#### Context Menu Tasks

This section shows how to perform specific context menu tasks.

**See Also**

[Adding a C1ContextMenu to a Control](#_bookmark151)

[Retrieving the C1ContextMenu Control Attached to the C1TextBox](#_bookmark152) [Linking the C1Context Menu to a NotifyIcon Control](#_bookmark153)

[Adding a C1ContextMenu to a C1DockingTab](#_bookmark154)

###### Adding a C1ContextMenu to a Control

You can add a [C1ContextMenu](#_bookmark576) to a control at design time or through code. Click on either of the following links to expand the steps for the designer or for the code.

To add a C1ContextMenu to a Control at design time

To create a context menu and associate it with a menu item, complete the following tasks:

**Note:** In this example, a [C1ContextMenu](#_bookmark576) is associated with a menu item.

1. Place a [C1MainMenu](#_bookmark773) on the form, right-click **New Command**, and then select **Edit** from its context menu. The **Link to Command** designer appears.
2. In the **Link to Command** designer, change the following command properties.
	* Text textbox to **File**
	* Name textbox to **MenuFile**
3. Select **ContextMenu from** the **Create a new command** listbox. This will create a

command with a submenu which can be used as a context menu. Select **OK**, in the **Link to Command** dialog box. The new **C1MainMenu**, **File** will appear.

1. Right-click on the **New Command** item in the **File** menu and select **Edit** from its context menu. Set the properties in the **Link to Command** designer to the following:
	* **Text** to **New**
	* **Name** to **cmdFileNew**
2. Select **C1Command** from the **Create a new command** listbox.
3. Click **OK**.
4. Right-click the **New** menu, select **Append Item** from its context menu, and then set its properties in the **Link to Command** designer to the following:
	* **Text** to Open
	* **Name** to cmdFileOpen
5. Select **C1Command** from the Create a new command listbox.
6. Click **OK**.
7. Select the **Windows Form** tab from the Toolbox and place the **RichTextBox** control onto your form using a drag-and-drop operation.
8. Select the **richTextBox1** control from the **Properties** drop-down list box then set its **Dock**

property to **Fill**.

1. Select the **MenuFile** from the Properties drop-down list, then select the **Category** property and enter **File** in its box.
2. Select **richTextBox1** from the Properties drop-down list, then select the **C1ContextMenu on C1CommandHolder** property, and then select **MenuFile**.
3. Run the application and then right-click your mouse anywhere in the rich text box. The context menu for the **File** menu appears like the following image.

To add a C1ContextMenu to a control programmatically

To add a C1ContextMenu to a control, complete the following steps:

1. Attach the **C1.Win.C1Command** to your reference project located in the solution explorer, and then add the **C1.Win.C1Command** namespace to your source file.
2. Place the **TextBox** control onto your form using a drag-and-drop operation.
3. To create a [C1CommandHolder](#_bookmark350) so it will hold the command, double-click on the form control to create a **Form\_Load** event handler and add the following code:

**Note:** Use the try and catch method if you want to create more than one command holder for a form. This will catch and ignore the exception when the second attempt to create another commandholder fails.

1. To write code in Visual Basic



|  |  |
| --- | --- |
| Visual Basic | Copy Code |
| Dim ch As C1.Win.C1Command.C1CommandHolder ch = C1CommandHolder.CreateCommandHolder(Me) |

1. To write code in C#

C# Copy Code

C1CommandHolder ch = C1CommandHolder.CreateCommandHolder(this);

1. Create and set up a copy command, then use the [Click](#_bookmark283) event to copy the command when you click on the menu item. Also set up the query handler for the commands so their command is kept up to date automatically by c1command. Add the following code within the form load event handler:

To write code in Visual Basic

|  |  |
| --- | --- |
| Visual Basic | Copy Code |
| 'Create and set up the Copy commandDim cmdCopy As C1Command = ch.CreateCommand() cmdCopy.Text = "&Copy"AddHandler cmdCopy.Click, AddressOf clickCopyAddHandler cmdCopy.CommandStateQuery, AddressOf queryCopy |

To write code in C#

|  |  |
| --- | --- |
| C# | Copy Code |
| // Create and set up the Copy command C1Command cmdCopy = ch.CreateCommand(); cmdCopy.Text = "&Copy";cmdCopy.Click += new C1.Win.C1Command.ClickEventHandler(clickCopy); cmdCopy.CommandStateQuery += new C1.Win.C1Command.CommandStateQueryEventHandler(queryCopy); |

