ComponentOne

OutlookBar for Silverlight

Table of Contents

ComponentOne OutlookBar for Silverlight

Group menus and controls into distinct categories with this side bar navigation system. **ComponentOne OutlookBar™ for Silverlight** mimics the Microsoft Outlook navigation pane. It consists of any number of categories represented by buttons where each button has a header as well as a panel for adding additional controls.

OutlookBar for Silverlight Features

The following are some of the main features of C1OutlookBar that you may find useful:

- **Microsoft Outlook-style UI**

  Group content and navigation menus into distinct categories just like the navigation system used in Microsoft Outlook 2007 and 2010. This model helps you organize content and allows end-users to navigate quickly.

- **Office 2007 and 2010 Themes**

  C1OutlookBar supports several different Office 2007 and 2010 themes including blue, black and silver. These themes work just like standard Silverlight themes. See the Themes (page 12) topic of this documentation for more information.

- **Expand/Collapse**

  Enable the **C1OutlookBar** control to collapse to the left or right of the page. See Expanding and Collapsing C1OutlookBar (page 8) for more information.

- **Stackable Buttons**

  Use the splitter bar to stack buttons into the collapsed item panel located at the bottom of the control. When buttons overflow they will appear in the drop-down menu for selection.

- **Flexible Data Binding**

  OutlookBar is an **ItemsControl** that can be bound to any list of objects of any data type using DataTemplates to create a visual representation of the items. See the Adding an Image to a C1OutlookItem (page 17) topic for an example.

- **Easily Change Colors with ClearStyle**

  OutlookBar supports **ComponentOne ClearStyle™ technology** which allows you to easily change control brushes without having to override templates. By just setting a few brush properties in Visual Studio you can quickly style each part of the accordion. See OutlookBar ClearStyle Properties (page 16) for more information on the ComponentOne ClearStyle technology.

OutlookBar for Silverlight Quick Start

The following quick start guide is intended to get you up and running with **OutlookBar for Silverlight**. In this quick start, you'll start in Visual Studio to create a new project, add a C1OutlookBar to your application, and then add *Inbox* and *Tasks* C1OutlookItems, which will look similar to the Microsoft Outlook navigation pane.

Step 1 of 3: Creating a Silverlight Application

In this step you'll create a Silverlight application using **ComponentOne OutlookBar for Silverlight**.

To set up your project and add a C1OutlookBar control to your application, complete the following steps:

1. In Visual Studio, select **File | New | Project**.
2. In the **New Project** dialog box, select a language in the left pane (in this example, C# is used), and in the templates list select **Silverlight Application**.
3. Enter a **Name** for your project and click **OK**. The N**ew Silverlight Application** dialog box will appear.

4. Uncheck the **Host the Silverlight application in a new Web site** box, if necessary, and click **OK**. The **MainPage.xaml** file should open.
5. In the XAML window of the project, place the cursor between the <Grid> and </Grid> tags and click once. Note that you cannot currently add Silverlight controls directly to the design area in Visual Studio, so you must add them to the XAML window as directed in the next step.
6. Navigate to the Toolbox and double-click the C1OutlookBar icon to add the C1OutlookBar to **MainPage.xaml**. The XAML markup will now look similar to the following:

```
<UserControl x:Class="OutlookBar.MainPage"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
    xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
    mc:Ignorable="d"

    d:DesignHeight="300" d:DesignWidth="400"
    xmlns:c1="http://schemas.componentone.com/winfx/2006/xaml"
xmlns:my="clr-namespace:C1.Silverlight;assembly=C1.Silverlight">

    <Grid x:Name="LayoutRoot" Background="White">
        <c1:C1OutlookBar c1:C1NagScreen.Nag="True">

        </c1:C1OutlookBar>
    </Grid>
</UserControl>
```

In the next step, you will add C1OutlookItem**s** to the C1OutlookBar.

Step 2 of 3: Adding C1OutlookItems

Next we are going to add C1OutlookItems to the C1OutlookBar. We'll add two items, an *Inbox* folder and a *Tasks* folder.

1. In the XAML markup, place your cursor between the `<c1:C1OutlookBar></c1:C1OutlookBar>` tags and press ENTER.
2. Add a C1OutlookItem control within these tags with **Header="Inbox"** using the following XAML markup:

```
<c1:C1OutlookItem Header="Inbox">
</c1:C1OutlookItem>
```

3. Next we'll add a **C1TreeView** containing folders that will appear in the C1OutlookBar. Add the following XAML within the `<c1:C1OutlookItem>` tags:

```
<c1:C1TreeView x:Name="contacts" BorderThickness="0" >
                   <c1:C1TreeViewItem Header="Inbox" IsExpanded="True">
                        <c1:C1TreeViewItem >
                            <c1:C1TreeViewItem.Header>
                                <StackPanel Orientation="Horizontal">
                                    <TextBlock Text="Folder1"/>
                                </StackPanel>
                            </c1:C1TreeViewItem.Header>
                        </c1:C1TreeViewItem>
                        <c1:C1TreeViewItem >
                            <c1:C1TreeViewItem.Header>
                                <StackPanel Orientation="Horizontal">
                                    <TextBlock Text="Folder2"/>
                                </StackPanel>
                            </c1:C1TreeViewItem.Header>
                        </c1:C1TreeViewItem>
                    </c1:C1TreeViewItem>
                </c1:C1TreeView>
```

4. Let's add another C1OutlookItem with a **C1TreeView** to display a **Tasks** folder. Add the following XAML markup after the closing `</c1:C1OutlookItem>` tag.

