ComponentOne

TileControl for WinForms

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)

Table of Contents

TileControl for WinForms Overview

The Windows 8-inspired TileControl for WinForms makes it easy to replicate the Windows 8 Modern UI experience in your desktop app. Get several different tile controls that support panning and tapping gestures. Combine tiles with different containers to achieve endless layout possibilities.	Getting Started • TileControl for WinForms Quick Start • TileControl for WinForms Key Features • TileControl for WinForms Task-Based Help

Help with WinForms Edition

Getting Started

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

TileControl for WinForms Key Features

The following are some of the main features of C1TileControl that you may find useful:

Custom Tile Layout

The TileControl uses either automatic or manual layout of tiles in the groups. Tiles may be any size; they are not limited to large and small sizes only. The interior layout of the tiles (or rather tile templates) is very flexible. You can use docked and stacked panels, nested panels, text elements, and images. Additionally, you can save the layout to an XML file and load it from an XML file at any time.

Two Display Orientations

Tile groups can be stacked vertically or horizontally.

Images

The background image of the control can be scrolled with tiles, as on Windows 8 Start Screen. In addition to user images, there is a set of standard symbols of various sizes that can be displayed on tiles. Also, it's easy to display the "badge number" or "5-star" rating image as a part of the template. These elements can be bound to an integer value specified in a property of the tile.

Templates

There is no need to design each tile separately. Instead, you can create one or several tile templates, then associate these templates with tiles. Tiles can provide data for templates, such as strings, colors, and images. Its possible to associate one template with several tiles, and to switch templates for a single tile; for example, to alternate text and image views by a timer. For an example of how to do this see the

Touch Support

The TileControl supports panning, tapping, and checking tiles (with a swipe gesture) using the touch input hardware on a machine with Windows 7 or Windows 8. When users press and hold a tile with their fingertip, the tile shows its tooltip.

Navigation

It is easy to navigate between tiles using the keyboard.

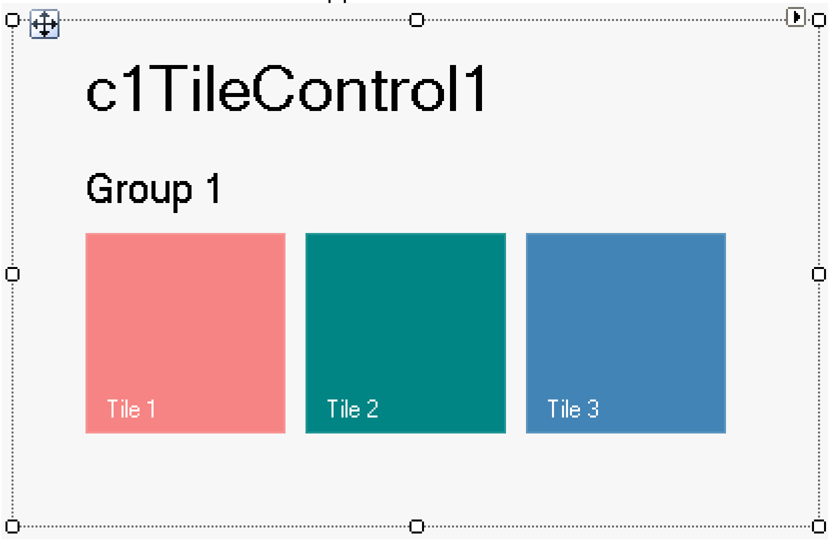
TileControl for WinForms Quick Start

The goal of this quick start guide is to get you acquainted with **Tile Control for WinForms**. In the first step of this Quick Start guide, you will add a C1TileControl to your WinForms project. This quick start guide will also explain how to add the C1Tile control to your application, add content that will be displayed in the C1Tile control, and observe some of the run-time interactions possible with **TileControl for WinForms**.

Step 1 of 3: Creating the C1Tile Application

In this step, you will create a .NET project using **Tile Control for WinForms**. When you add a C1Tile control to your application, you'll have an interface that you can display content in. To set up your project and add a C1Tile control to your application, complete the following steps:

1. Begin by creating a new Windows Forms Application. 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**. Select the **Browse** tab to locate **C1.Win.C1TileControl.2.dll**. In the **Add Reference** dialog box, select the C1.Win.C1TileControl.2.dll and click **OK** to add references to your project.
3. While in Design view, navigate to the Visual Studio Toolbox and double-click the **C1TileControl** to add it to your form. If its not there right-click in the toolbox area and select **Add Tab**. Name the tab, for example, **C1TileControl**. Right-click under the C1TileControl and select **Choose Items**. The **Choose Toolbox Items** appears. Browse to the **C1.Win.C1Tile** assembly and click **OK**.
4. The **C1TileControl** control appears.



Step 2 of 3: Creating the Template with Elements

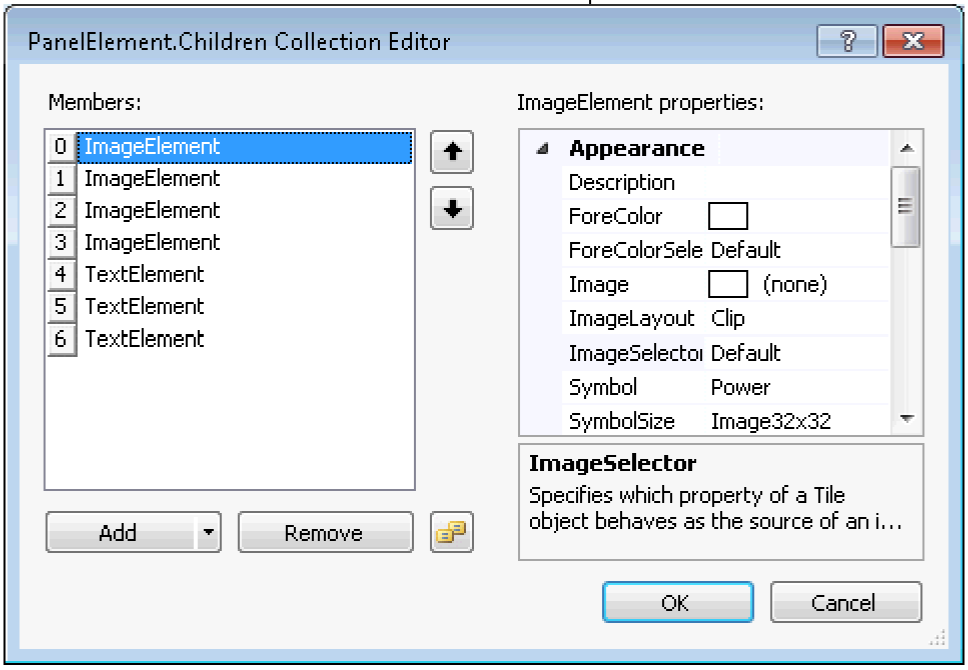
In the previous step you created a WinForms application and added the **C1TileControl** to your project.

To add a panel to your tile with image and text elements, complete the following steps:

To add a template with elements to the Tile at design time:

1. Right-click the default **Tile 1** and select **Edit Templates**. The **C1TileControl.Templates Collection Editor** appears.
2. Click Add to add a new template to Tile1.

3. Click on the ellipsis button next to **Elements** to open the **Template.Elements Collection Editor**. Select **Add** and click on the dropdown arrow to select the **PanelElement** and then the **TextElement**. This will add elements into the TemplateCollection.
4. Select PanelElement from the Members list and enter **9** next to the **ChildSpacing** property.
5. Click on the ellipsis button next to Children so the **PanelElement.Children Collection Editor** appears. Select **Add** and click on the dropdown arrow to select the **ImageElement**. Add 4 **ImageElements** and 3 **TextElements**. This will add elements into the TemplateCollection.



Description	
ForeColor	
ForeColorSele	Default
Image	(none)
ImageLayout	Clip
ImageSelectoı	Default
Symbol	Power
SymbolSize	Image32x32

ImageSelector
Specifies which property of a Tile object behaves as the source of an i...

6. Select the first **[0] ImageElement** from the **Members** list and set its properties to the following:
   - **ImageSelector** property to **UnboundSymbol**. This will make the symbol that you select act as an image for the specified Tile.
   - **Symbol** property to **LeftToRight**. This will make the LeftToRight image appear on the specified Tile.
   - **SymbolSize** property to **Image64x64**. This will change the default symbol size from **32x32** to **64x64**
   - **FixedHeight** to **70**. This will set the height of the contents in the panel to 70 pixels.
7. Select the second **[1]ImageElement** from the **Members** list and set its properties to the following:
   - **ForeColor** property to **255, 192, 255**
   - **ForeColorSelector** property to **Unbound**
   - **ImageSelector** property to **UnboundSymbol**
   - **Symbol** property to **CircleWithPlus**
8. Select the third **[2]ImageElement** from the **Members** list and set its properties to the following:
   - **ForeColor** property to **255, 192, 255**
   - **ForeColorSelector** property to **Unbound**
   - **ImageSelector** property to **UnboundSymbol**
   - **Symbol** property to **CircleWithMinus**
   - **Alignment** property to **TopCenter**. This will align the symbol to the TopCenter of the Panel.
9. Select the fourth **[3]ImageElement** from the **Members** list and set its properties to the following:
   - **ForeColor** property to **255, 224, 192**
   - **ForeColorSelector** property to **Unbound**
   - **ImageSelector** property to **UnboundSymbol**
   - **Symbol** property to **CircleWithMultiply**
   - **Alignment** property to **BottomCenter**
10. Select the fifth **[4]TextElement** from the **Members** list and set its properties to the following:
    - **ForeColor** property to **192, 192, 255**
    - **ForeColorSelector** property to **Unbound**
    - **Text** property to **Top**
    - **TextSelector** to **Unbound**

- **Alignment** to **TopCenter**

11. Select the sixth **[5]TextElement** from the **Members** list and set its properties to the following:
    - **BackColorSelector** property to **Unbound**
    - **ForeColor** property to **255**, **224, 192**
    - **ForeColorSelector** property to **Unbound**
    - **Text** property to **Bottom**
    - **TextSelector** property to **Unbound**. This will make the new text Bottom appear rather than the default text.
    - **Alignment** property to **BottomCenter**
    - **DirectionVertical** property to **True**
12. Select the seventh **[6]TextElement** from the **Members** list and set its properties to the following:
    - **BackColorSelector** property to **Unbound**
    - **ForeColor** property to **255**, **192**, **255**
    - **ForeColorSelector** property to **Unbound**
    - **Text** property to **Middle**
    - **TextSelector** property to **Unbound**
13. Click **OK** to save and close the **PanelElement.Children Collection Editor**
14. In the **Template.Elements Collection Editor** select the [1] TextElement from the Members list**.**
15. Select the second **[1]TextElement** from the **Members** list and set its properties to the following:
    - **Alignment** property to **BottomCenter**
    - **Margin** property to **0, 0, 0, 5**
16. Click **OK** to save and close the **Template.Elements Collection Editor**
17. Click **OK** to save and close the **C1TileControl.Templates Collection Editor**

Run and observe the following:

Tile1 will appear the same since the template that we created, Template1, has not been applied to the Tiles **Template** property.

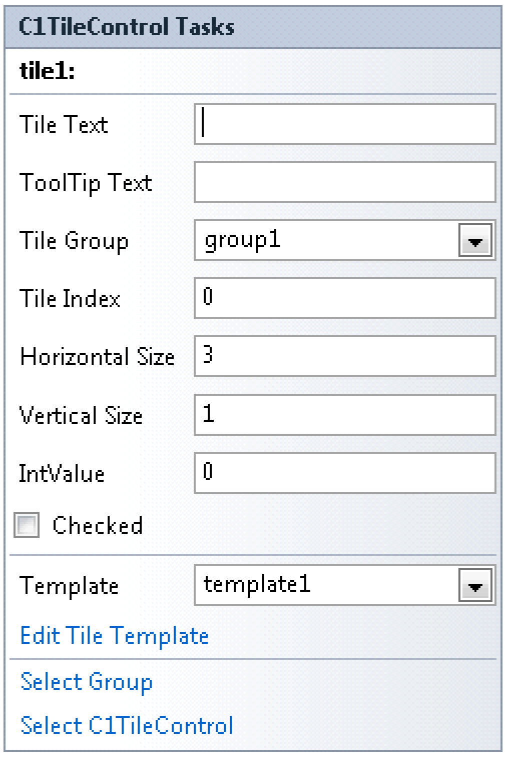
In the next step you will learn how to assign the new template to the specified tile as well as modify a few of the Tiles properties using the **C1TileControl Tasks** menu.

Step 3 of 3: Applying the Template to the Specified Tile

In the previous step you created a template and added template elements such as panels, images, and text. You also set the alignment and layout properties for the elements in the template. In this step we will apply the template to the specified tile and set a few tile properties such as the BackColor, Template, HorizontalSize, and Text property.

To apply the template to the first tile as well as modify a few properties for the first tile of the **C1TileControl**, complete the following steps:

1. Select **Tile 1** to open its **C1TileControl Tasks** menu.
2. In the **C1TileControl Tasks** menu remove the default text, Tile1, set the **Template** property to **Template** and **HorizontalSize** property to **3**.


C1TileControl Tasks

tile1:

Tile Text	
ToolTip Text	
Tile Group	group1
Tile Index	0
Horizontal Size	3
Vertical Size	1
IntValue	0

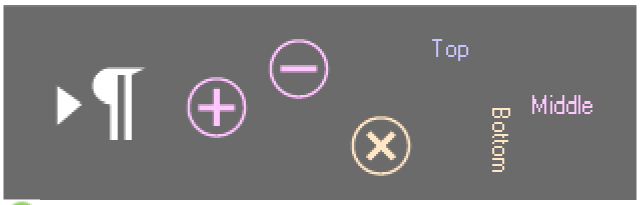
☐ Checked

Template	template1

3. Right-click on the first Tile you have modified and select **Edit Groups**. The **C1TileControl.Groups Collection Editor** appears.
4. Click on the ellipsis button next to the **Tiles** to open the **Group.Tiles Collection Editor** and select tile1[].
5. Set the **BackColor** property to **DimGrey** for **tile1[]**.
6. Click **OK** to save and close the **Group.Tiles Collection Editor** and click **OK** to save and close the **C1TileControl.Groups Collection Editor**.

Run and observe the following:

Template1 is applied to the first Tile as well as the Tile settings.



What You've Accomplished

Congratulations! You have successfully completed the C1TileControl quick start. In this topic, you added a C1TileControl to your windows form, created a template for a specific tile, and set a few of the tiles properties.

Design-Time Support

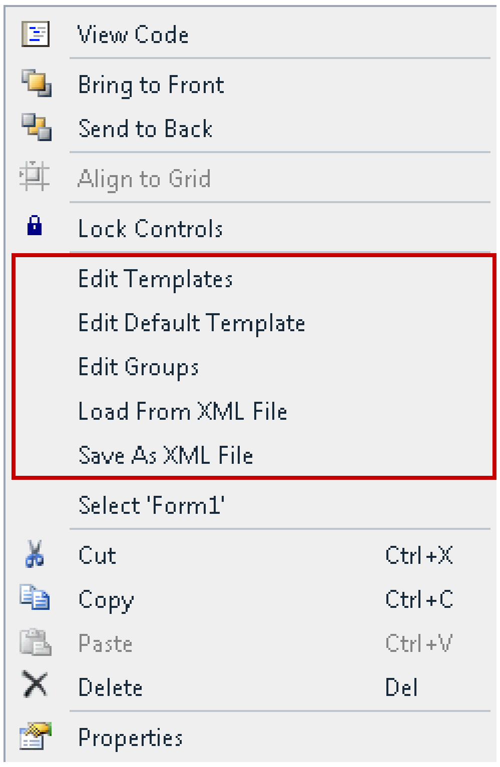
C1TileControl provides customized context menus, smart tags, and a designer that offers rich design-time support and simplifies working with the object model.

The following topics describe how to use **C1TileControl** design-time environment to configure **C1TileControl**.

C1TileControl Context Menu

The **C1TileControl** control provides a context menu for additional functionality to use at design time.

To access C1TileControls context menu, right-click on the **C1TileControl** control and the context menu for it appears like the following:



The **C1TileControl** context menu operates as follows:

- ### Edit Templates

  Selecting the **Edit Templates** opens the **C1TileControl.Templates Collection Editor** where you can add, remove, or modify templates.

- ### Edit Default Template

  Selecting the **Edit Default Template** item opens the **Template.Elements Collection Editor** where you can add text, image, and panels to the templates in the C1TileControl.

- ### Edit Groups

Selecting the **Edit Groups** item opens the **C1TileControl.Groups Collection Editor** where you can add, remove, or modify groups for the C1Tile control.

- ### Load From Xml File

  Selecting the **Load from Xml File** opens the **Load From Xml File** dialog box where you browse to the .xml file you wish to load

- ### Save As Xml File

  Selecting the **Save As Xml File** opens the **Save As Xml File** dialog box where you browse to the .xml file you wish to save.

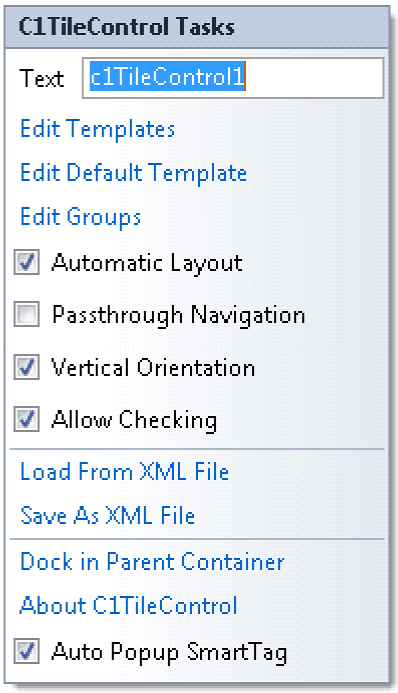
- ### About C1TileControl

  Clicking **About C1TileControl** shows a dialog box. This dialog box displays the version number and licensing information for the C1TileControl product.

C1TileControl Smart Tag

In Visual Studio, each component in **TileControl for WinForms** includes a smart tag. A smart tag represents a short-cut tasks menu that provides the most commonly used properties in each control.

To access the **C1TileControl Tasks** menu, click the smart tag (▸) in the upper right corner of the **C1TileControl** control. This will open the **C1TileControl Tasks** menu.



C1TileControl Tasks

Text: c1TileControl1

Edit Templates

Edit Default Template

Edit Groups

✓ Automatic Layout

☐ Passthrough Navigation

✓ Vertical Orientation

✓ Allow Checking

Load From XML File

Save As XML File

Dock in Parent Container

About C1TileControl

✓ Auto Popup SmartTag

The **C1TileControl Tasks** menu operates as follows:

Text

Clicking in the textbox next to the Text item will create text that appears on the top of the TileControl.

Edit Templates

Clicking the **Edit Templates** opens the **C1TileControl.Templates Collection Editor** where you can add, remove, or modify templates..

Edit Default Template

Clicking the **Edit Default Template** item opens the **Template.Elements Collection Editor** where you can add text, image, and panels to the templates in the **C1TileControl**.

Edit Groups

Clicking the **Edit Groups** item opens the **C1TileControl.Groups Collection Editor** where you can add, remove, or modify groups for the C1TileControl.

Vertical Orientation

Selecting the **Vertical Orientation** checkbox will align the C1TileControl vertically

Automatic Layout

When selected, enables automatic layout.

Passthrough Navigation

When selected, enables passthrough navigation.

Allow Checking

When selected, enables checking.

Load From Xml File

Clicking the **Load from Xml File** opens the **Load From Xml File** dialog box where you browse to the .xml file you wish to load.

Save As Xml File

Clicking the **Save As Xml File** opens the **Save As Xml File** dialog box where you browse to the .xml file you wish to save.

Dock in Parent Container

Clicking Dock in Parent Container will dock the C1TileControl in its parent container.

