ComponentOne

Ribbon for WinForms

GrapeCity US

GrapeCity
201 South Highland Avenue, Suite 301
Pittsburgh, PA 15206
Tel: 1.800.858.2739 | 412.681.4343
Fax: 412.681.4384
Website: https://www.grapecity.com/en/
E-mail: us.sales@grapecity.com

Trademarks

Warranty

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

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

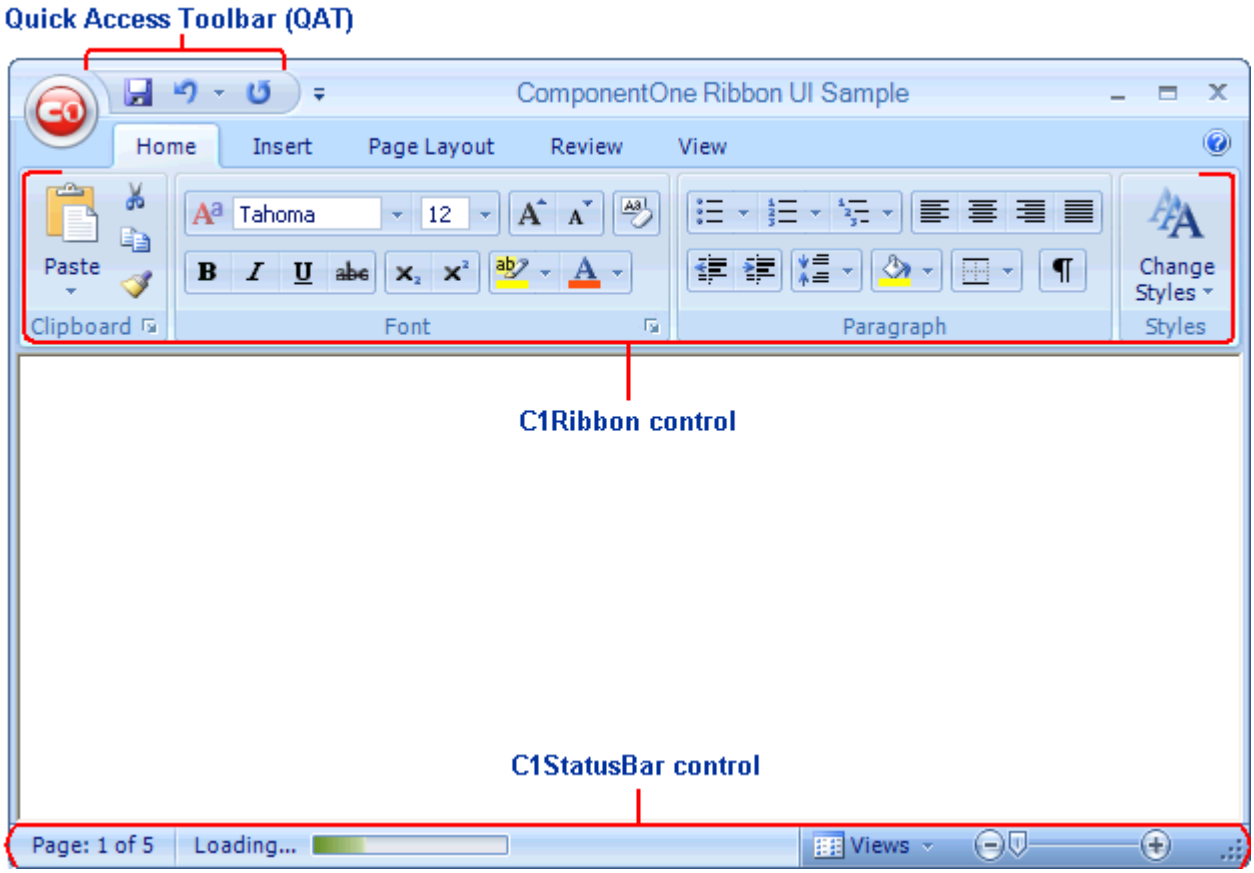
Copying and Distribution

Table of Contents

Ribbon for WinForms Overview

Make your application interface complete with a Ribbon style menu. **Ribbon for WinForms** delivers both Ribbon and Status Bar functionalities, with the same look and feel as Microsoft Office. This enables .NET developers to build Microsoft Office-style applications with little code and with great functionality.

Located at the top of the screen, the Ribbon replaces traditional menus, toolbars, and task panes with a simpler system of interfaces. It organizes related commands (in the form of controls) into groups and related groups into tabs so that the commands are easier to find. Perform tasks efficiently with reachable and discoverable functionalities.



Complete with ComponentOne's Smart Designer support, **Ribbon for WinForms**'s no-code design experience provides easy customization of the Quick Access Toolbar (QAT), tabs, groups, and other interface elements. With **Ribbon for WinForms**, creating Microsoft Office Ribbon style menus is this easy.

Help with WinForms Edition

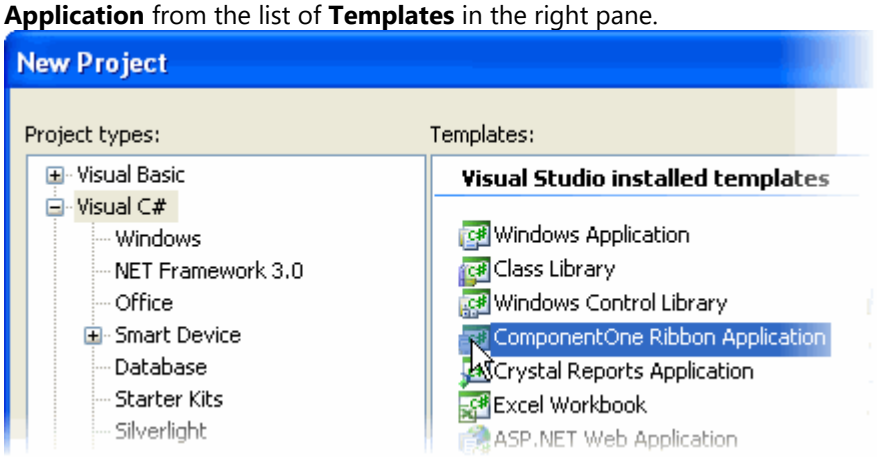
Getting Started

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.

Creating a Ribbon Application Project

If the Ribbon for WinForms Visual Studio templates are installed, you can very easily create a new Ribbon application. To create a new Ribbon application, follow these steps:

1. From the **File** menu in Microsoft Visual Studio, select **New** and click **Project**. The **New Project** dialog box opens.
2. Under **Installed | Templates** tab, select either **Visual Basic** or **Visual C#** project type and select **C1Ribbon Application** from the list of **Templates** in the right pane.



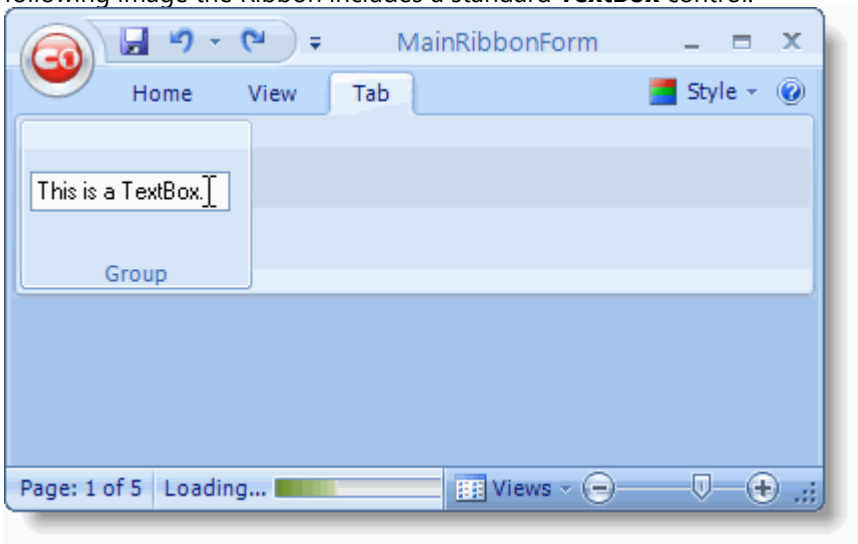
3. Enter or browse for a location for your application in the **Location** field and click **OK**.

A new Ribbon Application project is created at the specified location. In addition, two new Ribbon Forms, MainRibbonForm and ChildRibbonForm, are created.

Key Features

Create a sleek, Microsoft Office-style Ribbon by utilizing advanced features. Benefit from **Ribbon for WinForms**, featuring:

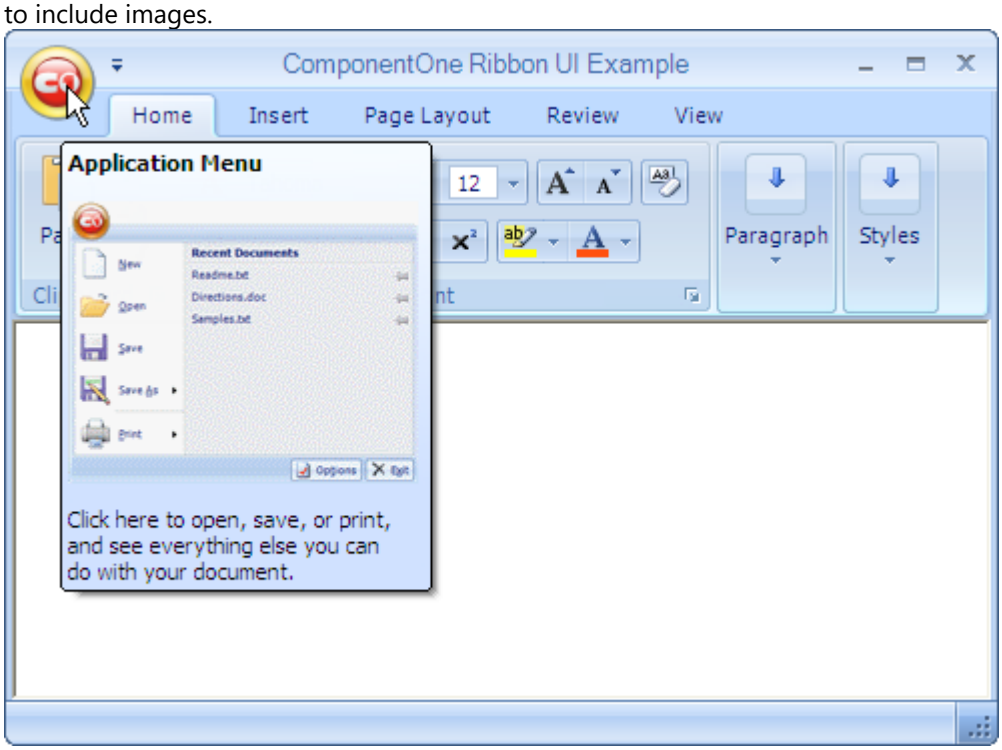
- **Widest range of Microsoft Office-style navigation elements**
  Ribbon for WinForms includes a rich set of Ribbon controls, from simple buttons to color palettes and galleries. Supported C1Ribbon elements include:
  Application Menu, Tabs, Groups (with optional Dialog Launcher Button), QAT (Quick Access Toolbar), Configuration Toolbar, Gallery, Toolbar, Menu, Color Picker, Combo Box, Font Combo Box, Edit Box, Check Box, Button, Toggle Button, Split Button, Label, Separator, Track Bar, Progress Bar
- **Same look and feel as of the Microsoft Office Ribbon user interface**
  Ribbon for WinForms UI elements achieve a look that is pixel-perfect with the MS Office interface, including: Ribbon controls, Status Bar controls, and a custom form.
- **Easily embed arbitrary controls in the C1Ribbon**
  Incorporate arbitrary controls in the Ribbon by using the RibbonControlHost element; for example in the following image the Ribbon includes a standard **TextBox** control.



  For more information about how to add arbitrary controls, see Embedding Controls in a Ribbon.
- **Design made easy through our design-time support**
  Provides a wide range of design-time support including smart tags, context-sensitive floating toolbars activated with a single mouse click, collection editors, and more.
- **Visual Studio templates make creating a Ribbon Application easy**
  Easily create Ribbon Applications and add Ribbon Forms to your project with installed Visual Studio templates. See Creating a Ribbon Application Project for more information.
- **Supports Office 2007, Office 2010, and Windows 7 Visual Styles**
  Change **Ribbon for WinForms**'s visual style by selecting one of the built-in Office 2007, Office 2010, Office 2016 or Windows 7 styles. C1Ribbon adapts to the cleaner, more powerful, more efficient Windows Aero interface.
- **Access to over 700 stock images for Ribbon items**
  Ribbon for WinForms includes a built-in image library for Application menu, groups, and group items. Two image sizes available: large (32x32) and small (16x16).
- **Accommodate a wealth of elements on an individual tab**
  When resizing is necessary, the group collapses.
- **Create a collection of ready-to-use templates**
  Create tabs and control groups and save as XML templates to import later – no need to start from scratch again.
- **Eliminate the tedious tasks associated with custom images**
  With the time-saving, automatic image processing, you do not have to provide highlighted or grayed version

of your custom images.

- **Easily create Microsoft Office-style ToolTips**
  The design-time multi-line editor enables you to easily create Microsoft Office -style ToolTips with the option to include images.



- **Easy access to the advanced capabilities using the dialog launcher button**
  Included in the functional group of the Ribbon, the dialog launcher connects the simple functionality of the Ribbon and the advanced functionality of the dialog box.
- **Supports Application Menu customization**
  At design time, change the Application button's image and customize the Application Menu to fit your needs; for example, display menu items as images, text, or both.
- **Place the QAT in the most accessible location**
  With just a mouse-click, move the position of the QAT above or below the Ribbon.
- **Quickly use a command with a keystroke**
  Use the keyboard to complete specific commands. To make text bold, for example, you could use a CTRL+B key combination.
- **Fully customizable Quick Access Toolbar (QAT)**
  With the design-time smart designer, a developer can effortlessly add C1Ribbon items or the C1Ribbon group to the QAT. The developer can easily simplify QAT customization at run time, and add C1Ribbon items to the drop-down menu adjacent to the QAT.
- **Context menu for Gallery items**
  Attach individual context menus to each item in the Gallery element.
- **Option to minimize the Ribbon**
  Easily minimize the Ribbon to make more space available on your screen.
- **32-bit and 64-bit compatibility**
  Ribbon for WnForms functions well in both 32-bit and 64-bit environments.

Ribbon for WinForms Quick Start

This section will lead you through the creation of a Ribbon Form that uses the C1Ribbon and C1StatusBar controls. In addition, it will show you how to modify the Ribbon's design, add some event handlers to the Ribbon, and load a sample text file in the editor. You will be able to create a simple text editor with the Ribbon user interface, by following the below steps:

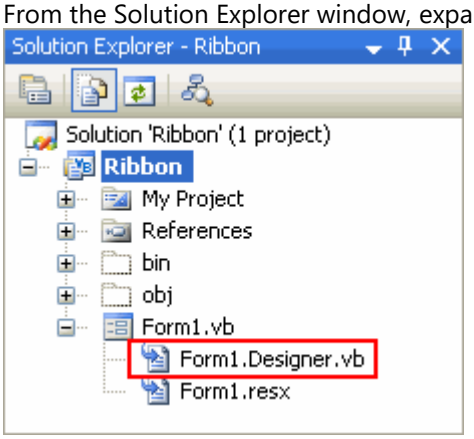
- Step 1 of 6: Add Controls to the Windows Form and Create the Ribbon Form
- Step 2 of 6: Add Ribbon Items to the Ribbon
- Step 3 of 6: Add Event Handlers to Ribbon Toggle Buttons in the Group
- Step 4 of 6: Set up the C1StatusBar
- Step 5 of 6: Load a Text File to the RichTextBox
- Step 6 of 6: Run the Quick Start Application

Step 1 of 6: Add Controls to the Windows Form and Create the Ribbon Form

To begin, create a Visual Studio WinForms project and add the C1Ribbon and C1StatusBar controls to your Toolbox. For more information on creating a new project, see Creating a .NET Project.

To set up your new Ribbon Form, complete the following steps:

1. **Add the C1Ribbon control to the Ribbon Form:**
   From the Toolbox, double-click the C1Ribbon control to add it to your form. The Ribbon docks at the top of the Ribbon form.
2. **Add the C1StatusBar control to the Ribbon Form:**
   From the Toolbox, double-click the C1StatusBar control to add it to your form. The status bar docks at the bottom of the Ribbon Form.
3. **Add the RichTextBox control to the Ribbon Form:**
   1. From the Toolbox, double-click the **RichTextBox** control to add it to your form.
   2. From the Properties window, set the **RichTextBox1.Dock** property to **Fill**.
4. **Create a Ribbon Form:**
   Change the Windows Form to a Ribbon Form by modifying the code that declares your form:
   In Visual Basic Language:
   1. From the Solution Explorer, click the **Show All Files** button in the toolbar.
   2. From the Solution Explorer window, expand the **Form1.vb** node to reveal the Form's Designer.



   3. Double-click **Form1.Designer.vb** to open the Code Editor.

In C# Language:
To open the Code Editor, right-click the Windows Form and select View Code.

Continue in all Languages:
In the Code Editor, replace the following:

To write code in Visual Basic

Visual Basic
```
Partial Class Form1
    Inherits System.Windows.Forms.Form
    '...
End Class
```

To write code in C#

C#
```
partial class Form1 : Form {
    //...
}
```

With

To write code in Visual Basic

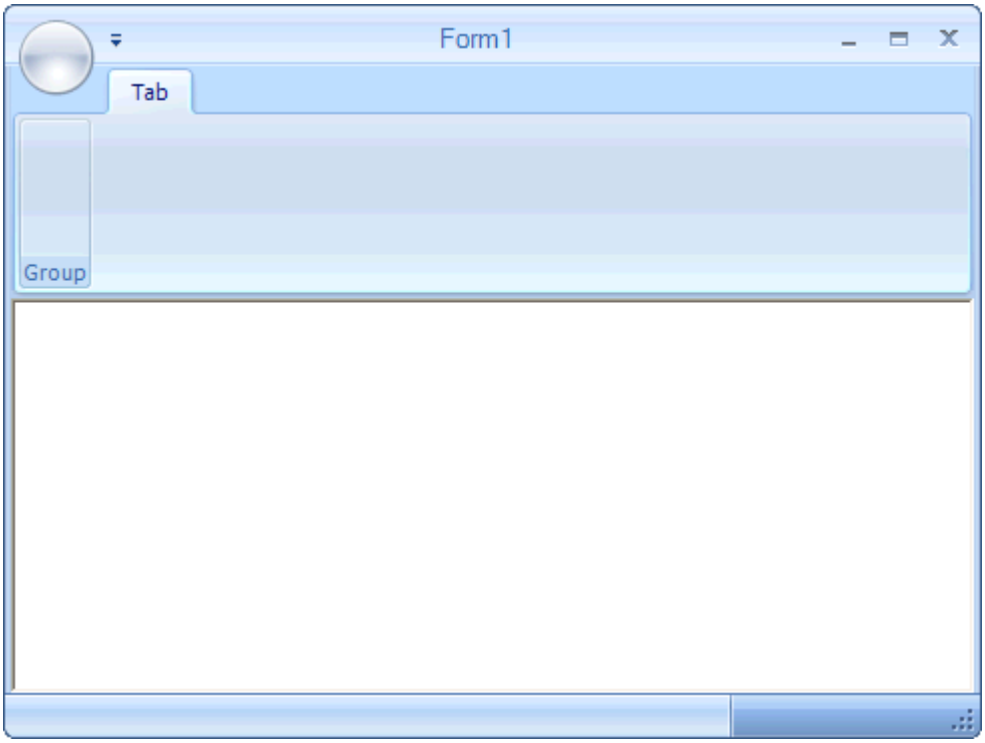
Visual Basic
```
Partial Class Form1
    Inherits C1.Win.C1Ribbon.C1RibbonForm
    '...
End Class
```

To write code in C#

C#
```
partial class Form1 : C1.Win.C1Ribbon.C1RibbonForm {
   //...
}
```

Run you application and observe:

You have successfully added the C1Ribbon, C1StatusBar, and **RichTextBox** controls to your project and created a Ribbon Form:



In the next step you will add Ribbon items to the Ribbon.

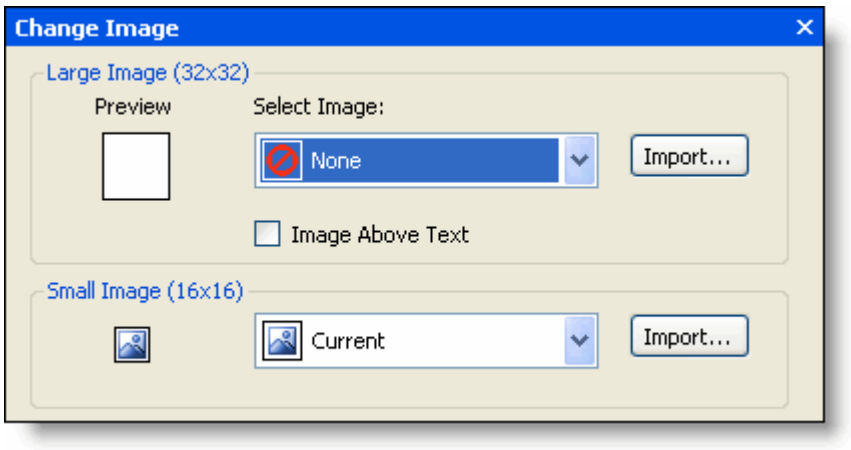
Step 2 of 6: Add Ribbon Items to the Ribbon

In the previous step when you added the C1Ribbon control to your Form, you may have noticed that the Ribbon included a pre-added tab with a group. In this step you'll add some Ribbon items to the group and then modify their properties using the smart designer.

Switch to Design view and complete the following steps:

1. **Add a toolbar to the Ribbon group:**
    1. Click the pre-added group **ribbonGroup1** to activate the item and enable the group's floating toolbar.
    2. Click the **Actions** button. A list of actions is revealed.
    3. Select **Add ToolBar** from the list of actions.

    A toolbar with a Ribbon button is added to the group.
2. **Add a toggle button and modify its properties:**
    1. Click the toolbar to activate the item and to enable the floating toolbar.
    2. From the floating toolbar, click the **Actions** button and select **Add ToggleButton.**
    A toggle button is added to the toolbar next to the existing button.
    3. From the ToggleButton1 floating toolbar, click the Change Image button. The Change Image dialog box appears.

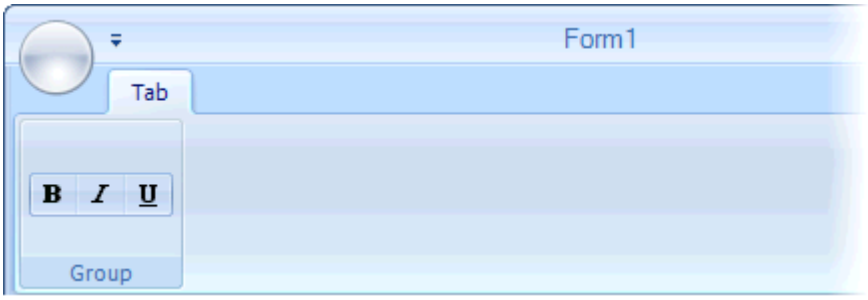


4. **From the** Small Image **(16x16) drop-down list, select** Bold**.**
5. **From the** ToggleButton1 **floating toolbar, click the** Text Settings **button. The** Text Settings **dialog box appears.**
6. **In the** Text Settings **dialog box, set the following properties:**
    - Delete the default "ToggleButton" Text.
    - Set the RibbonItem.ToolTip property to "Bold".

3. **Delete the first Ribbon button included with the toolbar:**
   When you add a Ribbon toolbar item there's a Ribbon button included in the container. Complete the following steps to remove the first Ribbon button (since this toolbar will contain three Ribbon *toggle* buttons):
    1. Select the Ribbon button to active the item and enable the item's floating toolbar.
    2. Click the **Actions** button and select **Delete** to remove the button from the toolbar.

   You should now have one toggle button in the toolbar.
4. **Add a second toggle button and modify its properties:**
   Click the toolbar to activate the item and to enable the floating toolbar, then complete the following steps:
    1. **From the floating toolbar, click the** Actions **button and select** Add ToggleButton**.**
       The second toggle button is added to the toolbar.
    2. **Use the toggle button's floating toolbar to modify the properties for the ToggleButton2. Click** ToggleButton2 **to activate the floating toolbar and complete the following tasks:**
        - Set the **Image** property to a 16x16 **Italic** preset image.
        - Delete the default "ToggleButton" Text.
        - Set the RibbonItem.ToolTip property to "Italic**"**.
5. **Add a third toggle button and modify its properties:**
   Click the toolbar to activate the item to enable the floating toolbar, then complete the following steps:
    1. **From the floating toolbar, click the** Actions **button and select** Add ToggleButton**.**
       The third toggle button is added to the toolbar.
    2. **Use the toggle button's floating toolbar to modify the properties for the ToggleButton3. Click** ToggleButton3 **to activate the floating toolbar and complete the following tasks:**
        - Set the **Image** property to a 16x16 **Underline** preset image.
        - Delete the default "ToggleButton" Text.
        - Set the RibbonItem.ToolTip property to "Underline".

The Ribbon group should now look like the following:



In this step you successfully added Ribbon items to the group and modified their properties. In the next step you will change the Ribbon tab and group names and add event handlers to the three toggle buttons in RibbonGroup1.

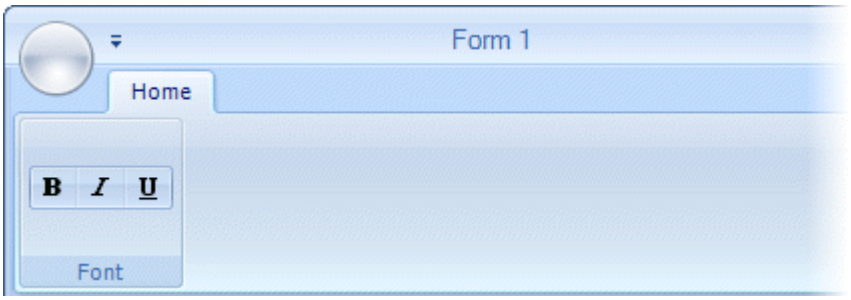
Step 3 of 6: Add Event Handlers to Ribbon Toggle Buttons in the Group

In this step you'll change the text that appears as the Ribbon tab and group names, and then add event handlers to enable the Bold, Italic, and Underline Ribbon buttons in the Ribbon's Font group.

Before you add event handlers to the toggle buttons, change the default tab and group **Text** properties. To do this, complete the following steps:

1. Double-click the default "Tab" text on the Ribbon to highlight it. The tab's text is ready to be edited.
2. Type the text "Home".
3. Press ENTER or click outside the editing box to accept the change. **Home** now appears as the tab's name.
4. Double-click the default "Group" text on the Ribbon to highlight it. The group's text is ready to be edited.
5. Type the text "Font".
6. Press ENTER or click outside the editing box to accept the change. The group's name now appears as **Font**.

The Ribbon group should now look like the following:



To enable the Ribbon buttons (Bold, Italic, and Underline) in the Ribbon's Font group, enter the following code in the Code Editor:

To write code in Visual Basic

Visual Basic

```
' type the Imports directive for the namespace
Imports C1.Win.C1Ribbon

' handles the Click event for the Bold
Private Sub RibbonToggleButton1_Click(ByVal sender As Object, ByVal e As EventArgs)
Handles RibbonToggleButton1.Click
    ' assign style for Bold button
```

```
    ToggleSelectionFontStyle(FontStyle.Bold)
End Sub

' handles the Click event for the Italic button
Private Sub RibbonToggleButton2_Click(ByVal sender As Object, ByVal e As EventArgs)
Handles RibbonToggleButton2.Click
    ' assign style for Italic button
    ToggleSelectionFontStyle(FontStyle.Italic)
End Sub

' handles the Click event for the Underline button
Private Sub RibbonToggleButton3_Click(ByVal sender As Object, ByVal e As EventArgs)
Handles RibbonToggleButton3.Click
    ' assign style for Underline button
    ToggleSelectionFontStyle(FontStyle.Underline)
End Sub

' apply font style to the RichTextBox
Sub ToggleSelectionFontStyle(ByVal fontStyle As FontStyle)
  If Me.RichTextBox1.SelectionFont Is Nothing Then
      MessageBox.Show("Cannot change font style while selected text has more than one
font.")
  Else
      Me.RichTextBox1.SelectionFont = New Font(Me.RichTextBox1.SelectionFont,
Me.RichTextBox1.SelectionFont.Style Xor fontStyle)
  End If
  Me.RichTextBox1.Focus
End Sub
```

To write code in C#

C#

```
// type the using directive for the namespace
using C1.Win.C1Ribbon;

// handles the Click event for the Bold button
private void ribbonToggleButton1_Click(object sender, EventArgs e)
{
    // assign style for Bold button
    ToggleSelectionFontStyle(FontStyle.Bold);
}

// handles the Click event for the Italic button
private void ribbonToggleButton2_Click(object sender, EventArgs e)
{
    // assign style for Italic button
    ToggleSelectionFontStyle(FontStyle.Italic);
}

// handles the Click event for the Underline button
private void ribbonToggleButton3_Click(object sender, EventArgs e)
```

```
{
    // assign style for Underline button
    ToggleSelectionFontStyle(FontStyle.Underline);
}

// apply font style to the richTextBox
void ToggleSelectionFontStyle(FontStyle fontStyle)
{
    if (this.richTextBox1.SelectionFont == null)
    {
        MessageBox.Show("Cannot change font style while selected text has more than
one font.");
    }
    else
    {
        this.richTextBox1.SelectionFont = new Font(this.richTextBox1.SelectionFont,
            this.richTextBox1.SelectionFont.Style ^ fontStyle);
    }

    this.richTextBox1.Focus();
}
```

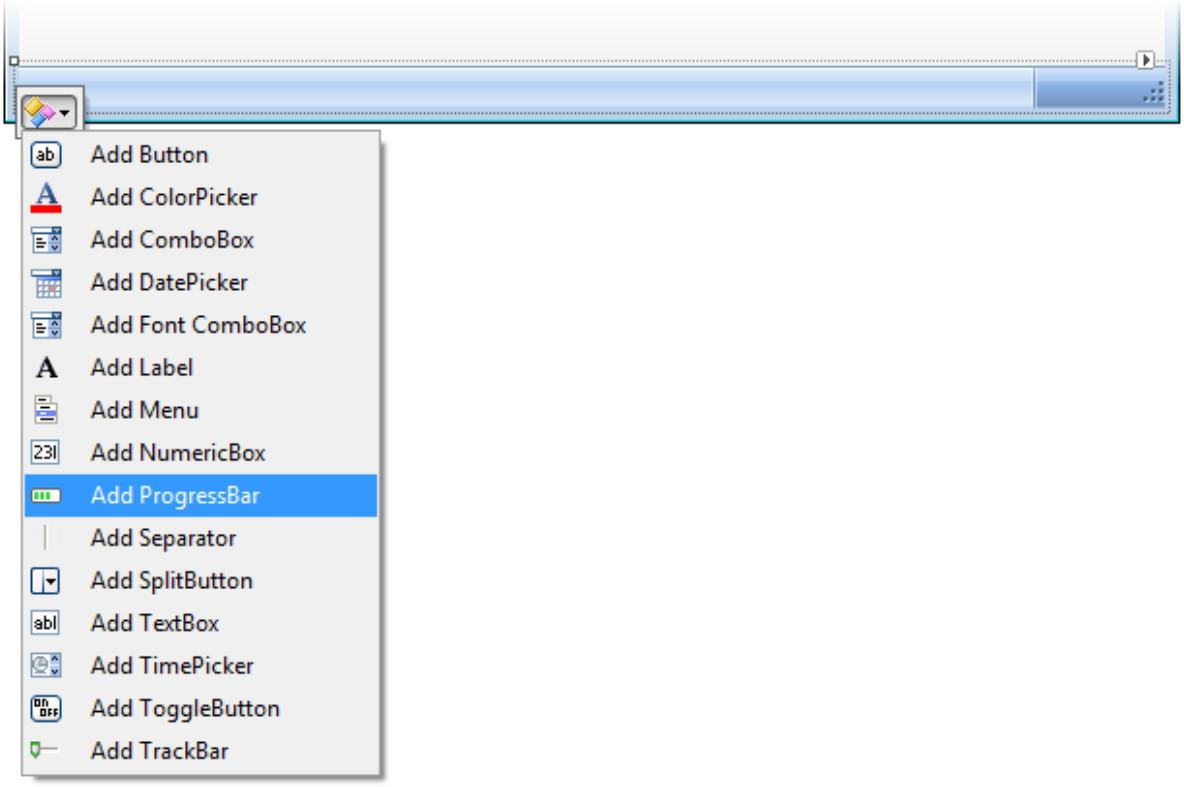
You have successfully added event handlers to the three toggle buttons in RibbonGroup1. In the next step, you will set up the status bar by adding a progress bar in the left pane and a track bar in the right pane.

Step 4 of 6: Set up the C1StatusBar

The C1StatusBar is displayed at the bottom of the Ribbon Form. The C1StatusBar provides panel styles which are used to provide feedback to the Ribbon end users.

To add panel styles which enable the end-users to see the progress bar in the left pane and the track bar in the right pane, complete the following steps:

1. **Add a ProgressBar to the left status bar pane:**
    1. Click **C1StatusBar LeftPaneItems** to activate the item to enable the floating toolbar.
    2. Click the **Actions** button. A list of actions is revealed.
    3. Select **Add ProgressBar**:



4. Select the progress bar to activate it, and set following properties in the Properties window:
   - **Name** = "progressbar"
   - RibbonProgressBar.Value = **30**

2. **Add a Button to the right status bar pane and modify its properties:**
   1. Click C1StatusBar.RightPaneItems to activate the item and to enable the floating toolbar.
   2. Click the **Actions** button. A list of actions is revealed.
   3. Select Add Button.
   4. Select the button to activate it, and set the following properties in the Properties window:
      - **Name** = "button"
      - RibbonItem.SmallImage = **None**
      - RibbonButton.Text = "30%"
3. **Add a TrackBar to the right status bar pane and modify its properties:**
   1. Click C1StatusBar.RightPaneItems to activate the item and to enable the floating toolbar.
   2. Click the **Actions** button. A list of actions is revealed.
   3. Select **Add TrackBar**. The track bar is added to the right status bar pane.
   4. Select the track bar to activate it, and in the **Properties** window set the **Name** property to "trackbar".
4. **Adjust the width of the right status bar pane:**
   Select the right status bar pane to activate it, and set the C1StatusBar.RightPaneWidth property to **150** in the Properties window.
5. **Add code to enable the left and right status bar pane items:**

   In the Code Editor, add the following code to enable the items on the left and right panel:

   To write code in Visual Basic

```
Visual Basic
' type the Imports directive for the namespace
Imports C1.Win.C1Ribbon

Private Sub Form1_Load(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles MyBase.Load
    trackbar.SmallChange = 1
```

```
    trackbar.LargeChange = 5
    trackbar.Minimum = 0
    trackbar.Maximum = 100
    trackbar.Value = 30
    AddHandler trackbar.Scroll, AddressOf trackbar_Scroll
End Sub

Sub trackbar_Scroll(ByVal sender As Object, ByVal e As EventArgs)
    Dim val As Integer = trackbar.Value
    progressbar.Value = val
    button.Text = val.ToString + "%"
End Sub
```

To write code in C#

C#

```
// type the using directive for the namespace
using C1.Win.C1Ribbon;

private void Form1_Load(object sender, EventArgs e)
{
    trackbar.SmallChange = 1;
    trackbar.LargeChange = 5;
    trackbar.Minimum = 0;
    trackbar.Maximum = 100;
    trackbar.Value = 30;
    trackbar.Scroll += new EventHandler(trackbar_Scroll);
}
void trackbar_Scroll(object sender, EventArgs e)
{
    int val = trackbar.Value;
    progressbar.Value = val;
    button.Text = val.ToString() + "%";
}
```

You have successfully added items to the status bar. Next, you will load a rich text file (RTF) for editing.

Step 5 of 6: Load a Text File to the RichTextBox

In this step you will load a rich text file (RTF) into your project as the main body of your text editor. To load a RTF for editing, complete the following steps:

Add the following code to the **Form.Load** event:

To write code in Visual Basic

Visual Basic

```
Me.RichTextBox1.LoadFile("C:\MyFile.rtf")
```

To write code in C#

C#

```
this.richTextBox1.LoadFile(@"C:\MyFile.rtf");
```

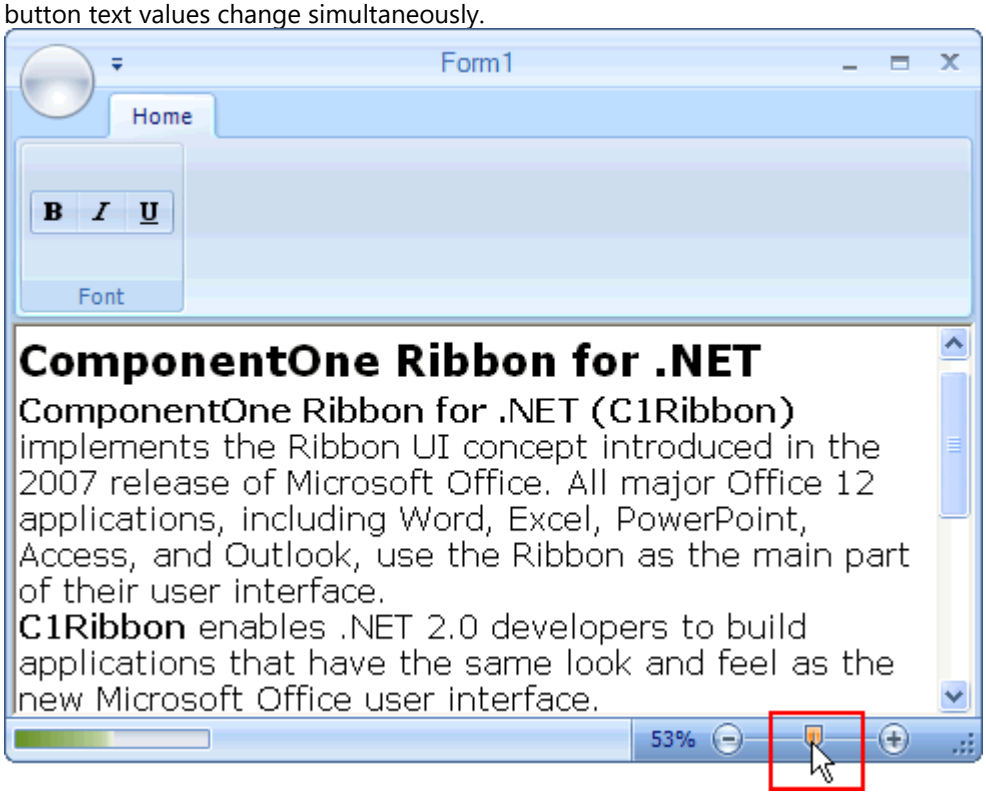
Note that you will have to change file path to match a file path on your machine. You may load any RTF file to use for editing.

Now that you have added the RTF for editing, you can run the application and explore the run-time interaction in the last step of the quick start.

Step 6 of 6: Run the Quick Start Application

To run the quick start application, click the **Start Debugging** button and notice the following results:

- Select some text in the text box and click the **Bold**, **Italic**, and **Underline** buttons to format the text.
- Click the and buttons on the track bar or click and drag the slider and notice the progress bar and the button text values change simultaneously.



Congratulations, you have completed the C1Ribbon quick start! In this quick start you have learned how to:

- Create a Ribbon Form that uses the C1Ribbon and C1StatusBar controls.
- Modify the Ribbon's design, by adding items to the Ribbon group and status bar.
- Add some event handlers to the Ribbon.
- Load a sample text file in the editor.

Ribbon for WinForms Elements

The following Ribbon elements make up the Ribbon User Interface (UI):

- Ribbon Form
  The Ribbon Form is represented by the C1RibbonForm class and has been calculated to display the C1Ribbon and C1StatusBar controls.
- Application Button
  The main large, round button at the top-left corner of the application window, which presents a drop-down menu (RibbonApplicationMenu) when the button is clicked.
- Quick Access Toolbar
  The RibbonQat is a customizable toolbar with frequently used buttons and other elements.
- Configuration Toolbar
  The RibbonConfigToolBar is a customizable toolbar with commonly used commands.
- Ribbon Tab
  The RibbonTab organizes related Ribbon groups on one page.
- Ribbon Item Group
  The RibbonGroup represents a group of items on the Ribbon tab.
- Ribbon Containers
  The RibbonItemContainer is a Ribbon item that can contain other Ribbon items.
- Ribbon Items
  The RibbonItem exists in a Ribbon container or group.
- Status Bar Items
  The C1StatusBar items exist in the right or left pane of the status bar.

For more information on each Ribbon item, see the following topics.

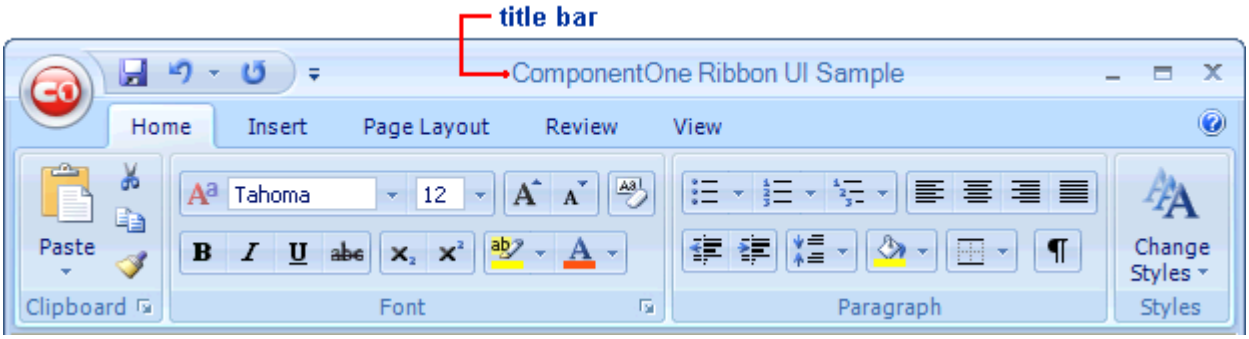
Ribbon Form

The Ribbon Form is represented by the C1RibbonForm class and has been calculated to display the C1Ribbon and C1StatusBar controls. The Ribbon Form, like a Windows Form, contains a title bar and a client region. When the C1Ribbon control is added to the Ribbon Form, the title bar is automatically displayed with the Quick Access Toolbar (QAT) within the title bar.

Ribbon Form's Elements

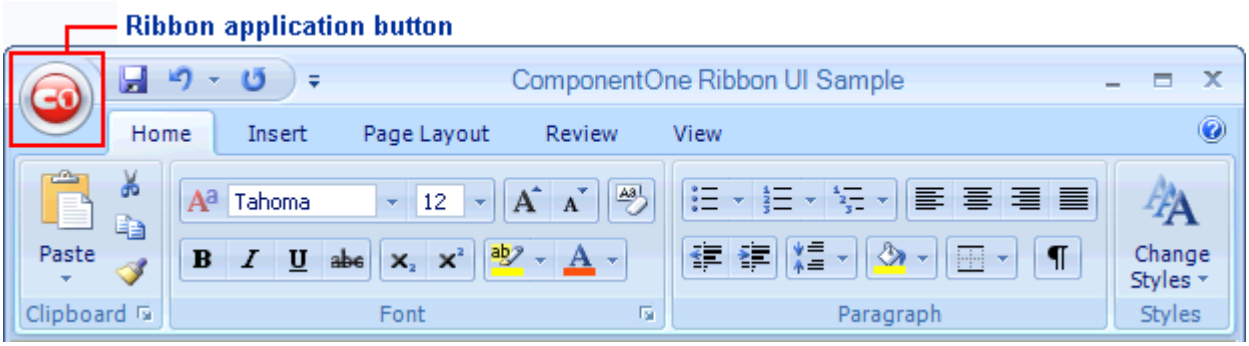
Title Bar

The Ribbon Forms title bar displays the form's caption, which can be changed using the **Form.Text** property.



Application Button

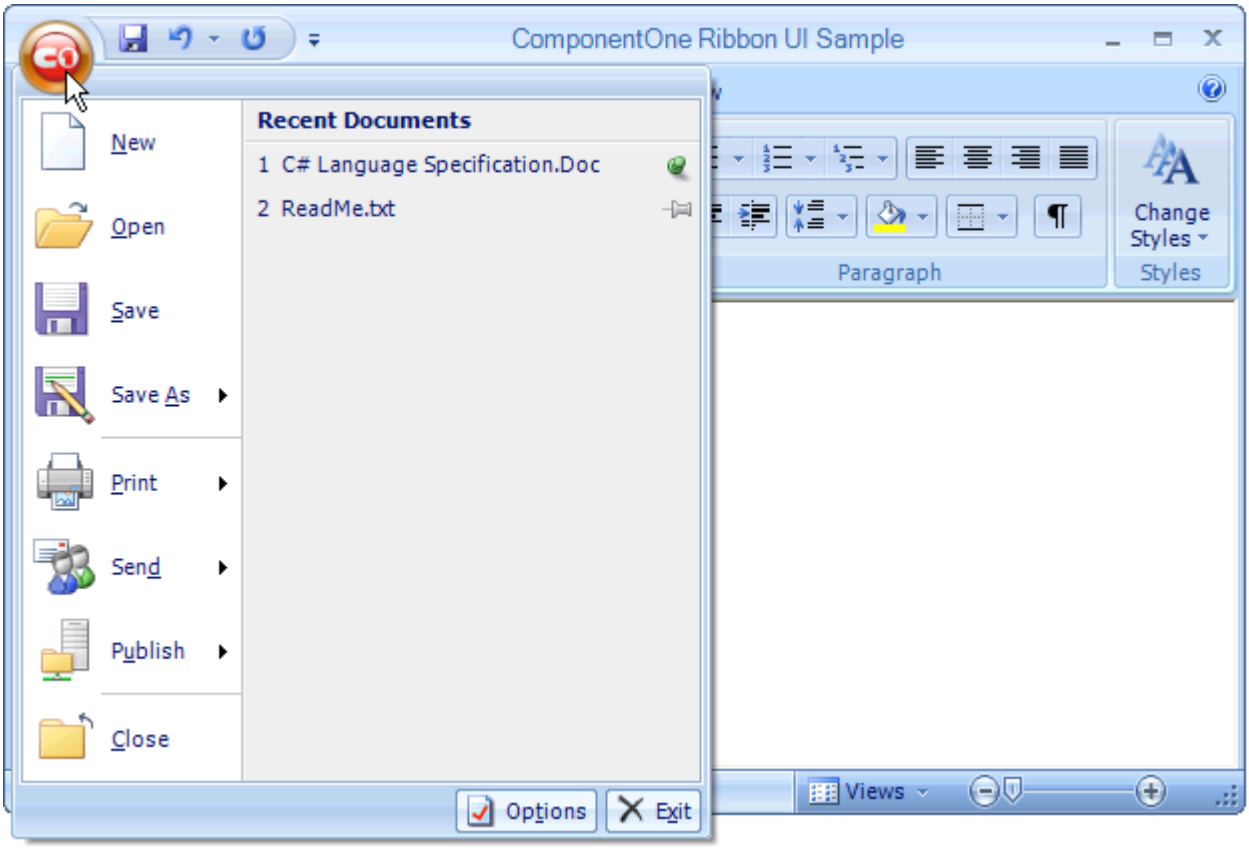
The main application button is located at the upper-left corner of the Ribbon within the Ribbon Form's title bar.



Application Menu

The RibbonApplicationMenu provides access to the Application menu of main commands that operate on the document as a whole. The menu items can be displayed as images, text, or both.

At run time, simply click the application button to reveal the list of main commands. Here is an example of an Application menu that was created:



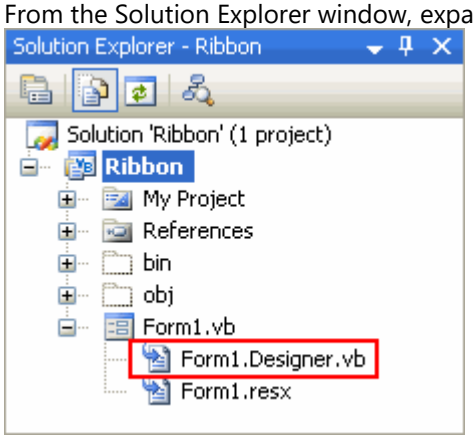
You can easily customize the main button to fit your needs. For example, you can add a custom image, add a Windows 7-style application button, and add your own items to the Start menu. For details on modifying the main application button, see the Importing a Custom Image for the Application Button, Making a Windows 7-Style Application Button, Changing the Color of the Application Button, and Creating the Application Menu topics.

How to Create a Ribbon Form

When you create a new Windows Application project, a Windows Form appears in design view. To change the Windows Form to a Ribbon Form, complete the following steps:

In Visual Basic Language:

1. From the Solution Explorer, click the **Show All Files** button in the toolbar.
2. From the Solution Explorer window, expand the **Form1.vb** node to reveal the Form's Designer.



3. Double-click **Form1.Designer.vb** to open the Code Editor.

In C# Language:

4. To open the Code Editor, right-click the Windows Form and select **View Code**.

Continue in all Languages:

5. In the Code Editor, replace the following:

To write code in Visual Basic

Visual Basic

```
Partial Class Form1
    Inherits System.Windows.Forms.Form
    '...
End Class
```

To write code in C#

C#

```
partial class Form1 : Form {
    //...
}
```

With

To write code in Visual Basic

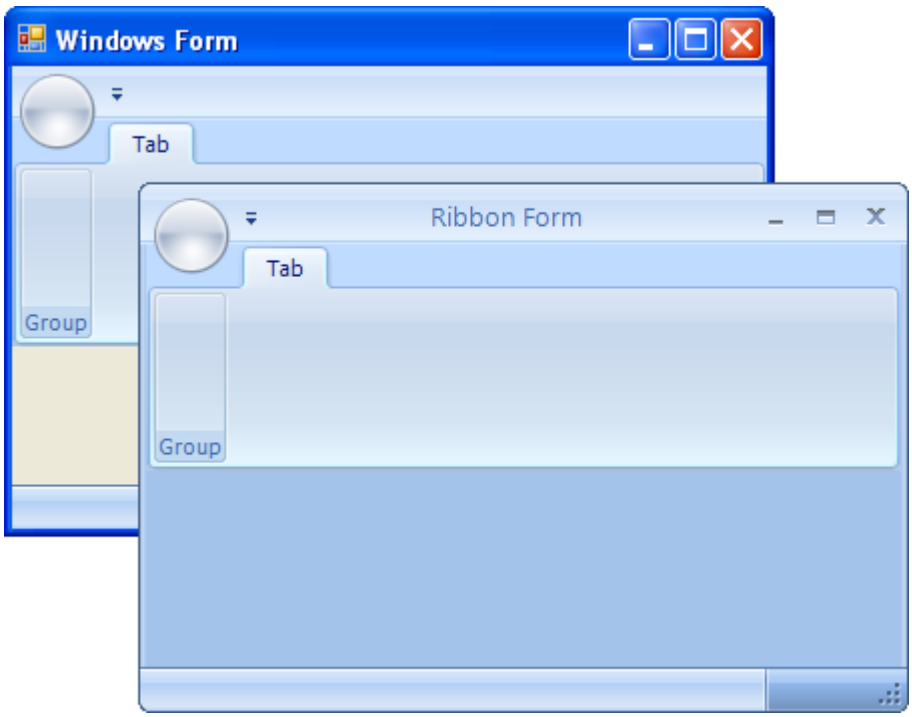
```
Visual Basic
Partial Class Form1
    Inherits C1.Win.C1Ribbon.C1RibbonForm
    '...
End Class
```

To write code in C#

```
C#
partial class Form1 : C1.Win.C1Ribbon.C1RibbonForm {
   //...
}
```

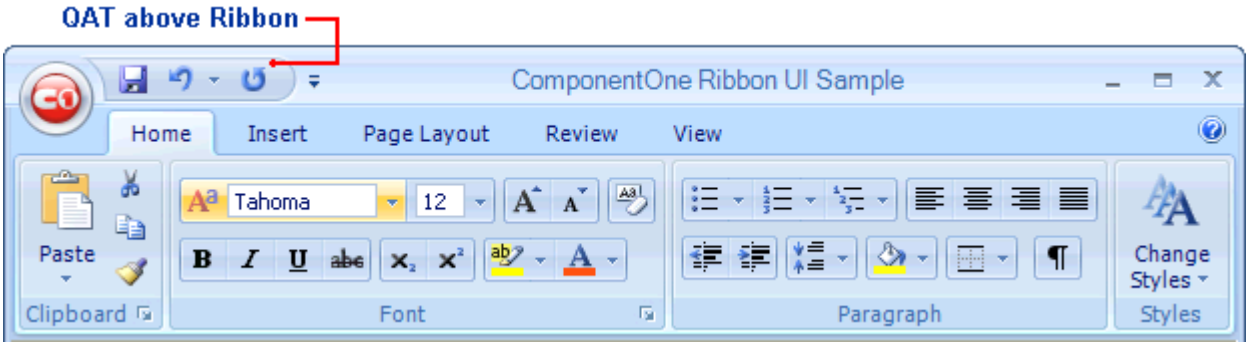
6. Switch back to Design view and notice that the Form now has the look and feel of the Ribbon style.

   Here you can see the difference between the Windows Form and the Ribbon Form:

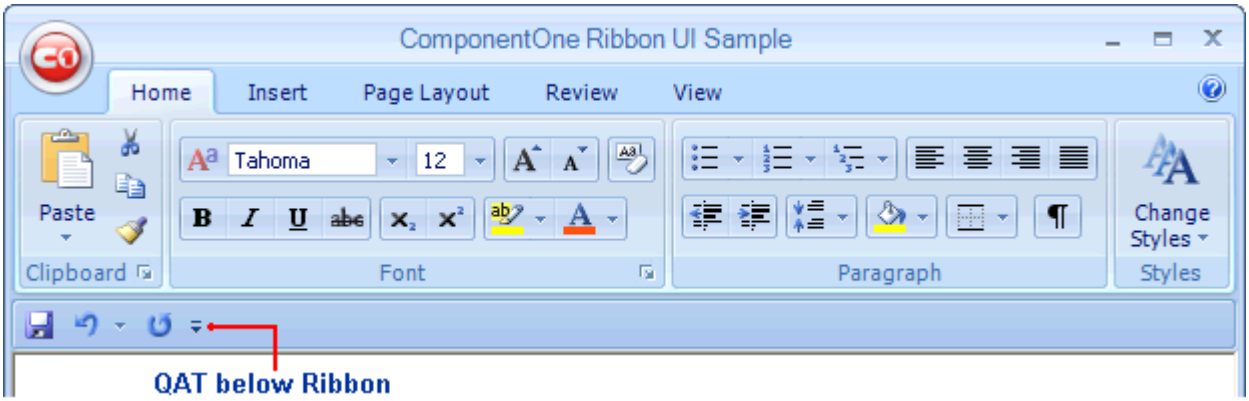


Quick Access Toolbar

The Quick Access Toolbar (QAT) provides quick access for commonly used commands to increase productivity. The QAT is located in the upper-left corner next to the main application button or below the Ribbon. The following Ribbon shows a QAT with three Ribbon button items (Save, Undo, and Repeat):



The user can position the QAT below the Ribbon. To change the position of the QAT at design time, use the RibbonQat.BelowRibbon property. To move the QAT below the Ribbon at run time, click the drop-down arrow next to the QAT and select **Show Below the Ribbon**.

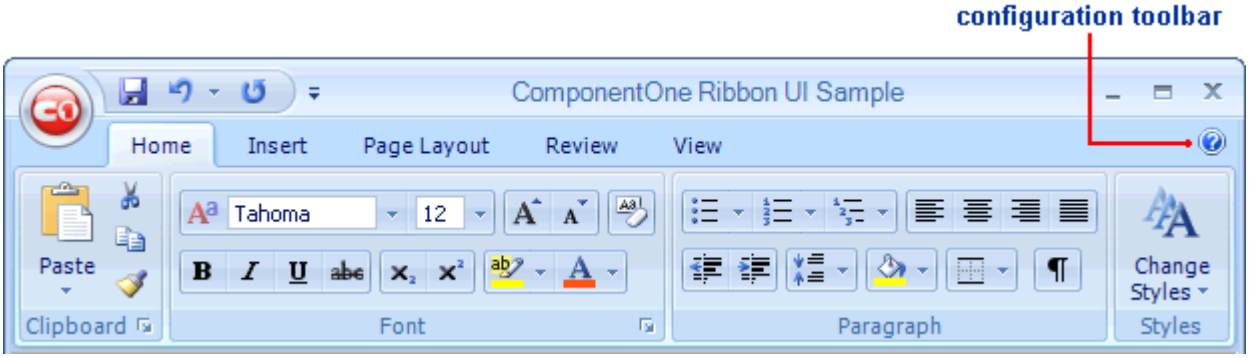


The user can set whether or not the **Customize QAT** drop-down button is visible using the MenuVisible property. To change the setting, expand the QAT properties and use the drop-down arrow next to MenuVisible to select **true** or **false**.

You can add or remove items to the QAT to customize it to fit your needs at design time. Additionally, the user can customize the QAT at run time. By placing the most commonly used items in the QAT, you eliminate the need to hunt through tabs and groups since your items are now in the forefront. For steps showing how to customize the QAT at design time, see Adding Items to the Quick Access Toolbar.