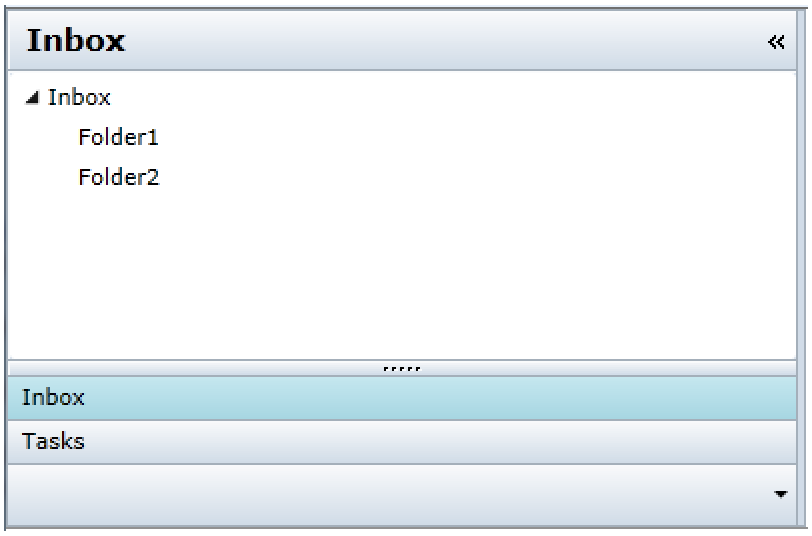
```
<c1:C1OutlookItem x:Name="selectedItem" Header="Tasks" >

                    <c1:C1TreeView x:Name="tasks" BorderThickness="0" >
                        <c1:C1TreeViewItem Header="My Tasks"
IsExpanded="True" >
                            <c1:C1TreeViewItem >
                                <c1:C1TreeViewItem.Header>
                                    <StackPanel Orientation="Horizontal">
                                      <TextBlock Text="To-Do List"/>
                                    </StackPanel>
                                </c1:C1TreeViewItem.Header>
                            </c1:C1TreeViewItem>
                            <c1:C1TreeViewItem >
                                <c1:C1TreeViewItem.Header>
                                    <StackPanel Orientation="Horizontal">

                                        <TextBlock Text="Tasks"/>
                                    </StackPanel>
                                </c1:C1TreeViewItem.Header>
                            </c1:C1TreeViewItem>
                        </c1:C1TreeViewItem>
                    </c1:C1TreeView>
                </c1:C1OutlookItem>
```

In the next step you will run the application.

Step 3 of 3: Running the Application

Now that you've created an application with a C1OutlookBar, you're ready to run it. Complete the following steps:

1. From the **Debug** menu, select **Start Debugging** to view how your application will appear at run time. Your application will look similar to the following:



2. Click the *Tasks* C1OutlookItem to switch to that item.

Congratulations! You have successfully completed the **OutlookBar for Silverlight** quick start.

XAML Quick Reference

This topic is dedicated to providing a quick overview of the XAML used to create a C1OutlookBar and C1OutlookItem.

To get started developing, add a **c1** namespace declaration in the root element tag:

```
xmlns:c1="http://schemas.componentone.com/winfx/2006/xaml"
```

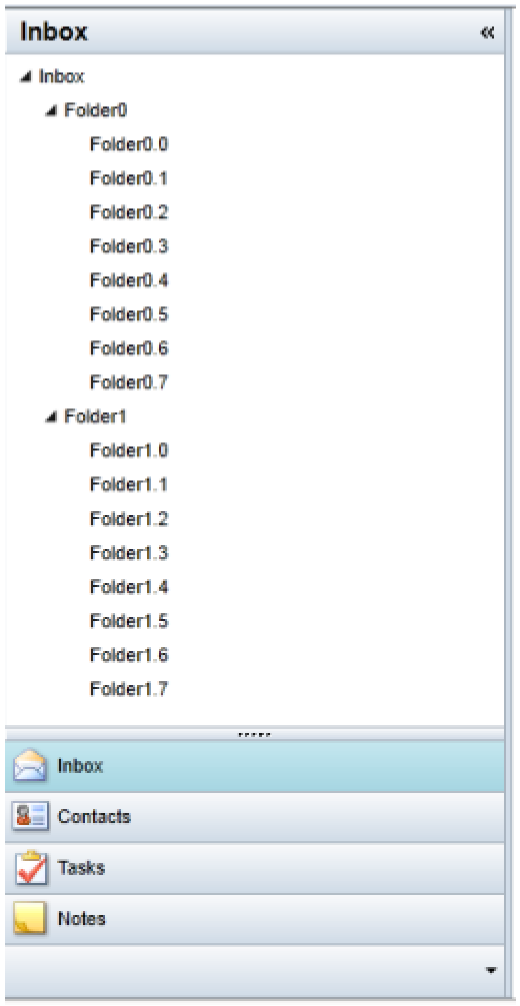
Here is an example of a very basic C1OutlookBar with multiple C1OutlookBar items.

```
<c1:C1OutlookBar>
            <c1:C1OutlookItem />
            <c1:C1OutlookItem />
            <c1:C1OutlookItem />

</c1:C1OutlookBar>
```