1. Create a context menu to hold the copy command, then assign the context menu to the text box control. In order to create a context menu you must create a command in the [C1CommandHolder](#_bookmark350) and assign it to the context menu. Use the **C1CommandHolder.CreateCommand** and **C1CommandHolder.SetC1ContextMenu** methods from the [C1CommandHolder](#_bookmark350) class. Add the following code within the **Form\_Load** event handler:

To write code in Visual Basic

|  |  |
| --- | --- |
| Visual Basic | Copy Code |
| Dim cm As C1ContextMenu = Ctype(ch.CreateCommand(GetType(C1ContextMenu)), C1ContextMenu)'Fill it with the links to the commands cm.CommandLinks.Add(New C1CommandLink (cmdCopy)) ch.SetC1ContextMenu(TextBox1, cm) |

To write code in C#

|  |  |
| --- | --- |
| C# | Copy Code |
| C1ContextMenu cm = ch.CreateCommand(typeof(C1ContextMenu)) as C1ContextMenu;// Fill it with the links to the commands cm.CommandLinks.Add(new C1CommandLink(cmdCopy)); ch.SetC1ContextMenu(textBox1, cm); |

1. Invoke the **clickCopy** method to handle the copy command action. Use the **queryCopy** method to provide the current state of the copy command. When you click the copy command from the context menu, the current text will be copied into the textbox. You can achieve this by using the following code:

To write code in Visual Basic

|  |  |
| --- | --- |
| Visual Basic | Copy Code |
| Private Sub clickCopy(ByVal sender As Object, ByVal e As C1.Win.C1Command.ClickEventArgs)Me.textBox1.Copy()End Sub'provides the current state of the copy commandPrivate Sub queryCopy(ByVal sender As Object, ByVal e As C1.Win.C1Command.CommandStateQueryEventArgs)e.Enabled = Me.textBox1.SelectionLength > 0End Sub |

To write code in C#

|  |  |
| --- | --- |
| C# | Copy Code |
| private void clickCopy(object sender, C1.Win.C1Command.ClickEventArgs e){this.textBox1.Copy();}//provides the current state of the copy commandprivate void queryCopy(object sender, C1.Win.C1Command.CommandStateQueryEventArgs e){e.Enabled = this.textBox1.SelectionLength > 0;} |

1. Save and run the application. Enter some text into the text box, then right-click the text to make the context menu appear. The following image shows how the context menu appears next to the text box control at run time:



###### Retrieving the C1ContextMenu Control Attached to the C1TextBox

Use the [GetC1ContextMenu](#_bookmark365) method of [C1CommandHolder](#_bookmark350) class to determine which [C1ContextMenu](#_bookmark576) is attached to a control. The [GetC1ContextMenu](#_bookmark365) method returns the context menu attached to a specific control.

To retrieve the name of the **C1ContextMenu** control attached to the **C1TextBox1**, use the following code:

To write code in Visual Basic

|  |  |
| --- | --- |
| Visual Basic | Copy Code |
| 'retrieves the contextmenu attached to the C1TextBox control Dim contextMenu As C1.Win.C1Command.C1ContextMenu contextMenu = C1CommandHolder1.GetC1ContextMenu(C1TextBox1) MessageBox.Show(ContextMenu.Name) |

To write code in C#

|  |  |
| --- | --- |
| C# | Copy Code |
| //retrieves the contextmenu attached to the C1TextBox controlC1.Win.C1Command.C1ContextMenu contextMenu;contextMenu = C1CommandHolder1.GetC1ContextMenu(C1TextBox1); MessageBox.Show(ContextMenu.Name); |

###### Linking the C1Context Menu to a NotifyIcon Control

[C1ContextMenu](#_bookmark576) supports linking to the standard **NotifyIcon** Control.

To link a [C1ContextMenu](#_bookmark576) to a **NotifyIcon** Control at design time, follow these steps:

* 1. Double-click on the [C1ContextMenu](#_bookmark576) control to add it to the component tray. This will also add a **C1ComponentHolder** control to the component tray.
	2. Right-click on the [C1ContextMenu](#_bookmark576) control and select **Edit** from its context menu. A **New Command** item will appear on the form.
	3. Right-click the **New Command** item and select **Edit** from its context menu to open the

**Link to Command** designer.

* 1. Change the following command properties in the **Link to Command** designer:
		+ Change the **Name** textbox text to **MenuFile**.
		+ Change the **Text** textbox text to **Menu**.
	2. Select [C1CommandMenu](#_bookmark500) from the Create a new command listbox.
	3. Select **OK** in the **Link to Command** designer and the new menu will appear on the form.
	4. Right-click the **New Command** item in the Menu and select **Edit** from its context menu to open the **Link to Command** designer.
	5. Change the following properties in the **Link to Command** designer:
		+ Change the **Name** textbox text to **OpenFile**.
		+ Change the **Text** textbox tect to **Open**.
	6. Select **View | Properties** from the Visual Studio toolbar. Select **C1ContextMenu1** from the

drop-down list at the top of the Properties window.

