ComponentOne

TileView 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 TileView for Silverlight Overview

Interactively browse through your data with **ComponentOne TileView for Silverlight™**. Expand and collapse tiles to view more or less information. Show-off the true power of Silverlight in your apps with this highly visual and interactive control. Create dashboards, detail views, photo galleries and more!

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:

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

- **Expand and Collapse Tiles**

  C1TileView provides complete control over the state of each tile. Tiles can be expanded (maximized) or collapsed (minimized). Display more or less information depending on the state of each tile.

- **Minimize Position**

  Minimize items to the top, left, bottom or right side of **C1TileView** by just setting one property. You can also specify the number of rows and columns when in the default state.

- **Data Binding**

  C1TileView is an items control which can be bound to any collection of business objects. Design different data templates to determine the amount of data viewable in each state.

- **Virtualization**

C1TileView has UI virtualization support so it can load hundreds of items instantly.

- **Drag-and-drop Interface**

  Users can drag-and-drop tiles to rearrange the order when the tiles are in the default state.

- **Animation**

  C1TileView provides built-in animation effects when expanding and collapsing tiles.

- **Silverlight Toolkit Themes**

  Add style to your UI with built-in support for the most popular Microsoft Silverlight Toolkit themes, including ExpressionDark, ExpressionLight, WhistlerBlue, RainerOrange, ShinyBlue, BureauBlack and Cosmopolitan.

TileView for Silverlight Quick Start

The following quick start guide is intended to get you up and running with **ComponentOne TileView for Silverlight**. In this quick start you'll create a simple project using a C1TileView control. You'll create a new Silverlight application, add the C1TileView control to your application, add content that will be displayed in the C1TileView control, and observe some of the run-time interactions possible with **TileView for Silverlight**.

Step 1 of 3: Creating the C1TileView Application

In this step you'll create a Silverlight application using **TileView for Silverlight**. When you add a C1TileView control to your application, you'll have a interface that you can display content in. To set up your project and add a C1TileView control to your application, complete the following steps:

1. Create a new Silverlight project in Visual Studio. In this example the application will be named "QuickStart". If you name the project something else, in later steps you may need to change references to "QuickStart" with the name of your project.
2. In the Solution Explorer, right-click the project name and choose **Add Reference**. In the **Add Reference** dialog box, locate and select the **C1.Silverlight** and **C1.Silverlight.TileView** assemblies and click **OK** to add references to your project.
3. Open the XAML view of the MainPage.xaml file; in this quick start you'll add the C1TileView control using XAML markup.
4. Add the XAML namespace to the UserControl tag with the following markup: `xmlns:c1="http://schemas.componentone.com/winfx/2006/xaml"`.

   The UserControl tag will now appear similar to the following:

   ```
   <UserControl x:Class="C1SilverlightCS111010.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"
   xmlns:c1="http://schemas.componentone.com/winfx/2006/xaml"
   mc:Ignorable="d" d:DesignHeight="262" d:DesignWidth="399">
   ```

   This is a unified namespace that will enable you to work with most ComponentOne WPF or Silverlight controls without adding multiple namespaces.
5. Add the `<c1:C1TileView x:Name="C1TileView1" />` tag within the Grid tags on the page to add the C1TileView control to the application.

   The XAML will appear similar to the following:

   ```
   <Grid>
       <c1:C1TileView x:Name="C1TileView1"></c1:C1TileView>
   </Grid>
   ```

   This will add a C1TileView control named "C1TileView1" to the application.

You've successfully set up your application's user interface, but if you run your application now you'll see that the C1TileView control currently contains no content. In the next steps you'll add content to the C1TileView control, and then you'll observe some of the run-time interactions possible with the control.

Step 2 of 3: Customizing the C1TileView Control

In the previous step you created a Silverlight application and added the C1TileView control to your project. To customize your application, complete the following steps:

1. Add `AllowDrop="True"` within the **C1TileView** tags on the page to allow users to perform drag-and-drop operations with items in the control. The XAML markup will appear similar to the following:

```
<c1:C1TileView x:Name="C1TileView1" AllowDrop="True"></c1:C1TileView>
```

2. Add three **C1TileViewItem**s within the **C1TileView** tags so the XAML markup appears similar to the following:

```
<c1:C1TileView Name="C1TileView1" AllowDrop="True">
    <c1:C1TileViewItem></c1:C1TileViewItem>
    <c1:C1TileViewItem></c1:C1TileViewItem>
    <c1:C1TileViewItem></c1:C1TileViewItem>
</c1:C1TileView>
```

3. Add **Background** and **Header** properties to each of the **C1TreeViewItem**s, so the markup appears like the following:

```
<c1:C1TileView Name="C1TileView1" AllowDrop="True">
    <c1:C1TileViewItem Background="Red" Header="Red"></c1:C1TileViewItem>
    <c1:C1TileViewItem Background="Blue"
Header="Blue"></c1:C1TileViewItem>
    <c1:C1TileViewItem Background="Yellow"
Header="Yellow"></c1:C1TileViewItem>
</c1:C1TileView>
```

   Each item will now appear a different color and have text in the header.