Below is the XAML for a more detailed example of an C1OutlookBar with multiple C1OutlookItems containing icons:

```
  <c1:C1OutlookBar IsExpanded="True" FontFamily="Arial" MinWidth="36"
ExpandedWidth="300" c1:C1NagScreen.Nag="True">

      <!-- Inbox -->
        <c1:C1OutlookItem Header="Inbox"
LargeIcon="Resources/Inbox24.png" SmallIcon="Resources/Inbox.png"
c1:C1NagScreen.Nag="True">
            <userControls:Inbox />
        </c1:C1OutlookItem>

        <!-- Contacts -->
        <c1:C1OutlookItem Header="Contacts"
LargeIcon="Resources/Contacts24.png" SmallIcon="Resources/Contacts.png"
c1:C1NagScreen.Nag="True">
            <userControls:Contacts />
        </c1:C1OutlookItem>

        <!-- Tasks -->
        <c1:C1OutlookItem Header="Tasks"
LargeIcon="Resources/Tasks24.png" SmallIcon="Resources/Tasks.png"
c1:C1NagScreen.Nag="True">
            <userControls:Tasks />
        </c1:C1OutlookItem>

        <!-- Notes -->
        <c1:C1OutlookItem Header="Notes"
LargeIcon="Resources/Notes24.png" SmallIcon="Resources/Notes.png"
c1:C1NagScreen.Nag="True">
            <userControls:Notes />
        </c1:C1OutlookItem>
    </c1:C1OutlookBar>
```

This XAML results in the following:

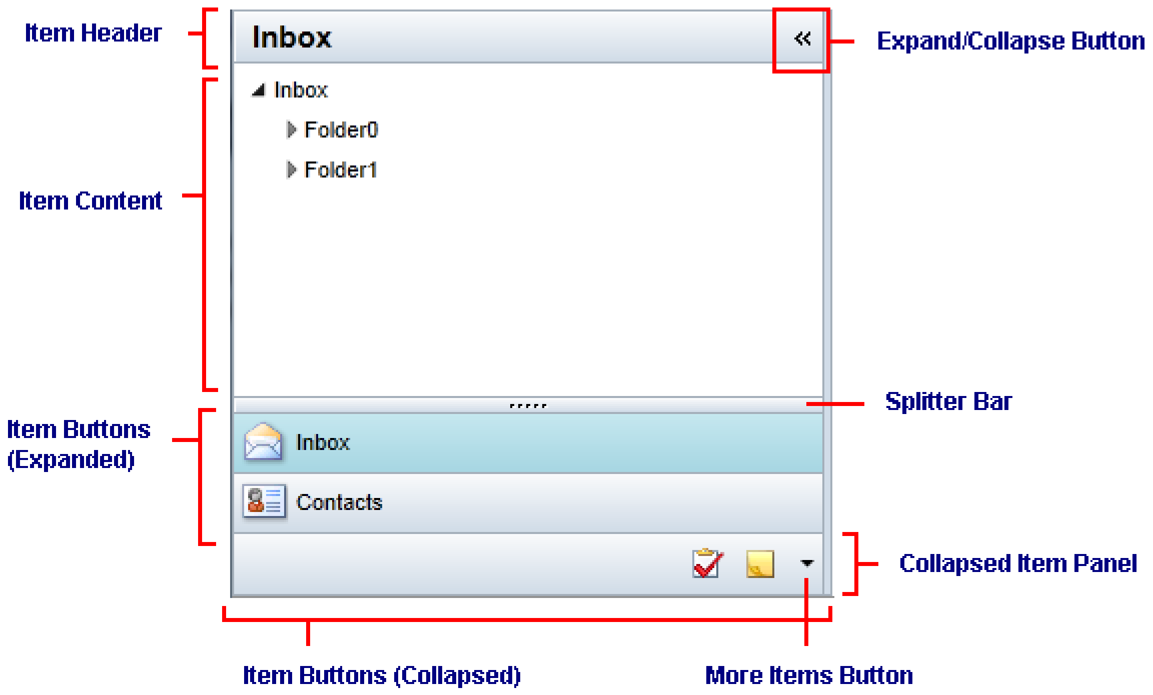


Working with OutlookBar for Silverlight

ComponentOne OutlookBar for Silverlight includes the C1OutlookBar control, which provides a side bar navigation system similar to Microsoft Outlook. The following topics provide more information on the C1OutlookBar and the elements that can be added to it.

C1OutlookBar Elements

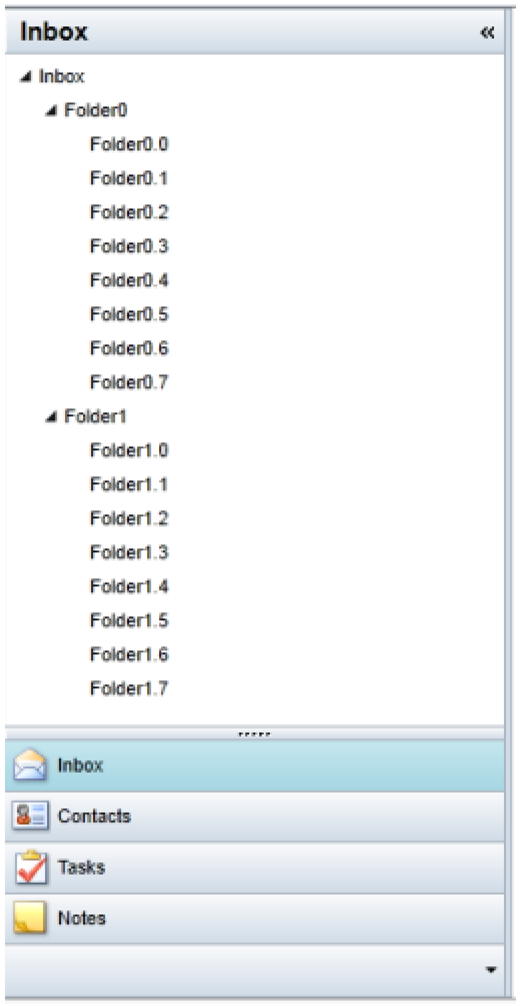
The C1OutlookBar serves as a container in which you can add C1OutlookItem**s** with icons to create a simple navigation system. The following elements make up the C1OutlookBar:

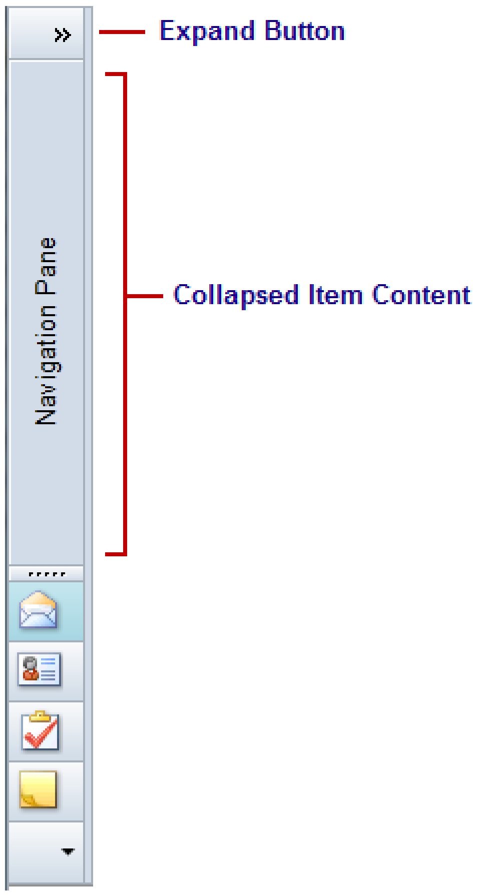


Expanding and Collapsing C1OutlookBar

OutlookBar for Silverlight provides the ability to expand and collaspe a C1OutlookBar so that you make room for other controls or elements. Here are examples of an expanded and collapsed C1OutlookBar:

Expanded **Collapsed**



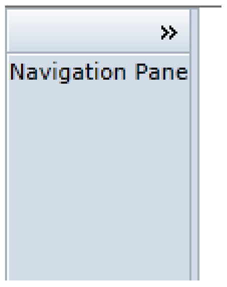


Collapsed Content

When a C1OutlookBar is collapsed, the central content region can display customized collapsed content. In Microsoft Outlook 2007-2010, this area is simply vertical text that reads "**Navigation Pane**". This region can be associated with a selected item for added flexibility, but this is not a requirement. There are two options for showing collapsed content.

Option A: Use the CollapsedContent property to display text when the C1OutlookBar is collapsed.

```
<c1:C1OutlookBar CollapsedContent = "Navigation Pane" />
```

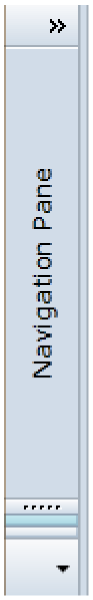


Option B: Use the CollapsedContent property to display the same collasped content no matter which item is selected.

```
<c1:C1OutlookBar>
   <c1:C1OutlookBar.CollapsedContent>
         …
   </c1:C1OutlookBar.CollapsedContent>
</c1:C1OutlookBar>
```

For example, see the following XAML displays "**Navigation Pane**" vertically.

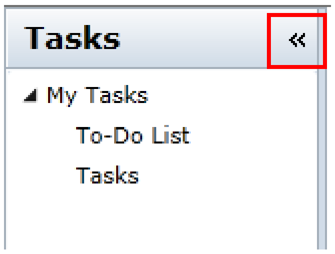
```
<c1:C1OutlookBar.CollapsedContent>
                <c1:C1LayoutTransformer>
                    <c1:C1LayoutTransformer.LayoutTransform>
                        <RotateTransform Angle="270" />
                    </c1:C1LayoutTransformer.LayoutTransform>
                    <TextBlock FontSize="13" TextAlignment="Center"
VerticalAlignment="Center"
                            Text="Navigation Pane" />
                </c1:C1LayoutTransformer>
            </c1:C1OutlookBar.CollapsedContent>
```



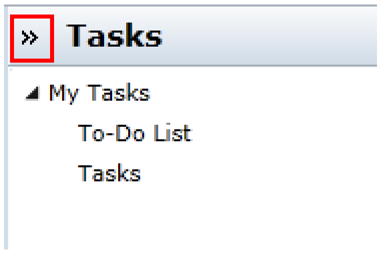
Collapsing Direction

C1OutlookBar supports collapsing in both left (default) and right directions. When the C1OutlookBar is set to dock **Left**, the default, the collasping arrow appears on the right side of the control. When the C1OutlookBar is set to dock **Right**, the collasping arrow appears on the left side of the control.

Dock = **Left**



Dock = Right



To specify which direction to collapse the C1OutlookBar, use the Dock property.