Configuration Toolbar

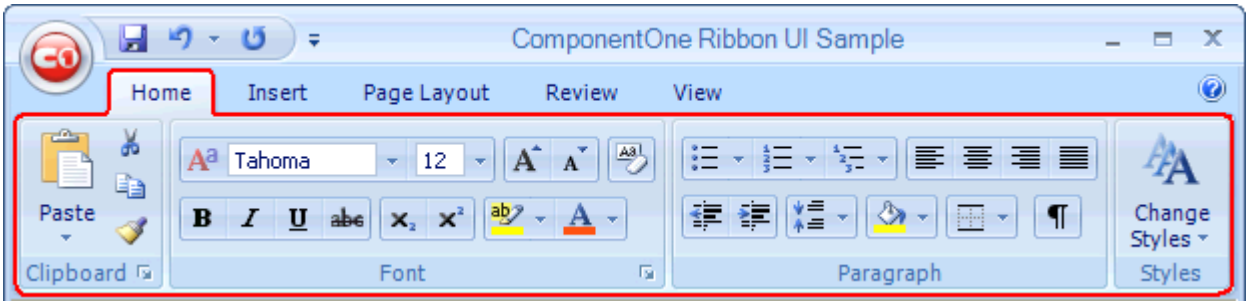
The Ribbon configuration toolbar (RibbonConfigToolBar) provides another way for the user to place commonly used commands. Unlike the QAT, the configuration toolbar cannot be changed by the user or moved below the Ribbon. The configuration toolbar is always located in the upper-right corner of the Ribbon on the same level as the Ribbon tabs. The following Ribbon shows the configuration toolbar with one Ribbon button item (Help):



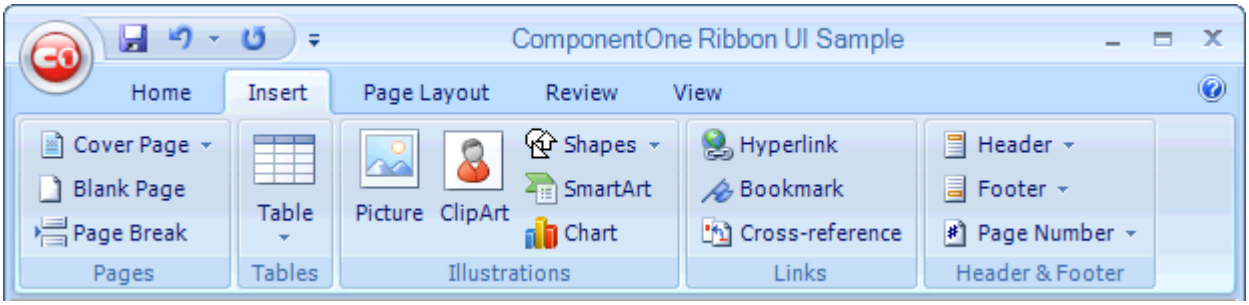
You can add or remove Ribbon items to the configuration toolbar to customize it to fit your needs. By placing the items in the configuration toolbar, you eliminate the need to hunt through tabs and groups since your items are now in the forefront. For steps showing how to customize the configuration toolbar at design time, see Adding Items to the Configuration Toolbar.

Ribbon Tab

The RibbonTab represents the main functionality groups of the program; the tab contains an organized collection of program tasks. The following image shows the **Home** tab with four groups: Clipboard, Font, Paragraph, and Styles.



One click on a tab provides access to commands with labels that help users identify the appropriate icon or button for a specific command. For example, the following Ribbon has an **Insert** tab, which has numerous tasks that are organized into groups and have labels. This organization makes it easy for the user to discover commands necessary to complete a specific action. Additionally, within the tab the commands are organized to provide a visual hierarchy, making it easier for the user to browse.

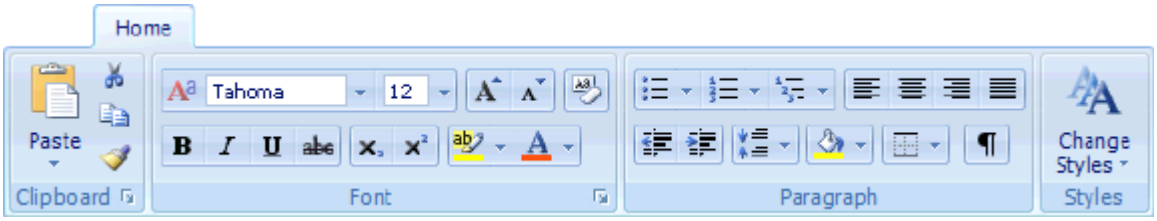


Ribbon Item Group

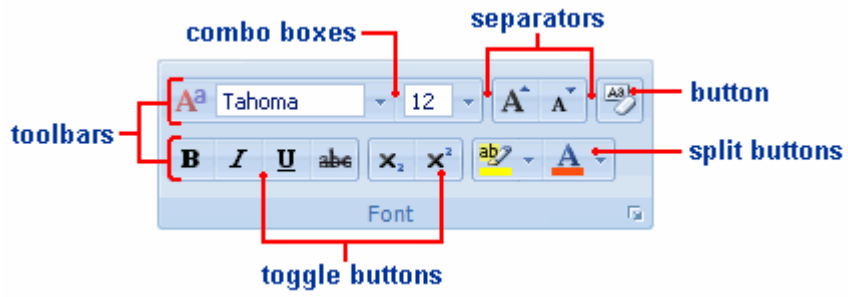
Each Ribbon tab contains a set of Ribbon groups. Grouping related commands in a RibbonGroup makes it easier to discover and locate the commands. The TrimLongCaption property allows the RibbonGroup caption to be trimmed if

it is longer than the contents of the group. The group can also include a dialog box launcher button which provides access to advanced capabilities.

The following image shows the **Home** tab with four groups: Clipboard, Font, Paragraph, and Styles:

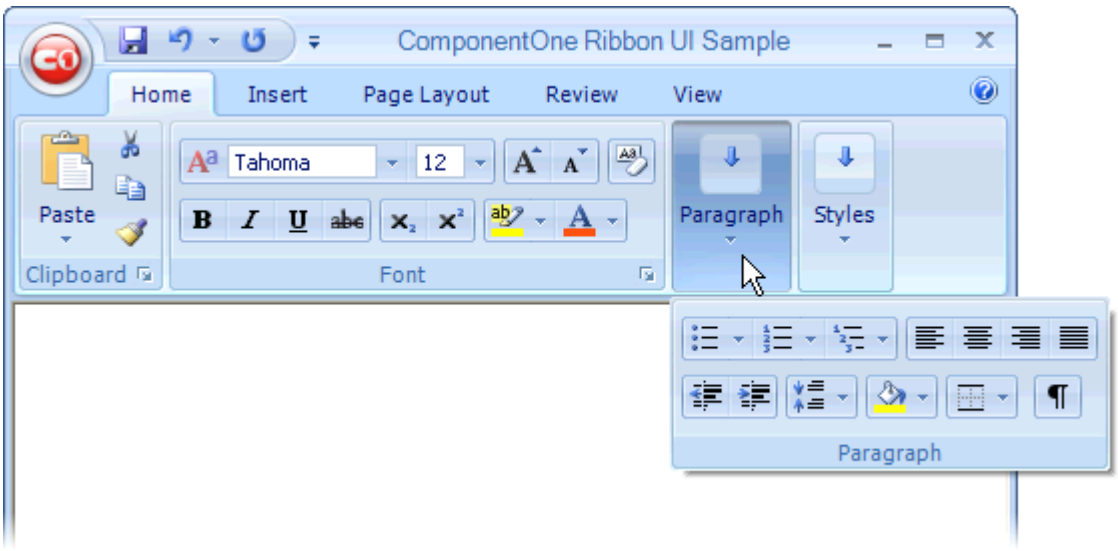


The RibbonGroup can contain a number of Ribbon items, such as buttons, check boxes, combo boxes, toolbars, menus, and so on. For example, the following Font group organizes related font commands and contains numerous Ribbon items:



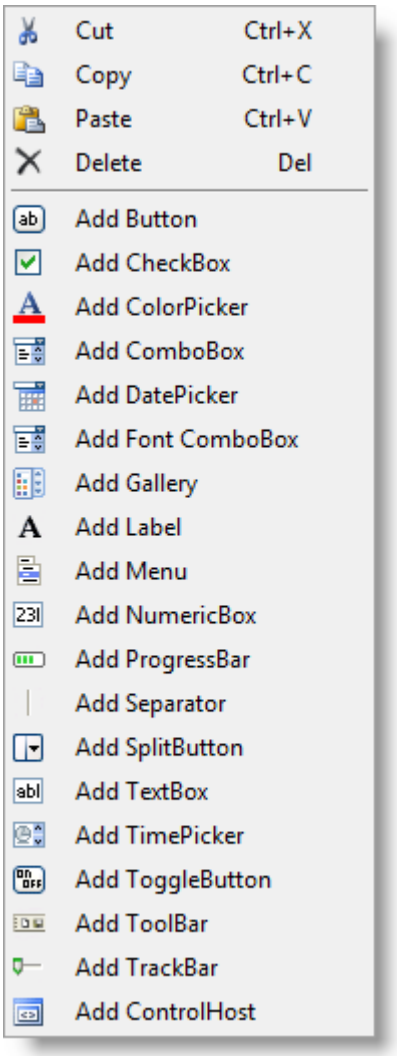
Collapsing Group

Note that when more Ribbon groups are added, the existing Ribbon groups may have to resize to make room on the tab for the new group(s). If resizing is necessary, the large icons in the group are automatically replaced with small icons and text is hidden if space does not allow for the text. If there is not enough room on the tab to display all of the group's items at one time, then the group collapses.



Items Available to Add to the Group

With the design-time support, you can easily add Ribbon items to the RibbonGroup. To see a list of available Ribbon items, click the **Actions** button located on the group floating toolbar and choose from the following list:

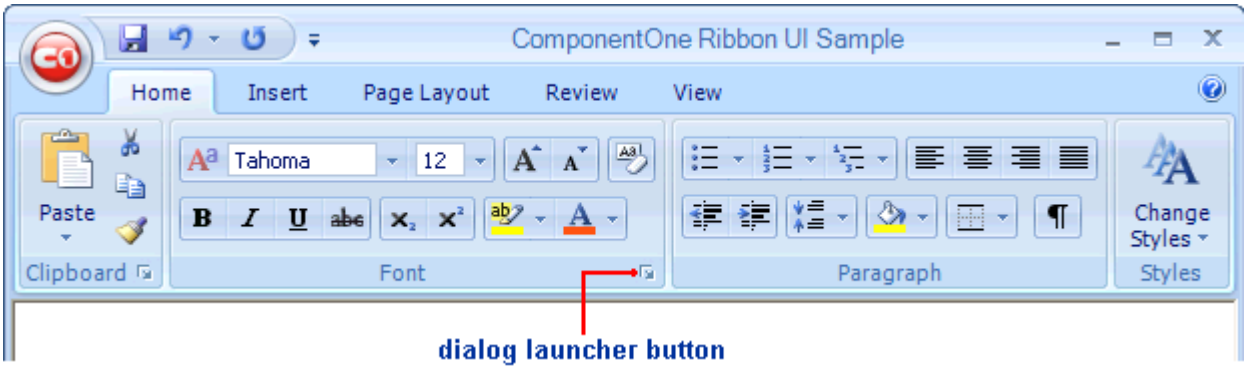


For steps on adding items to the Ribbon group, see the Adding Items to the Ribbon Group topic.

Dialog Launcher Button

Included in the functional group of the Ribbon, the dialog launcher connects the simple functionality of the Ribbon and the advanced functionality of the dialog box. A group can contain only a single dialog launcher button.

The following image shows the Font group's dialog launcher button:



To implement the dialog launcher button:

1. Select the group by clicking it.
2. In the Properties window, click the **Events** button.
3. Double-click the empty area to the right of the **DialogLauncherClick** event. An empty event handler will be added.
4. Enter the event handling code so that the entire event handler looks like this:

To write code in Visual Basic

Visual Basic

```
' type the Imports directive for the namespace
Imports C1.Win.C1Ribbon

Private Sub FontGroup_DialogLauncherClick (ByVal sender As System.Object, ByVal
e As System.EventArgs) Handles FontGroup.DialogLauncherClick
    Dim dlg As FontDialog = New FontDialog
    Dim font As Font = Me.RichTextBox1.SelectionFont
      If Not (font Is Nothing) Then
       dlg.Font = font
     End If
     If (dlg.ShowDialog = System.Windows.Forms.DialogResult.OK) Then
         Me.RichTextBox1.SelectionFont = dlg.Font
     End If
End Sub
```

To write code in C#

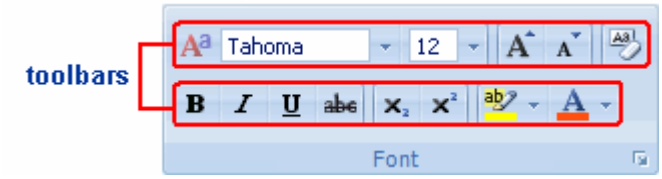
C#

```
// type the using directive for the namespace
using C1.Win.C1Ribbon;

private void FontGroup_DialogLauncherClick (object sender, EventArgs e)
{
    FontDialog dlg = new FontDialog ();
    Font font = this.richTextBox1.SelectionFont;
    if (font != null) dlg.Font = font;
    if (dlg.ShowDialog () != System.Windows.Forms.DialogResult.OK) return;
    this.richTextBox1.SelectionFont = dlg.Font;
}
```

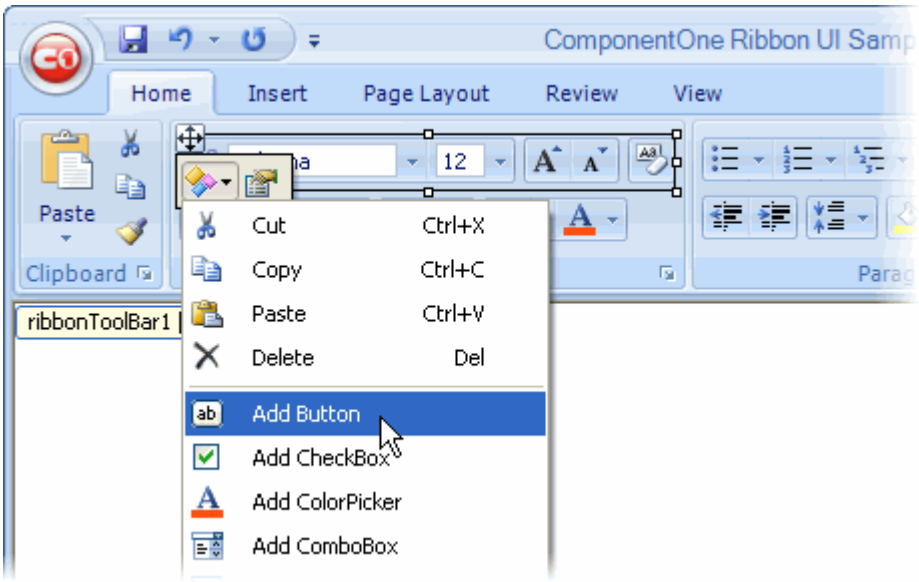
Note that for the above code the RibbonGroup.**Name** property has been set to **FontGroup**.

Ribbon Containers

The C1Ribbon includes one Ribbon toolbar container to hold an organize ribbon items. The RibbonToolBar is a horizontal styled container for other RibbonItem elements. For example, the following Ribbon group has two toolbar items, where each toolbar contains various Ribbon items:



When you add a toolbar to the Ribbon group, a Ribbon button appears in the container. You can quickly add more Ribbon items to the toolbar using the smart designer. Simply click the toolbar to enable its floating toolbar. Next, click the **Actions** button on the item's floating toolbar and select an item. For example, in the following image the **Add Button** is being selected:



You can continue to add more Ribbon items, such as, buttons, check boxes, combo boxes, labels, and so on, to the Ribbon toolbar.

Ribbon Items

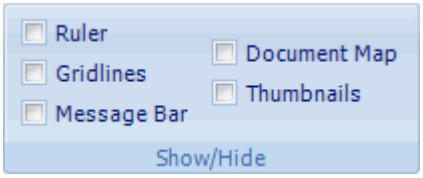
The C1Ribbon containers can hold the following Ribbon items:

Button

The RibbonButton items in this Font group are highlighted below:

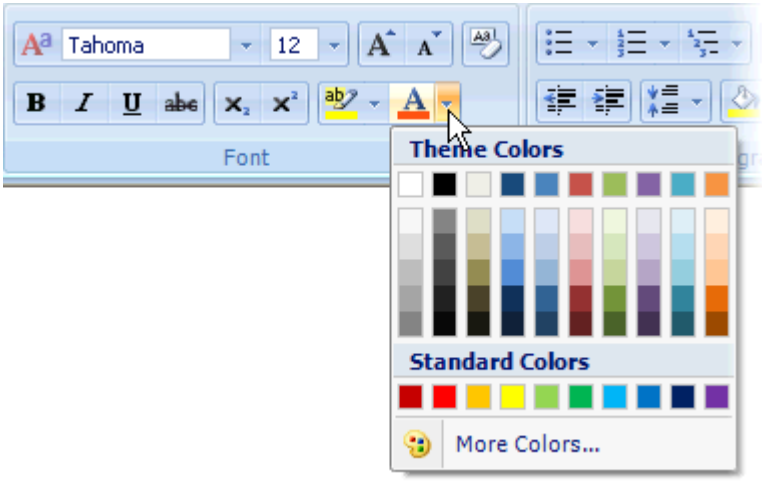
Check Box

The check box control allows the user to validate or invalidate an option. The following Show/Hide group contains numerous RibbonCheckBox elements:



Color Picker

A color picker is a button that shows a drop-down color palette. Use the RibbonColorPicker to select a specific color from a preset palette. Click the color picker's drop-down arrow to reveal the color palette, which contains theme colors, standard colors, and more:



Combo Box

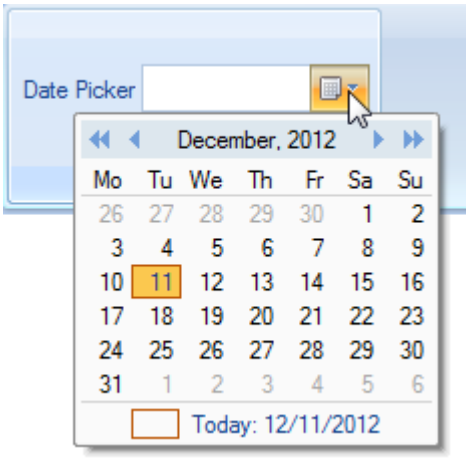
A combo box is a combination of a drop-down list or list box and a single-line text box. The RibbonComboBox item in this Font group is highlighted below:

Control Host

The RibbonControlHost element allows the user to host an arbitrary control in a RibbonGroup. For more information on using the RibbonControlHost, please see the Embedding Controls in a Ribbon topic.

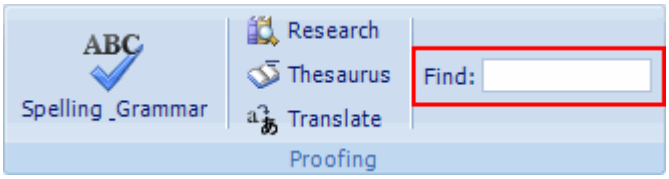
Date Picker

A date picker control allows a user to enter a specific date in the numeric box or to choose a specific date from a drop-down calendar. Click the DatePicker's drop-down arrow to choose a date from the calendar:



Text Box

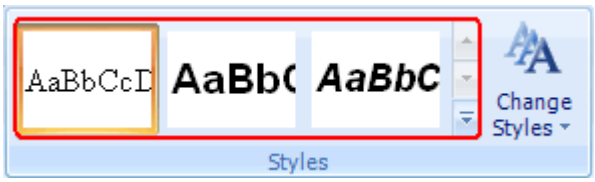
Represents a text box control. The RibbonTextBox element in this Proofing group is highlighted below:



Gallery

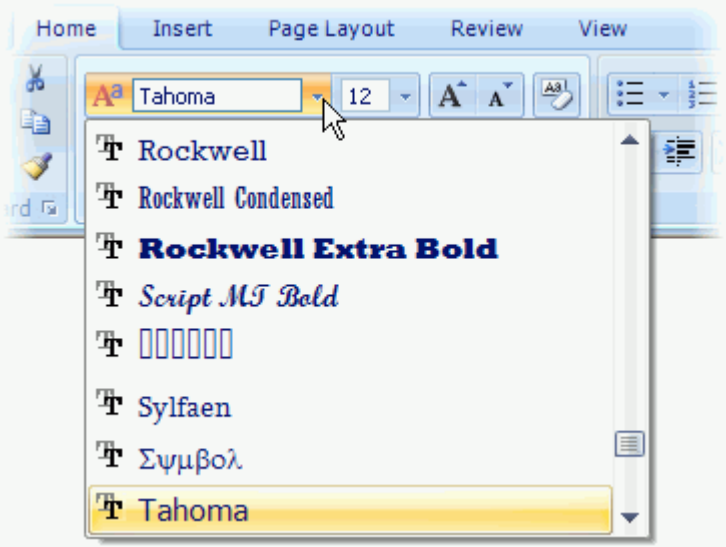
A gallery consists of gallery groups. The RibbonGallery is designed to allow the user to select items visually in a grid or menu-like layout. This results-oriented approach allows users to quickly see the features and make a selection from a group of items. A gallery may be displayed in either collapsed or expanded view.

For example, a user can view a gallery with different styles, preview the results, and choose a design without having to navigate through a number of menus and dialog boxes:



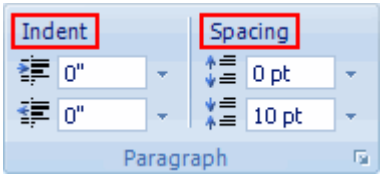
Font Combo Box

A RibbonFontComboBox is a combination of a drop-down list or list box that shows available font styles and a single-line text box. Click the font combo box to reveal the list of available font styles:



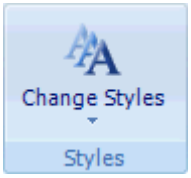
Label

The RibbonLabel elements in this Clipboard group are highlighted below:



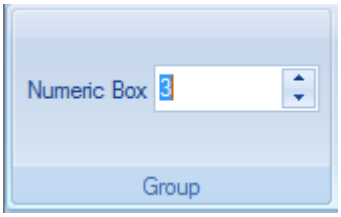
Menu

This is a button that shows a drop-down menu. Use a menu button when you need a menu for a small set of related commands. The following Styles group contains a RibbonMenu item:



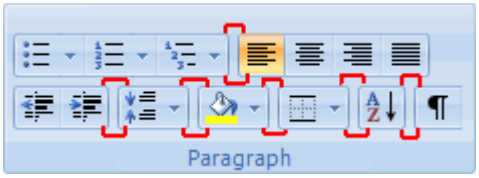
Numeric Box

The numeric box allows the user to to select a number by using the increment and decrement buttons or by entering a number is the numeric box:



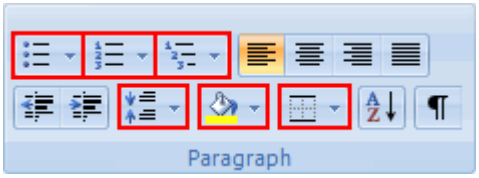
Separator

The separator item can be used in a Ribbon group, toolbar, or drop-down element to visually separate groups of Ribbon items. The RibbonSeparator items in this Paragraph group are highlighted below:



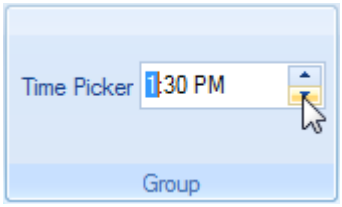
Split Button

A split button combines a regular button and a drop-down list. Use a split button to combine a set of variations of a command, especially when one of the commands is used often. The RibbonSplitButton elements in this Paragraph group are highlighted below:



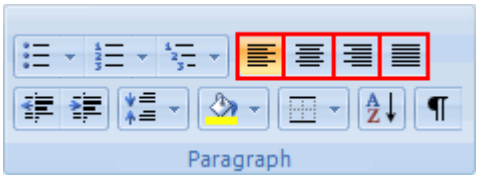
Time Picker

The time picker control allows the user to enter a specific time in the numeric box or to choose a specific time using the increment and decrement buttons.



Toggle Button

A toggle button displays like a command button, but works like a check box. This element can be grouped using the ToggleGroupName property. The RibbonToggleButton elements in this Paragraph group are highlighted below:



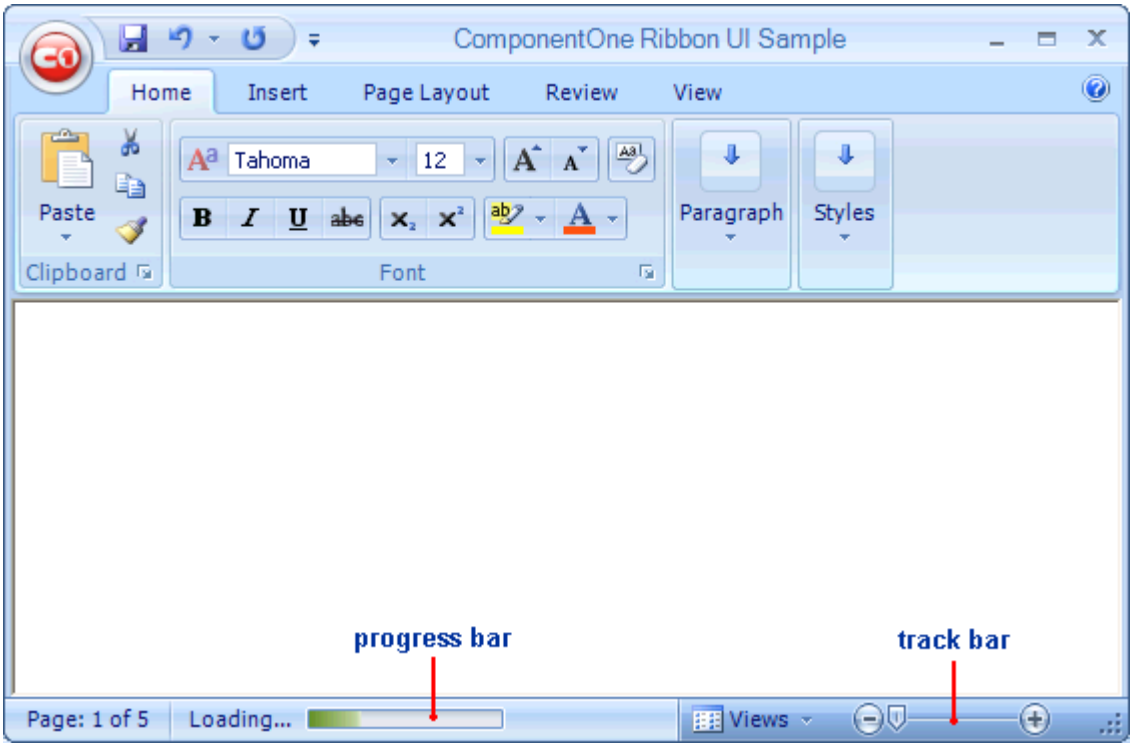
Status Bar Items

The C1StatusBar can hold the following items:

- button
- color picker
- combo box
- control host
- date picker
- font combo box
- gallery
- label
- menu
- numeric box
- separator
- split button
- text box
- time picker
- toggle button
- track bar
- progress bar

For information on each Ribbon item, see Ribbon Items.

The following Ribbon sample form shows a status bar with a progress bar in the left pane and a track bar in the right pane:



Progress Bar

The RibbonProgressBar element displays a bar that fills in from left to right with the system highlight color as an operation progresses.

Track Bar

The RibbonTrackBar functions similar to the standard System.Windows.Forms.TrackBar control and is designed specifically for use in a C1StatusBar. You might use a track bar to allow the user to zoom in and zoom out of a document window.

Design-Time Support

Ribbon for WinForms provides visual editing to make it easier to create the Ribbon. You can make changes to the Ribbon by using one or more of the following visual editors:

Properties Window

You can access the C1Ribbon and C1StatusBar properties simply by right-clicking on the control and selecting Properties or by selecting the class from the drop-down box of the **Properties** window.

Smart Tag

In Visual Studio, the C1Ribbon and C1StatusBar controls include a smart tag. A **smart tag** represents a short-cut tasks menu that provides the most commonly used properties in each control. You can invoke the tasks menu by clicking on the smart tag (▶) in the upper-right corner of each control. For more information on how to use the smart tags, see C1Ribbon and C1StatusBar Smart Tags.

Context Menu

You can use the C1Ribbon and C1StatusBar controls' context menu for additional functionality at design time. See the C1Ribbon and C1StatusBar Context Menus topic for details.

Collection Editors

The main part of each of the collection editor's application consists of a Windows form which conveniently allows you to add Ribbon items and edit their properties. C1Ribbon provides the following collection editors:

- RibbonApplicationMenu BottomPaneItems Collection Editor
- RibbonApplicationMenu LeftPaneItems Collection Editor
- RibbonApplicationMenu RightPaneItems Collection Editor
- QAT Items Collection Editor
- QAT MenuItems Collection Editor
- RibbonConfigToolBar Items Collection Editor
- RibbonTab Collection Editor
- RibbonGroup Collection Editor
- RibbonGroup Items Collection Editor
- RibbonGalleryItem Collection Editor
- RibbonGallery Menu Items Collection Editor
- RibbonToolBar Items Collection Editor
- RibbonMenu Items Collection Editor
- RibbonComboBox Items Collection Editor
- RibbonComboBox Menu Items Collection Editor
- RibbonFontComboBox Menu Items Collection Editor
- RibbonSplitButton Items Collection Editor
- StatusBar LeftPaneItems Collection Editor
- StatusBar RightPaneItems Collection Editor

Smart Designers

The smart designers enable you to have complete control over creating a powerful and enhancing Ribbon and status

bar. For more information on the C1Ribbon and C1StatusBar smart designer, see the C1Ribbon Smart Designer and C1StatusBar Smart Designer topics.

In-Place Text Editing

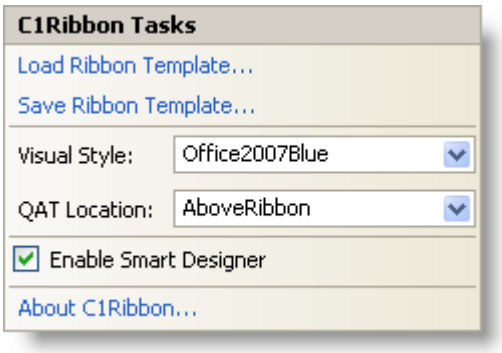
You can quickly edit Ribbon elements' labels using the in-place editing feature. For more information about using the in-place editing feature see the In-Place Text Editing topic.

C1Ribbon and C1StatusBar Smart Tags

The C1Ribbon and C1StatusBar controls each provide quick and easy access to common properties through their smart tag.

C1Ribbon Smart Tag

To access the **C1Ribbon Tasks** menu, click on the smart tag (▶) in the upper-right corner of the C1Ribbon control. This will open the **C1Ribbon Tasks** menu.

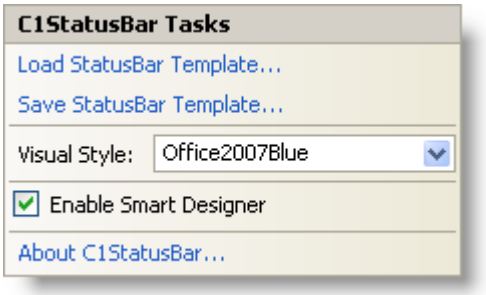


The **C1Ribbon Tasks** menu operates as follows:

- **Load Ribbon Template**
  Clicking on the **Load Ribbon Template** link opens the **Load Ribbon Template** dialog box where you can import an XML file that contains the pre-formatted Ribbon.
- **Save Ribbon Template**
  Clicking on the **Save Ribbon Template** link opens **Save Ribbon Template** dialog box where you can save the Ribbon layout as an XML file.
- **Visual Style**
  Clicking on the **Visual Style** drop-down arrow opens a list of available Office 2007 themes to choose from. The default style is **Office2007Blue**.
- **QAT Location**
  Clicking on the **QAT Location** drop-down arrow allows you to select the location, above or below, of the Quick Access Toolbar in relation to the Ribbon.
- **Enable Smart Designer**
  Deselecting the **Enable Smart Designer** check box turns off the smart designer functionality, and selecting the check box enables the smart designer.
- **About C1Ribbon**
  Clicking on the **About C1Ribbon** item displays a dialog box, which is helpful in finding the version number of C1Ribbon and online resources.

C1StatusBar Smart Tag

To access the **C1StatusBar Tasks** menu, click on the smart tag (▶) in the upper-right corner of the C1StatusBar control. This will open the **C1StatusBar Tasks** menu.



The **C1StatusBar Tasks** menu operates as follows:

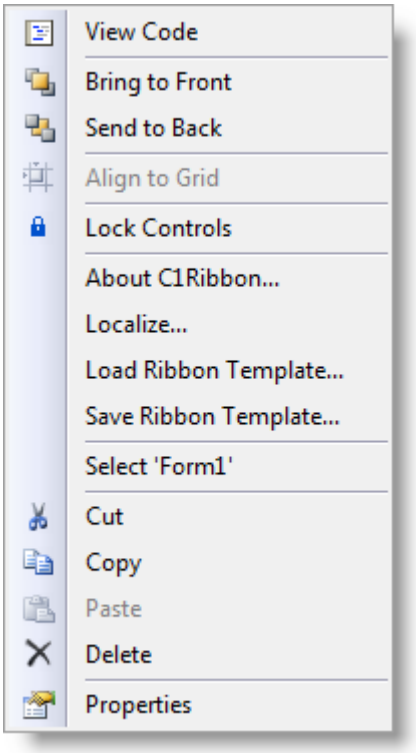
- **Load StatusBar Template**
  Clicking on the **Load StatusBar Template** link opens the **Load StatusBar Template** dialog box where you can import an XML file that contains the pre-formatted status bar.
- **Save StatusBar Template**
  Clicking on the **Save StatusBar Template** link opens **Save StatusBar Template** dialog box where you can save the status bar layout as an XML file.
- **Visual Style**
  Clicking on the **Visual Style** drop-down arrow opens a list of available Office 2007 themes to choose from. The default style is **Office2007Blue**.
- **Enable Smart Designer**
  Deselecting the **Enable Smart Designer** check box turns off the smart designer functionality, and selecting the check box enables the smart designer.
- **About C1StatusBar**
  Clicking on the **About C1StatusBar** item displays a dialog box, which is helpful in finding the version number of C1Ribbon and online resources.

C1Ribbon and C1StatusBar Context Menus

The C1Ribbon and C1StatusBar controls each provide a context menu for additional functionality to use at design time.

C1Ribbon Context Menu

Right-click on the C1Ribbon control to open its context menu.

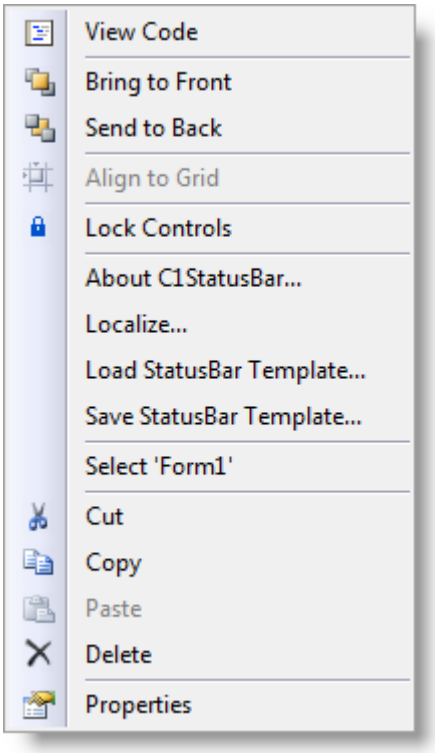


The C1Ribbon context menu operates as follows:

- **About C1Ribbon**
  Clicking on the **About C1Ribbon** item displays a dialog box, which is helpful in finding the version number of C1Ribbon and online resources.
- **Localize**
  Opens the **Localize** dialog box, from where you can add user-defined localization for run-time string resources.
- **Load Ribbon Template**
  Clicking on the **Load Ribbon Template** item opens the **Load Ribbon Template** dialog box where you can import an XML file that contains the pre-formatted Ribbon.
- **Save Ribbon Template**
  Clicking on the **Save Ribbon Template** item opens **Save Ribbon Template** dialog box where you can save the Ribbon layout as an XML file.

C1StatusBar Context Menu

Right-click on the C1StatusBar control to open its context menu.



The **C1StatusBar** context menu operates as follows:

- **About C1StatusBar**
  Clicking on the **About C1StatusBar** item displays a dialog box, which is helpful in finding the version number of C1Ribbon and online resources.
- **Localize**
  Opens the **Localize** dialog box, from where you can add user-defined localization for run-time string resources.
- **Load StatusBar Template**
  Clicking on the **Load StatusBar Template** link opens the **Load StatusBar Template** dialog box where you can import an XML file that contains the pre-formatted status bar.
- **Save StatusBar Template**
  Clicking on the **Save StatusBar Template** link opens **Save StatusBar Template** dialog box where you can save the status bar layout as an XML file.

C1Ribbon Collection Editors

The main part of each of the collection editor's application consists of a Windows form which conveniently allows you to add Ribbon items and edit their properties. C1Ribbon provides the following collection editors:

- Application Menu Collection Editors
- Quick Access Toolbar Collection Editors
- Configuration Toolbar Collection Editor
- Ribbon Tab Collection Editor
- Ribbon Group Collection Editor
- Ribbon Group Items Collection Editor
- Ribbon GalleryItem Collection Editor
- RibbonGallery Menu Items Collection Editor
- Ribbon ToolBar Items Collection Editor
- Ribbon Menu Items Collection Editor
- RibbonComboBox Items Collection Editor
- RibbonComboBox Menu Items Collection Editor

- RibbonFontComboBox Menu Items Collection Editor
- Ribbon SplitButton Items Collection Editor

The following topics briefly introduce the C1Ribbon collection editors and explain how to access each collection editor.

Application Menu Collection Editors

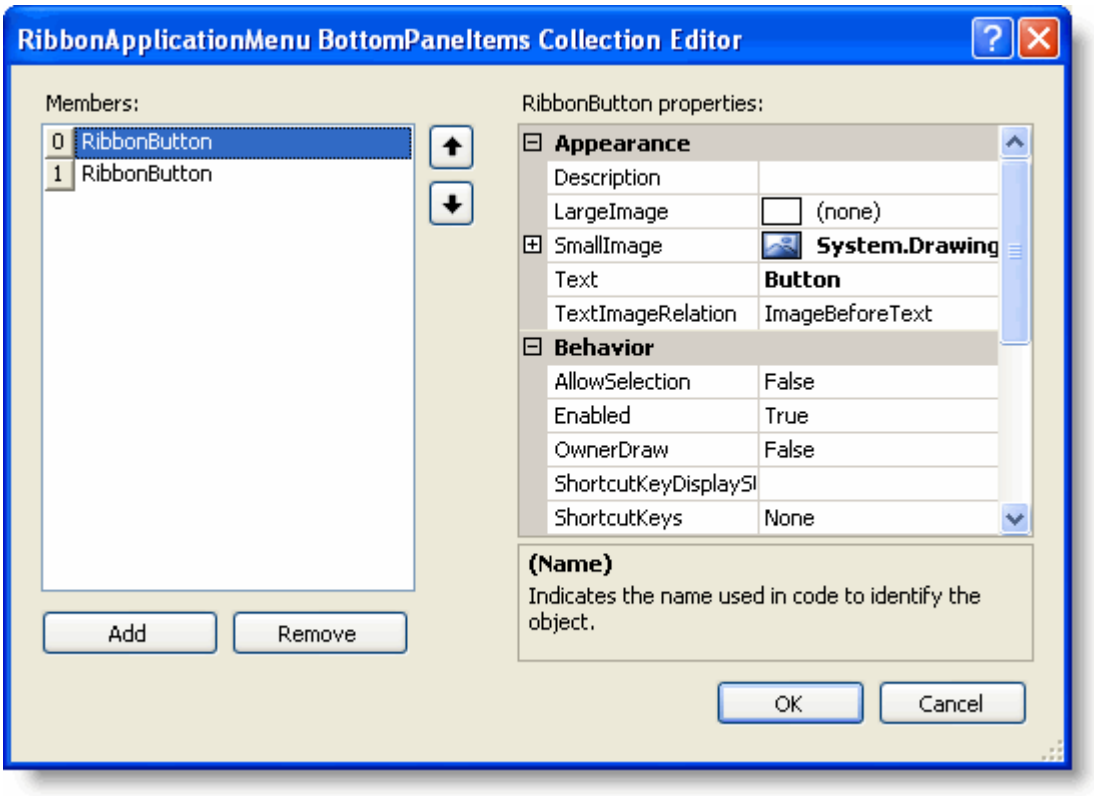
The Application menu consists of three collection editors:

- RibbonApplicationMenu BottomPaneItems Collection Editor
- RibbonApplicationMenu LeftPaneItems Collection Editor
- RibbonApplicationMenu RightPaneItems Collection Editor

The Ribbon application menu's collection editors (BottomPane, LeftPane, and RightPane) allow you to add Ribbon items to the Start menu or remove Ribbon items from the Start menu. Additionally, you can edit the item's properties.

Bottom Pane

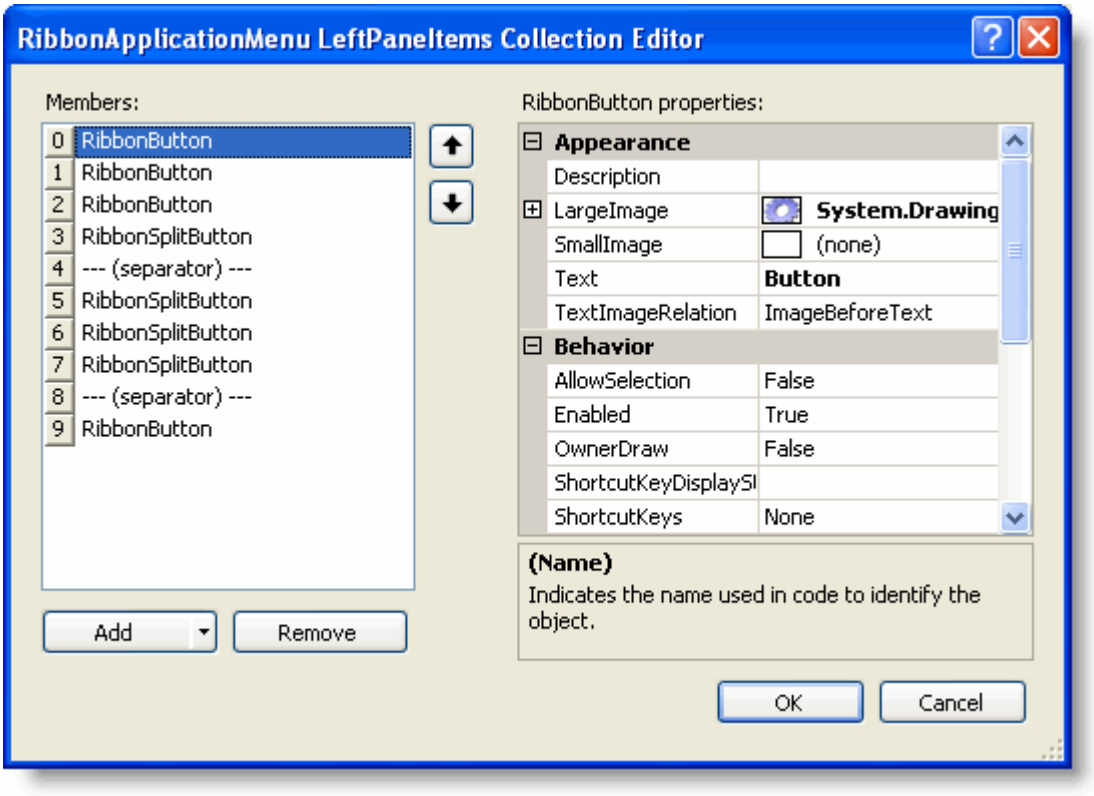
To edit the Application menu's bottom pane, use the **RibbonApplicationMenu BottomPaneItems Collection Editor**:



Clicking the **Add** button will add Ribbon button(s) to the Application menu.

Left Pane

To edit the Application menu's left pane, use the **RibbonApplicationMenu LeftPaneItems Collection Editor**:

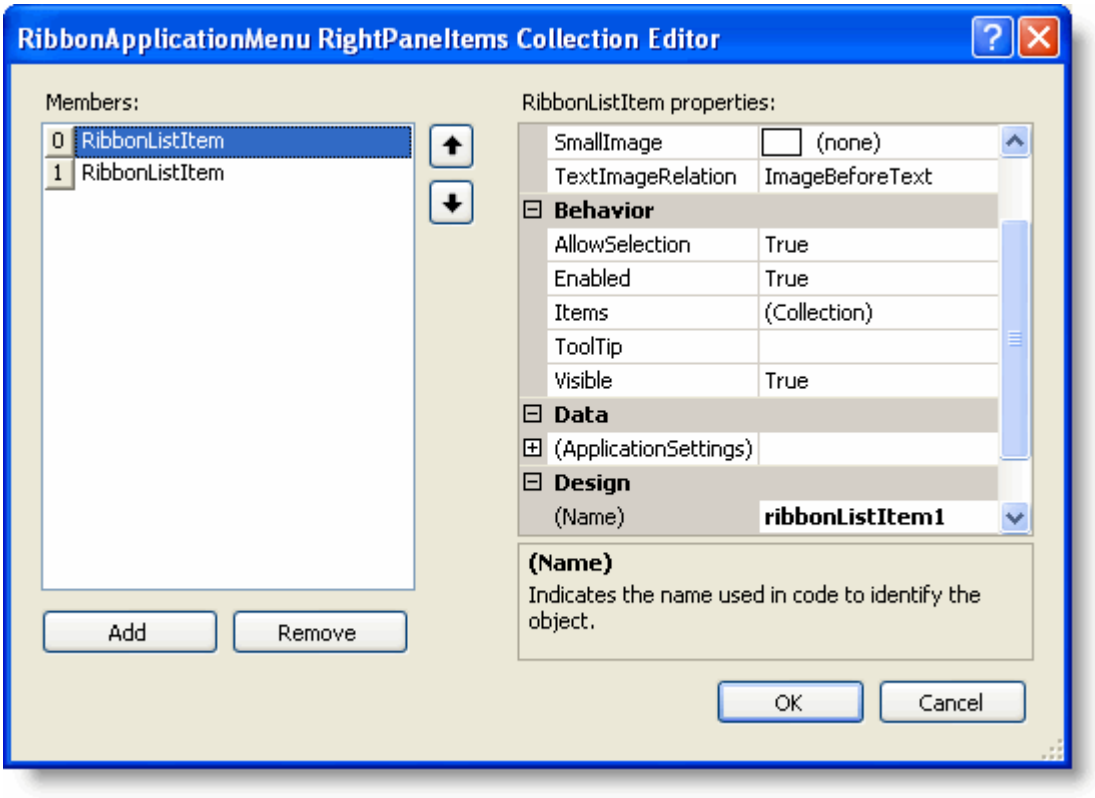


Clicking the **Add** drop-down button will reveal a drop-down list with the following Ribbon items available to add to the Application menu's left pane:

- RibbonButton
- RibbonLabel
- RibbonMenu
- RibbonSeparator
- RibbonSplitButton
- RibbonToggleButton

Right Pane

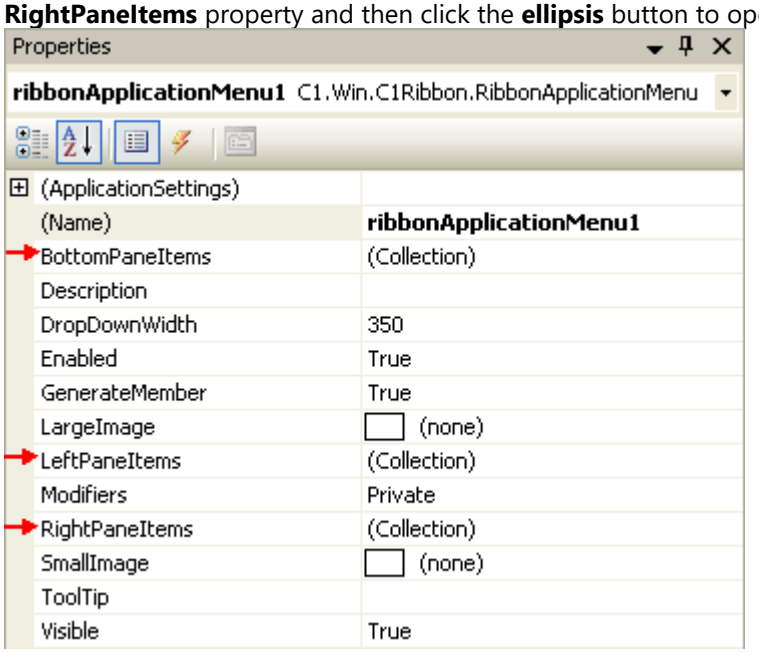
To edit the Application menu's right pane, use the **RibbonApplicationMenu RightPaneItems Collection Editor**:



Clicking the **Add** button will add list item(s) to the Application menu.

To access the Ribbon Application menu's collection editors:

1. Add a C1Ribbon control to the Ribbon Form.
2. Click the application button to activate it.
3. In the Properties window, click on the **(Collection)** next to the **BottomPaneItems**, **LeftPaneItems**, or **RightPaneItems** property and then click the **ellipsis** button to open the collection editor for each menu pane.



Quick Access Toolbar Collection Editors

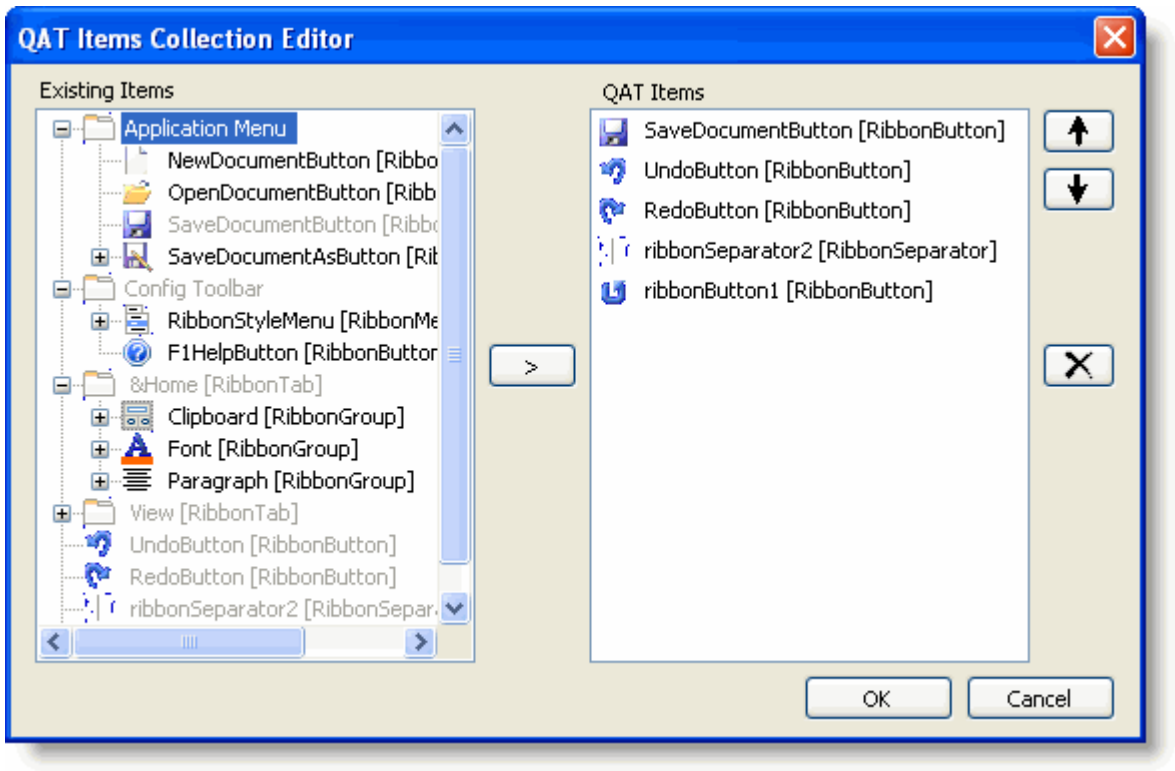
The Quick Access Toolbar (QAT) consists of the following collection editors:

- QAT Items Collection Editor
- QAT MenuItems Collection Editor

The QAT's collection editors allow you to add or remove Ribbon items to and from the QAT or QAT's menu. Additionally, you can edit the item's properties.

QAT Items Collection Editor

To add, remove, or edit the items in the QAT, use the **QAT Items Collection Editor**:



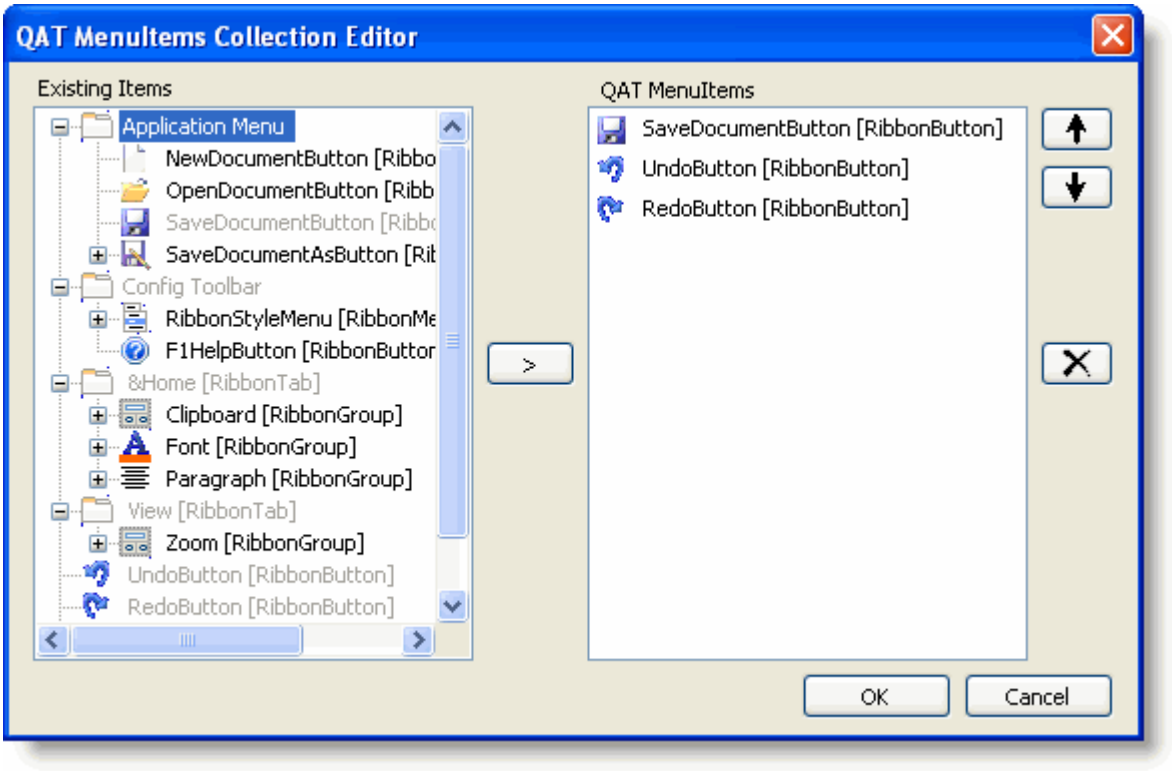
Selecting an item in the **Existing Items** list and clicking the **>** button adds the item to the **QAT Items** list. You can arrange the items in the **QAT Items** list using the up and down arrow buttons.

To access the QAT Items Collection Editor:

1. Add a C1Ribbon control to the Ribbon Form.
2. Click the C1Ribbon control to activate it.
3. In the Properties window, expand the **Qat** property node, click on the **(Collection)** next to the **Items** property, then click the **ellipsis** button.
   The **QAT Items Collection Editor** appears.

QAT MenuItems Collection Editor

To add, remove, or edit the items in the QAT's menu (hot list), use the **QAT MenuItems Collection Editor**:



Selecting an item in the **Existing Items** list and clicking the **>** button adds the item to the **QAT MenuItems** list. You can arrange the items in the **QAT MenuItems** list using the up and down arrow buttons.

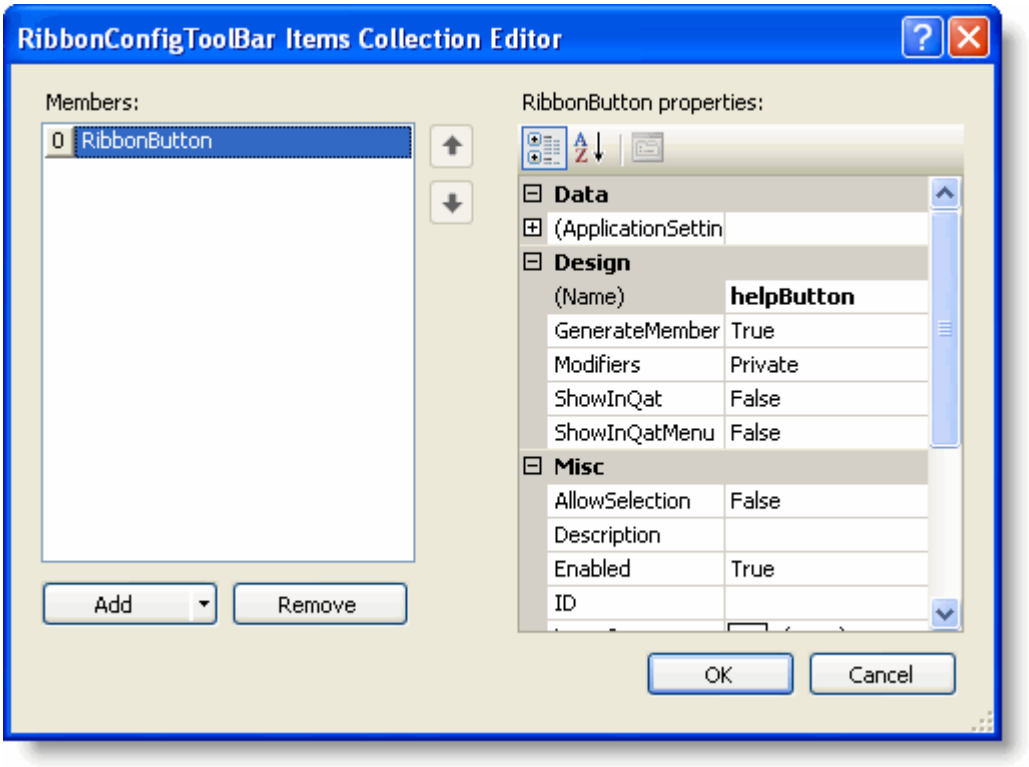
To access the QAT MenuItems Collection Editor:

1. Add a C1Ribbon control to the Ribbon Form.
2. Click the C1Ribbon control to activate it.
3. In the Properties window, expand the **Qat** property node, click on the **(Collection)** next to the **MenuItems** property, then click the **ellipsis** button.
   The **QAT MenuItems Collection Editor** appears.

Configuration Toolbar Collection Editor

The **RibbonConfigToolBar Items Collection Editor** allows you to add Ribbon items to the configuration toolbar or remove Ribbon items from the configuration toolbar. Additionally, you can edit the item's properties.