About C1TileControl

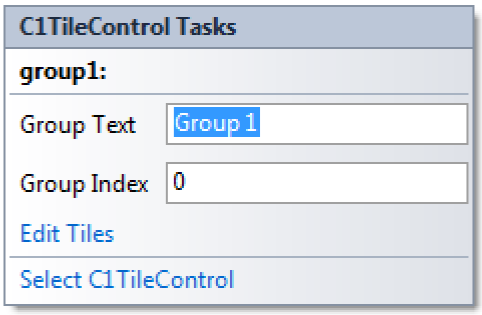
Clicking **About C1TileControl** shows a dialog box. This dialog box displays the version number and licensing information for the GanttView product.

Auto Popup SmartTag

Unselecting the **Auto Popup SmartTag** checkbox will disable the popup smart tag when you click on the group or each tile/panel.

Group Tasks

To access the **C1TileControl Tasks** menu, click any of the groups in the **C1TileControl** control. This will open the **C1TileControl Tasks** menu.


C1TileControl Tasks
group1:
Group Text: Group 1
Group Index: 0

The **C1TileControl Tasks** menu operates as follows:

Group Text

Clicking in the textbox next to the **Group Text** item will create text that appears on the top of the **Group** in the C1TileControl.

Group Index

Specifies the position of the Tile within the group.

Edit Tiles

Clicking the **Edit Tiles** item opens the **Group.Tiles Collection Editor** where you can add, remove, or modify the tiles within the group.

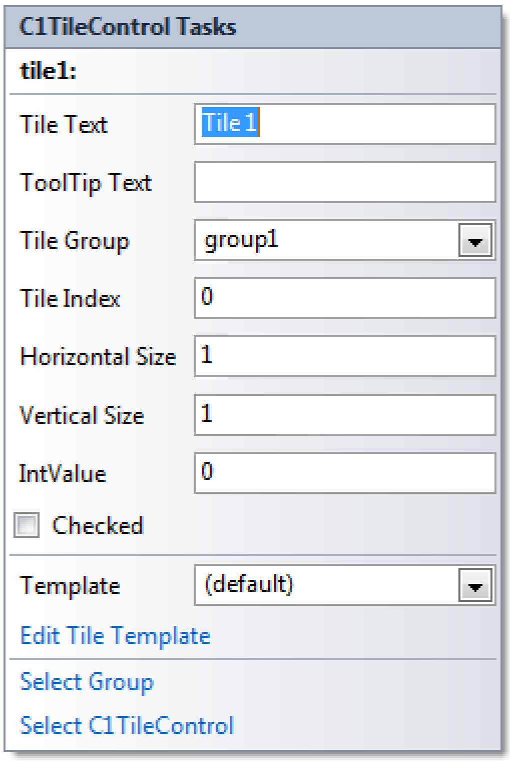
Select C1TileControl

Clicking the **Select C1TileControl** item selects the C1TileControl.

Tile Tasks

To access the **C1TileControl Tasks** menu, click any of the tiles in the **C1TileControl** control. This will open the

C1TileControl Tasks menu.



C1TileControl Tasks

tile1:

Tile Text	Tile 1
ToolTip Text	
Tile Group	group1
Tile Index	0
Horizontal Size	1
Vertical Size	1
IntValue	0

☐ Checked

Template	(default)

Edit Tile Template

Select Group

Select C1TileControl

The **C1TileControl Tasks** menu operates as follows:

Tile Text

Clicking in the textbox next to the **Tile Text** item will create text that appears on the top of the **Tile** in the C1TileControl.

ToolTip Text

Clicking in the textbox next to the **ToolTip Text** item will create text that appears when you hover over the text in the **Tile** of the C1TileControl.

Tile Group

Clicking the dropdown arrow will show a menu that lists the existing groups. Select the group that you wish the tile to be in.

Tile Index

Specifies the position of the Tile within the group.

Horizontal Size

Specifies the horizontal size of the **Tile**.

Vertical Size

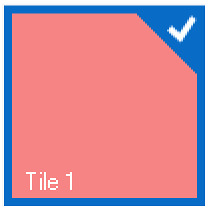
Specifies the vertical size of the **Tile**.

IntValue

Specifies the integer value of the **Tile**.

Checked

Clicking on the checkbox will enable the checkmark so the Tile will have a checkmark on it that appears like the following:



Edit Tile Template

Clicking the **Edit Tile Template** item opens the **Template.Elements Collection Editor** where you can add text, image, and panels to the templates in the **Tiles**.

Select Group

Clicking the **Select Group** item selects the Group where the Tile is located within.

Select C1TileControl

Clicking the **Select C1TileControl** item selects the C1TileControl.

C1TileControl Collection Editors

C1TileControl provides the following collection editors that allow you to apply properties to the C1TileControl elements at design time:

- **C1TileControl.Templates Collection Editor**
- **Template.Elements Collection Editor**
- **C1TileControl.Groups Collection Editor**
- **Group.Tiles Collection Editor**
- **C1TileControl.PropertyMappings Collection Editor**

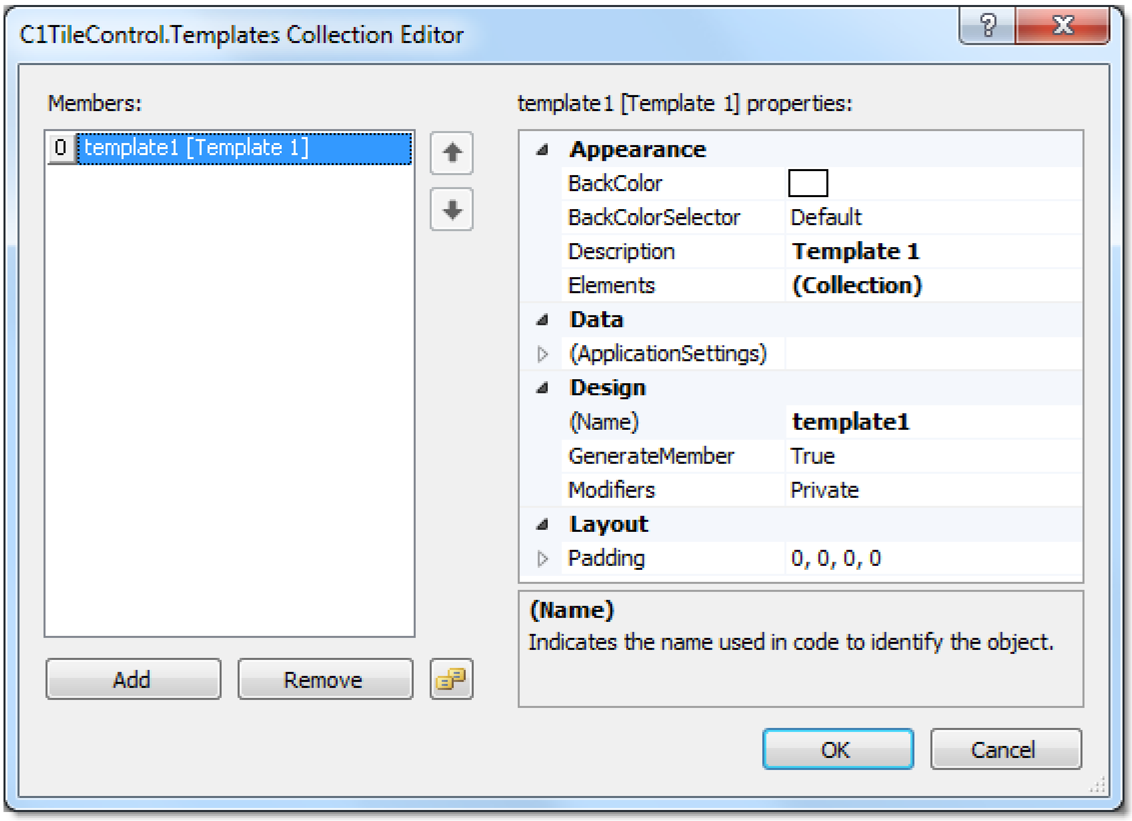
The following topics provide an overview of each C1TileControl collection editor and show how to access each of them:

C1TileControl.Templates Collection Editor

The **C1TileControl.Templates Collection Editor** is used for adding templates to the C1TileControl. A template can hold elements such as text, images, and panels. These elements can be added to each template at design time through the **Template.Elements Collection Editor** or programatically. Each text, image, or panel element can hold children elements (text, image, and/or panel). For example, a panel element can include multiple text and image elements.

To Access the C1TileControl.Templates Collection Editor

Right-click on the **C1TileControl** and select **Edit Templates** from its context menu. The **C1TileControl.Templates Collection Editor** appears like the following when a member is added to the collection:


C1TileControl.Templates Collection Editor

Members:

0 template1 [Template 1]

template1 [Template 1] properties:

Appearance	
BackColor	⬜
BackColorSelector	Default
Description	**Template 1**
Elements	**(Collection)**
Data	
(ApplicationSettings)	
Design	
(Name)	**template1**
GenerateMember	True
Modifiers	Private
Layout	
Padding	0, 0, 0, 0

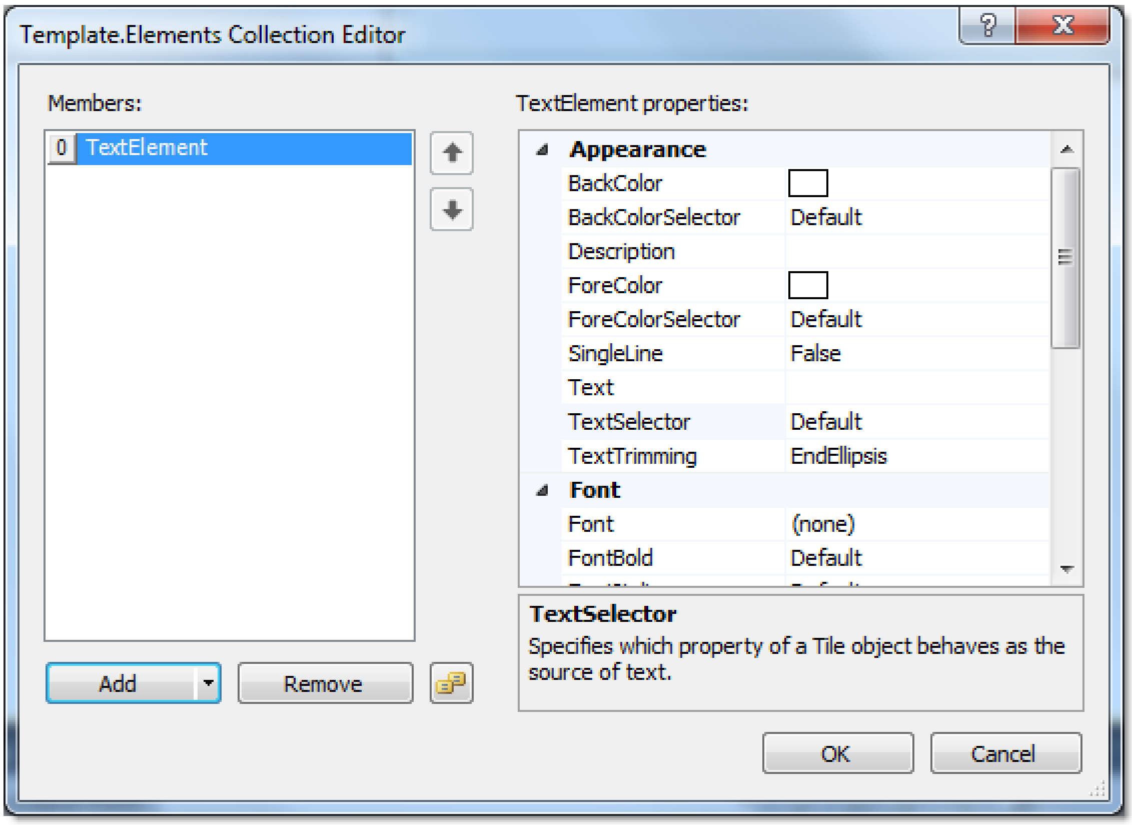
(Name)
Indicates the name used in code to identify the object.

Template.Elements Collection Editor

The **Template.Elements Collection Editor** is used for adding, removing, or modifying panel elements such as at design time.

To Access the Template.Elements Collection Editor

Right-click on the **C1TileControl** control and select **Edit Templates** from its context menu. The **C1TileControl.Templates Collection Editor** appears. Click **Add** to add a template item to the collection. Click on the ellipsis button in the **Elements** property. The **Template.Elements Collection Editor** appears. Click on the dropdown listbox and select a member such as **TextElement** to modify the TextElements properties.



TextElement properties:

Appearance	
BackColor	☐
BackColorSelector	Default
Description	
ForeColor	☐
ForeColorSelector	Default
SingleLine	False
Text	
TextSelector	Default
TextTrimming	EndEllipsis
Font	
Font	(none)
FontBold	Default

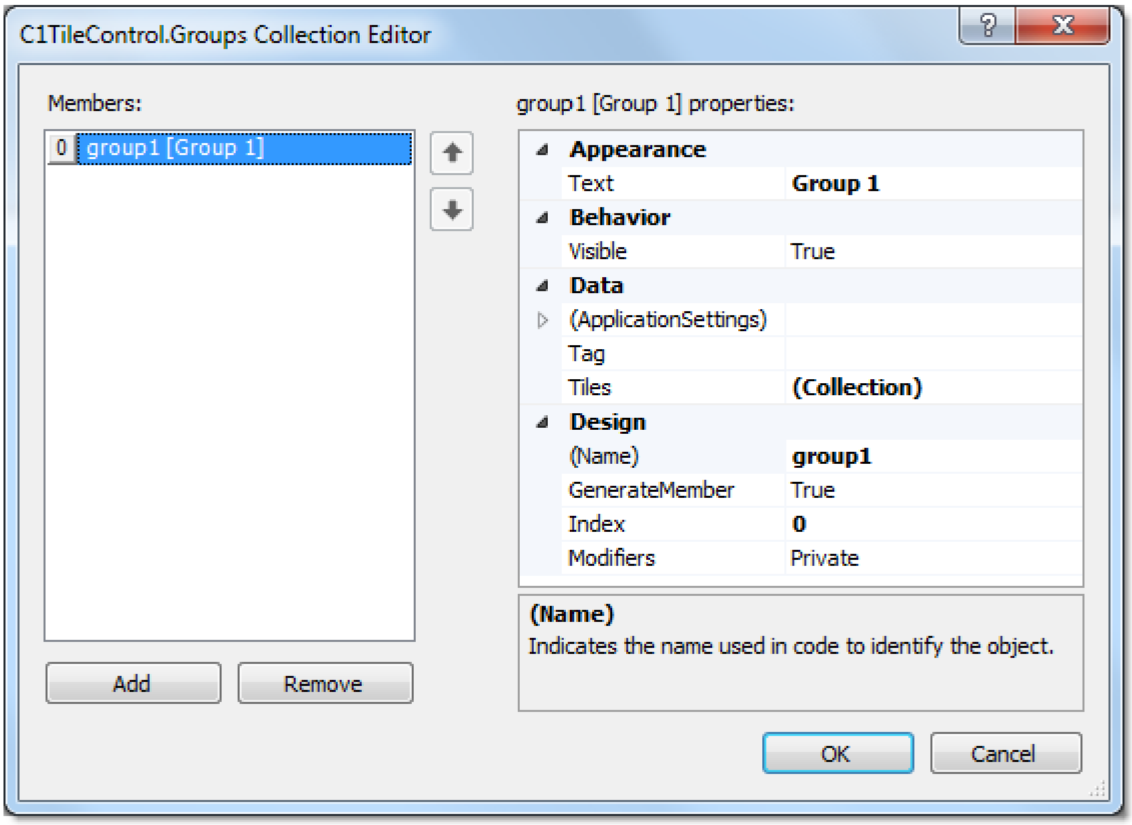
TextSelector
Specifies which property of a Tile object behaves as the source of text.

C1TileControl.Groups Collection Editor

The **C1TileControl.Groups Collection Editor** is used for adding, removing, or modifying groups within the C1TileControl at design time.

To Access the C1TileControl.Groups Collection Editor

Right-click on the **C1TileControl** and select **Edit Groups** from its context menu. The **C1TileControl.Groups Collection Editor** appears like the following when a member is added to the collection:



group1 [Group 1] properties:

Appearance	
Text	**Group 1**
Behavior	
Visible	True
Data	
(ApplicationSettings)	
Tag	
Tiles	**(Collection)**
Design	
(Name)	**group1**
GenerateMember	True
Index	**0**
Modifiers	Private

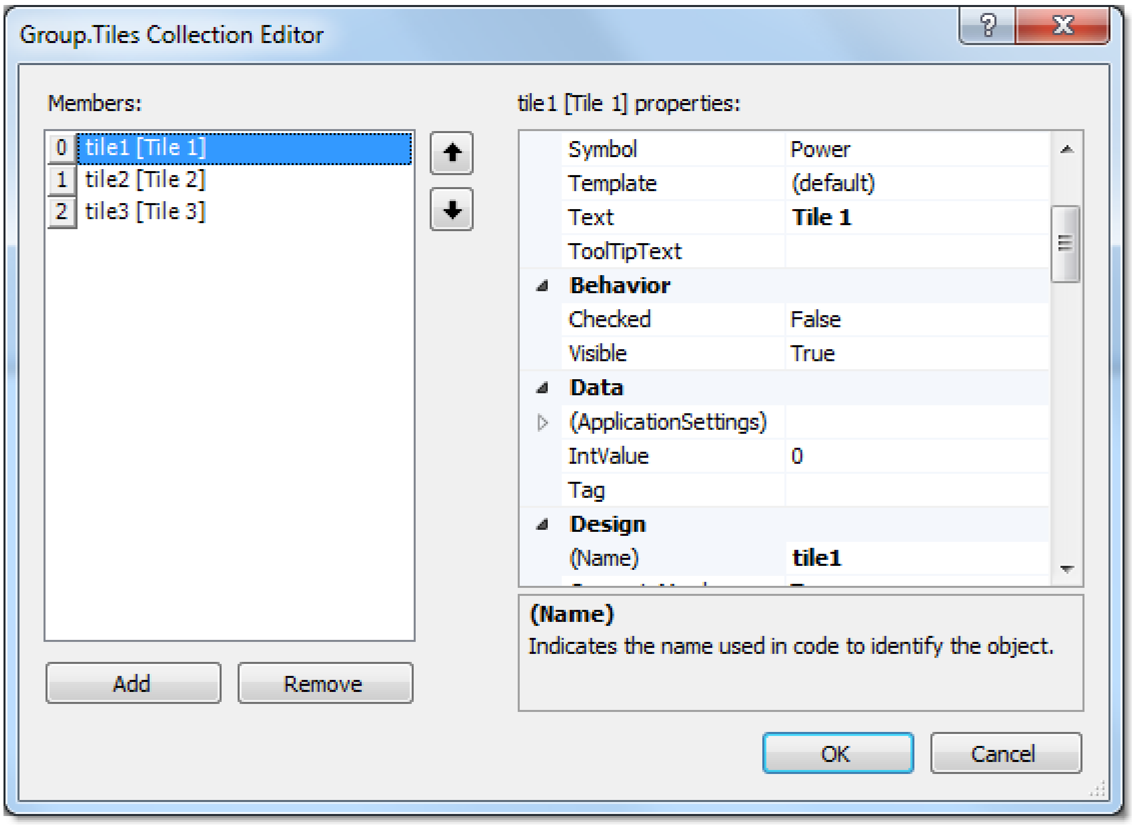
(Name)
Indicates the name used in code to identify the object.

Group.Tiles Collection Editor

The **Group.Tiles Collection Editor** is used for adding, removing, or modifying tiles within the group of the C1TileControl.