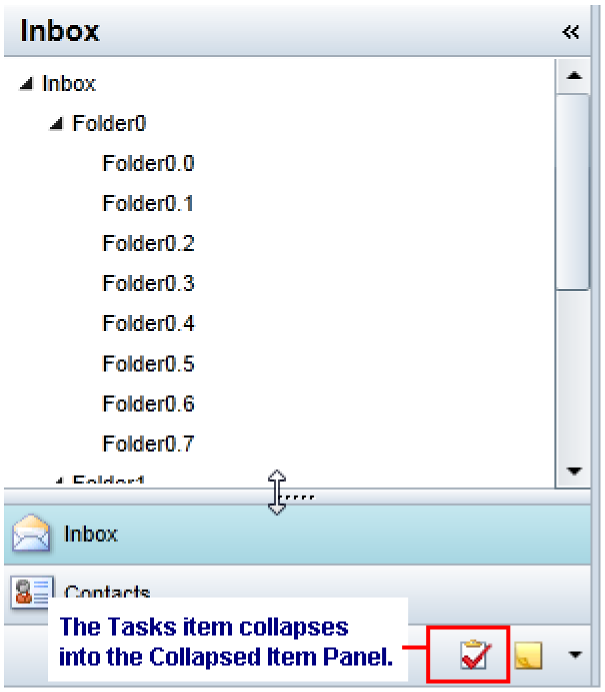
```
<c1:C1OutlookBar x:Name="C1OutlookBar1"  Dock="Right">
```

You can also set Dock property in the Visual Studio Properties window.

Collapsing Items in C1OutlookBar

Collapsing items is different than collapsing the C1OutlookBar control. C1OutlookBar provides a draggable splitter bar between the **Item Content** region and the **Item Buttons** region. When the splitter bar is dragged downward, items collapse into the **Collapsed Item Panel**.

Notice in the following image, when the splitter bar is dragged downward, the **Tasks** item collapses into the collapsed item panel.



You can have the C1OutlookBar open with items appearing in the **Item Buttons** and certain items collapsed in the **Collapsed Item Panel**. Use the FirstOverflowIndex property to determine which items should be sent to the **Collapsed Item Panel** based on their index.

For example, in the previous image, the **Tasks** item was the third item in the C1OutlookBar. If we set the FirstOverflowIndex property to 2, any items after the first two, which are considered the overflow items, are sent to the **Collapsed Item Panel**.

You can set the FirstOverflowIndex property in XAML:

```
<c1:C1OutlookBar FirstOverflowIndex="2">
```

Or set it in the Visual Studio properties window.

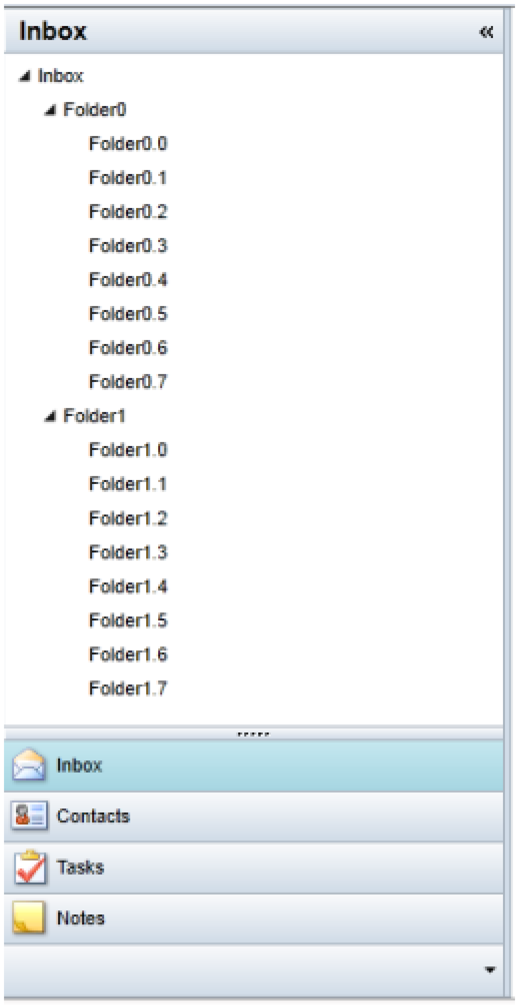
OutlookBar for Silverlight Layout and Appearance

The following topics detail how to customize the **C1OutlookBar's** layout and appearance. You can use templates to format and layout the control and to customize the control's actions. Themes allow you to customize the appearance of the control and take advantage of Silverlight's XAML-based styling.

Themes

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.

When C1OutlookBar is added to your project, it appears with the default theme, which looks similar to the following image:



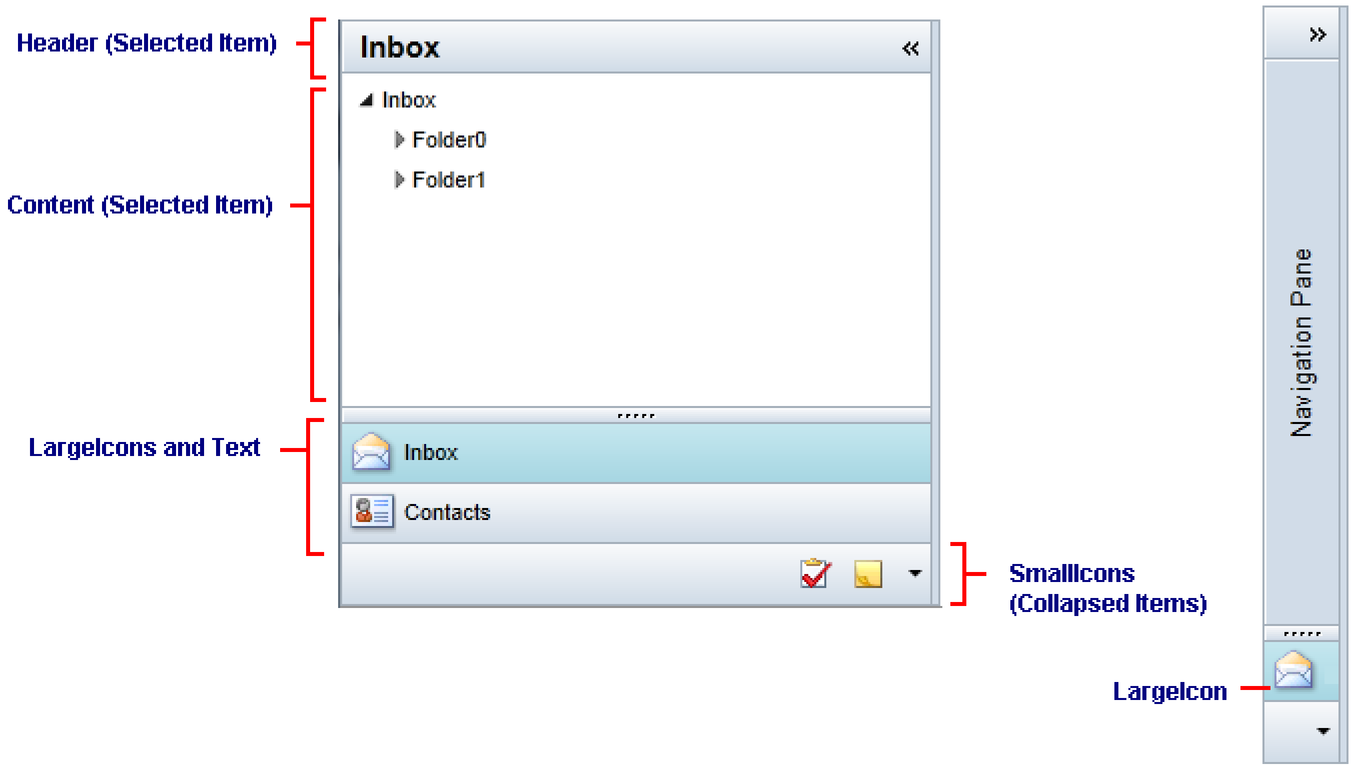
C1OutlookBar supports Office 2010 and Office 2007 themes. It can be themed with one of the many included Silverlight themes: Office2010 (*C1ThemeOffice2010Blue*, *C1ThemeOffice2010Black*, and *C1ThemeOffice2010Silver*), Office2007 *(C1ThemeOffice2007Blue*, *C1ThemeOffice2007Black*, and *C1ThemeOffice2007Silver*), *C1ThemeBureauBlack, C1ThemeExpressionDark, C1ThemeExpressionLight, C1ThemeCosmopolitan, C1ThemeRanierOrange, C1ThemeShinyBlue, and C1ThemeWhistlerBlue*.

Templates

You can customize the look of each C1OutlookItem by setting some of the basic C1OutlookBar and C1OutlookItem properties. Each of these properties has a corresponding **Template** property that binds them together.

Property	**Correpsonding Template Property**	**Description**
Header	HeaderTemplate	When the bar is expanded, the **Header** is shown to the right of the **LargeIcon** and above the content in the Office 2007 style.

Content	ContentTemplate	Gets or sets the content to display in the **Item Content** section when this item is selected.
LargeIcon	LargeIconTemplate	**LargeIcon** represents the **C1OutlookItem** image that appears in the vertical stack when **C1OutlookBar** is collapsed.
SmallIcon	SmallIconTemplate	Items above the FirstOverflowIndex or items that do not fit in the **Item Buttons** list are placed in a single horizontal line below the **LargeIcons** using **SmallIcon**.



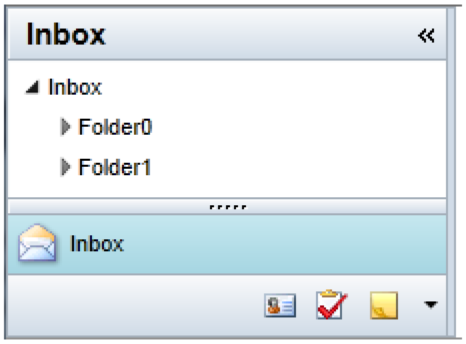
Note: A menu that uses a **SmallIcon** and **Header** allows hiding and showing individual items. Because the properties are displayed in several places, do not assign them a UIElement.

Here's an example of a C1OutlookBar containing a C1OutlookItem that has "**Inbox**" as the **Header** and a small and large icon bound through the corresponding templates.

```
<c1:C1OutlookBar.LargeIconTemplate>
            <DataTemplate>
                <Image Source="{Binding}" Width="24" Height="24" />
            </DataTemplate>
        </c1:C1OutlookBar.LargeIconTemplate>

<c1:C1OutlookBar.SmallIconTemplate>
            <DataTemplate>
                <Grid Height="24">
                    <Image Source="{Binding}" Width="16" Height="16" />
                </Grid>
            </DataTemplate>
        </c1:C1OutlookBar.SmallIconTemplate>
```

```
<c1:C1OutlookItem Header="Inbox" LargeIcon="Resources/Inbox24.png"
SmallIcon="Resources/Inbox.png">

            <userControls:Inbox />

        </c1:C1OutlookItem>
```

The C1OutlookBar looks like this:



ComponentOne ClearStyle Technology

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

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

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

How ClearStyle Works

Each key piece of the control's style is surfaced as a simple color property. This leads to a unique set of style properties for each control. For example, a **Gauge** has **PointerFill** and **PointerStroke** properties, whereas an **C1OutlookBar** has **SelectedBackground** and **MouseOverBrush**.