To edit the configuration toolbar, use the **RibbonConfigToolBar Items Collection Editor**:



Clicking the **Add** drop-down button will reveal a drop-down list with the following Ribbon items available to add to the configuration toolbar:

- RibbonButton
- RibbonCheckBox
- RibbonColorPicker
- RibbonComboBox
- RibbonDatePicker
- RibbonFontComboBox
- RibbonLabel
- RibbonMenu
- RibbonNumericBox
- RibbonProgressBar
- RibbonSeparator
- RibbonSplitButton
- RibbonTextBox
- RibbonTimePicker
- RibbonToggleButton
- RibbonTrackBar
- RibbonControlHost

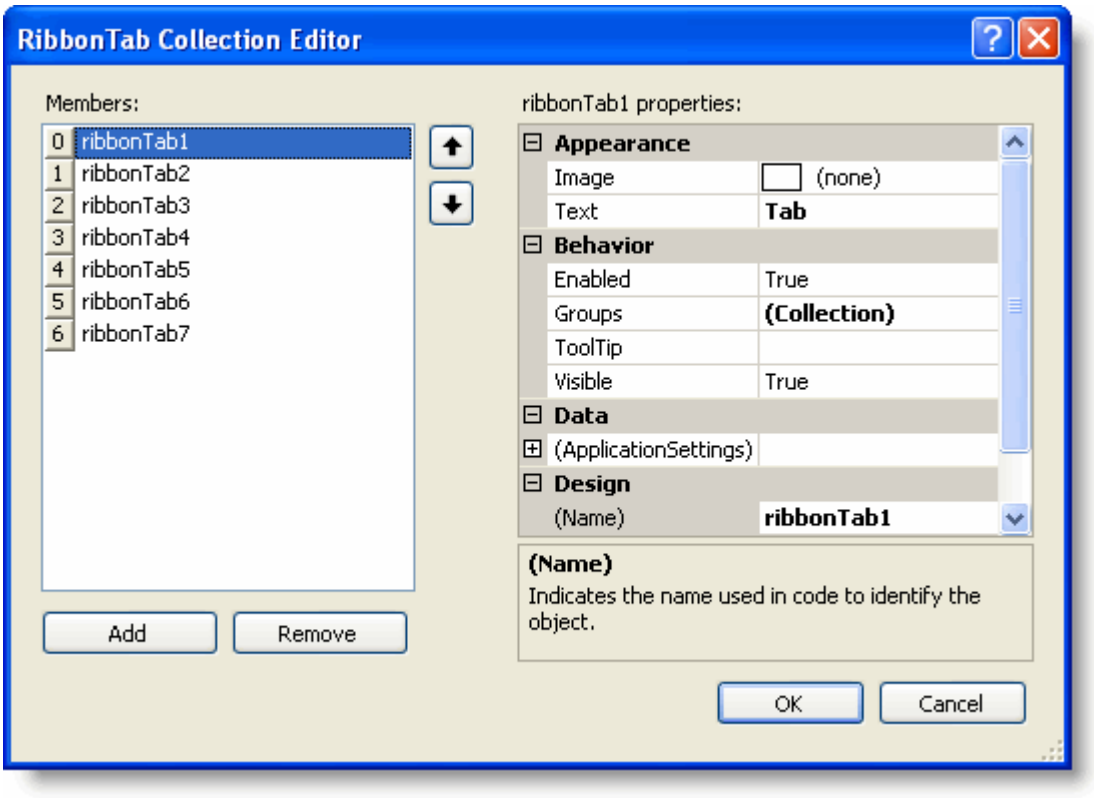
To access the RibbonConfigToolBar Items Collection Editor:

1. Add a C1Ribbon control to the Ribbon Form.
2. Click the C1Ribbon control to activate it.
3. In the Properties window, expand the **ConfigToolBar** property node, click on the **(Collection)** next to the **Items** property, then click the **ellipsis** button.
   The **RibbonConfigToolBar Items Collection Editor** appears.

RibbonTab Collection Editor

The **RibbonTab Collection Editor** allows you to add any number of tabs to the Ribbon or remove tabs from the Ribbon. Additionally, you can edit the tab's properties.

To edit the Ribbon tab(s), use the **RibbonTab Collection Editor**:



Clicking the **Add** button will add tab(s) to the Ribbon.

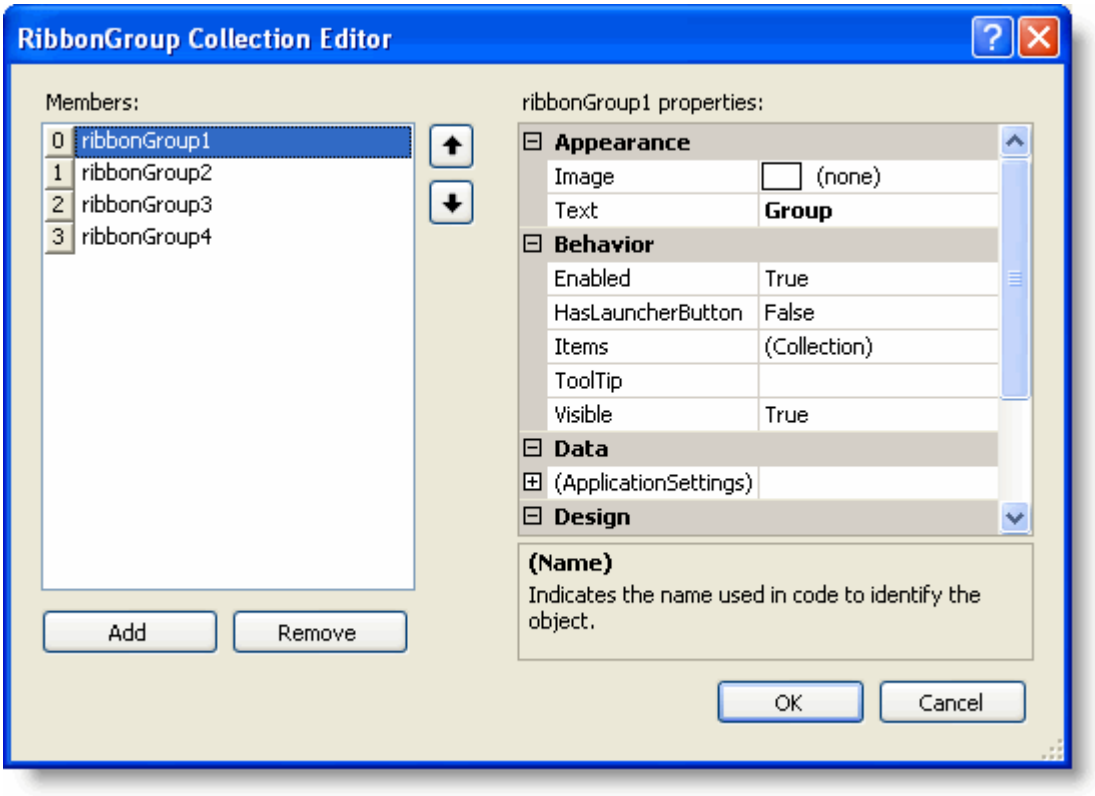
To access the RibbonTab Collection Editor:

1. Add a C1Ribbon control to the Ribbon Form.
2. Click the C1Ribbon control to activate it.
3. In the Properties window, click on the **(Collection)** next to the Tabs property and then click the **ellipsis** button. The **RibbonTab Collection Editor** appears.

RibbonGroup Collection Editor

The **RibbonGroup Collection Editor** allows you to add any number of groups to the tab or remove groups from the tab. Additionally, you can edit the group's properties.

To edit the Ribbon group(s), use the **RibbonGroup Collection Editor**:



Clicking the **Add** button will add group(s) to the Ribbon.

Accessing the RibbonGroup Collection Editor

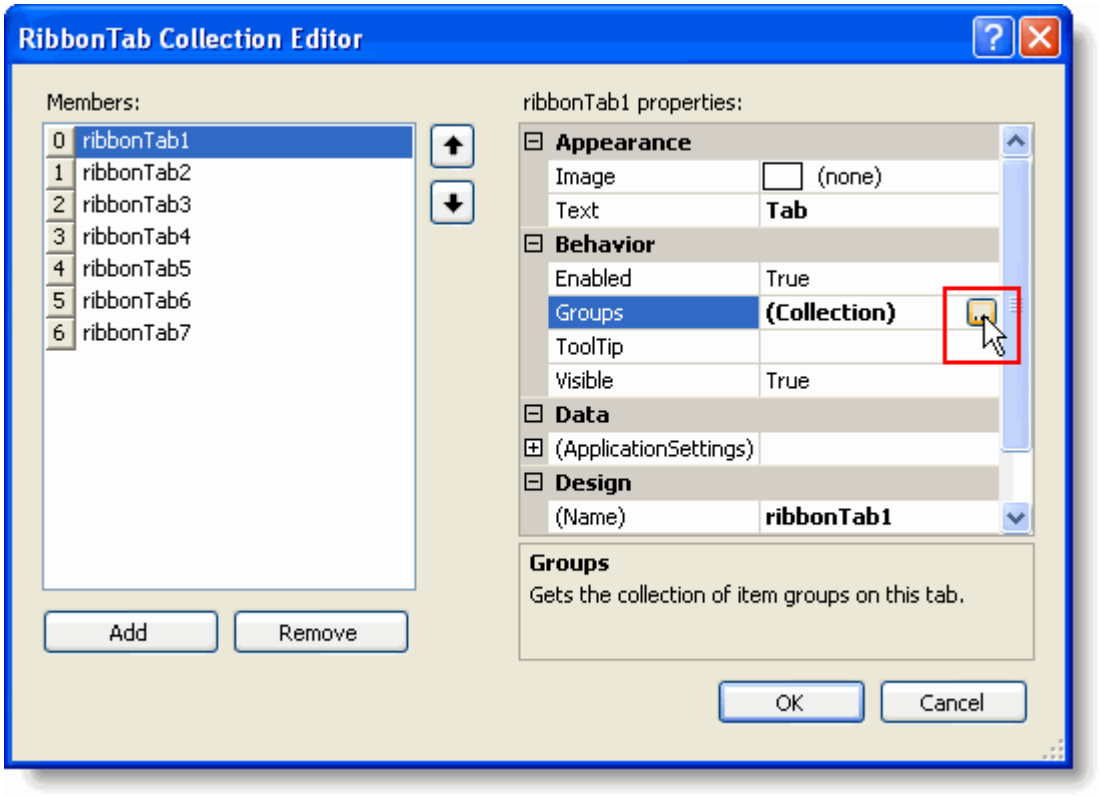
You can use one of the following two options to access the **RibbonGroup Collection Editor**:

Option 1

1. Add a C1Ribbon control to the Ribbon Form.
2. Click the RibbonTab element to activate it.
3. In the Properties window, click on the **ellipsis** button next to the **Groups** property.
   The **RibbonGroup Collection Editor** appears.

Option 2

1. Access the RibbonTab Collection Editor.
2. From the **RibbonTab Collection Editor**'s Properties window, click on the **ellipsis** button next to the **Groups** property.

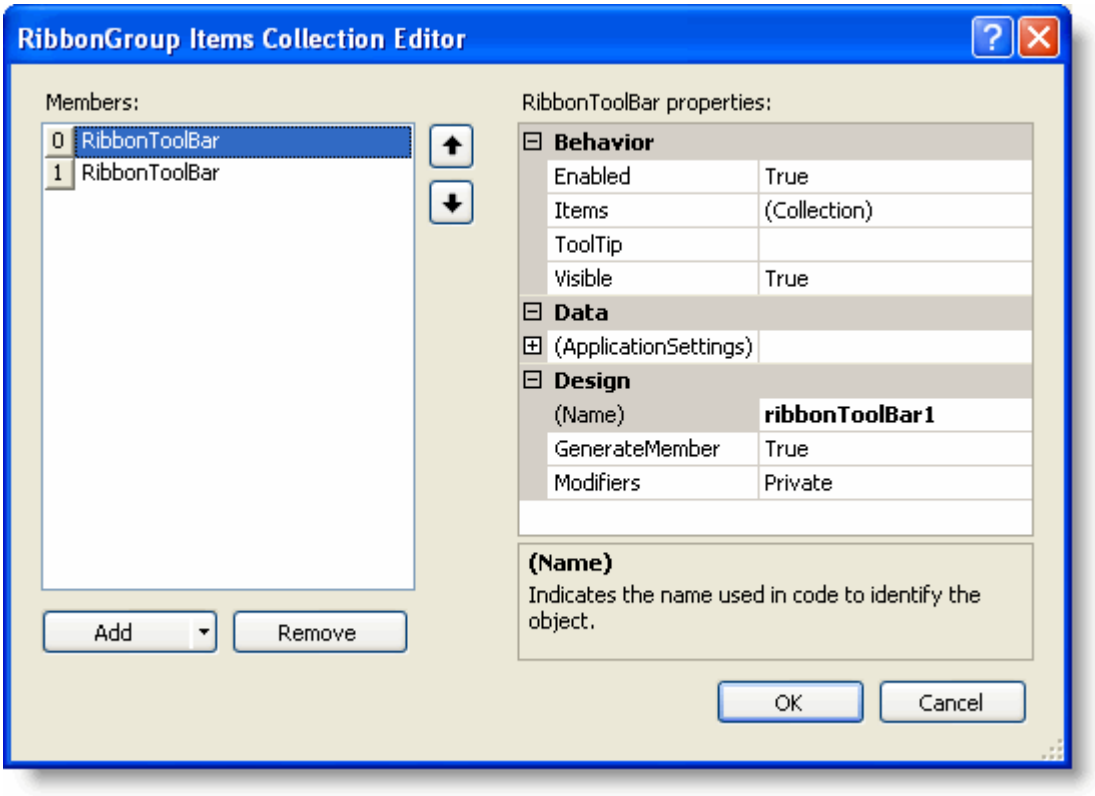


The **RibbonGroup Collection Editor** appears.

RibbonGroup Items Collection Editor

The **RibbonGroup Items Collection Editor** allows you to add any number of Ribbon items to the group or remove items from the group. Additionally, you can edit the item's properties.

To edit the Ribbon item(s), use the **RibbonGroup Items Collection Editor**:



Clicking the **Add** drop-down button will reveal a drop-down list with the following Ribbon items available to add to the group:

- RibbonButton
- RibbonCheckBox
- RibbonColorPicker
- RibbonComboBox
- RibbonDatePicker
- RibbonGallery
- RibbonFontComboBox
- RibbonLabel
- RibbonMenu
- RibbonNumericBox
- RibbonProgressBar
- RibbonSeparator
- RibbonSplitButton
- RibbonTextBox
- RibbonTimePicker
- RibbonToggleButton
- RibbonToolBar
- RibbonTrackBar
- RibbonControlHost

Accessing the RibbonGroup Items Collection Editor

You can use one of the following two options to access the **RibbonGroup Items Collection Editor**:

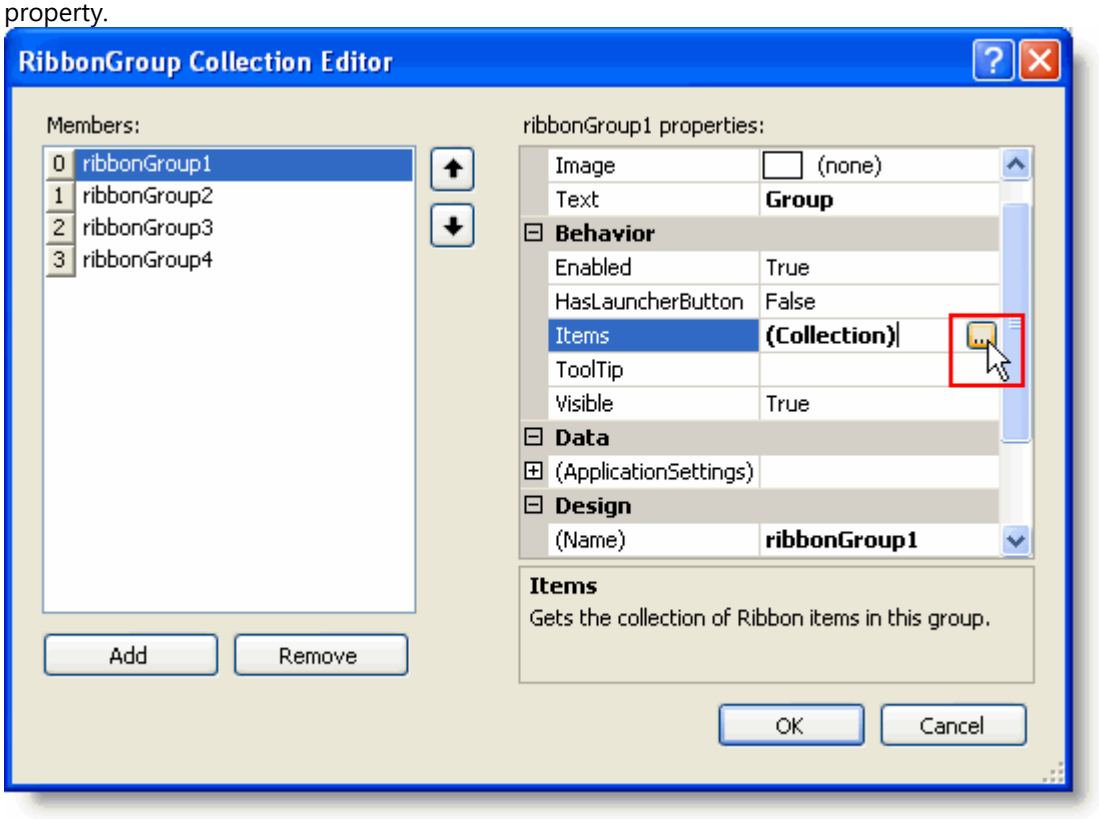
Option 1

1. Add a C1Ribbon control to the Ribbon Form.

2. Click the RibbonGroup element to activate it.
3. In the group's Properties window, click on the **ellipsis** button next to the **Items** property.
   The **RibbonGroup Items Collection Editor** appears.

Option 2

1. Access the RibbonGroup Collection Editor.
2. From the **RibbonGroup Collection Editor**'s Properties window, click on the **ellipsis** button next to the **Items** property.

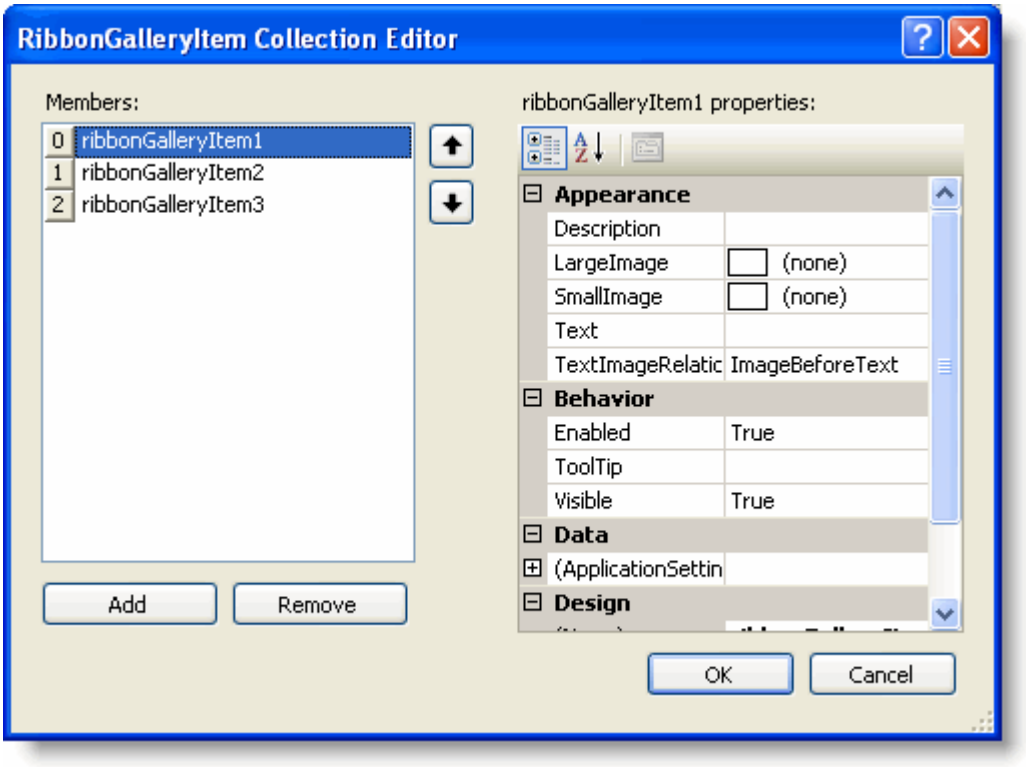


The **RibbonGroup Items Collection Editor** appears.

RibbonGalleryItem Collection Editor

The **RibbonGalleryItem Collection Editor** allows you to add any number of items to the gallery or remove items from the gallery. Additionally, you can edit the item's properties.

To edit the Ribbon gallery item(s), use the **RibbonGalleryItem Collection Editor**:



Clicking the **Add** button will add gallery item(s) to the Ribbon.

Accessing the RibbonGalleryItem Collection Editor

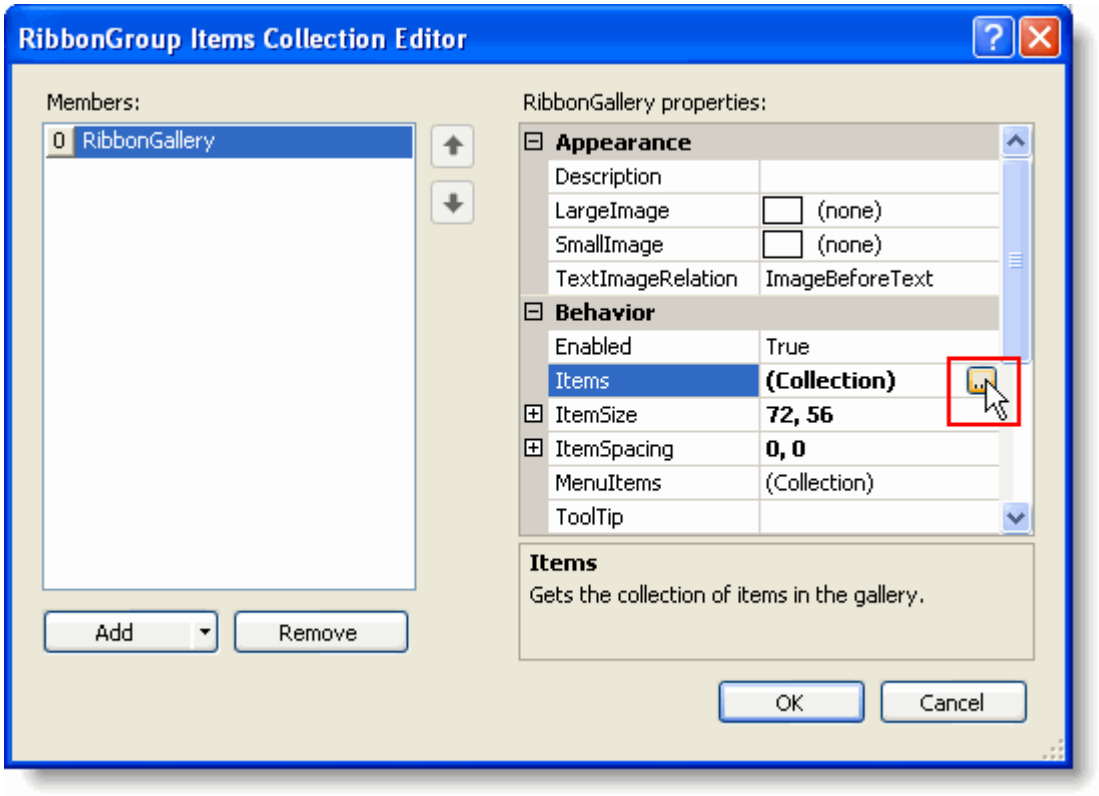
You can use one of the following two options to access the **RibbonGalleryItem Collection Editor**:

Option 1

1. Add a RibbonGallery to the Ribbon group.
2. Click the gallery element to activate it.
3. In the Properties window, click on the **ellipsis** button next to the **Items** property.
   The **RibbonGalleryItem Collection Editor** appears.

Option 2

1. Access the RibbonGroup Items Collection Editor.
2. Click the **Add** drop-down button and add a gallery item.
3. With the RibbonGallery selected in the **Members** list, click on the **ellipsis** button next to the Items property in the Properties window.

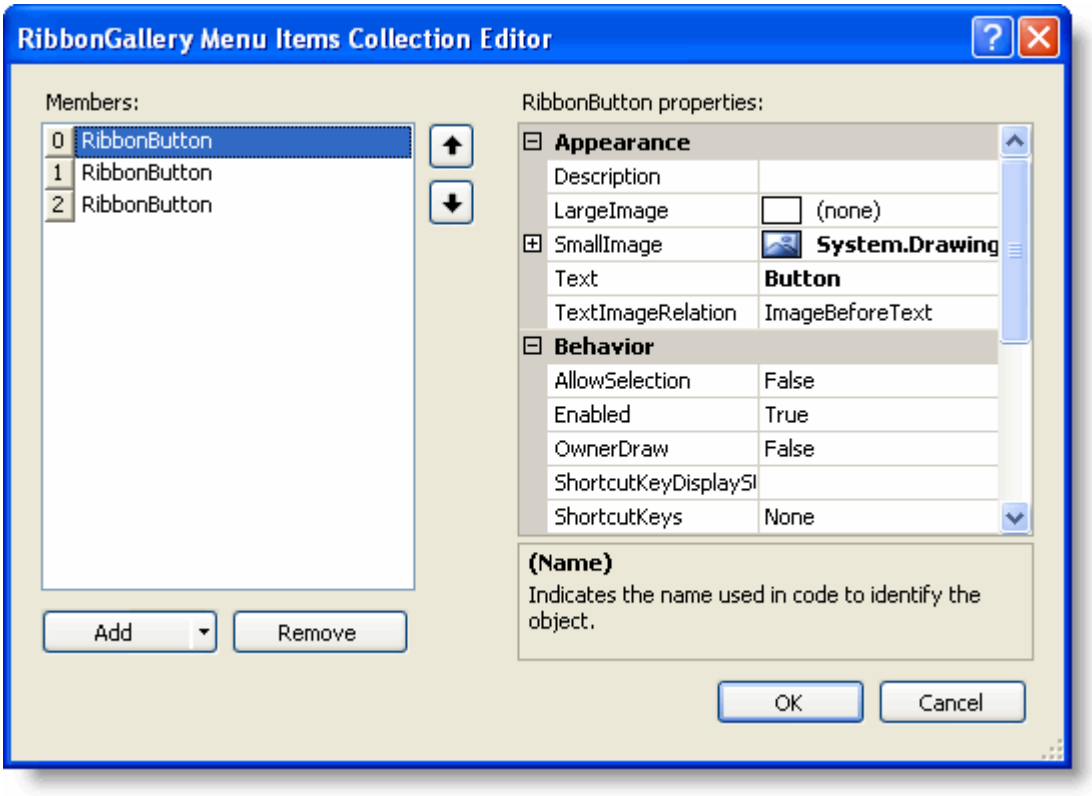


The **RibbonGalleryItem Collection Editor** appears.

RibbonGallery Menu Items Collection Editor

The **RibbonGallery Menu Items Collection Editor** allows you to add any number of menu items to the bottom of the drop-down portion of the gallery or remove menu items from the bottom of the drop-down portion of the gallery. Additionally, you can edit the item's properties.

To edit the Ribbon gallery menu item(s), use the **RibbonGallery Menu Items Collection Editor**:



Clicking the **Add** drop-down button will reveal a drop-down list with the following Ribbon items available to add to the drop-down portion of the gallery:

- RibbonButton
- RibbonColorPicker
- RibbonColorPickerItem
- RibbonComboBox
- RibbonLabel
- RibbonMenu
- RibbonSeparator
- RibbonSplitButton
- RibbonToggleButton

Accessing the RibbonGallery Menu Items Collection Editor

You can use one of the following two options to access the **RibbonGallery Menu Items Collection Editor**:

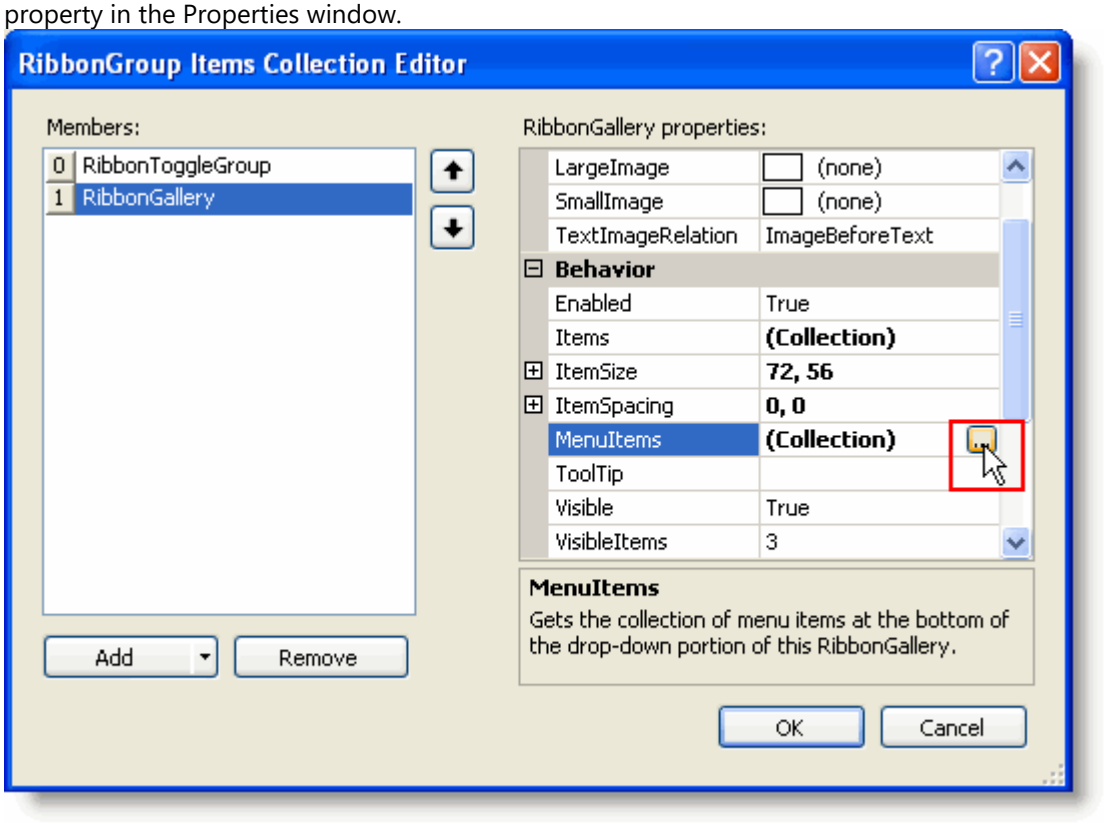
Option 1

1. Add a RibbonGallery to the Ribbon group.
2. Click the gallery element to activate it.
3. In the Properties window, click on the **ellipsis** button next to the MenuItems property.
   The **RibbonGallery Menu Items Collection Editor** appears.

Option 2

1. Access the RibbonGroup Items Collection Editor.
2. Click the **Add** drop-down button and add a gallery item.
3. With the RibbonGallery selected in the **Members** list, click on the **ellipsis** button next to the MenuItems

property in the Properties window.

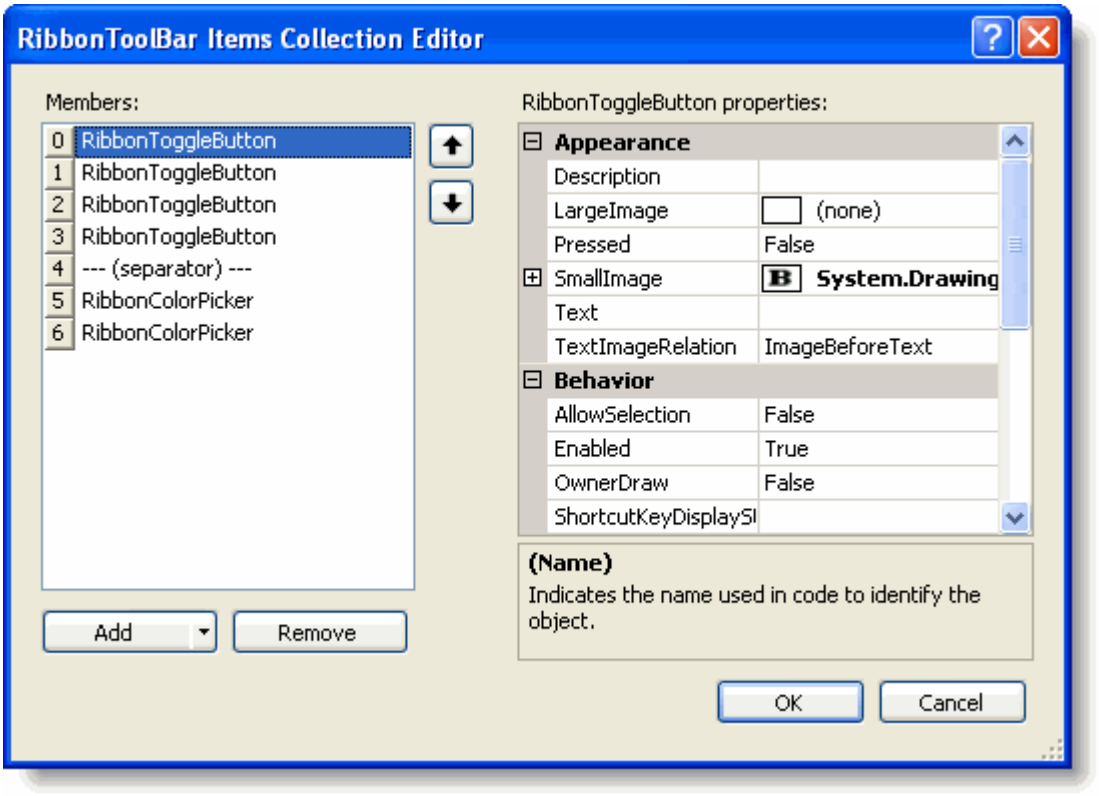


The **RibbonGallery Menu Items Collection Editor** appears.

RibbonToolBar Items Collection Editor

The **RibbonToolBar Items Collection Editor** allows you to add any number of Ribbon items to the toolbar or remove items from the toolbar. Additionally, you can edit the item's properties.

To edit the Ribbon item(s), use the **RibbonToolBar Items Collection Editor**:



Clicking the **Add** drop-down button will reveal a drop-down list with the following Ribbon items available to add to the toolbar:

- RibbonButton
- RibbonCheckBox
- RibbonColorPicker
- RibbonComboBox
- RibbonFontComboBox
- RibbonLabel
- RibbonMenu
- RibbonSeparator
- RibbonSplitButton
- RibbonTextBox
- RibbonToggleButton

Accessing the RibbonToolBar Items Collection Editor

You can use one of the following two options to access the **RibbonToolBar Items Collection Editor**:

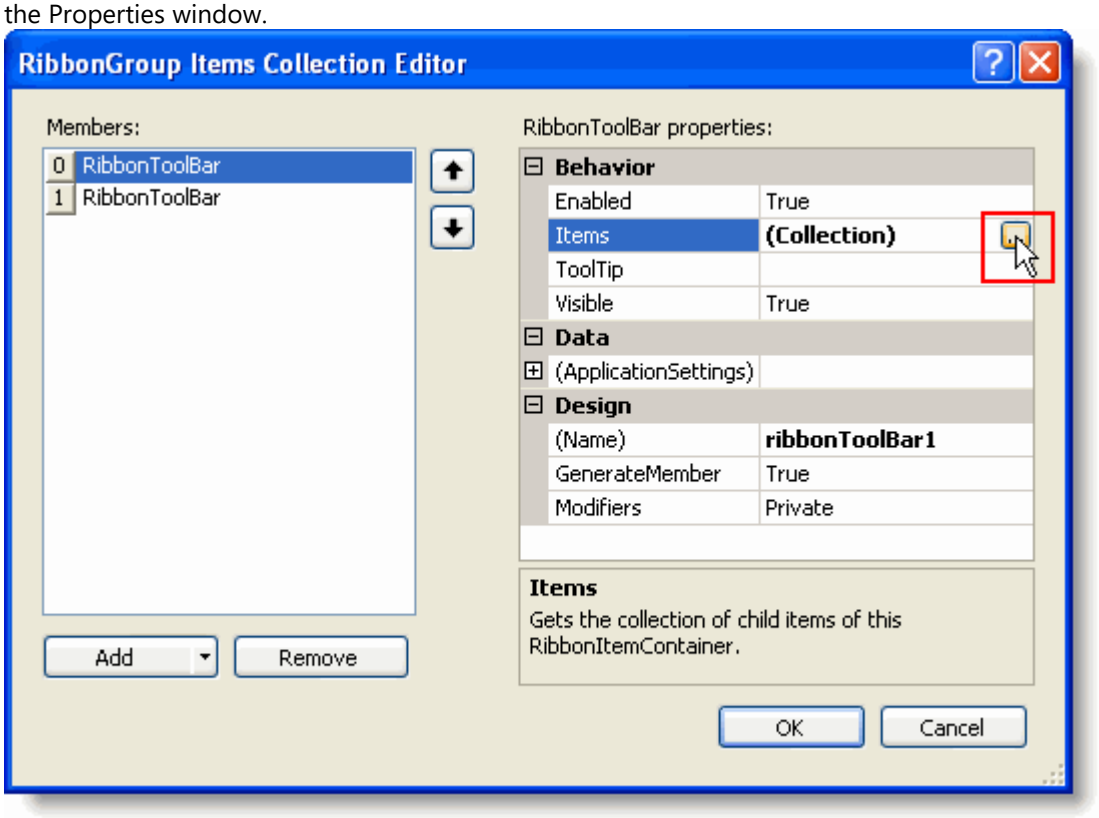
Option 1

1. Add a RibbonToolBar to the Ribbon group.
2. Click the toolbar element to activate it.
3. In the Properties window, click on the **ellipsis** button next to the **Items** property.
   The **RibbonToolBar Items Collection Editor** appears.

Option 2

1. Access the RibbonGroup Items Collection Editor.

2. Click the **Add** drop-down button and add a toolbar item.
3. With the RibbonToolBar selected in the **Members** list, click on the **ellipsis** button next to the **Items** property in the Properties window.

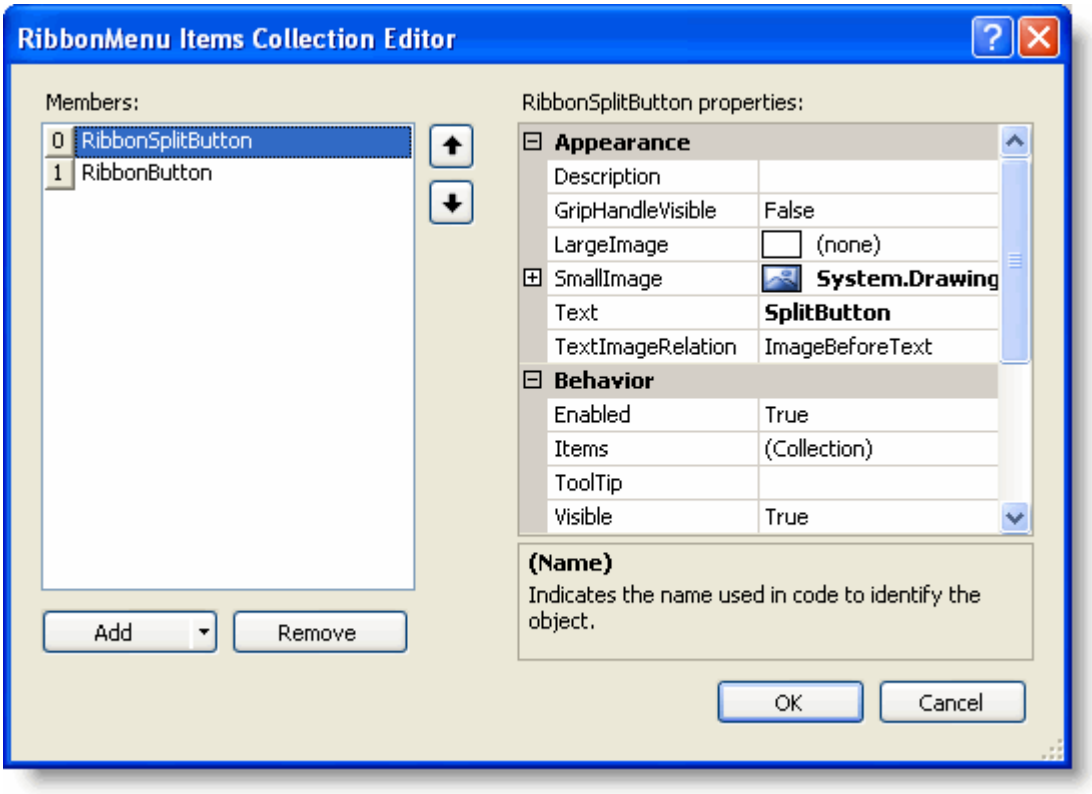


The **RibbonToolBar Items Collection Editor** appears.

RibbonMenu Items Collection Editor

The **RibbonMenu Items Collection Editor** allows you to add any number of Ribbon items to the menu or remove items from the menu. Additionally, you can edit the item's properties.

To edit the Ribbon item(s), use the **RibbonMenu Items Collection Editor**:



Clicking the **Add** drop-down button will reveal a drop-down list with the following Ribbon items available to add to the menu:

- RibbonButton
- RibbonColorPicker
- RibbonColorPickerItem
- RibbonComboBox
- RibbonLabel
- RibbonMenu
- RibbonSeparator
- RibbonSplitButton
- RibbonToggleButton

Accessing the RibbonMenu Items Collection Editor

You can use one of the following two options to access **RibbonMenu Items Collection Editor**:

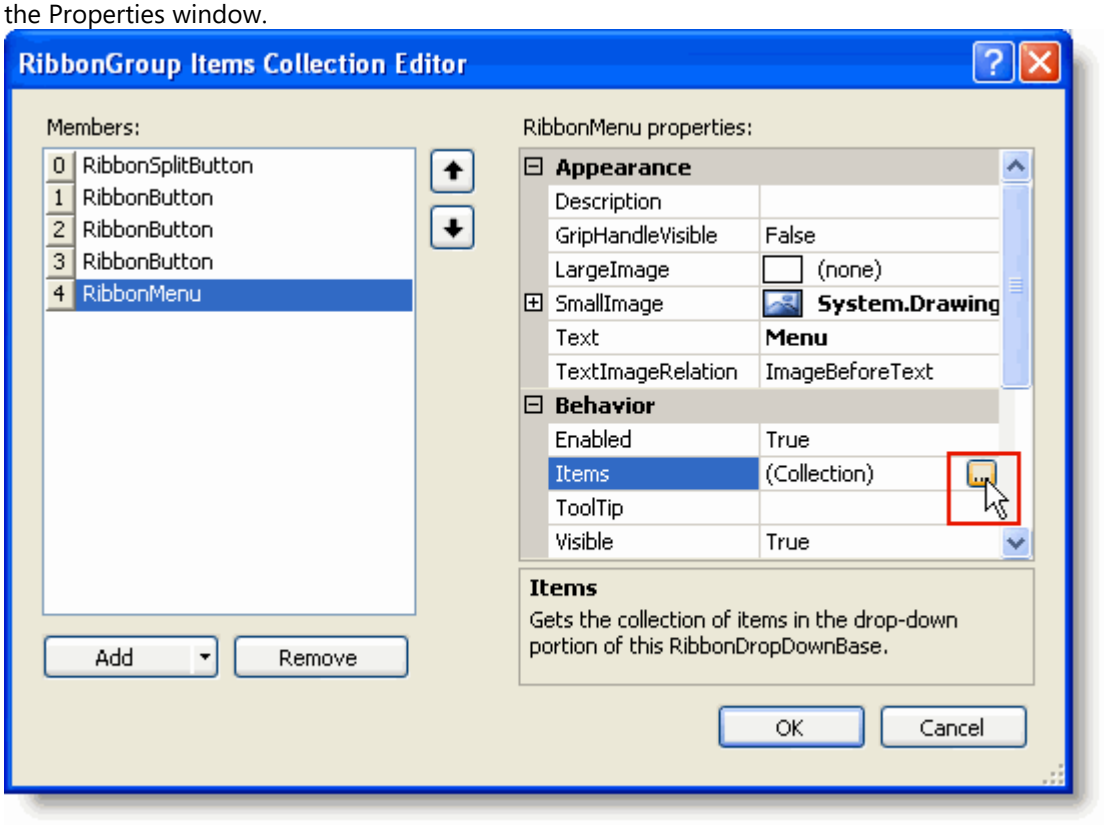
Option 1

1. Add a RibbonMenu to the Ribbon group.
2. Click the menu element to activate it.
3. In the Properties window, click on the **ellipsis** button next to the **Items** property.
   The **RibbonMenu Items Collection Editor** appears.

Option 2

1. Access the RibbonGroup Items Collection Editor.
2. Click the **Add** drop-down button and add a menu item.
3. With the RibbonMenu selected in the **Members** list, click on the **ellipsis** button next to the **Items** property in

the Properties window.

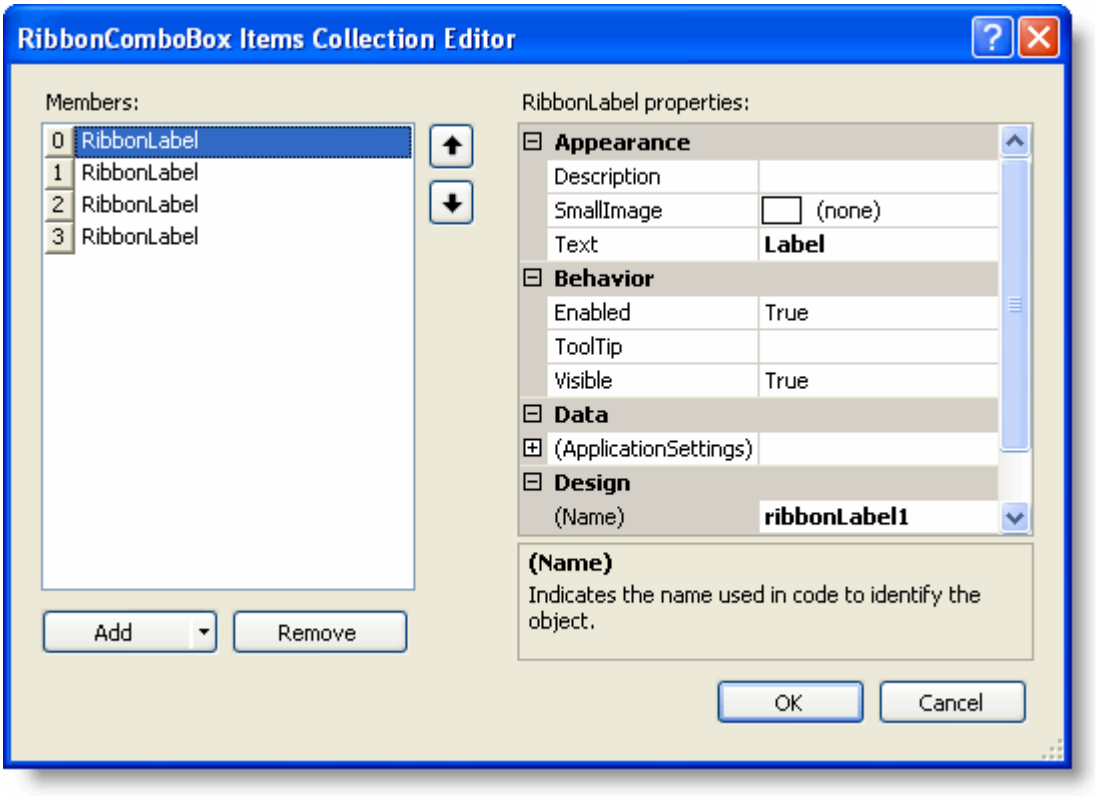


The **RibbonMenu Items Collection Editor** appears.

RibbonComboBox Items Collection Editor

The **RibbonComboBox Items Collection Editor** allows you to add any number of Ribbon items to the combo box or remove items from the combo box. Additionally, you can edit the item's properties.

To edit the Ribbon item(s), use the **RibbonComboBox Items Collection Editor**:



Clicking the **Add** drop-down button will reveal a drop-down list with the following Ribbon items available to add to the combo box:

- RibbonButton
- RibbonLabel

Accessing the RibbonComboBox Items Collection Editor

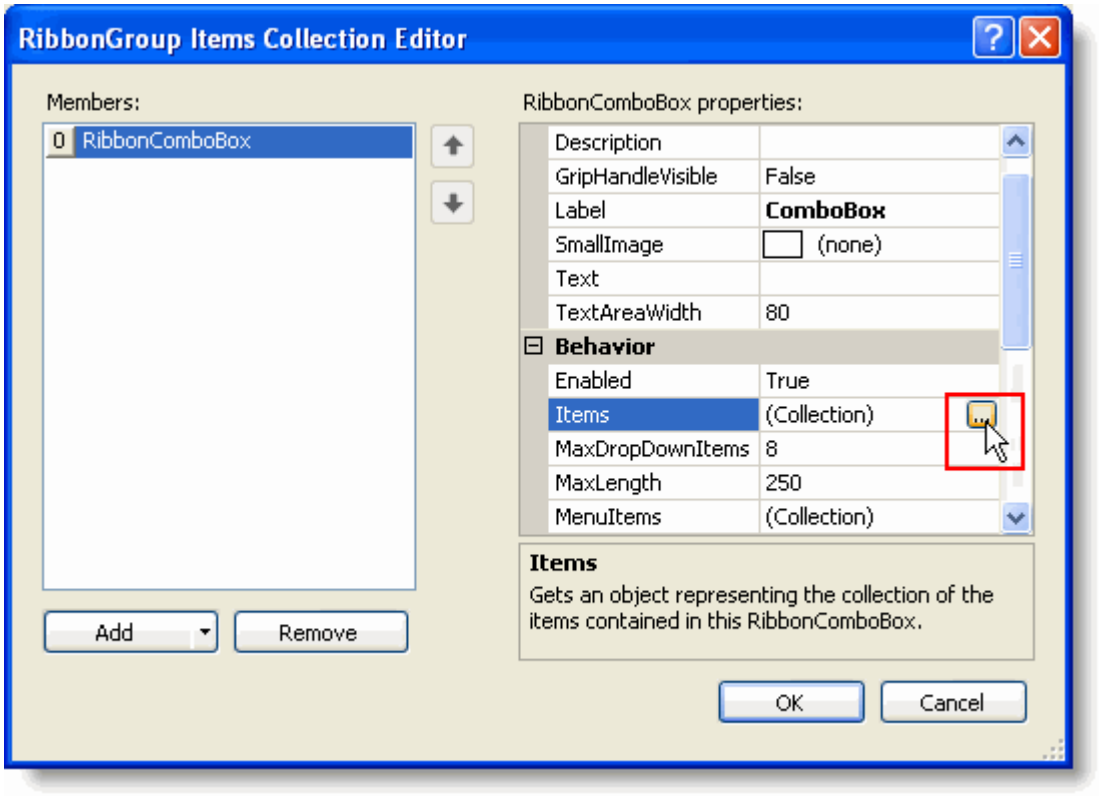
You can use one of the following two options to access **RibbonComboBox Items Collection Editor**:

Option 1

1. Add a RibbonComboBox to the Ribbon group.
2. Click the combo box element to activate it.
3. In the Properties window, click on the **ellipsis** button next to the Items property.
   The **RibbonComboBox Items Collection Editor** appears.

Option 2

1. Access the RibbonGroup Items Collection Editor.
2. Click the **Add** drop-down button and add a combo box item.
3. With the RibbonComboBox selected in the **Members** list, click on the **ellipsis** button next to the Items property in the Properties window.

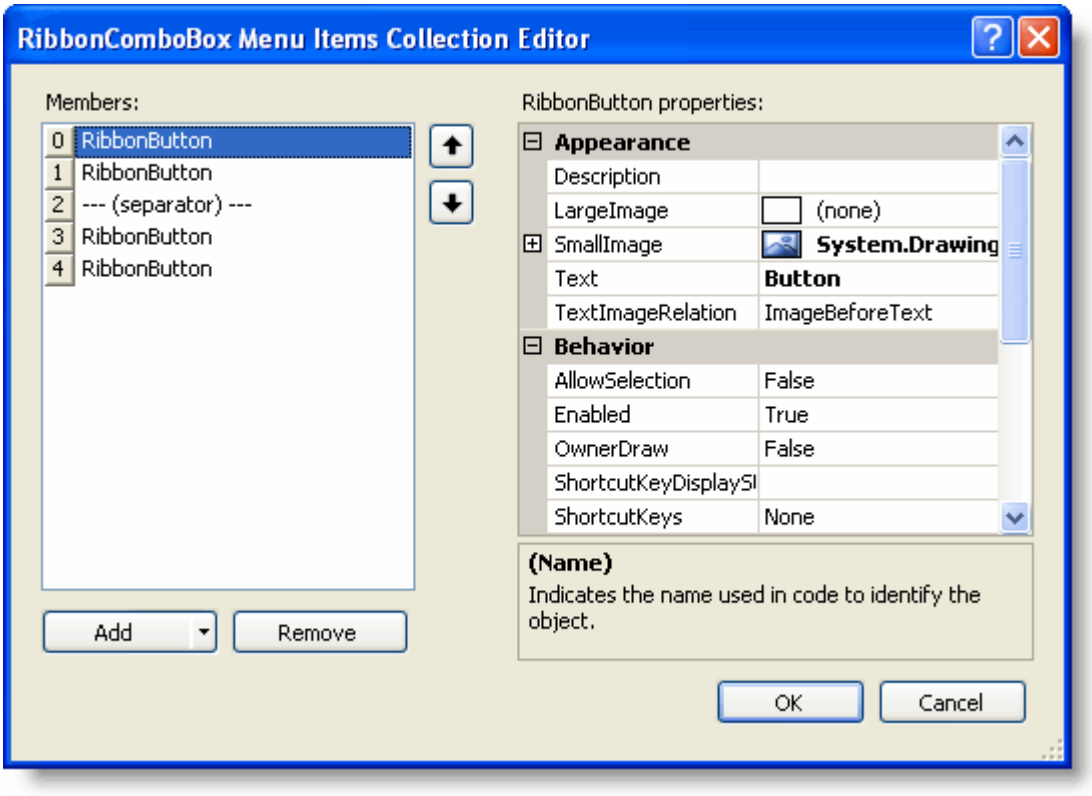


The **RibbonComboBox Items Collection Editor** appears.

RibbonComboBox Menu Items Collection Editor

The **RibbonComboBox Menu Items Collection Editor** allows you to add any number of Ribbon menu items to the drop-down portion of the combo box or remove menu items from the drop-down portion of the combo box. Additionally, you can edit the item's properties.

To edit the Ribbon item(s), use the **RibbonComboBox Menu Items Collection Editor**:



Clicking the **Add** drop-down button will reveal a drop-down list with the following Ribbon items available to add to the drop-down portion of the combo box:

- RibbonButton
- RibbonColorPickerItem
- RibbonComboBox
- RibbonLabel
- RibbonMenu
- RibbonSeparator
- RibbonSplitButton
- RibbonToggleButton

Accessing the RibbonComboBox Menu Items Collection Editor

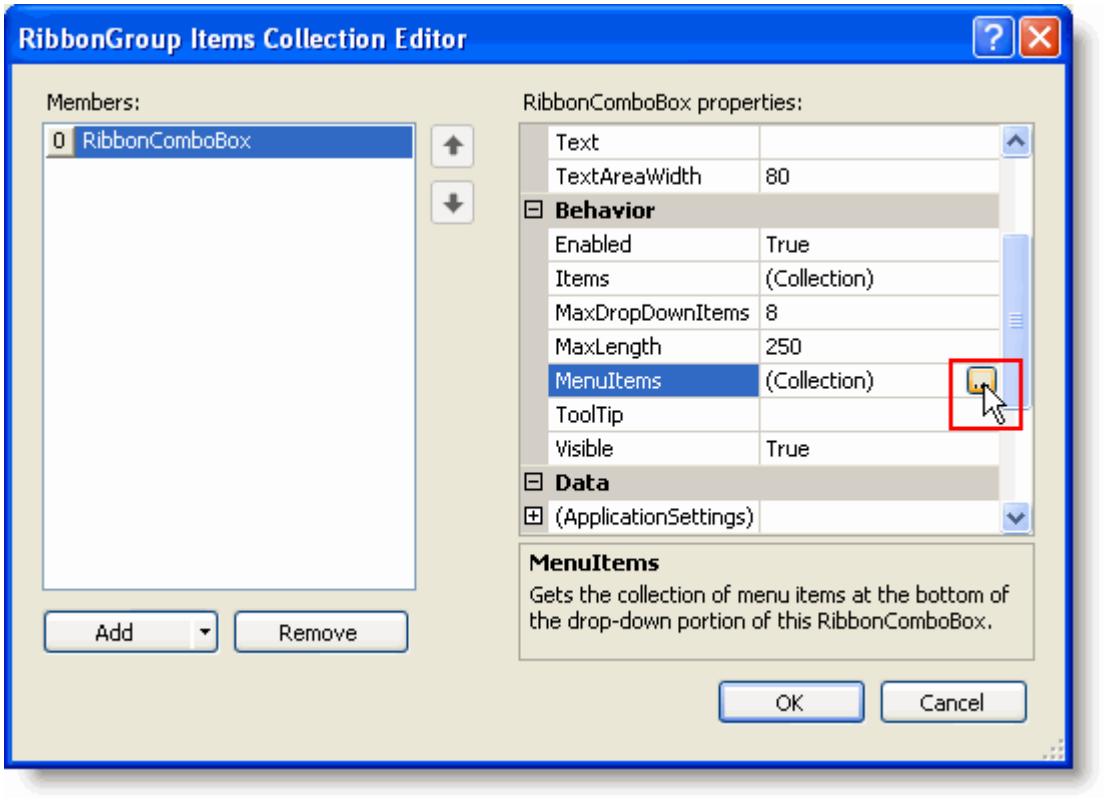
You can use one of the following two options to access **RibbonComboBox Menu Items Collection Editor**:

Option 1

1. Add a RibbonComboBox to the Ribbon group.
2. Click the combo box element to activate it.
3. In the Properties window, click on the **ellipsis** button next to the MenuItems property.
   The **RibbonComboBox Menu Items Collection Editor** appears.