* 1. Set the [C1ContextMenu](#_bookmark576) property to **C1ContextMenu1**.
	2. Double-click the **NotifyIcon** component in the Windows Forms Toolbox to add the component to the component tray.
	3. Right-click on the component and select **Choose an icon** from its context menu. Choose an icon to represent the component at run time.
	4. Select **View | Properties** from the Visual Studio toolbar. Select **notifyicon1** from the drop- down list at the top of the Properties window.
	5. Set the [C1ContextMenu](#_bookmark576) property to **C1ContextMenu1** to link the two components.
	6. Run your application. The icon you chose to represent the NotifyIcon component will appear in the System Tray. Note that when you right-click the icon, the context menu opens.

**This topic illustrates**

Linking a C1ContextMenu to a standard Windows Forms control.



###### Adding a C1ContextMenu to a C1DockingTab

Complete the following steps to add a [C1ContextMenu](#_bookmark576) control to a [C1DockingTab](#_bookmark602) control.

1. Locate the [C1ContextMenu](#_bookmark576) control in your Toolbox and double-click the control to add it to your Component Tray. A [C1CommandHolder](#_bookmark350) will also be added to the Component Tray.
2. Use the [C1ContextMenu](#_bookmark576) smart tag to open the [C1ContextMenu](#_bookmark576) **Tasks Menu**. Select **Add Item** from the **Tasks Menu** to add an item to the [C1ContextMenu.](#_bookmark576) Add two more items to

the menu.

1. Place a [C1DockingTab](#_bookmark602) control on the form and open the [C1DockingTab](#_bookmark602) Context Menu.

Select **Add Page** from the context menu. Add several pages to the [C1DockingTab.](#_bookmark602)

Select the C1DockingTab control on your form to view the properties. Click the Events button and locate the **MouseClick** event. Insert **c1DockingTab1\_MouseClick** to create the **MouseClick** event.

1. Right-click on the form and select **View Code** from the list. Insert the following code after the **InitializeComponent()** method:

To write code in Visual Basic

Visual Basic Copy Code

Private Sub c1DockingTab1\_MouseClick(ByVal sender As Object, ByVal e As MouseEventArgs)

If (e.Button = System.Windows.Forms.MouseButtons.Left) Then If (e.X > Me.c1DockingTabPage1.Location.X) Then

c1ContextMenu1.ShowContextMenu(Me.c1DockingTab1, New Point(e.X,

e.Y))

Else

Me.c1DockingTab1.ContextMenu = Nothing

End If End If

End Sub

To write code in C#

|  |  |
| --- | --- |
| C# | Copy Code |
| private void c1DockingTab1\_MouseClick(object sender, MouseEventArgs e){if (e.Button == System.Windows.Forms.MouseButtons.Left){if (e.X > this.c1DockingTabPage1.Location.X){c1ContextMenu1.ShowContextMenu(this.c1DockingTab1, new Point(e.X, e.Y));}elsethis.c1DockingTab1.ContextMenu = null;}} |

1. Press F5 to run your application. Note that when you click on a tab, the C1ContextMenu

opens:

DockingTab Tasks

This section shows how to perform specific docking tab tasks.

**See Also**

[Adding a Scrollbar to a C1DockingTab](#_bookmark156) [Closing a C1DockingTabPage](#_bookmark157)

[Determining if the C1DockingTab is Floating](#_bookmark158) [Displaying Multiple Tab Rows](#_bookmark159)

[Displaying the Same Set of Controls on each C1DockingTabPage](#_bookmark160)

[Enabling DockingTab Docking and Floating](#_bookmark161) [Enabling or Disabling Focus Cues](#_bookmark162)

[Loading and Saving the Layout of the C1DockingTab](#_bookmark163) [Moving Tab Pages at Run Time](#_bookmark164)

[Pinning the C1DockingTab](#_bookmark165)

[Preventing the Tabs from Receiving Focus on Mouse Click](#_bookmark166) [Restricting the Usage of Specific Tabs](#_bookmark167)