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

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

OutlookBar ClearStyle Properties

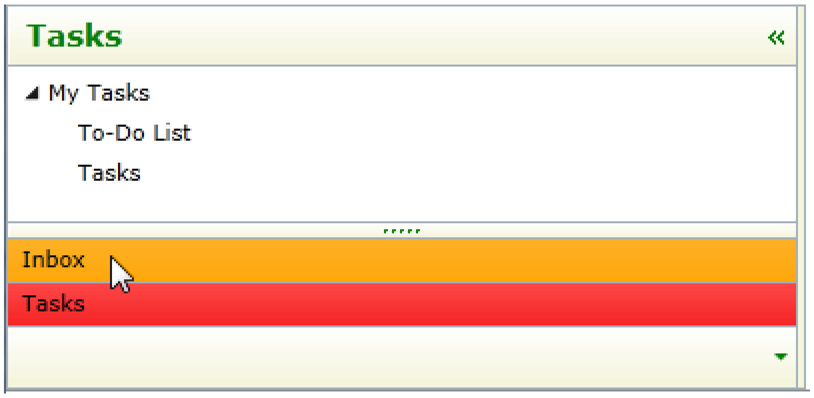
OutlookBar for WPF supports ComponentOne's ClearStyle technology, which allows you to easily change control colors without having to change control templates. By setting a few color properties, you can quickly style the C1OutlookBar elements. The supported properties for C1OutlookBar are listed in the following table:

Property	Description
Background	Gets or sets the background used to fill the **C1OutlookBar**.
Foreground	Gets or sets the foreground used to fill the **C1OutlookBar**.
MouseOverBrush	Gets or sets the brush used to highlight an item when the mouse is hovering over it.
SelectedBackground	Gets or sets the brush used to fill the selected C1OutlookItem in the **C1OutlookBar**.

You can completely change the appearance of the **C1OutlookBar** by setting these properties. For example, if you set the ItemBackground property to **Beige**, ItemForeground to **Green**, MouseOverBrush to **Orange**, and SelectedBackground to **Red** so the XAML markup appears similar to the following:

```
  <c1:C1OutlookBar Background="Beige" Foreground="Green"
MouseOverBrush="Orange" SelectedBackground="Red"
x:Name="C1OutlookBar1">
```

The C1OutlookBar will look similar to this:



OutlookBar 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 C1OutlookBar in general. If you are unfamiliar with the **ComponentOne OutlookBar for Silverlight** product, please see the OutlookBar for Silverlight Quick Start (page 3) first.

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

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

Adding an Image to a C1OutlookItem

C1OutlookBar is an **ItemsControl** that can be bound to any list of objects of any data type using DataTemplates to create a visual representation of the items. In this example, you'll see how easily you can add an image to a C1OutlookItem by binding through a **DataTemplate**.

Suppose you have a C1OutlookBar named **C1OutlookBar1** which contains a C1OutlookItem with the **Header** = **Mail**. You will also need an image for the C1OutlookItem. In this example, we'll use a *Resources* folder containing images in our project. Here's how the XAML might look:

```
<c1:C1OutlookBar Height="274" HorizontalAlignment="Left"
Margin="56,12,0,0" Name="c1OutlookBar1" VerticalAlignment="Top"
Width="274">
           <c1:C1OutlookItem Header="Mail"/>
</c1:C1OutlookBar>
```

1. Add XAML for a SmallIconTemplate and LargeIconTemplate for the images that will be shown in the **Collasped Item Panel** and for the **Item Button**, respectively. This markup should be placed within the C1OutlookBar tags.

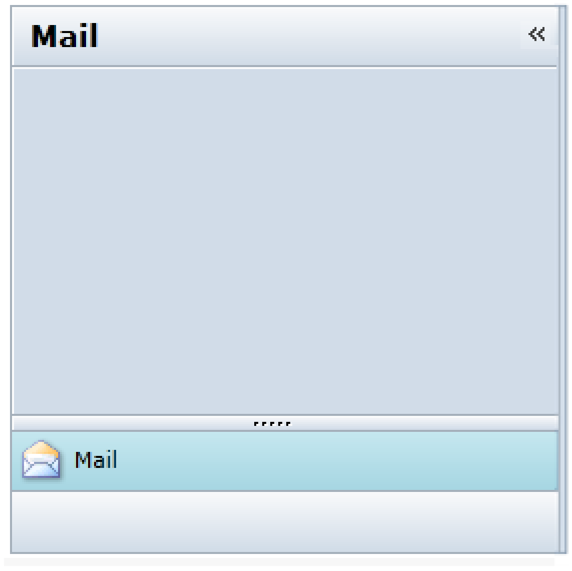
```
<c1:C1OutlookBar.LargeIconTemplate>
         <DataTemplate>
             <Image Source="{Binding}" Width="24" Height="24" />
         </DataTemplate>
</c1:C1OutlookBar.LargeIconTemplate>
<c1:C1OutlookBar.SmallIconTemplate>
         <DataTemplate>
             <Grid Height="24">
               <Image Source="{Binding}" Width="16" Height="16" />
             </Grid>
         </DataTemplate>
</c1:C1OutlookBar.SmallIconTemplate>
```

