ComponentOne

Tiles for UWP

ComponentOne, a division of GrapeCity
201 South Highland Avenue, Third Floor
Pittsburgh, PA 15206 USA

Website: http://www.componentone.com
Sales: sales@componentone.com
Telephone: 1.800.858.2739 or 1.412.681.4343 (Pittsburgh, PA USA Office)

Trademarks

Warranty

ComponentOne warrants that the media on which the software is delivered is free from defects in material and workmanship, assuming normal use, for a period of 90 days from the date of purchase. If a defect occurs during this time, you may return the defective media to ComponentOne, along with a dated proof of purchase, and ComponentOne will replace it at no charge. After 90 days, you can obtain a replacement for the defective media by sending it and a check for $2 5 (to cover postage and handling) to ComponentOne.

Except for the express warranty of the original media on which the software is delivered is set forth here, ComponentOne makes no other warranties, express or implied. Every attempt has been made to ensure that the information contained in this manual is correct as of the time it was written. ComponentOne is not responsible for any errors or omissions. ComponentOne's liability is limited to the amount you paid for the product. ComponentOne is not liable for any special, consequential, or other damages for any reason.

Copying and Distribution

Table of Contents

Tiles for UWP

Create tiled displays and navigation hubs with **Tiles for UWP**. Tiles make it easy to replicate the Windows 10 start screen experience in your own app. Get several different tile controls that support sliding and flipping animations with live updates. Combine tiles with different containers to achieve endless layout possibilities.

Key Features

Tiles for UWP includes the following key features:

- **Create Flipping and Sliding Tiles**

  With the C1FlipTile and C1SlideTile classes, you can create tiles that display alternating content with a sliding or flipping animation. Simply design your template and provide content to the control. The updates and animations are handled automatically.

- **Familiar Windows 10 Live Tile Behavior**

  Tiles have been specially designed for Universal Windows apps. Each tile control exhibits the same interactive behavior as the start screen live tiles on Windows 8 or 10 and Windows Phone. This means your app will present familiar behavior to the user and you don't have to lift a finger (except to select a tile).

- **Live Updates**

  Tiles can flip, slide and show updated, "live" content. You can easily control the update interval using the static C1TileService class and the UpdateInterval property. See Updating the Tile Content for details.

- **Host in Any Container**

  Tiles can be hosted in any **ItemsControl** containers such as **C1TileListBox**, **C1WrapPanel**, or the standard **GridView** and **ListBox** controls. Thus you can use **C1Tiles** in data bound scenarios. Each container presents a different way to arrange multiple **C1Tiles** together giving you endless combinations and possibilities.

- **Support for Different Sizes**

  Not all tiles must be created equally. Combine tiles of different types and sizes together to create displays uniquely catered to your application.

Tiles for UWP Quick Start

The following quick start guide is intended to get you up and running with **Tiles for UWP**. In this quick start you'll start in Visual Studio and create a new project, add **Tiles for UWP** controls to your application, and customize the appearance and behavior of the controls.

You will create an application that includes the C1Tile, C1SlideTile, and C1FlipTile controls. For more information about each of those controls, see Working with C1Tile, Working with C1SlideTile, and Working with C1FlipTile.

Step 1 of 3: Setting up the Tiles Application

In this step you'll begin in Visual Studio to create an application using **Tiles for UWP**. In this step you'll create a Universal Windows application and add a C1TileListBox to contain the **Tiles for UWP** controls. To begin, complete the following steps:

1. In Visual Studio, select **File | New | Project**.
2. Select Templates | Visual C# | Windows | Universal. From the templates list, select **Blank App (Universal Windows)**. Enter a **Name** for your project and click **OK** to create your project.

   The MainPage.xaml page will open.

3. Add the following markup within the <Page> tag, and just before the <Grid> tag:

Markup

```
<Page.Resources>
             <Style x:Key="listBoxItemStyle" TargetType="Xaml:C1ListBoxItem">
             <Setter Property="Tile:C1TileService.PointerDownAnimation"
            Value="True"/>
             </Style>
             <Style TargetType="Tile:C1Tile" x:Key="baseTileStyle">
             <Setter Property="Background" Value="#FFC410" />
             <Setter Property="Foreground" Value="White"/>
             <Setter Property="FontSize" Value="80"/>
             <Setter Property="HeaderForeground" Value="White"/>
             <Setter Property="HeaderFontSize" Value="12"/>
             <Setter Property="BorderThickness" Value="0" />
             <Setter Property="Width" Value="280" />
             <Setter Property="Height" Value="200" />
             </Style>
             <Style TargetType="Tile:C1Tile" BasedOn="{StaticResource
baseTileStyle}">
             </Style>
             <Style TargetType="Tile:C1SlideTile" BasedOn="{StaticResource
baseTileStyle}">
             </Style>
             <Style TargetType="Tile:C1FlipTile" BasedOn="{StaticResource
baseTileStyle}">
             </Style>
             </Page.Resources>
```

This markup adds page resources to style the appearance of the application.