###### Adding a Scrollbar to a C1DockingTab

To add a scrollbar to a [C1DockingTab,](#_bookmark602) complete the following steps:

**Note:** The [C1DockingTab](#_bookmark602) does not have its own scrollbar.

1. Place a [C1CommandDock](#_bookmark323) control on the form, and then set its **Dock** property to **Fill**.
2. Add a [C1DockingTab](#_bookmark602) inside the [C1CommandDock.](#_bookmark323)
3. Add a **Panel** control inside the [C1DockingTabPage,](#_bookmark724) and then set the Panel's **Dock** property to **Fill**.
4. Place a few Windows Form controls inside Panel1 of the [C1DockingTabPage.](#_bookmark724)



1. Set **Panel1**'s AutoScroll property to **True**.

The scrollbars appear for the [C1DockingTab.](#_bookmark602)



###### Closing a C1DockingTabPage

You can close the [C1DockingTabPage](#_bookmark724) in code by using the [TabVisible](#_bookmark756) property of the

[C1DockingTabPage.](#_bookmark724) The following code can be used for closing the first page:

To write code in Visual Basic

|  |  |
| --- | --- |
| Visual Basic | Copy Code |
| Me.C1DockingTab1.TabPages(0).TabVisible = False |

To write code in C#

|  |  |
| --- | --- |
| C# | Copy Code |
| this.C1DockingTab1.TabPages(0).TabVisible = False; |

###### Determining if the C1DockingTab is Floating

To find out if the docking tab is currently Floating, use the [Floating](#_bookmark655) (run time) only property of the

[C1DockingTab.](#_bookmark602) This syntax of the property is as following:

To write code in Visual Basic

|  |  |
| --- | --- |
| Visual Basic | Copy Code |
| Public Floating As Boolean |

To write code in C#

|  |  |
| --- | --- |
| C# | Copy Code |
| Public bool Floating {get;} |

###### Displaying Multiple Tab Rows

To display more than one row of tab in a [C1DockingTab,](#_bookmark602) set the [MultiLine](#_bookmark661) property of the

[C1DockingTab](#_bookmark602) class to **True**. To do this, complete the following tasks:

* 1. Set the [MultiLine](#_bookmark661) property of the [C1DockingTab](#_bookmark602) class to **True**.
	2. Right-click on **Page1** of the [C1DockingTabPage](#_bookmark724) and select **Add Page** from its context menu. Add several pages to the [C1DockingTab.](#_bookmark602) The tabs continue to another row.

###### Displaying the Same Set of Controls on each C1DockingTabPage

To display the first set of controls on [C1DockingTabPage](#_bookmark724) for the rest of the [C1DockingTabPage](#_bookmark724)s without duplicating them, perform the following steps:

1. Add a panel to the first [C1DockingTabPage](#_bookmark724) and then set its **Dock** property to **Fill** so it spans the whole [C1DockingTabPage.](#_bookmark724)
2. Add the controls on the panel.
3. In the **SelectedIndexChanging** event handler move the panel from the previous docking tab page to the new one.

To write code in Visual Basic

|  |  |
| --- | --- |
| Visual Basic | Copy Code |
| Me.dockingTab.SelectedTab.Controls.Add (myPanel) |

To write code in C#

|  |  |
| --- | --- |
| C# | Copy Code |
| this.dockingTab.SelectedTab.Controls.Add (myPanel); |

###### Enabling DockingTab Docking and Floating

To add a [C1DockingTab](#_bookmark602) to your form, complete the following basic operations:

1. Place the [C1CommandDock](#_bookmark323) control onto your form using a drag-and-drop operation.
2. The [C1CommandDock](#_bookmark323) will dock to the left side of the form. Select the drop-down arrow in the **C1CommandDock.Dock** property and click the top rectangle. This will dock the **C1CommandDock** control to the top of the form.
3. Place the [C1DockingTab](#_bookmark602) control inside the [C1CommandDock](#_bookmark323) control using a drag-and- drop operation. It will appear like this on your form:
4. Build and run the application. Select **Page1** with your mouse and drag it downward. Your docking tab at run time should look like the following docking tab:



**Note**: You can use the [C1CommandDock](#_bookmark323) property [FloatHide](#_bookmark341) to control how the [C1DockingTabPage](#_bookmark724) behaves at run-time. This property allows you to choose to keep focus on the tab pages when the application loses focus. There are three possible settings for [C1CommandDock.FloatHide:](#_bookmark341) **Default**, **Never**, or **FocusLost**.