2. Add XAML to the C1OutlookItem specifying the SmallIcon and LargeIcon bound to the DataTemplates so the markup should now look like this:

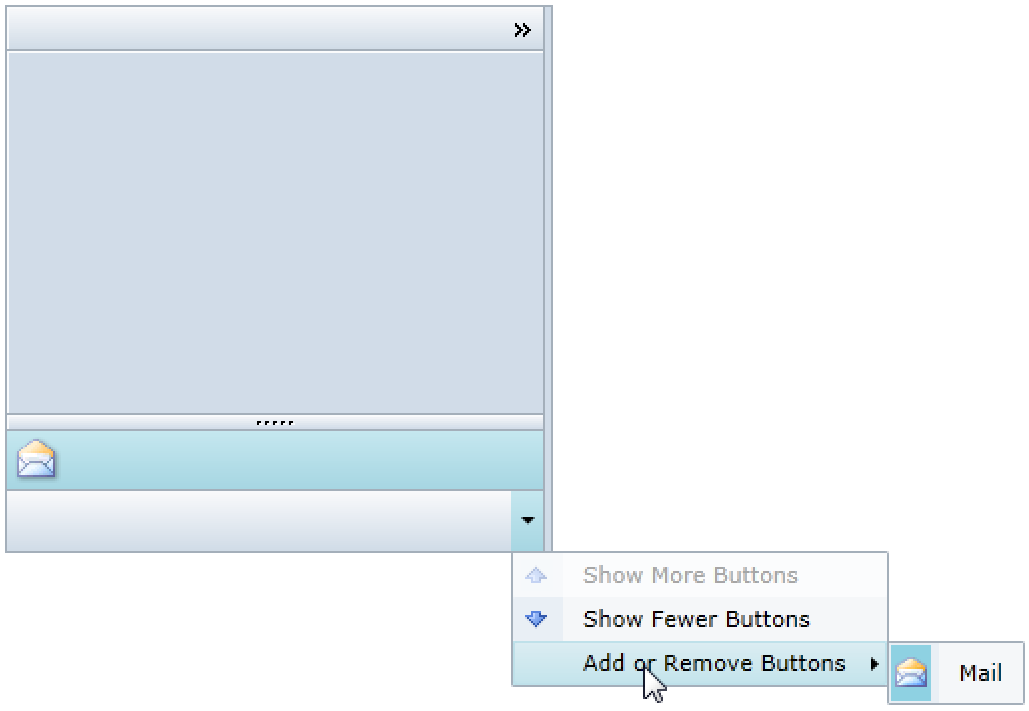
```
<c1:C1OutlookItem LargeIcon="Resources/Inbox24.png"
SmallIcon="Resources/Inbox.png" Header="Mail"/>
```

3. Run the project and notice when the C1OutlookBar is collasped, you will see the small icon in the **Collapsed Item Panel**. When the C1OutlookBar is expanded, the large icon will appear in the **Item Buttons (Expanded)** area.

Expanded



Collapsed



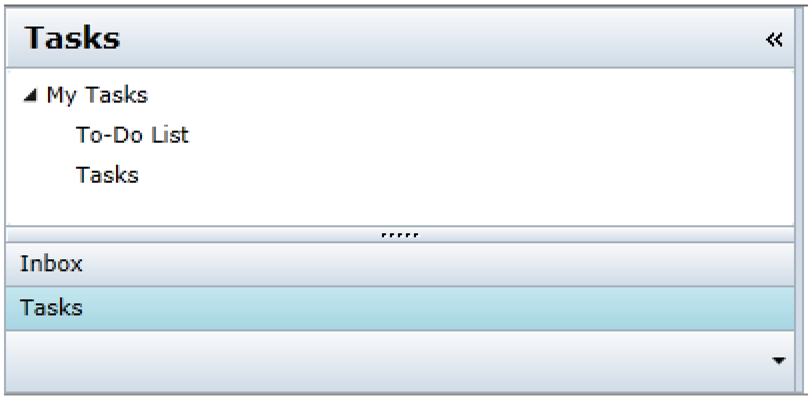
Selecting an Item in the C1OutlookBar

C1OutlookBar supports single item selection and allows you to programmatically select an item, giving you control over which item is selected when a user first views the C1Outlookbar.

Suppose you have a C1OutlookBar named **C1OutlookBar1** which contains several C1OutlookItem**s**, one of which is named *selectedItem*:

```
<c1:C1OutlookBar x:Name="C1OutlookBar1 ">
      <c1:C1OutlookItem x:Name="selectedItem" Header="Tasks">
```

1. Choose **View | Code** from the Visual Studio menu.
2. Add the following code to your project:
```
using C1.Silverlight.OutlookBar;

namespace C1OutlookBar
{
    public partial class MainPage : UserControl
    {
        public MainPage()
        {
            InitializeComponent();

            C1OutlookBar1.SelectedItem = (C1OutlookItem)selectedItem;
        }
    }
}
```
3. Run the project and notice the C1OutlookItem named **selectedItem**, or in this example the **Tasks** item, is selected.


Tasks
My Tasks
To-Do List
Tasks
Inbox
Tasks