Option 2

1. Access the RibbonGroup Items Collection Editor.
2. Click the **Add** drop-down button and add a combo box item.
3. With the RibbonComboBox selected in the **Members** list, click on the **ellipsis** button next to the MenuItems property in the Properties window.

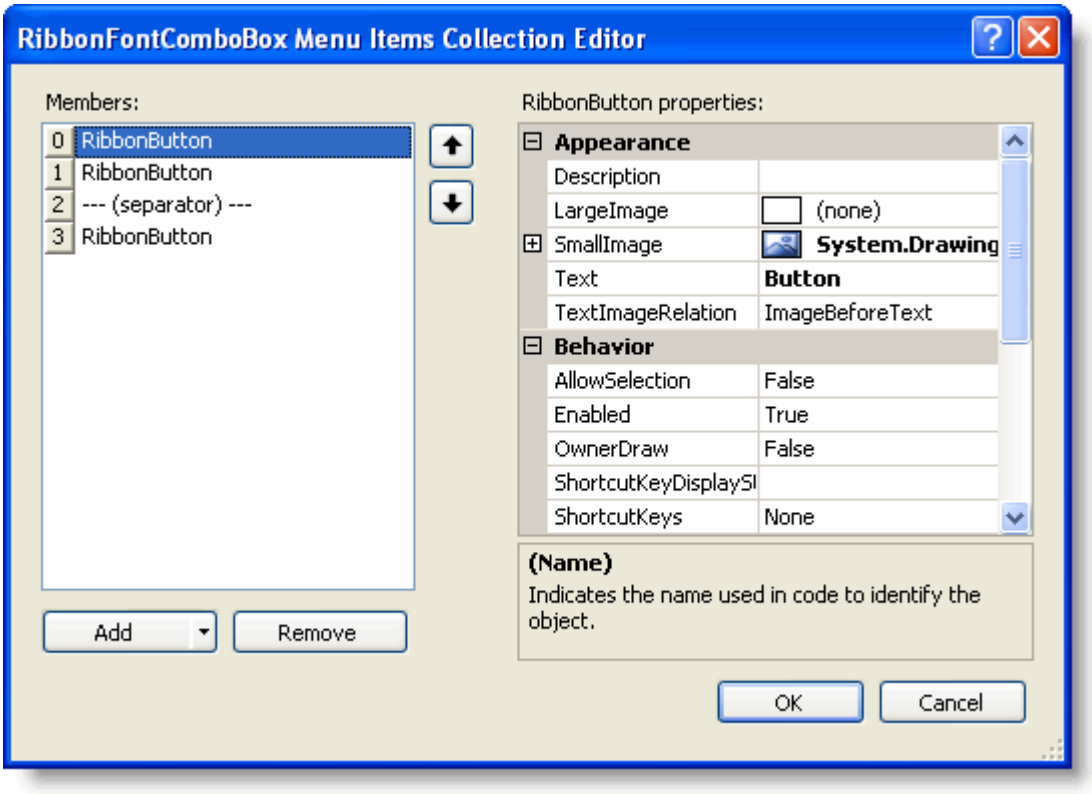


The **RibbonComboBox Menu Items Collection Editor** appears.

RibbonFontComboBox Menu Items Collection Editor

The **RibbonFontComboBox Menu Items Collection Editor** allows you to add any number of Ribbon menu items to the drop-down portion of the font combo box or remove menu items from the drop-down portion of the font combo box. Additionally, you can edit the item's properties.

To edit the Ribbon item(s), use the **RibbonFontComboBox Menu Items Collection Editor**:



Clicking the **Add** drop-down button will reveal a drop-down list with the following Ribbon items available to add to the drop-down portion of the font combo box:

- RibbonButton
- RibbonColorPickerItem
- RibbonComboBox
- RibbonLabel
- RibbonMenu
- RibbonSeparator
- RibbonSplitButton
- RibbonToggleButton

Accessing the RibbonFontComboBox Menu Items Collection Editor

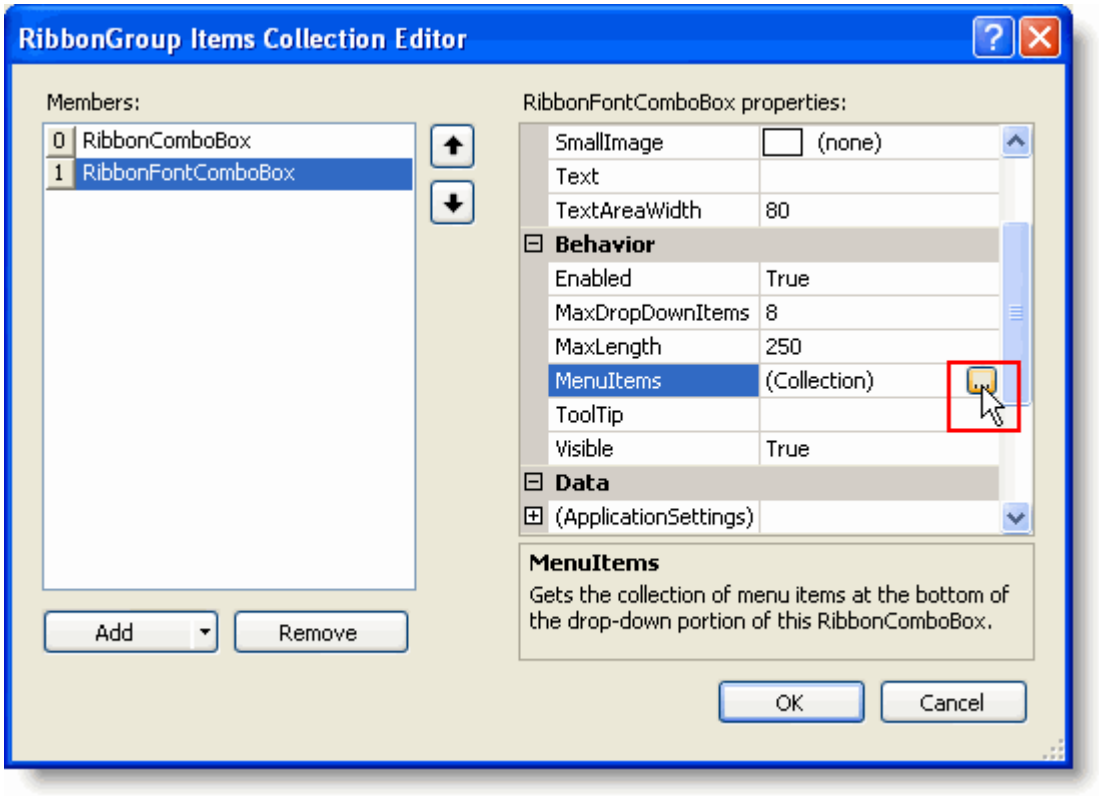
You can use one of the following two options to access **RibbonFontComboBox Menu Items Collection Editor**:

Option 1

1. Add a RibbonFontComboBox to the Ribbon group.
2. Click the font combo box element to activate it.
3. In the Properties window, click on the **ellipsis** button next to the **MenuItems** property.
   The **RibbonFontComboBox Menu Items Collection Editor** appears.

Option 2

1. Access the RibbonGroup Items Collection Editor.
2. Click the **Add** drop-down button and add a font combo box item.
3. With the RibbonFontComboBox selected in the **Members** list, click on the **ellipsis** button next to the **MenuItems** property in the Properties window.

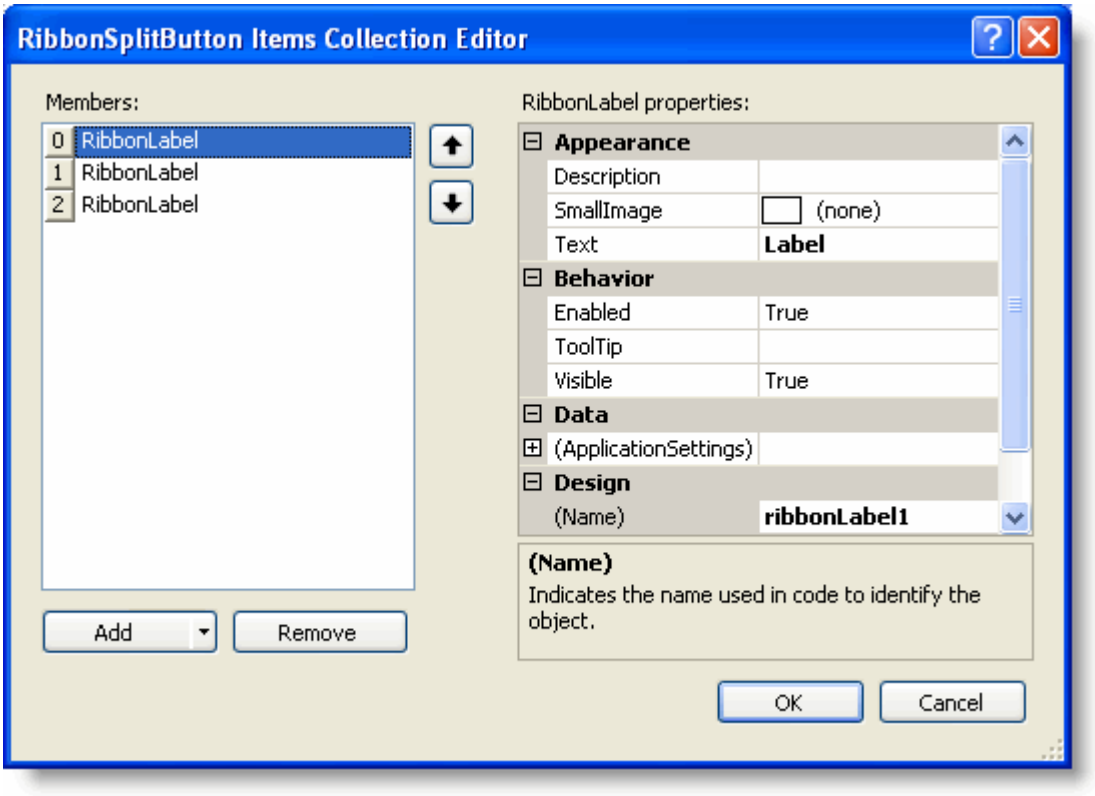


The **RibbonFontComboBox Menu Items Collection Editor** appears.

RibbonSplitButton Items Collection Editor

The **RibbonSplitButton Items Collection Editor** allows you to add any number of Ribbon items to the split button or remove items from the split button. Additionally, you can edit the item's properties.

To edit the Ribbon item(s), use the **RibbonSplitButton Items Collection Editor**:



Clicking the **Add** drop-down button will reveal a drop-down list with the following Ribbon items available to add to the menu:

- RibbonButton
- RibbonColorPickerItem
- RibbonComboBox
- RibbonLabel
- RibbonMenu
- RibbonSeparator
- RibbonSplitButton
- RibbonToggleButton

Accessing the RibbonSplitButton Items Collection Editor

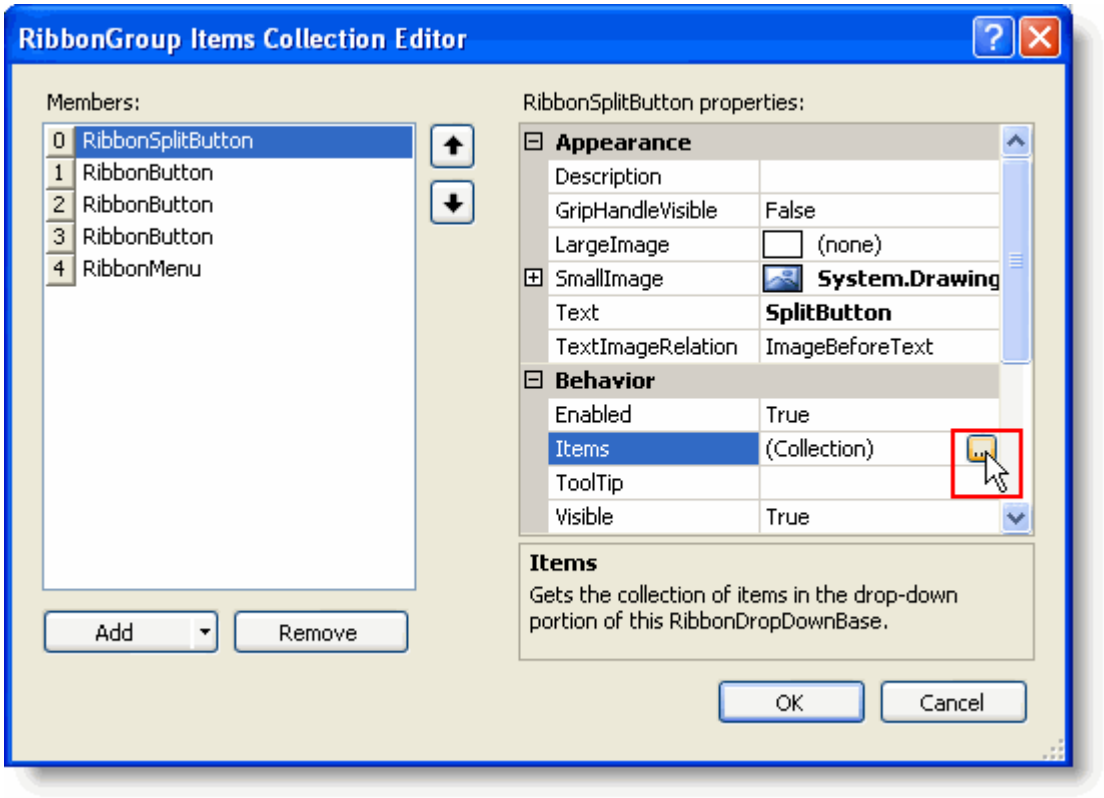
You can use one of the following two options to access **RibbonSplitButton Items Collection Editor**:

Option 1

1. Add a RibbonSplitButton to the Ribbon group.
2. Click the split button to activate it.
3. In the Properties window, click on the **ellipsis** button next to the **Items** property.
   The **RibbonSplitButton Items Collection Editor** appears.

Option 2

1. Access the RibbonGroup Items Collection Editor.
2. Click the **Add** drop-down button and add a split button.
3. With the RibbonSplitButton selected in the **Members** list, click on the **ellipsis** button next to the **Items** property in the Properties window.



The **RibbonSplitButton Items Collection Editor** appears.

C1StatusBar Collection Editors

The main part of the C1StatusBar editor's application consists of a Windows form which conveniently allows you to add status bar pane items and edit their properties. C1StatusBar provides the following collection editors:

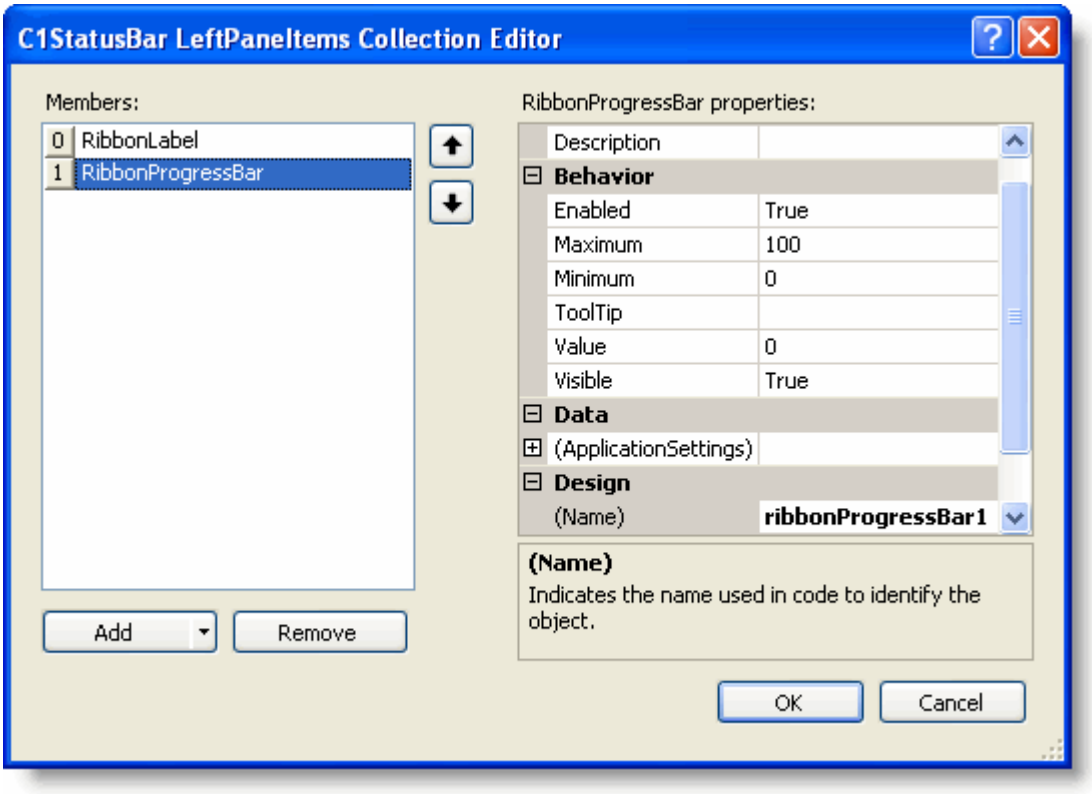
- C1StatusBar LeftPaneItems Collection Editor
- C1StatusBar RightPaneItems Collection Editor

The following topics briefly introduce the C1StatusBar collection editors and explain how to access each editor.

C1StatusBar LeftPaneItems Collection Editor

The **C1StatusBar LeftPaneItems Collection Editor** allows you to add Ribbon items to the left status bar pane or remove Ribbon items from the left status bar pane. Additionally, you can edit the item's properties.

To edit the left status bar pane, use the **C1StatusBar LeftPaneItems Collection Editor**:



Clicking the **Add** drop-down button will reveal a drop-down list with the following Ribbon items available to add to the left status bar pane:

- RibbonButton
- RibbonColorPicker
- RibbonComboBox
- RibbonDatePicker
- RibbonFontComboBox
- RibbonLabel
- RibbonMenu
- RibbonNumericBox
- RibbonProgressBar
- RibbonSeparator
- RibbonSplitButton
- RibbonTextBox
- RibbonTimePicker
- RibbonToggleButton

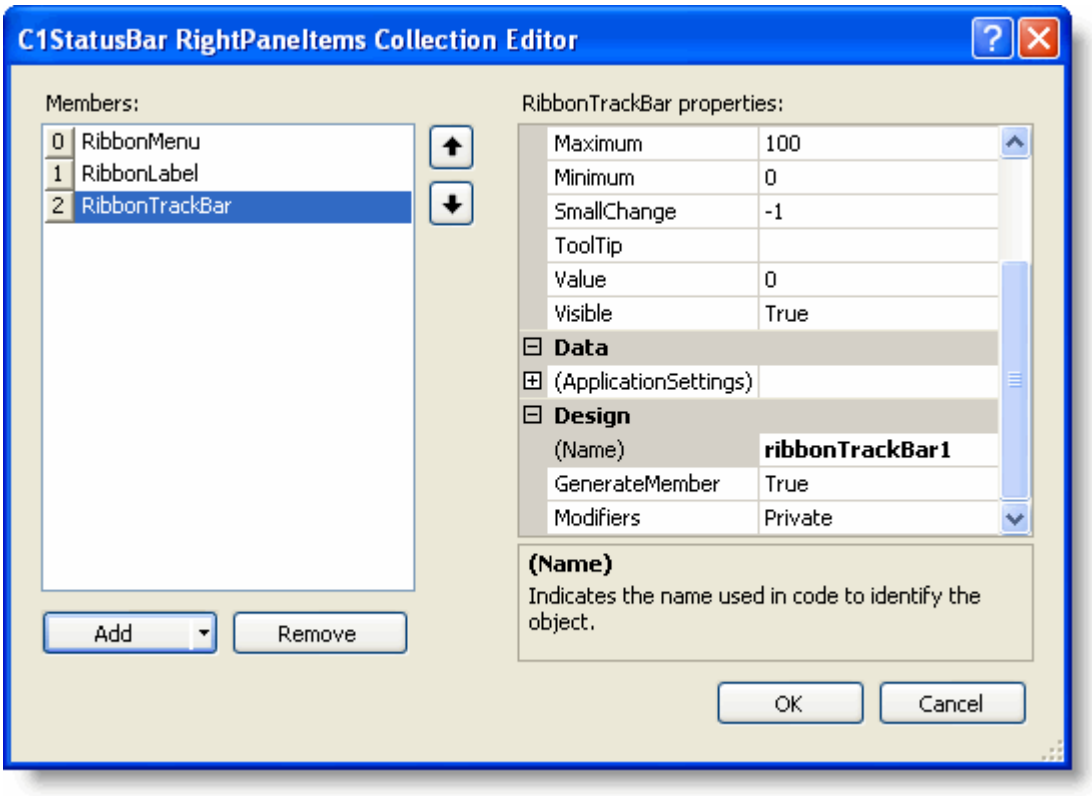
RibbonTrackBarTo access the C1StatusBar LeftPaneItems Collection Editor:

1. Add a C1StatusBar control to the Ribbon Form.
2. Click the C1StatusBar control to activate it.
3. In Properties window, click on the **(Collection)** next to the **LeftPaneItems** property, then click the **ellipsis** button.
   The **C1StatusBar LeftPaneItems Collection Editor** appears.

C1StatusBar RightPaneItems Collection Editor

The **C1StatusBar RightPaneItems Collection Editor** allows you to add Ribbon items to the right status bar pane or remove Ribbon items from the right status bar pane. Additionally, you can edit the item's properties.

To edit the right status bar pane, use the **C1StatusBar RightPaneItems Collection Editor**:



Clicking the **Add** drop-down button will reveal a drop-down list with the following Ribbon items available to add to the right status bar pane:

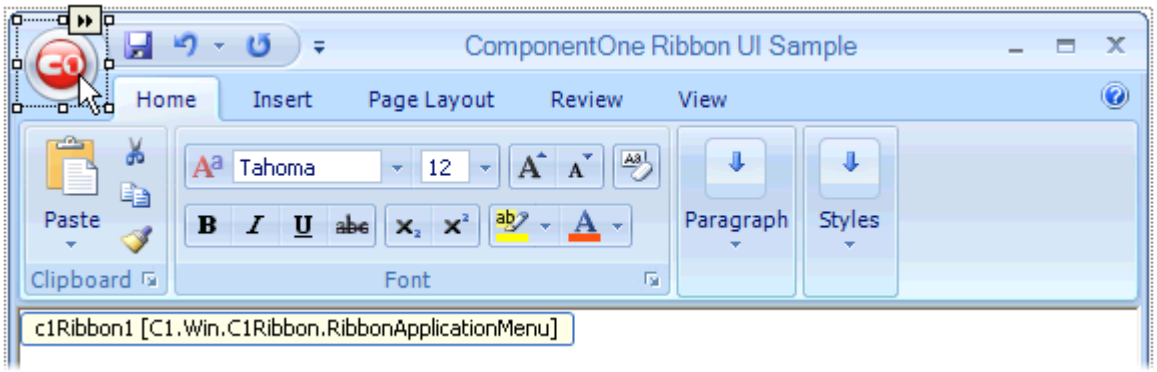
- RibbonButton
- RibbonColorPicker
- RibbonComboBox
- RibbonDatePicker
- RibbonFontComboBox
- RibbonLabel
- RibbonMenu
- RibbonNumericBox
- RibbonProgressBar
- RibbonSeparator
- RibbonSplitButton
- RibbonTextBox
- RibbonTimePicker
- RibbonToggleButton
- RibbonTrackBar

RibbonTrackBarTo access the C1StatusBar RightPaneItems Collection Editor:

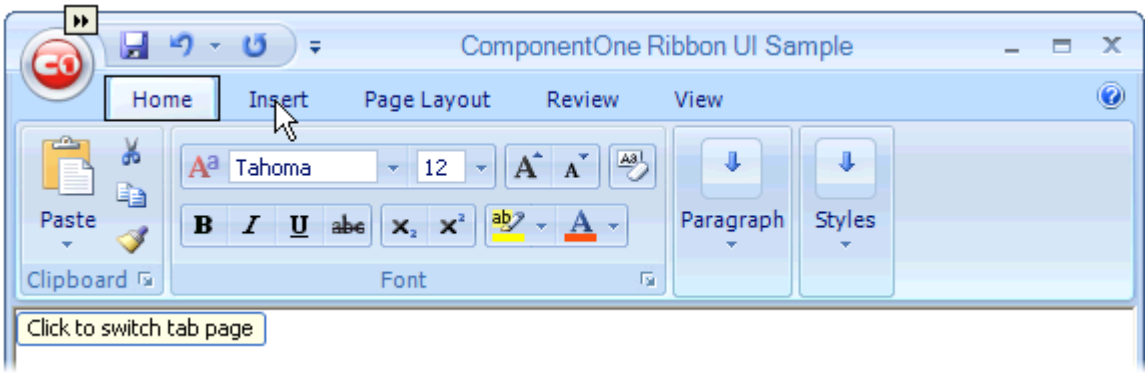
1. Add a C1StatusBar control to the Ribbon Form.
2. Click the C1StatusBar control to activate it.
3. In Properties window, click on the **(Collection)** next to the **RightPaneItems** property, then click the **ellipsis** button.
   The **C1StatusBar RightPaneItems Collection Editor** appears.

C1Ribbon Smart Designer

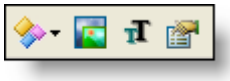
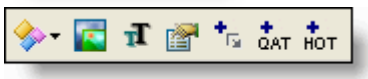
C1Ribbon provides visual editing to make it easier to create a Ribbon. Using the smart designer, you can set properties directly on the form. When you mouse over an item on the Ribbon, a tab at the lower left side of the form appears indicating what item the mouse is over. Clicking the item reveals a floating toolbar.



Some items also provide directions directly on the form on how to customize the item.



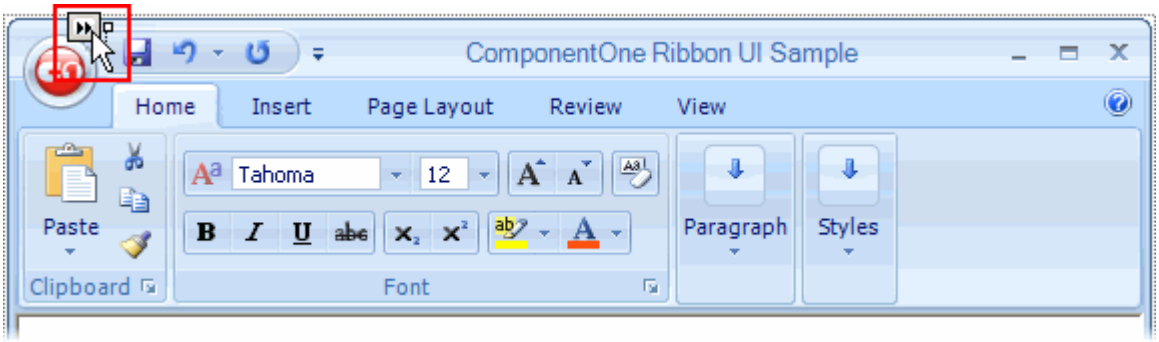
The smart designer consists of the following floating toolbars:

Floating Toolbar	Description
	**Ribbon floating toolbar:** The Ribbon's floating toolbar allows you to load and save a template, add a tab, open the tree-based designer, and set the visual style of the Ribbon.
	**Application menu floating toolbar:** The **Application** menu's floating toolbar allows you to change the **Application** button's image and enable or disable the menu.
	**Tab floating toolbar:** The tab's floating toolbar allows you to add a group, change the image, modify the text settings, and disable/enable the item.
 	**Group floating toolbars:** The group's floating toolbar allows you to load and save a group template, add Ribbon items, and disable/enable the item. Clicking the caption area of the group reveals the floating toolbar which enables you to modify text settings, and Add Launcher Button / Remove Launcher Button. You can also add the group to the QAT and hot list.

	**Button floating toolbar:** The button's floating toolbar allows you to change the image, modify the text settings, disable/enable the item, add the button to the QAT, and add the button to hot list.
	**Check Box floating toolbar:** The check box's floating toolbar allows you to change the image, modify the text settings, and disable/enable the item. You can also add the check box to the QAT and hot list.
	**Color Picker floating toolbar:** The color picker's floating toolbar allows you to change the image, modify the text settings, and disable/enable the color picker. You can also add the color picker to the QAT and hot list.
	**Combo Box floating toolbar:** The combo box's floating toolbar allows you to change the image, modify the text settings, set the text area width, max text length, and text properties, as well as disable/enable the item. You can also add the combo box to the QAT and hot list.
	**Edit Box floating toolbar:** The edit box's floating toolbar allows you to change the image, modify the text settings, set the text area width, max text length, and text properties, as well as disable/enable the item.
	**Gallery floating toolbar:** The gallery's floating toolbar allows you to change the image, modify the text settings, and disable/enable the gallery.
	**Label floating toolbar:** The label's floating toolbar allows you to change the image, modify the text settings, and disable/enable the item.
	**Menu floating toolbar:** The menu's floating toolbar allows you to change the image, modify the text settings, and disable/enable the item. You can also add the menu item to the QAT and hot list.
	**Separator floating toolbar:** The separator's floating toolbar allows you to cut, copy, paste, or delete the item. You can also add the menu item to the QAT.
	**Split Button floating toolbar:** The split button's floating toolbar allows you to change the image, modify the text settings, and disable/enable the item. You can also add the split button to the QAT and hot list.
	**Toggle Button floating toolbar:** The toggle button's floating toolbar allows you to change the image, modify the text settings, and disable/enable the item. You can also add the toggle button to the QAT and hot list.
	**Toolbar floating toolbar:** The toolbar's floating toolbar allows you to add Ribbon items and disable/enable the item.

Ribbon Floating Toolbar

The Ribbon's floating toolbar is the only floating toolbar that will appear on the form regardless of the item the mouse is positioned over. To display the main Ribbon floating toolbar, click the button that appears in the upper-left corner of the form. To close the floating toolbar, click the button.

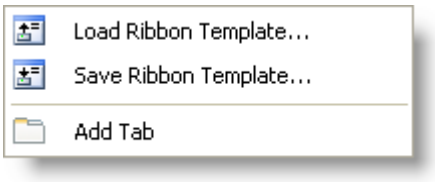


The Ribbon's floating toolbar consists of the following buttons:

	Actions: Load and save templates, and add tabs.
	Hide/show Ribbon items: Add or remove Ribbon items.
	Change Ribbon's visual style: Edit the Ribbon's visual style.
	Localize: Opens the Localize dialog box.

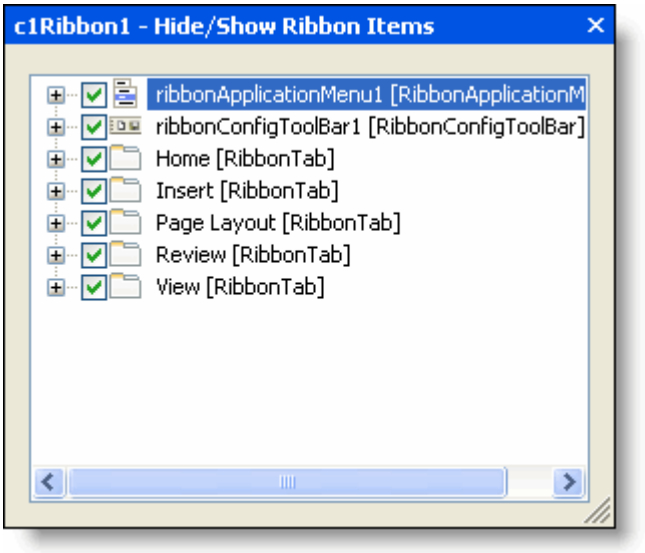
Actions

Clicking the **Actions** button opens a list of available actions. The Load Ribbon/Save Ribbon Template feature allows end users to create a collection of ready-to-use templates for the Ribbon.



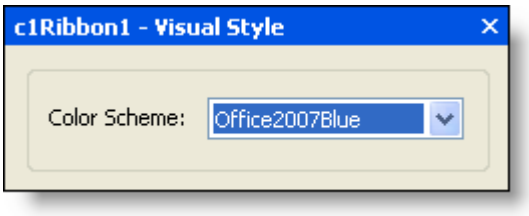
Hide/show Ribbon items

Clicking the **Hide/Show Ribbon Items** button opens the **Hide/Show Ribbon Items** designer. In the designer, you can easily select or deselect existing items on the Ribbon to hide them or show them.



Visual Style

Clicking the **Change Ribbon's Visual Style** button opens the **Visual Style** dialog box. Here you can change the visual style of the Ribbon to **Office2007Black** or **Office2007Silver** (or keep its default **Office2007Blue** style).



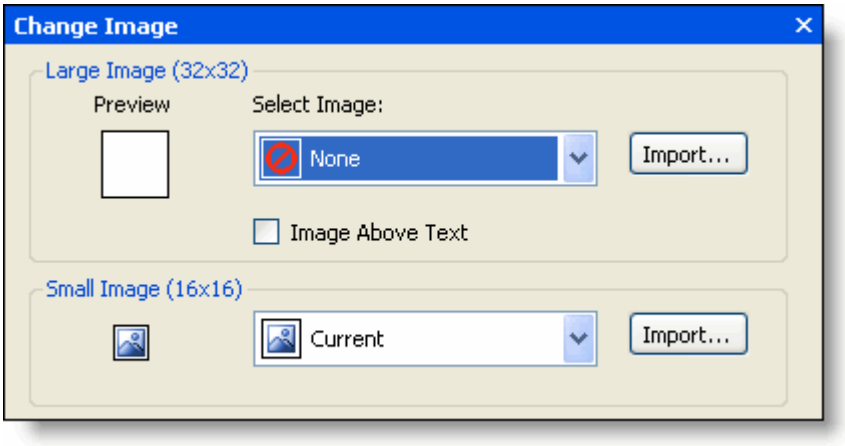
Application Menu Floating Toolbar

The **Application** menu's floating toolbar allows you to change the **Application** menu's image and enable or disable the menu. The tab's floating toolbar consists of the following buttons:

	**Change Image:** Edit image.
	**Miscellaneous settings:** Enable or disable the menu.

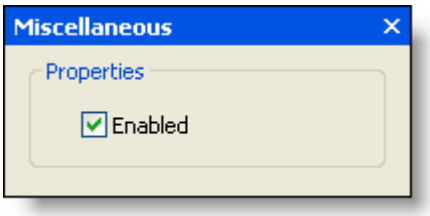
Change Image

Clicking the **Change Image** button opens the **Change Image** dialog box. You can click the **Import** button to browse for a custom image or you can click the **Select Image** drop-down arrow to select from a list of large (32x32) or small (16x16) preset images.



Miscellaneous settings

Clicking the **Miscellaneous settings** button opens the **Miscellaneous** dialog box. In the **Miscellaneous** dialog box, you can enable the menu (checked by default) or disable the menu (unchecked).



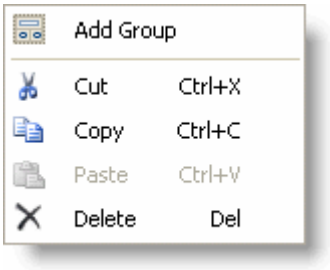
Tab Floating Toolbar

The tab's floating toolbar allows you to add groups, change the image, modify the text settings, and disable/enable the item. The tab's floating toolbar consists of the following buttons:

	**Actions:** Add groups, and Cut, Copy, Paste, or Delete the tab.
	**Change Image:** Edit image.
	**Text settings:** Edit Text and ToolTip properties.
	**Miscellaneous settings:** Enable or disable the tab.

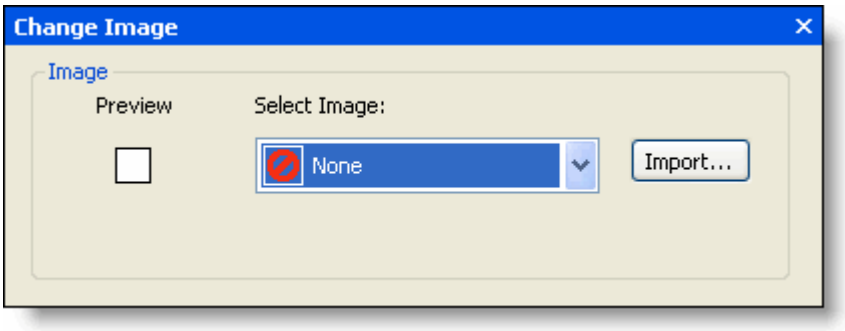
Actions

Clicking the **Actions** button opens a list of available actions.



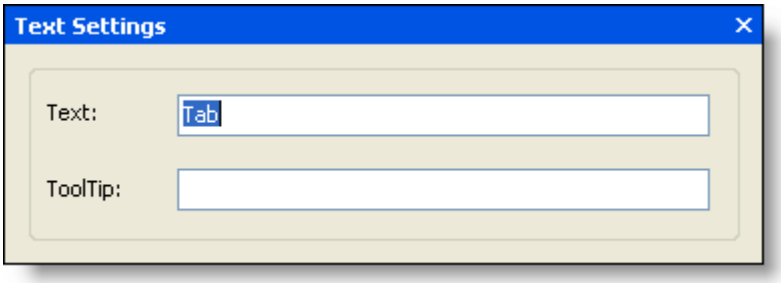
Change Image

Clicking the **Change Image** button opens the **Change Image** dialog box. You can click the **Import** button to browse for a custom 16x16 image or you can click the **Select Image** drop-down arrow to select from a list of preset 16x16 images.



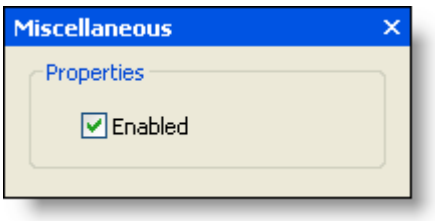
Text settings

Clicking the **Text settings** button opens the **Text Settings** dialog box. In the **Text Settings** dialog box, you can edit the tab's **Text** and **ToolTip** properties.



Miscellaneous settings

Clicking the **Miscellaneous settings** button opens the **Miscellaneous** dialog box. In the **Miscellaneous** dialog box, you can enable the tab (checked by default) or disable the tab (unchecked).



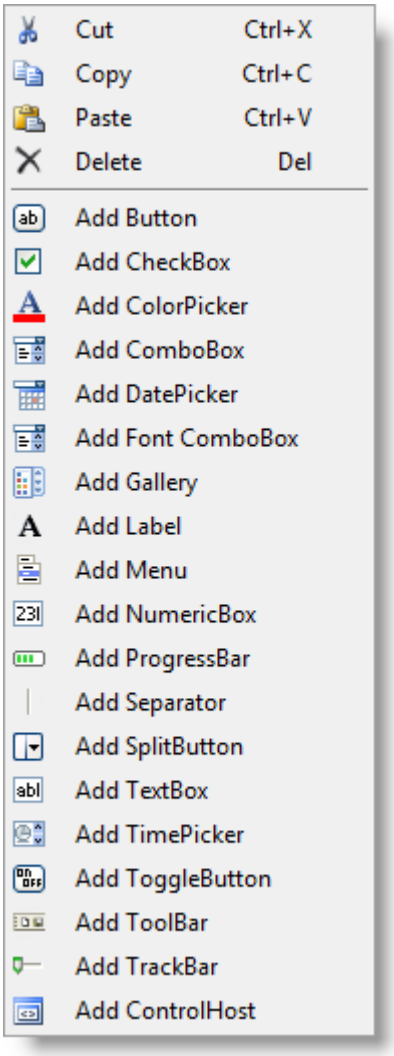
Group Floating Toolbar

The group's floating toolbar allows you to load and save a group template, add Ribbon items, and disable/enable the group. You can also add the group to the QAT and hot list. The group's floating toolbar consists of the following buttons:

	**Actions:** Cut, Copy, Paste, or Delete the group, load and save group templates, and add Ribbon items to the group.
	**Miscellaneous settings:** Enable or disable the group.
	**Text settings:** Edit Text and ToolTip properties.
	**Add Launcher Button / Remove Launcher Button:** Add or remove a dialog box launcher button to the group.
	**Add to Quick Access Toolbar:** Add the group to the QAT.
	**Add to Hot Item List:** Add the group to the hot item list.

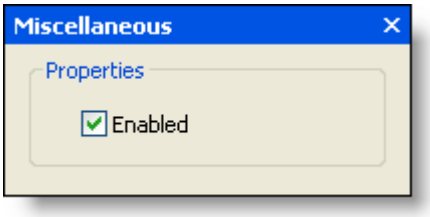
Actions

Clicking the **Actions** button opens a list of available actions. The Load Group/Save Group Template feature allows the end users to create a collection of ready-to-use templates for control groups.



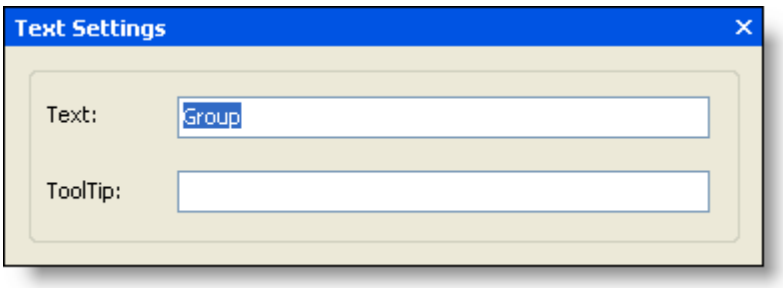
Miscellaneous settings

Clicking the **Miscellaneous settings** button opens the **Miscellaneous** dialog box. In the **Miscellaneous** dialog box, you can enable the group (checked by default) or disable the group (unchecked).



Text settings

Clicking the **Text settings** button opens the **Text Settings** dialog box. In the **Text Settings** dialog box, you can edit the group's **Text** and **ToolTip** properties.



Add Launcher Button / Remove Launcher Button

Clicking the **Add Launcher Button** will add a dialog launcher button to the group. With the dialog launcher button added to the group, clicking the **Remove Launcher Button** will remove the dialog launcher button from the group.

Add to Quick Access Toolbar

Clicking the **Add to Quick Access Toolbar** button adds the Ribbon group to the QAT. After clicking 

, the button switches to 

, which allows you to remove the Ribbon group from the QAT.

Add to Hot Item List

Clicking the **Add to Hot Item List** button adds the Ribbon group to the hot item list, which is available by clicking the drop-down arrow adjacent to the QAT. After clicking 

, the button switches to 

, which allows you to remove the Ribbon group from the hot list.

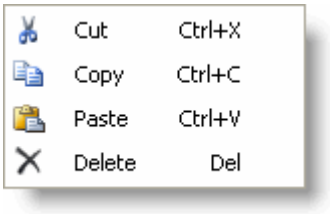
Button Floating Toolbar

The button's floating toolbar allows you to change the image, modify the text settings, and disable/enable the button. You can also add the button to the QAT and hot list. The button's floating toolbar consists of the following buttons:

	**Actions:** Cut, Copy, Paste, or Delete the item.
	**Change Image:** Edit image and image size.
	**Text settings:** Edit Text, Description, and ToolTip properties.
	**Miscellaneous settings:** Enable or disable the button.
	**Add to Quick Access Toolbar:** Add the button to the QAT.
	**Add to Hot Item List:** Add the button to the hot item list.

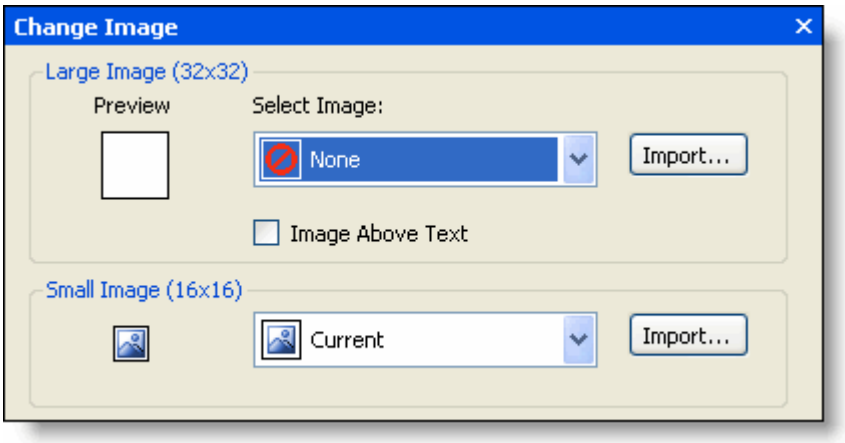
Actions

Clicking the **Actions** button opens a list of available actions.



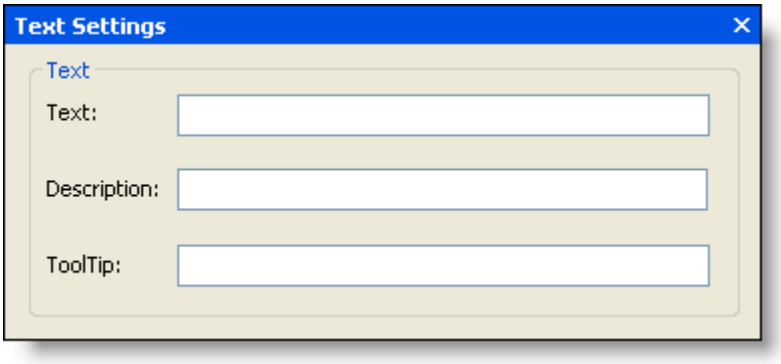
Change Image

Clicking the **Change Image** button opens the **Change Image** dialog box. You can click the **Import** button to browse for a custom image or you can click the **Select Image** drop-down arrow to select from a list of large (32x32) or small (16x16) preset images. Note that you also have the option to place the image above the text (default for the large image).



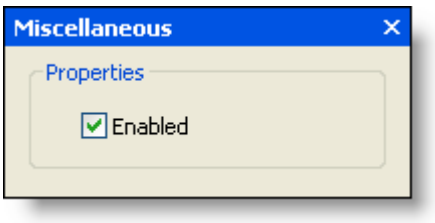
Text settings

Clicking the **Text settings** button opens the **Text Settings** dialog box. In the **Text Settings** dialog box, you can edit the button's text, including the **Text**, **Description**, and **ToolTip**.



Miscellaneous settings

Clicking the **Miscellaneous settings** button opens the **Miscellaneous** dialog box. In the **Miscellaneous** dialog box, you can enable the button (checked by default) or disable the button (unchecked).



Add to Quick Access Toolbar

Clicking the **Add to Quick Access Toolbar** button adds the Ribbon button to the QAT. After clicking 

, the button switches to 

, which allows you to remove the Ribbon button from the QAT.

Add to Hot Item List

Clicking the **Add to Hot Item List** button adds the Ribbon button to the hot item list, which is available by clicking the drop-down arrow adjacent to the QAT. After clicking 

, the button switches to 

, which allows you to remove the Ribbon button from the hot list.

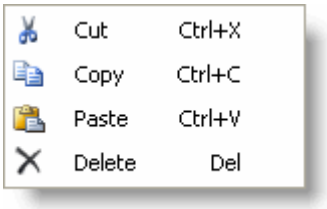
Check Box Floating Toolbar

The check box's floating toolbar allows you to change the image, modify the text settings, and disable/enable the check box. You can also add the check box to the QAT and hot list. The check box's floating toolbar consists of the following buttons:

	**Actions:** Cut, Copy, Paste, or Delete the item.
	**Change Image:** Edit image.
	**Text settings:** Edit Text, Description, and ToolTip properties.
	**Miscellaneous settings:** Enable or disable the check box.
	**Add to Quick Access Toolbar:** Add the check box to the QAT.
	**Add to Hot Item List:** Add the check box to the hot item list.

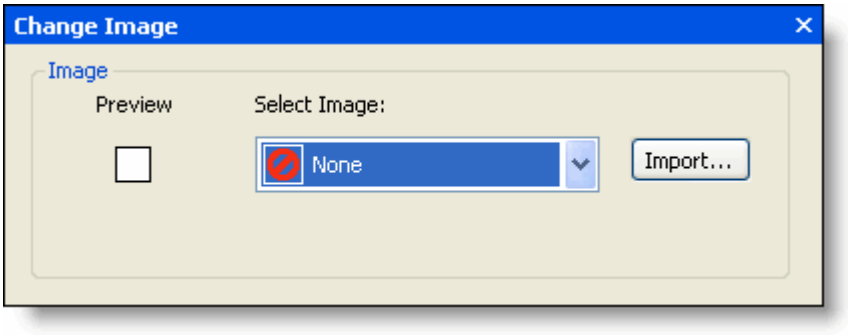
Actions

Clicking the **Actions** button opens a list of available actions.



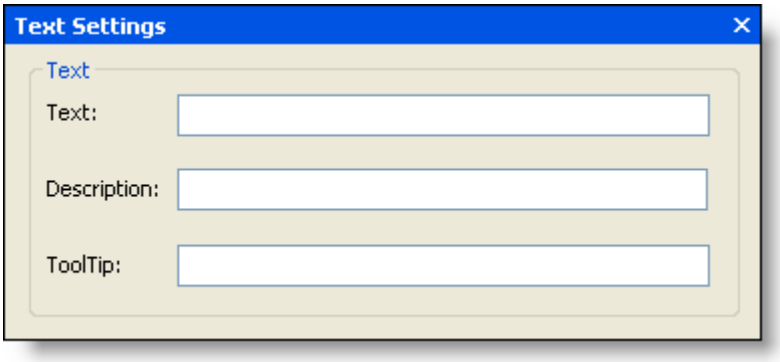
Change Image

Clicking the **Change Image** button opens the **Change Image** dialog box. You can click the **Import** button to browse for a custom 16x16 image or you can click the **Select Image** drop-down arrow to select from a list of preset 16x16 images.



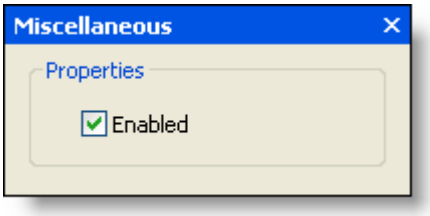
Text settings

Clicking the **Text settings** button opens the **Text Settings** dialog box. In the **Text Settings** dialog box, you can edit the check box's text, including the **Text**, **Description**, and **ToolTip**.



Miscellaneous settings

Clicking the **Miscellaneous settings** button opens the **Miscellaneous** dialog box. In the **Miscellaneous** dialog box, you can enable the check box (checked by default) or disable the check box (unchecked).



Add to Quick Access Toolbar

Clicking the **Add to Quick Access Toolbar** button adds the check box to the QAT. After clicking 

, the button

switches to  , which allows you to remove the check box from the QAT.

Add to Hot Item List

Clicking the **Add to Hot Item List** button adds the check box to the hot item list, which is available by clicking the drop-down arrow adjacent to the QAT. After clicking  , the button switches to  , which allows you to remove the check box from the hot list.

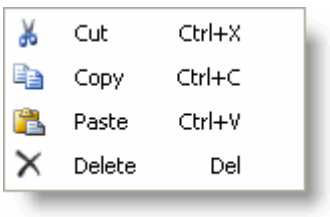
Color Picker Floating Toolbar

The color picker's floating toolbar allows you to change the image, modify the text settings, and disable/enable the color picker. You can also add the color picker to the QAT and hot list. The color picker's floating toolbar consists of the following buttons:

	**Actions:** Cut, Copy, Paste, or Delete the item.
	**Change Image:** Edit image.
	**Text settings:** Edit Text, Description, and ToolTip properties.
	**Miscellaneous settings:** Enable or disable the color picker.
	**Add to Quick Access Toolbar:** Add the color picker to the QAT.
	**Add to Hot Item List:** Add the color picker to the hot item list.

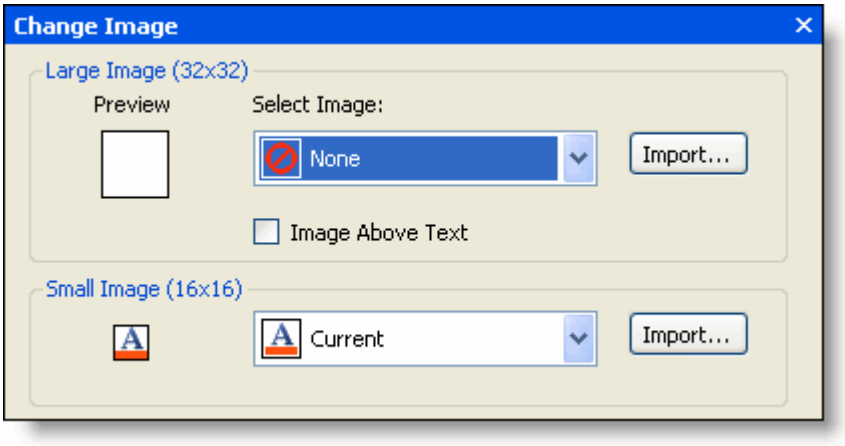
Actions

Clicking the **Actions** button opens a list of available actions.



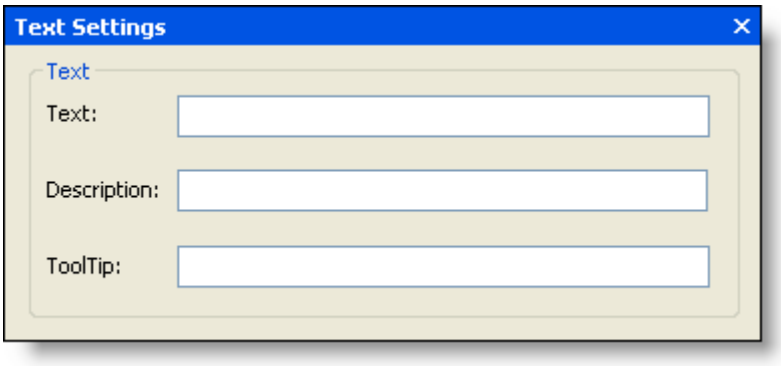
Change Image

Clicking the **Change Image** button opens the **Change Image** dialog box. You can click the **Import** button to browse for a custom image or you can click the **Select Image** drop-down arrow to select from a list of large (32x32) or small (16x16) preset images. Note that you also have the option to place the image above the text (default for the large image).



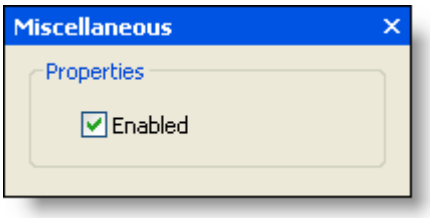
Text settings

Clicking the **Text settings** button opens the **Text Settings** dialog box. In the **Text Settings** dialog box, you can edit the check box's text, including the **Text**, **Description**, and **ToolTip**.



Miscellaneous settings

Clicking the **Miscellaneous settings** button opens the **Miscellaneous** dialog box. In the **Miscellaneous** dialog box, you can enable the color picker (checked by default) or disable the color picker (unchecked).



Add to Quick Access Toolbar

Clicking the **Add to Quick Access Toolbar** button adds the color picker to the QAT. After clicking 

, the button switches to 

, which allows you to remove the color picker from the QAT.

Add to Hot Item List

Clicking the **Add to Hot Item List** button adds the color picker to the hot item list, which is available by clicking the drop-down arrow adjacent to the QAT. After clicking 

, the button switches to 

, which allows you to remove the color picker from the hot list.

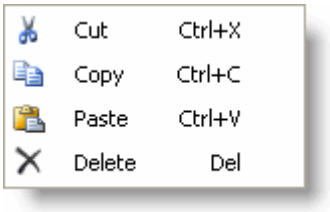
Combo Box Floating Toolbar

The combo box's floating toolbar allows you to change the image, modify the text settings, set the text area width, max text length, and text properties, as well as disable/enable the combo box. You can also add the combo box to the QAT and hot list. The combo box's floating toolbar consists of the following buttons:

	**Actions:** Cut, Copy, Paste, or Delete the item.
	**Change Image:** Edit image.
	**Text settings:** Edit Text Description, and ToolTip properties.
	**Miscellaneous settings:** Set the text area width, max text length, and text properties, as well as enable or disable the combo box.
	**Add to Quick Access Toolbar:** Add the combo box to the QAT.
	**Add to Hot Item List:** Add the combo box to the hot item list.

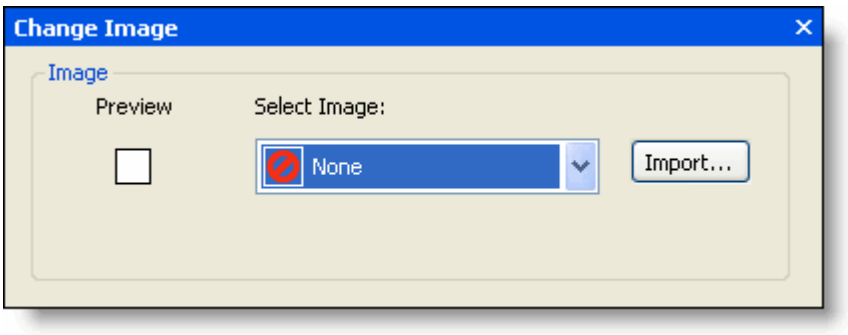
Actions

Clicking the **Actions** button opens a list of available actions.



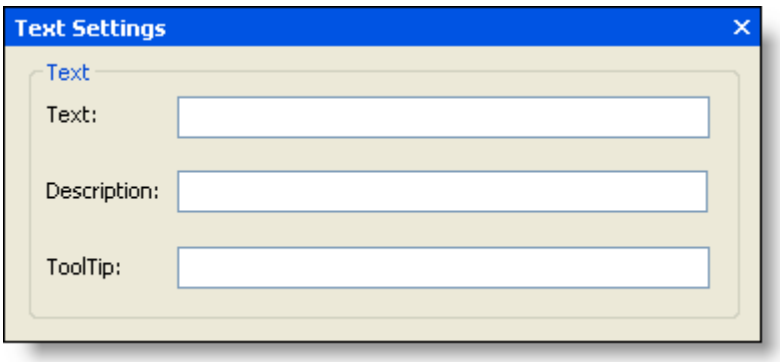
Change Image

Clicking the **Change Image** button opens the **Change Image** dialog box. You can click the **Import** button to browse for a custom 16x16 image or you can click the **Select Image** drop-down arrow to select from a list of preset 16x16 images.