4. Place the cursor between the <Grid> and </Grid> tags, navigate to the Toolbox, and double-click the **C1TileListBox** control to add it to the page. This will also add a reference to the C1.Xaml assembly.
5. Update the **C1TileListBox** markup so it looks like the following:

Markup

```
<Xaml:C1TileListBox ItemStyle="{StaticResource listBoxItemStyle}">
<Xaml:C1TileListBox.Items>
                </Xaml:C1TileListBox.Items>
            </Xaml:C1TileListBox>
```

In this step you created a Universal Windows application. In the next step you'll add **Tiles for UWP** controls to the application.

Step 2 of 3: Adding Tiles to the Application

In the previous step you created a Universal Windows application; in this step you'll add **Tiles for UWP** controls. Complete the following steps:

1. Place your cursor between the <Xaml:C1TileListBox.Items></Xaml:C1TileListBox.Items> tags, navigate to the Toolbox, and double-click the **C1Tile** control to add it to the page. This will also add a reference to the C1.Xaml.Tile assembly.
2. Update the **C1Tile** markup so it looks like the following:

Markup

```
<Tile:C1Tile Content="1" Header="C1Tile" HeaderPadding="12" Padding="0"
        HeaderBackground="#22000000" HorizontalHeaderAlignment="Stretch" />
```

This markup adds header and content text, adds padding, changes the color of the header background, and the alignment of the header.

3. Add the following markup just below the **C1Tile** markup:

Markup

```
<Tile:C1SlideTile Content="2" HeaderPadding="12" HorizontalContentAlignment="Stretch"
        VerticalContentAlignment="Stretch" Padding="0"
            Header="C1SlideTile">
        <Tile:C1Tile.ContentTemplate>
        <DataTemplate>
                <Border Background="#FFBC1C48" >
                <TextBlock Text="{Binding}" VerticalAlignment="Center" HorizontalAlignment="Center" />
            </Border>
        </DataTemplate>
    </Tile:C1Tile.ContentTemplate>
    <Tile:C1Tile.BackContentTemplate>
        <DataTemplate>
            <Border Background="#FF028541" >
                <StackPanel VerticalAlignment="Center" HorizontalAlignment="Center">
                        <TextBlock Text="{Binding}" Foreground="White" HorizontalAlignment="Center"/>
                        <TextBlock Text="Back Content Template" Margin="0 -10 0 0" FontSize="12"
                                Foreground="White" HorizontalAlignment="Center"/>
                </StackPanel>
            </Border>
        </DataTemplate>
     </Tile:C1Tile.BackContentTemplate>
</Tile:C1SlideTile>
```

This markup adds a **C1SlideTile** control with a **ContentTemplate** and a **BackContentTemplate**. The **ContentTemplate** determines the initial content and appearance of the control. The **BackContentTemplate** determines the content and appearance of the control once it transitions (in the case of the C1SlideTile control, it slides from one template to the other).

4. Add the following markup just below the **C1SlideTile** markup:

Markup

```
<Tile:C1SlideTile Content="3" BackContent="Back Content 3" FontSize="36" Header="C1SlideTile"
             HeaderPadding="12" Padding="0"/>
                        <Tile:C1SlideTile Content="4" Header="C1SlideTile, SlideDirection = Right"
                                HeaderPadding="12" Padding="0"
```

```
                                SlideDirection="Right" HorizontalHeaderAlignment="Right"/>
                        <Tile:C1SlideTile Content="5" Header="C1SlideTile" />
                        <Tile:C1FlipTile Content="6" Header="C1FlipTile" />
                        <Tile:C1FlipTile Content="7" Header="C1FlipTile" HeaderPadding="12" Padding="0"
                                HeaderBackground="#22000000">
                                <Tile:C1Tile.ContentTemplate>
                        <DataTemplate>
                          <Border Background="#FF8C0095" >
                              <TextBlock Text="{Binding}" VerticalAlignment="Center"
HorizontalAlignment="Center" />
                                </Border>
                        </DataTemplate>
                </Tile:C1Tile.ContentTemplate>
                <Tile:C1Tile.BackContentTemplate>
                        <DataTemplate>
                                <Border Background="#FFCD4900" >
                                        <StackPanel VerticalAlignment="Center"
HorizontalAlignment="Center">
                                            <TextBlock Text="{Binding}"  Foreground="White"
HorizontalAlignment="Center"/>
                                            <TextBlock Text="Back Content Template" Margin="0 -10 0 0"
                                                    FontSize="12"
                                                    Foreground="White" HorizontalAlignment="Center"/>
                                        </StackPanel>
                                </Border>
                        </DataTemplate>
            </Tile:C1Tile.BackContentTemplate>
                </Tile:C1FlipTile>
```

