medium with the **SmallImageResource** property used and the text for the **LabelTitle** property shown to the right of the image:

The following image displays the buttons as Small with the **SmallImageResource** property used:



Toolbar for Silverlight Appearance Properties

ComponentOne Toolbar 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 C1Toolbar control.

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.
C1HierarchicalPresenter.Header	Gets or sets the item that labels the control.

Content Positioning Properties

The following properties let you customize the position of header and content area content in the C1Toolbar control.

Property	Description

Padding	Gets or sets the padding inside a control. This is a dependency property.
HorizontalAlignment	Gets or sets the horizontal alignment characteristics applied to this element when it is composed within a parent element, such as a panel or items control. This is a dependency property..
HorizontalContentAlignment	Gets or sets the horizontal alignment of the control's content. This is a dependency property.
VerticalAlignment	Gets or sets the vertical alignment characteristics applied to this element when it is composed within a parent element such as a panel or items control. 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.

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 **C1Toolbar** 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.

C1Toolbar 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 Toolbar 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 C1Toolbar control and, in the menu, selecting **Edit Template**. Select **Edit a Copy** to create an editable copy of the current template or **Create Empty**, to create a new blank template.

Once you've created a new template, the template will appear in the **Objects and Timeline** window. Note that you can use the Template property to customize the 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.

Additional Templates

In addition to the default template, the C1Toolbar control includes a few additional templates. These additional templates can also be accessed in Microsoft Expression Blend – in Blend select the C1Toolbar control and, in the menu, select **Edit Additional Templates**. Choose a template, and select **Create Empty**:

C1Toolbar Styles

ComponentOne Toolbar for Silverlight's **Toolbar** control provides several style properties that you can use to change the appearance of the control. Some of the included styles are described in the table below:

Style	Description
Style	Gets or sets the style used by this element when it is rendered. This is a dependency property.

Toolbar ClearStyle Properties

Toolbar for Silverlight supports ComponentOne's ClearStyle technology, which allows you to easily change control colors without having to change control templates. By setting a few color properties, you can quickly style the **C1Toolbar** elements. The supported properties for **C1Toolbar** are listed in the following table:

Property	Description
Background	Gets or sets the background used to fill the **C1Toolbar**.
MouseOverBrush	Gets or sets the brush used to highlight the control when the mouse is hovering over it.
PressedBrush	Gets or sets the System.Windows.Media.Brush used to highlight the buttons when they are clicked.
FocusBrush	Gets or sets the brush of the control when the control is focused.

Toolbar for Silverlight Samples

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

Note: ComponentOne samples are also available at http://helpcentral.componentone.com.

C# Toolbar Samples

Sample	Description
C1ToolbarStrip	Shows a lightweight toolstrip control which can have vertical and horizontal orientation.
Toolbar	Ribbon-like toolbar control with groups.

Toolbar 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 C1Toolbar control in general. If you are unfamiliar with the **ComponentOne Toolbar for Silverlight** product, please see the Toolbar for Silverlight Quickstart (page 4) first.

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

Each task-based help topic also assumes that you have created a new WPF project. For additional information on this topic, see Creating a .NET Project in Visual Studio or Creating a Microsoft Blend Project.

Aligning Toolbar Buttons

The following XAML shows how to align the toolbar buttons in the center of the group panel:

```
<c1:C1ToolbarGroup Header="Group" >
  <c1:C1ToolbarGroup.ItemsPanel>
   <ItemsPanelTemplate>
      <c1:C1ToolbarGroupPanel HorizontalAlignment="Center" />
    </ItemsPanelTemplate>
  </c1:C1ToolbarGroup.ItemsPanel>
```

Adding an Image to the Toolbar Button

You can use the LargeImageSource or SmallImageSource properties to add large or small images to your C1ToolbarButtons.

Using the Designer

1. Right-click the **C1ToolbarButton** control on your page and select **Properties**.
2. Locate the **LargelImageSource** property and click the **ellipsis** button next to it.

   The **Choose Image** dialog box appears.
3. Click the **Add** button and select the image you wish to add to it and click **OK**.

Adding Logic Behind the ToolbarButton Click Event

This topic shows how to add logic behind the C1ToolbarButton **Click** event for C1ToolbarButtons through XAML representation and then adding the code for the method in the Code Editor.

1. Add a **C1ToolbarButton** to the C1Toolbar element in the XAML editor and set the LabelTitle property to Search.

   ```
   <c1:C1Toolbar Grid.Row="1" Height="100" Name="c1Toolbar1">
      <c1:C1ToolbarButton LabelTitle="Search"
   </c1:C1Toolbar>
   ```

2. Add an attribute named **Click** to the **C1ToolbarButton** element in the XAML editor, and set its value to New Click. This is the name that you will give the event handler in code. Also give the **C1ToolbarButton** element a unique name and set its value to Search.