In this step you added content to the C1TileView control. In the next step you'll view some of the run-time interactions possible in the control.

Step 3 of 3: Running the C1TileView Application

Now that you've created a Silverlight application and customized the C1TileView control, the only thing left to do is run your application. To run your application and observe **TileView for Silverlight**'s run-time behavior, complete the following steps:

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

   The application will appear similar to the following:

Notice that the **C1TileView** control appears with three **C1TileViewItem**s.

2. Click on the red item's header and drag the item to towards the blue item. The items will trade places.
3. Click on the maximize icon in the upper-right corner of the yellow item's header. Note that the other two items are minimized:

Congratulations! You've completed the **TileView for Silverlight** quick start and created a simple Silverlight application, added and customized a **TileView for Silverlight** control, and viewed some of the run-time capabilities of the control.

Working with TileView for Silverlight

ComponentOne TileView for Silverlight includes the C1TileView control, a panel that slows you to interactively browse through your data. When you add the C1TileView control to a XAML window it exists as a blank container control that can be customized and include loaded content.

TileViewItem Elements

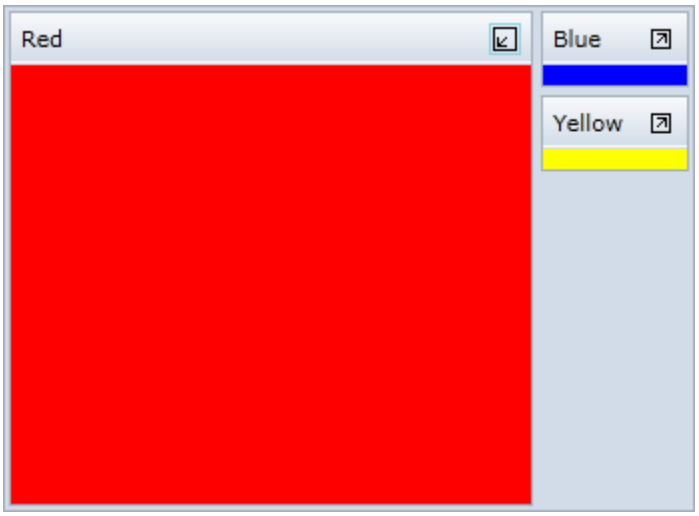
The C1TileViewItem control consists of two parts: a header and a content area. The image below identifies the toolbar and content area:

Any content that you add to the C1TileViewItem will be visible in the content area. You can add a caption bar title to the header area. The button in the upper right corner will either maximize or minimize the C1TileView control.

TileViewItem States

Each **C1TileViewItem** includes three different states – minimized, maximized, and the default state (which is neither minimized nor maximized). For example, in the following images all three of the **C1TileViewItem**s in the **C1TileView** control appear in the default state:

In the following image, the red **C1TileViewItem** is maximized and the other two items are minimized:



When one item is maximized, the other items are minimized and appear as specified by the **MinimizedItem** property. The default state uses the **Columns** and **Rows** properties to determine the layout. The minimized/maximized states use the **MinimizedItemPosition** property to determine layout.

Columns and Rows

The Columns and Rows properties get or set the number of columns and rows the C1TileViewItems are laid in respectively. If the value is zero, the minimum number that doesn't require scrolling is used. If both **Columns** and **Rows** are zero, the items are laid in a square.

The default state uses the **Columns** and **Rows** properties to determine the layout. The minimized/maximized states use the **MinimizedItemPosition** property to determine layout. For more information about states, see TileViewItem States (page 7).

Minimized Item Position

The **MinimizedItemsPosition** property allows you to determine where minimized items will appear within the **C1TileView** control. Options include **Left**, **Right**, **Top**, and **Bottom**. By default, minimized items appear at the right of the panel.

Note that the default **C1TileView** state uses the **Columns** and **Rows** properties to determine the layout. The minimized/maximized states use the **MinimizedItemPosition** property to determine layout. For more information about states, see TileViewItem States (page 7).