To Access the Group.Tiles Collection Editor

Click on any group in the C1TileControl. Select **Edit Tiles** from the C1TileControl-Group tasks menu. The **Group.Tiles Collection Editor** appears:


Group.Tiles Collection Editor

Members:

0 tile1 [Tile 1]
1 tile2 [Tile 2]
2 tile3 [Tile 3]

tile1 [Tile 1] properties:

Symbol	Power
Template	(default)
Text	**Tile 1**
ToolTipText	
Behavior	
Checked	False
Visible	True
Data	
(ApplicationSettings)	
IntValue	0
Tag	
Design	
(Name)	**tile1**

(Name)
Indicates the name used in code to identify the object.

C1TileControl PropertyMappings Collection Editor

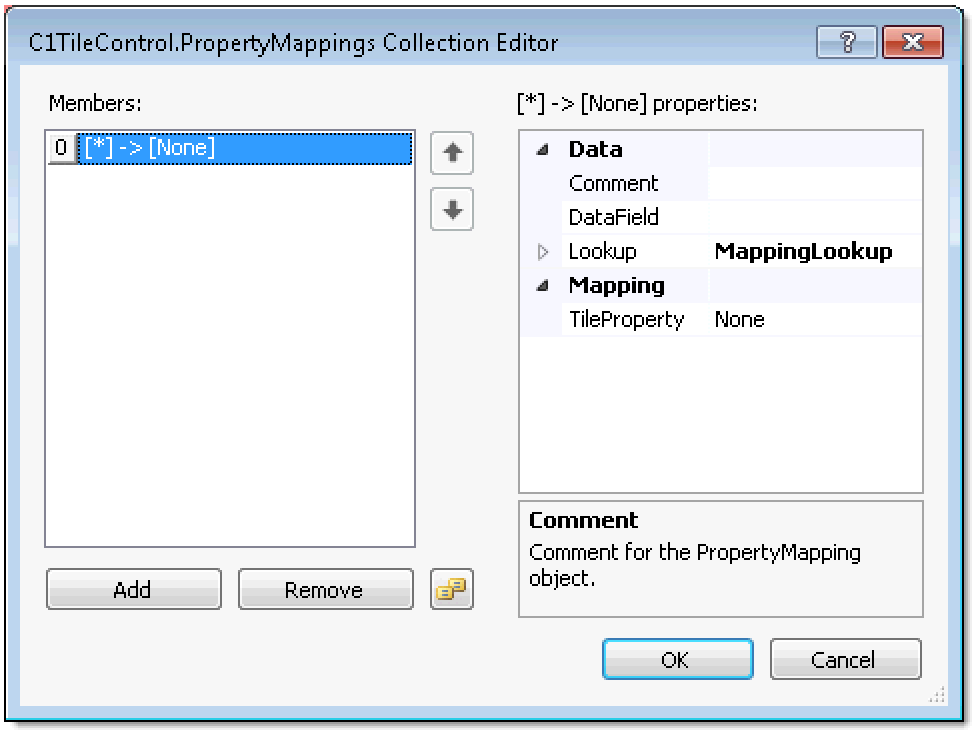
C1TileControl PropertyMappings Collection Editor

The **PropertyMapping Collection Editor** is used for adding, removing, or modifying data mapping objects within the C1TileControl.

To Access the C1TileControl PropertyMapping Collection Editor

1. Right-click on the **C1TileControl** and select **Properties**.
2. In the Properties window click on the ellipsis button next to **PropertyMappings**.

The **C1TileControl.PropertyMappings Collection Editor** appears:



[*] -> [None] properties:

Data	
Comment	
DataField	
Lookup	**MappingLookup**
Mapping	
TileProperty	None

Comment
Comment for the PropertyMapping object.

The **Members: listbox** includes the members. To add a member click the **Add** button. Once a member is added you can assign the properties to the member.

The following properties appear in the **C1TileControl.PropertyMappings Collection Editor**:

Property	Description
PropertyMapping.Comment	Gets or sets the comment for the C1.Win.C1Tile.PropertyMapping object.
PropertyMapping.DataField	Gets or sets the field of the data source which is mapped to the tile property.
MappingLookup.DataSource	Specifies the data source object for the lookup.
MappingLookup.DisplayMember	The field to use as the source of meaningful value.
MappingLookup.ValueMember	The field to use as the source of value (primary key) matching the foreign key in the DataField.
PropertyMapping.TileProperty	Specifies the property of the Tile object that is a destination for mapping.

TileControl Layout

The TileControl uses either automatic or manual layout of tiles in the groups. The AutomaticLayout property determines whether the tiles are arranged automatically or manually. When manual layout is used, you can drag and drop the tiles anywhere on the form. Tiles may be any size; they are not limited to large and small sizes only. The interior layout of the tiles (or rather tile templates) is very flexible. You can use docked and stacked panels, nested panels, text elements, and images. Additionally, you can save the layout to an XML file and load it from an XML file at any time.

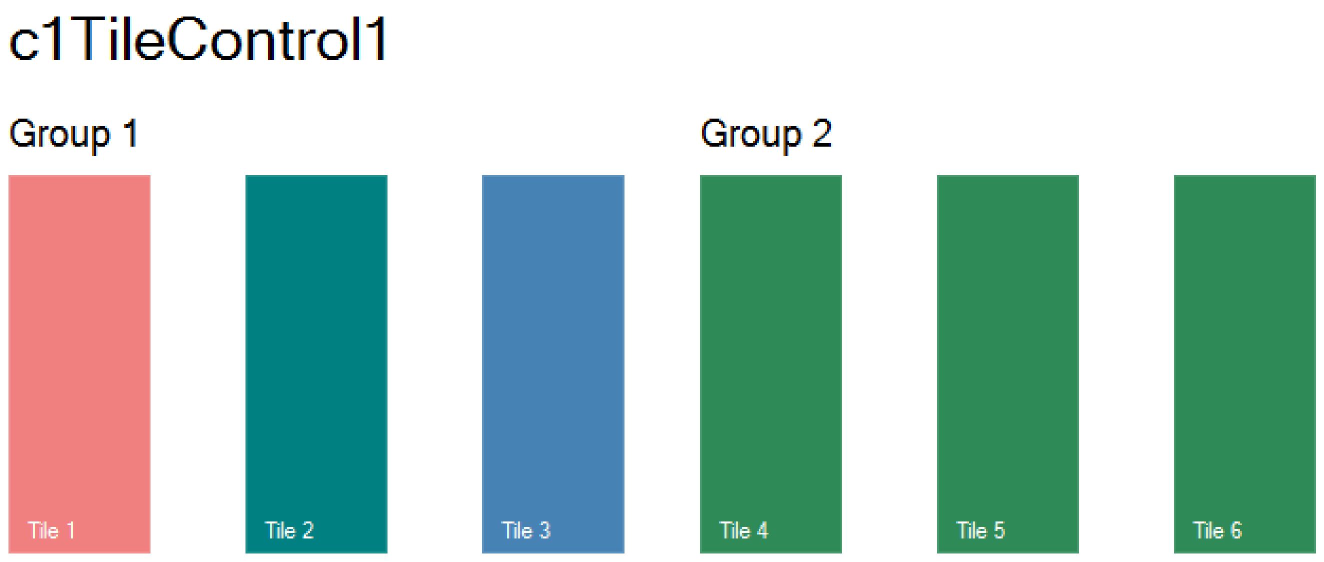
The layout of the Tiles in the TileControl is fully customizable through the layout properties. The tiles are arranged in each group horizontally by default, but can be changed to vertically through the Orientation property. Use the horizontal layout mode to make the Tile control appear like the Windows 8 UI and use the vertical layout to make the Tile control appear like an advanced listbox.

The following table lists the common surface/layout properties for all Tiles in the C1TileControl:

Property	Description
AutomaticLayout	Indicates whether the tiles should be arranged automatically or manually.
CellHeight	Specifies the height of a single tile cell.
CellSpacing	Specifies the gap between tile cells in a group.
CellWidth	Specifies the width of a single tile cell.
MaximumRowsOrColumns	Specifies the maximum number of cell rows or columns in automatic layout mode.
Orientation	Specifies the method of arranging of the tile groups.
ScrollBarStyle	Specifies whether the default or system scroll bar should appear.
ScrollOffset	Specifies the negative or zero offset of the scrollable area.
SurfaceContentAlignment	Specifies the alignment of groups on the scrollable surface.
SurfacePadding	The interior spacing of the scrollable surface.

Once you apply the settings to the preceding Surface properties any new Tiles added will have those same settings. If you want different layouts for the tiles then you will need to apply unique templates to the tiles.

When a new Group is added the Tiles appear horizontal by its default Orientation setting. The following image illustrates the Horizontal orientation of the Tiles.


c1TileControl1
Group 1
Group 2
Tile 1
Tile 2
Tile 3
Tile 4
Tile 5
Tile 6

TileControl Behavior

The following section details the behavior properties used to control the behavior of the tiles in the TileControl.

TileControl Scrolling

The Tiles in the TileControl are scrollable by default.

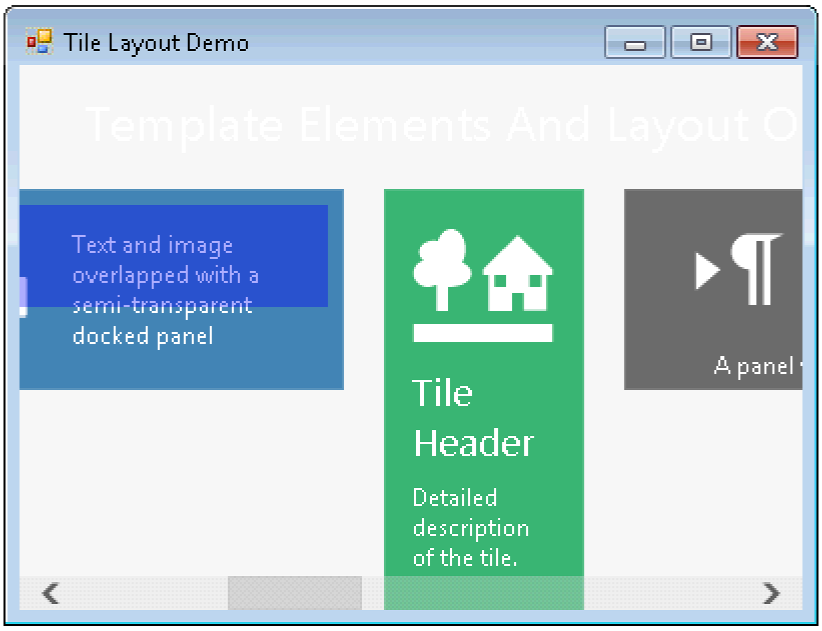
C1TileControls scrollbar appearance can be determined by the ScrollBarStyle property.

The following table represents the two difference scroll bar styles to choose from when you set its ScrollBarStyle property:

Value	Description
Default	Specifies the default scrollbar for the C1TileControl.
System	Specifies the System scrollbar for the C1TileControl.

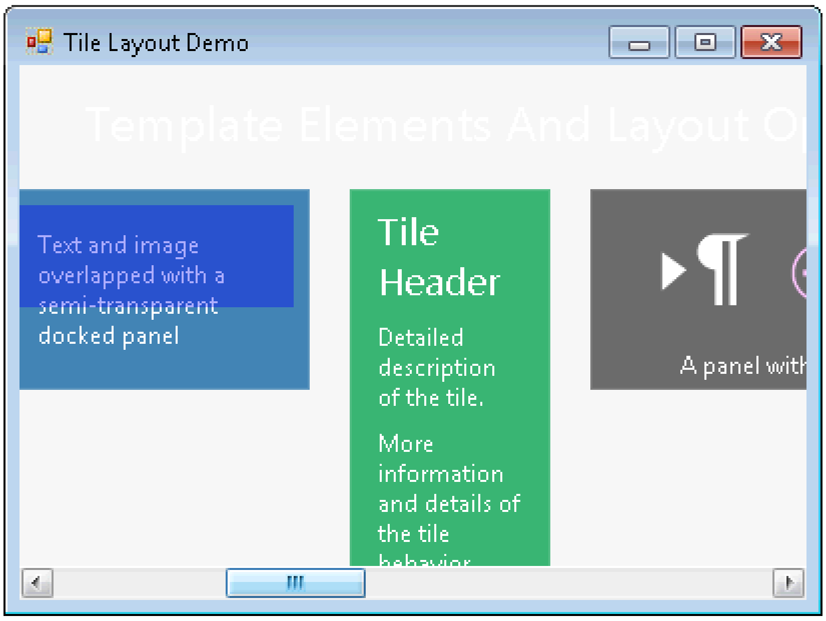
Default scrollbar

When you hover over the default scrollbar, the bar changes color. The default scrollbar appears like the following:


Tile Layout Demo
Template Elements And Layout O
Text and image overlapped with a semi-transparent docked panel
Tile Header
Detailed description of the tile.
A panel

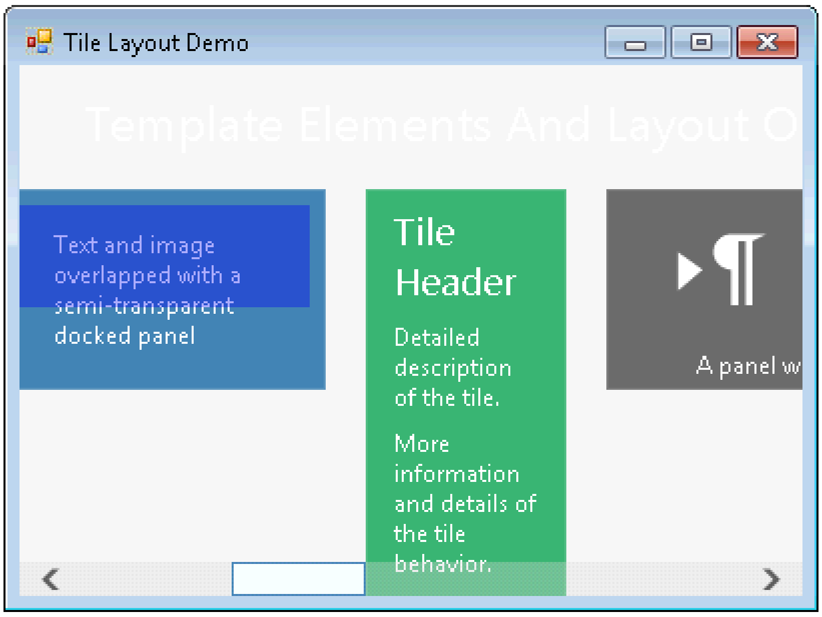
System scrollbar

When you hover over the system scrollbar, the bar changes color. The system scrollbar appears like the following:


Tile Layout Demo
Template Elements And Layout O
Text and image overlapped with a semi-transparent docked panel
Tile Header
Detailed description of the tile.
More information and details of the tile behavior.
A panel with

The color of the scrollbar thumb border and the scrollbar thumb interior can be specified using the SBThumbBorderColor and SBThumbInnerColor properties respectively. The negative or zero offset of the scrollable area can be determined using the ScrollOffset property.

The following image illustrates the effects of the SBThumbBorderColor and SBThumbInnerColor properties. The SBThumbBorderColor is set to **SteelBlue** and the SBThumbInnerColor is set to **AliceBlue**.


Tile Layout Demo
Template Elements And Layout O
Text and image overlapped with a semi-transparent docked panel
Tile Header
Detailed description of the tile.
More information and details of the tile behavior.
A panel w

TileControl Navigation

The PassthroughNavigation property gets or sets whether the keyboard should navigate to the next row/column after focusing the last tile in the current row/column for vertical and/or horizontal layout.

TileControl Touchscreen Support

The TileControl supports panning, tapping, and checking tiles with a swipe gesture using the touch input hardware on a machine with Windows 7 or Windows 8. A visual cue is shown when you reach the end of the pannable area of the

C1TileControl. To disable the visual cue, set the AllowPanningFeedback to **False**.

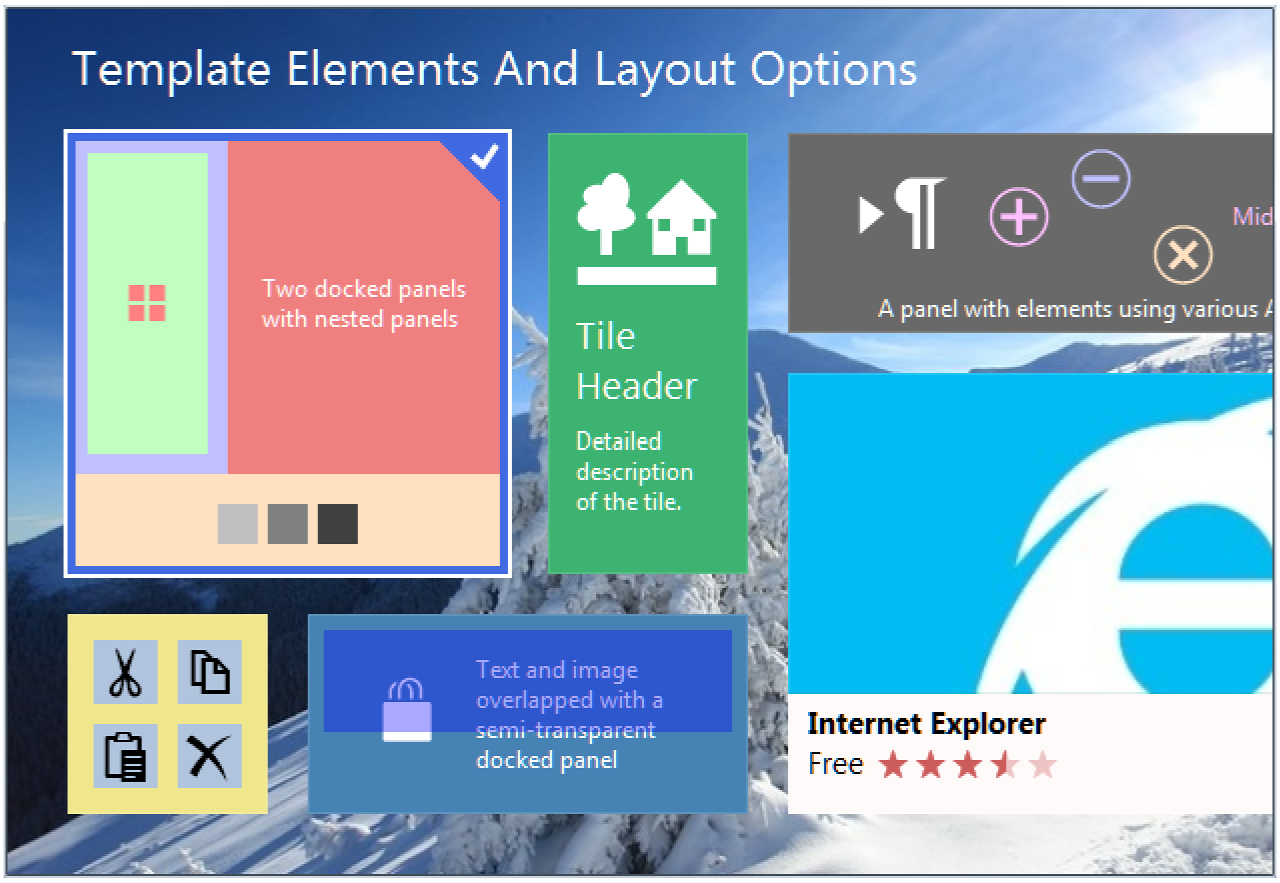
When the AllowChecking property is enabled you can check/uncheck tiles using a swipe gesture or right-clicking the mouse.

The ShowTooltips property is enabled by default so the tooltips appear when you press and hold a tile with your fingertip on a touchscreen. You can specify the time, in milliseconds, that passes before the tooltips appear using the ToolTipInitialDelay property.

TileControl Templates

Tiles and templates are the most important components of the C1TileControl. Tile provides the data and the template provides the visualization pattern. You can switch the templates for the same tile. For example, the first template may show the tile image and the second may show the detailed tile text. Also, you can apply the same template to multiple tiles.

Templates may consist of three possible elements: panel, text, and image. Panel elements of the type PanelElement class may contain child elements including nested panels. The child elements can be added at design time through the designer or programmatically through the Children property. Text elements of the type TextElement can be added at design time through the designer or programmatically through TextElement class. Image elements of the type ImageElement can be added at design time through the designer or programmatically through the ImageElement class. There are a number of layout settings that give you full freedom in laying out the template elements.


Template Elements And Layout Options
Two docked panels with nested panels
Tile Header
Detailed description of the tile.
Mid
A panel with elements using various A
Text and image overlapped with a semi-transparent docked panel
Internet Explorer
Free

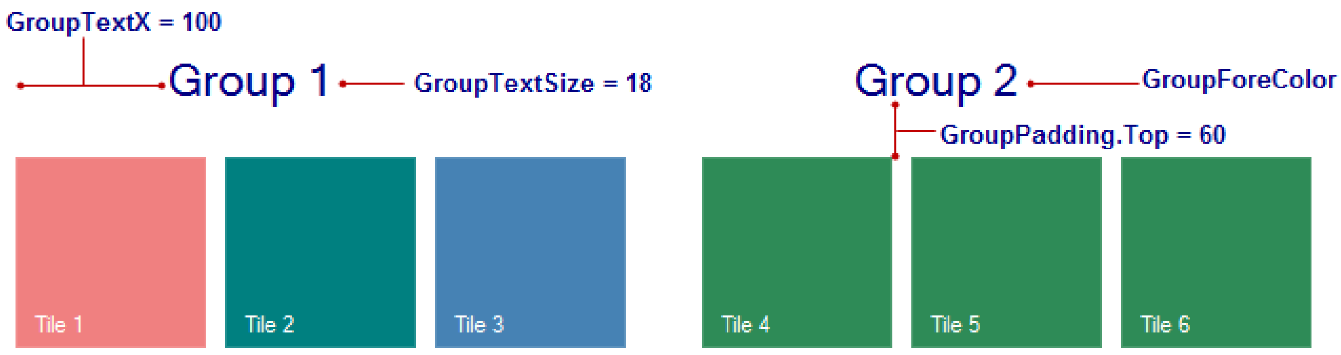
TileControl Groups

TileControl Groups are of the type Group class. A TileControl can have one or more groups. Each group can have one or more tiles of the type Tile class. Each group may include a caption that represents the name for the group. The groups caption is specified by the Text property. The group captions font, forecolor, padding, text size, and position can be modified. The groups caption can also be bolded.

A group arranges its items according to the Orientation property. The spaces between the groups can be specified through the GroupSpacing property so a different group can appear detached from the other group.

A Group can be added to the **C1TileControl** at design time through the **C1TileControl.Groups Collection Editor** or programmatically through the Groups property. When a Group is added to the designer it will appear empty on the C1TileControl. You will need to add tiles to the group. The tiles can be added by clicking on the ellipsis button next to the Tiles property and clicking the Add button. Once the tiles are added the default group caption will appear. To see how to add groups to a C1TileControl see Adding Groups to the C1TileControl.

The following image illustrates the effects of the Groups appearance properties:



The C1TileControl Groups appearance and layout properties are listed in the table below:

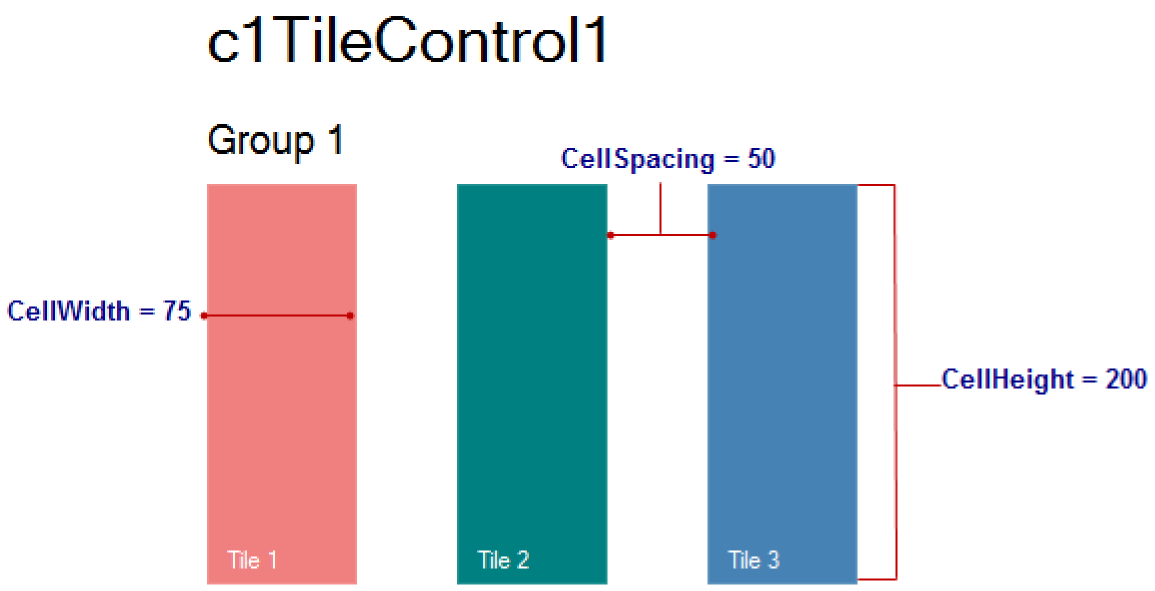
Property	Description
GroupFont	Specifies the font for group captions.
GroupForeColor	Specifies the foreground color of a group caption.
GroupPadding	The interior spacing of a tile group.
GroupTextBold	Indicates whether the group caption font is bold. This property takes precedence over the GroupFont properties.
GroupTextSize	Specifies the font size for the group captions. This property takes precedence over the GroupFont properties. The default text size is 15 pixels.
GroupTextX	Specifies the horizontal offset of a group caption. The default horizontal offset size is 20 pixels.
GroupTextY	Specifies the vertical offset of a group caption. The default vertical offset size is 5 pixels.

TileControl Tiles and Elements

The Tile class represents a single Tile which can hold several types of elements such as images (ImageElement), text (TextElement), and panels (PanelElement). The Tiles are one of the most important component of the TileControl; they are responsible for the data. The Tiles in the TileControl in their simplest form appear like button controls. The images, text, and panels can be easily be formatted through the properties.

A Tile can be added to the **C1TileControl** at design time through the **Group.Tiles Collection Editor** or programmatically through the Tiles property. When a Tile is added to the designer it will be added to the specified group of the C1TileControl. A single group appears by default so you can easily start adding the tiles into the group. The tiles can be added at design time by clicking on the ellipsis button next to the **Tiles** property and clicking the **Add** button. To see how to add tiles to a specific group in the C1TileControl see Adding Tiles to a Specific Group.

The following image illustrates the effects of a few of the Tiles appearance properties:



The C1TileControl Groups appearance and layout properties are listed in the table below:

Property	Description
BackColor	Gets or sets the background color for the tile.
BackColor1	Gets or sets the first additional background color for the tile.
BackColor2	Gets or sets the second additional background color for the tile.
BackColor3	Gets or sets the third additional background color for the tile.
BackColor4	Gets or sets the fourth additional background color for the tile.
BackColor5	Gets or sets the fifth additional background color for the tile.
Checked	Gets or sets whether the tile is checked.
ForeColor	Gets or sets the foreground color for the tile.
ForeColor1	Gets or sets the first additional foreground color for the tile.
ForeColor2	Gets or sets the second additional foreground color for the tile.
ForeColor3	Gets or sets the third additional foreground color for the tile.

ForeColor4	Gets or sets the fourth additional foreground color for the tile.
ForeColor5	Gets or sets the fifth additional foreground color for the tile.
Height	Gets or sets the height of the tile, in pixels.
HorizontalSize	Gets or sets the width of the tile, in pixels.
Image	Gets or sets an image displayed on the tile.
Image1	Gets or sets the first additional image that can be displayed on the tile.
Image2	Gets or sets the second additional image that can be displayed on the tile.
Image3	Gets or sets the third additional image that can be displayed on the tile.
Image4	Gets or sets the fourth additional image that can be displayed on the tile.
Image5	Gets or sets the fifth additional image that can be displayed on the tile.
Symbol	Gets or sets a symbol associated with the tile.
Template	Gets or sets the tile template.
Text	Gets or sets the text on the tile.
Text1	Gets or sets the first additional text string for the tile.
Text2	Gets or sets the second additional text string for the tile.
Text3	Gets or sets the third additional text string for the tile.
Text4	Gets or sets the fourth additional text string for the tile.
Text5	Gets or sets the fifth additional text string for the tile.
Text6	Gets or sets the sixth additional text string for the tile.
Text7	Gets or sets the seventh additional text string for the tile.
Text8	Gets or sets the eighth additional text string for the tile.
Text9	Gets or sets the ninth additional text string for the tile.
ToolTipText	Gets or sets the tooltip text for the tile.
VerticalSize	Gets or sets the height of the tile, in cells.
Width	Gets the width of the tile, in pixels.

The TileControl Tiles can include the following elements:

Images The images are represented by the class, ImageElement.

Panels The panels are represent by the class, PanelElement.

Text The text is represent the by class, TextElement.

Image Element

Tiles can display one or several images. An image can be specified in the Tile using its Image, ImageKey, or Symbol properties. Also, it can be specified as a part of the template's ImageElement or stored in one of the CommonImage objects that belong to C1TileControl.

There are a few tricks when working with images. For example, you can create a big image that consists of N images in width and M images in height. If so, the ImageColumns property must be set to N, and ImageRows property must be set to N. Also, the ColumnIndex and RowIndex properties can be used to pick a small image from large matrix.

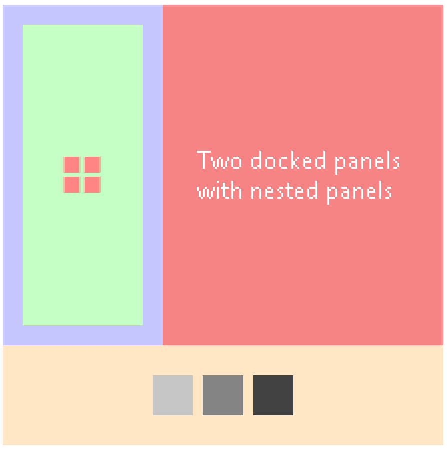
The ImageList collection should be specified first in order to use the ImageIndex property.

For more information see Adding Image Elements to a Tile.

Panel Element

Tiles can display one or more panels. A panel can have nested panels. Panel elements of the type PanelElement class may contain child elements including nested panels. The child elements can be added at design time through the **PanelElement.Children Collection** editor or programmatically through the Children property.

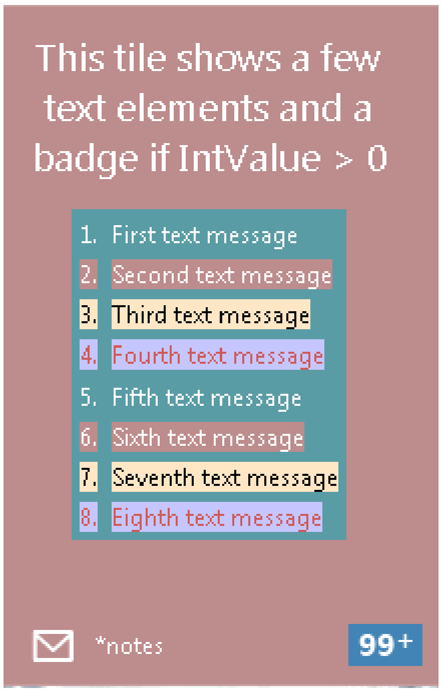
The following image displays a Tile with two docked panels with nested panels.



Text Element

Tiles can display one or more text elements. The text elements are represented by the TextElement class. The child elements can be added at design time through the designer or programmatically through the Text, Text1 through Text6 properties.

The following image displays a Tile with a few text element and a badge if IntValue is greater than zero.


This tile shows a few text elements and a badge if IntValue > 0

1. First text message
2. Second text message
3. Third text message
4. Fourth text message
5. Fifth text message
6. Sixth text message
7. Seventh text message
8. Eighth text message

*notes

99+

Data Binding Overview

The C1TileControl can universally bind to any generic .NET data source. Requiring little or no code at all, the C1TileControl can create a fully-navigational database browser in mere seconds.

TileControl for WinForms fully supports complex data binding to ADO.NET objects such as DataTable, DataView and DataSet objects.

To associate the **C1TileControl** with an **ADO.NET** or **DataObjects for WinForms** data source, set the C1TileControl.DataSource property of the TileControl to a DataSet on the same form. If the DataSet contains multiple tables, you select a table name in the C1TileControl.DataMember property combo box.

The C1TileControl.DataSource and C1TileControl.DataMember properties can be set both through code, and through Visual Studio's Properties window. This is all that is required to make **TileControl for WinForms** fully aware of the database or DataTable in your application.

Once such a link exists, **TileControl for WinForms** and the DataSet automatically notify and respond to operations performed by the other, simplifying your application development.

The Tile properties of the C1TileControl can be mapped using the **C1TileControl.PropertyMappings Collection Editor** at design time or programmatically though the PropertyMapping class.

For an example of how to use the Property Mappings see, **Step 5 of 7: Setting the TileControl's Property Mappings**.

For conceptual information about the Property Mappings see, **Property Mapping**.

Note: For a complete sample on binding TileControl to a data source , see the **Databound** sample installed with **Studio for Winforms**.

Property Mapping

The **C1.Win.C1Tile.Tile** component uses the PropertyMapping class to create an association between the property of an item from the data source and the property of the corresponding **C1.Win.C1Tile.Tile** component.

The PropertyMapping class includes the following members:

Property	Description
PropertyMapping.Comment	Specifies the comment for the PropertyMapping object.
PropertyMapping.DataField	The field of the data source which is mapped to the tile property.
PropertyMapping.DataSource	Specifies the gap between tile cells in a group.
PropertyMapping.Lookup	Encapsulates the lookup properties.
PropertyMapping.TileControl	Gets the owner of the C1.Win.C1Tile.C1TileControl.
PropertyMapping.TileProperty	Gets or sets the property of the C1.Win.C1Tile.Tile object that is a destination for mapping.

The PropertyMapping.TileProperty can be set to any of the TileProperty enumeration values.

C1TileControl uses the MappingLookup class to display information from one table based on the value of a foreign-key field in another table.

For example, consider a table of **Products** in a sales database. Each record in the **Products** table includes a **CategoryID** indicating which category the product belongs to. The **CategoryID** is a foreign key pointing to a category record in the Categories table. When presenting a list of Products (from the Products table) you may want to display the actual categories name for each product, as opposed to the **ProductsID**. Since the categories name is in the categories table, and you are presenting data from the Products table, you need to create a lookup table or MappingLookup class which takes the **CategoryID** value in the Products record, and uses that value to navigate the relationship and return the more readable, category name. This concept is known as a lookup table. For an example of creating a lookup table see **Adding Property Mappings to C1TileControl**.

The MappingLookup class includes the following members:

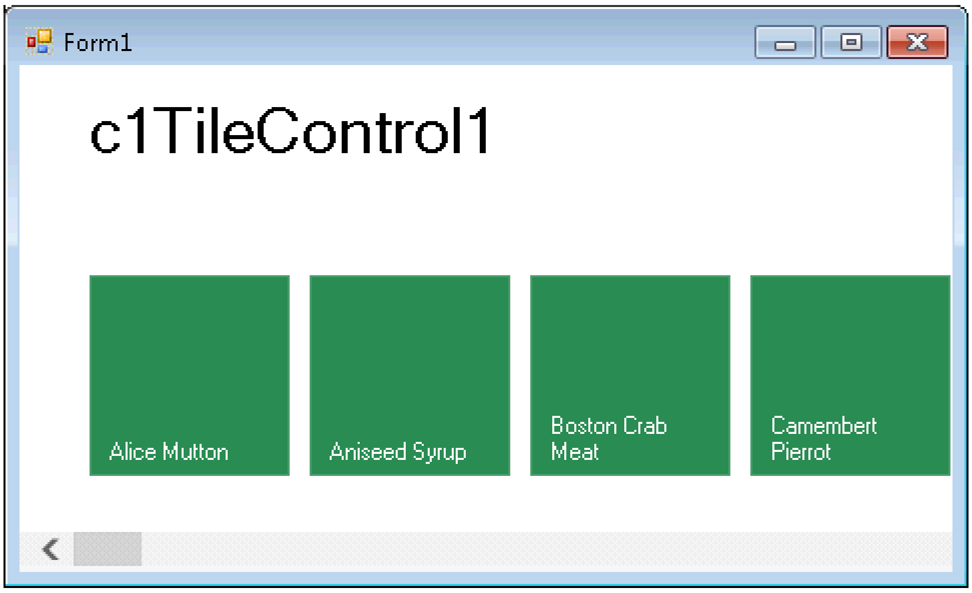
Property	Description
MappingLookup.DataSource	Gets or sets the data source object for the lookup.
MappingLookup.DisplayMember	The field of the data source which is mapped to the tile property.
MappingLookup.ValueMember	Gets or sets the field to use as the source of value (primary key) matching the foreign key in the C1.Win.C1Tile.PropertyMapping.DataField.

The PropertyMapping.TileProperty can be set to any of the TileProperty enumeration values.

Databinding Tutorial - Creating a Simple Databound TileControl

This section provides step-by-step instructions for binding a tile control to a database. You will bind **C1TileControl** to a collection of products data items to create the products name on each tile.

When you run the project, you should see a **C1TileControl** that looks similar to the screen shot below.



Step 1 of 4: Connecting to a New Data Source

In this step you will create a data source that you can later bind the **C1TileControl** to using the TileControl's databinding properties.

1. Add the **C1TileControl** to your form.
2. In the project toolbar, from the **Data** menu select **Add New Data Source**. **The Data Source Configuration Wizard** dialog box appears.
3. Select **Database** and then click **Next**.
4. Select **Dataset** and then click **Next**.
5. Click **New Connection**.
6. In the **Add Connection** dialog box, click **Browse**.
7. In the **Select Microsoft Access Database File** dialog box locate the **C1NWind.mdb** (located by default in the **Documents\ComponentOne Samples\Common**), click **Open**, and click **OK**.
8. Click the **Next** button to continue. A dialog box will appear asking if you would like to add the data file to your project and modify the connection string. Since it is not necessary to copy the database to your project, click **No**.
9. Verify the **Yes, save the connection as** check box is checked and click **Next** to continue.
10. The connection string is saved as **C1NWindConnectionString**.
11. In the **Data Source Configuration Wizard** select the **Tables** node.
12. Select **Finish**.

Step 2 of 4: Configuring a Data Connection and Data Adapter

In this step, you will add one data adapter for the table: products.

1. From the Toolbox, double-click the OleDbDataAdapter component. If it's not in the toolbox, right-click and select **Choose Items**. On the .NET Framework Components tab in the dialog box, select **OleDbDataAdapter**. The OleDbDataAdapter appears in the component tray and the **Data Adapter Configuration Wizard** appears.
2. In the Data Adapter Configuration Wizard, choose the connection you wish to use for the data adapter from the drop-down listbox (in this case, **C:\Users\UserName\Documents\ComponentOne Samples\Common\C1NWind.mdb**) and then click **Next**.
3. The Use SQL Statements is selected by default, click **Next**.
4. Copy and paste the following SQL statement in the textbox of the **Data Adapter Configuration Wizard**:

   ```
   SELECT      Products.*
   FROM          Products
   ```

5. Click **Next**.
6. Click **Finish**.

Step 3 of 4: Binding the Products Table to C1TileControl

In this step you will bind the products data table to the TileControl and fill the dataset.

1. Select the C1TileControl and expand the Data node in the C1TileControl's property window.
2. Click on the dropdown arrow next to the DataSource property and expand Other Data Sources| Form1 List Instances and select c1NwindDataSet.
3. Set the DataMember to Products.

   Setting these two properties binds the products data table in the **c1NwindDataSet** to the **TileControl**.

   The TileControl is bound to the c1NwindDataSet, but is not automatically filled in. To fill the dataset, double click on the form and call the data adapter method in the Form1_Load like the following:

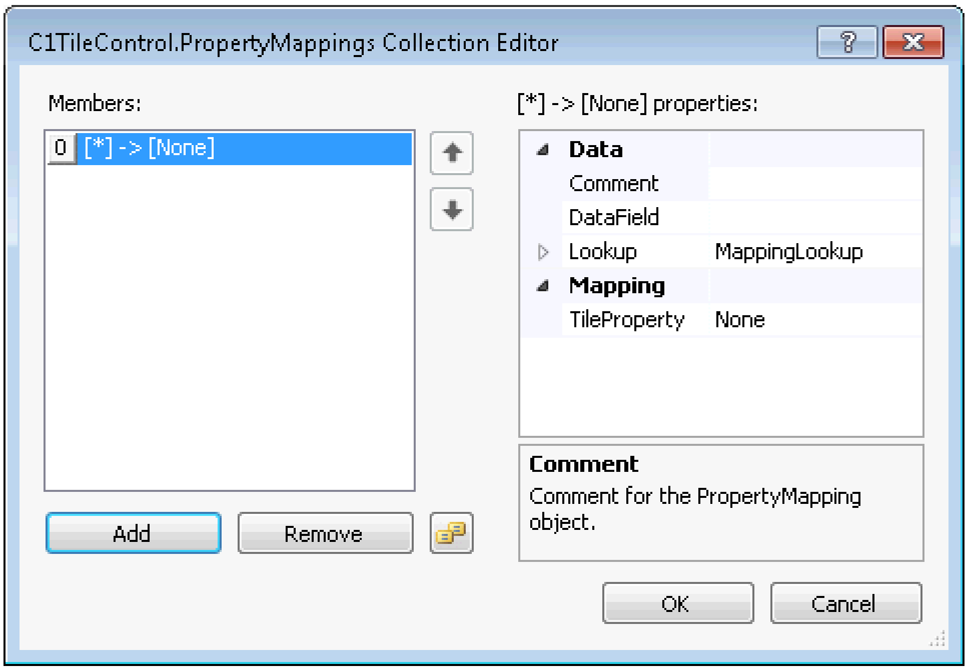
Visual Basic
oleDataAdapter1.Fill(c1NWindDataSet)
C#
oleDbDataAdapter1.Fill(c1NWindDataSet);

When you run the project you will notice there is a tile for each product name in the product table, but the product name does not appear on each tile. In this case we will need to map the **ProductName** and **Text** to the **C1TileControl**.

Step 4 of 4: Setting TileControl's Property Mappings

In this step you will create a PropertyMapping member to bind the ProductName and the ProductName's text to the C1TileControl.

1. Select the **C1TileControl** and expand the **Data** node in the C1TileControl's property window.
2. In the C1TileControl's Properties window click on the ellipsis button next to the **PropertyMappings** property. The PropertyMapping Collection Editor appears.
3. Click **Add**. A **C1.Win.C1Tile.PropertyMapping** item will appear like the following.


C1TileControl.PropertyMappings Collection Editor
Members:
0 [*] -> [None]
[*] -> [None] properties:
Add
Remove
OK
Cancel

Data	
Comment	
DataField	
Lookup	MappingLookup
Mapping	
TileProperty	None

Comment
Comment for the PropertyMapping object.

4. Click Add and set the [0] [*] -> [None] properties to the following:
   - **PropertyMapping.DataField** to **ProductName**

   This will bind the product name to each tile.

   - **PropertyMapping.TileProperty** to **Text**

   This will bind the **ProductName** to the **Text** property of the **C1TileControl**.

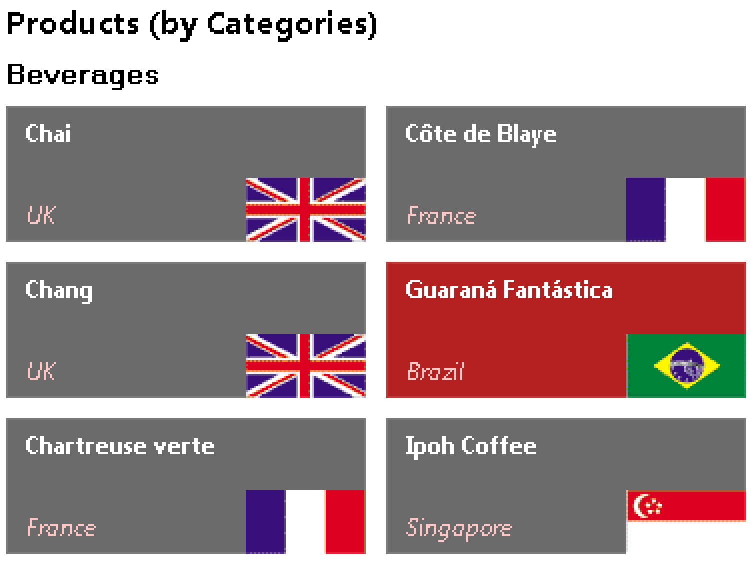
Run the project and observe

Press **F4** to run the project and notice that the ProductName's **Text** is bound to each tile in the **C1TileControl**.

Databinding Tutorial - Creating a Complex Databound TileControl

This section provides step-by-step instructions for binding a tile control to a database. You will bind C1TileControl to a collection of category data items to create a tile for each category. Each group of tiles will display the category's name in its header.

When you run the project, you should see a C1TileControl that looks similar to the screen shot below.



Step 1 of 8: Creating a Data Source for C1TileControl

In this step you will create a data source that you can later bind the C1TileControl to using the TileControl's databinding properties.

Connect to a new data source

1. Add the **C1TileControl** to your form.
2. In the project toolbar, from the **Data** menu select **Add New Data Source**. **The Data Source Configuration Wizard** dialog box appears.
3. Select **Database** and then click **Next**.
4. Select **Dataset** and then click **Next**.
5. Click **New Connection**.
6. In the **Add Connection** dialog box, click **Browse**.
7. In the Select Microsoft Access Database File dialog box locate the C1NWind.mdb (located by default in the **C:\Users\UserName\Documents\ComponentOne Samples\Common)**, click **Open**, and click **OK**.
8. Click the **Next** button to continue. A dialog box will appear aslking if you would like to add the data file to your project and modify the connection string. Since it is not necessary to copy the database to your project, click **No**.
9. Verify the **Yes**, save the connection as check box is checked and click **Next** to continue.
10. The connection string is saved as **C1NWindConnectionString**.
11. In the **Data Source Configuration Wizard** select the **Tables** node.
12. dSelect **Finish**.

Step 2 of 8: Configuring a Data Connection and Data Adapter

In this step you will add three separate data adapters for each table: products, categories, and suppliers. Since the dataset we use contains separate data adapters for each table we must call each adapter's **Fill** method separately.

1. From the Toolbox, double-click the **OleDbDataAdapter** component. If it's not in the toolbox, right-click and select **Choose Items**. On the .NET Framework Components tab in the dialog box, select **OleDbDataAdapter**. The OleDbDataAdapter appears in the component tray and the **Data Adapter Configuration Wizard** appears.
2. In the Data Adapter Configuration Wizard, choose the connection you wish to use for the data adapter from the drop-down listbox (in this case, **C:\Users\UserName\Documents\ComponentOne Samples\Common\C1Nwind.mdb**) and then click **Next**.
3. The **Use SQL Statements** is selected by default, click **Next**.
4. Copy and paste the following SQL statement in the textbox of the **Data Adapter Configuration Wizard**:

   ```
   SELECT      Products.*

   FROM          Products
   ```

5. Click **Next** and then click **Yes** to add primary key columns to your query.
6. Click **Finish.**
7. Set the **OldDataAdaper1 Name** property to **productsDataAdapter** and **OleDbConnection1 Name** property to **productsConnection**.
8. Add another **OleDbDataAdapter**.
9. In the **Data Adapter Configuration Wizard**, choose the connection you wish to use for the data adapter from the drop-down listbox (in this case **C:\Users\UserName\Documents\ComponentOne Samples\Common\C1Nwind.mdb)** and then click **Next**.
10. The **Use SQL Statements** is selected by default, click **Next**.
11. Copy and paste the following SQL statement in the textbox of the **Data Adapter Configuration Wizard**:

    ```
    SELECT      Categories.*

    FROM          Categories
    ```

12. Click **Next** and then click **Yes** to add primary key columns to your query.
13. Click **Finish**.
14. Set the **OldDataAdaper2 Name** property to **categoriesDataAdapter** and **OleDbConnection2 Name** property to **categoriesConnection**.
15. Add another **OleDbDataAdapter**.
16. In the **Data Adapter Configuration Wizard**, choose the connection you wish to use for the data adapter from the drop-down listbox (in this case **C:\Users\UserName\Documents\ComponentOne Samples\Common\C1Nwind.mdb**) and then click **Next**.
17. The **Use SQL Statements** is selected by default, click **Next**.
18. Copy and paste the following SQL statement in the textbox of the **Data Adapter Configuration Wizard**:

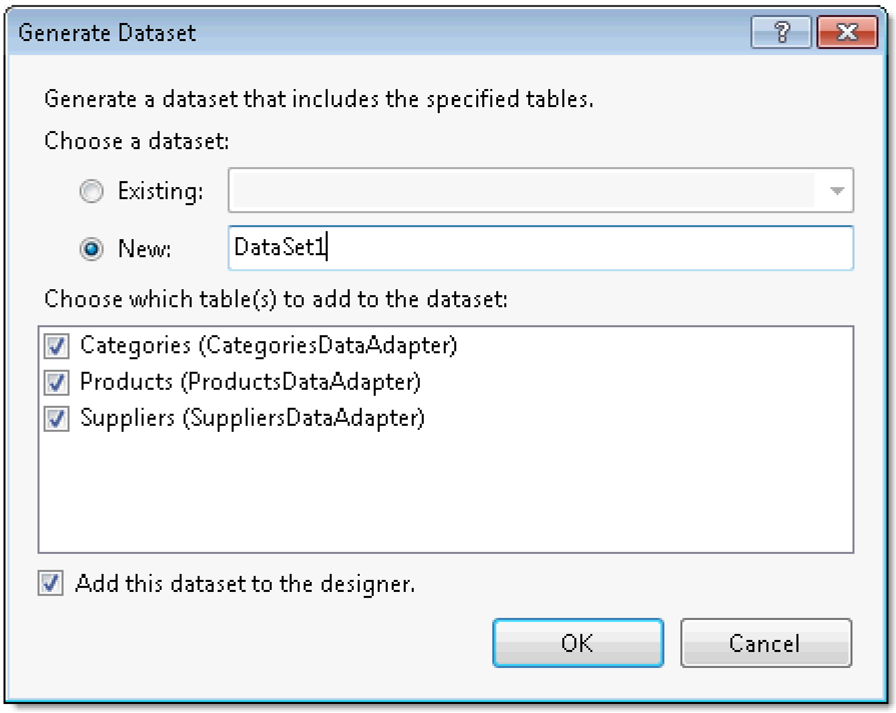
    ```
    SELECT      Suppliers.*

    FROM          Suppliers
    ```

19. Click **Next** and then click **Yes** to add primary key columns to your query.
20. Click **Finish**.
21. Set the **OldDataAdaper3 Name** property to **SuppliersDataAdapter** and **OleDbConnection3 Name** property to **SuppliersConnection**.

Step 3 of 8: Generating the Dataset

In this step you will create a dataset based on the queries you specified for the data adapters. The dataset is an instance of the DataSet class based on the corresponding XML Schema (.xsd) file that describes the class's elements (table, columns, and constraints).

1. From the **Data** menu, choose **Generate DataSet**. The **Generate DataSet** dialog box appears.


Generate Dataset

Generate a dataset that includes the specified tables.

Choose a dataset:

Existing:

New: DataSet1

Choose which table(s) to add to the dataset:

- [x] Categories (CategoriesDataAdapter)
- [x] Products (ProductsDataAdapter)
- [x] Suppliers (SuppliersDataAdapter)

- [x] Add this dataset to the designer.

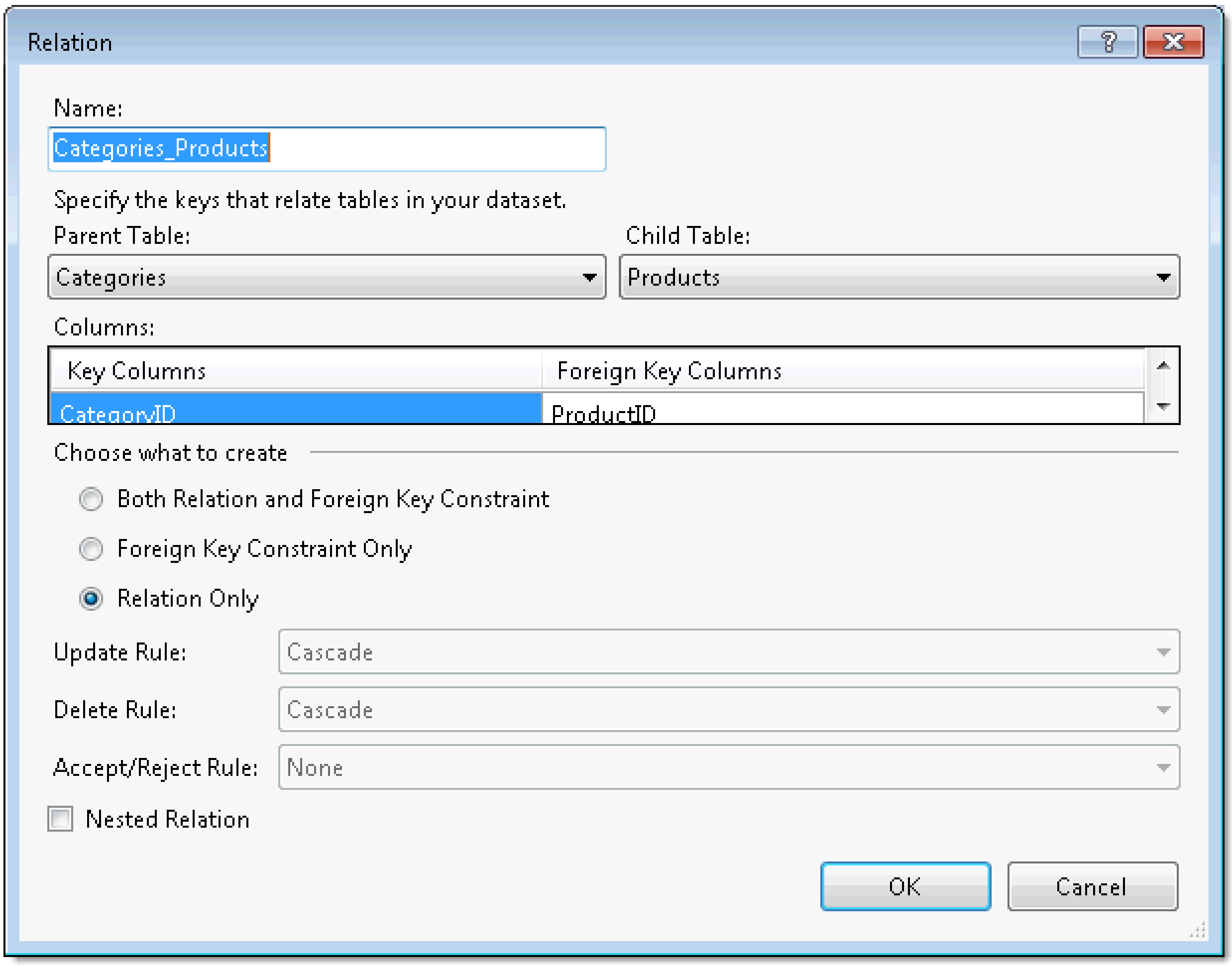
2. Select the New option to name the dataset Dataset1.
3. In the list under Choose which table(s) to add to the dataset, the **Categories**, **Products**, and **Suppliers** table should be selected.
4. Make sure the **Add this dataset to the designer** is selected and then click **OK**.

   Visual Studio adds an instance of the new dataset class **DataSet1** to the form.

Step 4 of 8: Preparing the Data Schema

In this step you will create a **Categories_Product** and a **Suppliers_Product** relation for the **DataSet1**.

1. Take the **CategoryID** field with the mouse in the Categories table and drag&drop it to the **CategoryID** field in the Products table. You'll see the following window that creates a new relation between two tables:


Relation

Name: Categories_Products

Specify the keys that relate tables in your dataset.

Parent Table: Categories

Child Table: Products

Columns:

Key Columns	Foreign Key Columns
CategoryID	ProductID

Choose what to create

- ( ) Both Relation and Foreign Key Constraint
- ( ) Foreign Key Constraint Only
- (●) Relation Only

Update Rule: Cascade

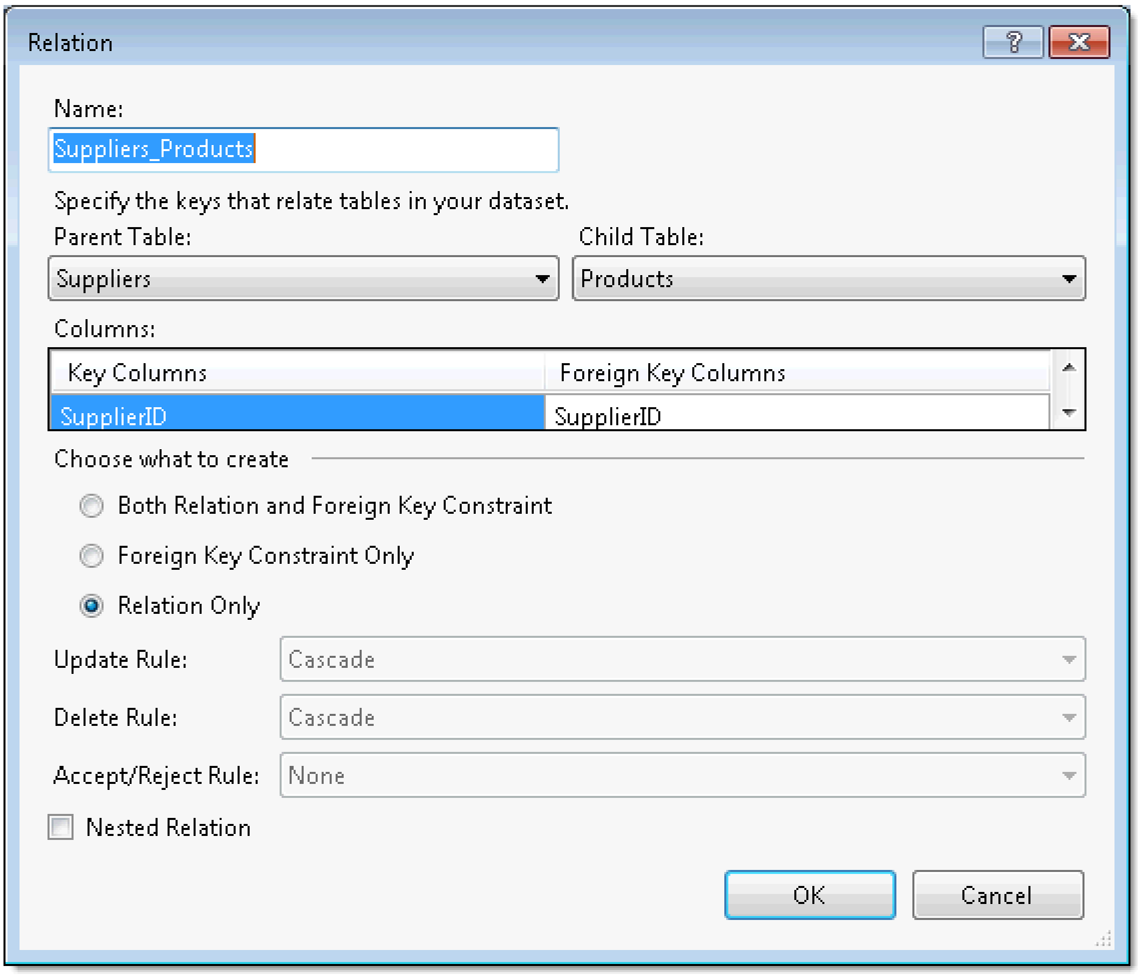
Delete Rule: Cascade

Accept/Reject Rule: None

☐ Nested Relation

2. Take the **SupplierID** field with the mouse in the Suppliers table and drag&drop it to the **SupplierID** field in the Products table.

   You'll see the following window that creates a new relation between two tables:


Relation

Name: Suppliers_Products

Specify the keys that relate tables in your dataset.

Parent Table: Suppliers

Child Table: Products

Columns:

Key Columns	Foreign Key Columns
SupplierID	SupplierID

Choose what to create

( ) Both Relation and Foreign Key Constraint

( ) Foreign Key Constraint Only

[✓ Relation Only]

Update Rule: Cascade

Delete Rule: Cascade

Accept/Reject Rule: None

Nested Relation

Step 5 of 8: Binding the Products Table to C1TileControl

In this step you will bind the products data table to the TileControl and fill in the datasets for the products, suppliers, and categories.

Set the TileControl's Data Properties

1. Select the **C1TileControl** and expand the **Data** node in the C1TileControl's property window.
2. Set the **C1TileControl.DataSource** property to **dataSet11**. You will need to expand the **Other Data Source** node.
3. Set the **DataMember** to **Products**.

Setting these two properties binds the products data table in the **dataSet11** to the TileControl.

Populating the TileControl

The TileControl is bound to the **dataSet11**, but is not automatically filled in. To fill the dataset, call the data adapter method like the following:

Visual Basic
categoriesDataAdapter.Fill(dataSet11)
suppliersDataAdapter.Fill(dataSet11)
productsDataAdapter.Fill(dataSet11)

C#

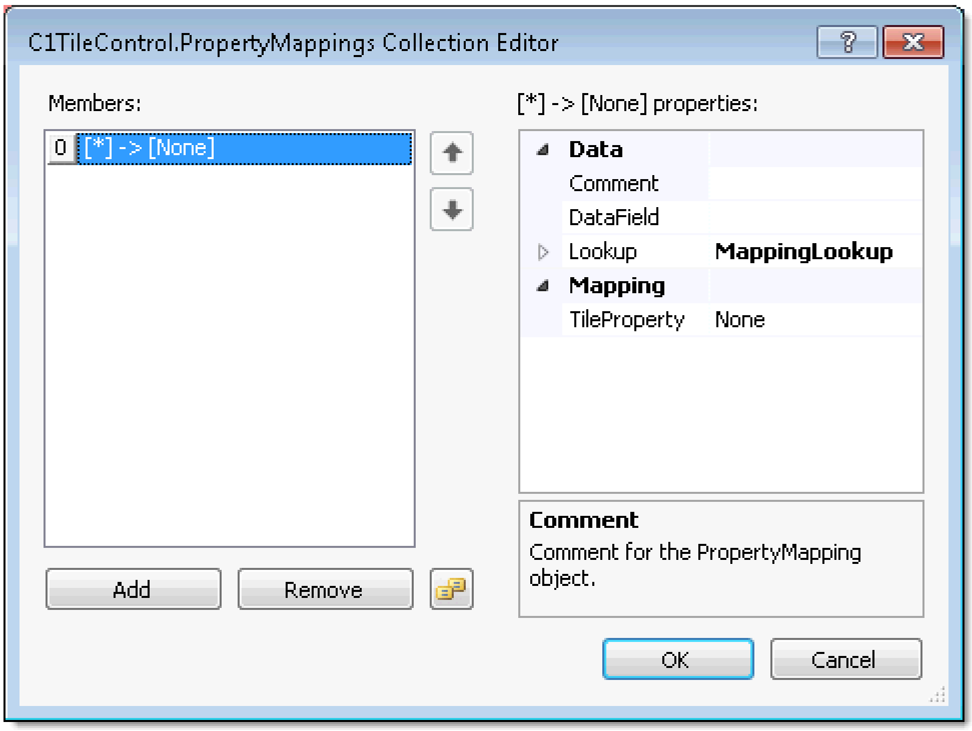
```
categoriesDataAdapter.Fill(dataSet11);
suppliersDataAdapter.Fill(dataSet11);
productsDataAdapter.Fill(dataSet11);
```

Step 6 of 8: Setting C1TileControl's Property Mappings

In this step you will add the property mappings to the PropertyMapping class using the **C1TileControl.PropertyMappings** collection editor at design time.

Select the C1TileControl on the form. In the C1TileControl's Properties window click on the ellipsis button next to the **PropertyMappings** property. The **C1TileControl.PropertyMappings Collection Editor** appears.

1. Click Add. A C1.Win.C1Tile.PropertyMapping item will appear like the following.


C1TileControl.PropertyMappings Collection Editor
Members:
0 [*] -> [None]
Add
Remove
[*] -> [None] properties:
OK
Cancel

Data	
Comment	
DataField	
Lookup	**MappingLookup**
Mapping	
TileProperty	None

Comment
Comment for the PropertyMapping object.

3. Set the [*]->[None] member PropertyMapping.DataField to ProductID and PropertyMapping.TileProperty to Tag.
4. Click Add and set the [*]->[None] [1] properties to the following:

   - PropertyMapping.DataField to CategoryID

- PropertyMapping.Lookup.DataSource to dataSet11
- PropertyMapping.Lookup.DisplayMember to Categories.CategoryName
- PropertyMapping.Lookup.ValueMember to Categories.CategoryID
- The CategoryID is a foreign key pointing to a category record in the Categories table. Here we create a lookup table which takes the CategoryID value in the Products record, and uses that value to navigate the relationship and return the more readable, category name.
- PropertyMapping.TileProperty to Group

This will display the categories name for each group of tiles. For example, Beverages, Condiments, Dairy Products, etc.

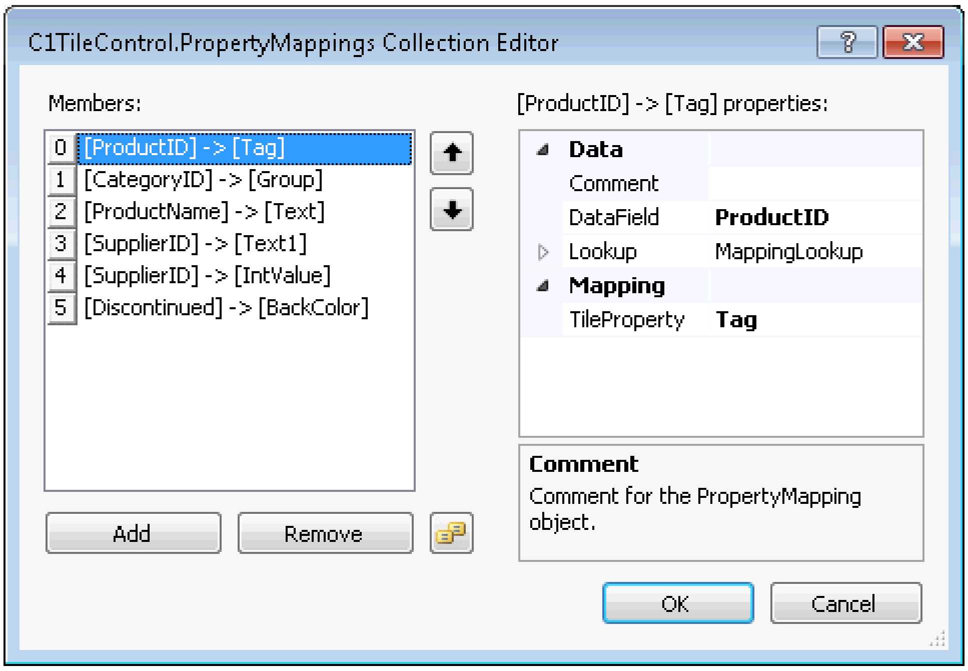
5. Click Add and set the [*]->[None] [2] properties to the following:
6. PropertyMapping.DataField to ProductName
7. PropertyMapping.TileProperty to Text
8. This will display the product's name on each tile. For example, Chai, Chang, etc.
9. Click Add and set the [*]->[None] [3] properties to the following:
   - PropertyMapping.DataField to SupplierID
   - PropertyMapping.Lookup.DataSource to dataSet11
   - PropertyMapping.Lookup.DisplayMember to Suppliers.Country
   - PropertyMapping.Lookup.ValueMember to Suppliers.SupplierID
   - PropertyMapping.TileProperty to Text1
10. This will assign the country name to each tile.
11. Click Add and set the [*]->[None] [4] properties to the following:

    - PropertyMapping.DataField
    - PropertyMapping.Lookup.DataSource to dataSet11
    - PropertyMapping.Lookup.DataMember to Suppliers.Country
    - PropertyMapping.Lookup.ValueMember to Suppliers.SupplierID
    - PropertyMapping.TileProperty to IntValue

12. Click Add and set the [*]->[None] [5] properties to the following:
13. PropertyMapping.DataField to Discontinue.

    - PropertyMapping.TileProperty to BackColor.

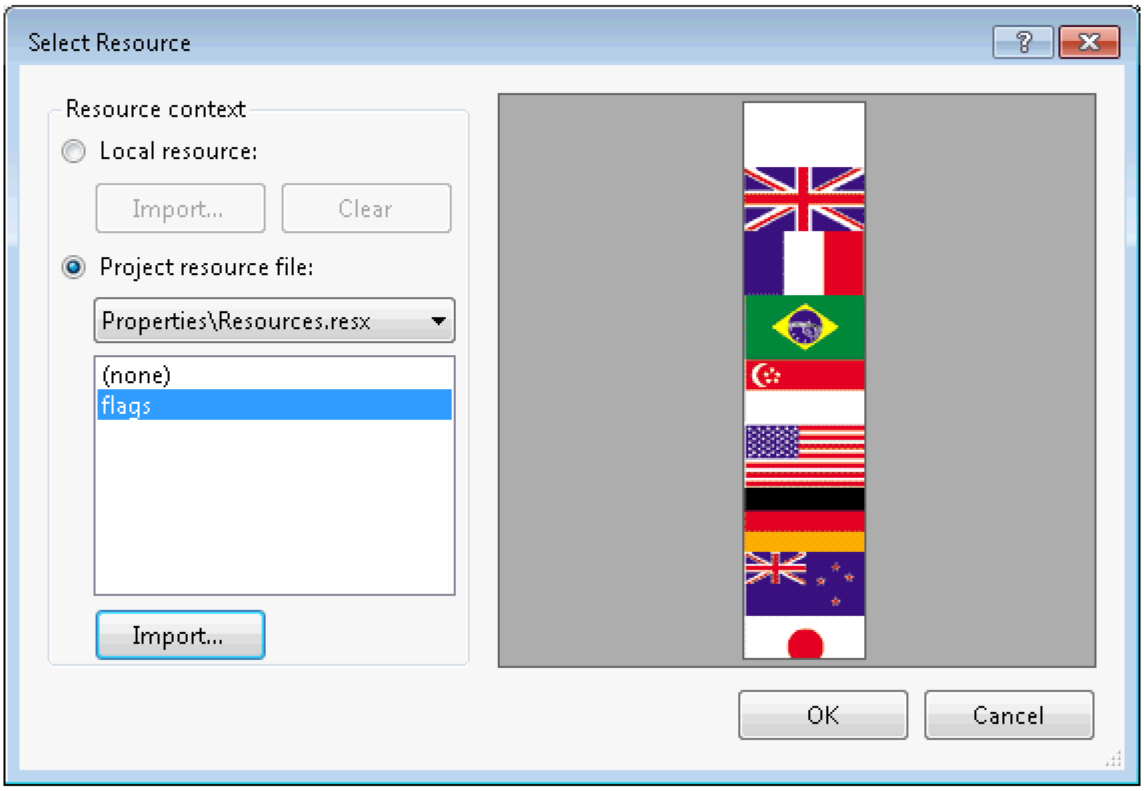
The property mappings appear like the following in the members list:


C1TileControl.PropertyMappings Collection Editor
Members:
0 [ProductID] -> [Tag]
1 [CategoryID] -> [Group]
2 [ProductName] -> [Text]
3 [SupplierID] -> [Text1]
4 [SupplierID] -> [IntValue]
5 [Discontinued] -> [BackColor]
[ProductID] -> [Tag] properties:
Data
Comment
DataField ProductID
Lookup MappingLookup
Mapping
TileProperty Tag
Comment
Comment for the PropertyMapping object.
Add
Remove
OK
Cancel

Step 7 of 8: Modifying C1TileControl's Default Template

In this step you will modify the default template for the C1TileControl.

1. In the Solution Explorer right-click on the project name and select Add New Folder.
2. Copy the flags.png from the Databound sample project located in **C:\Users\username\Documents\ComponentOne Samples\WinForms Edition\C1TileControl\Samples\Resources** to your local project.
3. Rename the folder **Resources** and add the flags.png image from the folder.
4. In the Solution Explorer click on the **Show All Files** button.
5. Right-click on the **Resources** folder and select the image and select **Include In Project**.
6. In **C1TileControl's Properties Window** expand the **DefaultTemplate** node and click on the ellipsis button next to the **Templates** property.
7. Click on the ellipsis button next to the **Elements** property. The **Template.Elements Collection Editor** appears.
8. Click on the ellipsis button next to the **Children** property.
9. Remove the **ImageElement** from the **Members** list.
10. Select the **TextElement** from the Members list and set its **FontBold** to **True**.
11. Click **OK** to save and close the **PanelElement.Children Collection Editor**.
12. Click on the dropdown arrow next to the **Add** item and select **TextElement**.
13. Select the **TextElement** and set the **ForeColorSelector** to **Unbound**.
14. Set the **ForeColor** to **255, 192, 192**.
15. Set the **TextSelector** to **Text1** and **Alignment** to **BottomLeft**.
16. This will set the text for each country name to appear bottom left of each Tile.
17. In the **Template.Elements Collection Editor** click on the dropdown arrow next to the **Add** button and select **ImageElement**.
18. Click on the ellipsis button next to Image. The Select Resource dialog box appears.
19. Click on the Import button and browse to the Resources folder in your project and select the flags.png.



20. Click OK to save and close the Select Resource dialog box.
21. In the Template.Elements Collection Editor set the ImageSelector to Unbound.

22. Set the ImageRows to 17 and Alignment to BottomRight.
23. Click OK to save and close the Template.Elements Collection Editor.

Step 8 of 8: Formatting the Tiles in the FormatValue Event

In this step the C1TileControl.FormatValue event is used to format the tile's backcolor, tile's images, and the tile's country name.

1. In the C1TileControl's events property window double click on FormatValue item to create an event handler.
2. In code view, add the following code to the C1TileControl1_FormatValue event handler:

To write code in Visual Basic

Visual Basic

```
Private Sub c1TileControl1_FormatValue(sender As Object, e As FormatValueEventArgs)
    If e.TileProperty = TileProperty.BackColor Then
          If TypeOf e.Value Is Boolean AndAlso CBool(e.Value) Then
                e.Value = Color.Firebrick
          Else
                e.Value = Color.DimGray
          End If
    ElseIf e.TileProperty = TileProperty.IntValue Then
          Dim result As Integer = 0
          Select Case TryCast(e.Value, String)
                Case "UK"
                      result = 1
                      Exit Select
                Case "France"
                      result = 2
                      Exit Select
                Case "Brazil"
                      result = 3
                      Exit Select
                Case "Singapore"
                      result = 4
                      Exit Select
                Case "USA"
                      result = 5
                      Exit Select
                Case "Germany"
                      result = 6
                      Exit Select
                Case "Australia"
                      result = 7
                      Exit Select
                Case "Japan"
                      result = 8
                      Exit Select
                Case "Canada"
                      result = 9
                      Exit Select
                Case "Netherlands"
                      result = 10
                      Exit Select
                Case "Finland"
                      result = 11
                      Exit Select
                Case "Norway"
                      result = 12
                      Exit Select
                Case "Italy"
                      result = 13
                      Exit Select
                Case "Spain"
                      result = 14
                      Exit Select
                Case "Sweden"
                      result = 15
                      Exit Select
                Case "Denmark"
                      result = 16
                      Exit Select
          End Select
          e.Value = result
    End If
End Sub
```

To write code in C#

C#

```
{
  if (e.TileProperty == TileProperty.BackColor)
    {
          if (e.Value is bool && (bool)e.Value)
              e.Value = Color.Firebrick;
           else
              e.Value = Color.DimGray;
      }
      else if (e.TileProperty == TileProperty.IntValue)
      {
          int result = 0;
          switch (e.Value as string)
          {
             case "UK":
                  result = 1; break;
             case "France":
                  result = 2; break;
             case "Brazil":
                  result = 3; break;
             case "Singapore":
                  result = 4; break;
             case "USA":
                 result = 5; break;
            case "Germany":
                result = 6; break;
            case "Australia":
                result = 7; break;
            case "Japan":
                result = 8; break;
            case "Canada":
                result = 9; break;
            case "Netherlands":
                result = 10; break;
            case "Finland":
                result = 11; break;
            case "Norway":
                result = 12; break;
            case "Italy":
                result = 13; break;
            case "Spain":
                result = 14; break;
            case "Sweden":
                result = 15; break;
            case "Denmark":
                result = 16; break;
        }
        e.Value = result;
    }
}
```

TileControl for WinForms Samples

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

Please refer to the pre-installed product samples through the following path:

Documents\ComponentOne Samples\WinForms

The following table provides a short overview of each sample.

Sample	Description
Databound	Shows an example of binding C1TileControl to a data source. A tile element is generated for each row in the associated DataTable (except those skipped in the FilterItem event). You can map data columns to the properties of the autogenerated Tile components. If some columns contain cryptic foreign key values (such as Product.SupplierID), use the Lookup property of the PropertyMapping object to replace those values with meaningful strings (such as Supplier.Country) from another table or data source.
TileImages	The sample shows how to display images on tiles using various techniques.
TileLayout	The sample shows various methods of laying out the template elements.

TileControl for WinForms Task-Based Help

The task-based help section assumes that you are familiar with programming in the Visual Studio environment and have a general understanding of the **TileControl**.

Each topic provides a solution for specific tasks using the C1TileControl. By following the steps outlined in each topic, you will be able to create projects using a variety of **C1TileControl** features.

Adding Templates to the C1TileControl

Templates can be created at design time or programmatically. At design time they can be created using the **C1TileControl.Templates Collection Editor**. Templates can also be created programmatically through the TemplateCollection class. The template can be inserted into the TemplateCollection at the specified index using the InsertItem method.

Design-Time

To add a template at design time, complete the following:

1. Select the **C1TileControl** and click on its smart tag to open the **C1TileControl Tasks** menu.
2. Select **Edit Tiles**.

   The **C1TileControl.Templates Collection Editor** editor appears.

3. Click **Add** to add a template to the Members list.

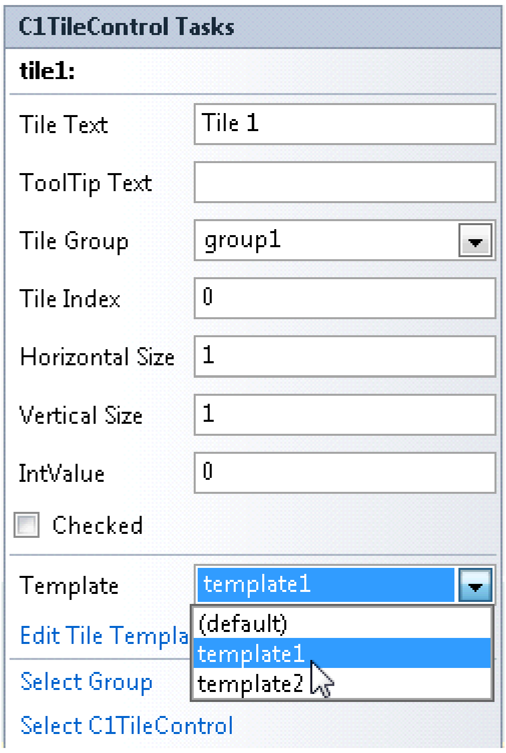
Assigning a Template to a Specified Tile

Once a template is created at design time through the **C1TileControl.Templates Collection Editor** or programatically through the TemplateCollection class it then can be assigned to a specified tile.

Design-Time

To assign a template to a specified tile, complete the following:

1. Select the tile you wish to add the template to. The **C1TileControl Tasks** menu appears for the selected tile.
2. Select the template from the **Template** dropdown listbox you wish to assign to the specified tile. For example, **template1** is being assigned to **tile1**.


C1TileControl Tasks

tile1:

Tile Text	Tile 1
ToolTip Text	
Tile Group	group1
Tile Index	0
Horizontal Size	1
Vertical Size	1
IntValue	0

☐ Checked

Template: template1

(default)
template1
template2

Adding Elements to a Template

Template elements can be created at design time or programmatically. At design time they can be created using the **C1TileControl.Templates Collection Editor**. Template elements can also be created programmatically through the TemplateCollection class. The template can be inserted into the TemplateCollection at the specified index using the InsertItem method.

Design Time

To add elements to the template at design time, complete the following:

1. Select the **C1TileControl** and click on its smart tag to open the **C1TileControl Tasks** menu.
2. Select **Edit Tiles**.

   The **C1TileControl.Templates Collection Editor** editor appears.

3. Click **Add** to add a template to the **Members** list.
4. Select the ellipsis button next to the **Elements** property. The **Template.Elements Collection Editor** appears.
5. Click on the dropdown arrow and select the **PanelElement** twice. Two panels are added to the template.

Creating a Lookup Table for C1TileControl

To create a lookup table using the **PropertyMapping Collection Editor**, complete the following:

1. Create a new data source. For more information see **Connect to a new data source**.
2. In the Data Source Configuration Wizard expand the Tables and select Categories, Products, and Suppliers.
3. Select **Finish**.
4. Select the C1TileControl and expand the **Data** node in the C1TileControl's property window.
5. Set the **C1TileControl.DataSource** property to **c1NwindDataSet**.
6. Set the **DataMember** to **Products**.

Setting these two properties binds the products data table in the c1NwindDataSet to the TileControl.

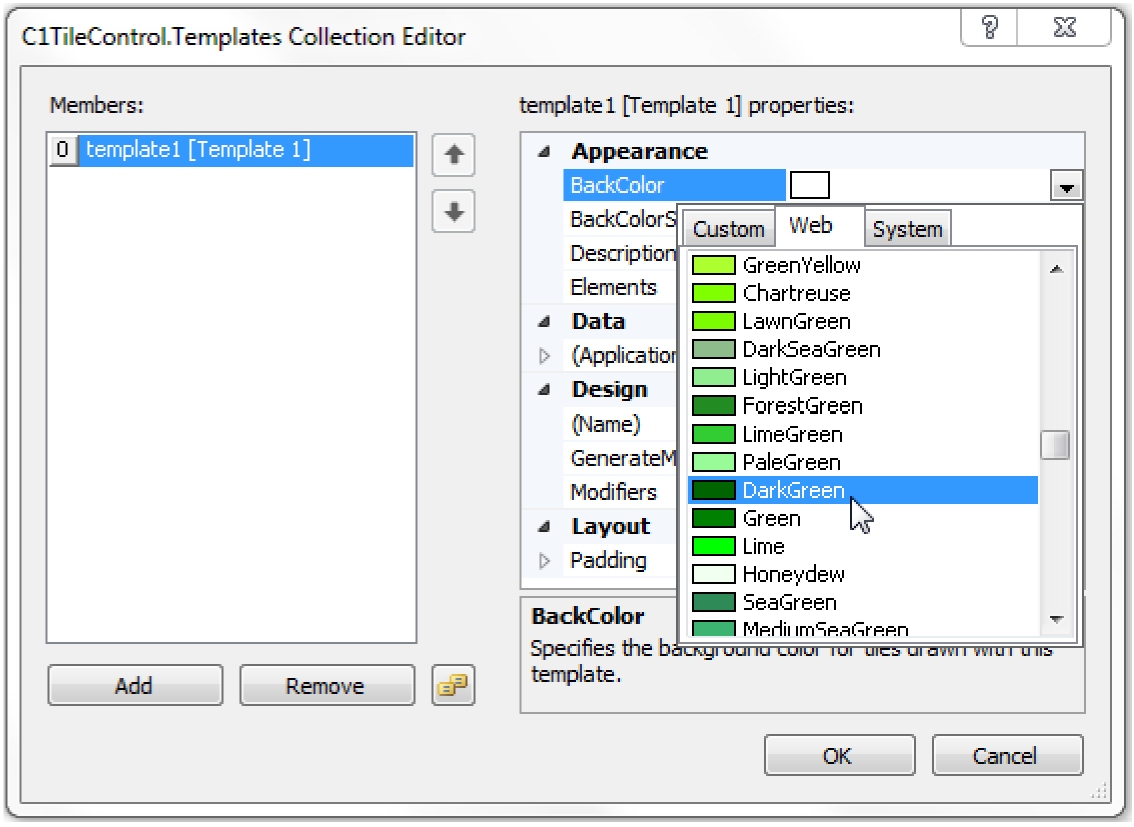
Changing the BackColor of the Template

To change the BackColor of the Template at design time, complete the following:

1. Select the **C1TileControl** and click on its smart tag to open the **C1TileControl Tasks** menu.
2. Select **Edit Tiles**.

   The **C1TileControl.Templates Collection Editor** editor appears.

3. Click **Add** to add a template to the Members list.
4. Set the **BackColor** to **DarkGreen**.



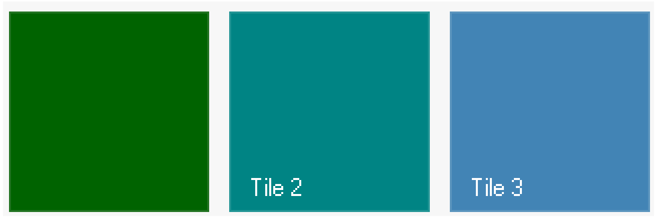
5. Set the **BackColorSelector** to **Unbound**.

   This will assign the DarkGreen backcolor to this template rather than the default backcolor. Once the Template is assigned to the specified Tile the new BackColor will appear.

6. Select the first Tile and set its Template property to Template1. The template with the new backcolor will be updated for the first Tile.

This topic illustrates the following:

The new backcolor for the template appears in the first Tile.



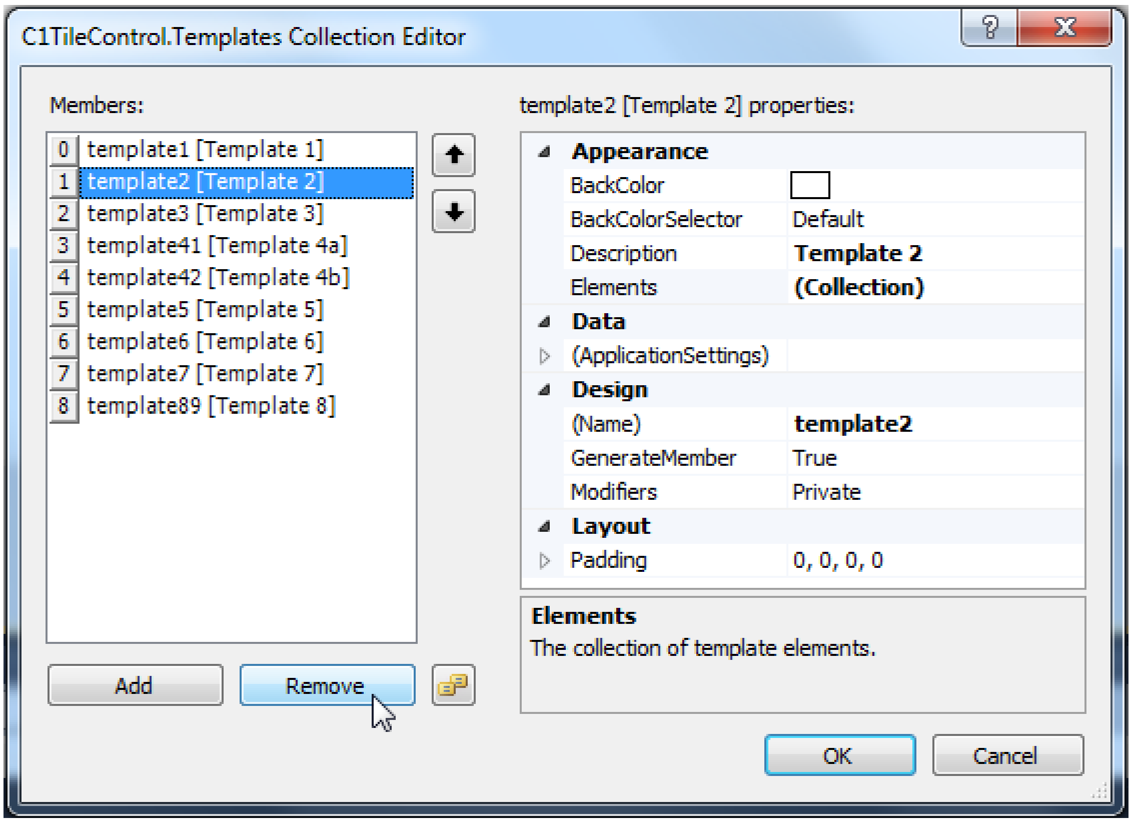
Removing Specific Templates

The TileControls template and be removed programmatically or at design time.

Design-Time

To remove a specific template from the TileControl at design time, complete the following:

1. Right-click on the C1TileControl and select **Edit Templates**. **The C1TileControl.Templates Collection Editor** appears.
2. Select the Template from the Members: list and click **Remove**.


C1TileControl.Templates Collection Editor
Members:
template2 [Template 2] properties:
Add
Remove
OK
Cancel

0 template1 [Template 1]
1 template2 [Template 2]
2 template3 [Template 3]
3 template41 [Template 4a]
4 template42 [Template 4b]
5 template5 [Template 5]
6 template6 [Template 6]
7 template7 [Template 7]
8 template89 [Template 8]

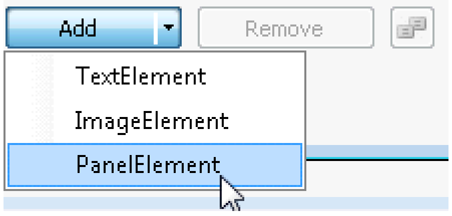
Appearance	
BackColor	⬜
BackColorSelector	Default
Description	**Template 2**
Elements	**(Collection)**
Data	
(ApplicationSettings)	
Design	
(Name)	**template2**
GenerateMember	True
Modifiers	Private
Layout	
Padding	0, 0, 0, 0

Elements
The collection of template elements.

Alternating the Text View by a Timer

To alternate the text view by a timer, complete the following:

Add the First Template

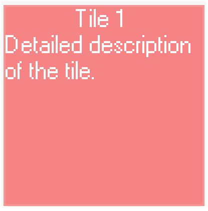
1. Right-click the Tile control and select **Edit Templates**. The **C1TileControl.Templates Collection Editor** appears.
2. Click **Add** twice to add two templates to the C1TileControl.
3. Select **template1** and click on th ellipsis button next to the **Elements collection**. The **Template.Elements** collection editor appears.
4. Select the **PanelElement** from the **Add** dropdown listbox.
   

5. Set the PanelElements properties to the following:
   - Alignment property to **TopLeft**.
   - ChildSpacing property to **0**. This will decrease the default spacing between the child elements from 5 pixels to 0 pixels.
   - Orientation property to **Vertical**.
6. Click on the ellipsis button next to the **Children** property.
7. Add two **TextElements** to the **PanelElement**.
8. Select the second text element, **[1] TextElement** and set its **TextSelector** property to **Text1**. This will assign the value of the Text1 property to this template.
9. Click **OK** to save and close the **PanelElement.Children Collection Editor** and click **OK** to save and close the **Template.Elements Collection Editor**.

Add the Second Template

10. Select **template2** in the **C1TileControl.Templates Collection Editor**.
11. Click on the Ellipsis button next to the **Elements Collection**. The **Template.Elements Collection Editor** appears.
12. Click the dropdown arrow next to the **Add** button to add a **PanelElement**.
13. Set the **[0]Panel Element** properties to the following:
    - Alignment property to **TopLeft**.
    - ChildSpacing property to **0**.
    - Orientation property to **Vertical**.
14. Click the ellipsis button next to the **Children (Collection)** property and add two **TextElements**.
15. Select the first text element, **[0] TextElement** and set its **TextSelector** property to **Text1**.
16. Select the second text element, **[1] TextElement** and set its **TextSelector** property to **Text2**.
17. Click **OK** to save and close the **PanelElement.Children Collection Editor** and click **OK** to save and close the **Template.Elements Collection Editor**.
18. Right-click on Tile1 and select **Edit Groups**. The **C1TileControl.Groups Collection Editor appears**.
19. Click on the ellipsis button next to the **Tiles Collection**.
20. Select tile1 and set its properties to the following:
    - Template property to template1. The settings for template1 are applied to Tile1.
    - Text1property to **Detailed description of the Tile**.
    - Text2 property to **More information and details of the Tile behavior.**
21. Click **OK** to save and close the **Group.Tiles Collection Editor**.

Tile1 should appear like the following:



Add a Timer to alternate the template views for Tile1.

22. Double-click on the WindowsForm **Timer** control to add it to your component tray.
23. Set the timer1 **Interval** property to **3000** and the **Enabled** property **True**.
24. Right-click on the TileControl and select **View Code**.
25. Add the following code to your project to create an animation that alternates the text views of each template:

To write code in Visual Basic

Visual Basic

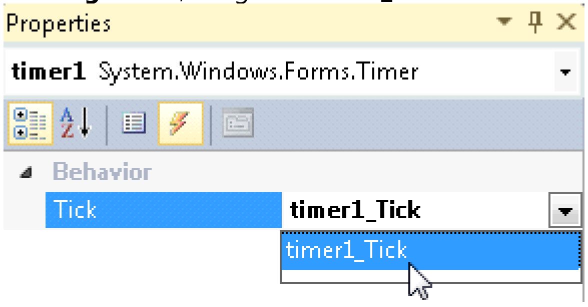
```
Public Partial Class Form1
   Inherits Form
   Private _tile1Flipped As Boolean

   Public Sub New()
          InitializeComponent()
   End Sub

   Private Sub timer1_Tick(sender As Object, e As EventArgs)
          Dim a As Boolean = _tile1Flipped
          tile1.Template = If(a, template1, template2)
          _tile1Flipped = Not a
   End Sub
End Class
```

To write code in C#

C#

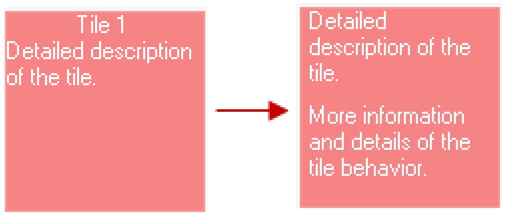
```
public partial class Form1 : Form
    {
        bool _tile1Flipped;

        public Form1()
        {
            InitializeComponent();
        }

        private void timer1_Tick(object sender, EventArgs e)
        {
            bool a = _tile1Flipped;
            tile1.Template = a ? template1 : template2;
            _tile1Flipped = !a;
        }
    }
```

26. In **Design view**, assign the **timer_Tick** event handler to **timer1**.



This topic illustrates the following:

The Tile alternates templates based opon a timer. The first template is displayed for a few seconds and then the second template for the tile appears in place of the first.


Tile 1
Detailed description of the tile.
Detailed description of the tile.
More information and details of the tile behavior.

Saving and Loading TileControl as an XML File

This topic shows how to save the **C1TileControl** as an XML file and how to load an existing **C1TileControl** from an xml file.

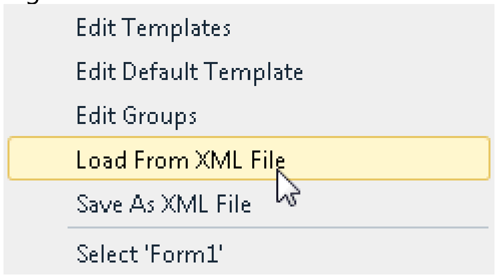
Loading TileControl From an XML File

This task shows how to load the C1TileControl as an XML File at run time and in code.

Load C1TileControl as an XML file at run time

To load the C1TileControl as an XML file at run time, complete the following:

1. Right-click the C1TileControl and select **Load From XML File** item from the context menu.

   The **Load From Xml File** dialog box appears.

2. Browse to the location you wish to load the xml file.
3. Click Open in the **Load From Xml File** dialog box.

Load C1TileControl from XML file in code

To load template1 as an XML file in code, complete the following:

To write code in Visual Basic

Visual Basic

```
Private Sub btnLoadXml_Click(sender As Object, e As EventArgs)
       Using dlg As New OpenFileDialog()
              dlg.DefaultExt = ".xml"
              dlg.Filter = "XML files|*.xml|All files|*.*"
              dlg.Title = "Load From Xml File"
              If dlg.ShowDialog() = DialogResult.OK Then
                     Try
                           template1.LoadXml(dlg.FileName)
                     Catch
                           MessageBox.Show("Bad tilecontrol XML.", dlg.Title)
                     End Try
              End If
       End Using
End Sub
```

To write code in C#

C#

```
private void btnLoadXml_Click(object sender, EventArgs e)
{
    using (OpenFileDialog dlg = new OpenFileDialog())
    {
        dlg.DefaultExt = ".xml";
        dlg.Filter = "XML files|*.xml|All files|*.*";
        dlg.Title = "Load From Xml File";
        if (dlg.ShowDialog() == DialogResult.OK)
        {
            try
            {
                Tilecontrol.LoadXml(dlg.FileName);
            }
            catch
            {
                MessageBox.Show("Bad tilecontrol XML.", dlg.Title);
            }
        }
    }
}
```

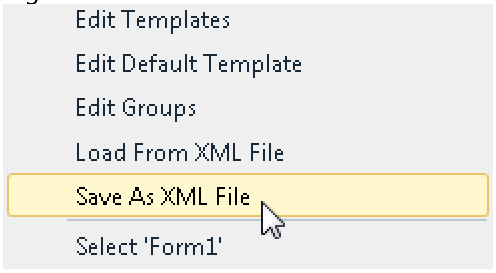
Saving TileControl as an XML File

This task shows how to save the C1TileControl as an XML File at design time and in code.

Save C1TileControl as an XML file at design time

To save the C1TileControl as an XML file at design time, complete the following:

1. Right-click the C1TileControl and select **Save as XML File** item from the context menu.



   The **Save As Xml File** dialog box appears.

2. Browse to the location you wish to save the .xml file.
3. Click **Save** in the **Save As Xml File** dialog box.

Save C1TileControl from XML file in code

To save the C1TileControl as an XML file in code, complete the following:

To write code in Visual Basic

Visual Basic

```
Private Sub menuitemSaveXml_Click(sender As Object, e As EventArgs)
       Using dlg As New SaveFileDialog()
              dlg.DefaultExt = ".xml"
              dlg.FileName = "tilecontrol
              dlg.Filter = "XML files|*.xml|All files|*.*"
              dlg.Title = "Save As Xml File"
              If dlg.ShowDialog() = DialogResult.OK Then
                     TileControl.SaveXml(dlg.FileName)
              End If
       End Using
End Sub
```

To write code in C#

C#

```
private void menuitemSaveXml_Click(object sender, EventArgs e)
{
    using (SaveFileDialog dlg = new SaveFileDialog())
    {
        dlg.DefaultExt = ".xml";
        dlg.FileName = "tilecontrol";
        dlg.Filter = "XML files|*.xml|All files|*.*";
        dlg.Title = "Save As Xml File";
        if (dlg.ShowDialog() == DialogResult.OK)
        {
            TileControl.SaveXml(dlg.FileName);
            }
        }
}
```

Setting Text for the TileControl and Group

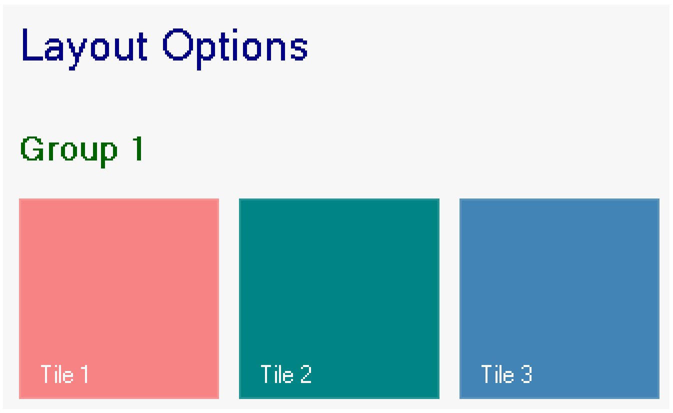
The TileControls text and its font size and color as well as the groups text, font size, and color can be applied to the TileControl and group programmatically or at design time.

To modify the TileControl and Group text, complete the following:

1. Right-click the **C1TileControl** and select **Properties**. The C1TileControl properties pane appears.
2. Enter the text inside the **Text** texbox that you wish to appear on the C1TileControl, for example **Layout Options**.
3. Set the **ForeColor** property to **Navy**. Note that this sets the groups forecolor as well. To change the groups forecolor it can be specified in the GroupForeColor property.
4. Set the **TextSize** property to **16**. This overrides the **Font.Size** property.
5. Expand the **Groups** node in the C1TileControls properties pane.
6. Set the **GroupFont** to **True** so the groups font appears bold.
7. Set the **GroupForeColor** to **DarkGreen**.

This topic illustrates the following:

The C1TileControl and Groups Text is modified.



Adding Groups to the C1TileControl

Groups can be created at design time or programmatically. At design time they can be created using the **C1TileControl.Groups Collection Editor**. Groups can also be created programmatically through the Group class. The group can be inserted into the GroupCollection at the specified index using the InsertItem method.

Design-Time

To add a template at design time, complete the following:

1. Right-click the **C1TileControl** and select **Edit Groups** to open the **C1TileControl.Groups Collection Editor**.
2. Click the **Add** button to add a new group to the C1TileControl. The Groups text will appear empty and there will be no tiles until you add the tiles to the group.

Removing Groups from the C1TileControl

Groups can be removed at design time or programmatically. At design time they can be removed using the **C1TileControl.Groups Collection Editor**. Groups can also be removed programmatically through the GroupCollection class. The group can be removed from the GroupCollection at the specified index using the RemoveItem method.

Design-Time

To remove a group from the C1TileControl complete the following:

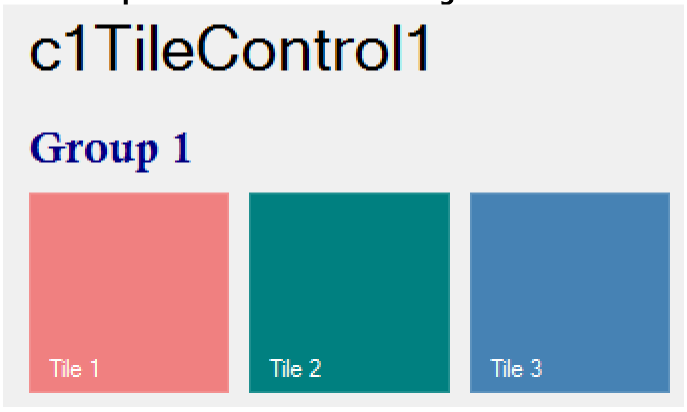
1. Add the TileControl to the windows Form.
2. Right-click the **C1TileControl** and select **Edit Groups** to open the **C1TileControl.Groups Collection Editor**.
3. Select the Group from the Members list that you want to remove and click the **Remove** button to remove a group from the C1TileControl.

Modifying the Groups Font Properties

To modify the Groups font properties complete the following:

1. Right-click on the **C1TileControl** and select **Properties**.
2. Locate the **GroupFont** property under the **Groups** node and click on the ellipsis button.
3. Set the Font to Calisto MT, FontStyle to Bold, and Size to 20. Note that the GroupTextBold and GroupTextSize will override these settings so if you set the FontStyle to Bold, but the GroupTextBold is set to False then the Group will inherit the setting from the Group.TextBold property.
4. Click **OK** to close and save the Font dialog box.
5. Set the GroupTextSize to 16. Note that this property takes precedence over the Font Size property applied in the Font dialog box.

This topic illustrates the following:



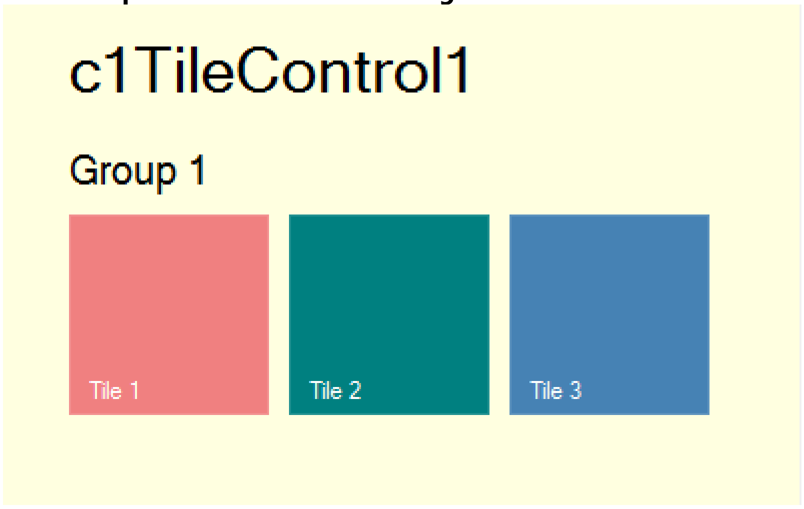
Setting the TileControls BackColor

The TileControls BackColor can be applied to the TileControl programmatically or at design time.

To set the TileControls BackColor using the Properties window, complete the following:

1. Right-click on the **C1TileControl** and select **Properties**.
2. Set the **BackColor** property to **LightYellow**.

This topic illustrates the following:



Adding Tiles to a Specific Group

Tiles can be created at design time or programmatically. At design time they can be created using the **C1TileControl.Groups Collection Editor** and **Group.Tiles Collection Editor**. Tiles can also be created programmatically through the Tile class. The tiles can be inserted into the TilesCollection at the specified index using the InsertItem method. You can then add the tiles to the appropriate group using the Group.Tiles property.

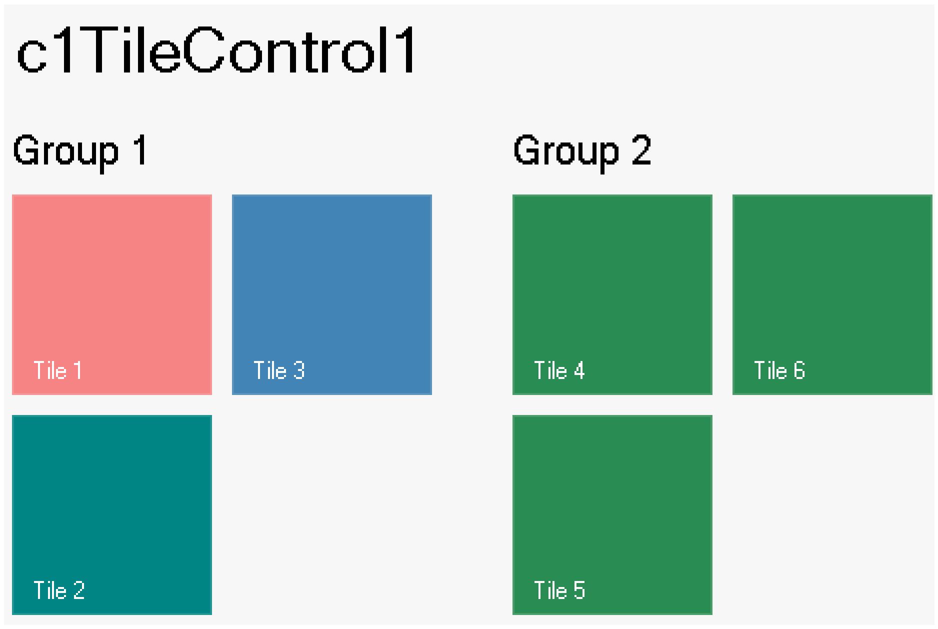
Design-Time

To add a template at design time, complete the following:

1. Right-click the **C1TileControl** and select **Edit Groups** to open the **C1TileControl.Groups Collection Editor**.
2. Select the Group from the members list where you want to add the tiles, for example **Group2**.
3. Click on the ellipsis button next to **Tiles** collection property.
4. Click add three times to add three **Tiles** to the second group, **group2**.
5. Click **OK** to save and close the **Group.Tiles Collection Editor** and click **OK** to save and close the **C1TileControl.Groups Collection Editor**.

This topic illustrates the following:

The three Tiles appear horizontally by default and they all have the same default dark green color:

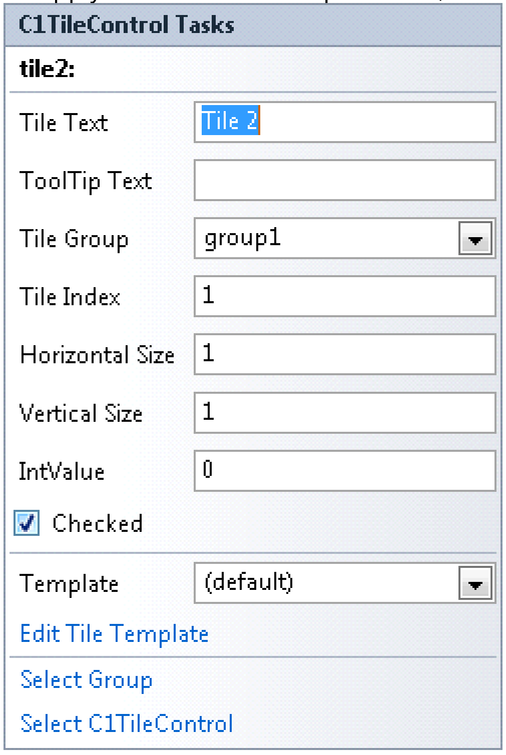


Creating a CheckMark for the Tile

A checkmark can be displayed it the upper right corner of the tile by setting the Checked property to True.

Design-Time

1. To apply a checkmark to a specific tile, select **tile2** and check the checkbox next to the Checked property.



C1TileControl Tasks

tile2:

Tile Text	Tile 2
ToolTip Text	
Tile Group	group1
Tile Index	1
Horizontal Size	1
Vertical Size	1
IntValue	0

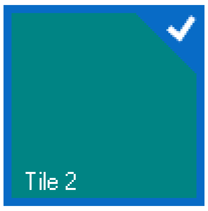
☑ Checked

Template: (default)

2. Right-click the **C1TileControl** and select **Properties**.
3. Under the **Appearance** node set the CheckMarkColor property to **Silver**.

This topic illustrates the following:

A checkmark appears in the upper right corner of tile2.



Increasing the Size of a Specific Tile

A Tiles size can be increased using the HorizontalSize and VerticalSize properties.

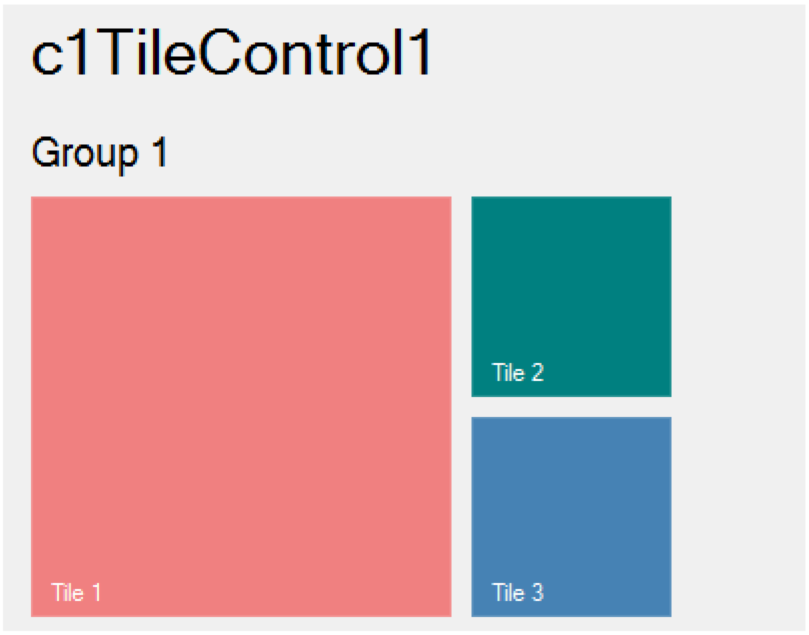
Design-Time

To increase the Tiles size at design-time, complete the following:

1. Select the Group and click on **Edit Tiles** from the C1TileControl Group Tasks menu.
2. In the **Group.Tiles Collection Editor** select **tile1[Tile 1]** and set the **HorizontalSize** to **2** and the **VerticalSize** to **2**.
3. Click **OK** to save and close the **Group.Tiles Collection Editor**.

This topic illustrates the following:

Tile 1s size in increased so it appears twice as large as the other two tiles.



Adding Image Elements to a Tile

The following tasks show different methods on how to add images to the tiles.

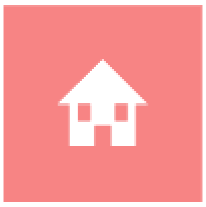
Adding a Symbol to a Tile

To add a symbol to a tile, complete the following:

1. Right-click on the **C1TileControl** and select **Edit Templates**. The **C1TileControl.Templates Collection Editor** appears.
2. Click **Add** to add a template.
3. Click on the ellipsis button next to the **Elements Collection** property. The **Template.Elements Collection Editor** appears.
4. Select **ImageElement** from the **Add** dropdown listbox.
5. Select **Library** from the **Symbol** dropdown listbox.
6. Select **Symbol** from the dropdown listbox of the **ImageSelector** property. This binds the value of the Symbol property to the specified Tile.
7. Click **OK** to save and close the **Template.Elements Collection Editor**.
8. Right-click the **C1TileControl** and select **Edit Groups**.
9. Click on the ellipsis button next to Tiles. The **Group.Tiles Collection Editor** appears.
10. Select the first Tile from the members list.
11. Set the **Symbol** property to **Home** and **Template** property to **template1**.

This topic illustrates the following:

The Home symbol is applied to the first Tile.



Drawing an Image at Runtime

Images can be drawn at runtime using the Paint event, like the following:

1. Right-click on the **C1TileControl** and select **Edit Templates**.
2. Click **Add** to add a template to the **C1TileControl**.
3. Add the following code to your project to draw an image at runtime:

To write code in Visual Basic

Visual Basic

```
Private Sub template1_Paint(sender As Object, e As
C1.Win.C1Tile.TemplatePaintEventArgs)
   Dim g As Graphics = e.Graphics
   g.SmoothingMode = System.Drawing.Drawing2D.SmoothingMode.HighQuality
   Dim rect As Rectangle = e.ClipRectangle
   rect.X += (rect.Width - 28) \ 2
   rect.Y += (rect.Height - 28) \ 2
   rect.Width = 28
   rect.Height = 28
   Dim brush As Brush = New SolidBrush(e.Tile.GetBackColor())
   Dim pen As New Pen(e.Tile.GetForeColor())
   Select Case e.Tile.IntValue
          Case 1
                 g.FillPie(brush, rect, 50F, 270F)
                 g.DrawPie(pen, rect, 50F, 270F)
                 Exit Select
          Case 2
                 g.FillRectangle(brush, rect)
                 g.DrawRectangle(pen, rect)
                 Exit Select
          Case Else
                 g.FillEllipse(brush, rect)
                 g.DrawEllipse(pen, rect)
                 Exit Select
   End Select
   brush.Dispose()
   pen.Dispose()
End Sub
```

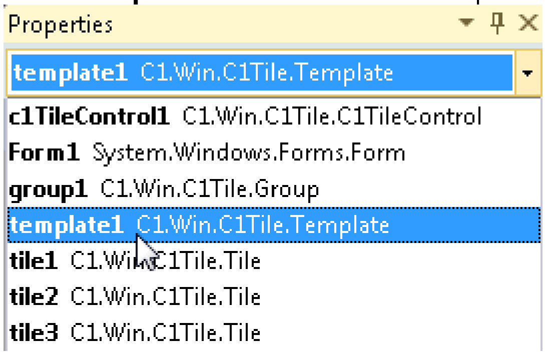
To write code in C#

C#

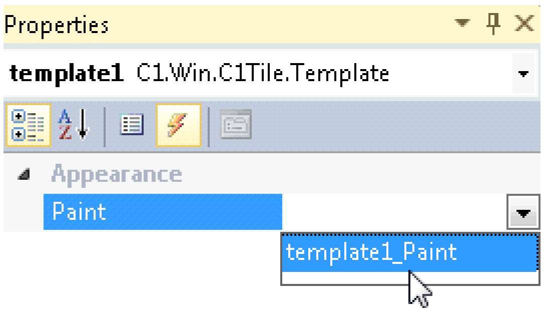
```
private void template8_Paint(object sender, C1.Win.C1Tile.TemplatePaintEventArgs
e)
   {
       Graphics g = e.Graphics;
       g.SmoothingMode = System.Drawing.Drawing2D.SmoothingMode.HighQuality;
       Rectangle rect = e.ClipRectangle;
       rect.X += (rect.Width - 28) / 2;
       rect.Y += (rect.Height - 28) / 2;
       rect.Width = 28;
       rect.Height = 28;
       Brush brush = new SolidBrush(e.Tile.GetBackColor());
       Pen pen = new Pen(e.Tile.GetForeColor());
       switch (e.Tile.IntValue)
       {
           case 1:
              g.FillPie(brush, rect, 50f, 270f);
              g.DrawPie(pen, rect, 50f, 270f);
              break;
           case 2:
              g.FillRectangle(brush, rect);
              g.DrawRectangle(pen, rect);
              break;
          default:
             g.FillEllipse(brush, rect);
             g.DrawEllipse(pen, rect);
             break;
       }
```

```
        brush.Dispose();
        pen.Dispose();
    }
```

4. In design view, right-click on the **C1TileControl** and select **Properties**.
5. Select **Template1** from the Windows dropdown listbox.


Properties
template1 C1.Win.C1Tile.Template
c1TileControl1 C1.Win.C1Tile.C1TileControl
Form1 System.Windows.Forms.Form
group1 C1.Win.C1Tile.Group
template1 C1.Win.C1Tile.Template
tile1 C1.Win.C1Tile.Tile
tile2 C1.Win.C1Tile.Tile
tile3 C1.Win.C1Tile.Tile

6. Click on the events button and set the Paint event to template1_Paint.



7. Select tile1 so its **C1TileControl Tasks** menu appears and set its properties to the following:
   - **Horizontal Size** to **1**.
   - **Vertical Size** to **1**.
   - **IntValue** to **0**. This will apply the first drawing to the first tile.
   - **Template** to **template1**.
8. Select tile2 so its **C1TileControl Tasks** menu appears and set its properties to the following:
   - **Horizontal Size** to **1**.
   - **Vertical Size** to **1**.
   - **IntValue** to **1**. This will apply the second drawing to the second tile.
   - **Template** to **template1**.
9. . Select tile2 so its **C1TileControl Tasks** menu appears and set its properties to the following:
   - **Horizontal Size** to **1**.
   - **Vertical Size** to **1**.
   - **IntValue** to **2**. This will apply the third drawing to the third tile.
   - **Template** to **template1**.
10. Run your project and observe the drawings made on the tiles.