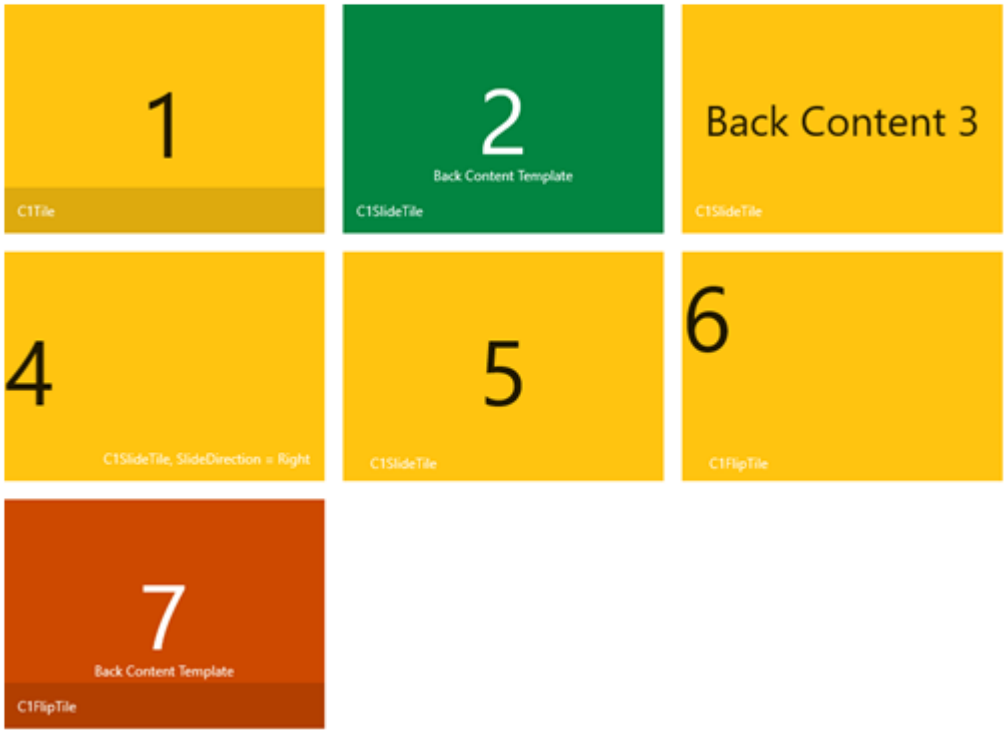
5. This markup adds **C1SlideTile** and **C1FlipTile** controls. Note that one of the **C1FlipTile** controls also includes a **ContentTemplate** and a **BackContentTemplate**.

In this step you added **Tiles for UWP** controls to your application. In the next step you'll run the application so see how the application appears at run time.

Step 3 of 3: Running the Tiles Application

In the previous step you created a Universal Windows application and added and customized the **Tiles for UWP** controls. In this step you'll run your application.

1. Select **Debug | Start Debugging** to run your application. It will appear similar to the following:



> The green **C1SlideTile** in the image above is displayed using the **ContentTemplate**. The orange **C1FlipTile** is displayed in the **BackContentTemplate**.

2. Observe the behavior of each of the controls. The **C1SlideTile** appears to slide between content selections, the **C1FlipTile** appears to flip.
3. Observe that some **C1SlideTile** controls appear to move in different directions, you can set this behavior using the SlideDirection property.

Congratulations!

You have completed the **Tiles for UWP** quick start. In the documentation that follows, you'll get to know more about how to use the **Tiles for UWP** controls.

Working with Tiles for UWP

The following topics give an overview of working with **Tiles for UWP**.