Drag-and-Drop Interaction

You can easily determine whether drag-and-drop operations are allowed within the **C1TileView** control by setting the **CanUserReorder** property. By default, this property is set to **True** and users can reorder items at run time. If this property is set to **False**, instead, users will no longer be able to reorder items at run time. See Disabling Drag-and-Drop Functionality (page 14) for an example.

Basic Properties

ComponentOne TileView for Silverlight includes several properties that allow you to set the functionality of the control. Some of the more important properties are listed below. Note that you can see TileView for Silverlight Appearance Properties (page 10) for more information about properties that control appearance.

The following properties let you customize the C1TileView control:

Property	Description
AnimationDuration	Gets or sets the time that item reordering takes.
AnimationEasingFunction	Gets or sets how to soften the reordering movement.
CanUserReorder	Gets or sets whether the user is allowed to drag and drop and reorder C1TileViewItems from this control.
Columns	Gets or sets the number of columns the C1TileViewItems are laid in. If the value is zero, the minimum number that doesn't require scrolling is used. If both **Columns** and **Rows** are zero, the items are laid in a square.
ItemTemplateHeader	Gets or sets the **DataTemplate** used as title for the items.
ItemTemplateMaximized	Gets or sets the **DataTemplate** used for items in the **Maximized()** state.
ItemTemplateMinimized	Gets or sets the **DataTemplate** used for items in the **Minimized()** state.
MaximizedIndex	Gets or sets the index in the **Items** collection of the selected item.
MaximizedItem	Gets or sets the member of the **Items** collection currently highlighted.
MinimizedItemsPosition	Gets or sets where to place the strip with the minimized items. The **ScrollBar** is at the right or bottom of the strip.
Rows	Gets or sets the number of rows the **C1TileViewItem**s are laid in. If the value is zero, the minimum number that doesn't require scrolling is used. If both **Columns** and **Rows** are zero, the items are laid in a square.
ScrollBarStyle	Gets or sets the style used for the inner scrollbar.
ScrollBarVisibility	Gets or sets whether the scrollbar should be visible.
UpdateSourceCollection	Gets or sets whether changes in the order of the items are written to **Items** or **ItemsSource**.

TileView for Silverlight Layout and Appearance

The following topics detail how to customize the C1TileView control's layout and appearance. You can use built-in layout options to lay your controls out in panels such as Grids or Canvases. Themes allow you to customize the appearance of the grid and take advantage of WPF's XAML-based styling. You can also use templates to format and lay out the control and to customize the control's actions.

TileView for Silverlight Appearance Properties

ComponentOne TileView for Silverlight includes several properties that allow you to customize the appearance of the control. You can change the color, border, and height of the control. The following topics describe some of these appearance properties.

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.
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.
Foreground	Gets or sets a brush that describes the foreground color. This is a dependency property.
HeaderForeground	Gets or sets the **Brush** used as foreground of the header of the contained C1TileViewItems.
ItemBackground	Gets or sets the **Brush** used as background of the contained C1TileViewItems.
ItemForeground	Gets or sets the **Brush** used as foreground of the contained C1TileViewItems.
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.

Alignment Properties

The following properties let you customize the control's alignment:

Property	Description
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.
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.

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 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.

TileView 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 TileView 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

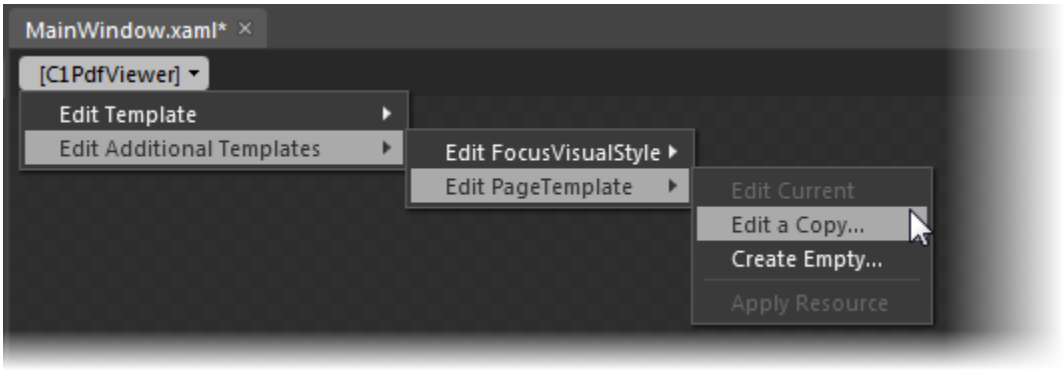
You can access templates in Microsoft Expression Blend by selecting the C1TileView control and, in the menu, selecting **Edit Template**. Select **Edit a Copy** to create an editable copy of the current template or **Create Empty**, to create a new blank template.

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 C1TileView control includes a few additional templates. These additional templates can also be accessed in Microsoft Expression Blend – in Blend select the C1TileView control and, in the menu, select **Edit Additional Templates**. Choose a template, and select **Create Empty**:



C1TileView Styles and Templates

ComponentOne TileView for Silverlight's C1TileView 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
FocusVisualStyle	Gets or sets a property that enables customization of appearance, effects, or other style characteristics that will apply to this element when it captures keyboard focus. This is a dependency property.
FontStyle	Gets or sets the font style. This is a dependency property.
ScrollBarStyle	Determines the style of the scroll bar.
Style	Gets or sets the style used by this element when it is rendered. This is a dependency property.

Some of the included templates are described in the table below:

Style	Description
ItemTemplateHeader	
ItemTemplateMaximized	
ItemTemplateMinimized	

C1TileView Visual States

In Microsoft Expression Blend, you can add custom states and state groups to define a different appearance for each state of your user control – for example, the visual state of the control could change on mouse over. You can view and edit visual states by creating a new template. Once you've done so the available visual states for that part will be visible in the **States** window.

TileView for Silverlight Task-Based Help

The following task-based help topics assume that you are familiar with Visual Studio and Expression Blend and know how to use the C1TileView control in general. If you are unfamiliar with the **ComponentOne TileView for Silverlight** product, please see the TileView for Silverlight Quick Start (page 3) first.

Each topic in this section provides a solution for specific tasks using the **ComponentOne TileView for Silverlight** product. Most task-based help topics also assume that you have created a new Silverlight project and added a C1TileView control to the project.

Adding C1TileView to the Application

In this topic you'll add a C1TileView control to your application. Complete the following steps:

1. From the Visual Studio **File** menu select **New** and choose **Project**.
2. In the **New Project** dialog box choose a language in the left-side menu, choose **.NET Framework 4** in the **Framework** drop-down list, and enter a name for the project.
3. In the Solution Explorer, right-click the project name and choose **Add Reference**. In the **Add Reference** dialog box, locate and select the following assemblies and click **OK** to add references to your project:
   - C1.Silverlight
   - C1.Silverlight.TileView
4. Open the XAML view of the MainPage.xaml file and add the XAML namespace to the UserControl tag with the following markup:
   `xmlns:c1="http://schemas.componentone.com/winfx/2006/xaml"`.

   The namespaces will now appear similar to the following:
   ```
   <UserControl x:Class="QuickStart.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"
   xmlns:c1="http://schemas.componentone.com/winfx/2006/xaml"
   mc:Ignorable="d" d:DesignHeight="300" d:DesignWidth="400">
   ```

   This is a unified namespace that will enable you to work with most ComponentOne WPF or Silverlight controls without adding multiple namespaces.
5. Add the `<c1:C1TileView x:Name="C1TileView1" />` tag within the Grid tags on the page to add the C1TileView control to the application.

   The XAML will appear similar to the following:
   ```
   <Grid x:Name="LayoutRoot" Background="White">
       <c1:C1TileView x:Name="C1TileView1" />
   </Grid>
   ```

   This will add a C1TileView control named "C1TileView1" to the application.

What You've Accomplished

You've successfully set up your application's user interface, but if you run your application now you'll see that the C1TileView control currently contains no content. See the Adding Items to C1TileView (page 14) topic for more information.

Note: If the **C1TileView** control was installed to the Visual Studio Toolbox, simply dragging the control onto a page will automatically perform all the steps above.

Adding Items to C1TileView

In this topic you'll add C1TileViewItems to the **C1TileView** control. Note that this topic assumes you have already added an empty **C1TileView** control to your application.

Edit the `<c1:C1TileView x:Name="C1TileView1" />` tag to add several C1TileViewItems. The XAML will appear similar to the following:

```
<c1:C1TileView Name="C1TileView1">
    <c1:C1TileViewItem Background="Red" Header="Red"></c1:C1TileViewItem>
    <c1:C1TileViewItem Background="Orange"
Header="Orange"></c1:C1TileViewItem>
    <c1:C1TileViewItem Background="Yellow"
Header="Yellow"></c1:C1TileViewItem>
    <c1:C1TileViewItem Background="Green"
Header="Green"></c1:C1TileViewItem>
    <c1:C1TileViewItem Background="Blue"
Header="Blue"></c1:C1TileViewItem>
    <c1:C1TileViewItem Background="Purple"
Header="Purple"></c1:C1TileViewItem>
</c1:C1TileView>
```

You've successfully added six C1TileViewItems to the **C1TileView** control.