   ```
   <c1:C1Toolbar Grid.Row="1" Height="100" Name="c1Toolbar1">
       <c1:C1ToolbarButton LabelTitle="Search" Click="New_Click"
   Name="Search"/>
   </c1:C1Toolbar>
   ```

3. Right-click the designer and then click **View Code**.
4. Add the following event handler to the Window1 class. This code displays a message whenever you click the button.

   ```
   private void New_Click(object sender, RoutedEventArgs e)
   {
     MessageBox.Show("Event handler was created by clicking the " + Search);
   }
   ```

Run the application and observe:

The name of the toolbar button appears in the message box.

Commanding with C1Toolbar for Silverlight

Silverlight 4 supports the **ICommand** interface on any object that derives from the ButtonBase class. Any button, including **C1ToolbarButton**, can be associated with an object that implements ICommand through the control's Command property. Although the ICommand interface is supported, Silverlight does not provide any built-in implementation. ComponentOne Studio for Silverlight includes a couple implementations, **C1Command** and **C1ToolbarCommand**, so you do not have to write one yourself.

The C1Command class is included in the C1.Silverlight assembly. C1ToolbarCommand, included in the C1.Silverlight.Toolbar assembly, extends C1Command by adding a few extra toolbar related properties for labels and images.

The following steps demonstrate how to use commanding with C1Toolbar.

Using C1ToolbarCommand

1. Open or create a new Silverlight application.
2. Add two RowDefinitions to the default Grid element, giving the first row an automatic height, such as this:

```
<Grid.RowDefinitions>
    <RowDefinition Height="Auto" />
    <RowDefinition />
</Grid.RowDefinitions>
```

3. Add a **C1Toolbar** to the fill the first row.
4. Add a TextBox in the second row.
5. Paste the following **C1ToolbarCommand** to the UserControl Resources such as this:

```
<UserControl.Resources>
    <c1:C1ToolbarCommand x:Key="cmdClear" LabelTitle="Clear Text"
LargeImageSource="/Resources/delete.png" />
</UserControl.Resources>
```

   This command will be used to clear the contents from the TextBox. You can specify a LargeImageSource and a SmallImageSource for C1ToolbarCommand, although this is not required.

6. Paste the following XAML to fill C1Toolbar with a tab, group and button:

```
<c1:C1Toolbar Name="c1Toolbar1">
    <c1:C1ToolbarTabControl>
        <c1:C1ToolbarTabItem Header="Home">
            <c1:C1ToolbarGroup Header="Application">
                <c1:C1ToolbarButton
c1:CommandExtensions.Command="{StaticResource cmdClear}" />
            </c1:C1ToolbarGroup>
        </c1:C1ToolbarTabItem>
```

```
    </c1:C1ToolbarTabControl>
</c1:C1Toolbar>
```

Here we are setting the attached c1:CommandExtensions.Command property to our command. Note that C1ToolbarButton also has an inherited Command property. It is better to use the attached property CommandExtensions.Command to set the command when using C1Commands.

7. Add a TextBox to the page below the toolbar named "textBox1"
8. Next, you need to register the command after the page initializes with this code:

```
// register command methods

CommandManager.RegisterClassCommandBinding(GetType(), new
CommandBinding((C1ToolbarCommand)Resources["cmdClear"], Clear, CanClear));

// check the ability of registered commands to execute

CommandManager.InvalidateRequerySuggested();
```

The C1.Silverlight.CommandManager provides command related utility methods for registering commands. Here we pass event handlers for the Clear and CanClear

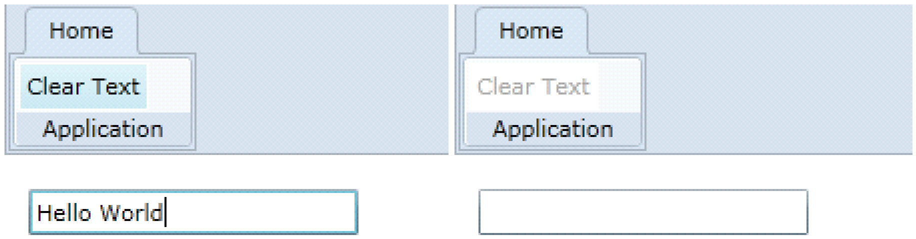
9. Paste the following Clear and CanClear event handlers which perform the logic for the command:

```
private void Clear(object sender, ExecutedRoutedEventArgs e)
{
    textBox1.Text = "";
}


private void CanClear(object sender, CanExecuteRoutedEventArgs e)
{
    if (textBox1.Text.Length > 0)
    {
        e.CanExecute = true;
    }
    else
        e.CanExecute = false;


}
```

10. In Silverlight you need to explicitly check the ability of registered commands in code. In the TextBox TextChanged event, add the following code:

```
private void textBox1_TextChanged(object sender, TextChangedEventArgs e)
{
    CommandManager.InvalidateRequerySuggested();
}
```

11. Run the sample and you can now clear the contents of the textbox using a command in C1Toolbar using C1Command.


Home
Clear Text
Application
Hello World
Home
Clear Text
Application