###### Enabling or Disabling Focus Cues

You can easily enable or disable focus cues for the C1DockingTab using the C1DockingTab.TabsShowFocusCues property. By default, this property is set to **true**. This topic will cover setting this property to false in design time and in code.

**In Design Time**

In the Properties window, set the [C1DockingTab.TabsShowFocusCues](#_bookmark685) to **false**.

**In Code**

To write code in Visual Basic

|  |  |
| --- | --- |
| Visual Basic | Copy Code |
|  |

c1DockingTab1.TabsShowFocusCues = false

To write code in C#

|  |  |
| --- | --- |
| C# | Copy Code |
| c1DockingTab1.TabsShowFocusCues = false; |

###### Loading and Saving the Layout of the C1DockingTab

You can save and load the layout of the [C1DockingTab](#_bookmark602) by using the [SaveLayout](#_bookmark631) and [RestoreLayout](#_bookmark628)

methods. The syntax for the methods is as follows:

To write code in Visual Basic

|  |  |
| --- | --- |
| Visual Basic | Copy Code |
| 'Saves the current layout of the tabs on the form to the specified file.Shared Sub SaveLayout(form As Form, filename As String)'Restores the previously saved layout of the tabs on the form from the specified fileSub RestoreLayout(form As Form, filename As String) |

To write code in C#

|  |  |
| --- | --- |
| C# | Copy Code |
| //Saves the current layout of the tabs on the form to the specified file.static void SaveLayout(Form form, string filename);//Restores the previously saved layout of the tabs on the form from the specified fileStatic void RestoreLayout(Form form, string filename); |

The following code is an example of how these methods can be applied:

To write code in Visual Basic

|  |  |
| --- | --- |
| Visual Basic | Copy Code |
| //Saves the current layout of the tabs on the form to the specified file.C1DockingTab.SaveLayout(myForm, "myLayoutFile.xml")'Restores the previously saved layout of the tabs on the form from the specified fileC1DockingTab.RestoreLayout(myForm, "myLayoutFile.xml") |

To write code in C#

|  |  |
| --- | --- |
| C# | Copy Code |
| //Saves the current layout of the tabs on the form to the specified file.C1DockingTab.SaveLayout (myForm, "myLayoutFile.xml");//Restores the previously saved layout of the tabs on the form from the specified fileC1DockingTab.RestoreLayout(myForm, "myLayoutFile.xml"); |

###### Moving Tab Pages at Run Time

To move tabs in different positions, use the [CanMoveTabs](#_bookmark649) property. In the properties toolbox set the [CanMoveTabs](#_bookmark649) to **True**.

###### Pinning the C1DockingTab

To pin the [C1DockingTab](#_bookmark602) programmatically, set the [AutoHiding](#_bookmark644) property of the [C1DockingTab](#_bookmark602) to

**False**. Use the following code:

To write code in Visual Basic

|  |  |
| --- | --- |
| Visual Basic | Copy Code |
| Me.C1DockingTab1.AutoHiding = False |

To write code in C#

|  |  |
| --- | --- |
| C# | Copy Code |
| this.C1DockingTab1.AutoHiding = False; |
|  | **Note:** To correctly pin and unpin the C1DockingTab, it must be placed in a C1CommandDock control. |

###### Preventing the Tabs from Receiving Focus on Mouse Click

To prevent the tabs from receiving focus, use the [TabsCanFocus](#_bookmark683) property. Setting this property to

**False** will prevent the tabs from receiving focus on Mouse click.

###### Restricting the Usage of Specific Tabs

You can add a handler to the [SelectedIndexChanging](#_bookmark710) event to test whether the index of the page is the page that you don't want the user to switch to. If so, e.Cancel would be set to **True**. For example, the following code shows how to add a handler to the [SelectedIndexChanging](#_bookmark710) event:

To write code in Visual Basic

|  |  |
| --- | --- |
| Visual Basic | Copy Code |
| Private Sub c1DockingTab1\_SelectedIndexChanging(sender As Object, e As C1.Win.C1Command.SelectedIndexChangingEventArgs)If e.NewIndex = 1 And e.CanCancel Then e.Cancel = TrueEnd If End Sub |

To write code in C#

|  |  |
| --- | --- |
| C# | Copy Code |
| private void c1DockingTab1\_SelectedIndexChanging(object sender, C1.Win.C1Command.SelectedIndexChangingEventArgs e){if(e.NewIndex == 1 && e.CanCancel) e.Cancel = true; |

}