Disabling Drag-and-Drop Functionality

By default drag-and-drop functionality is enabled allowing users to re-order C1TileViewItem elements at run time. If you choose, however, you can disable drag-and-drop functionality by setting the CanUserReorder property to **False**.

At Design Time

To disable drag-and-drop functionality in the C1TileView control in the Properties window at design time, complete the following steps:

1. Click the C1TileView control once to select it.
2. Navigate to the Properties window, and locate the **CanUserReorder** property.
3. Click the drop-down arrow next to the **CanUserReorder** property and select **False**.

   This will disable drag-and-drop functionality.

In XAML

To disable drag-and-drop functionality in the C1TileView control in XAML add `CanUserReorder="False"` to the `<c1:TileView>` tag so that it appears similar to the following:

```
<c1:C1TileView Name="C1TileView1" CanUserReorder="False">
```

In Code

Right-click the window and select **View Code** to open the Code Editor. Add code to the main class, so it appears similar to the following:

- Visual Basic

```
Public Sub New()
InitializeComponent()
    Me.C1TileView1.CanUserReorder = False
End Sub
```

- C#

```
MainPage(){
InitializeComponent();
    this.c1TileView1.CanUserReorder = false;
}
```

Run your project and observe:

You will not be able to perform drag-and-drop operations at run time.

Customizing the Header's Appearance

C1TileView includes several properties that enable you to change the appearance of the **C1TileViewItem**'s **Header**. These properties include: **Header**, **HeaderBackground**, **HeaderFontFamily**, **HeaderFontSize**, **HeaderFontStretch**, **HeaderFontStyle**, **HeaderFontWeight**, **HeaderForeground**, **HeaderPadding**, and **HeaderTemplate**.

For example, the following marlup sets several of these properties:

```
<c1:C1TileViewItem Header="News" HeaderPadding="10 5 5 5"
HeaderForeground="#FF507494" HeaderFontFamily="Trebuchet MS"
HeaderFontSize="16">
   <c1:C1TileViewItem.HeaderBackground>
      <LinearGradientBrush EndPoint="0.5,1" StartPoint="0.5,0">
        <GradientStop Color="#FFE9ECF0" Offset="0" />
            <GradientStop Color="#FFDDE1E7" Offset="0.2" />
            <GradientStop Color="#FFCCD3DC" Offset="0.2" />
            <GradientStop Color="#FFFAFAFB" Offset="0.647" />
      </LinearGradientBrush>
    </c1:C1TileViewItem.HeaderBackground>
</c1:C1TileViewItem>
```

Creating Minimized and Maximized Item Templates

You can customize how items in the C1TileView control are displayed when minimized and maximized. For example, you may wish to display an icon in the content area of a minimized item indicating the type of content that item contains. You can use the ContentMinimized and ContentMaximized properties to set a display template. If these properties are not set, the **Content** is used.

For example, the following markup adds ContentMinimized and ContentMaximized templates:

```
<c1:C1TileView Name="C1TileView1">
    <c1:C1TileViewItem Background="Red" Header="Red">
        <c1:C1TileViewItem.ContentMinimized >
            <Label Content="Red Minimized" Height="28" Name="Label1"
Foreground="White"/>
        </c1:C1TileViewItem.ContentMinimized>
        <c1:C1TileViewItem.ContentMaximized >
            <Label Content="Red Maximized" Height="28" Name="Label2"
Foreground="White"/>
        </c1:C1TileViewItem.ContentMaximized>
    </c1:C1TileViewItem>
    <c1:C1TileViewItem Background="Orange" Header="Orange">
        <c1:C1TileViewItem.ContentMinimized >
            <Label Content="Orange Minimized" Height="28" Name="Label3"
Foreground="White"/>
        </c1:C1TileViewItem.ContentMinimized>
        <c1:C1TileViewItem.ContentMaximized >
```

```
            <Label Content="Orange Maximized" Height="28" Name="Label4"
Foreground="White"/>
        </c1:C1TileViewItem.ContentMaximized>
    </c1:C1TileViewItem>
</c1:C1TileView>
```

What You've Accomplished

You've added templates for the minimized and maximized **C1TileView** states. Run your application and maximize one of the items, observe that the content of both the minimized and maximized items has changed. Maximize the minimized items, the content of each item changes again.