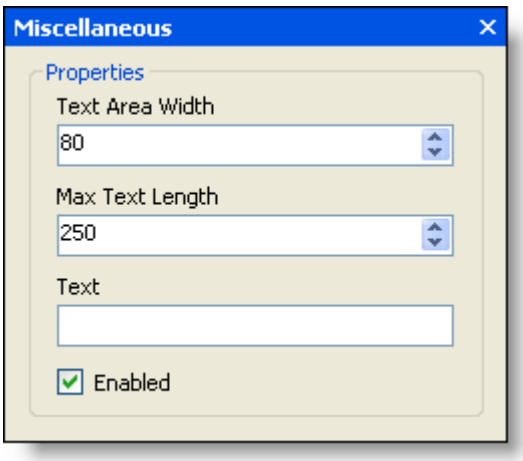
Text settings

Clicking the **Text settings** button opens the **Text Settings** dialog box. In the **Text Settings** dialog box, you can edit the combo box's text properties, including the **Text**, **Description**, and **ToolTip**.



Miscellaneous settings

Clicking the **Miscellaneous settings** button opens the **Miscellaneous** dialog box. In the **Miscellaneous** dialog box, you can set the text area width, max text length, and text properties. Additionally, you can enable the combo box (checked by default) or disable the combo box (unchecked).



Add to Quick Access Toolbar

Clicking the **Add to Quick Access Toolbar** button adds the combo box to the QAT. After clicking  , the button switches to  , which allows you to remove the combo box from the QAT.

Add to Hot Item List

Clicking the **Add to Hot Item List** button adds the combo box to the hot item list, which is available by clicking the drop-down arrow adjacent to the QAT. After clicking  , the button switches to  , which allows you to remove the combo box from the hot list.

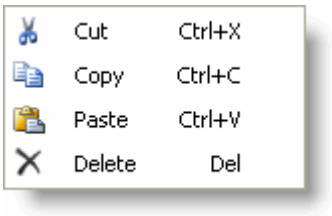
Edit Box Floating Toolbar

The edit box's floating toolbar allows you to change the image, modify the text settings, set the text area width, max text length, and text properties, as well as disable/enable the edit box. The edit box's floating toolbar consists of the following buttons:

	**Actions:** Cut, Copy, Paste, or Delete the item.
	**Change Image:** Edit image.
	**Text settings:** Edit Text, Description, and ToolTip properties.
	**Miscellaneous settings:** Set the text area width, max text length, and text properties, as well as enable or disable the edit box.

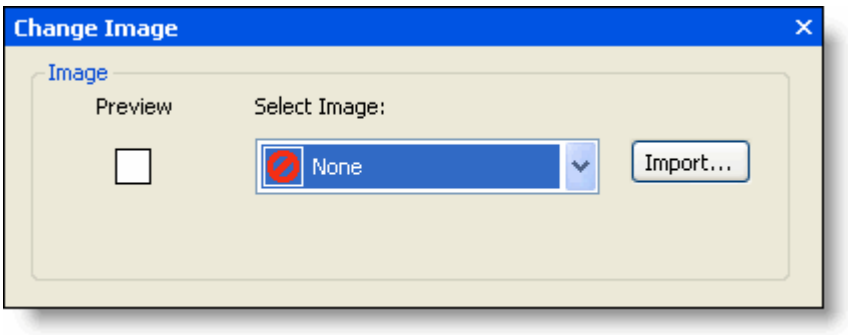
Actions

Clicking the **Actions** button opens a list of available actions.



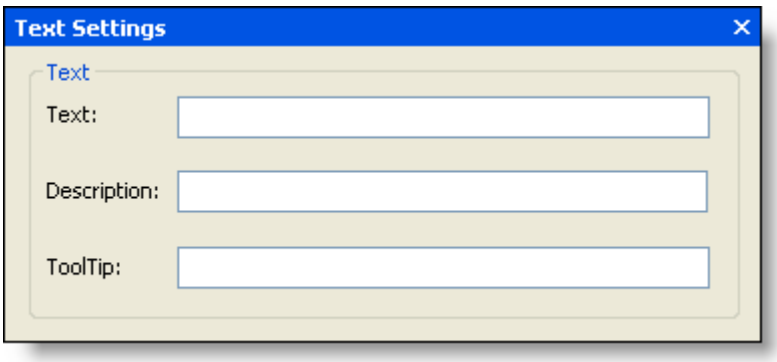
Change Image

Clicking the **Change Image** button opens the **Change Image** dialog box. You can click the **Import** button to browse for a custom 16x16 image or you can click the **Select Image** drop-down arrow to select from a list of preset 16x16 images.



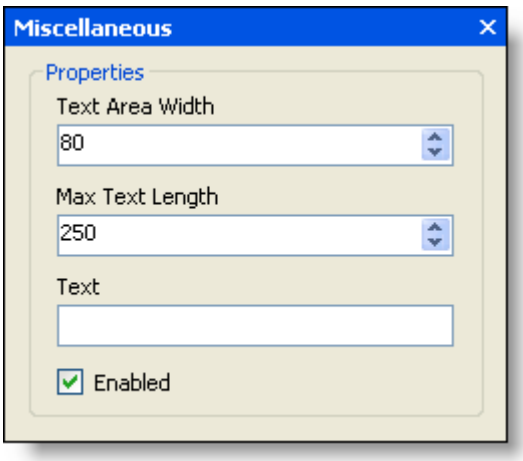
Text settings

Clicking the **Text settings** button opens the **Text Settings** dialog box. In the **Text Settings** dialog box, you can edit the edit box's text properties, including the **Text**, **Description**, and **ToolTip**.



Miscellaneous settings

Clicking the **Miscellaneous settings** button opens the **Miscellaneous** dialog box. In the **Miscellaneous** dialog box, you can set the text area width, max text length, and text properties. Additionally, you can enable the edit box (checked by default) or disable the edit box (unchecked).



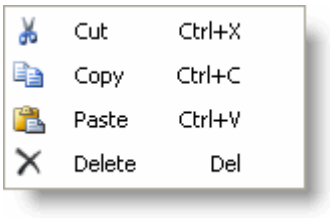
Gallery Floating Toolbar

The gallery's floating toolbar allows you to change the image, modify the text settings, and disable/enable the gallery. The gallery's floating toolbar consists of the following buttons:

	**Actions:** Cut, Copy, Paste, or Delete the item.
	**Change Image:** Edit image.
	**Text settings:** Edit Text, Description, and ToolTip properties.
	**Miscellaneous settings:** Enable or disable the gallery.

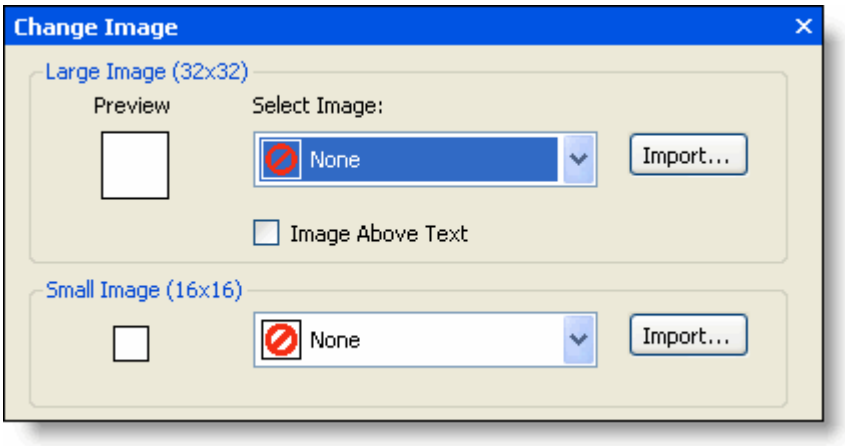
Actions

Clicking the **Actions** button opens a list of available actions.



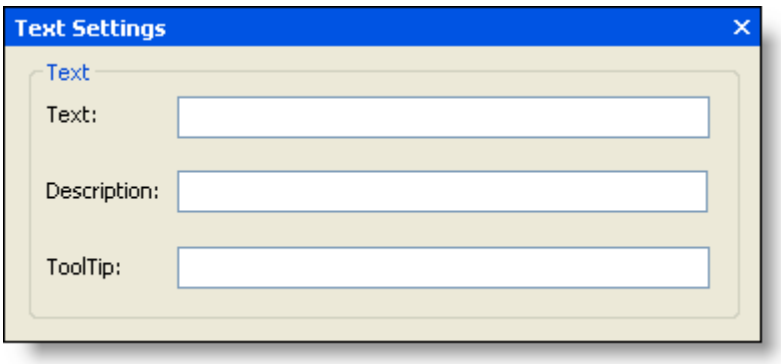
Change Image

Clicking the **Change Image** button opens the **Change Image** dialog box. You can click the **Import** button to browse for a custom image or you can click the **Select Image** drop-down arrow to select from a list of large (32x32) or small (16x16) preset images. Note that you also have the option to place the image above the text (default for the large image).



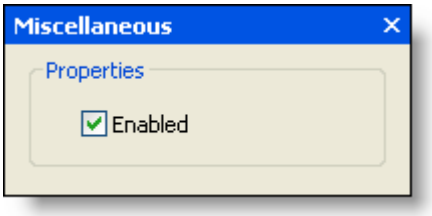
Text settings

Clicking the **Text settings** button opens the **Text Settings** dialog box. In the **Text Settings** dialog box, you can edit the gallery's text, including the **Text**, **Description**, and **ToolTip**.



Miscellaneous settings

Clicking the **Miscellaneous settings** button opens the **Miscellaneous** dialog box. In the **Miscellaneous** dialog box, you can enable the gallery (checked by default) or disable the gallery (unchecked).



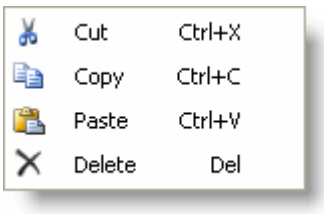
Label Floating Toolbar

The label's floating toolbar allows you to change the image, modify the text settings, and disable/enable the label. The label's floating toolbar consists of the following buttons:

	Actions: Cut, Copy, Paste, or Delete the item.
	Change Image: Edit image.
	Text settings: Edit Text, Description, and ToolTip properties.
	Miscellaneous settings: Enable or disable the label.

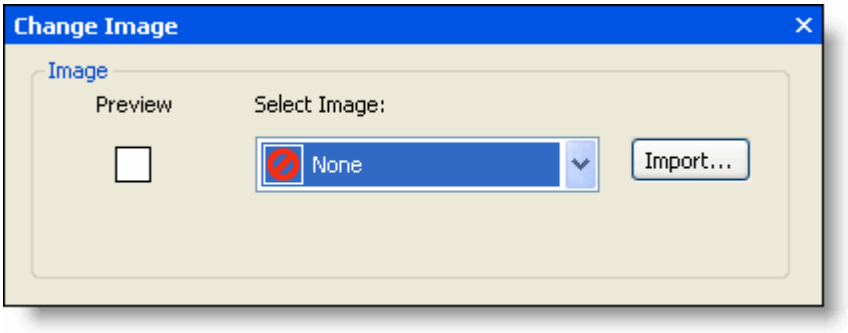
Actions

Clicking the **Actions** button opens a list of available actions.



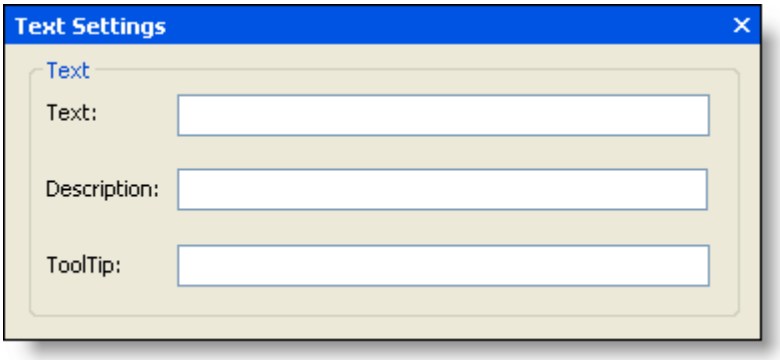
Change Image

Clicking the **Change Image** button opens the **Change Image** dialog box. You can click the **Import** button to browse for a custom 16x16 image or you can click the **Select Image** drop-down arrow to select from a list of preset 16x16 images.



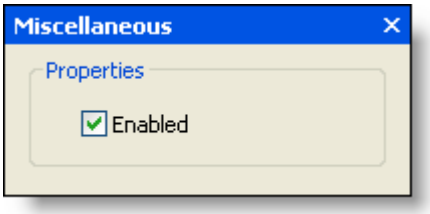
Text settings

Clicking the **Text settings** button opens the **Text Settings** dialog box. In the **Text Settings** dialog box, you can edit the label's text properties, including the **Text**, **Description**, and **ToolTip**.



Miscellaneous settings

Clicking the **Miscellaneous settings** button opens the **Miscellaneous** dialog box. In the **Miscellaneous** dialog box, you can enable the label (checked by default) or disable the label (unchecked).



Menu Floating Toolbar

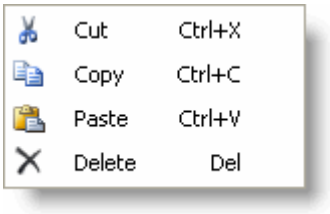
The menu's floating toolbar allows you to change the image, modify the text settings, and disable/enable the menu.

You can also add the menu item to the QAT and hot list. The menu's floating toolbar consists of the following buttons:

	Actions: Cut, Copy, Paste, or Delete the menu item.
	Change Image: Edit image and image size.
	Text settings: Edit Text, Description, and ToolTip properties.
	Miscellaneous settings: Enable or disable the menu item.
QAT	**Add to Quick Access Toolbar:** Add the menu item to the QAT.
HOT	**Add to Hot Item List:** Add the menu item to the hot item list.

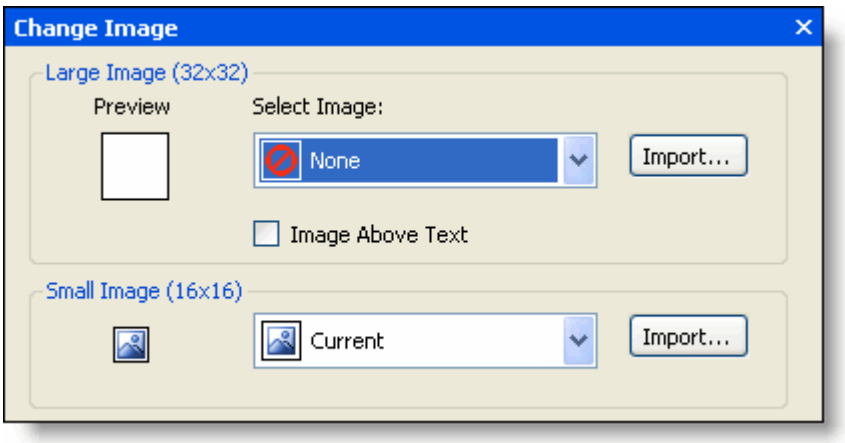
Actions

Clicking the **Actions** button opens a list of available actions.



Change Image

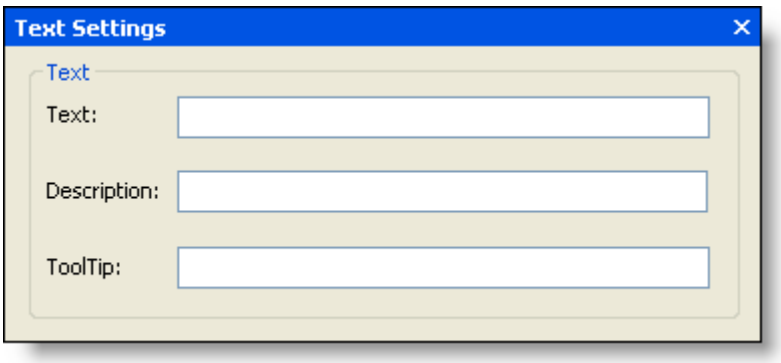
Clicking the **Change Image** button opens the **Change Image** dialog box. You can click the **Import** button to browse for a custom image or you can click the **Select Image** drop-down arrow to select from a list of large (32x32) or small (16x16) preset images. Note that you also have the option to place the image above the text (default for the large image).



Text settings

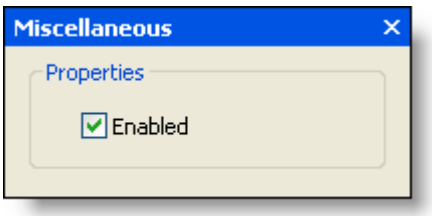
Clicking the **Text settings** button opens the **Text Settings** dialog box. In the **Text Settings** dialog box, you can edit

the menu's text properties, including the **Text**, **Description**, and **ToolTip**.



Miscellaneous settings

Clicking the **Miscellaneous settings** button opens the **Miscellaneous** dialog box. In the **Miscellaneous** dialog box, you can enable the menu (checked by default) or disable the menu (unchecked).



Add to Quick Access Toolbar

Clicking the **Add to Quick Access Toolbar** button adds the menu item to the QAT. After clicking 

, the button switches to 

, which allows you to remove the menu item from the QAT.

Add to Hot Item List

Clicking the **Add to Hot Item List** button adds the menu item to the hot item list, which is available by clicking the drop-down arrow adjacent to the QAT. After clicking 

, the button switches to 

, which allows you to remove the menu item from the hot list.

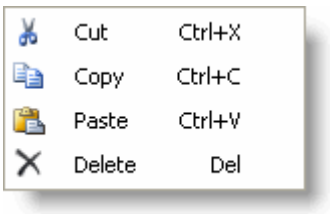
Separator Floating Toolbar

The separator's floating toolbar allows you to cut, copy, paste, or delete the item. The separator floating toolbar consists of the following buttons:

	**Actions:** Cut, Copy, Paste, or Delete the item.
	**Add to Quick Access Toolbar:** Add the menu item to the QAT.

Actions

Clicking the **Actions** button opens a list of available actions.



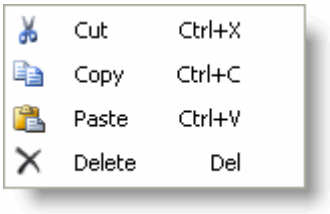
Split Button Floating Toolbar

The split button's floating toolbar allows you to change the image, modify the text settings, and disable/enable the split button. You can also add the split button to the QAT and hot list. The split button's floating toolbar consists of the following buttons:

	**Actions:** Cut, Copy, Paste, or Delete the item.
	**Change Image:** Edit image and image size.
	**Text settings:** Edit Text, Description, and ToolTip properties.
	**Miscellaneous settings:** Enable or disable the split button.
	**Add to Quick Access Toolbar:** Add the split button to the QAT.
	**Add to Hot Item List:** Add the split button to the hot item list.

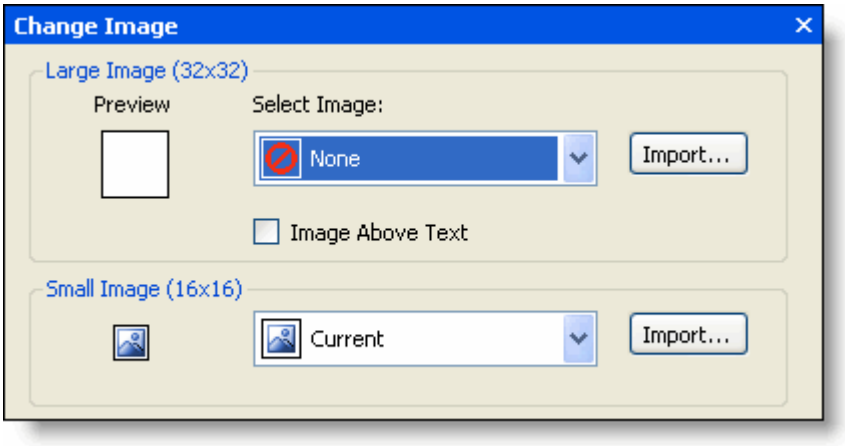
Actions

Clicking the **Actions** button opens a list of available actions.



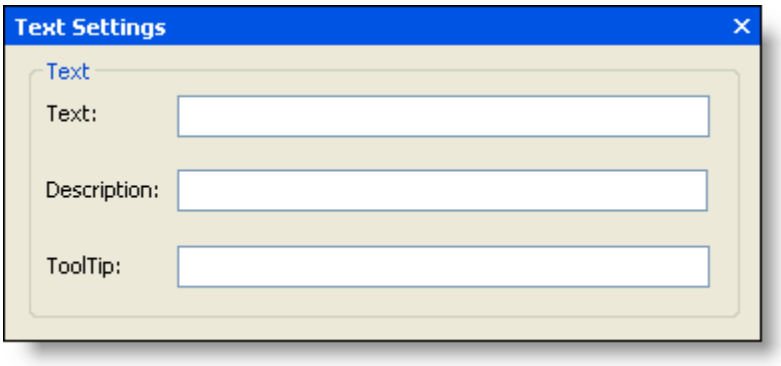
Change Image

Clicking the **Change Image** button opens the **Change Image** dialog box. You can click the **Import** button to browse for a custom image or you can click the **Select Image** drop-down arrow to select from a list of large (32x32) or small (16x16) preset images. Note that you also have the option to place the image above the text (default for the large image).



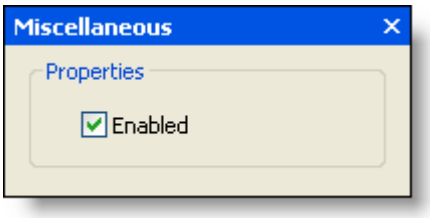
Text settings

Clicking the **Text settings** button opens the **Text Settings** dialog box. In the **Text Settings** dialog box, you can edit the split button's text properties, including the **Text**, **Description**, and **ToolTip**.



Miscellaneous settings

Clicking the **Miscellaneous settings** button opens the **Miscellaneous** dialog box. In the **Miscellaneous** dialog box, you can enable the split button (checked by default) or disable the split button (unchecked).



Add to Quick Access Toolbar

Clicking the **Add to Quick Access Toolbar** button adds the split button to the QAT. After clicking 

, the button switches to 

, which allows you to remove the split button from the QAT.

Add to Hot Item List

Clicking the **Add to Hot Item List** button adds the split button to the hot item list, which is available by clicking the drop-down arrow adjacent to the QAT. After clicking 

, the button switches to 

, which allows you to remove the split button from the hot list.

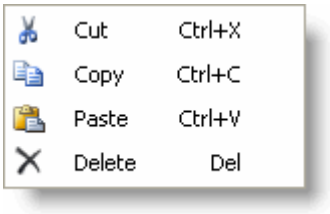
Toggle Button Floating Toolbar

The toggle button's floating toolbar allows you to change the image, modify the text settings, and disable/enable the toggle button. You can also add the toggle button to the QAT and hot list. The toggle button's floating toolbar consists of the following buttons:

	**Actions:** Cut, Copy, Paste, or Delete the item.
	**Change Image:** Edit image and image size.
	**Text settings:** Edit Text, Description, and ToolTip properties.
	**Miscellaneous settings:** Enable or disable the toggle button.
	**Add to Quick Access Toolbar:** Add the split button to the QAT.
	**Add to Hot Item List:** Add the split button to the hot item list.

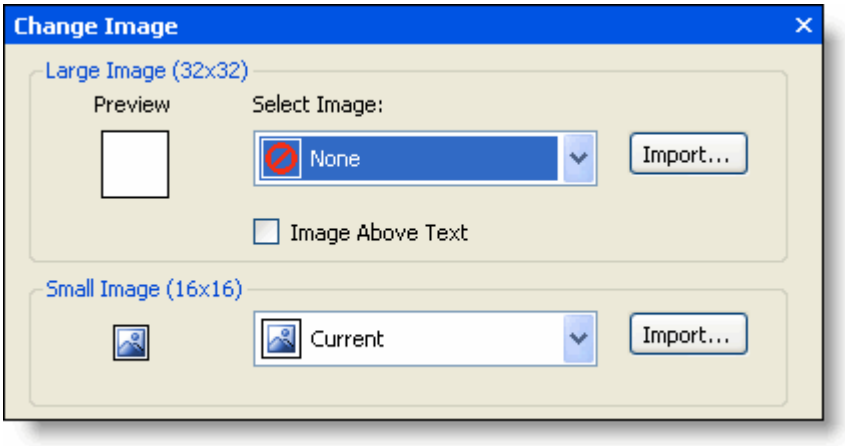
Actions

Clicking the **Actions** button opens a list of available actions.



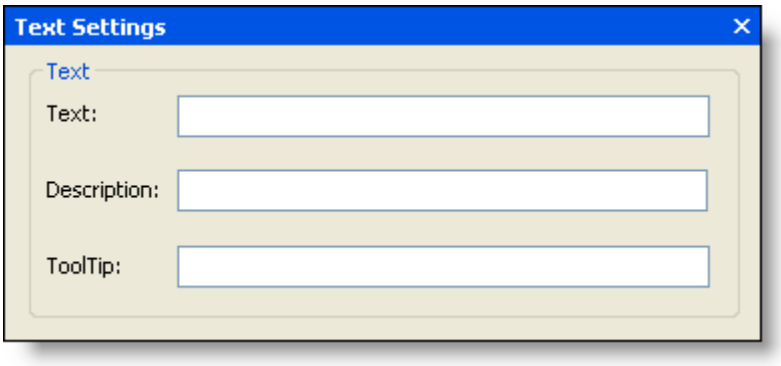
Change Image

Clicking the **Change Image** button opens the **Change Image** dialog box. You can click the **Import** button to browse for a custom image or you can click the **Select Image** drop-down arrow to select from a list of large (32x32) or small (16x16) preset images. Note that you also have the option to place the image above the text (default for the large image).



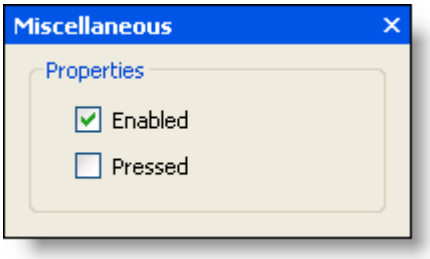
Text settings</2H>

Clicking the **Text settings** button opens the **Text Settings** dialog box. In the **Text Settings** dialog box, you can edit the toggle button's text properties, including the **Text**, **Description**, and **ToolTip**.



Miscellaneous settings

Clicking the **Miscellaneous settings** button opens the **Miscellaneous** dialog box. In the **Miscellaneous** dialog box, you can select the **Pressed** checkbox to show the toggle button as pressed (orange background). Additionally, you can enable the toggle button (checked by default) or disable the toggle button (unchecked).



Add to Quick Access Toolbar

Clicking the **Add to Quick Access Toolbar** button adds the toggle button to the QAT. After clicking 

, the button switches to 

, which allows you to remove the toggle button from the QAT.

Add to Hot Item List

Clicking the **Add to Hot Item List** button adds the toggle button to the hot item list, which is available by clicking the drop-down arrow adjacent to the QAT. After clicking 

, the button switches to 

, which allows you to remove the toggle button from the hot list.

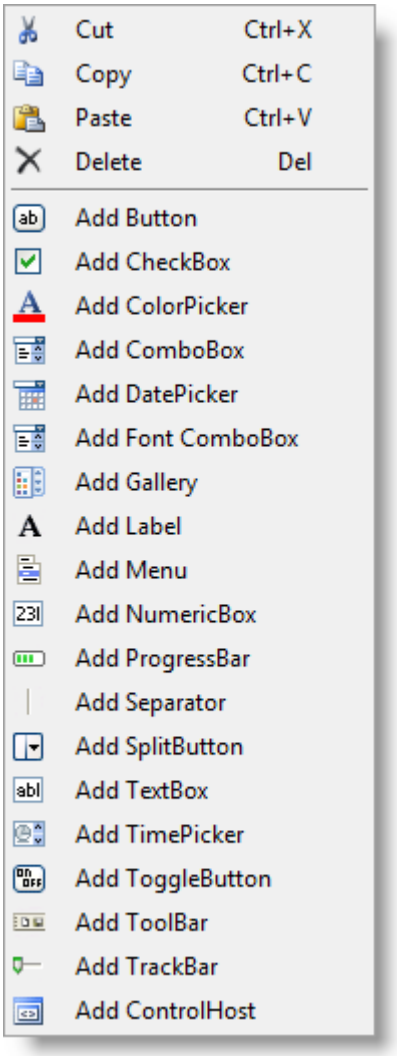
Toolbar Floating Toolbar

The Ribbon toolbar's floating toolbar allows you to add Ribbon items and disable/enable the Ribbon toolbar. Click the item to enable the item's floating toolbar. The Ribbon toolbar floating toolbar consists of the following buttons:

	**Actions:** Cut, Copy, Paste, or Delete the toolbar, and add Ribbon items to the toolbar.
	**Miscellaneous settings:** Enable or disable the Ribbon toolbar.

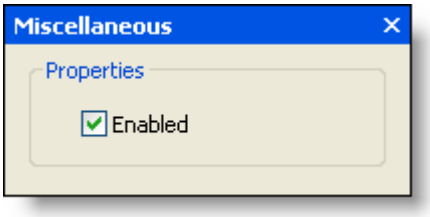
Actions

Clicking the **Actions** button opens a list of available actions.



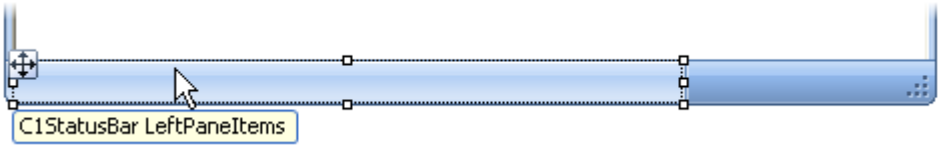
Miscellaneous settings

Clicking the **Miscellaneous settings** button opens the **Miscellaneous** dialog box. In the **Miscellaneous** dialog box, you can enable the Ribbon toolbar (checked by default) or disable the Ribbon toolbar (unchecked).



C1StatusBar Smart Designer

The C1StatusBar provides visual editing to make it easier to create a status bar. Using the smart designer, you can add items to the status bar directly on the form. When you mouse over the C1StatusBar control, a tab at the lower left side of the form appears indicating what item the mouse is over.



Clicking the item reveals the following floating toolbars:

Floating Toolbar	Description
	**LeftPaneItems floating toolbar:** The LeftPaneItems' floating toolbar allows you to add items to the left pane of the status bar.
	**RightPaneItems floating toolbar:** The RightPaneItems' floating toolbar allows you to add items to the right pane of the status bar.

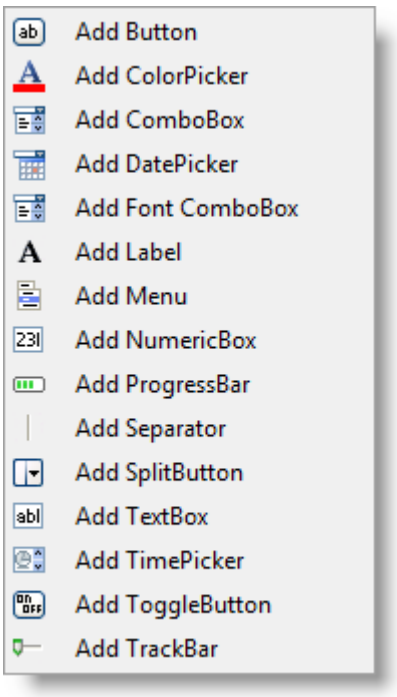
LeftPaneItems Floating Toolbar

The LeftPaneItems' floating toolbar allows you to add status bar items to the C1StatusBar control's left pane. Click the item to enable the item's floating toolbar. The LeftPaneItems' floating toolbar consists of the following button:

	**Actions:** Add status bar items to the C1StatusBar control's left pane.

Actions

Clicking the **Actions** button opens a list of available actions.



In-Place Text Editing

To edit the C1Ribbon element's label you can simply click on the text with your mouse pointer to highlight the text and then type in the appropriate text.

To use this feature, complete the following steps:

1. Select the C1Ribbon element's text, the tab for example, to highlight it. The tab's text is ready to be edited.

2. Enter a new tab name, for example, **Home**.
3. Press ENTER or click outside the editing box to accept the change.

ToolTip Editor

The **ToolTip Editor** is used to create rich ToolTips for RibbonTabs, RibbonGroups, and RibbonItems.

Accessing the ToolTip Editor

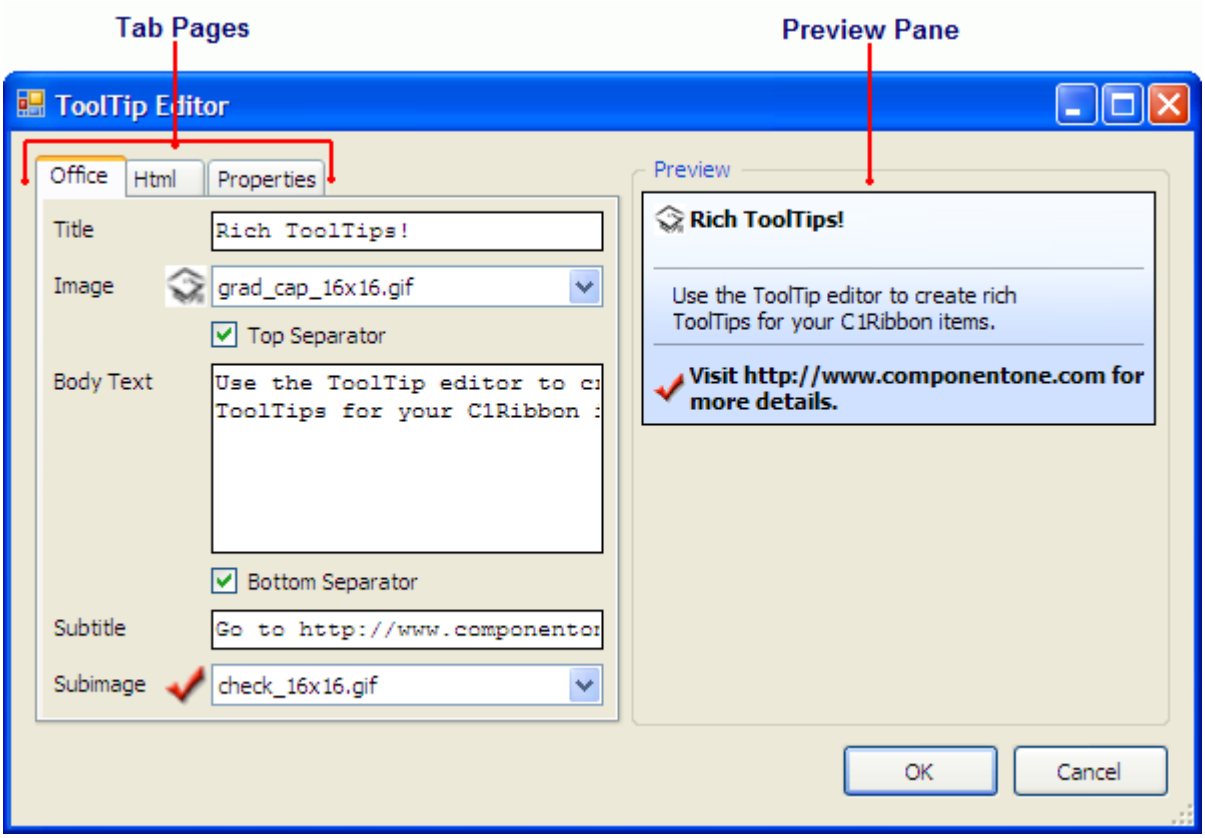
To open the **ToolTip Editor**, follow these steps:

1. Select the RibbonGroup, RibbonTab, or RibbonItem you wish to add the ToolTip to.

2. In the Properties window, locate the **ToolTip** property and press its ellipsis (...) button. The **ToolTip Editor** appears.

ToolTip Editor Layout

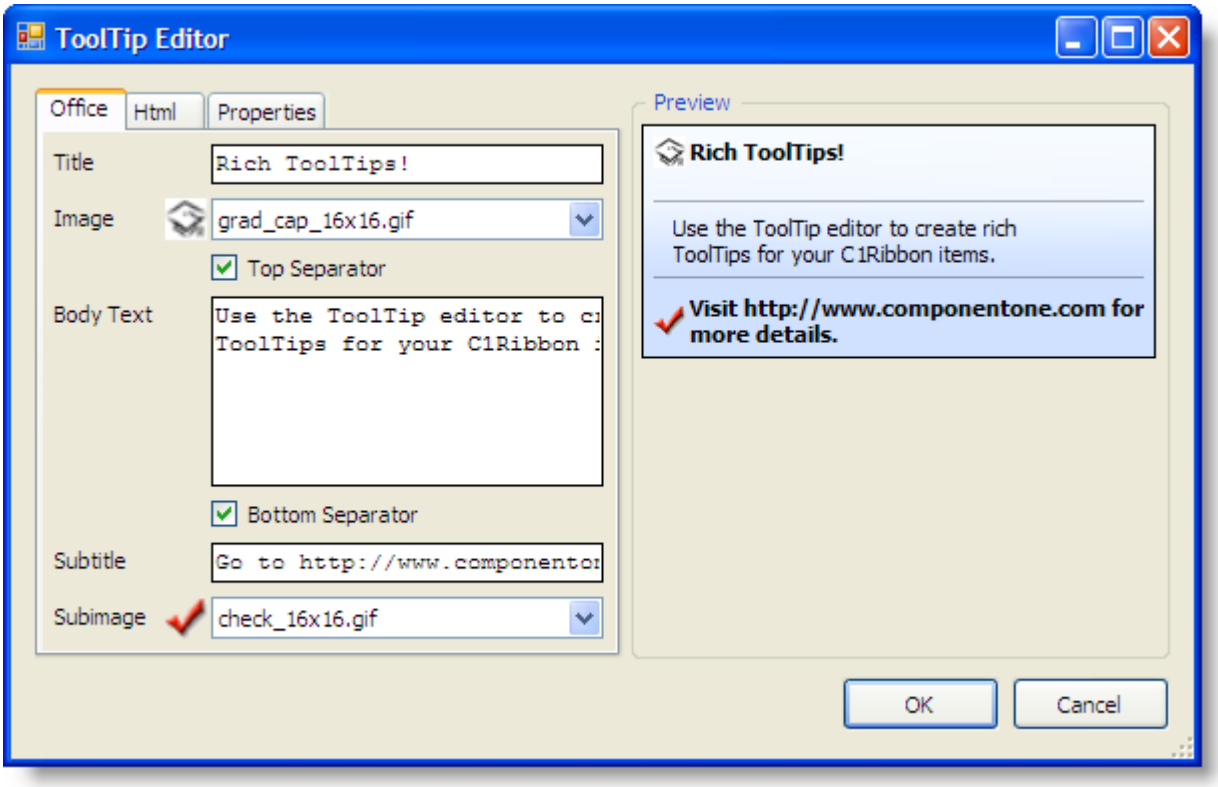
When the **ToolTip Editor** opens, you will find three tabs - Office, HTML, and Properties - on the left side of the dialog box. Each tab page features an editor where you can alter the content, source, or properties of a ToolTip. On the right side of the editor is a Preview pane, which provides a real-time preview of the ToolTip.



The following topics will detail the Office, HTML, and Properties tab pages.

Office Tab

The Office tab page is used to add content, graphics, and separator lines to your ToolTip. The editor automatically creates all of the HTML code behind the ToolTip, saving you time and effort. Any additions or changes you make to your ToolTip under the Office tab will be reflected in the Preview pane.



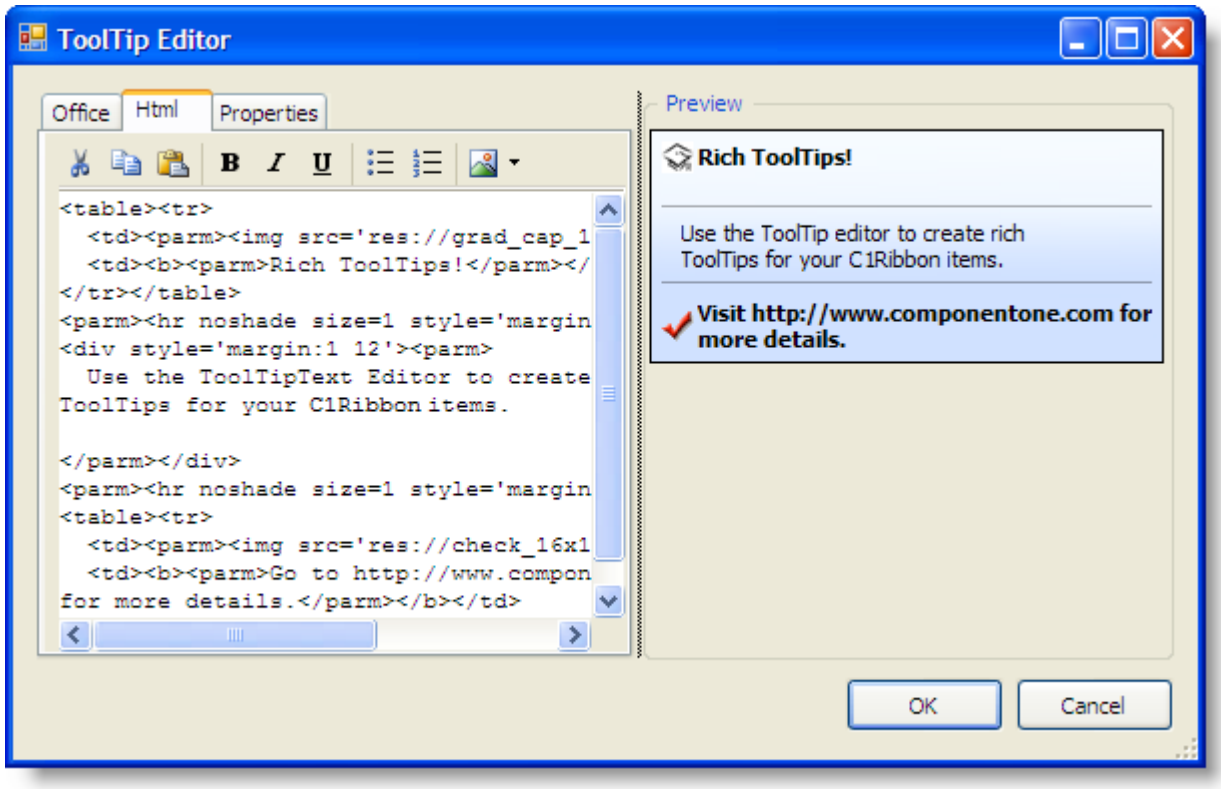
The following table describes the elements of the Office editor:

Element	Description
Title	The title element appears at the top of the ToolTip; it is bold by default.
Image	This image will appear over the body text of the ToolTip. By default, it will appear on the left-hand side.
Top Separator	Places a horizontal rule beneath the title and/or image. The horizontal rule is a dark grey.
Body Text	The body text holds the main content of the ToolTip; it is where the description and/or instructions for the tool appear. This will appear beneath the title and above the subtitle.
Bottom Separator	Places a horizontal rule beneath the the body text. The horizontal rule is a dark grey.
Subtitle	The subtitle element appears at the bottom of the ToolTip.
Subimage	This image will appear beneath the body text of the ToolTip. By default, it will appear on the left-hand side.

Html Tab

The Html tab page is used to add or modify the content of your ToolTip. When creating a ToolTip under the Html tab, you have more control over the design of the ToolTip. Any changes you make to your ToolTip under the **Html** tab will be reflected in the Preview pane.

The Html tab page consists of a toolbar of command buttons and an HTML editor window. You can use the command buttons to add markup tags to your project, or you can code by hand in the HTML editor window.



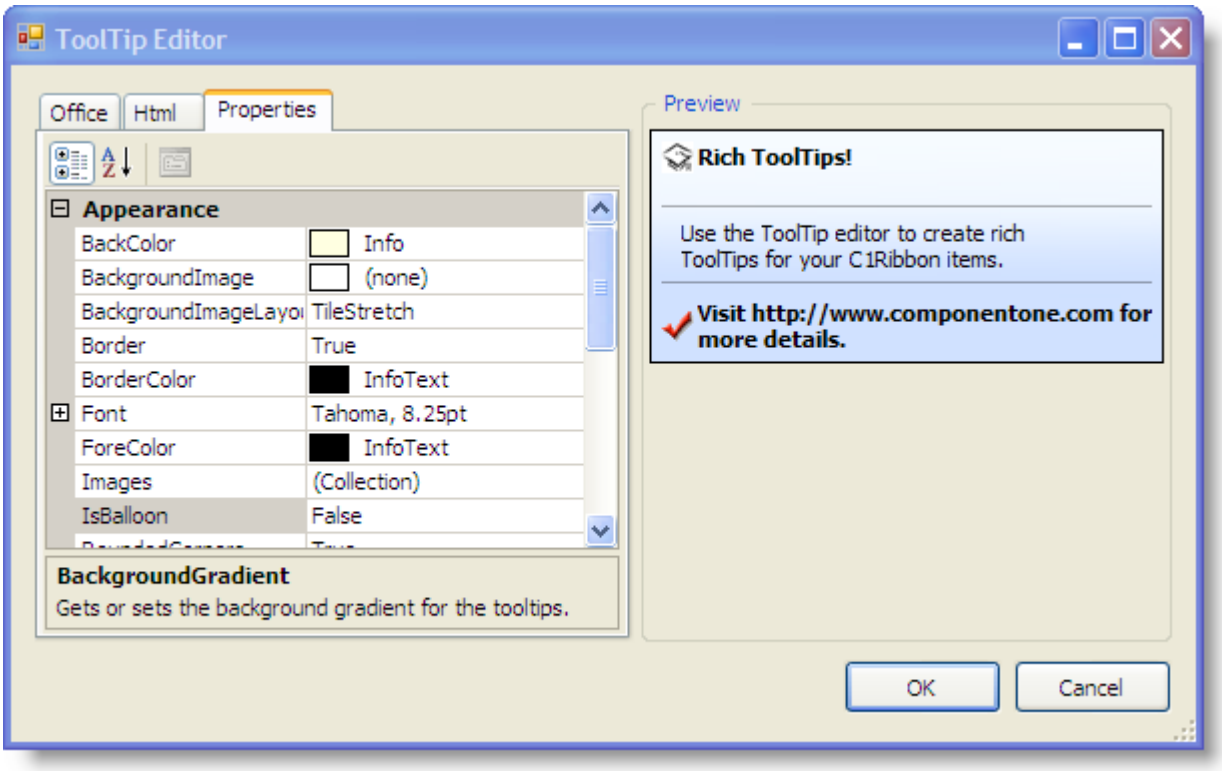
The following table provides descriptions for all of the command buttons under the Html tab:

Button	Name	Description
	Cut	Removes selected content from the HTML editor window and places it on the clipboard.
	Copy	Copies selected content from the HTML editor window to the clipboard.
	Paste	Pastes clipboard conent to the HTML editor window.
	Bold	Adds HTML tags for bold text () around selected text.
	Italics	Adds HTML tags for italicized text () around selected text.
	Underline	Adds HTML tags for underlined text (<u></u>) around selected text.
	Bulleted List	Adds HTML tags for a bulleted list ( ) to the HTML editor.
	Numbered List	Adds HTML tags for numbered list (to the HTML editor.

	Insert Image	Inserts an image into the tooltip using HTML tags (<img src = img.gif>).

Properties Tab

Within the Properties tab page is the Properties grid, which is used to customize the appearance and behavior of your ToolTips. Many of the changes that you make to the ToolTip properties will be displayed in the Preview pane.



The following table provides a list of properties that affect the appearance of a ToolTip:

Property	Description
BackColor	Sets background color of the ToolTip window.
BackgroundImage	Sets the background image displayed in the ToolTip window.
BackgroundImageLayout	Sets the background image layout to None (default), Tile, Center, Stretch, Zoom, or TileStretch.
Border	Sets whether the ToolTip window should display a solid border.
BorderColor	Sets the color of the ToolTip window border.
Font	Sets the default font used within the ToolTip.
ForeColor	Sets the foreground color of the ToolTip window.
Images	Gets a collection of images that may be shown in the ToolTips.
IsBalloon	Sets whether or not the ToolTip is shown in a balloon shape. If this is set to True, the ToolTip will resemble a dialogue bubble in a comic strip.

RoundedCorners	Sets whether or not the ToolTip window will have rounded corners.
Shadow	Sets whether or not the ToolTip window will display a dropshadow.
StripAmpersands	Sets whether ampersands in the text should be displayed or hidden.
UseFading	Sets whether a fade effect should be used when displaying a tooltip.

The following table provides a list of properties that affect the behavior of a ToolTip:

Property	Description
Active	Sets a value indicating whether or not the ToolTip is currently active.
AutomaticDelay	Sets the automatic delay for the rich ToolTip.
AutoPopDelay	Sets the period of time (in milliseconds) that the aToolTip will remain visible if the mouse pointer is stationary.
InitialDelay	Sets the time that passes before the ToolTip.
MaximumWidth	Sets the maximum width of the ToolTip.
ReshowDelay	Sets the length of time that must transpire before subsequent ToolTips appear as the mouse pointer moves from one control to another.
ShowAlways	Sets a value indicating whether a ToolTip window is displayed even when its parent control is not active.

Ribbon Appearance

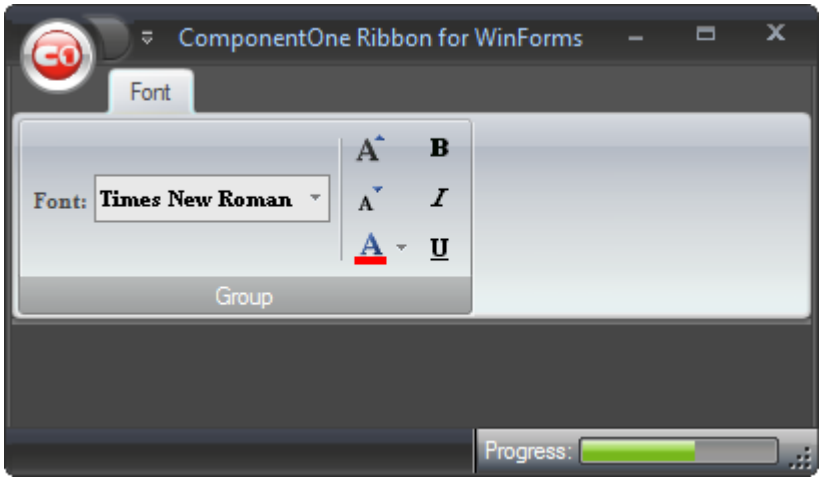
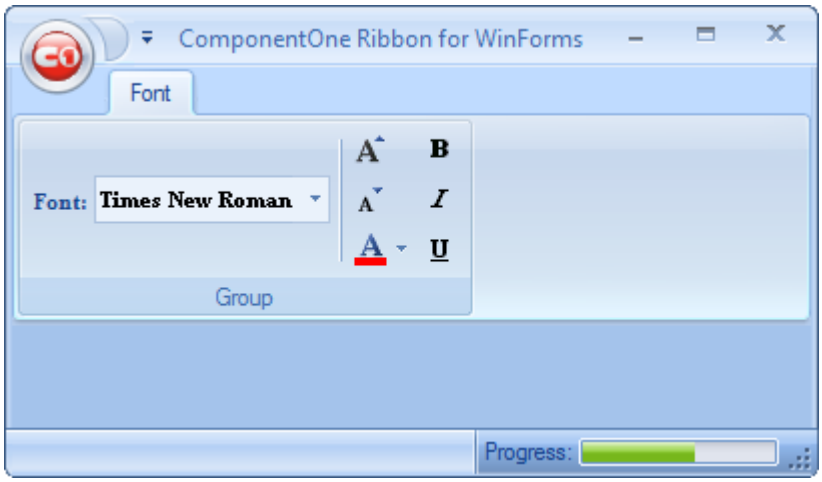
C1Ribbon is designed to make customization easy for you. C1Ribbon provides Office 2007/2010/2013/2016 and Windows 8 themes and over 700 stock images for Ribbon items.

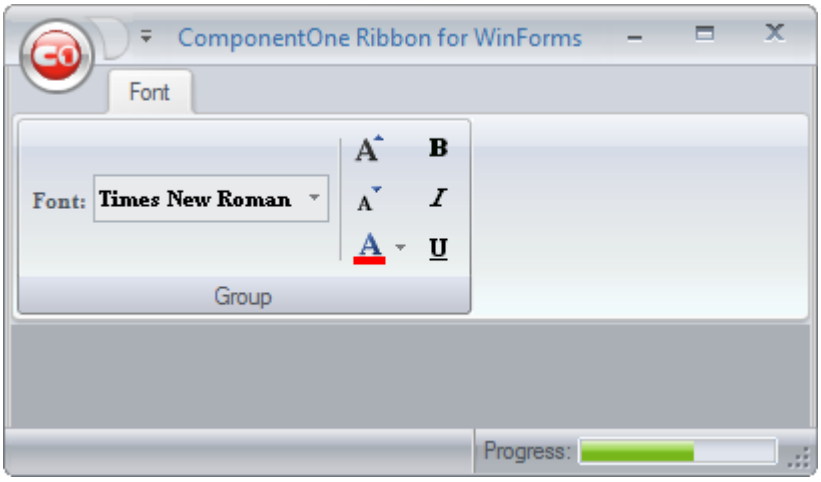
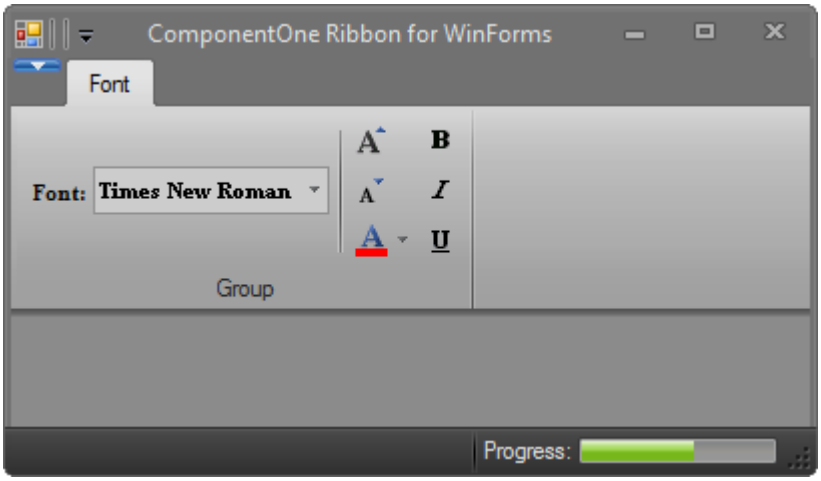
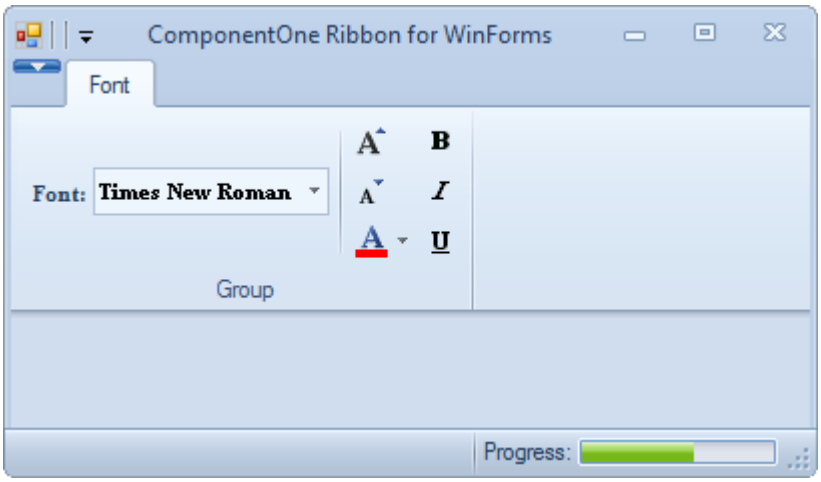
Save the Ribbon layout as an XML file and load the XML file for future usage with the Save/Load Template feature. Eliminate the need to start from scratch again!

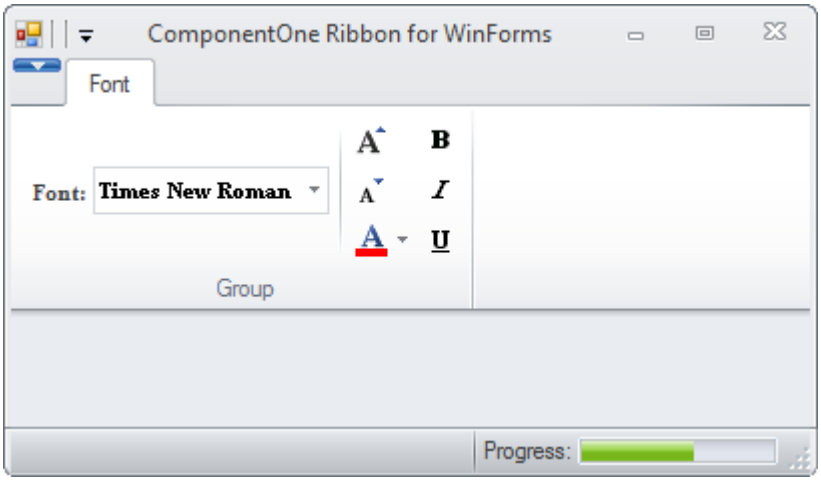
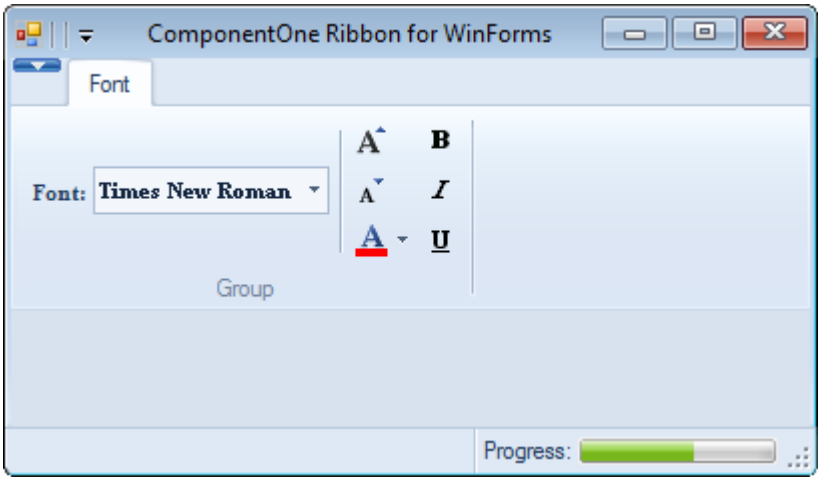
Visual Styles

Change C1Ribbon's visual style by selecting one of the preset Office-inspired and Windows-inspired styles. Once this is done, C1Ribbon will adopt the cleaner, more powerful, more efficient Windows Aero interface.

To learn how to change the visual style, see the Changing the Visual Style topic.