- C1Tile Control
- C1FlipTile Control
- C1SlideTile Control
- ContentTemplates and BackContentTemplates

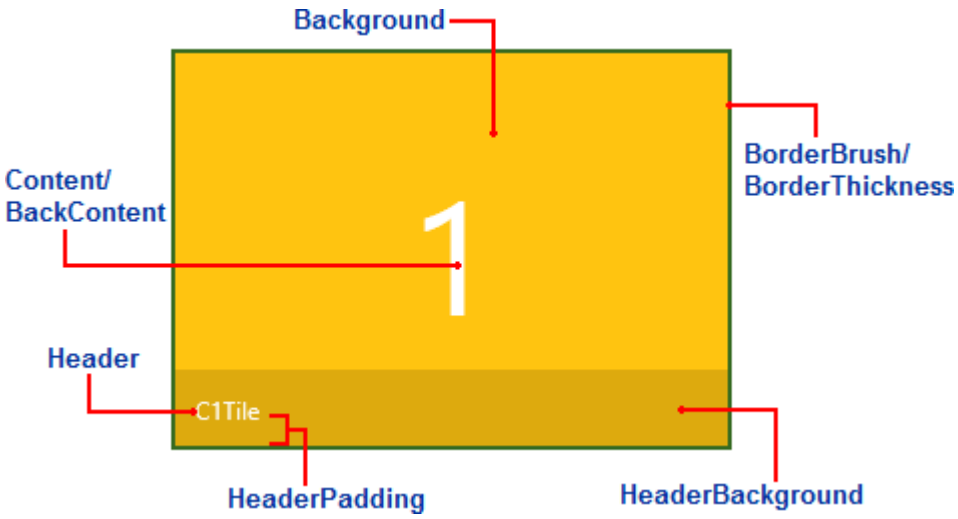
C1Tile Control

The C1Tile control is an animated headered content control that mimics the Windows 10 live tiles behavior. For example, the markup below creates a basic **C1Tile** control with several properties set:

Markup

```
<Tile:C1Tile Content="1" BackContent="1" Header="C1Tile" HeaderBackground="#22000000"
HeaderPadding="12" Padding="0" BorderBrush="#FF356A21" BorderThickness="2"
Background="#FFC410" Foreground="White" FontSize="80" HeaderForeground="White"
HeaderFontSize="12" Width="280" Height="200"/>
```

The following image illustrates the result of the above markup, noting some of the set properties:



The following properties are set in the image/markup above:

- **Content**: This property sets the initial content of the **C1Tile** control. In the above image, **Content** is set to "1".
- BackContent: This sets the alternative content of the **C1Tile** control. In the above image, BackContent is also set to "1" (the same as Content). But if the BackContent property is set to another value, that value would appear when the tile's content changes.
- **Header**: This property sets the content of the **C1Tile** control's header. By default the header appears at the bottom of the tile, in the above image, **Header** is set to "C1Tile".
- **HeaderBackground**: This property sets the color and transparency of the header's background. In the above

example, the HeaderBackground property is set to "22000000". The first two values indicate the transparency of the color; the last six values indicate that the color is black.

- **HeaderPadding**: The **HeaderPadding** property sets the padding around the **Header** value. In the above example, **HeaderPadding** is set to "12". The greater the number, the futher the value indicated by the **Header** property will appear from the edges of the tile.
- **Padding**: The **Padding** property sets the padding around the **Content** and **Header** within the tile. In the above example, **Padding** is set to "0". This indicates that the header area appears flush against the bottom border of the tile. The greater the number, the futher the values indicated by the **Content** and **Header** properties will appear from the edges of the tile.
- **BorderBrush**: The **BorderBrush** property indicates the color of the border around the **C1Tile** control. In the above example, this is set to "#FF356A21", an opaque green color.
- **BorderThickness**: The **BorderThickness** indicates the thickness of the border surrounding the **C1Tile**. In the above example, this property is set to "2".

For information about templates, see ContentTemplates and BackContentTemplates.

C1FlipTile Control

The C1FlipTile control is based on the C1Tile control. However, the **C1FlipTile** control includes a different sort of transition animation effect between alternating content. In the **C1FlipTile** control, the animation makes the tile appear to be flipping between content.

Here's an example of markup for the **C1FlipTile** control:

Markup

```
<Tile:C1FlipTile Content="7" Header="C1FlipTile" HeaderBackground="#22000000"
HeaderPadding="12" Padding="0" BorderThickness="2" Background="#8B008B"
Foreground="White" FontSize="80" HeaderForeground="White" HeaderFontSize="12"
VerticalContentAlignment="Center" HorizontalContentAlignment="Center" Width="280"
Height="200"/>
```

The markup above creates the following **C1FlipTile** control:

C1SlideTile Control

Based on the C1Tile control, the C1SlideTile control includes a sliding animation between templates; however, the C1SlideTile.SlideDirection property determines the direction the content slides. You can set the SlideDirection property to one of the following values (the default value is **All**):

Value	Direction
Up	Bottom to top
Left	Right to left
Down	Top to bottom
Right	Left to right
All (default)	All

For example, here's an example of a **C1SlideTile** control with its SlideDirection property set to **Right**:

Markup

```
<Tile:C1SlideTile Content="4" Header="C1SlideTile, SlideDirection = Right"
Padding="0" SlideDirection="Right" HorizontalHeaderAlignment="Right"
Background="#FFC410" Foreground="White" FontSize="80" HeaderForeground="White"
HeaderFontSize="12" Width="280" Height="200"/>
```

The markup above creates the following **C1SlideTile** control:

ContentTemplates and BackContentTemplates

You can use **ContentTemplate**s and **BackContentTemplates** to customize the appearance of the **Tiles for UWP** controls. Place the content you want to appear initially in the tile in the **ContentTemplate** and place content that you want to appear when the tile slides or flips to alternative content in the **BackContentTemplate**.

For example, the following markup adds a **ContentTemplate** and a **BackContentTemplate** to a **C1SlideTile** control:

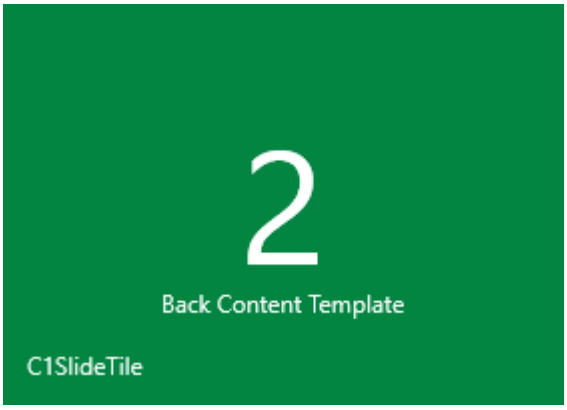
Markup

```
<Tile:C1SlideTile Content="2" HeaderPadding="12" HorizontalContentAlignment="Stretch"
VerticalContentAlignment="Stretch" Padding="0" Header="C1SlideTile"
Foreground="White" FontSize="80" HeaderForeground="White" HeaderFontSize="12"
```

```
Width="280" Height="200">
            <Tile:C1Tile.ContentTemplate>
                <DataTemplate>
                    <Border Background="#FFBC1C48" >
                        <TextBlock Text="{Binding}" VerticalAlignment="Center"
HorizontalAlignment="Center" />
                    </Border>
                </DataTemplate>
            </Tile:C1Tile.ContentTemplate>
            <Tile:C1Tile.BackContentTemplate>
                <DataTemplate>
                    <Border Background="#FF028541" >
                        <StackPanel VerticalAlignment="Center"
HorizontalAlignment="Center">
                            <TextBlock Text="{Binding}" Foreground="White"
HorizontalAlignment="Center"/>
                            <TextBlock Text="Back Content Template" Margin="0 -10 0
0" FontSize="12" Foreground="White" HorizontalAlignment="Center"/>
                        </StackPanel>
                    </Border>
                </DataTemplate>
            </Tile:C1Tile.BackContentTemplate>
        </Tile:C1SlideTile>
```

At run time, the control will initially appear like the following with the ContentTemplate displayed:

After transitioning to the alternative content template, the control will appear with the following **BackContentTemplate** displayed:



Tiles for UWP Task-Based Help

The task-based help assumes that you are familiar with programming in Visual Studio, and know how to use create and use Universal Windows applications in general. If you are a novice to the **Tiles for UWP** product, please see the Tiles for UWP Quick Start first.

Each topic provides a solution for specific tasks using the **Tiles for UWP** product. By following the steps outlined in the help, you will be able to create projects demonstrating a variety of **Tiles for UWP** features.

Note that the following topics assume you have created a new Universal Windows project.

Adding Content to the Tile Header

The **Tiles for UWP** controls do not support direct XAML content in their **Header** properties. You should either bind the **Header** to some CLR value, or leave it empty and include your desired **Header** content (such as an image) into the Content template.

For example:

Markup

```
<c1tile:C1FlipTile Width="280" Height="200" Header="{Binding Name}" />
```

Updating the Tile Content

The **C1SlideTile** and **C1FlipTile** controls will automatically alternate between content at regular intervals. You can adjust this interval using the static C1TileService class and the UpdateInterval property. For example:

Visual Basic

```
C1TileService.UpdateInterval = TimeSpan.FromSeconds(20)
```

C#

```
C1TileService.UpdateInterval = TimeSpan.FromSeconds(20);
```

The higher the update interval value, the less frequent updates will become.

You can also programmatically update a tile by calling the UpdateTile method on each specific tile. For example, to update the tile when it is clicked use the following code:

Visual Basic

```
Private Sub C1Tile_Click(sender As Object, e As System.EventArgs)
      Dim tile As C1Tile = TryCast(sender, C1Tile)
      If tile IsNot Nothing Then
            tile.UpdateTile()
      End If
End Sub
```

C#

```
private void C1Tile_Click(object sender, System.EventArgs e)
{
    C1Tile tile = sender as C1Tile;
```

```
    if(tile != null)
        tile.UpdateTile();
}
```

Binding to a Collection of Items

You can show more than just two alternating items in a **C1SlideTile** or **C1FlipTile** control. By using the ContentSource property you can provide a collection of any number of items that the control will flip through. Define the **ContentTemplate** (and optionally the **AlternateContentTemplate**) to define the appearance of the bound content.

Markup

```
<c1:C1SlideTile Header="Photos" ContentSource="{Binding Items}">
    <c1:C1SlideTile.ContentTemplate>
        <DataTemplate>
            <Grid>
                <TextBlock Text="{Binding Author}" Foreground="White"
VerticalAlignment="Top" Margin="4,2,0,2"/>
                <Image Source="{Binding Thumbnail}" Stretch="UniformToFill"
Margin="24, 24, 1, 1"/>
            </Grid>
        </DataTemplate>
    </c1:C1SlideTile.ContentTemplate>
</c1:C1SlideTile>
```

Using Tiles in a Bound Control

You can use **C1Tiles** in any **ItemsControl** such as a **C1TileListBox** or **GridView** control. Here's a markup example using **C1FlipTile**s in a bound **ListBox** control:

Markup

```
<ListBox x:Name="listBox" ItemsSource="{Binding}">
    <ListBox.ItemsPanel>
                    <ItemsPanelTemplate>
                        <Xaml:C1WrapPanel
    Background="YellowGreen"/>
                    </ItemsPanelTemplate>
                </ListBox.ItemsPanel>
                <ListBox.ItemContainerStyle>
                    <Style TargetType="ListBoxItem">
                        <Setter
    Property="Tile:C1TileService.PointerDownAnimation" Value="True"/>
                    </Style>
                </ListBox.ItemContainerStyle>
                <ListBox.ItemTemplate>
                    <DataTemplate>
                        <Tile:C1FlipTile Height="200"
    Width="200" Header="{Binding Title}" Content="{Binding}"
        Background="DarkGreen" HeaderBackground="#88000000" HeaderFontSize="18"
Command="{Binding
```

```
    TileCommand, ElementName=pageRoot}" HeaderPadding="2" Padding="0"
HeaderForeground="White"
    CommandParameter="{Binding Content,
    RelativeSource={RelativeSource Self}}">
                    <Tile:C1Tile.ContentTemplate>
                        <DataTemplate>
                          <Grid>
                              <TextBlock Text="{Binding Author}" Foreground="White"
VerticalAlignment="Top" Margin="4,2,0,2"/>
                              <Image Source="{Binding Thumbnail}"
Stretch="UniformToFill" Margin="1, 24, 24, 1"/>
                          </Grid>
                        </DataTemplate>
                    </Tile:C1Tile.ContentTemplate>
                    <Tile:C1Tile.BackContentTemplate>
                        <DataTemplate>
                          <Grid>
                              <Image Source="{Binding Thumbnail}"
Stretch="UniformToFill"/>
                          </Grid>
                        </DataTemplate>
                    </Tile:C1Tile.BackContentTemplate>
                </Tile:C1FlipTile>
            </DataTemplate>
        </ListBox.ItemTemplate>
    </ListBox>
```

If you use **C1Tile** controls in the **ListBox** with **VirtualizingStackPanel** as **ItemsPanel**, set **VirtualizingStackPanel.VirtualizationMode** to **Standard** to avoid tile animations while scrolling the **ListBox**.