Visual Style Name	Image
Office2007Black	
Office2007Blue (default)	
Office2007Silver	

	
Office2010Black	
Office2010Blue	
Office2010Silver	

	
Windows7	

Images for Ribbon Items

C1Ribbon offers large (32x32) and small (16x16) preset images for the Ribbon items as well as the option to import custom images. You can display a small image for the Ribbon tab. The Ribbon items can display small or large images.

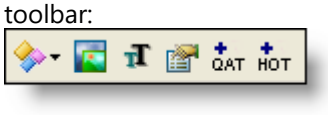
To change the Ribbon item's image, the following options are available:

> **Note:** The following tasks assume that you have added C1Ribbon control to your Ribbon Form and added a Ribbon item (for example, a toggle button) to the group.

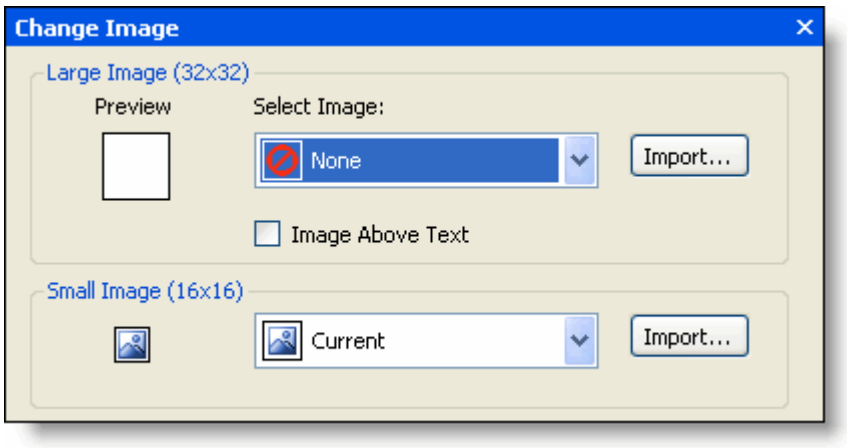
Using the Smart Designer

To add an image to the Ribbon item using the smart designer, complete the following steps:

1. Click the Ribbon item, for example, the toggle button to activate it and enable the toggle button's floating toolbar:



2. Click the Change Image button  . The Change Image dialog box appears.



3. Click the Small Image (16x16) drop-down arrow and select the Bold image.

Using the Properties Window

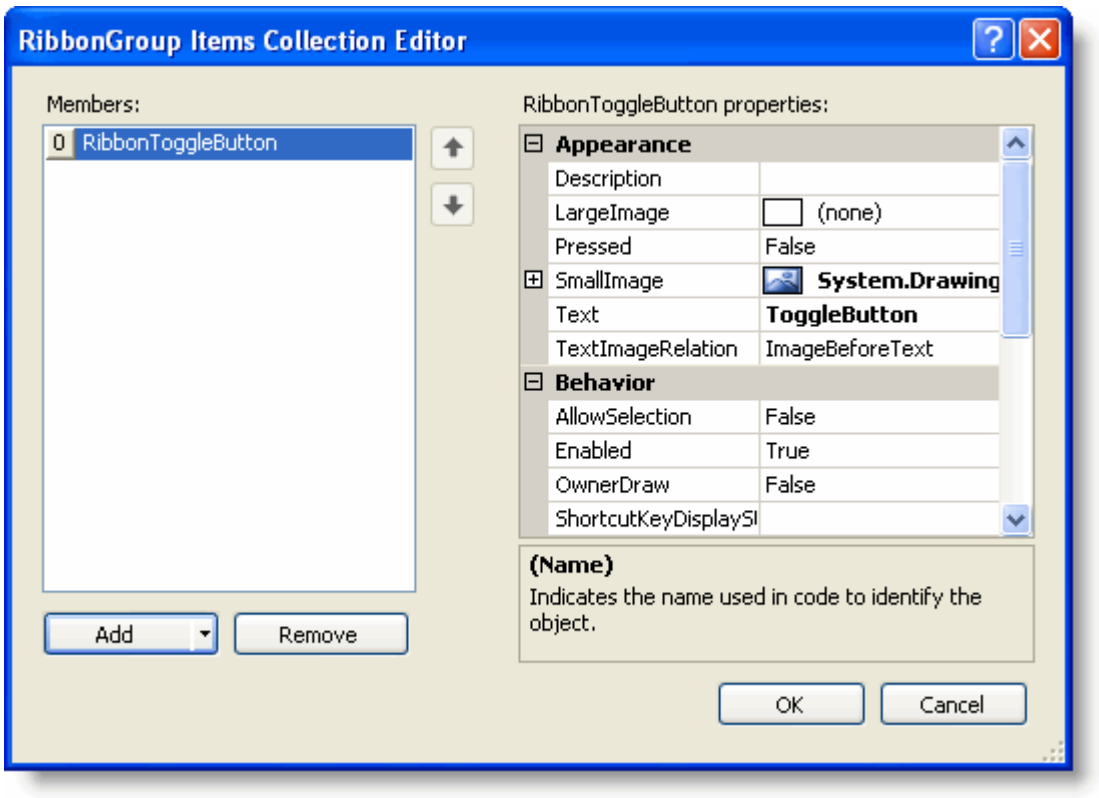
To add an image to the Ribbon item using the Properties window, complete the following steps:

1. Click the Ribbon item, for example, the toggle button to activate it and reveal its properties in the Properties window.
2. Locate the RibbonItem.SmallImage property, click the drop-down arrow, and click the second drop-down arrow to reveal the list of predefined small images.
3. Select **Bold**.

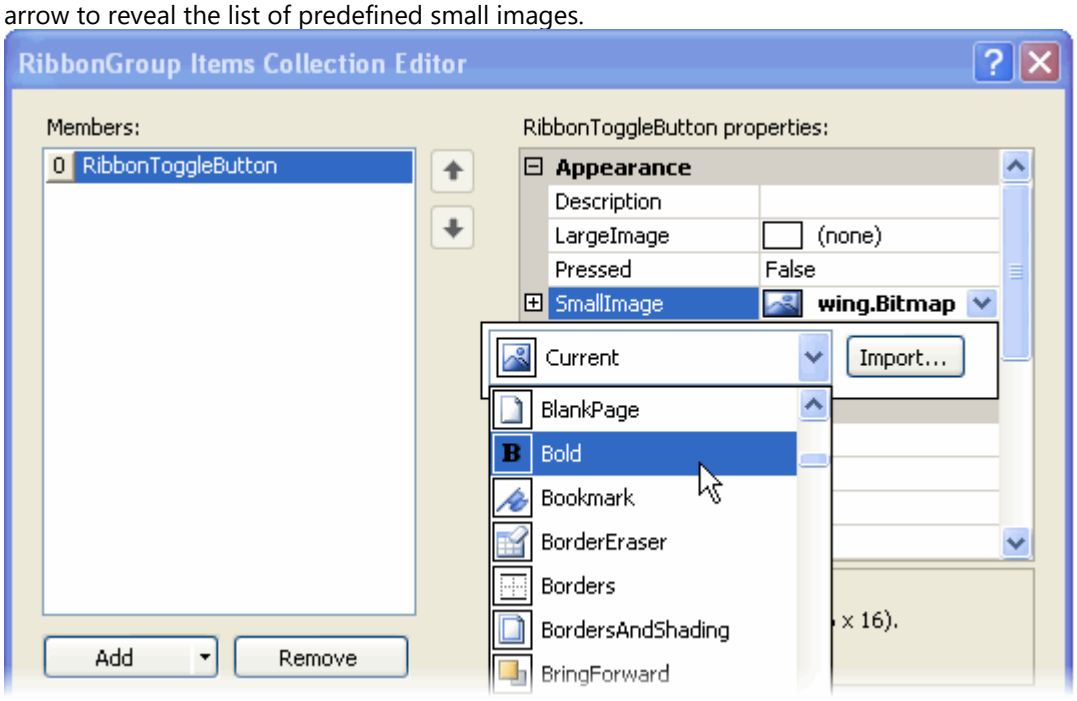
Using the Collection Editor

To add an image to the Ribbon item using the **RibbonGroup Items Collection Editor**, complete the following steps:

1. Click the group that contains the toggle button item. Clicking the group activates it.
2. In the group's Properties window, click on the **ellipsis** button next to the **Items** property.
   The **RibbonGroup Items Collection Editor** appears. The editor shows a list of **Members** and the available properties for each selected member. For this example, the toggle button is highlighted in the **Members** list:



3. Locate the RibbonItem.SmallImage property, click the drop-down arrow, and then click the second drop-down arrow to reveal the list of predefined small images.



4. Select **Bold**.

XML Serialization of the Ribbon Layout

The Load/Save Template feature allows end users to create a collection of ready-to-use templates for tabs and control groups. For example, many applications are likely to have the Clipboard and Font groups. Instead of creating a new Ribbon with these groups from scratch for each application, you can save the tabs and control groups as XML files and reuse them later. Then you can quickly import the XML and add the code to handle the commands.

Load Ribbon Template

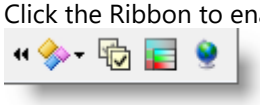
To import an existing XML file to the Ribbon, complete the following steps:

1. Click the Ribbon to enable the Ribbon's floating toolbar.

2. Click the **Actions** drop-down button.
3. From the list, select **Load Ribbon Template**.
4. The **Load Ribbon Template** dialog box appears. Browse to the location of the XML.
5. Click **Open**. The template loads the Ribbon.
6. Add the code to handle the commands.

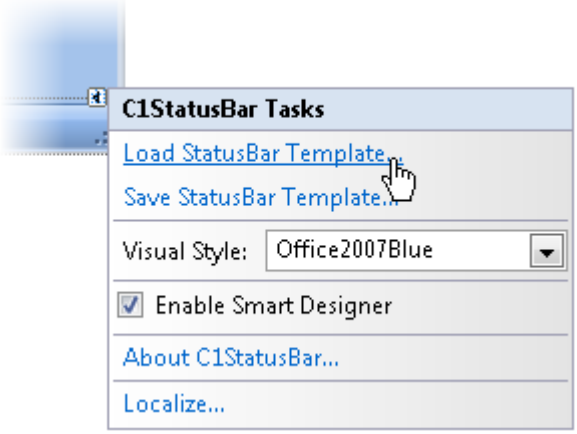
Save Ribbon Template

To save a Ribbon template, complete the following steps:

1. Create a Ribbon group and add Ribbon items to the group
2. Click the Ribbon to enable the Ribbon's floating toolbar.

3. Click the **Actions** drop-down button.
4. From the list, select **Save Ribbon Template**.
5. The **Save Ribbon Template** dialog box appears. Enter the XML's name in the **File name** text box and browse to the save location for the XML.
6. Click **Save**.

Load StatusBar Template

To load a status bar template, complete the following steps:

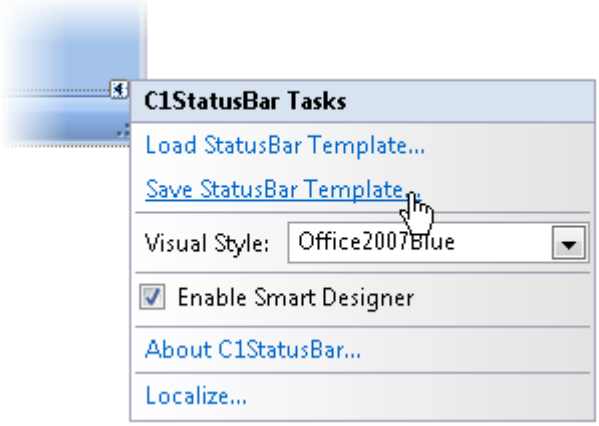
1. Click the C1StatusBar smart tag (▶) to enable the **C1StatusBar Tasks** menu.
2. Select **Load StatusBar Template** from the menu:


3. The **Load StatusBar Template** dialog box appears. Browse to the location of the XML.
4. Click **Open**. The template loads the status bar items.
5. Add code to handle the commands.

Save StatusBar Template

To save a status bar template, complete the following steps:

1. Add status bar itemsto the status bar.
2. Click the C1StatusBar smart tag (▶) to enable the **C1StatusBar Tasks** menu.
3. Select **Save StatusBar Template** from the menu:

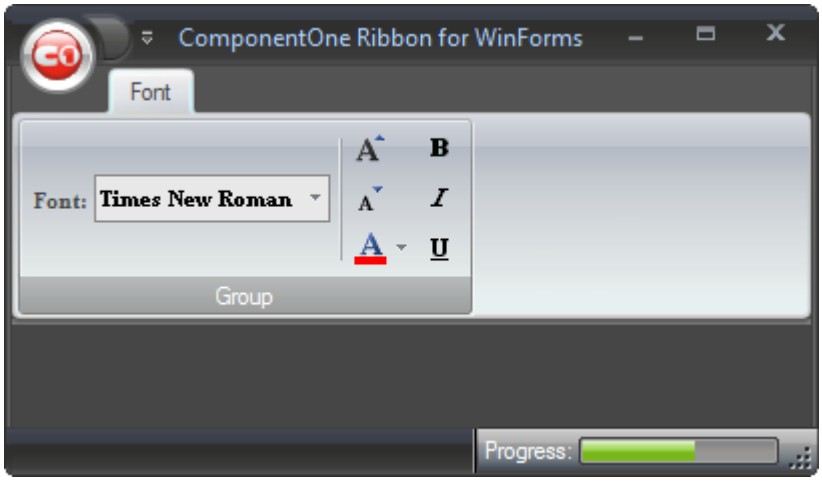


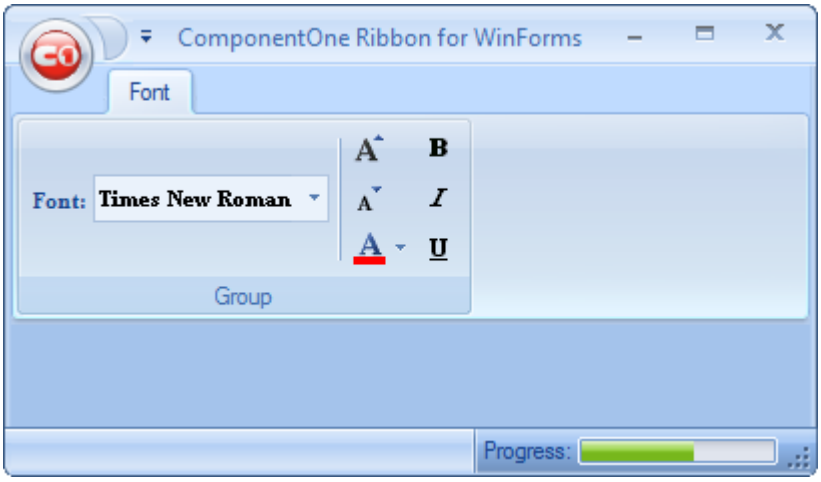
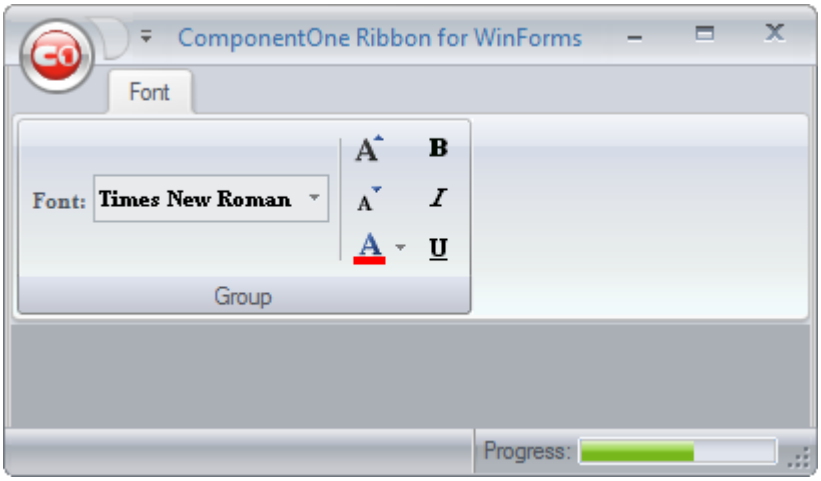
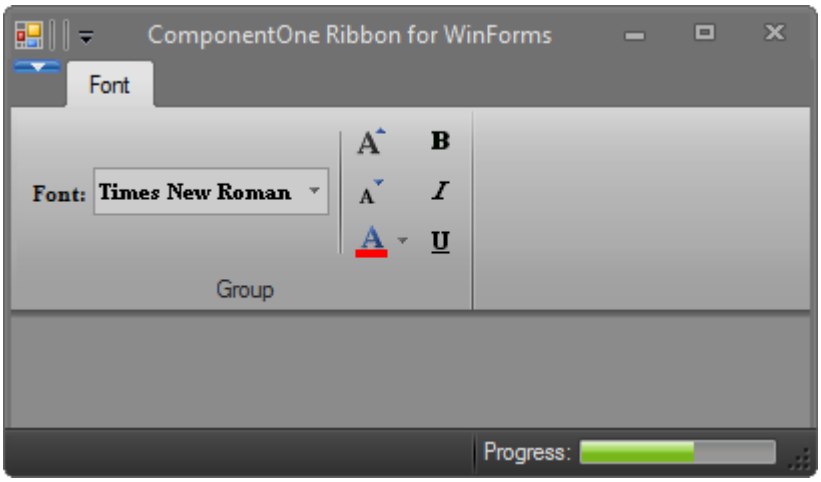
4. The **Save StatusBar Template** dialog box appears. Enter the XML's name in the **File name** text box and browse to the save location for the XML.
5. Click **Save**.

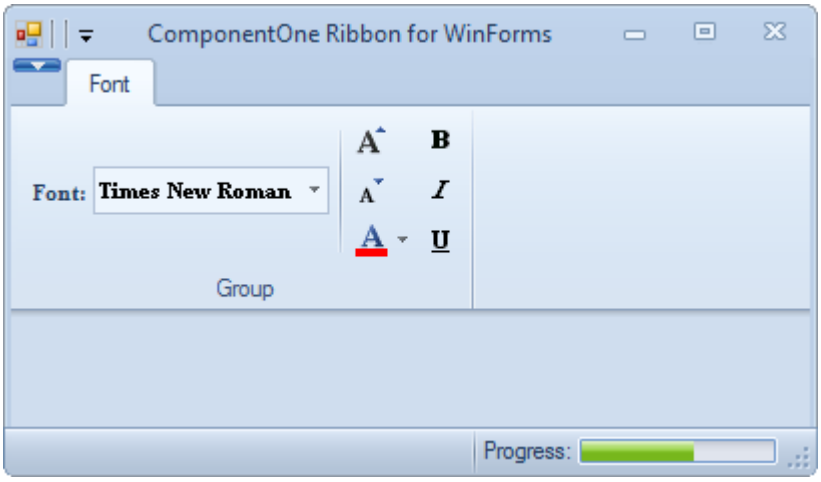
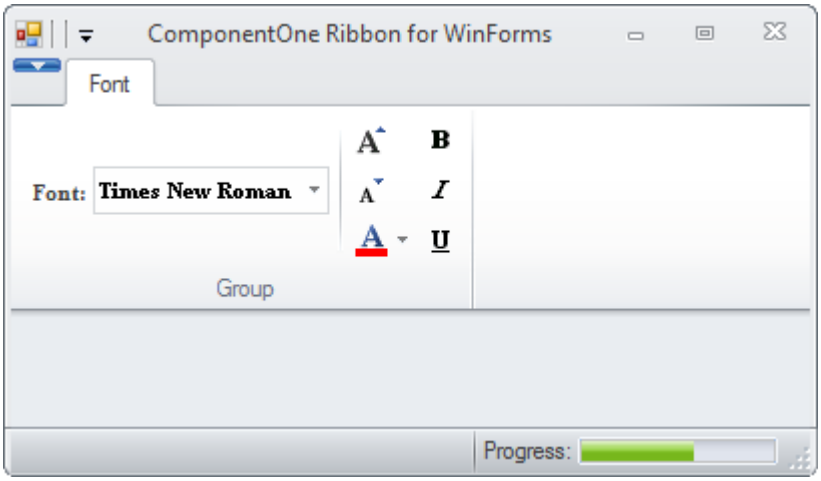
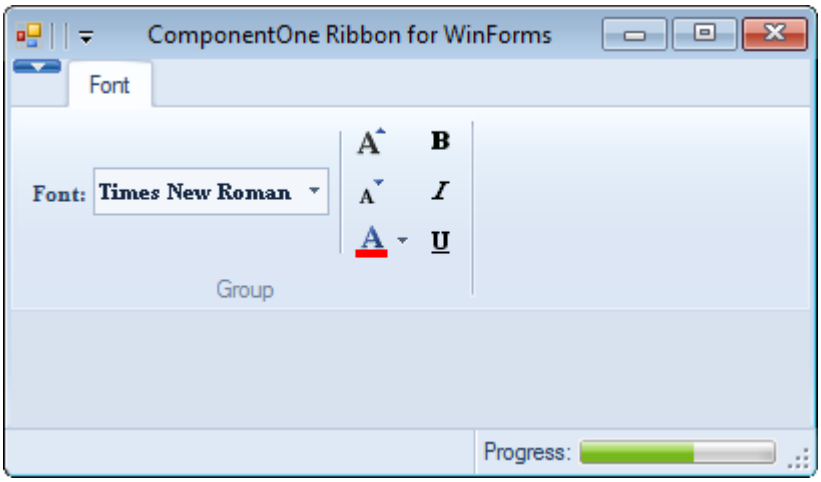
Themes

Theme C1Ribbon control in your WinForms application visual style by selecting one of the preset Office-inspired and Windows-inspired styles. Once this is done, C1Ribbon will adopt the cleaner, more powerful, more efficient Windows Aero interface.

To learn how to change the visual style, see the Changing the Visual Style topic.

Visual Style Name	Image
Office2007Black	
Office2007Blue (default)	

	
Office2007Silver	
Office2010Black	
Office2010Blue	

	
Office2010Silver	
Windows7	

Run-Time Interaction

At run time, end users can customize the Ribbon to fit their needs. For example, the user can minimize the Ribbon to save space on the page, use key tips to interact with the Ribbon through the keyboard, customize the Quick Access Toolbar (QAT) by adding or removing commands as needed as well as move the QAT below the Ribbon. See the following topics for details on interacting with the Ribbon at run time.

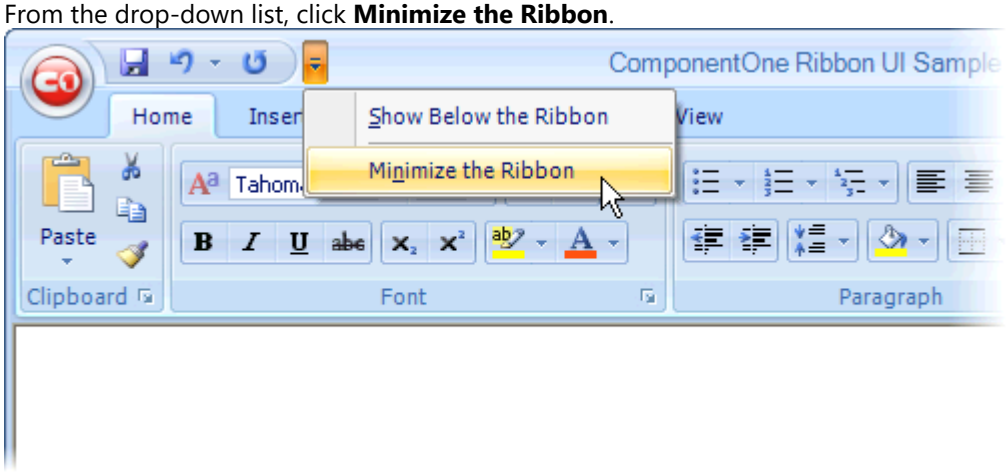
Minimizing the Ribbon

If the Ribbon is cluttering your screen, you can make some tabs visible only when they are needed. To make more space available, you can easily minimize the Ribbon using the Quick Access Toolbar (QAT), double-clicking the active tab, or using the keyboard shortcut.

Minimize the Ribbon

Option 1: Use the QAT

1. Click the drop-down arrow next to the QAT.
2. From the drop-down list, click **Minimize the Ribbon**.



3. To use the Ribbon while it is minimized, click the tab you want to use, and then click the option or command you want to use.
   For example, with the Ribbon minimized, you can select text in your document, click the **Home** tab, and then in the **Font** group, click the size of the text you want. After you click the text size you want, the Ribbon goes back to being minimized.

Option 2: Double-click the active tab

To quickly minimize the Ribbon, double-click the name of the active tab. Double-click a tab again to restore the Ribbon.

Restore the Ribbon

To restore the Ribbon, complete the following steps:

1. Click the drop-down arrow next to the QAT.
2. From the drop-down list, click **Minimize the Ribbon**.

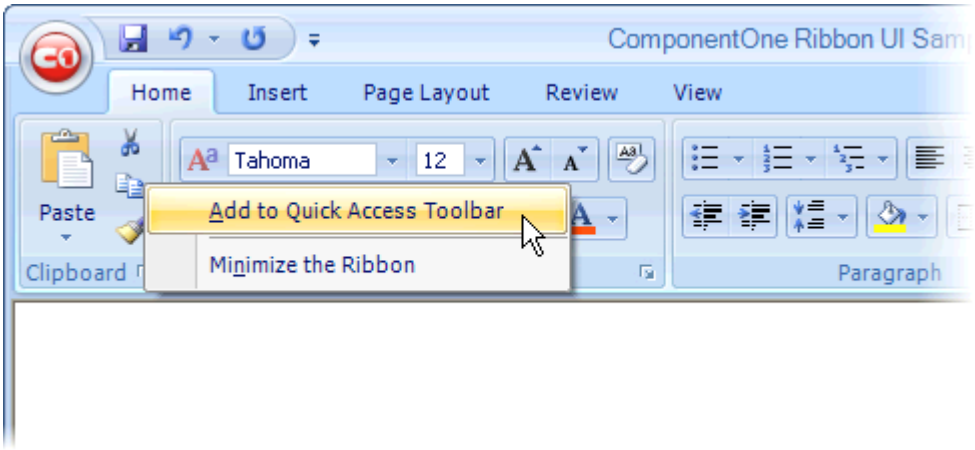
Customizing the Quick Access Toolbar

Customizing the Quick Access Toolbar (QAT) at run time is simple; with just a mouse-click you can add items to the QAT as well as move the position of the QAT above or below the Ribbon.

To Add Items to the Quick Access Toolbar at Run Time

At run time, you can add items to the Quick Access Toolbar (QAT). As you work, you may find that you use some commands frequently. For easier access, the commands you use the most can be added to the QAT.

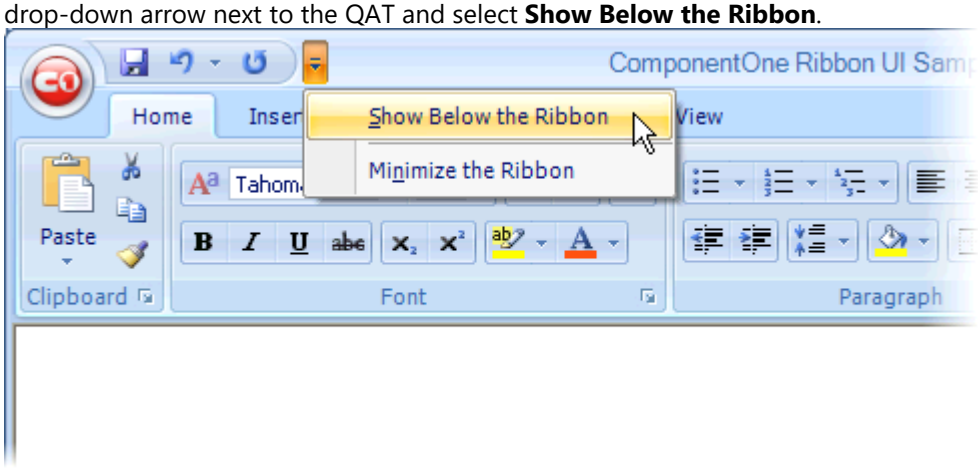
To add a command to the QAT at run time, simply right-click the item and select **Add to Quick Access Toolbar** from the context menu.



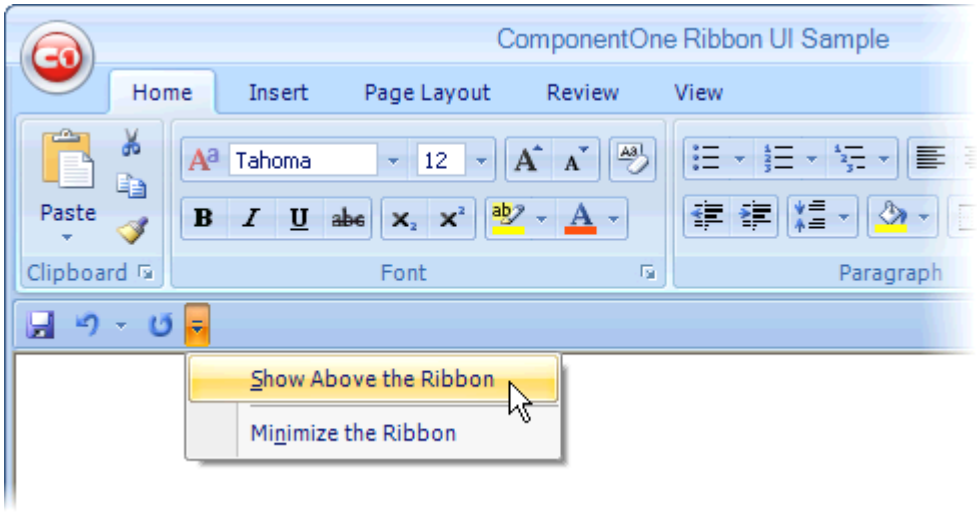
To Move the Quick Access Toolbar at Run Time

At run time, you can display the Quick Access Toolbar (QAT) above or below the Ribbon. Place the QAT in the most accessible location:

- **Show Below the Ribbon** – To move the Quick Access Toolbar (QAT) below the Ribbon at run time, click the drop-down arrow next to the QAT and select **Show Below the Ribbon**.



- **Show Above the Ribbon** – To move the QAT back to its location above the Ribbon, click the drop-down arrow next to the QAT and select **Show Above the Ribbon**.



Ribbon 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 ComponentOne Studio.

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

Documents\ComponentOne Samples\WinForms

Visual Basic and C# Samples

Sample	Description
AddQatItems	The sample shows how to add buttons to the Quick Access Toolbar (QAT) from code.
AddRibbonItems	The sample shows how to add a C1Ribboncontrol to the form and populate it with various items programmatically. The items added include toggle buttons, radio buttons, and a combo box. The sample also demonstrates the usage of the Ribbon toolbar for arranging the items into rows.
CreateAppMenu	The sample shows how to create an Application menu containing the New, Open, Save, Save As, and Print commands in the left menu pane, and a Recent Document list on the right pane.
WordPad	The sample demonstrates how to use C1Ribbonto build a simple Office 2007 style text editor with the following capabilities: · Text editing and formatting · Clipboard functions · Undo/Redo functions · View zooming The sample shows how to: · Populate a RibbonComboBox with names of all fonts installed in the system. · Manage the input focus so that it goes back to the text area after the user is done using the Ribbon. · Use the **RibbonButton.Click** event to handle button clicks. · Use the RibbonCombobox.CommitChanges event. · Update the **Enabled** and **Pressed** properties of RibbonButton based on current text selection. · Use the application settings to save and retrieve the QAT position and the visual style of the Ribbon. · Build and maintain a list of recent documents; persist the list to application settings. · Launch a dialog box with the launcher button of a Ribbon item group.

Ribbon for WinForms Task-Based Help

The task-based help assumes that you are familiar with programming in Visual Studio. By following the steps outlined in the help, you will be able to utilize the features of C1Ribbon.

Each task-based help topic also assumes that you have created a new Windows application project and have placed a C1Ribbon control on the form. For additional information, see Creating a .NET Project and Adding ComponentOne Controls to a Project.

Ribbon Forms Title Bar Caption Alignment

The Ribbon Forms title bar displays the form's caption, which can be aligned as per user requirements. The caption can be horizontally aligned on the top left, centre or top right corners of the title bar of control. CaptionAlignment property of C1Ribbon control can be used to change the horizontal alignment of the Ribbon Forms title bar caption. To change the title bar caption alignment, add the following code to your project:

To write code in Visual Basic

Visual Basic

```
Me.Ribbon.CaptionAlignment = HorizontalAlignment.Left
```

To write code in C#

C#

```
this.Ribbon.CaptionAlignment = HorizontalAlignment.Left;
```

Adding Ribbon Items

When you add the C1Ribbon control to the Windows Form, the Ribbon appears at the top of the form with a tab and a group within the tab. To build on the Ribbon, you must add items. The following topics demonstrate how to add items to the Quick Access Toolbar, add items to the configuration toolbar, add a tab to the Ribbon, add a group to the tab, and add items to the group.

Tip: Ultimately, you will have to write event handling code that performs a specific action on the form's content whenever a button is clicked or a selection is made from a combo box or drop-down menu. Therefore, it is a good idea to assign meaningful names to Ribbon controls as you create them, making your code easier to understand and maintain.

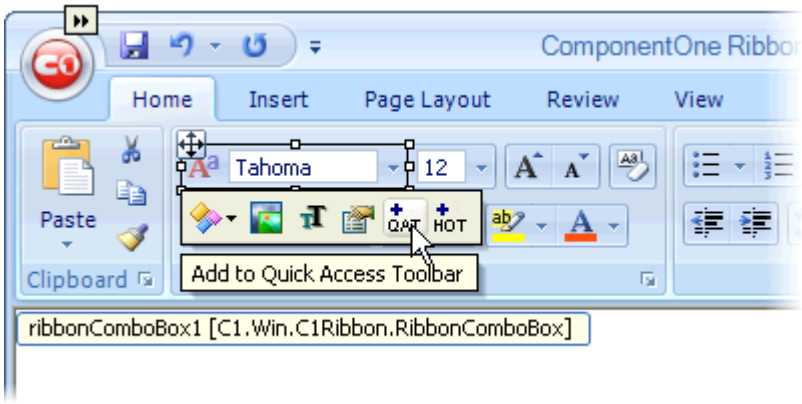
Adding Items to the Quick Access Toolbar

The Quick Access Toolbar (QAT) can grow to accommodate as many commands as needed. To add items to the QAT, use the smart designer or add code. Each option is described below.

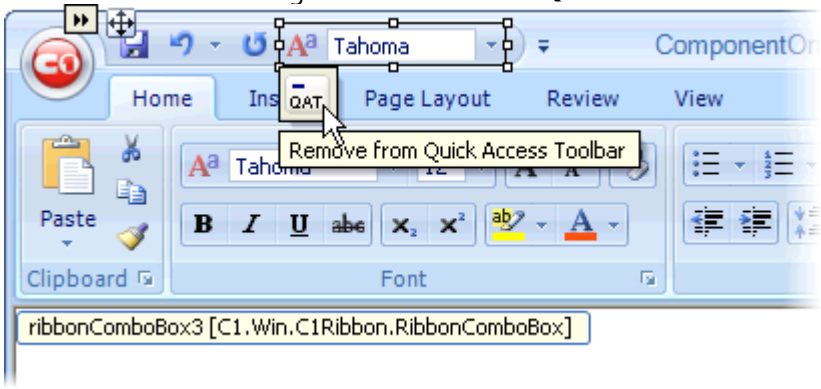
To Add QAT Items Using the Smart Designer

Complete the following steps:

1. Click the Ribbon item on the Ribbon to enable its floating toolbar.
2. Select the **+QAT** button.



The combo box is added to the QAT. You can remove the item from the QAT by selecting the combo box item, which enables the floating toolbar. Select the **–QAT** button:



To Add QAT Items Programmatically

> **Note:** In the following example embedded resources containing the following images are used: *save.png*, *undo.png*, and *repeat.png*. To embed a resource, from the **Project** menu, choose ***YourProjectName* Properties**. From the **Resources** tab, select **Add Resource** and choose to add an existing file or add a new one.

To add Ribbon items (for example, Save, Undo, and Repeat) to the QAT, add the following code to your project:

To write code in Visual Basic

Visual Basic

```
' include the Imports directive for the namespace
Imports C1.Win.C1Ribbon

Private Sub Form1_Load(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles MyBase.Load
    C1Ribbon1.Qat.Items.Add(New RibbonButton(My.Resources.Resources.save))
    C1Ribbon1.Qat.Items.Add(New RibbonButton(My.Resources.Resources.undo))
    C1Ribbon1.Qat.Items.Add(New RibbonButton(My.Resources.Resources.repeat))
End Sub
```

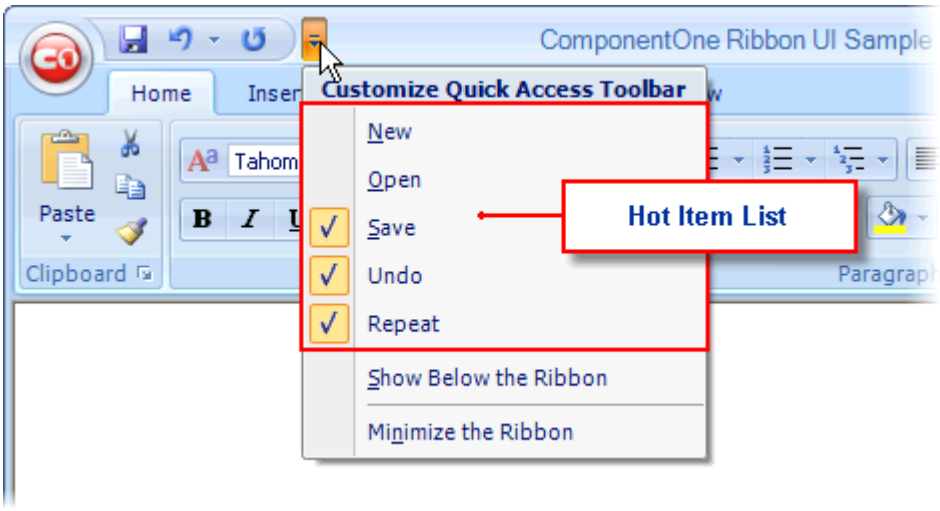
To write code in C#

C#

```
// include the using directive for the namespace
using C1.Win.C1Ribbon;

private void Form1_Load(object sender, System.EventArgs e)
{
    C1Ribbon1.Qat.Items.Add (new RibbonButton (Properties.Resources.save));
    C1Ribbon1.Qat.Items.Add (new RibbonSplitButton (Properties.Resources.undo));
    C1Ribbon1.Qat.Items.Add (new RibbonButton (Properties.Resources.repeat));
}
```

Adding Items to the Hot List

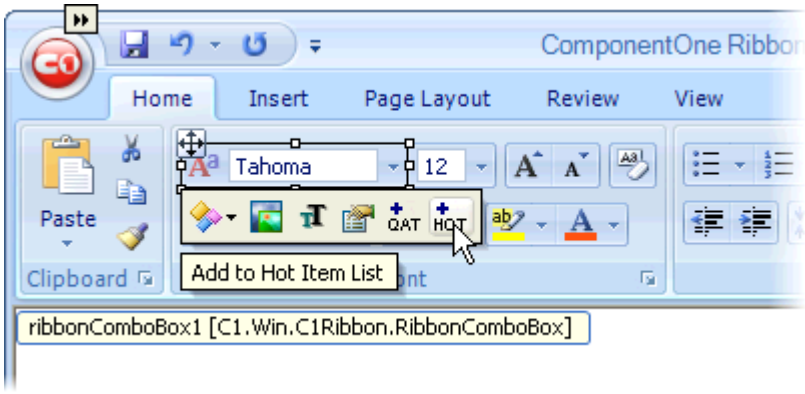
The hot item list, which includes a list of items to add to the Quick Access Toolbar at run time, is available at run time by clicking the Quick Access Toolbar's drop-down arrow. Items added to the hot item list at design time appear in the Customize QAT menu at run time:



To Add Hot List Items Using the Smart Designer

Complete the following steps:

1. Click the Ribbon item on the Ribbon to enable its floating toolbar.
2. Select the **+HOT** button.



The combo box is added to the hot list.

To Add Hot List Items Programmatically

Note: In the following example embedded resources containing the following images are used: *NewBtn.png* and *OpenBtn.png*. To embed a resource, from the **Project** menu, choose ***YourProjectName* Properties**. From the **Resources** tab, select **Add Resource** and choose to add an existing file or add a new one.

Complete the following steps:

1. Add the following code to your project:

To write code in Visual Basic

Visual Basic

```
' type the Imports directive for the namespace
Imports C1.Win.C1Ribbon

Private Sub Form1_Load(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles MyBase.Load
    ' add items to the hot list
    Dim NewFileRibbonBtn As RibbonButton = New RibbonButton()
    Dim OpenFileRibbonBtn As RibbonButton = New RibbonButton()

    C1Ribbon1.Qat.MenuItems.Add(NewFileRibbonBtn)
    C1Ribbon1.Qat.MenuItems.Add(OpenFileRibbonBtn)

    ' set some properties for the hot list items
    NewFileRibbonBtn.SmallImage = My.Resources.Resources.NewBtn
    NewFileRibbonBtn.Text = "&New"
    OpenFileRibbonBtn.SmallImage = My.Resources.Resources.OpenBtn
    OpenFileRibbonBtn.Text = "&Open"
End Sub
```

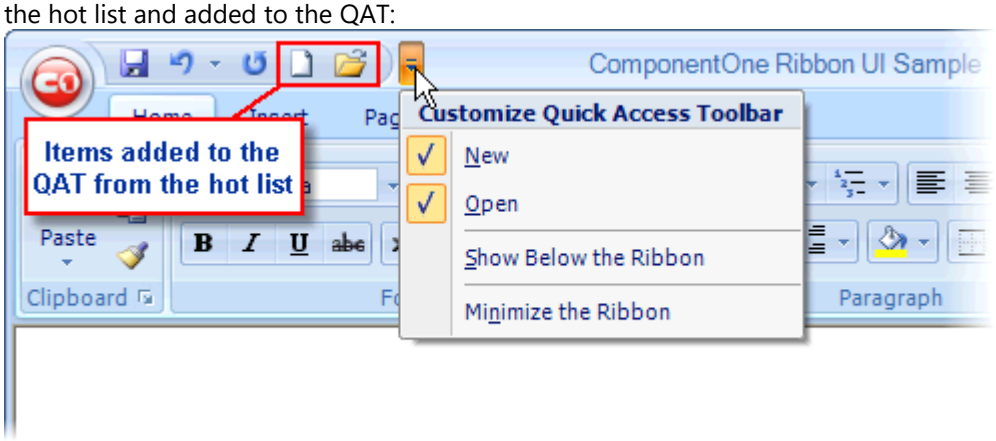
To write code in C#

C#

```
// type the using directive for the namespace
using C1.Win.C1Ribbon;

private void Form1_Load(object sender, System.EventArgs e)
{
    // add items to the hot list
    RibbonButton NewFileRibbonBtn = new RibbonButton();
    RibbonButton OpenFileRibbonBtn = new RibbonButton();

    C1Ribbon1.Qat.MenuItems.Add(NewFileRibbonBtn);
    c1Ribbon1.Qat.MenuItems.Add(OpenFileRibbonBtn);

    // set some properties for the hot list items
    NewFileRibbonBtn.SmallImage = Properties.Resources.NewBtn;
    NewFileRibbonBtn.Text = "&New";
```

```
    OpenFileRibbonBtn.SmallImage = Properties.Resources.OpenBtn;
    OpenFileRibbonBtn.Text = "&Open";
}
```

2. Click the **Start Debugging** button to run the application and click the QAT drop-down arrow to reveal the list of items in the hot list.
3. Select an item from the hot item list to add it to the QAT. The **Open** and **New** buttons have been selected in the hot list and added to the QAT:



4. To remove the items from the QAT simply deselect each item in the hot list.

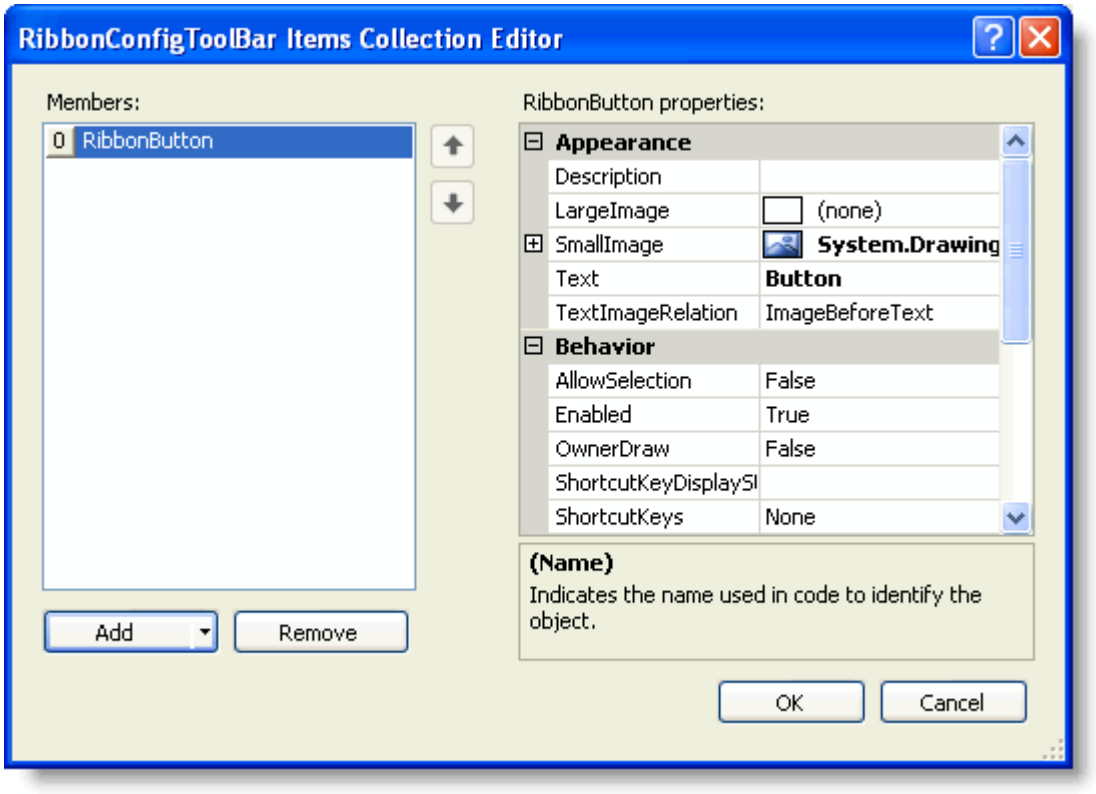
Adding Items to the Configuration Toolbar

The configuration toolbar (RibbonConfigToolBar) can contain as many commands as needed. To add Ribbon items to the configuration toolbar, complete the following steps:

Adding Items to the Configuration Toolbar Using the Properties Window

Complete the following steps:

1. Click the Ribbon to reveal the list of properties in the Properties window.
2. Expand the **ConfigToolBar** property node, select the **Items** property and click the **ellipsis** button at the right side of the **(Collection)**.
   The **RibbonConfigToolBar Items Collection Editor** appears.
3. Click the **Add** drop-down button and select **RibbonButton** from the list. The item is added to the **Members** list.



4. In the RibbonButton's Properties window, set the following properties:
   - By default, the RibbonButton.Text property is set to **Button**. Delete the text.
   - Click the RibbonItem.SmallImage property drop-down button, then click the second drop-down button and choose the **Help** image from the list.
5. Click **OK** to close the collection editor.

Adding Items to the Configuration Toolbar Programmatically:

Note: The following example uses an embedded resources containing an image. To embed a resource, from the **Project** menu, choose ***YourProjectName* Properties**. From the **Resources** tab, select **Add Resource** and choose to add an existing file or add a new one.

To add items to the configuration toolbar programmatically, add the following code to your Form_Load event:

To write code in Visual Basic

Visual Basic

```
my.C1Ribbon1.ConfigToolBar.Items.Add(new RibbonButton(Properties.Resources.question))
```

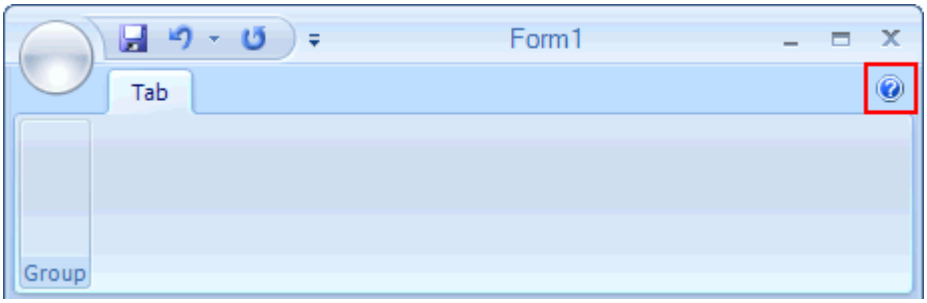
To write code in C#

C#

```
this.C1Ribbon1.ConfigToolBar.Items.Add(new
RibbonButton(Properties.Resources.question));
```

This topic illustrates the following:

The following configuration toolbar provides quick access to the **Help** button:

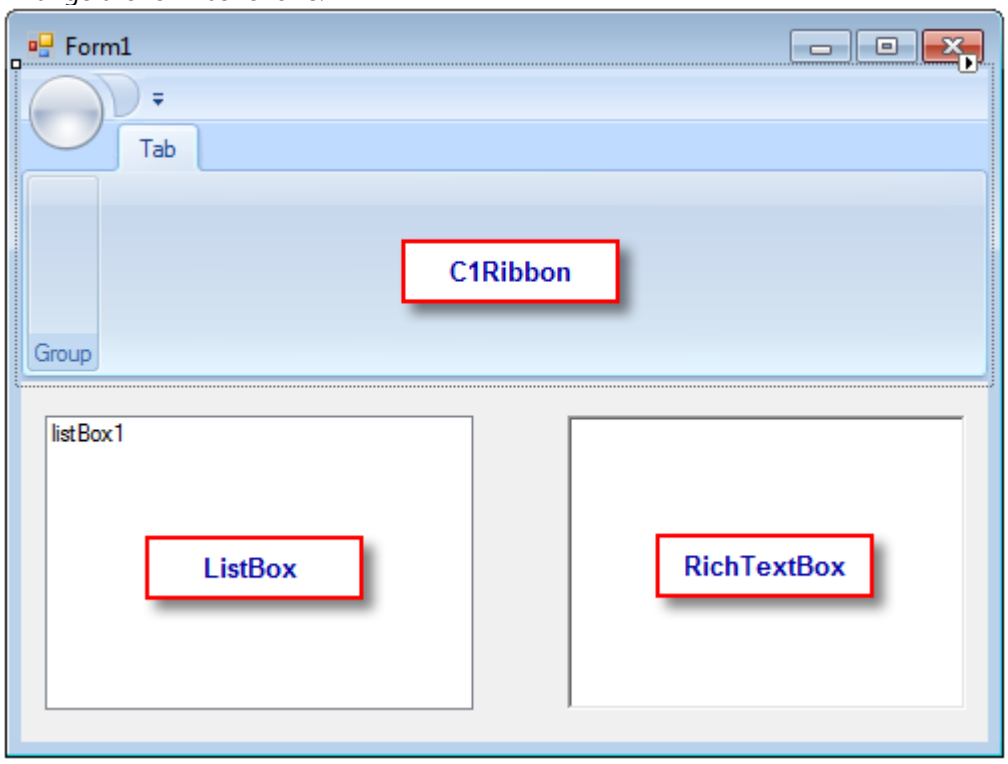


Adding a Contextual Tab to the Ribbon

The actions that users need to use regularly should always be available on the Ribbon control, where the user can access them within a few clicks. However, certain actions are exclusive to a certain element; for example, an action that allows you to format the font of text may only be necessary when users are editing text in a **RichTextBox**. In the aforementioned case, it would be beneficial to place this action (and all other actions specific to RichTextBoxes) under a contextual tabs that only appears when users have selected the **RichTextBox**. This topic demonstrates how to add a contextual tab group that appears only when a **RichTextBox** is selected.

Complete the following steps:

1. From the Visual Studio Toolbox, add the following controls to your Windows form:
   - (1) C1Ribbon control
   - (1) **ListBox** control
   - (1) **RichTextBox** control
2. Arrange the form as follows:



3. Hover over the Ribbon to enable the floating toolbar, click the **Actions** button  , and select **Add Contextual Tab Group**.
4. Select **View | Code** to enter Code view and add the following code to your project:

   To write code in Visual Basic

   Visual Basic

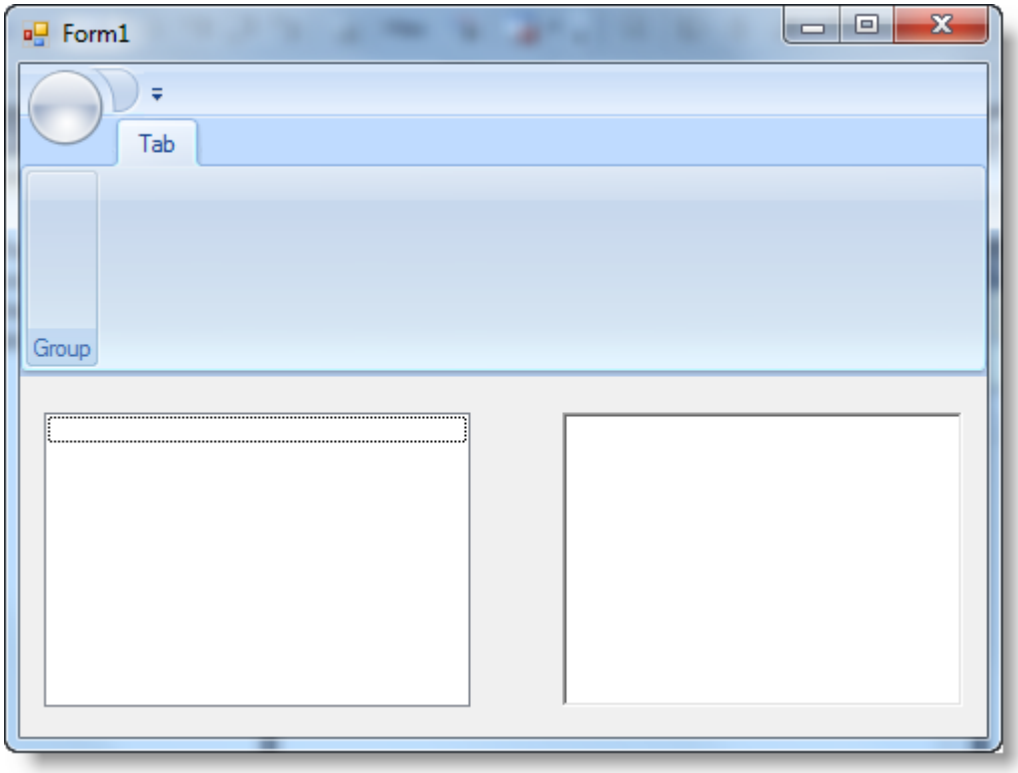
   ```
   Private Sub richTextBox1_Enter(sender As Object, e As EventArgs)
      ribbonContextualTabGroup1.Visible = True
   End Sub
   Private Sub richTextBox1_Leave(sender As Object, e As EventArgs)
      ribbonContextualTabGroup1.Visible = False
   End Sub
   ```

   To write code in C#

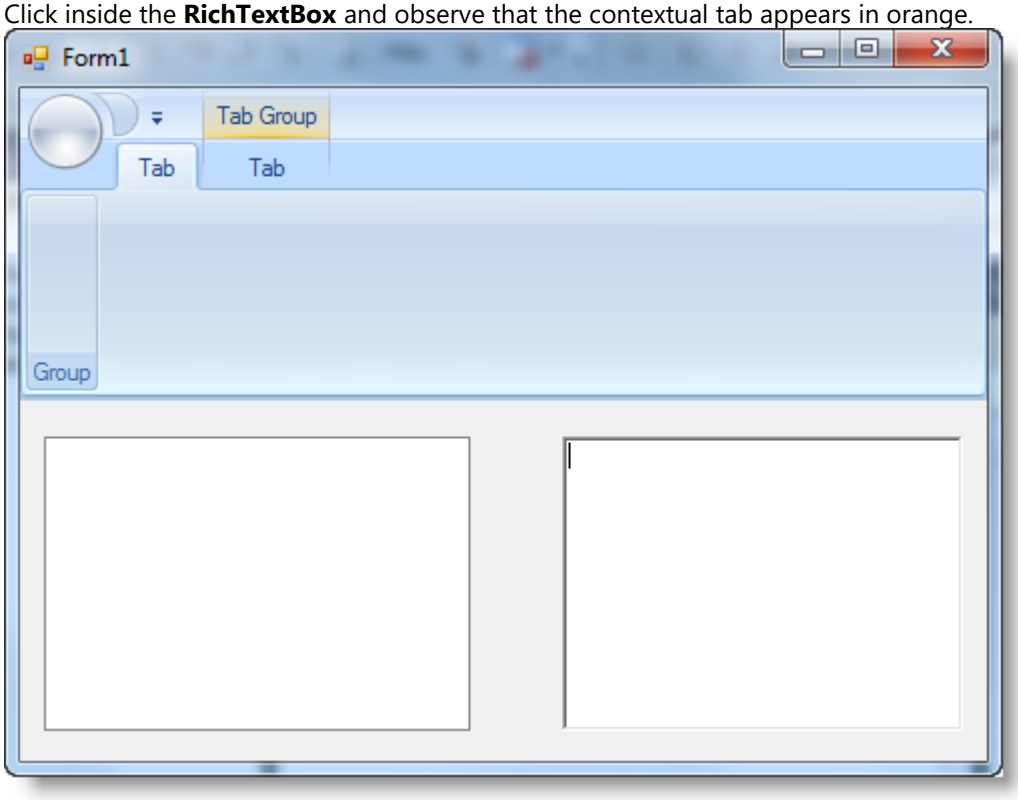
   C#

   ```
   private void richTextBox1_Enter(object sender, EventArgs e)
   {
              ribbonContextualTabGroup1.Visible = true;
   }
   private void richTextBox1_Leave(object sender, EventArgs e)
   {
               ribbonContextualTabGroup1.Visible = false;
   }
   ```

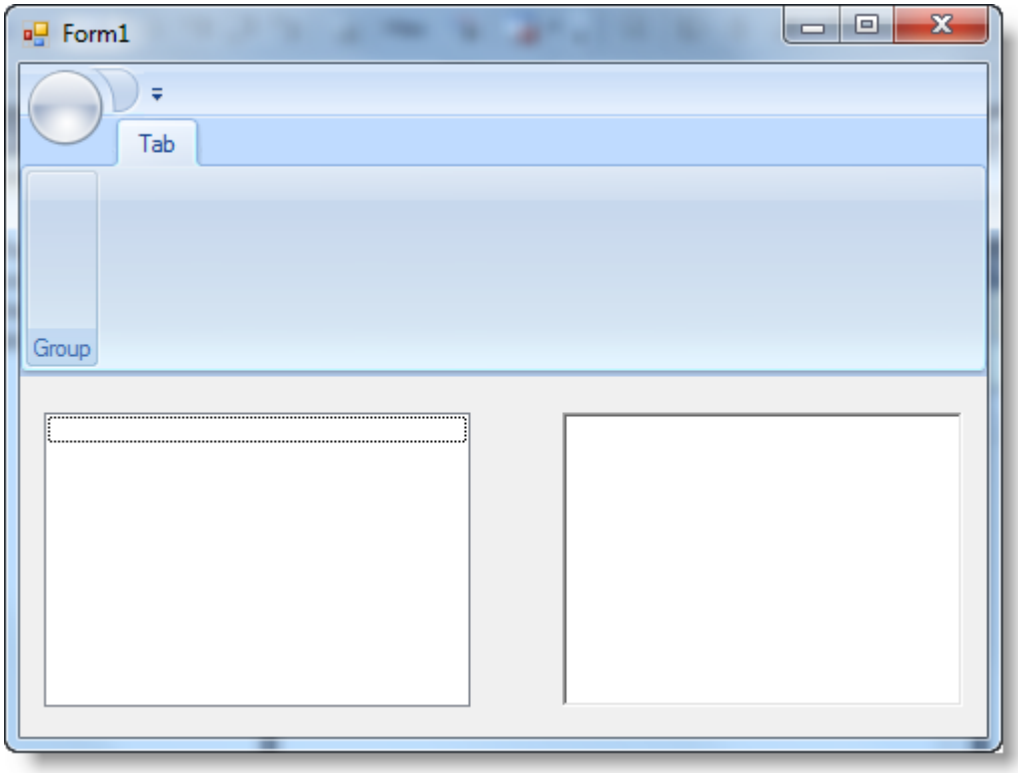
5. Select **View | Designer** to return to Design view.
6. Complete the following actions in the **Properties** window:
   - From the drop-down list, select **c1Ribbon1** and then set the Selectable property to **False**.
   - From the drop-down list, select **ribbonContextualTabGroup1** and then set the **Visible** property to **False**.
   - From the drop-down list, select **richTextBox1**, click the **Events** button  , and then set the following event handlers so that the **RichTextBox** control can handle the code you added to the project:
     - Set **Enter** to **richTextBox1_Enter**.
     - Set **Leave** to **richTextBox1_Leave**.
7. Press **F5** to run the project. Observe that the contextual tab that you added to the project is hidden.



8. Click inside the **RichTextBox** and observe that the contextual tab appears in orange.



9. Click inside the **ListBox** and observe that the contextual tab disappears.



And that's it! **Ribbon for WinForms** makes it that simple to add a contextual tab to the Ribbon interface.

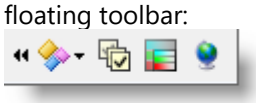
Adding a Tab to the Ribbon

To add a tab to the Ribbon, you can use the smart designer, collection editor, or add code. Each option is described below. To learn how to add a contextual tab, see Adding a Contextual Tab to the Ribbon.

Add a Ribbon Tab Using the Smart Designer

Complete the following steps:

1. Using your mouse pointer, hover over the Ribbon and click the smart designer tag. This enables Ribbon floating toolbar:

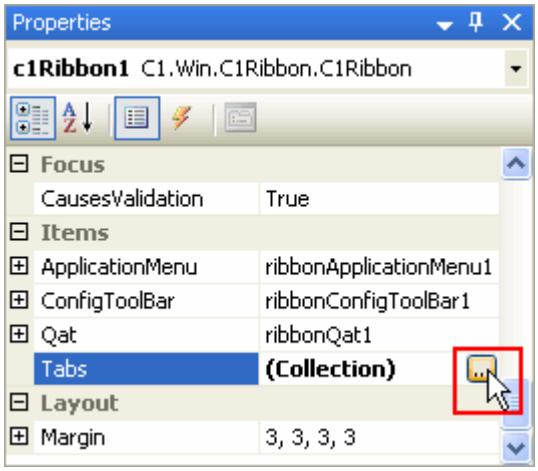


2. From the Ribbon floating toolbar, click the **Actions** drop-down button.
3. Select **Add Tab**. This adds a new tab to the Ribbon.
4. To edit the label, click the Tab text so that it is highlighted. Enter a new Tab name, for example, **Write**.
5. Press ENTER or click outside the editing box to accept the change.
6. Next, build on the tab by adding a group to the Ribbon tab.

Add a Ribbon Tab Using the RibbonTab Collection Editor

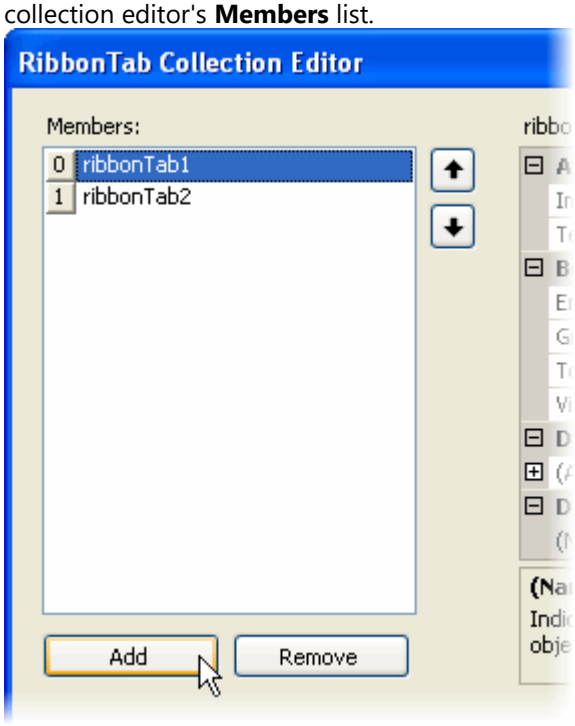
Complete the following steps:

1. Select the Ribbon to activate it.
2. From the Ribbon's Properties window, select the **Tabs** property and click on the **ellipsis** button next to the **(Collection)**.



The **RibbonTab Collection Editor** appears.

3. From the collection editor, click the **Add** button. A new tab is added to the Ribbon; the new tab is listed in the collection editor's **Members** list.



With the new RibbonTab highlighted, you can edit the properties through the Properties list.

4. Locate the **Text** property from the Properties list, and change the text to **Write**.
5. Click **OK** to close the collection editor.

Add a Ribbon Tab Programmatically

To add a Ribbon tab, add the following code to your project:

To write code in Visual Basic

Visual Basic

```
' Add a tab to the Ribbon
Dim RibbonTab2 As RibbonTab = New RibbonTab()
' Label the tab
RibbonTab2.Text = "Write"
C1Ribbon1.Tabs.Add(RibbonTab2)
```

To write code in C#

C#

```
// Add a tab to the Ribbon
RibbonTab RibbonTab2 = new RibbonTab();
// Label the tab
RibbonTab2.Text = "Write";
C1Ribbon1.Tabs.Add(RibbonTab2);
```

Adding a Group to the Ribbon Tab

To add a group to the Ribbon tab, you can use the smart designer, collection editor, or add code. Each option is described below.

To Add a Ribbon Group Using the Smart Designer

Complete the following steps:

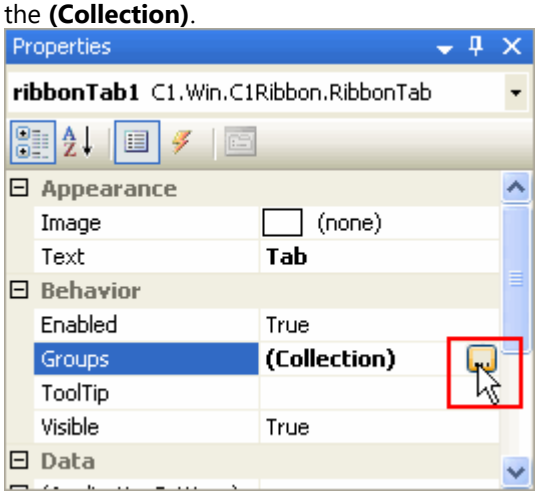
1. Select the tab that you want to add a new group to. For steps on creating a new tab, see the Adding a Tab to the Ribbon topic.
2. Click the tab to activate it and enable the tab's floating toolbar:

3. From the tab's floating toolbar, click the **Actions** drop-down button.
4. Select **Add Group**. This adds a new group to the tab.
5. To edit the label, click the group text so that it is highlighted. Enter a new group name, for example, **Font**.
6. Press ENTER or click outside the editing box to accept the change.
7. Build on the group by adding items to the Ribbon group.

To Add a Ribbon Group Using the RibbonGroup Collection Editor

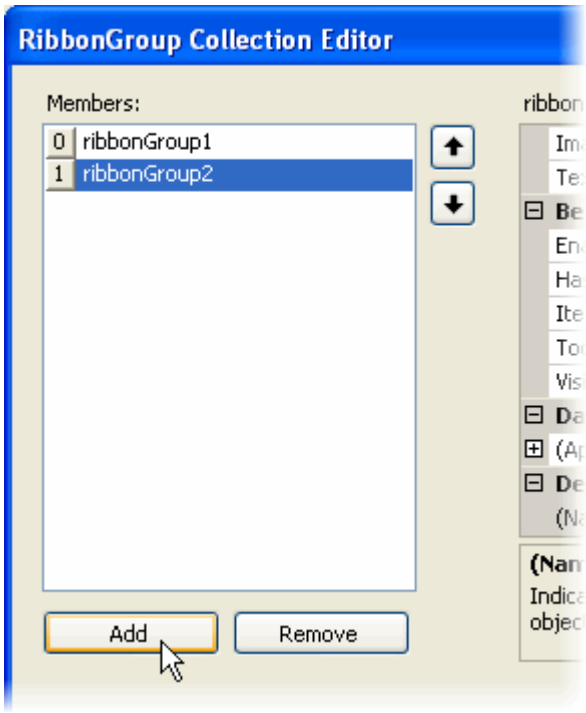
Complete the following steps:

1. Select the tab that you want to add a new group to. This activates the tab.
2. From the RibbonTab's Properties window, select the **Groups** property and click on the **ellipsis** button next to the **(Collection)**.



The **RibbonGroup Collection Editor** appears.

3. From the collection editor, click the **Add** button. A new group is added to the tab; the new group is listed in the **Members** list.



With the new group highlighted, you can edit the properties through the Properties list.

4. Locate the RibbonGroup.Text property from the Properties list, and change the text to **Font**.
5. Click **OK** to close the collection editor.

To Add a Ribbon Group Programmatically

To add a RibbonGroup to the RibbonTab, add the following code to your project:

To write code in Visual Basic

Visual Basic

```
Imports C1.Win.C1Ribbon

Private Sub Form1_Load(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles MyBase.Load
    ' Add a tab to the Ribbon
    Dim RibbonTab2 As RibbonTab = New RibbonTab()
    ' Label the tab
    RibbonTab2.Text = "Write"
    C1Ribbon1.Tabs.Add(RibbonTab2)

    ' Add a group to the Write tab
    Dim RibbonGroup2 As RibbonGroup = New RibbonGroup()
    ' Label the group
    RibbonGroup2.Text = "Font"
    RibbonTab2.Groups.Add(RibbonGroup2)
End Sub
```

To write code in C#

C#

```
// type the using directive for the namespace
using C1.Win.C1Ribbon;

private void Form1_Load(object sender, System.EventArgs e)
{
    // Add a tab to the Ribbon
    RibbonTab RibbonTab2 = new RibbonTab();
    // Label the tab
    RibbonTab2.Text = "Write";
    c1Ribbon1.Tabs.Add(RibbonTab2);

    // Add a group to the Write tab
    RibbonGroup RibbonGroup2 = new RibbonGroup();
    // Label the group
    RibbonGroup2.Text = "Font";
    RibbonTab2.Groups.Add(RibbonGroup2);
}
```

Note: If your RibbonGroup.Text property is set to a value that is longer than the contents of the RibbonGroup, you can set the TrimLongCaption property to **True** to trim the caption to the same length as the RibbonGroup.

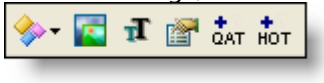
Adding Items to the Ribbon Group

To add items to the Ribbon group, you can use the smart designer, collection editor, or add code. Each option is described below.

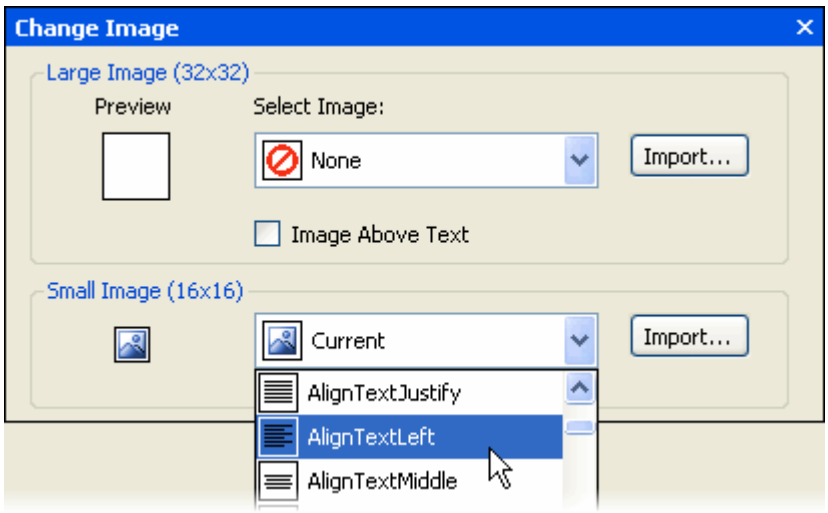
To Add Ribbon Items Using the Smart Designer

Complete the following steps:

1. Select the Ribbon group to enable the group's floating toolbar:

2. Click the **Actions** drop-down button. This reveals a list of Ribbon items to add to the group.
3. Select an item to add, **Add ToggleButton**, for example.
4. To delete the toggle button's label, click the text so that it is highlighted and press DELETE.
5. Press ENTER or click outside the editing box to accept the change.
6. To add an image, click the toggle button to activate it and enable the toggle button's floating toolbar:


7. Click the **Change Image** button  . The **Change Image** dialog box appears.
8. Click the **Small Image (16x16)** drop-down arrow and select the **AlignTextLeft** image.

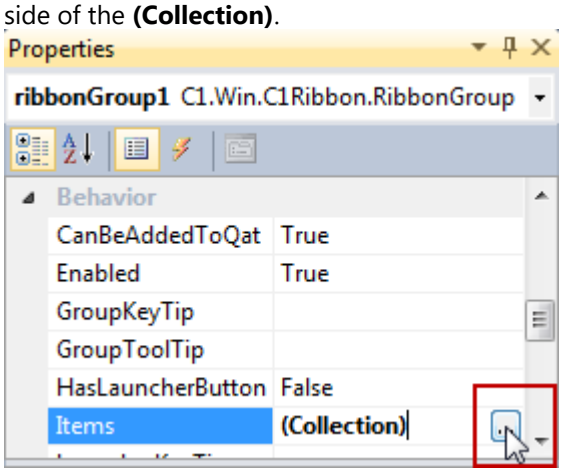


9. Continue to build on the group by adding Ribbon elements, such as buttons, check boxes, combo boxes, toolbars, and so on to fit your needs.

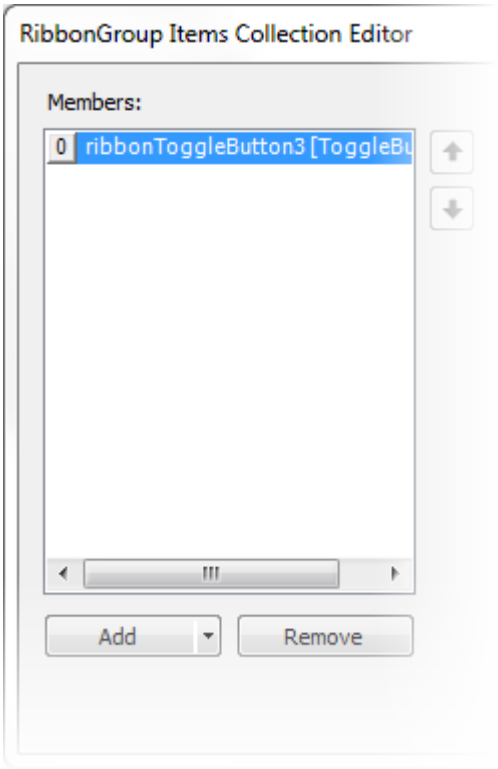
To Add Ribbon Items Using the Collection Editor

Complete the following steps:

1. Select the group that you want to add Ribbon items to. This activates the group.
2. From the Ribbon group's Properties window, select the **Items** property and click the **ellipsis** button at the right side of the **(Collection)**.



   The **RibbonGroup Items Collection Editor** appears.
3. From the collection editor, click the **Add** drop-down button.
4. Select **RibbonToggleButton** from the list of available items. The new item is listed in the **Members** list.



With the new item is highlighted, you can edit the properties through the Properties list.

5. Click **OK** to close the collection editor.

To Add Ribbon Items Programmatically

Note: In the following example embedded resources containing the following (16x16) images are used: *AlignLeft.png*, *AlignCenter.png*, and *AlignRight.png*. To embed a resource, from the **Project** menu, choose ***YourProjectName* Properties**. From the **Resources** tab, select **Add Resource** and choose to add use an existing file or add a new one.

To add Ribbon items to the group, for example, a toggle buttons, add the following code to your project:

To write code in Visual Basic

Visual Basic

```
' type the Imports directive for the namespace
Imports C1.Win.C1Ribbon
Private Sub Form1_Load(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles MyBase.Load
        ' Add a tab
        Dim RibbonTab2 As New RibbonTab()
        ' Label the tab
        RibbonTab2.Text = "Write"
        c1Ribbon1.Tabs.Add(RibbonTab2)
        ' Add a group to the Write tab
        Dim RibbonGroup2 As New RibbonGroup()
        ' Label the group
        RibbonGroup2.Text = "Font"
        RibbonTab2.Groups.Add(RibbonGroup2)
        Dim ToggleBtn1 As New RibbonToggleButton()
```

```
        ' Add RibbonToggleButton to the Ribbon Group
        RibbonGroup2.Items.Add(ToggleBtn1)
        ' Edit the toggle button properties
        ToggleBtn1.SmallImage = My.Resources.Resources.AlignLeft
        ToggleBtn1.Text = ""
        ToggleBtn1.ToolTip = "Align Left"
End Sub
```

To write code in C#

C#

```
// type the using directive for the namespace
using C1.Win.C1Ribbon;
private void Form1_Load(object sender, System.EventArgs e)
{
      // Add a tab
      RibbonTab RibbonTab2 = new RibbonTab();
      // Label the tab
      RibbonTab2.Text = "Write";
      C1Ribbon1.Tabs.Add(RibbonTab2);
      // Add a group to the Write tab
      RibbonGroup RibbonGroup2 = new RibbonGroup();
      // Label the group
      RibbonGroup2.Text = "Font";
      RibbonTab2.Groups.Add(RibbonGroup2);
      RibbonToggleButton ToggleBtn1 = new RibbonToggleButton();
      // Add RibbonToggleButton to the Ribbon Group
      RibbonGroup2.Items.Add(ToggleBtn1);
      // Edit the toggle button properties
      ToggleBtn1.SmallImage = Properties.Resources.AlignLeft;
      ToggleBtn1.Text = "";
      ToggleBtn1.ToolTip = "Align Left";
}
```

Displaying Images on RibbonTab

To display images of different sizes on RibbonTab, you can now set the images from size 16x16 to large image of size 32x32. Small images (16x16) can be easily set on RibbonTab using Image property. When an image of size other than 16X16 is applied to **RibbonTab**, size of the **RibbonTab** increases to accommodate the image. Large image (32x32) can now be set on **RibbonTab** by using DisableGlassEffects() method in Application.Designer.vb file of Visual Basic application and program.cs file in case of CSharp Application as shown in given code below:

To write code in Visual Basic

Visual Basic

```
C1RibbonForm.DisableGlassEffects()
```

To add code in C#

C#

```
C1RibbonForm.DisableGlassEffects();
```

The **AllowImageScaling** property allows you to prevent image scaling issues in high-resolution applications.

> **Note:** Large images (32x32) cannot be displayed as it is in Custom **Visual Style**, however, they are scaled to a smaller size to adjust in the Tab.

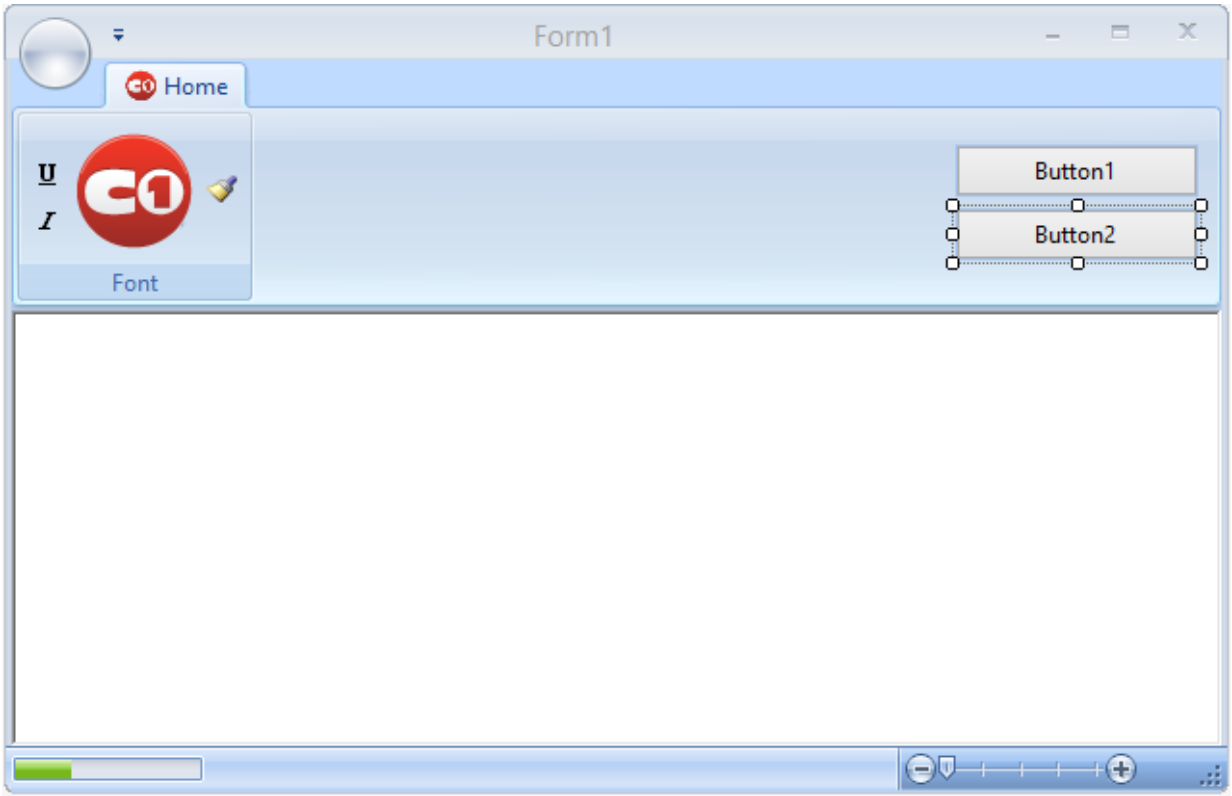
Changing the Orientation of Ribbon Items

Ribbon for WinForms allows users to change the orientation of the items added to the C1Ribbon control. You can change the default orientation (left-to-right) of the ribbon items such as images to right-to-left by using the FlipImageRtl property.

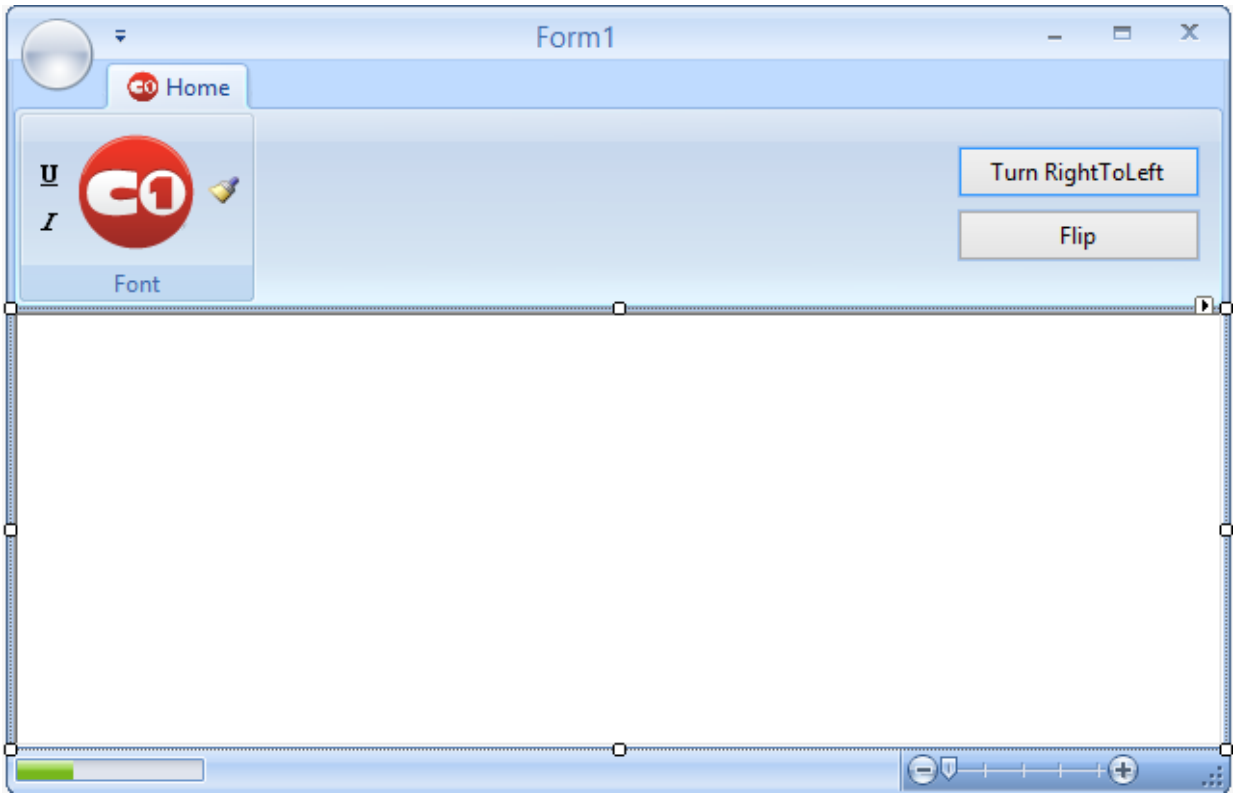
The following code example shows how you can change the orientation of the added ribbon items from left-to-right to right-to-left. The example given below is created using the Quick Start sample.

In Code

1. Add a toggle button in the ribbon toolbar and modify its properties through the Properties window as follows:
   - Name - ribbonToggleButton1
   - Set the **Image** property to a 16x16 **Brush** preset image from the floating toolbar
2. Add another toggle button in the ribbon toolbar and modify its properties through the Properties window as follows:
   - Name - ribbonToggleButton4
   - Set the **Image** property to a 32x32 image from the floating toolbar. In this example, the C1Logo image is used.
3. Drag-and-drop two general button controls onto the Form and place them to the right-most corner of the ribbon control such that the **Design** view appears similar to the following image.



4. Set the **Text** property for the first button to **Turn RightToLeft**, and the Text property of the second button to **Flip**. The **Design** view appears similar to the image below.



5. Subscribe the **Click** events on the button controls added in Step 4 through the **Properties** window.
6. Switch to the code view and add the following code in the event handlers created for the button click events subscribed in Step 5.
    - **Visual Basic**

```
Private Sub button1_Click(sender As Object, e As EventArgs) Handles button1.Click
    If Not RightToLeftLayout Then
        RightToLeftLayout = True
        RightToLeft = RightToLeft.Yes
    Else
        RightToLeftLayout = False
        RightToLeft = RightToLeft.No
    End If
End Sub

Private Sub FlipImage_Click(sender As Object, e As EventArgs) Handles FlipImage.Click
    ribbonToggleButton1.FlipImageRtl = Not ribbonToggleButton1.FlipImageRtl
    ribbonToggleButton2.FlipImageRtl = Not ribbonToggleButton2.FlipImageRtl
    ribbonToggleButton4.FlipImageRtl = Not ribbonToggleButton4.FlipImageRtl
End Sub
```

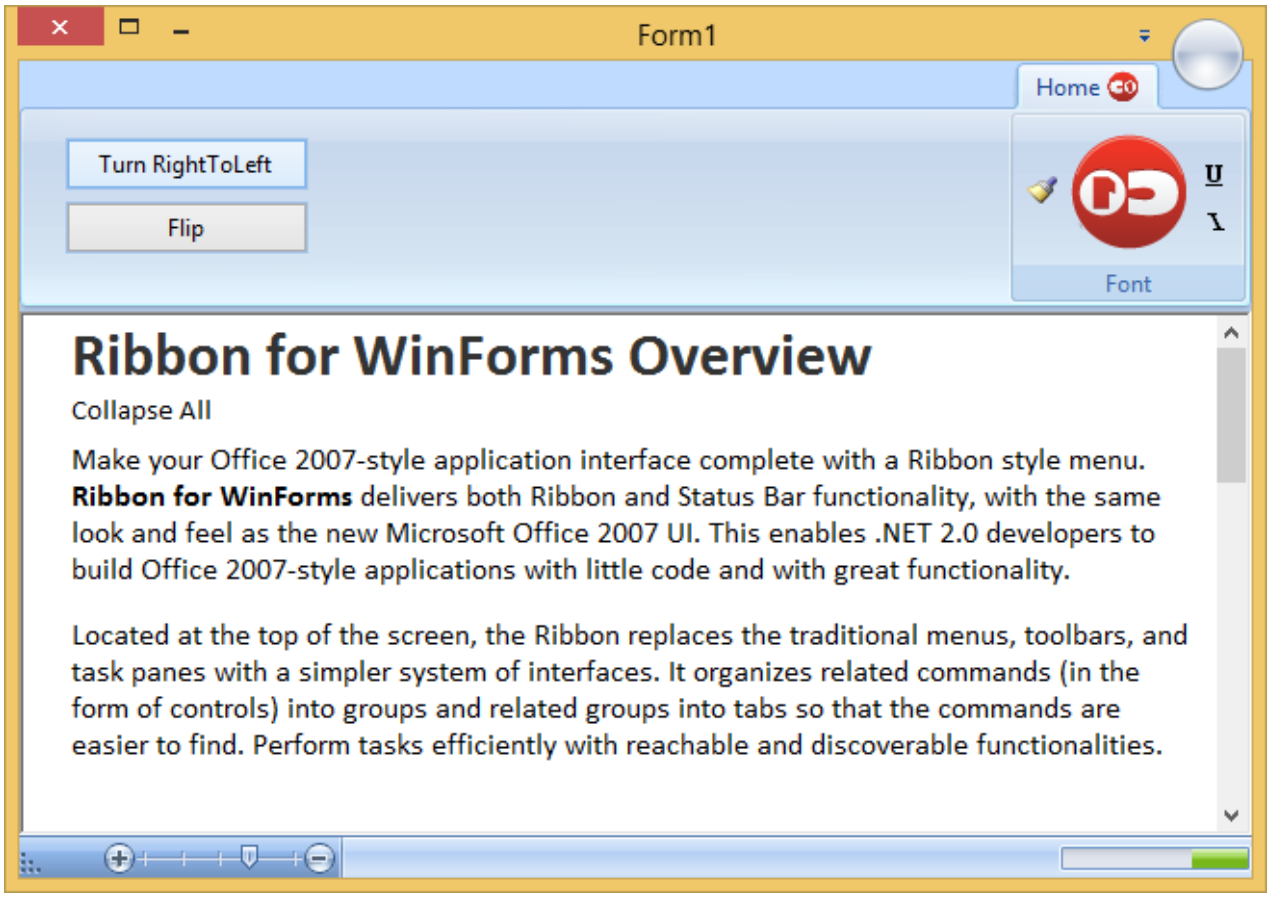
    - **C#**

```
private void FlipImage_Click(object sender, EventArgs e)
{
    ribbonToggleButton1.FlipImageRtl = !ribbonToggleButton1.FlipImageRtl;
    ribbonToggleButton2.FlipImageRtl = !ribbonToggleButton2.FlipImageRtl;
    ribbonToggleButton4.FlipImageRtl = !ribbonToggleButton4.FlipImageRtl;
}

private void button1_Click(object sender, EventArgs e)
```

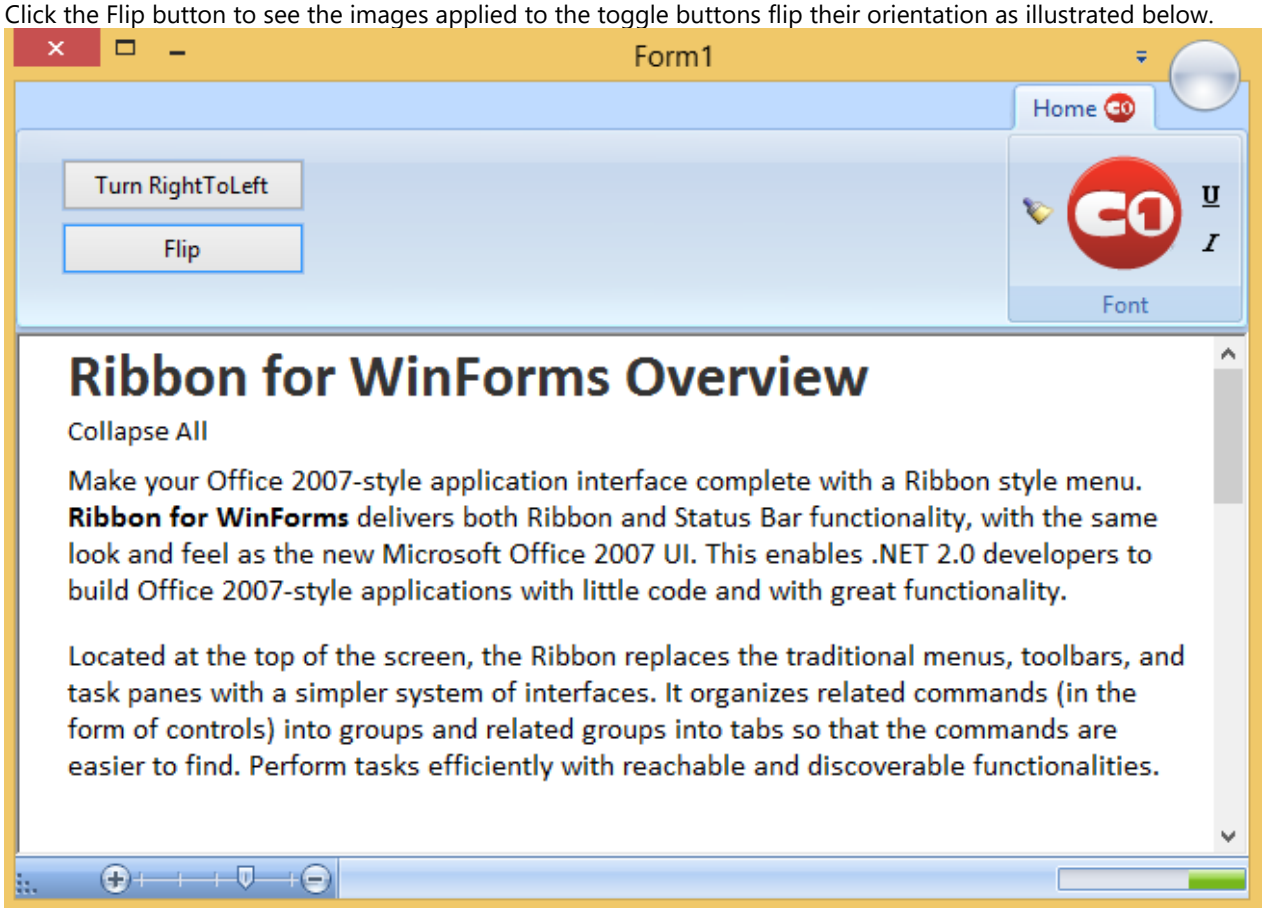
```
{
    if (!RightToLeftLayout)
    {
        RightToLeftLayout = true;
        RightToLeft = RightToLeft.Yes;
    }
    else
    {
        RightToLeftLayout = false;
        RightToLeft = RightToLeft.No;
    }
}
```

7. Run the application and observe that the form appears similar to the image below.

8. Click the **Turn RightToLeft** button and notice that the form's default orientation changes as illustrated in the image below.



9. Click the Flip button to see the images applied to the toggle buttons flip their orientation as illustrated below.

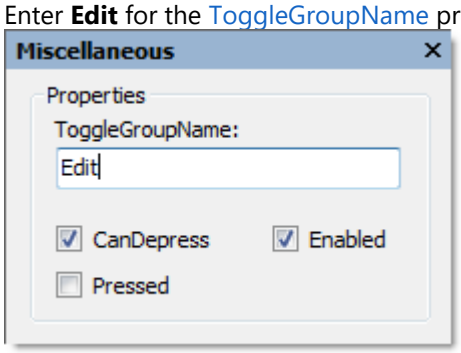


Creating a Toggle Button Group

You can create a group of mutually exclusive Toggle Buttons with the ToggleGroupName property.

Complete the following steps:

1. Select the Ribbon Group to which you would like to add a group of toggle buttons. This will enable the group's floating toolbar.
2. Click the **Actions** drop-down button. This reveales the Ribbon items available to add to the group.
3. Select **Add Toggle Button** from the list. A labeled toggle button will appear on your Ribbon control.
4. Select the toggle button to activate the button's floating toolbar.
5. Click the **Miscellaneous Settings** button  to open the **Miscellaneous** dialog box.
6. Enter **Edit** for the ToggleGroupName property as in the following image:



7. Continue to build the group by adding more toggle buttons and setting their ToggleGroupName property to **Edit**.

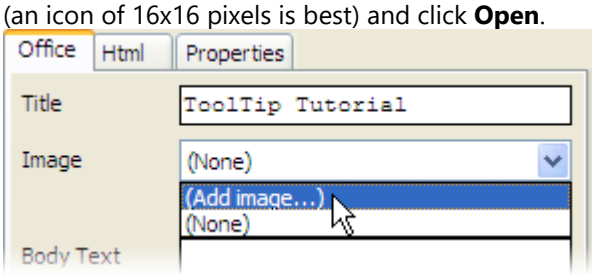
Creating A Rich ToolTip

You can add rich ToolTips to RibbonItems, RibbonGroups, and RibbonTabs using the ToolTip Editor. In this topic, we will add a rich ToolTip to a RibbonTab.

Note: To learn how to create a regular ToolTip for a ribbon item, see Displaying ToolTips for the Ribbon Items.

Complete the following steps:

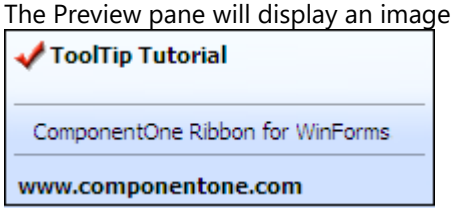
1. Select **C1Ribbon**'s default tab to activate it. A list of the tab's properties will be revealed in the Properties window.
2. In the Properties window, locate the tab's ToolTip property and click the ellispis (...) button.
   The **ToolTip** editor opens with the **Office** tab page in focus.
3. Complete the following tasks in the **Office** tab page:
   1. Enter "ToolTip Tutorial" into the **Title** text box.
   2. Click the **Image** drop-down and select **Add Image** to access the **Open** dialog box. Select a small image (an icon of 16x16 pixels is best) and click **Open**.



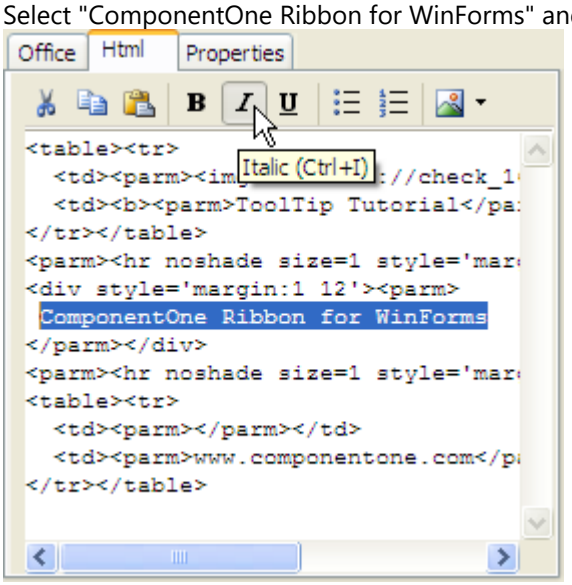
   3. Check the **Top Separator** checkbox.
   4. Type the following text into the **Body Text** text box: "ComponentOne Ribbon for WinForms."

5. Check the **Bottom Separator** checkbox.
6. Type "www.componentone.com" into the **Subtitle** text box.

The Preview pane will display an image similar to the following:

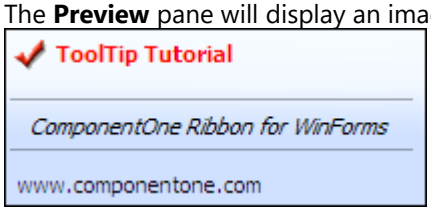


4. Click the Html tab to bring its tab page into focus and then complete the following steps:
   1. In the editor, delete the and tags that enclose "www.componentone.com".
   2. Select "ComponentOne Ribbon for WinForms" and press the **Italic** button.



   3. Place your cursor in front of the title ("ToolTip Tutorial") and type . Move your cursor to the end of the title and type .

The **Preview** pane will display an image similar to the following:

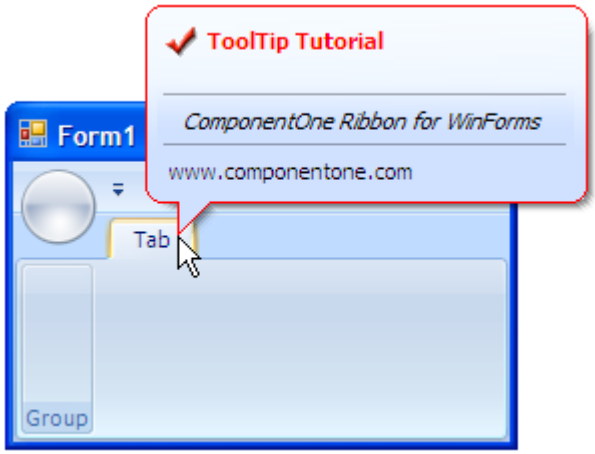


5. Click the Properties tab to bring its tab page into focus and then set the following properties:
   - Set the Border property to **True**
   - Set the BorderColor property to **Red**.
   - Set the IsBalloon property to **True**.
   - Set the IntialDelay property to "50". This means that the ToolTip will appear 50 milliseconds (.05 seconds) after a user hovers over the tab with the cursor.
6. Press **OK** to close the **ToolTip Editor**, then press **OK** to close the **C1InputPanel Collection Editor**.

> You can also use html table header cells (<th> tags) in the tooltips.

This topic illustrates the following:

Once the project is built, hover over the RibbonTab with your cursor to make the ToolTip appear. The resulting ToolTip will resemble the image below:



Embedding Controls in a Ribbon

Beginning with the 2008 v2 release, you can add arbitrary controls to **Ribbon for WinForms**. You can easily add a control using the RibbonControlHost element by defining a new class inheriting **C1.Win.C1Ribbon.RibbonControlHost**.

Embedding a TextBox in a Ribbon Group

Complete the following steps to add a standard TextBox control to a Ribbon group:

1. Create a new Ribbon application. For more information see Creating a Ribbon Application Project.
2. Open the MainRibbonForm to view the Ribbon form, and select **View | Code** to open Code view.
3. Add the following code to your project to create a new TextBoxHost class that inherits the RibbonControlHost element:

To write code in Visual Basic

Visual Basic

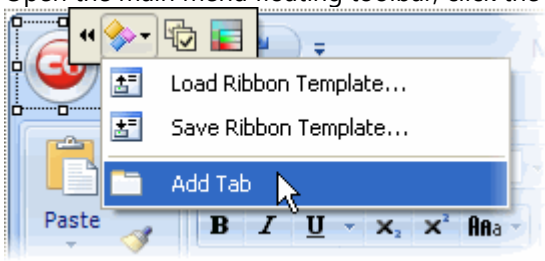
```
Public Class TextBoxHost
    Inherits C1.Win.C1Ribbon.RibbonControlHost
    Public Sub New()
        MyBase.New(New System.Windows.Forms.TextBox())
    End Sub
End Class
```

To write code in C#

C#

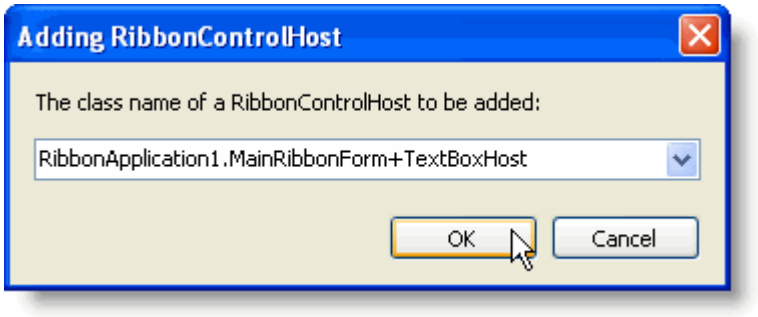
```
public class TextBoxHost : C1.Win.C1Ribbon.RibbonControlHost
{
    public TextBoxHost()
        : base(new System.Windows.Forms.TextBox())
    {
    }
}
```

4. Build and close your project, and then return to Design view.
5. Open the main menu floating toolbar, click the **Actions** button, and select **Add Tab**.

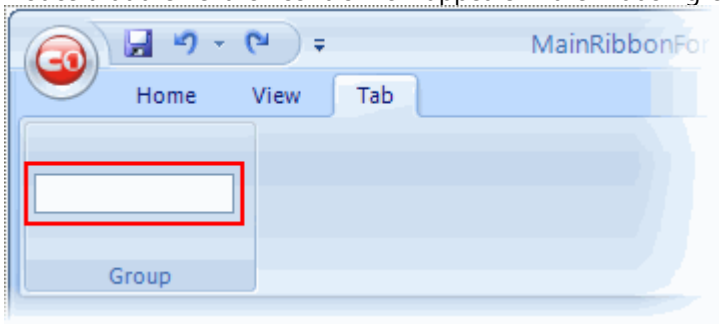


A new Ribbon tab with an empty Ribbon group will be added to the Ribbon.

6. Select the new Ribbon group, and from the floating toolbar click the **Action** button and select **Add ControlHost**.
The **Adding RibbonControlHost** dialog box will open.
7. Type the name of the control host in the **Adding RibbonControlHost** dialog box, for example "*ProjectName.FormName*+TextBoxHost" replacing *ProjectName* and *FormName* with the names of your project and form.



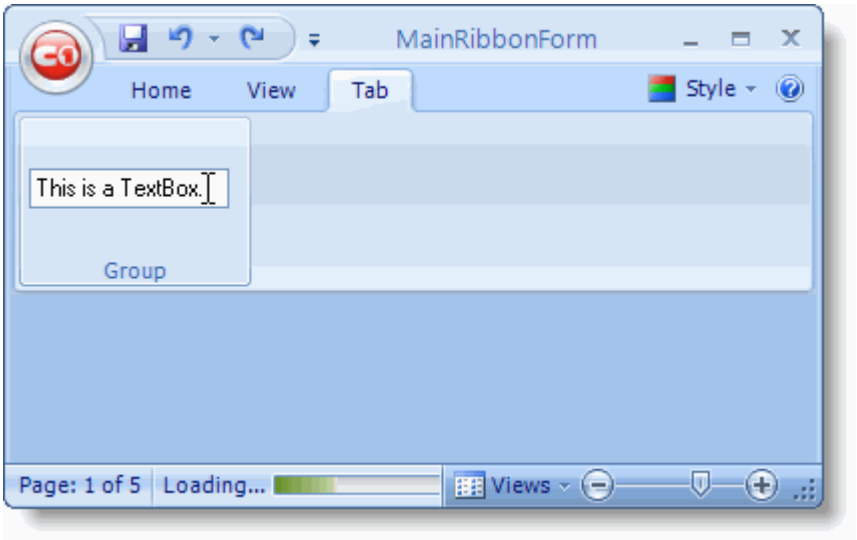
8. Click **OK** to close the **Adding RibbonControlHost** dialog box.
Notice that the TextBox control now appears in the Ribbon group:



Tip: When you next add the TextBoxHost element you won't have to type its name. The name of this class will be available in the drop-down list in the **Adding RibbonControlHost** dialog box.

This topic illustrates the following:

At run time, you will be able to interact with the TextBoxHost element as you would a standard TextBox.



The RibbonControlHost class publishes the standard properties and events of the hosted control. To access other properties, methods, and events of the hosted control use the RibbonControlHost.Control property.

It is also possible to override methods with events, such as OnSubscribeControlEvents, to handle events raised by the hosted control. You can add custom functionality into properties to enhance the hosted control.

Embedding a Gauge in a Ribbon Group

You can host a **C1Gauge** control in a C1RibbonGroup or in a C1StatusBar control through using the RibbonControlHost element. Complete the following steps to embed a **C1Gauge** in a C1RibbonGroup:

> **Note**: This help uses a template file to create the ruler view for the C1Gauge control. To create this file, add a **C1Gauge** control to a Windows Form and choose the Linear Gauge Simple Ruler template from the **New Gauge Gallery** dialog box. Open the **C1Gauge** smart tag and click **Save to XML File**. Enter the file name **Ruler.xml** and click **Save**.

1. Create a new Ribbon application. (For more information see Creating a Ribbon Application Project)
2. Add a C1Ribbon control and a C1StatusBar control to the form. Change the Windows Form to a Ribbon Form by modifying the code that declares your form.
3. Right-click the Windows Form and select **View Code**.
4. Replace the class declaration with the following code to change the Windows Form to aRibbon Form:

   To write code in Visual Basic

   Visual Basic

   ```
   Partial Class Form1
   Inherits C1.Win.C1Ribbon.C1RibbonForm
   End Class
   ```

   To write code in C#

   C#

   ```
   public partial class Form1 : C1RibbonForm
   ```

5. While still in code view, we will add a **C1Gauge** control by defining a new class using the RibbonControlHost. This code will also handle the **PointerDragMove** event to handle the pointer dragging in the control. Add the following code to the top of the page:

To write code in Visual Basic

Visual Basic

```
Imports C1.Win.C1Gauge
Imports System.Xml
```

To write code in C#

C#

```
using C1.Win.C1Gauge;
using System.Xml;
```

6. Add the following code after the **InitializeComponent()** method:

To write code in Visual Basic

Visual Basic

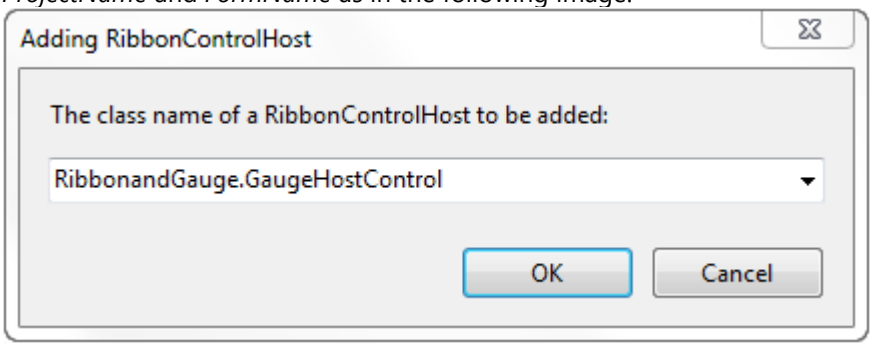
```
Public Class GaugeHostControl
     Inherits C1.Win.C1Ribbon.RibbonControlHost
           Private linearGauge As C1LinearGauge
          Public Sub New()
         MyBase.New(New C1.Win.C1Gauge.C1Gauge)
         linearGauge = New C1LinearGauge
         Dim doc As XmlDocument = New XmlDocument
         doc.LoadXml(Properties.Resources.Ruler)
         linearGauge.Load(doc)
         C1Gauge.Gauges.Add(linearGauge)
         C1Gauge.BackColor = System.Drawing.Color.Azure
         linearGauge.PointerDragMove = (linearGauge.PointerDragMove +
LinearGauge_PointerDragMove)
     End Sub
          <Browsable(false),  _

DesignerSerializationVisibility(DesignerSerializationVisibility.Hidden)> _
                        Public ReadOnly Property C1Gauge As
C1.Win.C1Gauge.C1Gauge
         Get
             Return CType(Control,C1.Win.C1Gauge.C1Gauge)
         End Get
     End Property

     Private Sub LinearGauge_PointerDragMove(ByVal sender As Object, ByVal e As
PointerDragEventArgs)
         e.Pointer.UpdateValue(e.NewValue, 0.5)
     End Sub
 End Class
```

To write code in C#

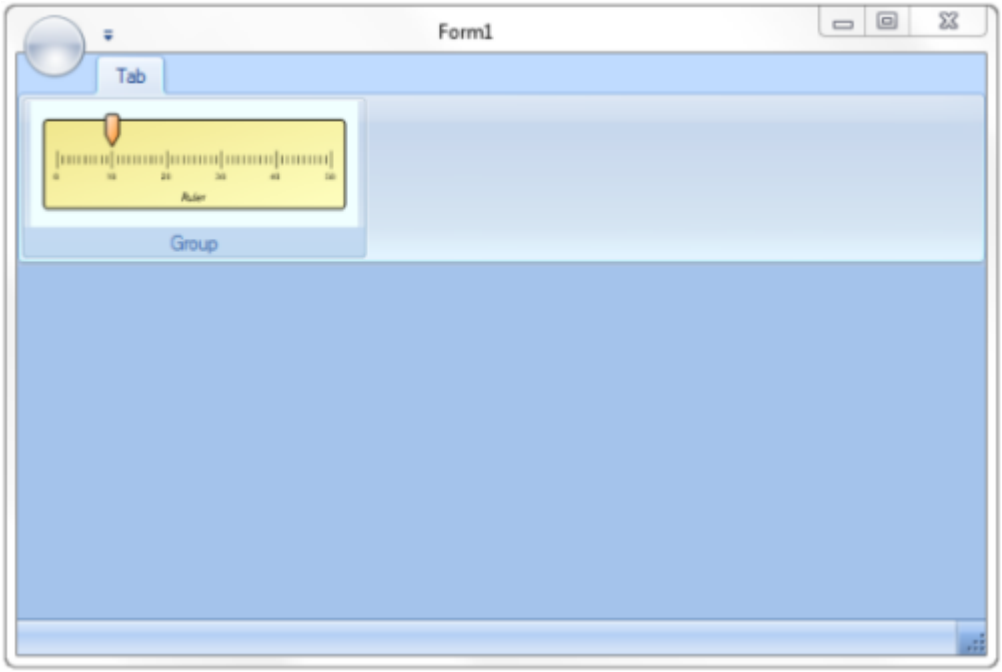
C#

```
public class GaugeHostControl : C1.Win.C1Ribbon.RibbonControlHost
{
    private C1LinearGauge linearGauge;
    public GaugeHostControl() : base(new C1.Win.C1Gauge.C1Gauge())
    {
        linearGauge = new C1LinearGauge();
        XmlDocument doc = new XmlDocument();
        doc.LoadXml(Properties.Resources.Ruler);
        linearGauge.Load(doc);
        C1Gauge.Gauges.Add(linearGauge);
        C1Gauge.BackColor = System.Drawing.Color.Azure;
        linearGauge.PointerDragMove += LinearGauge_PointerDragMove;
    }
    [Browsable(false)]
[DesignerSerializationVisibility(DesignerSerializationVisibility.Hidden)]
    public C1.Win.C1Gauge.C1Gauge C1Gauge
    {
        get
        {
            return (C1.Win.C1Gauge.C1Gauge)Control;
        }
    }
    private void LinearGauge_PointerDragMove(System.Object sender,
PointerDragEventArgs e)
    {
        e.Pointer.UpdateValue(e.NewValue, 0.5);
    }
}
```

7. From the Solution Explorer, open the **Resources.resx** file. Click **Add Resource | Add Existing File** and choose the **Ruler.xml** file from the folder in which it was saved.
8. Build your project before switching back to Design View. Open theRibbon Group Floating Toolbar by selecting the empty Ribbon Group.
9. Click the **Actions** button  and select **Add Control Host** from the list. The **Adding RibbonControlHost** dialog window will appear.
10. Enter a class name for the **RibbonControlHost** to be added. The class name should follow the *ProjectName.FormName+GaugeHostControl* format, substituting the names of your project and form for *ProjectName* and *FormName* as in the following image:



11. Run your application. The C1Ribbon application should appear as follows:



Handling Ribbon Events

While the majority of the Ribbon can be created using the visual designers, to make the items on the Ribbon perform specific functions requires coding. The following topics show how to add **Click** event handlers for various items on the Ribbon.

Handling the RibbonButton.Click Event

Note: This topic assumes that you have added three Ribbon buttons to the Ribbon and a **RichTextBox** control to the Ribbon Form. For steps on how to add a Ribbon control to the Ribbon, see Adding Items to the Ribbon Group.

To copy, cut, or paste text in the rich text box, create RibbonButton.Click event handlers for the **Copy**, **Cut**, and **Paste** buttons. Add the following code to your project:

To write code in Visual Basic

Visual Basic

```
' type the Imports directive for the namespace
Imports C1.Win.C1Ribbon

' handles the Click event for the Copy button
Private Sub CopyBtn_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles CopyBtn.Click
    Me.RichTextBox1.Copy()
    Me.RichTextBox1.Focus()
End Sub

' handles the Click event for the Cut button
Private Sub CutBtn_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles Cutbtn.Click
```

```
    Me.RichTextBox1.Cut()
    Me.RichTextBox1.Focus()
End Sub

' handles the Click event for the Paste button
Private Sub PasteBtn_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles PasteBtn.Click
    Me.RichTextBox1.Paste()
    Me.RichTextBox1.Focus()
End Sub
```

To write code in C#

C#

```
// type the using directive for the namespace
using C1.Win.C1Ribbon;

// handles the Click event for the Copy button
private void CopyBtn_Click(object sender, EventArgs e)
{
    this.richTextBox1.Copy();
    this.richTextBox1.Focus();
}

// handles the Click event for the Cut button
private void CutBtn_Click(object sender, EventArgs e)
{
    this.richTextBox1.Cut();
    this.richTextBox1.Focus();
}

// handles the Click event for the Paste button
private void PasteBtn_Click(object sender, EventArgs e)
{
    this.richTextBox1.Paste();
    this.richTextBox1.Focus();
}
```

Note that the following properties have been set for the RibbonButton.**Name** properties:

- RibbonButton1.**Name** = **CopyBtn**
- RibbonButton2.**Name** = **CutBtn**
- RibbonButton3.**Name** = **PasteBtn**

Handling the RibbonToggleButton.Click Event

Note: This topic assumes that you have added a RibbonToggleButton to the Ribbon and a **RichTextBox** control to the Ribbon Form. For steps on how to add a Ribbon control to the Ribbon, see Adding Items to the Ribbon Group.

To make the selected text bold for a rich text box, create a **RibbonToggleButton.Click** event handler for the **Bold**

button. Add the following code to your project:

To write code in Visual Basic

Visual Basic

```
' type the Imports directive for the namespace
Imports C1.Win.C1Ribbon

' handles the Click event for the Bold button
Private Sub BoldBtn_Click(ByVal sender As Object, ByVal e As EventArgs) Handles
BoldBtn.Click
    ' assign style for Bold button
    ToggleSelectionFontStyle(FontStyle.Bold)
End Sub

' apply font style to the RichTextBox
Sub ToggleSelectionFontStyle(ByVal fontStyle As FontStyle)
 If Me.RichTextBox1.SelectionFont Is Nothing Then
   MessageBox.Show("Cannot change font style while selected text has more than one
font.")
 Else
   Me.RichTextBox1.SelectionFont = New Font(Me.RichTextBox1.SelectionFont,
Me.RichTextBox1.SelectionFont.Style Xor fontStyle)
 End If
 Me.RichTextBox1.Focus
End Sub
```

To write code in C#

C#

```
// type the using directive for the namespace
using C1.Win.C1Ribbon;

// handles the Click event for the Bold button
private void BoldBtn_Click(object sender, EventArgs e)
{
    // assign style for Bold button
    ToggleSelectionFontStyle(FontStyle.Bold);
}

// apply font style to the richTextBox
void ToggleSelectionFontStyle(FontStyle fontStyle)
{
    if (this.richTextBox1.SelectionFont == null)
    {
        MessageBox.Show("Cannot change font style while selected text has more than
one font.");
    }
    else
    {
        this.richTextBox1.SelectionFont = new Font(this.richTextBox1.SelectionFont,
```

```
            this.richTextBox1.SelectionFont.Style ^ fontStyle);
    }

    this.richTextBox1.Focus();
}
```

Note that the **RibbonToggleButton.Name** property has been set to **BoldBtn** for this example.

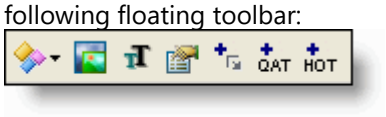
Adding a Launcher Button to the Ribbon Group

You can add a dialog box launcher button to the Ribbon group using the smart designer, Properties window, or Code Editor.

To Add a Launcher Button Using the Smart Designer

Complete the following steps:

1. Using your mouse pointer, click the Ribbon group and hover over the group's caption area to enable the following floating toolbar:



2. Click the **Add Launcher Button** button.

To Add a Launcher Button Using the Properties Window

Optionally, you can add a launcher button to the Ribbon group using the Properties window. Click the group on the Form to reveal the group properties in the Properties window. Locate the RibbonGroup.HasLauncherButton property's drop-down arrow and select **True** from the drop-down list.

To Add a Launcher Button Programmatically:

To add a launcher button to the group (RibbonGroup1), add the following code to your project:

To write code in Visual Basic

Visual Basic

```
' type the Imports directive for the namespace
Imports C1.Win.C1Ribbon

Private Sub Form1_Load(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles MyBase.Load
    ' add launcher button
    Me.RibbonGroup1.HasLauncherButton = True
End Sub
```

To write code in C#

C#

```
// type the using directive for the namespace
```

```
using C1.Win.C1Ribbon;

private void Form1_Load(object sender, System.EventArgs e)
{
    // add launcher button
    this.ribbonGroup1.HasLauncherButton = true;
}
```

Adding Status Bar Items

Before you begin this task, double-click the C1StatusBar icon in the Visual Studio Toolbox to add the C1StatusBar control to the Ribbon Form. The status bar docks at the bottom of the Ribbon Form. Note that you can add items to the status bar using the smart designer, Collection Editor, or Code Editor. This topic shows how to add status bar items using the Collection Editors.

Complete the following steps:

1. Click the C1StatusBar control to activate it.
2. In C1StatusBars Properties window, click on the **(Collection)** next to the LeftPaneItems property, then click the **ellipsis** button.
   The **C1StatusBar LeftPaneItems Collection Editor** appears.
3. Click the **Add** drop-down button and select **Ribbon ProgressBar** from the list.
4. With the progress bar selected in the **Members** list, set the RibbonProgressBar.Value property to **30**.
5. Click **OK** to close the collection editor.
6. In C1StatusBars Properties window, click on the **(Collection)** next to the RightPaneItems property, then click the **ellipsis** button.
   The **C1StatusBar RightPaneItems Collection Editor** appears.
7. Click the **Add** drop-down button and select RibbonButton and RibbonTrackBar to add the items to the Members list.
8. With the button selected in the **Members** list, set the following properties in the Properties window:
   - RibbonItem.SmallImage = **None**
   - RibbonButton.Text = **30%**
9. Click **OK** to close the collection editor.
10. In C1StatusBars Properties window, set the C1StatusBar.RightPaneWidth property to **150**.
11. In the Code Editor, add the following code to enable the items on the left and right panel:

    To write code in Visual Basic

    Visual Basic

    ```
    ' type the Imports directive for the namespace
    Imports C1.Win.C1Ribbon

    Private Sub Form1_Load(ByVal sender As System.Object, ByVal e As
    System.EventArgs) Handles MyBase.Load
        RibbonTrackBar1.SmallChange = 1
        RibbonTrackBar1.LargeChange = 5
        RibbonTrackBar1.Minimum = 0
        RibbonTrackBar1.Maximum = 100
        RibbonTrackBar1.Value = 30
        AddHandler RibbonTrackBar1.Scroll, AddressOf RibbonTrackBar1_Scroll
    End Sub
    ```

```
Sub RibbonTrackBar1_Scroll(ByVal sender As Object, ByVal e As EventArgs)
    Dim val As Integer = RibbonTrackBar1.Value
    RibbonProgressBar1.Value = val
    RibbonButton1.Text = val.ToString + "%"
End Sub
```

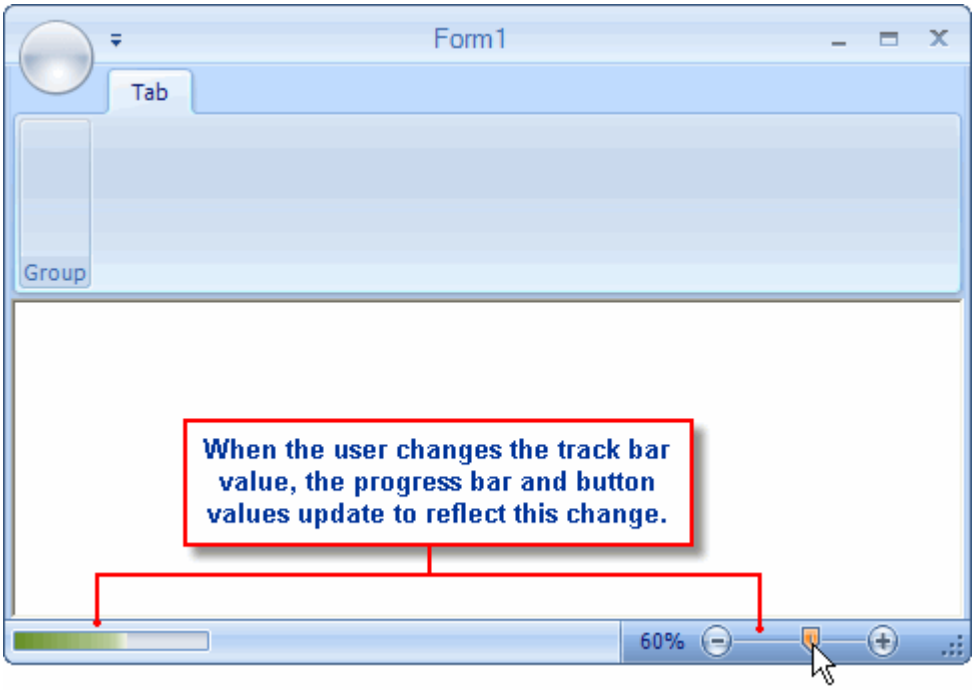
To write code in C#

C#

```
// type the using directive for the namespace
using C1.Win.C1Ribbon;

private void Form1_Load(object sender, EventArgs e)
{
    ribbonTrackBar1.SmallChange = 1;
    ribbonTrackBar1.LargeChange = 5;
    ribbonTrackBar1.Minimum = 0;
    ribbonTrackBar1.Maximum = 100;
    ribbonTrackBar1.Value = 30;
    ribbonTrackBar1.Scroll += new EventHandler(ribbonTrackBar1_Scroll);

}
void ribbonTrackBar1_Scroll(object sender, EventArgs e)
{
    int val = _trackbar.Value;
    ribbonProgressBar1.Value = val;
    ribbonButton1.Text = val.ToString() + "%";
}
```

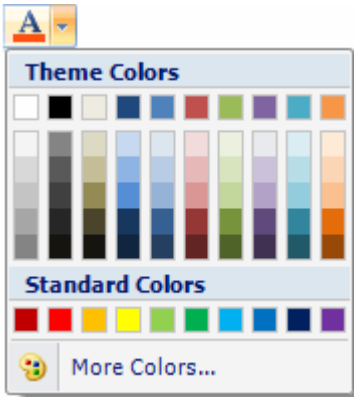
This topic illustrates the following:

Run the application, click and drag the track bar control. Notice that the progress bar and button control values change simultaneously:



Changing the Color Picker Theme Colors

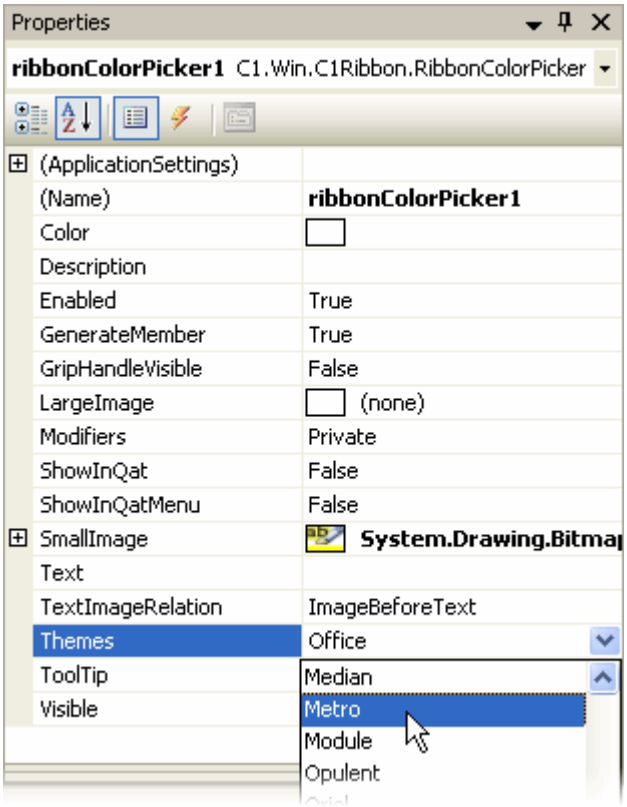
The RibbonColorPicker has two sections the theme colors and the standard colors:



You can change the theme colors by modifying the OfficeColorPalette property through the Properties window or programmatically.

To Change the Color Picker Theme Using the Properties Window

You can change the OfficeColorPalette property using the Properties window. Add a color picker to the Ribbon group, and then click the color picker item to reveal the RibbonColorPicker properties in the Properties window. Locate the OfficeColorPalette property's drop-down arrow and select the new color theme from the list, for example, **Metro**:



To Change the Color Picker Theme Programmatically

Note: In the following example an embedded resource containing the *FontColor.png* (16x16) image is used. To embed a resource, from the **Project** menu, choose ***YourProjectName* Properties**. From the **Resources** tab, select **Add Resource** and choose to an existing file or add a new one.

To change the color picker's theme to Office2007ColorThemes.Metro, for example, add the following code to your project:

To write code in Visual Basic

Visual Basic

```
' type the Imports directive for the namespace
Imports C1.Win.C1Ribbon

Private Sub Form1_Load(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles MyBase.Load

    Dim ColorPicker As RibbonColorPicker = New RibbonColorPicker()

    ' add a color picker to the group
    RibbonGroup1.Items.Add(ColorPicker)

    ' add the FontColor image to your Resources folder
```

```
    ' set the image for the color picker
    ColorPicker.SmallImage = Properties.Resources.FontColor

    ' set the theme color for the color picker
    ColorPicker.OfficeColorPalette = Office2007ColorThemes.Metro
End Sub
```

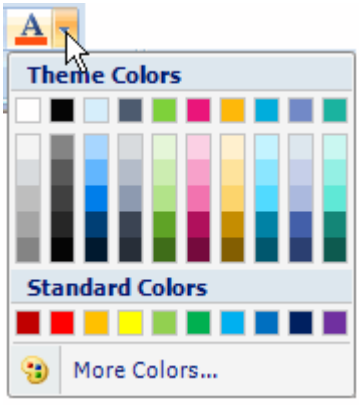
To write code in C#

C#

```
// type the using directive for the namespace
using C1.Win.C1Ribbon;

private void Form1_Load(object sender, System.EventArgs e)
{
    RibbonColorPicker ColorPicker = new RibbonColorPicker();

    // add a color picker to the group
    ribbonGroup1.Items.Add(ColorPicker);

    // add the FontColor image to your Resources folder
    // set the image for the color picker
    ColorPicker.SmallImage = Properties.Resources.FontColor;

    // set the theme color for the color picker
    ColorPicker.OfficeColorPalette = Office2007ColorThemes.Metro;
}
```

This topic illustrates the following:

Run the application, and click the color picker. The new color theme is visible under the **Theme Colors** section:



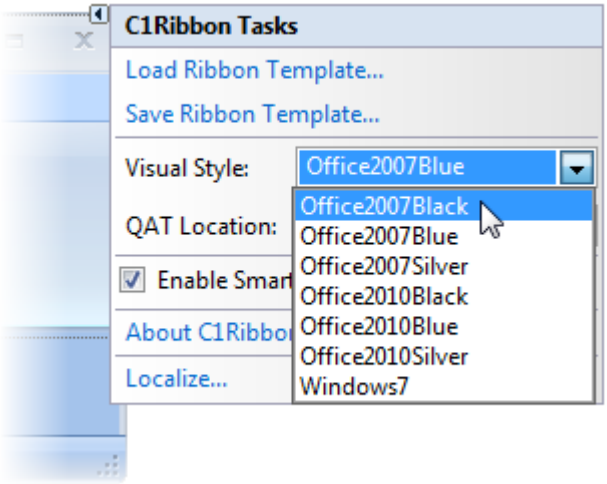
Changing the Visual Style

You can change the Ribbon's visual style, using the smart tag, smart designer, Properties window, or by adding code to set the C1Ribbon.VisualStyle property. Each option is described below.

To Change the Visual Style Using the Smart Tag

Complete the following steps:

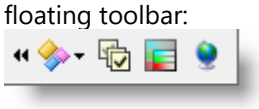
1. Select the Ribbon to activate the control.
2. Click the smart tag (▶) to enable the **C1Ribbon Tasks** menu.
3. Select the **Visual Style** drop-down arrow and choose **Office2007Black**.



To Change the Visual Style Using the Smart Designer

Complete the following steps:

1. Using your mouse pointer, hover over the Ribbon and click the smart designer tag ⏩. This enables the Ribbon floating toolbar:



2. From the main toolbar, click the **Change Ribbon's Visual Style** button  . The **Visual Style** dialog box appears.
3. From the **Ribbon Visual Style** drop-down list, select the new color scheme from the list, for example, **Office2007Black**.

To Change the Visual Style Using the Properties Window

Optionally, you can change the C1Ribbon.VisualStyle property using the Properties window. Click the Ribbon to reveal the Ribbon properties in the Properties window. Locate the C1Ribbon.VisualStyle property's drop-down arrow and select the new Windows XP theme from the list, for example, **Office2007Black**.

To Change the Visual Style Programmatically

To change the Ribbon's visual style to **Office2007Black**, add the following code to your project:

To write code in Visual Basic

Visual Basic

```
' type the Imports directive for the namespace
Imports C1.Win.C1Ribbon
```

```
Private Sub Form1_Load(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles MyBase.Load
    Me.C1Ribbon1.VisualStyle = C1.Win.C1Ribbon.VisualStyle.Office2007Black
End Sub
```

To write code in C#

C#

```
// type the using directive for the namespace
using C1.Win.C1Ribbon;

private void Form1_Load(object sender, System.EventArgs e)
{
    this.c1Ribbon1.VisualStyle = C1.Win.C1Ribbon.VisualStyle.Office2007Black;
}
```

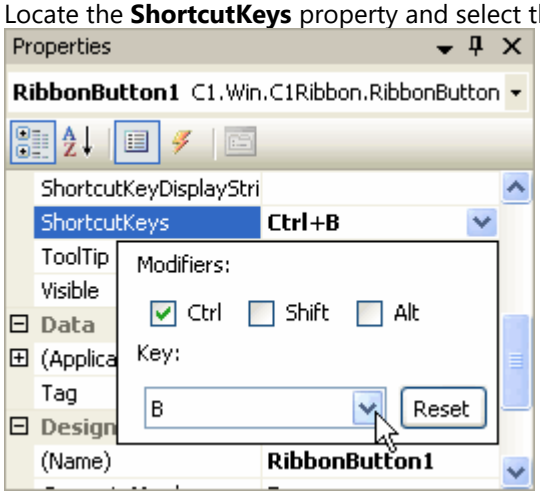
Creating Shortcut Keys

You can use the keyboard to complete specific commands. To make text bold, for example, you could use a direct key combination. To trigger the action, the keys need to be pressed together and most, but not all, key combinations involve pressing CTRL plus other keys (for example, CTRL+B to make text bold).

Note: This topic assumes that you have added a Ribbon button to the group and created a RibbonToggleButton.Click event handler for the **Bold** button.

Complete the following steps:

1. Change the Windows Form to a Ribbon Form.
2. Select the Ribbon button (in this case, the **Bold** button) to display its properties in the Properties window.
3. Locate the **ShortcutKeys** property and select the **Ctrl** check box and select **B** from the drop-down list



4. Click outside the **Shortcut Keys** editor to accept the changes.
5. Save and run the application.
   Now when you run the application pressing the CTRL+B key combination will trigger the **RibbonToggleButton.Click** event for the **Bold** button and make the selected text Bold font style.

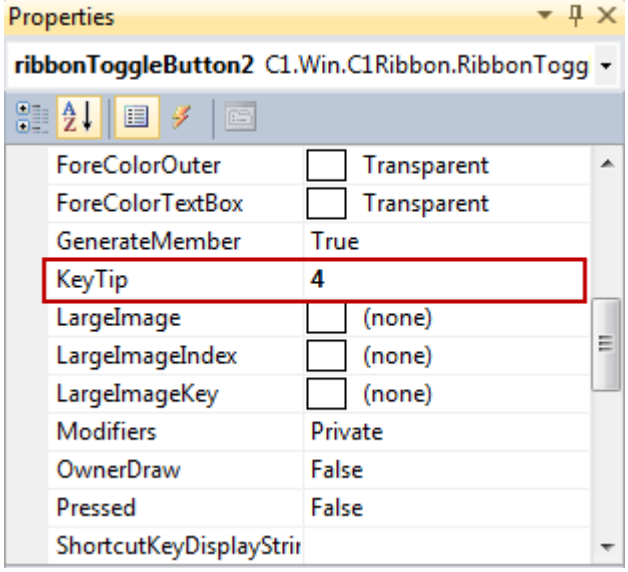
Creating and Displaying Key Tips

C1Ribbon supports Key Tips. These offer you an easy way to display all the commands on the Ribbon Toolbar and also allow you to move between elements on the C1Ribbon control using your keyboard commands. Key Tips use the SupportsKeyTips property and the CaptureF10Key property on the main C1Ribbon control and the **KeyTip** property for the individual elements of the C1Ribbon control.

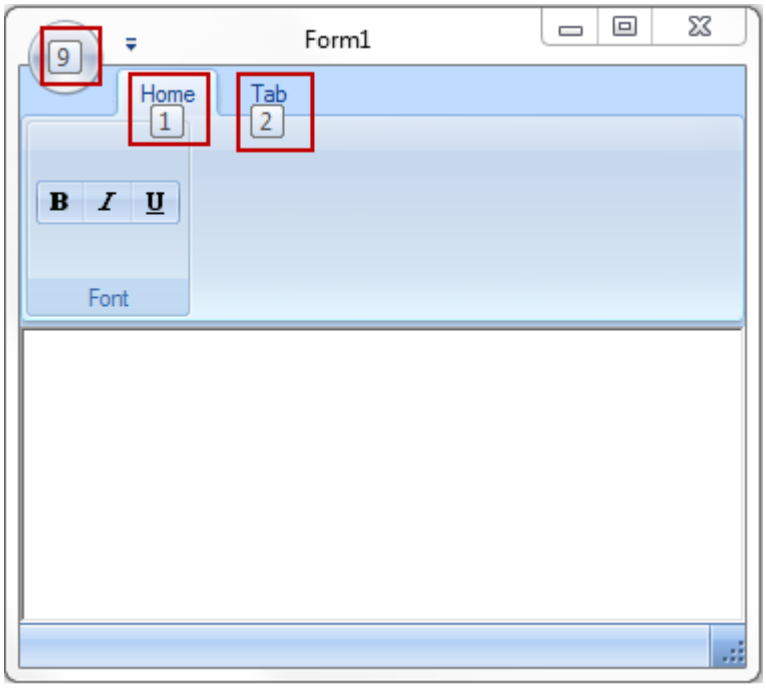
> **Note:** This topic assumes that you have added a Ribbon button to the group and created a RibbonToggleButton.Click event handler for the **Bold** button.

Follow these steps:

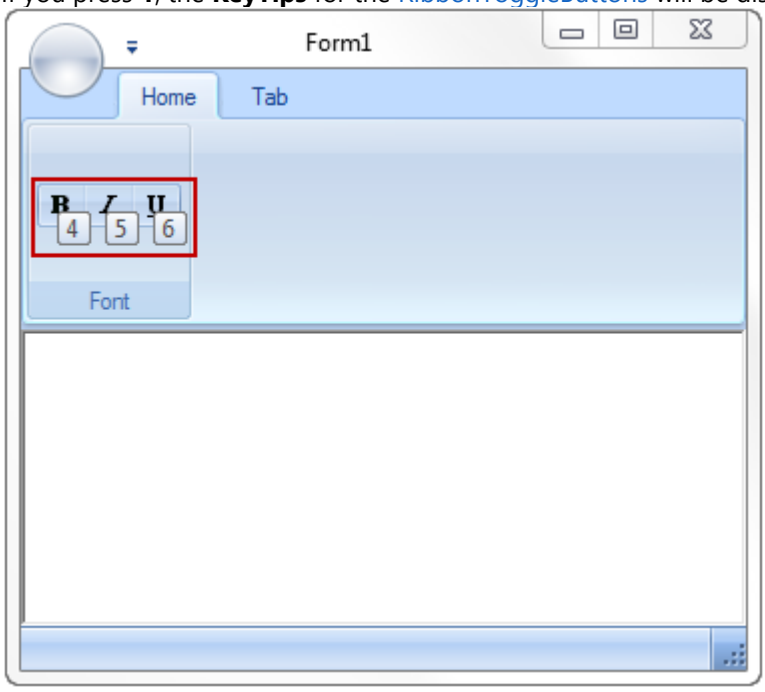
1. Select the tab that contains your RibbonToggleButton. View the RibbonTab properties in the Properties window and locate the **KeyTips** property. Note that this property is set to **1** by default.
2. Select the **Bold** button on your C1Ribbon control. Locate the **KeyTips** property for the RibbonToggleButton in the Properties window.
3. For this Help, set your RibbonToggleButton.KeyTip property to **4**, as in the following image.



4. Create **KeyTips** for other elements on the C1Ribbon control. **KeyTip**s can be used for RibbonGroups, RibbonTabs, RibbonButtons, etc.
5. Press F5 to run your application. Press **Alt** to display the **KeyTips** and then press **F10** to show or hide the **KeyTips**. Your C1Ribbon application should resemble the following image:



6. If you press **1**, the **KeyTips** for the RibbonToggleButtons will be displayed.



Displaying ToolTips for the Ribbon Items

To display a ToolTip for a Ribbon item, use the smart designer, Properties window, or add code to set the RibbonItem.ToolTip property. Each option is described below.

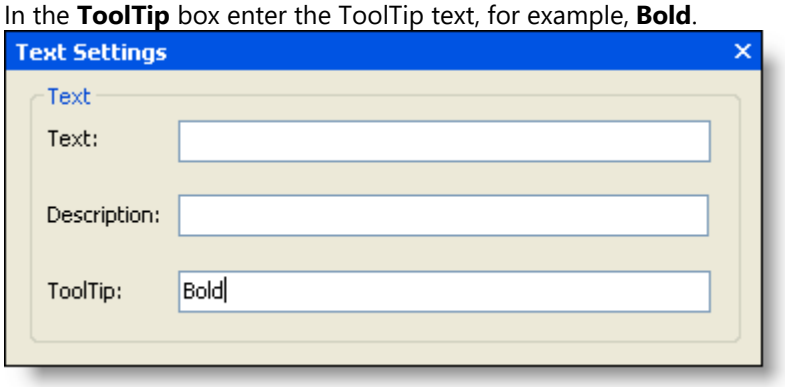
Note: To learn how to create a rich ToolTip, see Creating a Rich ToolTip.

To Display ToolTips Using the Smart Designer

Complete the following steps:

1. Select the toggle button to activate the element and to enable the toggle button's floating toolbar:

2. From the floating toolbar, click the **Text Settings** button  . The **Text Settings** dialog box appears.
3. In the **ToolTip** box enter the ToolTip text, for example, **Bold**.



4. Click the **x** or click outside the **Text Settings** dialog box to accept your changes.
5. Save and run your application. The "Bold" ToolTip text will appear for the Ribbon toggle button when you mouse over the **Bold** button.

To Display ToolTips Using the Properties Window

Optionally, you can set the RibbonItem.ToolTip property using the Properties window. Click the toggle button to reveal the item's properties in the Properties window. Locate the RibbonItem.ToolTip property and enter the ToolTip text in the box, for example, **Bold**.

To Display ToolTips Programmatically

> **Note:** In the following example an embedded resource containing the (16x16) image is used: *Bold.png*. To embed a resource, from the **Project** menu, choose ***YourProjectName* Properties**. From the **Resources** tab, select **Add Resource** and choose to add an existing file or add a new one.

To set the ToolTip text for a toggle button to Bold, add the following code to your project:

To write code in Visual Basic

Visual Basic

```
' type the Imports directive for the namespace
Imports C1.Win.C1Ribbon

Private Sub Form1_Load(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles MyBase.Load
    With RibbonToggleButton1
        ' No text label
        .Text = ""
        ' Set the 16x16 image
```

```
        .SmallImage = My.Resources.Bold;
        ' Show a ToolTip
        .ToolTip = "Bold"
    End With
End Sub
```

To write code in C#

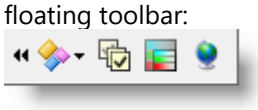
C#

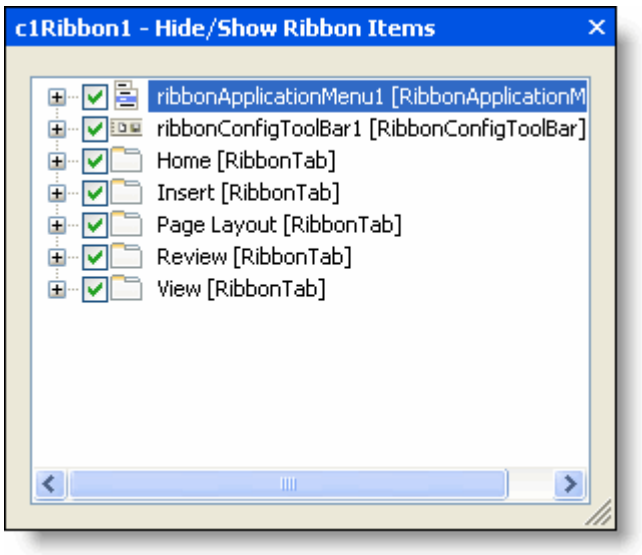
```
// type the using directive for the namespace
using C1.Win.C1Ribbon;

private void Form1_Load(object sender, System.EventArgs e)
{
    // No text label
    this.ribbonToggleButton1.Text = "";
    // Set the 16x16 image
    this.ribbonToggleButton1.SmallImage = My.Resources.Bold;
    // Show a ToolTip
    this.ribbonToggleButton1.ToolTip = "Bold";
}
```

Hiding/Showing Ribbon Items Using the Tree-based Designer

You can easily customize the design of the Ribbon at design time using the smart designers. This topic shows how to modify the design of the Ribbon using the **Hide/Show Ribbon Items** designer.

Complete the following steps:

1. Using your mouse pointer, hover over the Ribbon and click the smart designer tag  . This enables Ribbon floating toolbar:

2. From the main floating toolbar, click the **Hide/show Ribbon items** button  . The tree-based designer appears:



3. From the **Hide/Show Ribbon Items** designer, you can simply deselect the check box to remove objects from the Ribbon. For this example, deselect the following object: **View [RibbonTab]**.
   Notice that the tab has been removed from the Ribbon. You can easily add the tab back to the Ribbon by selecting the **View [RibbonTab]** check box in the designer.

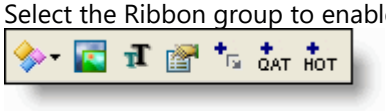
Lining Up Combo/Edit Boxes on a Group

You can line up text box parts of several combo/edit boxes vertically on a Ribbon group using the **RibbonComboBox**.RibbonComboBox.GapBeforeTextArea and RibbonTextBox. GapBeforeTextArea properties. This property specifies the gap (in pixels) between the label and text box parts of the element. The maximum allowable value for this property has been set to 80.

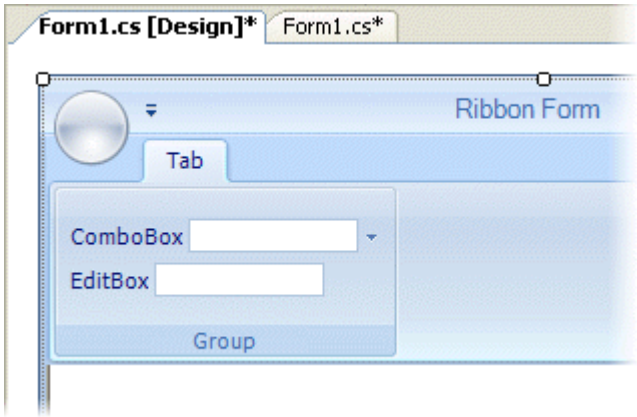
To Line up Combo/Edit Boxes on a Group Using the Smart Designer and Properties Window

Complete the following steps:

1. Select the Ribbon group to enable the group's floating toolbar:



2. Click the **Actions** drop-down button. This reveals a list of Ribbon items to add to the group.
3. Select **Add ComboBox** and then **Add EditBox** from the list. This adds each item to the Ribbon group:



Notice that the text box parts do not line up vertically. The next step shows how to specify the gap between

the label and text box parts of the element so that the parts line up.

> **Note:** Depending on the length of your label, you may need to increase or decrease the size of the gap so that the parts line up vertically.

4. Select the combo box to activate it and from the Ribbon combo box's Properties window:
   - Locate the RibbonComboBox.GapBeforeTextArea property and set the gap (in pixels) to **3**.
   - Locate the RibbonComboBox.Label property and set the text to "Favorites:".
5. Next, select the edit box to activate it and from the Ribbon edit box's Properties window:
   - Locate the RibbonComboBox.GapBeforeTextArea property and set the gap (in pixels) to **28**.
   - Locate the RibbonTextBox.Label property and set the text to "Find:".

To Line up Combo/Edit Boxes on a Group Programmatically

Optionally, you can add the Ribbon combo/edit boxes to the Ribbon group and specify the gap between the label and text box parts of the combo/edit box elements using code.

Add the following code to your project:

To write code in Visual Basic

Visual Basic

```
' type the Imports directive for the namespace
Imports C1.Win.C1Ribbon

Private Sub Form1_Load(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles MyBase.Load
    ' add a RibbonComboBox and RibbonTextBox items
    ' to the Ribbon group
    Dim ComboBox1 As RibbonComboBox = New RibbonComboBox()
    Dim EditBox1 As RibbonTextBox = New RibbonTextBox()
    RibbonGroup1.Items.Add(ComboBox1)
    RibbonGroup1.Items.Add(EditBox1)

    ' add a label
    ComboBox1.Label = "Favorites:"
    EditBox1.Label = "Find:"

    ' specify the gap between the "label" and "text box"
    ' parts of the combo/edit box elements
    ComboBox1.GapBeforeTextArea = 3
    EditBox1.GapBeforeTextArea = 28
End Sub
```

To write code in C#

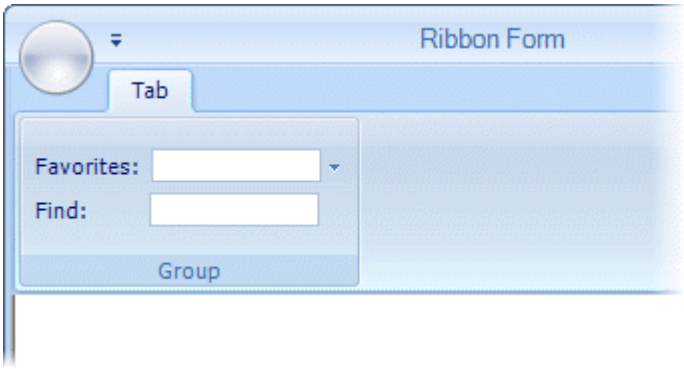
C#

```
// type the using directive for the namespace
using C1.Win.C1Ribbon;

private void Form1_Load(object sender, System.EventArgs e)
{
    // add a RibbonComboBox and RibbonTextBox items
```

```
    // to the Ribbon group
    RibbonComboBox ComboBox1 = new RibbonComboBox();
    RibbonTextBox EditBox1   = new RibbonTextBox();
    ribbonGroup1.Items.Add(ComboBox1);
    ribbonGroup1.Items.Add(EditBox1);

    // add a label
    ComboBox1.Label = "Favorites:";
    EditBox1.Label = "Find:";

    // specify the gap between the "label" and "text box"
    // parts of the combo/edit box elements
    ComboBox1.GapBeforeTextArea = 3;
    EditBox1.GapBeforeTextArea = 28;
}
```

This topic illustrates the following:

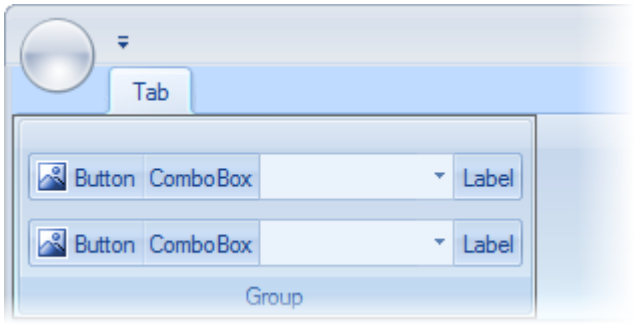
With the GapBeforeTextArea property set for the Ribbon combo box and edit box items, the text box parts line up vertically on the Ribbon group:



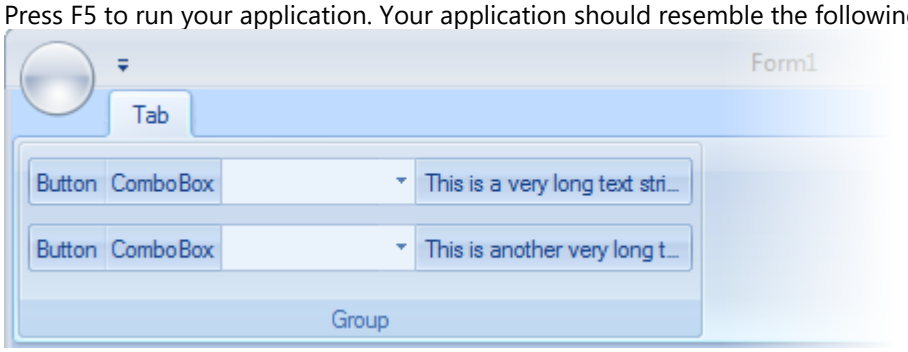
Aligning Multiple Labels

C1Ribbon allows you to align multiple labels within your RibbonGroup using the MaxTextWidth and MinTextWidth properties. For this help, we will create two RibbonToolBars containing the default RibbonButton, a RibbonComboBox, and a RibbonLabel.

1. Add two RibbonToolBars to your application. By default, the RibbonToolBar contains a RibbonButton.
2. Float over the upper left corner of the first Toolbar until the glyph appears. Click the action button and select **Add ComboBox** from the list.
3. Click the glyph again and click the action button again. Select **Add Label** from the list.
4. Repeat steps 1- 3 with the second Toolbar.
5. The Ribbon application should appear as in the following image:



6. Select the RibbonLabel on the first Toolbar and choose Text Settings from the RibbonLabel Toolbar. In the **Text** textbox, enter **This is a very long text string**.
7. In the RibbonLabel Properties window, scroll down to the MaxTextWidth property and set it to **130**. Enter the same number for the MinTextWidth property.
8. Select the RibbonLabel on the second Toolbar and choose Text Settings from the RibbonLabel Toolbar. In the **Text** textbox, enter **This is another very long text string**.
9. In the RibbonLabel Properties window, scroll down to the MaxTextWidth property and set it to **130**. Enter the same number for the MinTextWidth property.
10. Press F5 to run your application. Your application should resemble the following image:



Working with the Application Menu

The following topics explain how to complete various application menu tasks, from creating the application menu to customizing the appearance of the application button.

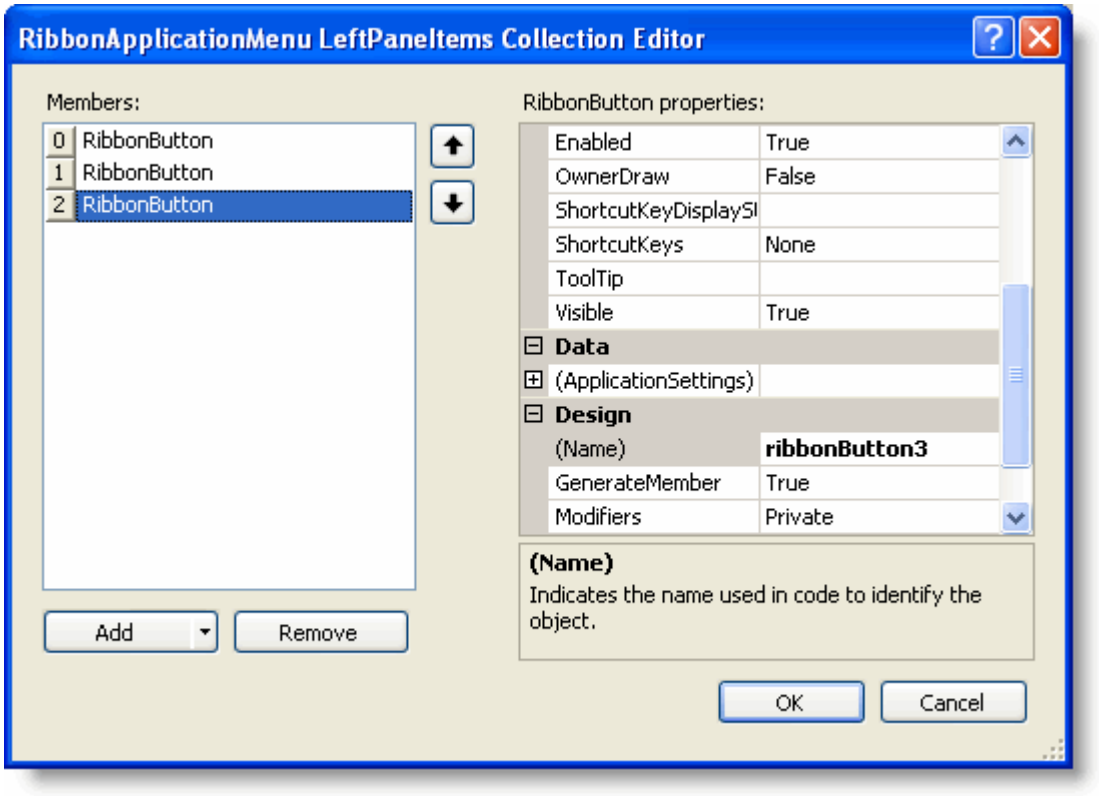
Creating the Application Menu

By default, the Application menu is empty, and you can add as many or as few items to the menu to fit your needs. The following steps demonstrate how to customize the main application button's drop-down menu.

To Create the Start Menu Using the Collection Editor

Complete the following steps:

1. Click the Application button to reveal the list of Application menu properties in the Properties window.
2. From the Properties window, select the LeftPaneItems property and click the **ellipsis** button at the right side of the **(Collection)**.
   The **RibbonApplicationMenu LeftPaneItems Collection Editor** appears.
3. Click the **Add** drop-down arrow and select RibbonButton. Repeat this step to add two more Ribbon buttons. You should now have three buttons added to the **Members** list:



4. Select the first button in the list to edit its properties using the collection editor's Properties window. Set the following properties for the first button:
    - By default, the RibbonButton.Text property is set to **Button**. Delete the text and set it to **&New**.
    - Click the RibbonItem.LargeImage property drop-down button, then click the second drop-down button and choose the **New** image from the list.
5. Select the second button in the list and set the following properties using the collection editor's Properties window:
    - By default, the RibbonButton.Text property is set to **Button**. Delete the text and set it to **&Open**.
    - Click the RibbonItem.LargeImage property drop-down button, then click the second drop-down button and choose the **Open** image from the list.
6. Select the third button in the list and set the following properties using the collection editor's Properties window:
    - By default, the RibbonButton.Text property is set to **Button**. Delete the text and set it to **&Save**.
    - Click the RibbonItem.LargeImage property drop-down button, then click the second drop-down button and choose the **Save** image from the list.
7. Click **OK** to close the collection editor.

To Create the Start Menu Programmatically

Note: In the following example embedded resources containing the following (32x32) images are used: *NewBtn.png*, *OpenBtn.png*, and *SaveBtn.png*. To embed a resource, from the **Project** menu, choose ***YourProjectName* Properties**. From the **Resources** tab, select **Add Resource** and choose to add an existing file or add a new one.

To create a Start menu of commands, add the following code to your project:

To write code in Visual Basic

Visual Basic

```
' Include the Imports directive for the namespace
```

```
Imports C1.Win.C1Ribbon

Private Sub Form1_Load(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles MyBase.Load
    ' Create the items for the left pane
    Dim NewButton As RibbonButton = New RibbonButton()
    Dim OpenButton As RibbonButton = New RibbonButton()
    Dim SaveButton As RibbonButton = New RibbonButton()

    ' Add items to the left pane menu
    Me.C1Ribbon1.ApplicationMenu.LeftPane.Items.Add(NewButton)
    Me.C1Ribbon1.ApplicationMenu.LeftPane.Items.Add(OpenButton)
    Me.C1Ribbon1.ApplicationMenu.LeftPane.Items.Add(SaveButton)

    ' Set properties for the left pane items
    NewButton.Text = "&New"
    NewButton.LargeImage = My.Resources.Resources.NewBtn
    OpenButton.Text = "&Open"
    OpenButton.LargeImage = My.Resources.Resources.OpenBtn
    SaveButton.Text = "&Save"
    SaveButton.LargeImage = My.Resources.Resources.SaveBtn
End Sub
```

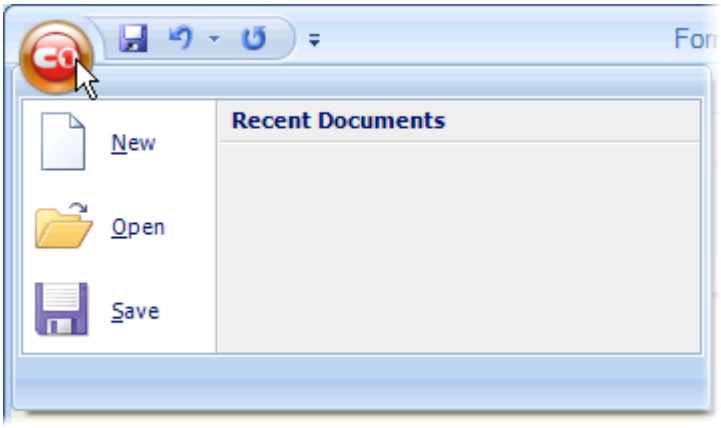
To write code in C#

C#

```
// Include the using directive for the namespace
using C1.Win.C1Ribbon;

private void Form1_Load(object sender, System.EventArgs e)
{
    // Create the items for the left pane
    RibbonButton NewButton = new RibbonButton();
    RibbonButton OpenButton = new RibbonButton();
    RibbonButton SaveButton = new RibbonButton();

    // Add items to the left pane menu
    this.c1Ribbon1.ApplicationMenu.LeftPane.Items.Add(NewButton);
    this.c1Ribbon1.ApplicationMenu.LeftPane.Items.Add(OpenButton);
    this.c1Ribbon1.ApplicationMenu.LeftPane.Items.Add(SaveButton);

    // Set properties for the left pane items
    NewButton.Text = "&New";
    NewButton.LargeImage = Properties.Resources.NewBtn;
    OpenButton.Text = "&Open";
    OpenButton.LargeImage = Properties.Resources.OpenBtn;
    SaveButton.Text = "&Save";
    SaveButton.LargeImage = Properties.Resources.SaveBtn;
}
```

This topic illustrates the following:

Run the application and click the main application button to reveal the drop-down menu of commands:



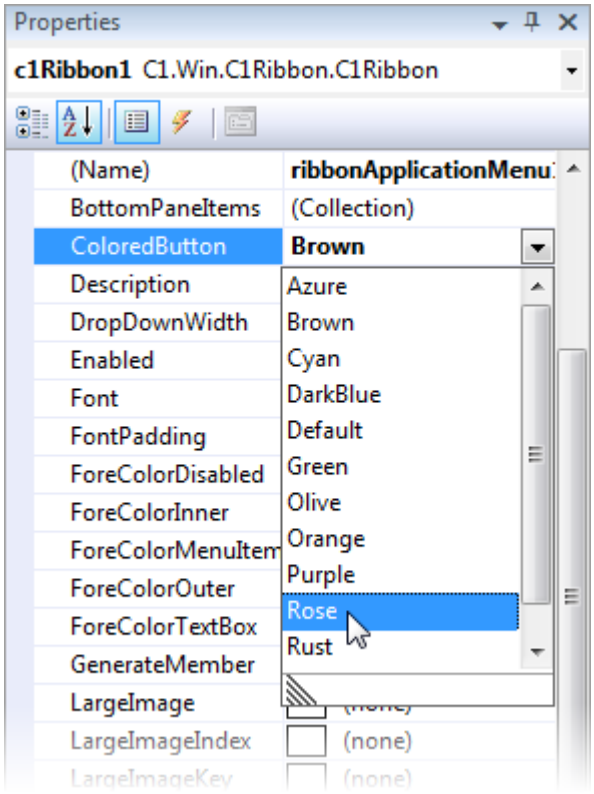
Changing the Color of the Application Button

When the Ribbon is displaying a Windows 7-style application button, you can change the color of the button to another color using the RibbonApplicationMenu.ColoredButton property. This topic assumes that you already have already set the Ribbon's application button to the Windows 7-style. For more information, see Making a Windows-7 Style Application Button.

To Change the Color of the Application Button Using the Smart Designer

Complete the following steps:

1. In the Properties window, expand the **Application Menu** node.
2. Click the ColoredButton drop-down arrow and select a color from the list. For this example, pick **Rose**.



To Change the Color of the Application Button in Code

To change the color of the application to Rose using code, add the following code to the project:

To write code in Visual Basic

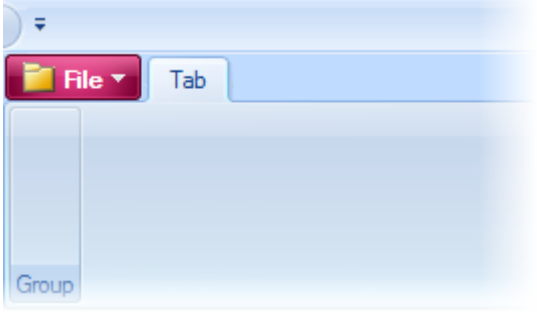
Visual Basic
`C1Ribbon1.ApplicationMenu.ColoredButton = C1.Win.C1Ribbon.ColoredButton.Rose`

To write code in C#

C#
`c1Ribbon1.ApplicationMenu.ColoredButton = C1.Win.C1Ribbon.ColoredButton.Rose;`

This topic illustrates the following:

The result of this topic will resemble the following image:

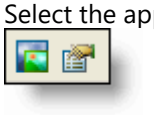
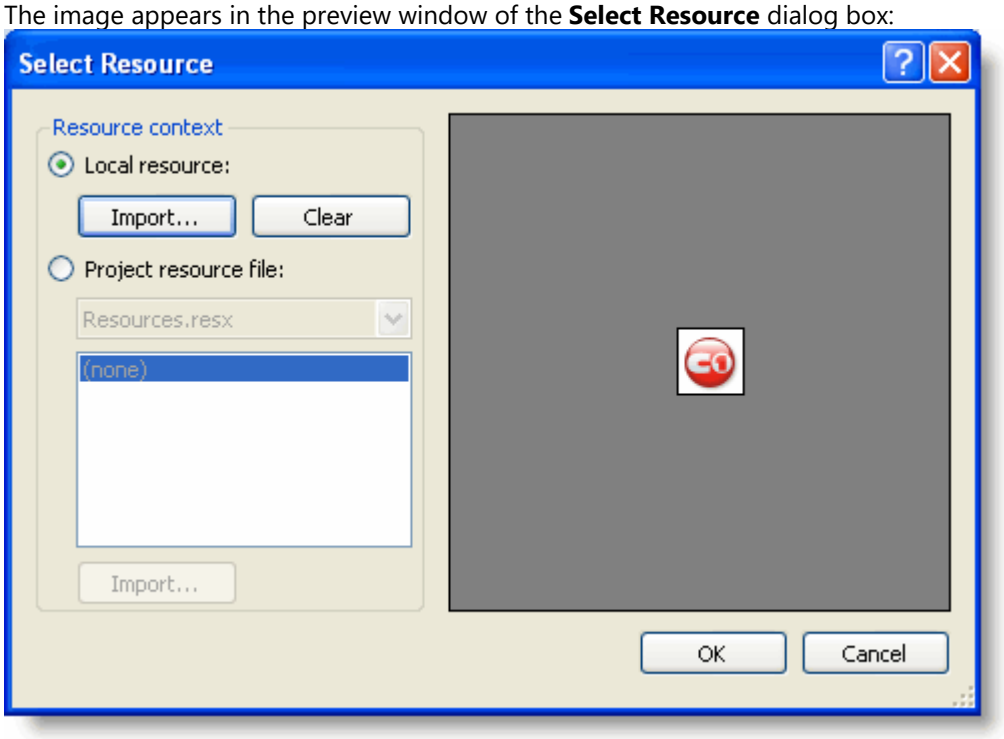


Importing a Custom Image for the Application Button

You can customize the main application button (RibbonApplicationMenu) to fit your needs. To display a custom image for the application button, use the smart designer, Properties window, or add code. Each option is described below.

To Import a Custom Image Using the Smart Designer

Complete the following steps:

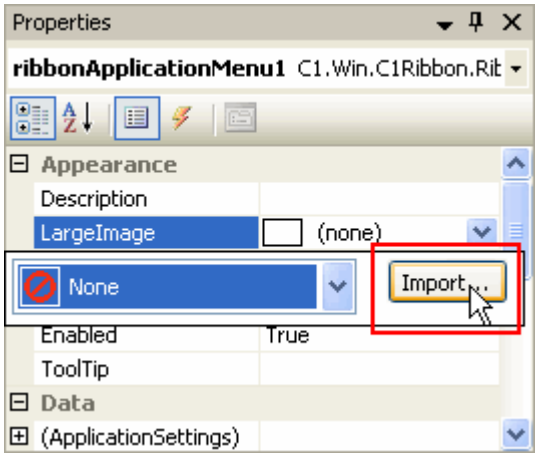
1. Select the application button to activate the element and enable the application button floating toolbar:

2. Click the **Change Image** button. The **Change Image** dialog box appears.
3. Click the 32x32, **Import** button. The **Select Resource** dialog box appears.
4. Select the **Local resource** option (or select Project resource file if you have an embedded image), then click the **Import** button.
5. The **Open** dialog box appears. Browse to your custom image and click **Open**.
The image appears in the preview window of the **Select Resource** dialog box:


6. Click **OK**. The image now appears for the application button.

To Import a Custom Image Using the Properties Window

Complete the following steps:

1. Click the application button to reveal its properties in the Properties window.
2. Click the RibbonItem.LargeImage drop-down arrow, and select **Import**.



3. The **Select Resource** dialog box appears. Follow steps 4 – 6 above.

To Import a Custom Image Programmatically

To display a custom image for the application button, add the following code to your project:

To write code in Visual Basic

Visual Basic

```
' type the Imports directive for the namespace
Imports C1.Win.C1Ribbon

Private Sub Form1_Load(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles MyBase.Load
    Me.C1Ribbon1.ApplicationMenu.LargeImage = My.Resources.Resources.AppButtonImage
End Sub
```

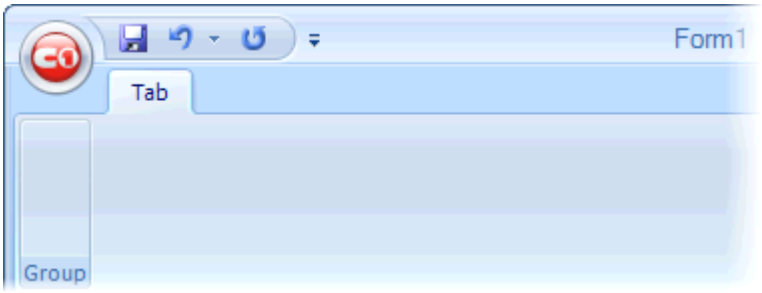
To write code in C#

C#

```
// type the using directive for the namespace
using C1.Win.C1Ribbon;

private void Form1_Load(object sender, System.EventArgs e)
{
    this.c1Ribbon1.ApplicationMenu.LargeImage = Properties.Resources.AppButtonImage;
}
```

This topic illustrates the following:

The following Ribbon application button uses a 32x32 custom image:



Making a Windows 7-Style Application Button

Even if you're using one of the Office2007 visual styles, you can make the Ribbon's menu application button mimic the Windows 7-style application button that appears in applications like Word and Paint. In this topic, you'll create a Windows 7-style application button by setting the **RibbonApplicationMenu.Win7Look** property. You'll also add text and an image to that button.

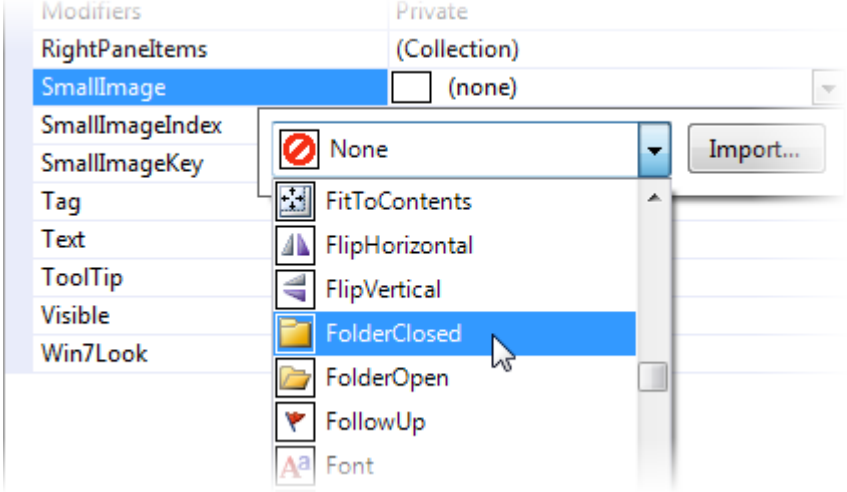
> **Tip:** The ability to programmatically reference the standard Ribbon icons is not supported by **Ribbon for WinForms**. If you need to add the icons at run time, you can use images from the Visual Studio 2008 Image Library or the Visual Studio 2010 Image Library. These are installed by default with Visual Studio, and they are located in the following directories:
>
> - **Visual Studio 2008**: \...\Program Files\Microsoft Visual Studio 9.0\Common7\VS2008ImageLibrary\1033\
> - **Visual Studio 2010**: \...\Program Files\Microsoft Visual Studio 10.0\Common7\VS2010ImageLibrary\1033\

Complete the following steps:

1. In the Properties window, expand the **Application Menu** node.
2. Set the following properties:
   - Set the RibbonApplicationMenu.Win7Look property to **True** to change the look of the Ribbon application button.

     > **Note**: You don't have to do this if you are using one of the Office2010 visual styles, as those already have the rectangular buttons.

   - Set the RibbonApplicationMenu.Text property to "File" to add text to the Ribbon application button.
   - Set the RibbonItem.SmallImage property to **FolderClosed** by clicking the drop-down arrow, clicking the drop-down arrow on the **SmallImage** editor screen, and selecting **FolderClosed** from the list.



This topic illustrates the following:

The result of this topic will resemble the following image: