
ComponentOne

Command for WinForms

ComponentOne, a division of GrapeCity

201 South Highland Avenue, Third Floor
Pittsburgh, PA 15206 USA

Website: <http://www.componentone.com>

Sales: sales@componentone.com

Telephone: 1.800.858.2739 or 1.412.681.4343 (Pittsburgh, PA USA Office)

Trademarks

The ComponentOne product name is a trademark and ComponentOne is a registered trademark of GrapeCity, Inc. All other trademarks used herein are the properties of their respective owners.

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 \$25 (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

While you are welcome to make backup copies of the software for your own use and protection, you are not permitted to make copies for the use of anyone else. We put a lot of time and effort into creating this product, and we appreciate your support in seeing that it is used by licensed users only.

Table of Contents

Menus and Toolbars for WinForms Overview	7
Help with WinForms Edition	7
Key Features	8
Menus and Toolbars Key Features	8-9
DockingTab Key Features	9
NavBar Key Features	9-10
OutBar Key Features	10
TopicBar Key Features	10-11
RadialMenu Key Features	11
Overview	12
Terminology	12
Class Hierarchy	12
Commands	12
Other Components and Controls	12-13
Design-Time Support	14-15
In-Place Text Editing	15
Command Smart Tags	15
C1CommandHolder Smart Tag	15-16
C1MainMenu Smart Tag	16-17
C1ContextMenu Smart Tag	17-18
C1NavBar Smart Tag	18
C1NavBar Panel Smart Tag	18-19
C1ToolBar Smart Tag	19-20
C1TopicBar Smart Tag	20-21
C1DockingTab Smart Tag	21-22
C1DockingTabPage Smart Tag	22-23
C1OutBar Smart Tag	23-25
C1OutPage Smart Tag	25-26
Command Context Menus	26-31
Link to Command Designer	31-32
Command Collection Editors	32
Command Collection Editor	32-33
CommandLink Collection Editor	33-35
DockingTabPage Collection Editor	35-36

NavBarPanel Collection Editor	36-37
OutPage Collection Editor	37-38
TopicPage Collection Editor	38-39
TopicLink Collection Editor	39-40
Command Smart Designer	40-41
Smart Designer Floating Toolbars	41-42
C1MainMenu Toolbar	42-46
C1CommandLink Toolbar	46-49
C1CommandMenu Toolbar	49-55
C1DockingTab Toolbars	55-62
C1NavBar Toolbars	62-66
C1OutBar Toolbars	66-69
C1TopicBar Toolbars	69-74
C1ToolBar Toolbar	74-81
Menus and Toolbars Overview	82
Menus and Toolbars Functionality	82-83
Common Objects Used to Create Menus and Toolbars	83
CommandHolder Component	83-84
CommandMenu Command	84
ContextMenu Control	84
CommandControl Command	84
CommandMdiList Command	84-85
Unique Objects Among Menus and Toolbars	85
MainMenu Control	85-86
ToolBar Control	87
Menus Appearance and Behavior	87
Menus Visual Styles	87-89
Look and Feel of Menu Items	89
Special Side Caption Styles in Menus	89
Mouse-Over Styles in Menu Items	89-90
Merging Menus	90
Layout and Text Wrapping in Menus	90
ToolTips in Menus	90
Toolbars Appearance and Behavior	90
Toolbar Visual Styles	90-91
Look and Feel of Toolbars	91

Special Border Styles in Toolbars	91-93
Mouse-Over Styles in Toolbar Buttons	93
Docking and Floating Toolbars	93-94
Embedded Controls in Toolbars	94-95
Run-Time Customization for Toolbars	95-100
Wrapping Toolbar Buttons and Text	100-101
ToolTips in Toolbars	101-102
Toolbar and Button Layout Behavior	102-105
DockingTab Overview	106
DockingTab Appearance and Behavior Properties	106
DockingTab Visual Styles	106-108
Docking Styles	108-109
Tab Styles	109-110
Tab Sizing	110-111
Tab Orientation	111-112
Text Orientation	113-114
Text Editing	114
Hiding, Closing, and Moving Individual Tab Pages	114-115
Mouse-Over Styles in Tab Pages	115
NavBar Overview	116
NavBar Appearance and Behavior Properties	116
NavBar Visual Styles	116-119
NavBar Panel Styles	119-120
NavBar Button Styles	120-121
NavBar Vertical Text	121
Embedded Controls in NavBar Panels	121-122
Run-Time Customization for NavBar	122-124
OutBar Overview	125
OutBar Appearance and Behavior Properties	125
OutBar Visual Styles	125-126
Page Styles	126-127
Embedded Controls in Pages	127
Page Sizing	127
Page Layout and Alignment	127-129
Scroll Bars	129
TopicBar Overview	130

TopicBar Appearance and Behavior Properties	130
TopicBar Visual Styles	130-132
Collapsible and Expandable Topic Pages	132
TopicBar Styles	132-133
TopicBar Animation	133
TopicPage Layout and Alignment	133
ToolTips in TopicBar	133
RadialMenu Overview	134-135
RadialMenu Appearance and Behavior	135
RadialMenu Animation	135
ToolTips in Radial Menu	135
Radius and Inner Radius Properties	135
Hiding the Radial Menu	135-136
Decreasing the Size of the Center Button	136-137
RadialMenu Tutorial	137
RadialMenu Tutorial Step 1 of 3: Adding the First Menu and Submenu Items	137-140
RadialMenu Tutorial Step 2 of 3: Adding the Second Menu and Submenu Items	140-141
RadialMenu Tutorial Step 3 of 3: Adding the Remaining RadialMenuItems	141-143
Menus and Toolbars for WinForms Samples	144-145
Menus and Toolbars for WinForms Task-Based Help	146
Menu Tasks	146
Adding a Menu Item to MainMenu	146-148
Adding an Icon to a Menu Item	148-149
Adding a Menu Item Before the Current Menu Item	149-150
Adding a Menu Item After the Current Menu Item	150
Adding a Standard Menu Item from the Link to Command Designer	150-151
Adding a Submenu	151-153
Adding Multiple SubMenus	153-157
Applying ShortCut Keys to Menus	157-158
Localizing the Text for the Shortcut Key	159
Creating a Side Caption for a Command Menu	159
Creating a Separate Click Event for a Command Object	159-160
Creating a Window List for an MDI Form	160-162
Deleting Menu Items	162-163
Displaying the Delimiter in Drop-Down Menus	163-164

Displaying ToolTips for Menus and Toolbars	164-165
Hiding Rarely Used Menu Items	165
Merging Menu Items	165-166
Modifying the Appearance of the Menus	166-167
Setting the Width of the Image/Checkmarks Bar	167
Showing a Dialog Form when a Message Filter is not Installed	167
Wrapping Items at the End of the Menu	167-168
ToolBar Tasks	168
Adding an Arbitrary Control to the Toolbar	168
Adding an Image to the Toolbar Button	168-169
Adding Separators Between the Buttons	169
Changing the Position of the Toolbar from Horizontal to Vertical	169
Creating a Toolbar	169-173
Docking a Toolbar Programmatically	173
Increasing the Image Size in the Toolbar	174
Modifying the Appearance of the Toolbar	174
Making the Toolbar Appear Like the Default Toolbar in Internet Explorer(IE)	174-177
Making the Image in the Toolbar Button Appear More Vibrant	177
Specifying a Docking/Floating Position	177-178
Turning on the Customization Feature	178
Wrapping Text in a Toolbar	178
Context Menu Tasks	178
Adding a ContextMenu to a Control	178-181
Retrieving the ContextMenu Control Attached to the TextBox	181-182
Linking the Context Menu to a NotifyIcon Control	182
Adding a ContextMenu to a DockingTab	183-184
DockingTab Tasks	184
Adding a Scrollbar to a DockingTab	184-185
Closing a DockingTabPage	185
Determining if the DockingTab is Floating	185
Displaying Multiple Tab Rows	186
Displaying the Same Set of Controls on each DockingTabPage	186
Enabling DockingTab Docking and Floating	186-187
Enabling or Disabling Focus Cues	187-188
Loading and Saving the Layout of the DockingTab	188-189
Moving Tab Pages at Run Time	189

Pinning the DockingTab	189
Preventing the Tabs from Receiving Focus on Mouse Click	189
Restricting the Usage of Specific Tabs	189-190
NavBar Tasks	190
Adding a Panel	190-192
Creating a Panel Header	192-193
Using Vertical Text for Collapsed Panels	193
OutBar Tasks	193-194
Customizing the Titles of OutPages	194-195
Creating and Configuring the OutBar Control	195-196
Adding a OutPage to the OutBar	196-197
Adding Multiple OutPages to OutBar	197
Modifying the Appearance of the OutBar	197-199
TopicBar Tasks	199
Adding and Removing TopicBar Items	199
Adding Topic Pages to the TopicBar	199-203
Removing a Topic Page from the TopicBar	203-205
Adding Topic Links to Topic Pages	205-207
Removing Topic Links from Topic Pages	207-209
Customizing the Appearance	209
Adding a Background Image	210-211
Adding an Icon to a Topic Page	211-212
Adding an Icon to a Topic Link	212-213
Changing the Visual Style	213-214
Customizing the Expand/Collapse Behaviors	214
Changing the Expand / Collapse Animation	214-216
Creating a Collapsed Topic Page	216-217
Using Topic Bar ToolTips	217-218
Adding ToolTips to Topic Pages	218-220
Adding ToolTips to Link Pages	220-222
Disabling ToolTips	222-223

Menus and Toolbars for WinForms Overview

Create versatile menus and docking/floating toolbars for your Windows Forms applications with **Menus and Toolbars for WinForms**. The suite includes nine user-interface and navigational tools. SmartDesigner technology makes it easy for you to build and style your menus and toolbars. This unique visual development feature allows you to work right on the design surface - no coding is required. And with the latest built-in Microsoft Office 2010 visual styles, you can achieve an Office 2010 look and feel by setting just one property. **Menus and Toolbars for WinForms** provides extensive design-time support, enabling you to create fully functional menus and toolbars in little time.

Menus and Toolbars for WinForms contains a variety of projects, and you can learn more about the individual products by clicking on the following links:

- [Menus and Toolbars Overview](#)
- [DockingTab Overview](#)
- [NavBar Overview](#)
- [OutBar Overview](#)
- [TopicBar Overview](#)
- [RadialMenu Overview](#)

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](#).

Key Features

The **Menus and Toolbars for WinForms** suite contains five products: **Menus and Toolbars for WinForms**, **DockingTab for WinForms**, **NavBar for WinForms**, **OutBar for WinForms**, and **TopicBar for WinForms**. This section highlights the key features of each product.

Menus and Toolbars Key Features

Highlights of **Menus and Toolbars for WinForms** include:

- **Integrated C1Command Framework**

The **Menus and Toolbars** suite integrates menus and toolbars into a single system, allowing you to reuse the same objects and code for menu items and toolbar buttons. The same command item (text, image, event handling code) can be used in several menus and/or toolbars at the same time.

- **Docking/Floating Toolbars**

With docking/floating behavior you can create a layout that can be easily made end-user customizable. To make a toolbar that end-users are able to move around the form, dock to its sides, or make it a floating toolbar, simply put your [C1ToolBar](#) controls inside [C1CommandDock](#) containers.

- **No-code Design Experience**

Menus and Toolbars features extensive design-time support, including **SmartDesigner** technology. The context-sensitive floating toolbars are activated with a single mouse click, where you can make changes right on the design surface. The C1Command designers even come with over 50 common commands with text, icons, and shortcut keys already configured for you.

- **Built-in Visual Styles**

All menu and toolbar controls support visual styles that mimic the styles available in Microsoft Office 2010 including Blue, Silver, and Black. Also choose from Office 2003, 2007 and Windows XP styles.

- **Global Key Shortcuts**

You can use shortcut and mnemonic keys to access a menu by the keyboard instead of the mouse. The shortcut keys are also usable for commands not present in any menus or toolbars. You can also localize the text for the shortcut keys.

- **Support for MDI Applications**

Special support for MDI (Multiple Document Interface) applications, including built-in MDI child windows list and hierarchical shortcut key processing. In MDI applications, you can restrict the amount of menu items the list displays, show hidden MDI windows in the menu's list, and merge the menu items.

- **Merging Menus**

With **Menus and Toolbars**, you can easily enable the merging of MDI child windows with MDI parent menus and toolbars. You can also specify the type of behavior for the merge and choose whether to add, replace, remove, or merge menu items, as well as specify the merging order of the menu items or toolbar buttons.

- **Multi-level Menus**

Create a hierarchy of commands or options by adding submenus to the main menu.

- **Embedded Controls**

You can easily embed arbitrary controls such as text boxes, radio buttons, and check boxes in the menus and toolbars.

- **Feature-rich Context Menus**

Easily add a context menu to any arbitrary control on your form with the [C1ContextMenu](#) control. [C1ContextMenu](#) shares the same features set as [C1MainMenu](#).

DockingTab Key Features

Highlights of **DockingTab for WinForms** include:

- **Docking and Floating Tabs**

DockingTab for WinForms provides docking and floating behavior, where the whole control or individual pages (tabs) can be torn off and automatically docked to one of the other sides of the form, to another [C1DockingTab](#) control, or floated in a separate tool window.

- **Visual Studio-like Docking**

[C1DockingTab](#) supports the common docking diamond interface made popular from Visual Studio 2005. Just set the `DockingStyle` property of the [C1CommandDock](#) control to achieve this behavior and use whole form docking layout support.

- **Auto-hiding**

[C1DockingTab](#) supports auto-hiding mode when placed inside a [C1CommandDock](#) control. This means tabs can be minimized to any edge of the container and slide in/out when the user clicks on them. To implement this behavior just set the `AutoHiding` property to `True`. To allow users to turn this on themselves set the `CanAutoHide` property.

- **Design-time Editors for Quick Styling**

With the easy-to-use editors you can quickly edit the tab appearance such as the tab style, size, and layout.

- **Tab Behaviors**

You have control over the behavioral properties for docking, floating, closing and reordering tab pages. Give end-users the ability to close and move tabs with the `CanCloseTabs` and `CanMoveTabs` properties.

- **Tab List**

Show all available tabs in a drop-down list so users can quickly navigate. Just set the `ShowTabList` property to `True`.

- **More Alignment Options** Select from various tab alignment options:

- Align tabs along the Top, Bottom, Left, or Right of the control
- Position tabs Near, Far, or Center relative to the alignment
- Stretch and squeeze tabs to fit the available space, or turn on multiline or scrolling to handle many more tabs

- **Hide Tabs**

Simply hide the tabs of [C1DockingTab](#) to create multi-page forms such as Wizards. The benefit of using [C1DockingTab](#) is that you get full, drag-and-drop design-time support for each tab page, so you can easily have multiple forms contained in one.

- **More Tab Styles**

Add more style to your tabbed interface with [C1DockingTab](#). Choose from 12 visual styles including all Office 2010, 2007 and 2003 designs. You can also set the `TabStyle` property to get sloping or rounded tabs.

NavBar Key Features

Highlights of **NavBar for WinForms** include:

- **Microsoft Outlook-style UI**

Group content and navigation menus into distinct categories just like the navigation system used in Microsoft Outlook. This model helps you organize content and allows end-users to navigate quickly.

- **Built-in Visual Styles**
Choose from 8 built-in visual styles including Office 2007, 2010, and 2003 styles. These are all a property setting away.
- **Collapsible**
Enable the [C1NavBar](#) to collapse on any edge of its container, including left, right, top and bottom.
- **Embed Controls**
Place any arbitrary controls inside each nav panel. This can be done at design-time by simply dragging controls into the panels.
- **Run-time Customization**
The [C1NavBar](#) buttons can be customized at run-time by clicking on the lower drop-down arrow button. Users can rearrange buttons and choose to show or hide certain buttons.
- **Stackable Buttons**
Use the splitter bar to stack buttons into the strip bar located at the bottom of the navigation bar. When buttons overflow they will appear in the drop-down menu for selection.
- **Close Button**
Allow end-users to close a single panel or the entire navigation bar with a single property.
- **Extensive Design-time Support**
NavBar features extensive design-time support, including **SmartDesigner** technology. The context-sensitive floating toolbars are activated with a single mouse click, where you can make changes right on the design surface. This no-code design experience provides hassle-free customization of the entire [C1NavBar](#) control.

OutBar Key Features

Highlights of **OutBar for WinForms** include:

- **Accordion-like Animation**
Expand and collapse each page like a traditional accordion control. Only one page can be expanded at one time, thus saving on screen real estate.
- **Embed Arbitrary Controls**
Arbitrary controls such as TextBoxes and charts can be embedded in any pages.
- **Built-in Visual Styles**
Select from 12 built-in visual styles including all of the Microsoft Office color schemes.
- **Hide Pages**
Easily turn any page invisible with its [PageVisible](#) property.
- **Smart Scrolling**
Use [C1OutBar](#) with [C1ToolBar](#) to provide scrolling through a long list of toolbar commands. [C1OutBar](#) can display smart scroll buttons above and below the selected page.

TopicBar Key Features

Highlights of **TopicBar for WinForms** include:

- **Create Collapsible/Expandable Lists**
Generate a highly navigational form with collapsible/expandable lists using the [C1TopicBar](#) control. This powerful control also offers the ability for you to use animation, such as smooth transitions, when collapsing/expanding pages.
- **Built-in Visual Style**
[C1TopicBar](#) provides 12 built-in visual styles such as the Office 2010 color schemes and a Windows XP look and feel.
- **No Code Design Experience**
TopicBar features extensive design-time support, including **SmartDesigner** technology. The context-sensitive

floating toolbars are activated with a single mouse click, where you can make changes right on the design surface.

- **Tooltips**

The topic bar has a **TooltipText** property for you to create a user-friendly application. For example, you can add tooltips to each topic to provide more information to the user about that group.

- **Right-to-Left Support**

[C1TopicBar](#) supports RTL (or "right-to-left") layout for Arabic and Hebrew cultures. Just set the **RightToLeft** property to **Yes**.

RadialMenu Key Features

Highlights of **RadialMenu for WinForms** include:

- **Powerful Command Framework:** RadialMenu for WinForms leverages the power and versatility of C1Command to provide the best possible performance and flexibility.
- **Complete Customization:** Utilize the power of Themes for WinForms to change the appearance of the RadialMenu. Also dig and customize text, add images and much more.
- **Touch Friendly:** RadialMenu for WinForms provides a customizable interface that is easily accessible from touch and non-touch devices alike.

Overview

The **Menus and Toolbars for WinForms** suite of components allow you to add nice-looking menus and docking/floating toolbars to your Windows Forms applications. Useful features of the suite go beyond just looks though. Some highlights include:

- Microsoft Visual Studio/Office XP and Office 2007/2010 look and feel.
- Close integration between menus and toolbars; the same command item (text, image, event handling code) can be used in several menus and/or toolbars at the same time.
- Full design-time support integrated into the Visual Studio forms designer.
- Docking/floating behavior, with layout that can be easily made end-user customizable.
- Global key shortcuts, usable also for commands not present in any menus or toolbars.
- Special support for MDI applications, including built-in MDI child windows list and hierarchical shortcut key processing.
- Idle-time automatic update of menu/toolbar items' state.
- Context menus that can be attached to arbitrary controls on the form.
- And more.

Terminology

In this text, the following words are used to designate specific classes or groups of classes of components provided by the **Menus and Toolbars for .NET** product:

- **C1Command** (two meanings): used as a short name to designate the whole **Menus and Toolbars for .NET** product, and used to designate the key class in the product.
- **Command**: used to designate objects of the type [C1Command](#) and of derived types, which represent the actual executable commands invoked from menu items and toolbar buttons.
- **Command link**: used to designate objects of the type [C1CommandLink](#).
- **Command holder**: used to designate objects of the type [C1CommandHolder](#).

Class Hierarchy

This section summarizes the class relationships between the more important of the components included in the [C1Command](#) suite.

Commands

C1Command contains the following types of commands used to create menus and toolbars:

- [C1Command](#) : System.ComponentModel.Component
Basic executable command. Base class for all other command classes.
- [C1CommandMenu](#) : C1Command
Command which can contain a submenu.
- [C1ContextMenu](#) : C1CommandMenu
Command with a submenu that can be attached to another control as a context menu.
- [C1CommandMdiList](#) : C1CommandMenu
Command which at run time expands to the list of existing MDI child windows.
- [C1CommandControl](#) : C1CommandMenu
Command which can be associated with an arbitrary control. The control is displayed inside the command link connected to the command (this command allows at most one command link to be connected to it).

Other Components and Controls

The following components and controls are included in C1Command:

- **C1CommandHolder** : System.ComponentModel.Component
Container for all commands on a form. Only one object per form is allowed, and is always automatically created when a menu or toolbar is added to the form.
- **C1MainMenu** : System.Windows.Forms.Control
Control representing the main menu of a form.
- **C1ToolBar** : System.Windows.Forms.Control
Control representing a toolbar. Must be put in a **C1CommandDock** to have docking and floating behavior.
- **C1CommandDock** : System.Windows.Forms.Panel
A container providing docking and floating behavior to toolbars.
- **C1DockingTab** : System.Windows.Forms.Control
A tab control which manages a related set of pages.
- **C1OutBar** : System.Windows.Forms.Control
An outbar control which manages a related set of pages. It provides an Outlook-style tab container.
- **C1NavBar**: System.Windows.Forms.Control
The **C1NavBar** is used for grouping information into distinct categories to help organize and navigate information quickly. It consists of a number of categories represented by preset buttons.
- **C1TopicBar**: System.Windows.Forms.Control
C1TopicBar represents a topic bar. In **C1OutBar** the control provides a collection of single pages organized into one group whereas **C1TopicBar** contains a collection of pages organized into various groups.

Design-Time Support

[C1Command](#) provides visual editing to make it easier to create menus, toolbars, outbars, and docking tabs with the Microsoft Visual Studio look and feel.

You can make changes to the [C1Command](#) controls by using one or more of the following visual editors:

In-Place Editing

You can quickly edit menu and toolbar items' text using the in-place editing feature. For more information about using the in-place editing feature, see [In-Place Text Editing](#).

Invoking the Smart Tags

You can easily set common properties for the [C1Command](#) controls using its smart tags. For more information about the smart tags in [C1Command](#), see [C1Command Smart Tags](#).

Invoking the Context Menus

You can easily configure any of the [C1Command](#) components at design time by using its associated context menu. For more information on [C1Command](#) context menus, see the [C1Command Context Menus](#).

Invoking the C1Command Editor

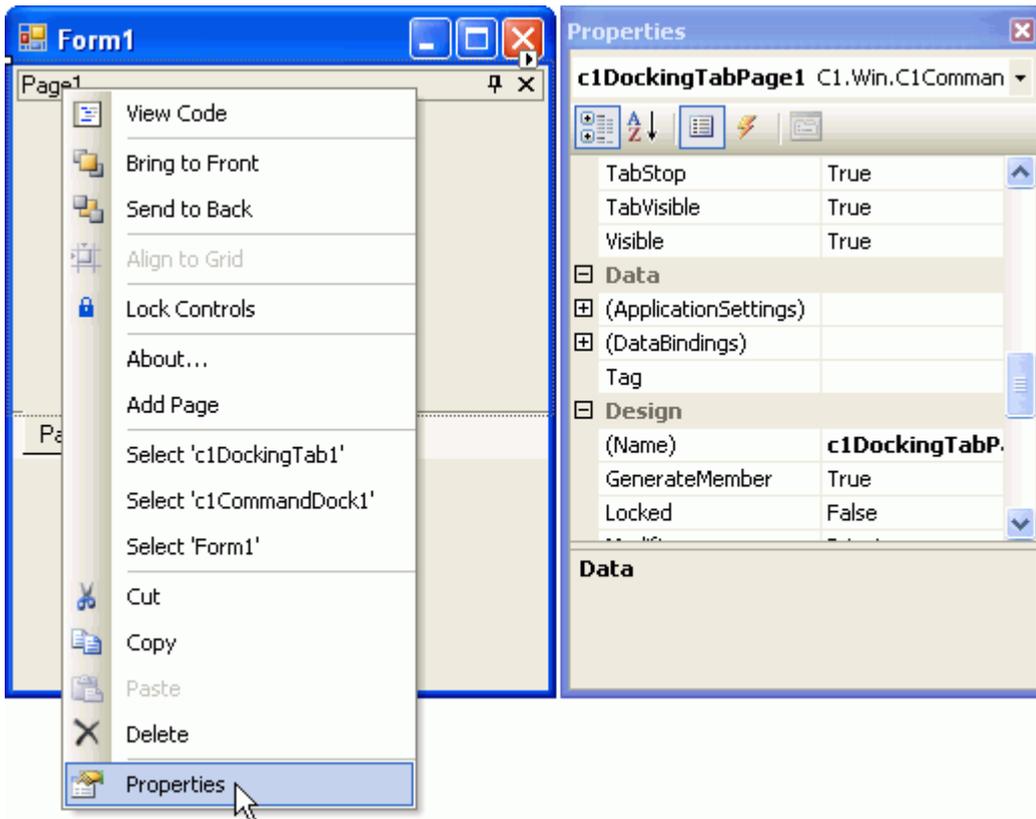
You can easily edit [C1CommandLinks](#) at design-time through its **Edit designer**.

Design Time Editors

[C1Command](#) provides seven collection editors: **C1Command Collection Editor**, **C1CommandLink Collection Editor**, **C1DockingTabPage Collection Editor**, **C1NavBarButton Collection Editor**, **C1OutPage Collection Editor**, **C1TopicPage Collection Editor**, and **C1TopicLink Collection Editor**. The main part for each of the editor's application consists of a windows form which conveniently allows the user to edit the [C1MainMenu](#), [C1ToolBar](#), [C1DockingTab](#), [C1NavBar](#), [C1OutBar](#), or [C1TopicBar](#) controls.

Showing the C1Command Control's Properties

You can access the properties for any of [C1Commands](#) components simply by right-clicking on the control and selecting **Properties** or by selecting the class from the drop-down box of the Properties window.



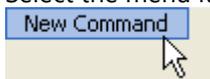
The following section details each type of support available in [C1Command](#).

In-Place Text Editing

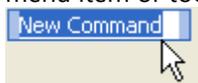
Rather than going through the menu's or toolbar's **Edit** designer or **Properties** window to edit their text you can simply edit their text directly on their control.

To use this feature, perform the following:

1. Select the menu item or toolbar item so it's highlighted.



2. Press ENTER.
menu item or toolbar item's text is ready to be edited.



3. Press ENTER again to accept the change or press ESC to cancel the change.

Command Smart Tags

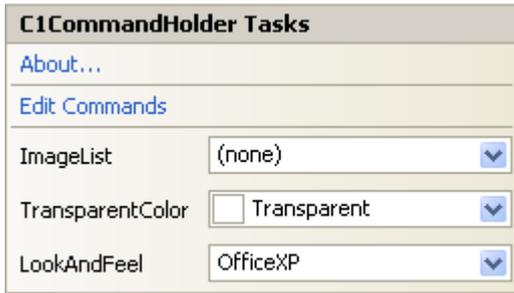
In Visual Studio, each component/command in [C1Command](#) includes a smart tag. A smart tag represents a short-cut tasks menu that provides the most commonly used properties in each component/command.

The following section introduces each smart tag for [C1Commands](#) components/commands.

C1CommandHolder Smart Tag

The [C1CommandHolder](#) component provides quick and easy access to common properties through its smart tag.

To access the **C1CommandHolder Tasks** menu, click on the smart tag (📌) in the upper right corner of the [C1CommandHolder](#) control. This will open the **C1CommandHolder Tasks** menu.



The **C1CommandHolder Tasks** menu operates as follows:

VisualStyle

Selecting the **VisualStyle** drop-down box opens a list box with several items to choose from (**Custom**, **System**, **Office2010Blue**, **Office2010Black**, **Office2010Silver**, **Office2007Blue**, **Office2007Black**, **Office2007Silver**, **Office2003Blue**, **Office2003Olive**, **Office2003Silver**, **OfficeXP**, **Classic**, and **WindowsXP**) to set the look and feel style of the commands. Note that the [C1CommandHolder.VisualStyle](#) property and [VisualStyle](#) enumeration superseded the **LookAndFeel** property and **LookAndFeelEnum** enumeration, which are now obsolete.

ImageList

Selecting the **ImageList** drop-down button opens a list box that contains an imagelist item if an [ImageList](#) component is present.

TransparentColor

Selecting the **TransparentColor** drop-down box opens the list box of Custom, Web, and System colors to choose from.

Edit Commands

Clicking **Edit Commands** opens the **C1Command Collection Editor**. For more information on how to use the **C1Command Collection Editor**, see [C1Command Collection Editor](#).

About

Clicking on the **About** item displays the **About Command** dialog box, which is helpful in finding the version number of [C1Command](#) and online resources.

C1MainMenu Smart Tag

The [C1MainMenu](#) control provides quick and easy access to common properties through its smart tag.

To access the **C1MainMenu Tasks** menu, click on the smart tag (📌) in the upper right corner of the [C1MainMenu](#)

control. This will open the **C1MainMenu Tasks** menu.



The **C1MainMenu Tasks** menu operates as follows:

VisualStyle

Selecting the **VisualStyle** drop-down box opens a list box with several items to choose from (**Custom**, **System**, **Office2010Blue**, **Office2010Black**, **Office2010Silver**, **Office2007Blue**, **Office2007Black**, **Office2007Silver**, **Office2003Blue**, **Office2003Olive**, **Office2003Silver**, **OfficeXP**, **Classic**, and **WindowsXP**) to set the look and feel style of the **C1MainMenu** control. Note that the **C1MainMenu.VisualStyle** property and **VisualStyle** enumeration supersede the **LookAndFeel** property and **LookAndFeelEnum** enumeration, which are now obsolete.

Enable Smart Designer

Clicking on the **Smart Designer** check box enables the Smart Designer of the **C1MainMenu** control.

About

Clicking on the **About** item displays the **About Command** dialog box, which is helpful in finding the version number of **C1Command** and online resources.

C1ContextMenu Smart Tag

The **C1ContextMenu** component provides quick and easy access to add menu items and edit them through its **Link to Command** designer. This is available through its smart tag.

To access the **C1ContextMenu Tasks** menu, click on the smart tag (🔗) in the upper right corner of the **C1ContextMenu** component. This will open the **C1ContextMenu Tasks** menu.



The **C1ContextMenu Tasks** menu operates as follows:

Add Item

Clicking on the **Add Item** adds a new command after the current command and opens the **Link to Command** designer.

For more information how to use the **Link to Command** designer, see [Link to Command designer](#).

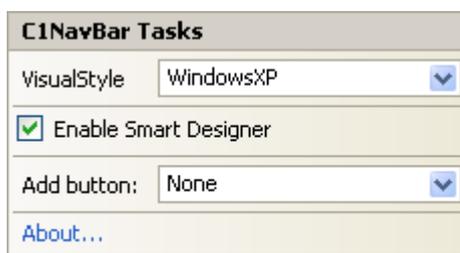
Edit Menu

Clicking on the **Edit Menu** item displays the command items in the context menu that is attached to a specific control such as [C1MainMenu](#) or [C1ToolBar](#).

C1NavBar Smart Tag

The [C1NavBar](#) control provides quick and easy access to common properties and its most common editing actions such as adding buttons through its smart tag.

To access the **C1NavBar Tasks** menu, click on the smart tag (🔗) in the upper right corner of the [C1NavBar](#) control. This will open the **C1NavBar Tasks** menu.



The **C1NavBar Tasks** menu operates as follows:

VisualStyle

Selecting the **VisualStyle** drop-down box opens a list box with several items to choose from (**Custom**, **System**, **Office2010Blue**, **Office2010Black**, **Office2010Silver**, **Office2007Blue**, **Office2007Black**, **Office2007Silver**, **Office2003Blue**, **Office2003Olive**, **Office2003Silver**, **OfficeXP**, **Classic**, and **WindowsXP**) to set the look and feel style of the [C1NavBar](#) control. Note that the [C1NavBar.VisualStyle](#) property and [VisualStyle](#) enumeration supersede the **LookAndFeel** property and **LookAndFeelEnum** enumeration, which are now obsolete.

Enable Smart Designer

Clicking on the **Enable Smart Designer** check box enables the Smart Designer for the [C1NavBar](#) control.

Add button

Clicking on the **Add button** drop-down box opens a list box of pre-defined buttons to choose from such as **Custom**, **Mail**, **Calendar**, **Contacts**, **Tasks**, **Notes**, **Folder**, **Shortcut**, or **Journal**.

About

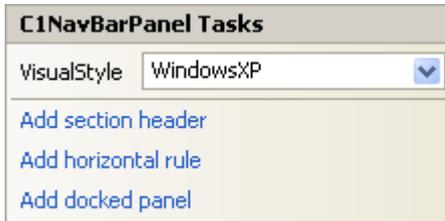
Clicking on the **About** item displays the **About Command** dialog box, which is helpful in finding the version number of [C1Command](#) and online resources.

C1NavBar Panel Smart Tag

The **C1NavBarPanel Tasks** menu makes it simple to change the [VisualStyle](#) and add section headers, horizontal rules,

and docked panels.

To access the **C1NavBarPanel Tasks** menu, click on the smart tag (📌) in the upper right corner of the **C1NavBarPanel**. This will open the **C1NavBarPanel Tasks** menu.



The **C1NavBarPanel Tasks** menu operates as follows:

VisualStyle

Selecting the **VisualStyle** drop-down box opens a list box with several items to choose from (**Custom**, **System**, **Office2010Blue**, **Office2010Black**, **Office2010Silver**, **Office2007Blue**, **Office2007Black**, **Office2007Silver**, **Office2003Blue**, **Office2003Olive**, **Office2003Silver**, **OfficeXP**, **Classic**, and **WindowsXP**) to set the look and feel style of the section headers, horizontal rules, and docked panels. Note that the **C1NavBar.VisualStyle** property and **VisualStyle** enumeration supersede the **LookAndFeel** property and **LookAndFeelEnum** enumeration, which are now obsolete.

Add Section Header

Clicking on the **Add Section Header** item adds a section header to the panel area of the selected button.

Add Horizontal Rule

Clicking on the **Add Horizontal Rule** item adds a horizontal line across the panel area of the selected button.

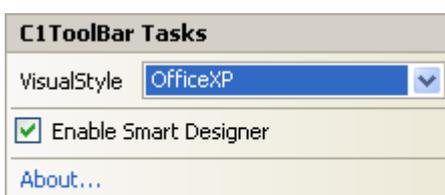
Add Docked Panel

Clicking on the **Add Docked Panel** item enables a dockable panel area of the selected button.

C1ToolBar Smart Tag

The **C1ToolBar** component provides quick and easy access to its most common properties.

To access the **C1ToolBar Tasks** menu, click on the smart tag (📌) in the upper right corner of the **C1ToolBar** control. This will open the **C1ToolBar Tasks** menu.



The **C1ToolBar Tasks** menu operates as follows:

VisualStyle

Selecting the **VisualStyle** drop-down box opens a list box with several items to choose from (**Custom**, **System**, **Office2010Blue**, **Office2010Black**, **Office2010Silver**, **Office2007Blue**, **Office2007Black**, **Office2007Silver**, **Office2003Blue**, **Office2003Olive**, **Office2003Silver**, **OfficeXP**, **Classic**, and **WindowsXP**) to set the style of the **C1ToolBar**. Note that the **C1ToolBar.VisualStyle** property and **VisualStyle** enumeration supersede the **LookAndFeel** property and **LookAndFeelEnum** enumeration, which are now obsolete.

Enable Smart Designer

Selecting the **Enable Smart Designer** check box enables the Smart Designer for the **C1ToolBar** control.

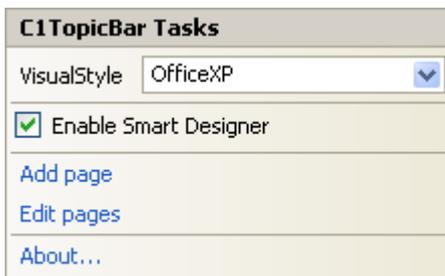
About

Clicking on the **About** item displays the **About Command** dialog box, which is helpful in finding the version number of **C1Command** and online resources.

C1TopicBar Smart Tag

The **C1TopicBar** component provides quick and easy access to its **C1TopicPage Collection Editor** and its most common editing actions such as adding topic pages through its smart tag.

To access the **C1TopicBar Tasks** menu, click on the smart tag (📌) in the upper right corner of the **C1TopicBar** control. This will open the **C1TopicBar Tasks** menu.



The **C1TopicBar Tasks** menu operates as follows:

VisualStyle

Selecting the **VisualStyle** drop-down box opens a list box with several items to choose from (**Custom**, **System**, **Office2010Blue**, **Office2010Black**, **Office2010Silver**, **Office2007Blue**, **Office2007Black**, **Office2007Silver**, **Office2003Blue**, **Office2003Olive**, **Office2003Silver**, **OfficeXP**, **Classic**, and **WindowsXP**) to set the style of the **C1TopicBar**. Note that the **C1TopicBar.VisualStyle** property and **VisualStyle** enumeration supersede the **LookAndFeel** property and **LookAndFeelEnum** enumeration, which are now obsolete.

Enable Smart Designer

Clicking on the **Enable Smart Designer** check box enables the Smart Designer for the **C1TopicBar** control.

Add Page

Clicking on the **Add Page** item, adds a new page below the current page in the [C1TopicBar](#).

Edit Pages

Clicking on the **Edit Pages** item, opens the **C1TopicPage Collection Editor**. For additional information on the **C1TopicPage Collection Editor**, see [C1TopicPage Collection Editor](#).

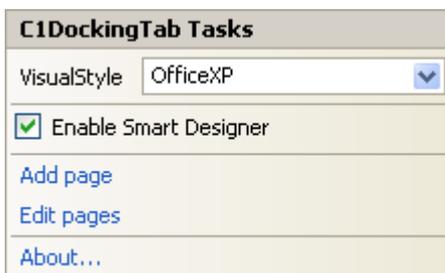
About

Clicking on the **About** item displays the **About Command** dialog box, which is helpful in finding the version number of [C1Command](#) and online resources.

C1DockingTab Smart Tag

The [C1DockingTab](#) control provides quick and easy access to its **C1DockingTabPage Collection Editor** and its most common editing actions such as adding new tab pages through its smart tag.

To access the **C1DockingTab Tasks** menu, click on the smart tag (▾) in the upper right corner of the [C1DockingTab](#) control. This will open the **C1DockingTab Tasks** menu.



The **C1DockingTab Tasks** menu operates as follows:

VisualStyle

Selecting the **VisualStyle** drop-down box opens a list box with several items to choose from (**Custom**, **System**, **Office2010Blue**, **Office2010Black**, **Office2010Silver**, **Office2007Blue**, **Office2007Black**, **Office2007Silver**, **Office2003Blue**, **Office2003Olive**, **Office2003Silver**, **OfficeXP**, **Classic**, and **WindowsXP**) to set the style of the tabs. Note that the [C1DockingTab.VisualStyle](#) property and [VisualStyle](#) enumeration supersede the **LookAndFeel** property and **LookAndFeelEnum** enumeration, which are now obsolete.

Enable Smart Designer

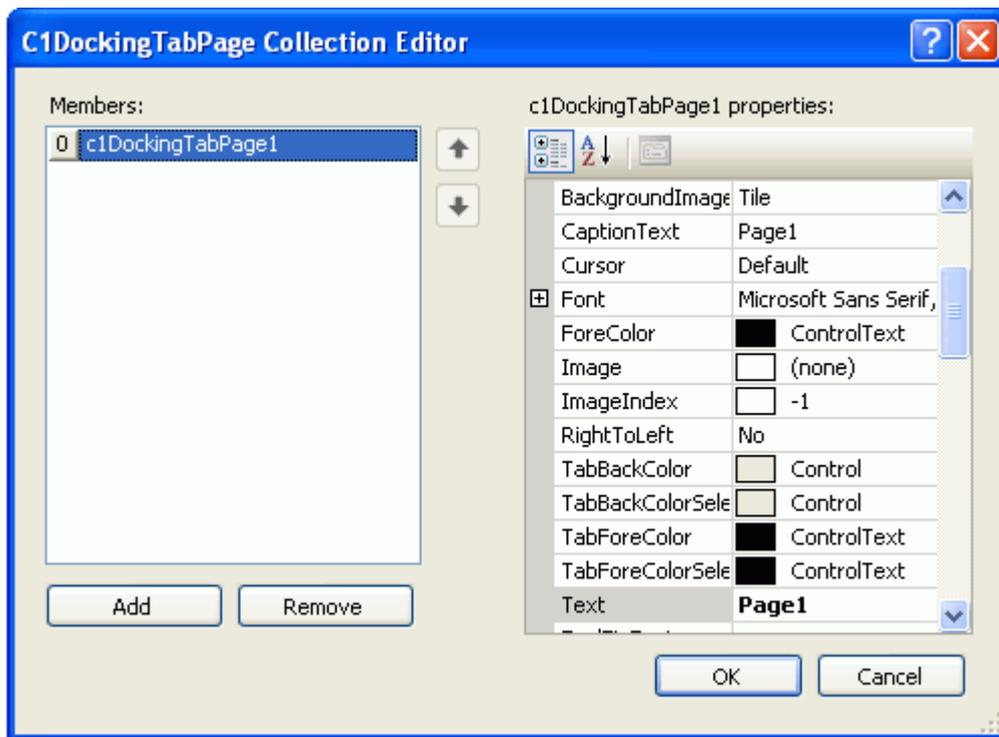
Selecting the **Enable Smart Designer** check box enables the Smart Designer of the [C1DockingTab](#) control. The default value is **True** (checked). For more information about the Smart Designer's elements see, [C1Command Smart Designer](#).

Add page

Clicking **Add page** adds a new **C1DockingTabPage**. **C1DockingTabPage** has a smart tag as well. For more information about **C1DockingTabPage Tasks** menu, see [C1DockingTabPage smart tag](#).

Edit pages

Selecting **Edit pages** opens the **C1DockingTabPage Collection Editor**.



For more information on how to use the **C1DockingTabPage Collection Editor**, see [C1DockingTabPage Collection Editor](#).

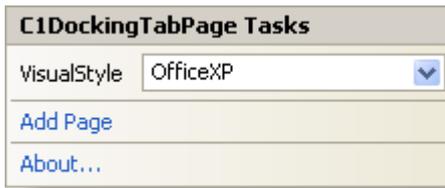
About

Clicking on the **About** item displays the **About Command** dialog box, which is helpful in finding the version number of [C1Command](#) and online resources.

C1DockingTabPage Smart Tag

The [C1DockingTabPage](#) control makes it simple to change the VisualStyle and add new tab pages through its smart tag.

To access the **C1DockingTabPage Tasks** menu, click on the smart tag (📌) in the upper right corner of the [C1DockingTabPage](#). This will open the **C1DockingTabPage Tasks** menu.



The **C1DockingTabPage Tasks** menu operates as follows:

VisualStyle

Selecting the **VisualStyle** drop-down box opens a list box with several items to choose from (**Custom**, **System**, **Office2010Blue**, **Office2010Black**, **Office2010Silver**, **Office2007Blue**, **Office2007Black**, **Office2007Silver**, **Office2003Blue**, **Office2003Olive**, **Office2003Silver**, **OfficeXP**, **Classic**, and **WindowsXP**) to set the style of the tab. Note that the **C1DockingTab.VisualStyle** property and **VisualStyle** enumeration supersede the **LookAndFeel** property and **LookAndFeelEnum** enumeration, which are now obsolete.

Add Page

Clicking **Add Page** adds a new **C1DockingTabPage**.

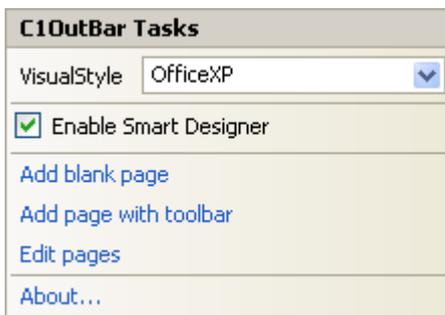
About

Clicking on the **About** item displays the **About Command** dialog box, which is helpful in finding the version number of **C1Command** and online resources.

C1OutBar Smart Tag

The **C1OutBar** control provides quick and easy access to its **C1OutPage Collection Editor** and its most common editing actions such as adding a blank page or a page with a toolbar, through its smart tag.

To access the **C1OutBar Tasks** menu, click on the smart tag (📌) in the upper right corner of the **C1OutBar** control. This will open the **C1OutBar Tasks** menu.



The **C1OutBar Tasks** menu operates as follows:

VisualStyle

Selecting the **VisualStyle** drop-down box opens a list box with several items to choose from (**Custom**, **System**, **Office2010Blue**, **Office2010Black**, **Office2010Silver**, **Office2007Blue**, **Office2007Black**, **Office2007Silver**, **Office2003Blue**, **Office2003Olive**, **Office2003Silver**, **OfficeXP**, **Classic**, and **WindowsXP**) to set the style of the

[C1OutBar](#) control. Note that the [VisualStyle](#) property and [VisualStyle](#) enumeration supersede the **LookAndFeel** property and **LookAndFeelEnum** enumeration, which are now obsolete.

Enable Smart Designer

Selecting the **Enable Smart Designer** check box enables the Smart Designer of the [C1OutBar](#) control. The default value is **True** (checked). For more information about the Smart Designer's elements see, [C1Command Smart Designer](#).

Add blank page

Clicking **Add blank page** adds a new [C1OutPage](#) below the current [C1OutPage](#). Clicking inside the new [C1OutPage](#) exposes the smart tag anchor for the **C1OutPage Tasks** menu.

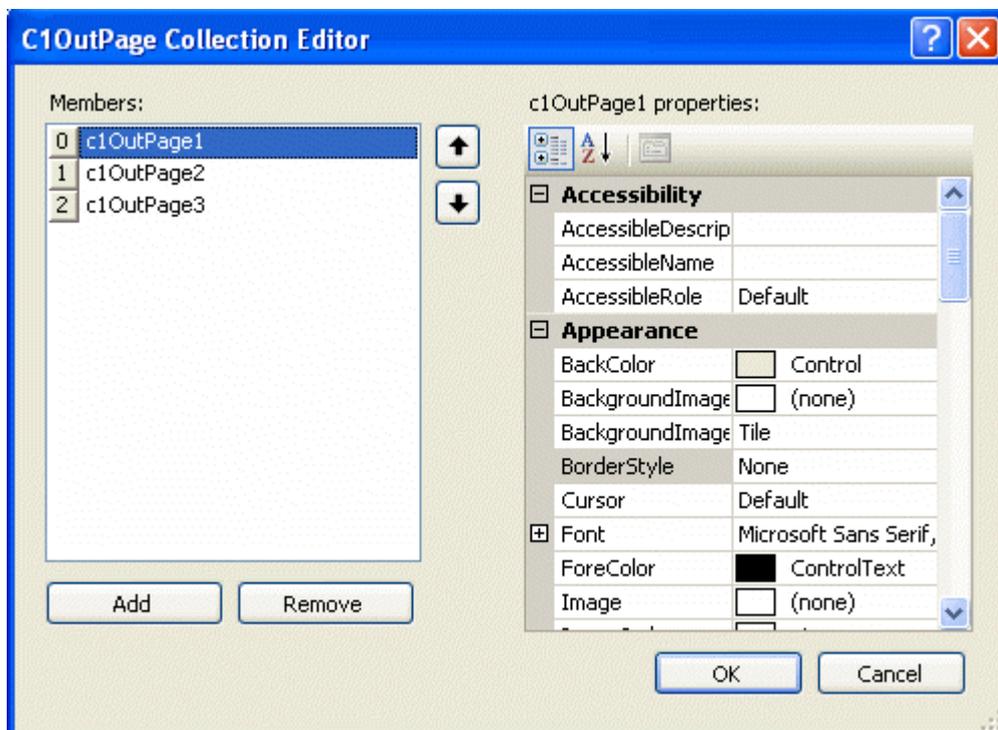
Add page with toolbar

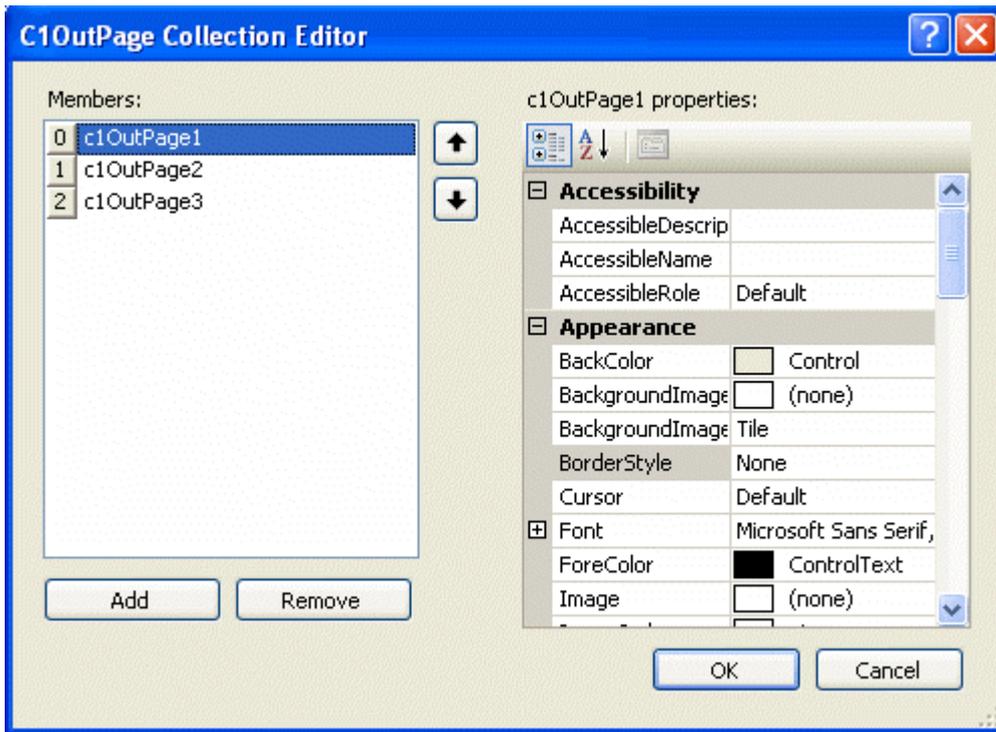
Clicking **Add page with toolbar** adds a new [C1OutPage](#) which contains a [C1ToolBar](#). A **C1OutPageTasks** menu appears for the newly added [C1OutPage](#). Clicking inside the new page exposes the smart tag anchor for the **C1ToolBar Tasks** menu.

For more information on how to use the **C1ToolBar Tasks** menu, see [C1ToolBar Smart Tag](#).

Edit pages

Selecting **Edit pages** opens the [C1OutPage Collection Editor](#).





For more information on how to use the **C1OutPage Collection Editor**, see [C1OutPage Collection Editor](#).

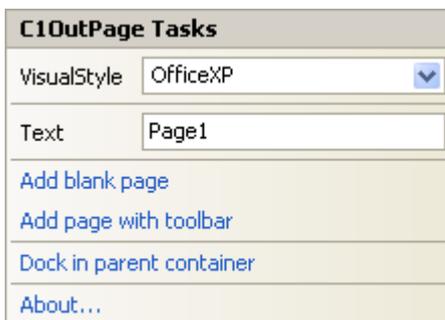
About

Clicking on the **About** item displays the **About Command** dialog box, which is helpful in finding the version number of [C1Command](#) and online resources.

C1OutPage Smart Tag

The [C1OutPage](#) control provides quick and easy access to its [C1OutPage Collection Editor](#) and its most common editing actions such as adding a new blank page or a new page with a toolbar with only one click through its smart tag.

To access the **C1OutPage Tasks** menu, select the [C1OutPage](#) and click on the smart tag (📌) in the upper right corner of the [C1OutPage](#). This will open the **C1OutPage Tasks** menu.



The **C1OutPage Tasks** menu operates as follows:

VisualStyle

Selecting the **VisualStyle** drop-down box opens a list box with several items to choose from (**Custom**, **System**, **Office2010Blue**, **Office2010Black**, **Office2010Silver**, **Office2007Blue**, **Office2007Black**, **Office2007Silver**, **Office2003Blue**, **Office2003Olive**, **Office2003Silver**, **OfficeXP**, **Classic**, and **WindowsXP**) to set the style of the **C1OutPage** control. Note that the **C1OutBar.VisualStyle** property and **VisualStyle** enumeration supersede the **LookAndFeel** property and **LookAndFeelEnum** enumeration, which are now obsolete.

Text

Selecting the **Text** box and entering a name gets the text for the **C1OutPage**.

Add blank page

Clicking **Add blank page** adds a new empty **C1OutPage** (without a **C1ToolBar** inside it) below the current **C1OutPage**.

Add page with toolbar

Clicking **Add page with toolbar** adds a new **C1OutPage** below the current **C1OutPage**. It also adds a new **C1ToolBar** inside it.

Dock in parent container

Clicking **Dock in parent container** docks the **C1OutPage** inside its parent container.

About

Clicking on the **About** item displays the **About Command** dialog box, which is helpful in finding the version number of **C1Command** and online resources.

Command Context Menus

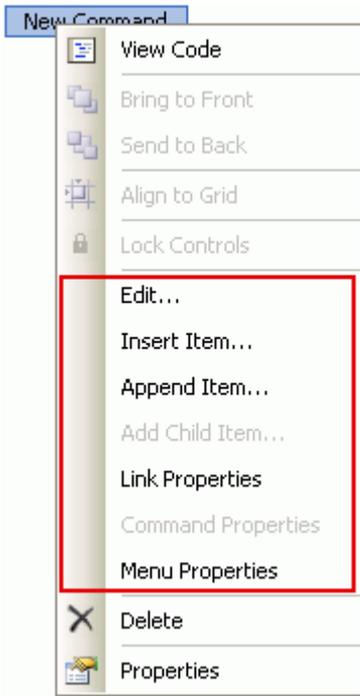
Each of **C1Commands** components provides a context menu for additional functionality to use at design time.

Command Links context menu

Right-click on the command link item for a menu item or toolbar button to open the following context menu.



Note: The only difference between the Command Links context menu for menus and the one for toolbar is the Menu Properties/Toolbar Properties command.



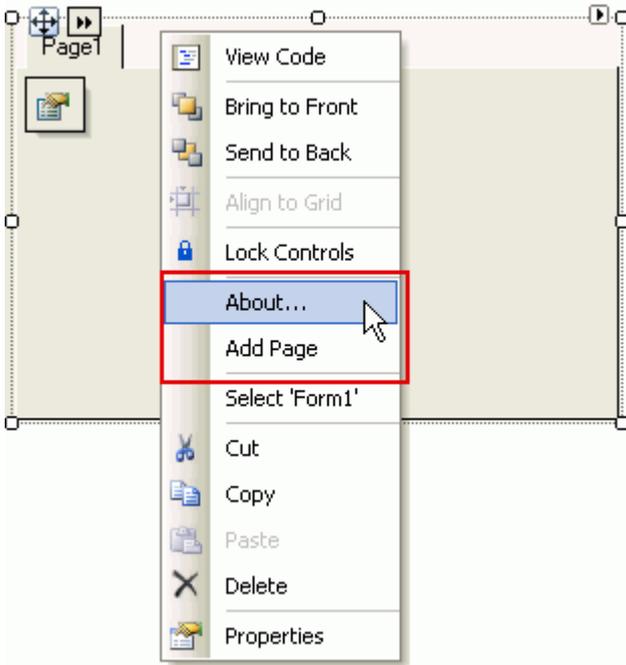
The following table provides a brief description of the custom items added by [C1MainMenu](#) and [C1ToolBar](#) controls to their context menus:

Custom Items	Description
Edit	Brings up the Link to Command designer for the currently selected command link. You can edit the currently linked command, or link to another (new or existing) command. This item is disabled when there is no currently selected command link.
Insert Item	Inserts a new command link before the current one, and opens the Link to Command designer. This item is disabled when there is no currently selected command link.
Append Item	Adds a new command link after the last one in the current menu or toolbar, and brings up the command edit designer for the new link.
Add Child Item	Adds a new command link to the menu (C1CommandMenu or C1ContextMenu) linked by the currently selected command link and it opens the Link to Command designer. This item is disabled unless the current command link is linked to a C1CommandMenu or C1ContextMenu type command.
Link Properties	Selects the command link and shows its properties in the Properties window.
Command Properties	Selects the command and shows its properties in the Properties window. Note that for non-empty links, clicking a link toggles between link's own and linked command's properties.
Menu Properties	Selects the menu and shows its properties in the Properties window.

 **Note:** All menu-editing actions are fully undoable using the Visual Studio's Undo facility.

C1DockingTab context menu

Right-click on the [C1DockingTab](#) control to open its context menu.

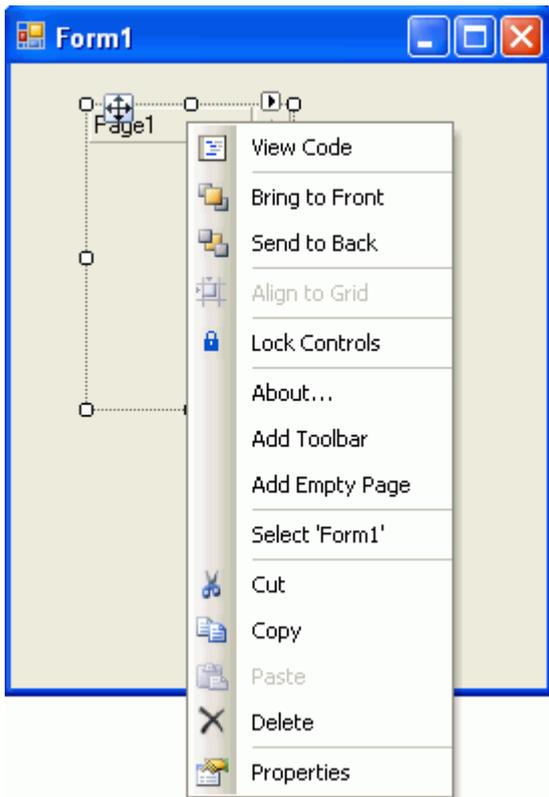


The following table provides a brief description of the custom items added by [C1DockingTab](#) control to its context menu:

Custom Items	Description
About	Opens the About Command dialog box, which is helpful in finding the version number of C1Command and online resources.
Add Page	Adds a new page to the C1DockingTab .

C1OutBar context menu

Right-click on the [C1OutBar](#) control to open its context menu.

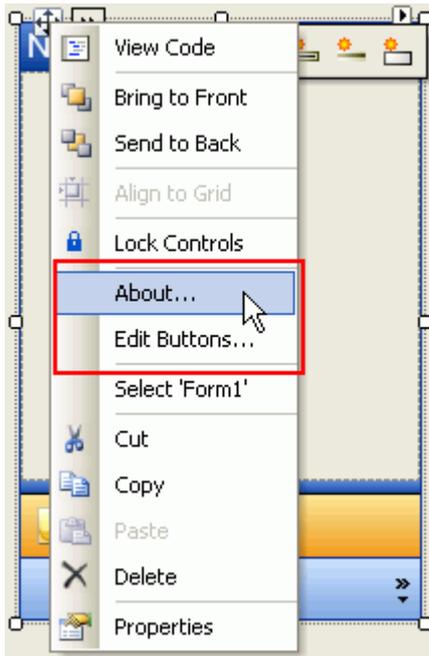


The following table provides a brief description of the custom items added by the [C1OutBar](#) control to its context menu:

Custom Items	Description
About	Opens the About Command dialog box, which is helpful in finding the version number of C1Command and online resources.
Add Toolbar	Adds a new C1OutPage with a C1ToolBar inside it.
Add Empty Page	Adds a new C1OutPage without a C1ToolBar inside it.

C1NavBar context menu

Right-click on the [C1NavBar](#) control to open its context menu.

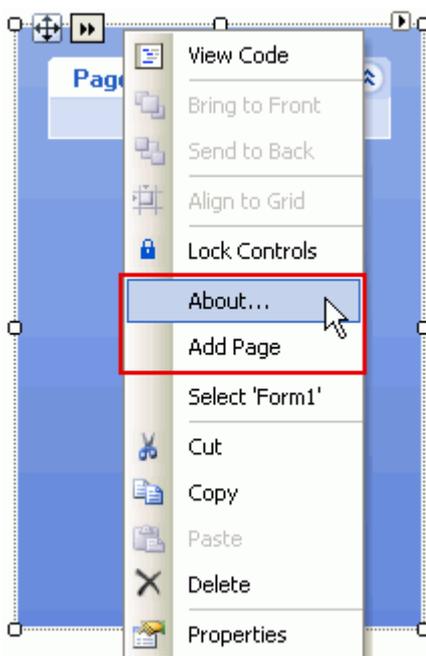


The following table provides a brief description of the custom items added by the [C1NavBar](#) control to its context menu:

Custom Items	Description
About	Opens the About Command dialog box, which is helpful in finding the version number of C1Command and online resources.
Edit Buttons	Opens the C1NavBarButton Collection Editor .

C1TopicBar context menu

Right-click on the [C1TopicBar](#) control to open its context menu.



The following table provides a brief description of the custom items added by the [C1TopicBar](#) control to its context menu:

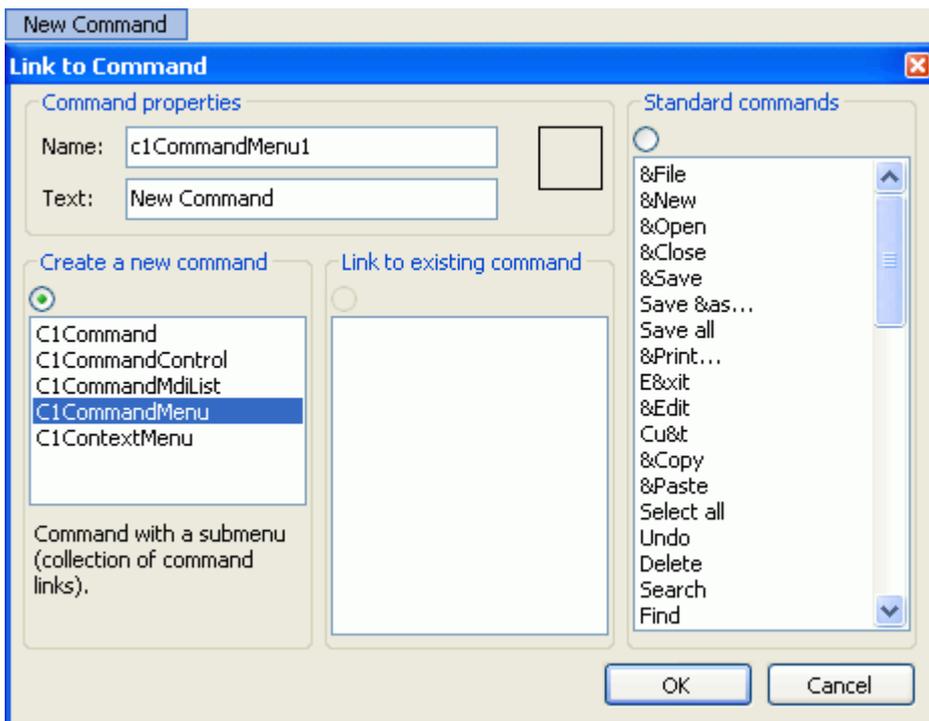
Custom Items	Description
About	Opens the About Command dialog box, which is helpful in finding the version number of C1Command and online resources.
Add Page	Adds a new C1TopicPage to the C1TopicBar control.

Link to Command Designer

[C1Command](#) has a **Link to Command** designer for the command items in the menus and toolbars controls so you can add commands to them or modify existing commands at design time.

To access the Edit designer:

Click on any command item in the menu or toolbar control twice or right-click and select **Edit** from its context menu to open the **Link to Command** designer.



This dialog box is used to create new commands, edit existing ones, or link command links to commands. The same dialog box is used to edit command links in drop-down menus and toolbars. The **Link to Command** designer contains the following categories: Command properties, Create a new Command, Link to existing command, and Standard commands.

The following table lists and describes the functionality of the items in the **Link to Command** designer:

Group Box	Description
Command properties	The Command properties group box includes a Name text box and a Text text box. The Name text box displays the default command name

	for the command or you can enter a new command name. The Text text box displays the text that appears on the command item.
Create New Command	The Create a new command group box contains a list box of various command types to choose a specific command type for the new command. The radio button enables you to toggle between creating a new command and linking to an existing one in the Link to existing command list box.
Link to Existing Command	The Link to existing command group box displays a list box of commands already created on the form. When you select an item from this list, the Create New Command radio button is automatically unselected.
Standard Commands	The Standard command group box provides 54 built-in commands and 40 images for standard commands. When you select a standard command that includes a built-in image from the Standard commands list box, the preview image appears in the square box located in the Command properties group box.

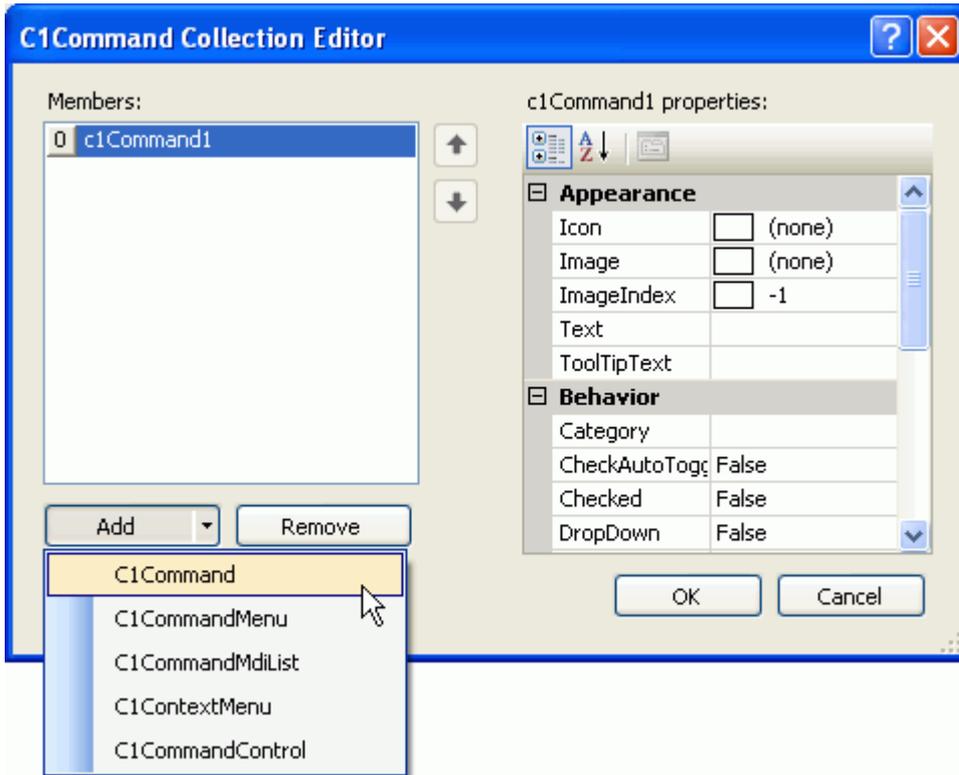
Command Collection Editors

C1Command provides seven collection editors: **C1Command Collection Editor**, **C1CommandLink Collection Editor**, **C1DockingTabPage Collection Editor**, **C1NavBarButton Collection Editor**, **C1OutPage Collection Editor**, **C1TopicPage Collection Editor**, and **C1TopicLink Collection Editor**. The main part for each of the editor's application consists of a windows form which conveniently allows the user to edit the [C1MainMenu](#), [C1ToolBar](#), [C1DockingTab](#), [C1NavBar](#), [C1OutBar](#), or [C1TopicBar](#) controls.

The following section briefly introduces the [C1Command](#) collection editors and explains how to access them.

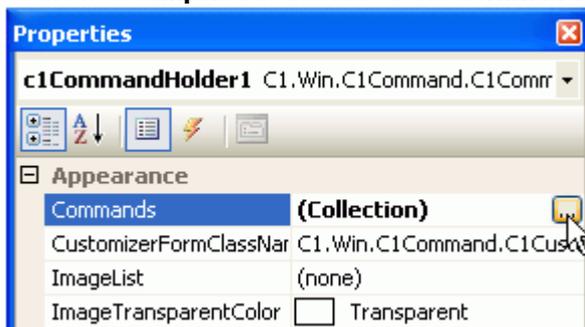
Command Collection Editor

The **C1Command Collection Editor** allows the user to edit command properties and add or remove commands of the type [C1Command](#), [C1CommandMenu](#), [C1CommandMdiList](#), [C1ContextMenu](#), and [C1CommandControl](#) in the hierarchy.



To access the C1Command Collection Editor:

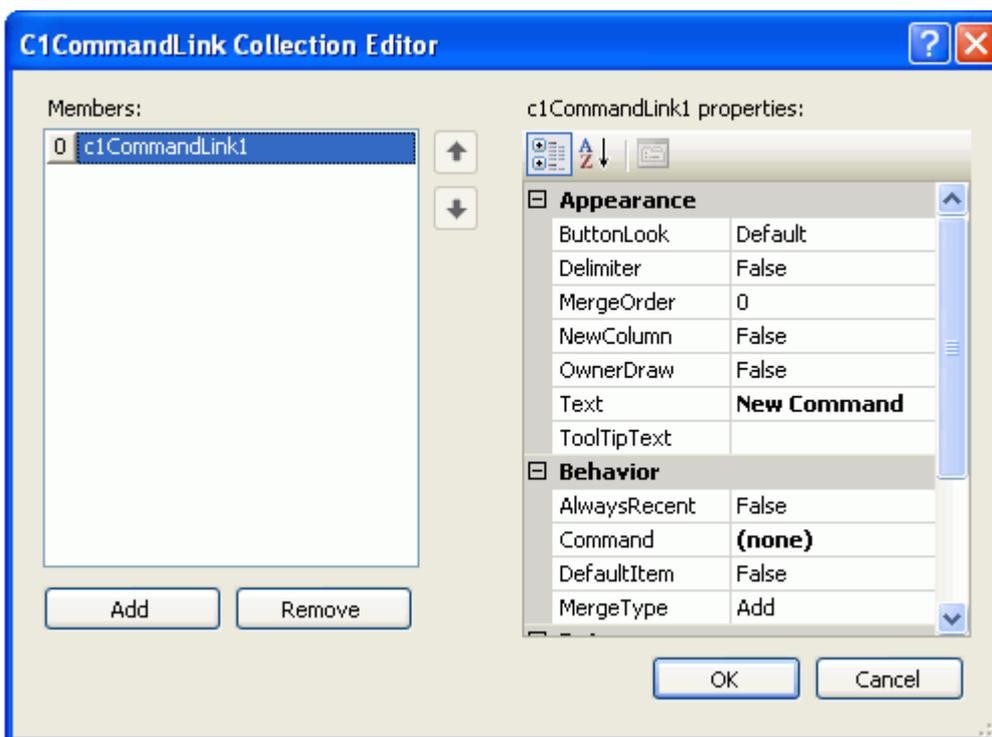
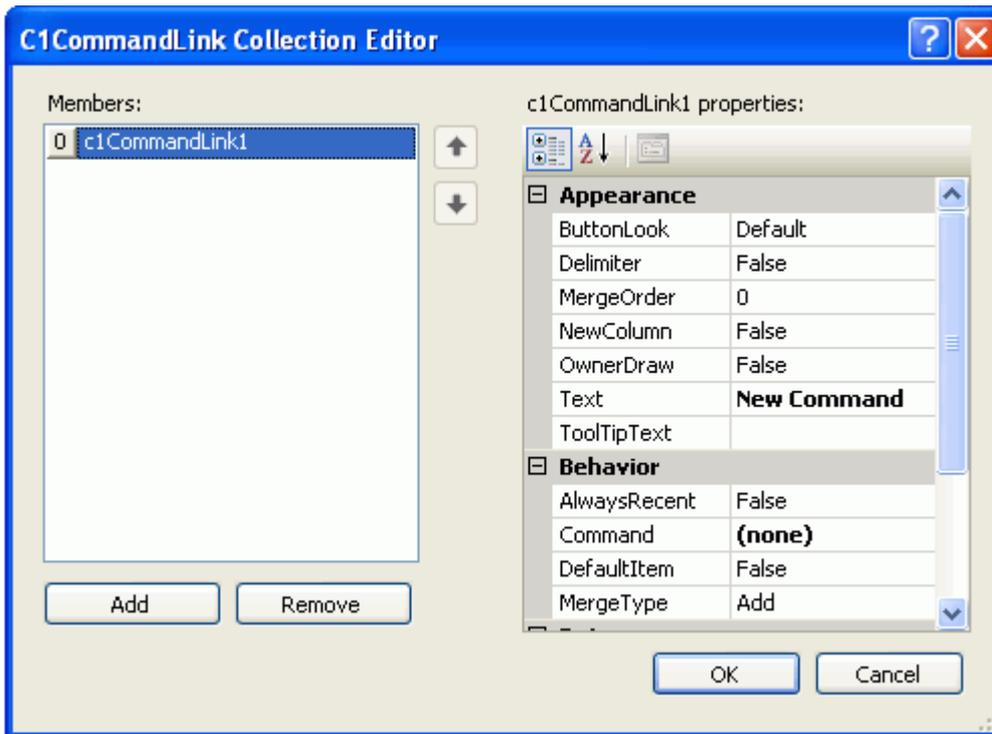
1. Drop a [C1MainMenu](#) or a [C1ToolBar](#) onto the form.
This will also automatically create a [C1CommandHolder](#) component in the form's component tray.
2. Click on the **ellipsis** button next to the **Commands** property in the [C1CommandHolder](#)'s Properties window.



The **C1Command Collection Editor** appears.

CommandLink Collection Editor

The **C1CommandLink Collection Editor** allows the user to add or remove command links as well as link existing commands to the command link.

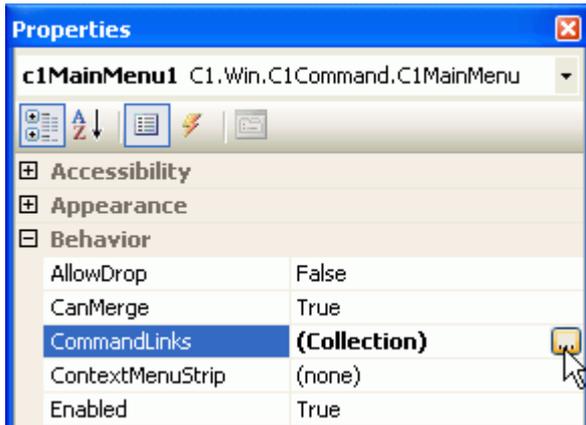


To access the C1CommandLink Collection Editor:

1. Drop a **C1MainMenu** or a **C1ToolBar** onto the form.

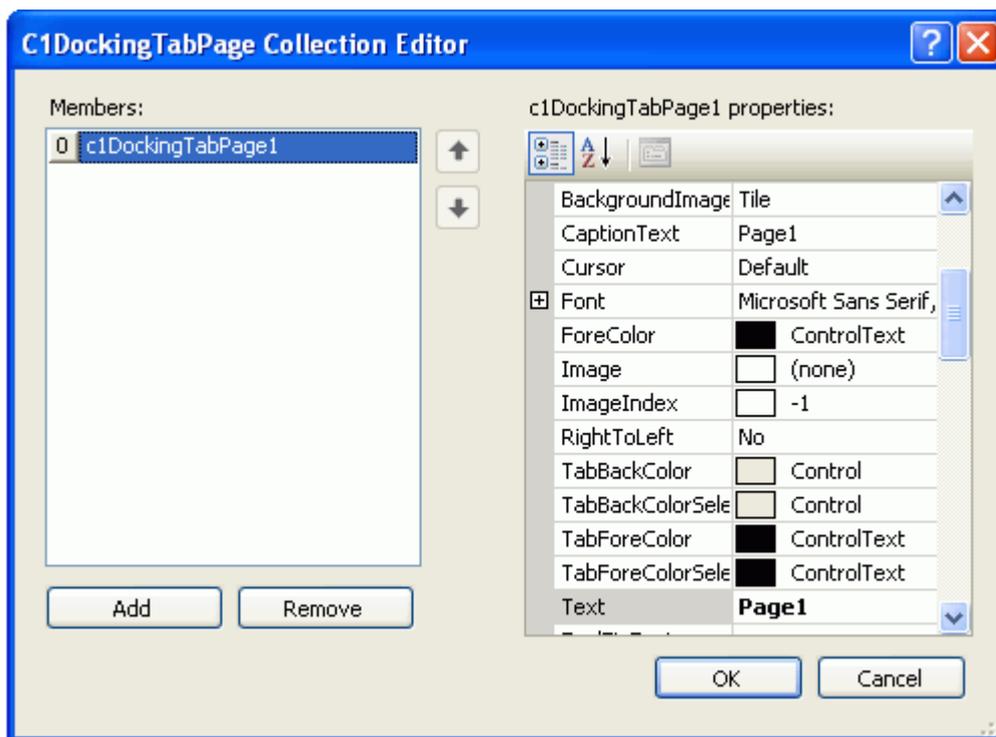
This will also automatically create a **C1CommandHolder** component in the form's component tray.

2. Click on the **ellipsis** button next to the **CommandLinks** property in **C1MainMenus** or **C1ToolBars** Properties window.



DockingTabPage Collection Editor

The **C1DockingTabPage Collection Editor** allows the user to add or remove tab pages as well as edit the properties in the tab pages.



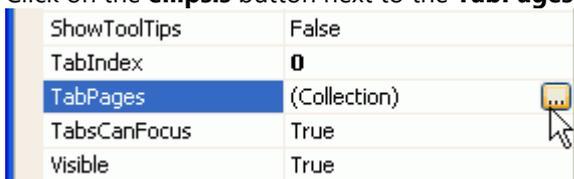
To access the C1DockingTabPage Collection Editor:

There are two ways to access the **C1DockingTabPage Collection Editor**, either through the **TabPages** property in the **Properties** window, or the **Edit tab pages** in the **C1DockingTab Tasks** menu. For more information about how to use the **C1DockingTab Tasks** menu, see [C1DockingTab Smart Tag](#).

Option 1

1. Drop a **C1DockingTab** onto the form.

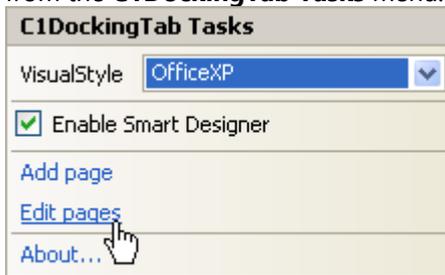
2. Click on the **ellipsis** button next to the **TabPage** property in **C1DockingTabs** Properties window.



The **C1DockingTabPage Collection Editor** appears.

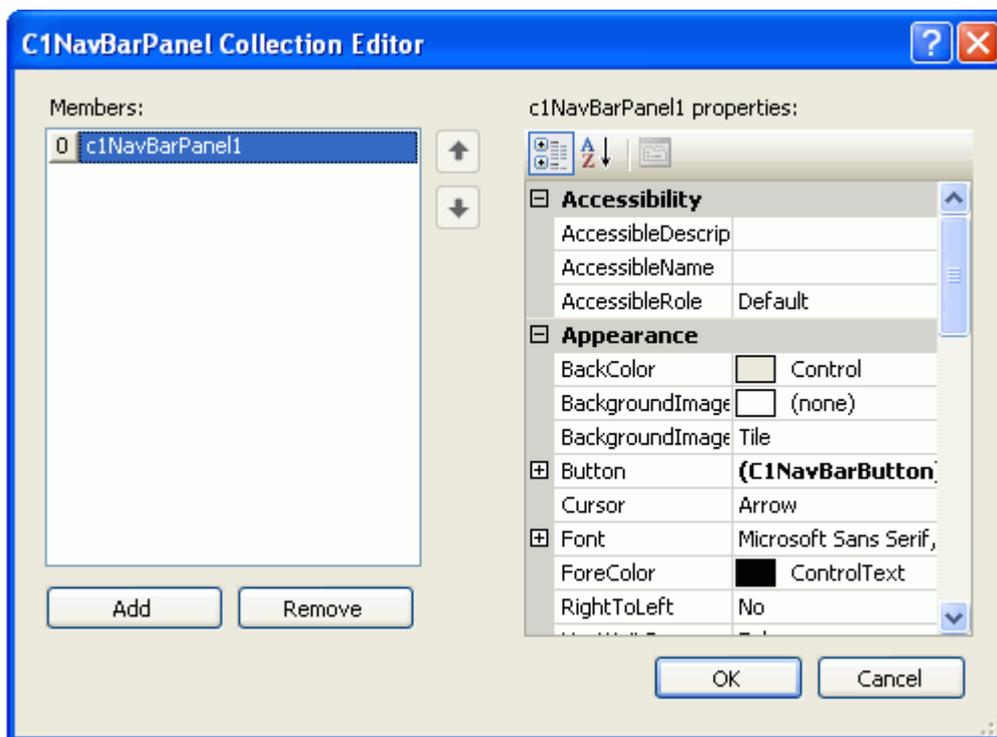
Option 2

1. Drop a **C1DockingTab** control onto the form.
2. Click on the smart tag (Ⓛ) in the upper right corner of the **C1DockingTab** control and then click on **Edit pages** from the **C1DockingTab Tasks** menu.



NavBarPanel Collection Editor

The **C1NavBarPanel Collection Editor** allows the user to add or remove panels in the **C1NavBar** control as well as edit the panels' properties.

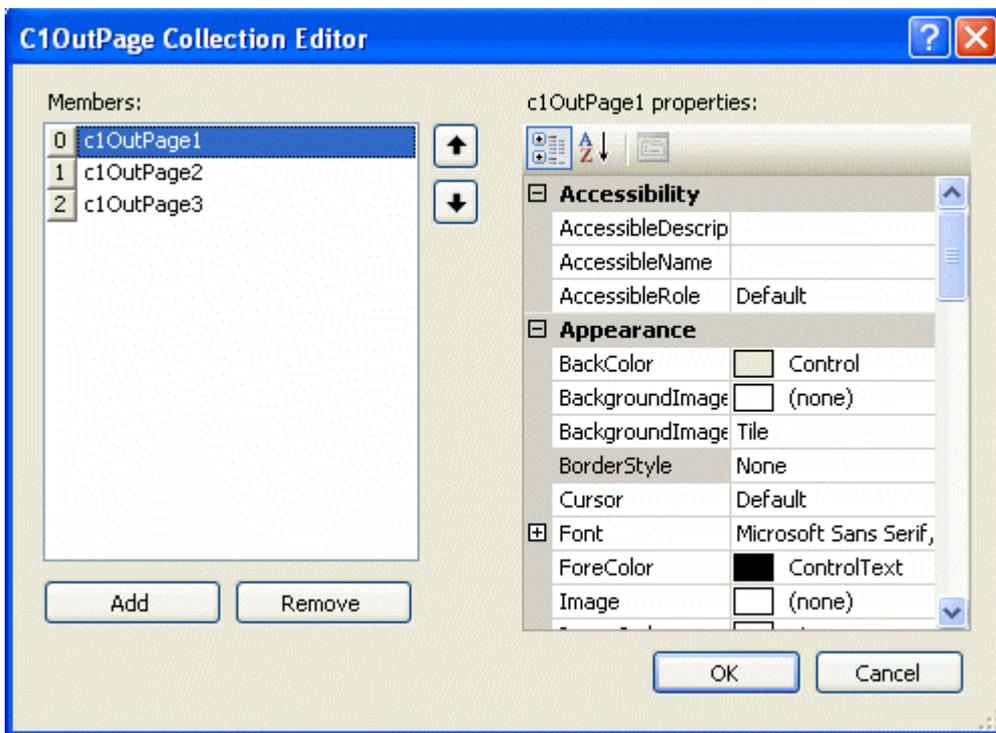


To access the C1NavBarPanel Collection Editor:

1. Drop a [C1NavBar](#) onto the form.
2. Click on the **ellipsis** button next to the **Panels** property in the [C1NavBar](#)'s Properties window. The **C1NavBarPanel Collection Editor** appears.

OutPage Collection Editor

The **C1OutPages Collection Editor** allows the user to add or remove the pages in the [C1OutBar](#) control as well as edit the pages' properties.

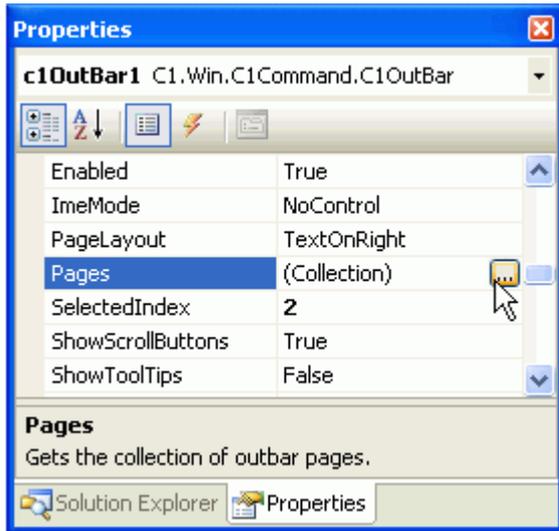


To access the C1OutPage Collection Editor:

There are two ways to access the **C1OutPage Collection Editor**, either through the **Pages** property in the **Properties** window, or the **Edit Pages** in the **C1OutBar Tasks** menu. For more information about how to use the **C1OutBar Tasks** menu, see [C1OutBar Smart Tag](#).

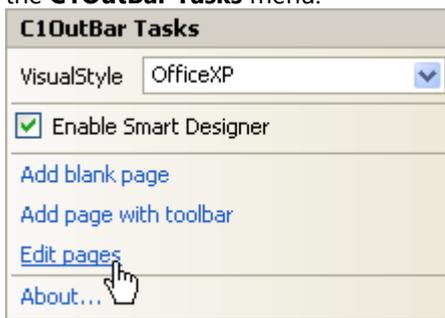
Option 1

1. Drop a [C1OutBar](#) control onto the form.
2. Click on the **ellipsis** button next to the **Pages** property in [C1OutBar](#)'s **Properties** window. The **C1OutPage Collection Editor** appears.



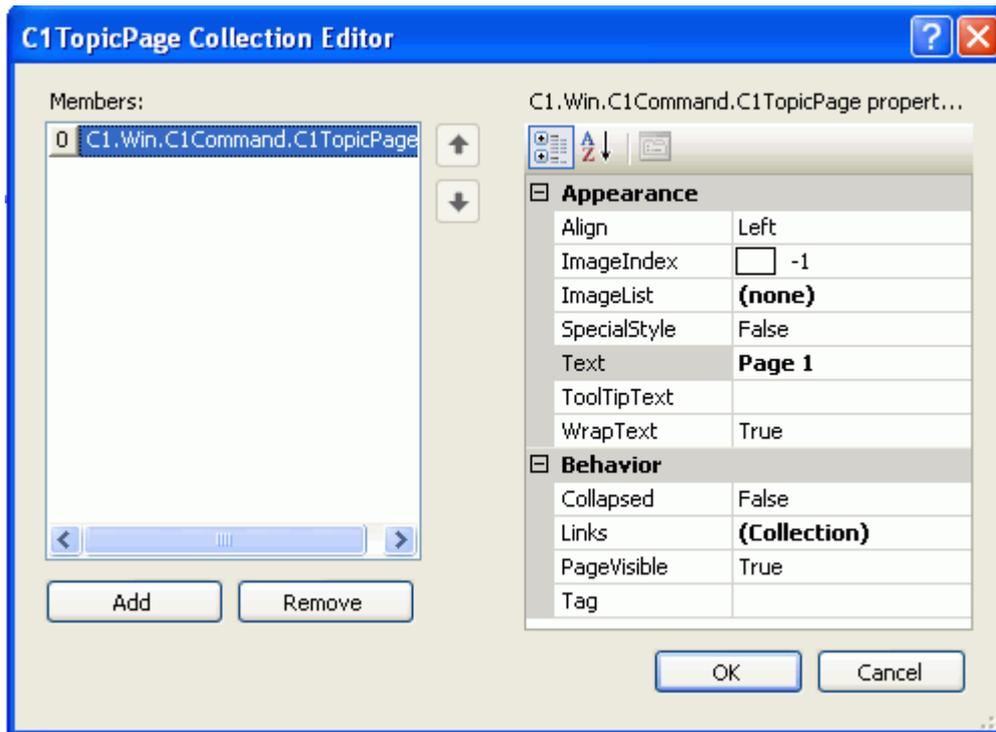
Option 2

1. Drop a `C1OutBar` control onto the form.
2. Click on the smart tag (📌) in the upper right corner of the `C1OutBar` control and then click on **Edit Pages** from the `C1OutBar Tasks` menu.



TopicPage Collection Editor

The **C1TopicPage Collection Editor** allows the user to add or remove the pages in the `C1TopicBar` control as well as edit the pages' properties.



To access the C1TopicPage Collection Editor:

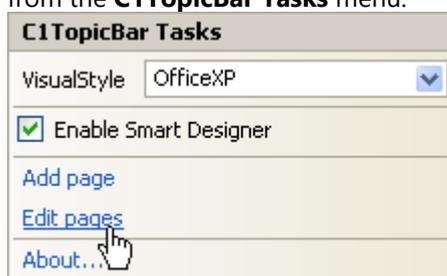
There are two ways to access the **C1TopicPage Collection Editor**, either through the **Pages** property in the **C1TopicBar Properties** window, or the **Edit Pages** in the **C1TopicBar Tasks** menu. For more information about how to use the **C1TopicBar Tasks** menu, see [C1TopicBar Smart Tag](#).

Option 1

1. Drop a [C1TopicBar](#) onto the form.
2. Click on the **ellipsis** button next to the **Pages** property in the [C1TopicBar's Properties](#) window.

Option 2

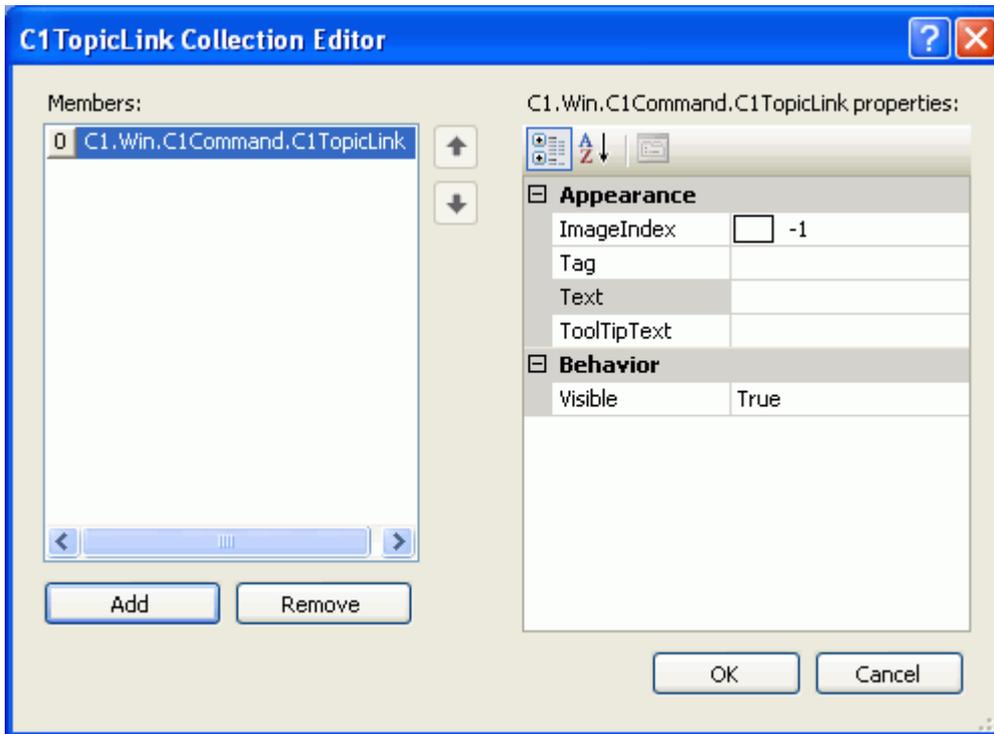
1. Drop a [C1TopicBar](#) onto the form.
2. Click on the smart tag (☰) in the upper right corner of the [C1TopicBar](#) control and then click on **Edit Pages** from the **C1TopicBar Tasks** menu.



TopicLink Collection Editor

The **C1TopicLink Collection Editor** allows the user to add or remove the links in the [C1TopicBar](#) control as well as edit

the links' properties.



To access the C1TopicLink Collection Editor:

You can access the **C1TopicLink Collection Editor** through the **Links** property in the **C1TopicPages Collection Editor**.

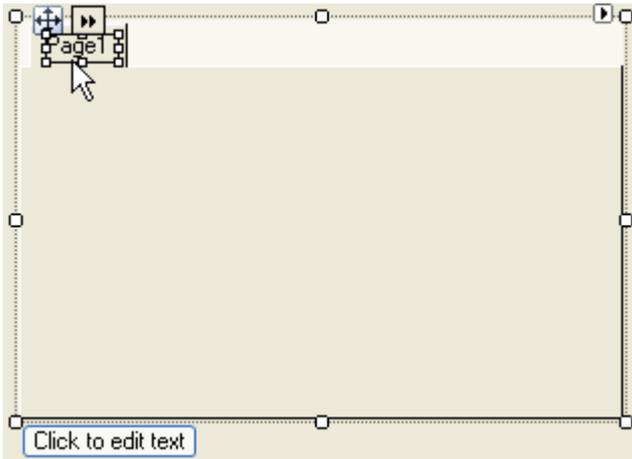
1. Drop a **C1TopicBar** onto the form.
2. Click on the smart tag (P) in the upper right corner of the **C1TopicBar** control and then click on **Edit Pages** from the **C1TopicBar Tasks** menu. Click on the **ellipsis** button next to the **Links** property.

Command Smart Designer

C1Command features a **Smart Designer** for improved design-time interaction for **C1MainMenu**, **C1ToolBar**, **C1DockingTab**, **C1OutBar**, **C1NavBar**, and **C1TopicBar** controls. An **Open** button appears, , when you select one of the **C1Command** controls and mouse over it on the form. Clicking on the **Open** button opens the floating toolbar associated with the selected control on the form.

Each toolbar includes ToolTips for its toolbar items to enhance the user interaction. Additionally, each toolbar provides command buttons and dialog boxes with common properties to quickly configure the **C1Command** controls without leaving the design form. This solves the earlier problem of having to drill down through the **C1Command** controls' properties in the Properties window.

In addition to the built-in toolbars, dialog boxes, and command buttons, the Smart Designers include directions for customizing tabs items on the **C1DockingTab** control to make your design-time experience more intuitive. When you mouse over a tab item in the **C1DockingTab** control, a label appears with a simple command statement instructing you what action to perform.



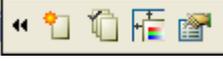
You can use the **Smart Designer** feature to create a functional menu, toolbar, docking tab, outbar, navbar, or topicbar at design time.

This section describes the functionality of the **Smart Designer's** toolbars associated with each **C1Command** control and how to expose each floating toolbar.

Smart Designer Floating Toolbars

The Smart Designer for the **C1Command** suite consists of the following floating toolbars:

Floating Toolbar	Description
	C1MainMenu toolbar: The C1MainMenu toolbar allows you to add new command links, edit command links, edit main menus appearance, and edit miscellaneous main menu properties.
	C1CommandLink/C1Command toolbar: The C1CommandLink toolbar allows you to add, remove, or modify existing commands.
	C1CommandMenu toolbar: The C1CommandMenu toolbar allows you to add new command links, edit command links, edit command properties, edit C1CommandMenu's appearance, and edit the side caption properties.
	C1ToolBar toolbar: The C1ToolBar toolbar allows you to Edit or add command links to the toolbar, set the appearance properties for the toolbar, Enable or disable merging, wrapping, or ToolTips for toolbar buttons, and set the color and font styles for the toolbar buttons.
	C1DockingTab toolbar: The C1DockingTab toolbar allows you to add tab pages to C1DockingTab control, enable or disable behavior settings for C1DockingTab , and set the color and font styles for the C1DockingTab control.
	C1DockingTabPage toolbar: The C1DockingTabPage toolbar allows you to set the color and font styles for the tab pages.
	C1NavBar toolbar: The C1NavBar toolbar allows you to add buttons to the C1NavBar , set the C1NavBar appearance properties, and set miscellaneous properties.
	C1NavBarButton toolbar: The C1NavBarButton toolbar allows you to modify the properties for the selected button in the C1NavBar control.

	C1OutBar toolbar: The C1OutBar toolbar allows you to add empty pages or pages with a toolbar, set the appearance properties for the C1OutBar , and set the miscellaneous properties for the C1OutBar control.
	C1OutPage toolbar: The C1OutPage toolbar allows you to apply property settings to the selected C1OutPage .
	C1TopicBar toolbar: The C1TopicBar toolbar allows you to add a new page link to the C1TopicBar , edit the pages, edit the topic bar's appearance and layout, and edit miscellaneous properties.
	C1TopicPage toolbar: The C1TopicPage toolbar allows you to add a topic link, remove the topic page, or edit the topic page's appearance properties.
	C1TopicLink toolbar: The C1TopicLink toolbar enables you to modify the selected topic link's appearance or remove the existing topic link.

The Smart Designer's toolbars are referred as floating toolbars since their behavior is slightly different from typical toolbars. The Smart Designer's toolbars appear only when the control is active and they can't be docked to other controls.

This section describes the functionality of the buttons in each floating toolbar.

C1MainMenu Toolbar

The **C1MainMenu** toolbar appears for the **C1MainMenu** control. To expose the **C1MainMenu** toolbar, select the **C1MainMenu** control and move your cursor so it appears on the **C1MainMenu** control. The **Open** button, , appears for the **C1MainMenu** toolbar.

Opening and Closing the C1MainMenu Toolbar

To open the **C1MainMenu** toolbar, click on the  button. To close the **C1MainMenu** toolbar, click on the  button.

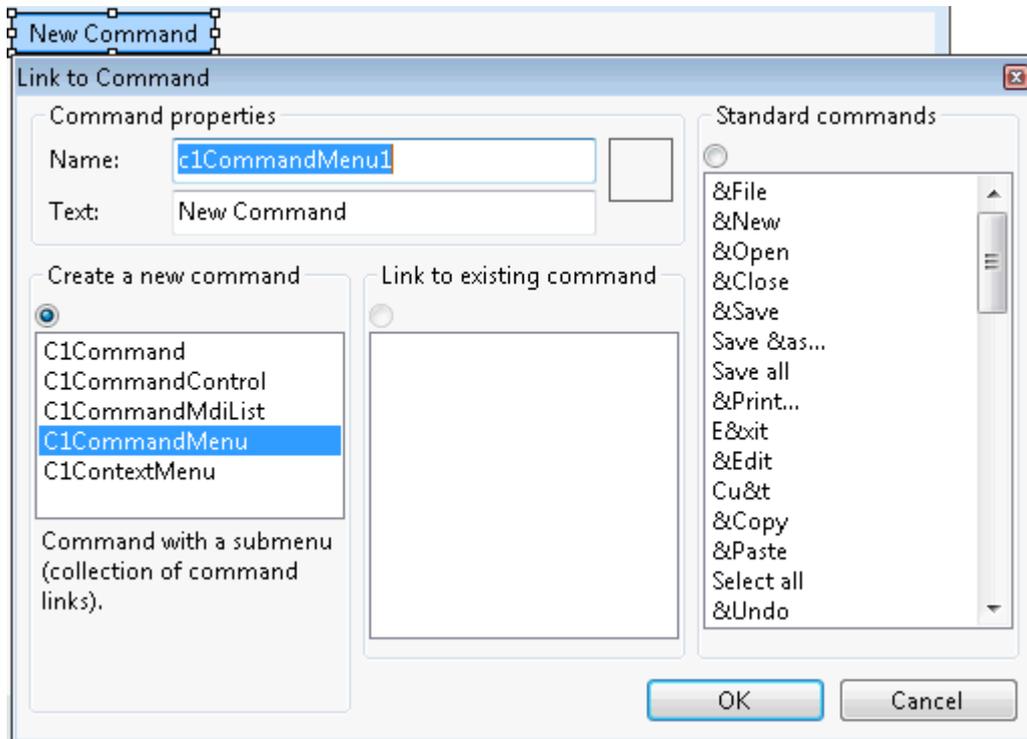
The **C1MainMenu** toolbar consists of the following command buttons:

Toolbar Button	Description
	Add new command Link/command: Adds a new command after the current command to the main menu.
	Edit command links: Opens the C1CommandLink Collection Editor for you to edit the command links.
	Edit main menu appearance: Opens the C1MainMenu appearance dialog box where you can set the general appearance properties for the C1MainMenu control.
	Edit miscellaneous properties: Opens the Miscellaneous dialog box for the C1MainMenu control where you can apply behavior settings to the C1MainMenu control.

Add new command link/command

Clicking the **Add new command link/command** button adds a new command after the current command. It displays the **Link to Command** designer below the new command so you can easily edit the new command without leaving

the design surface.

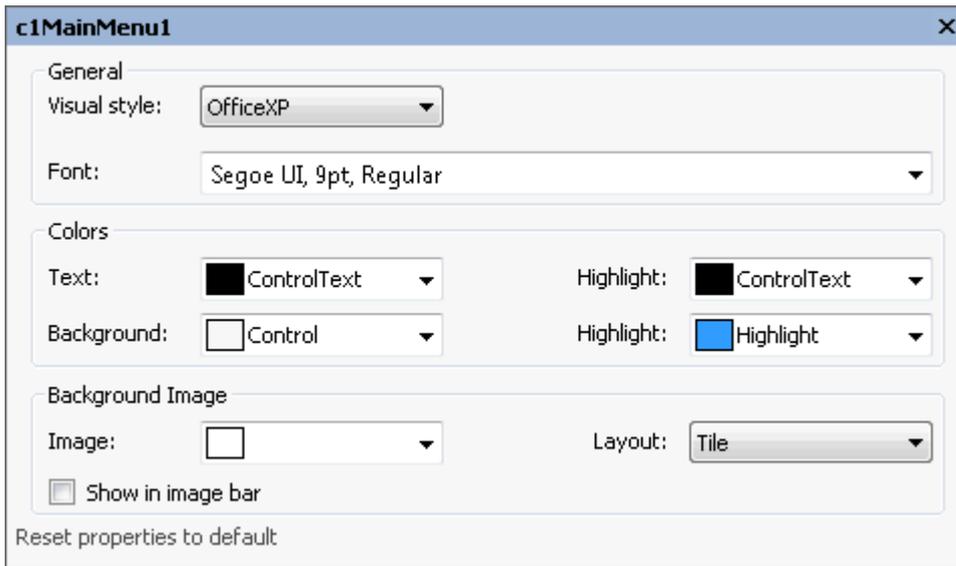


Edit command links

Clicking on the **Edit Command Links** button opens the **C1CommandLink Collection Editor** where you can add or remove command links and edit the commandlink's properties.

Edit main menu appearance

Clicking on the **Edit main menu appearance** button opens the **C1MainMenu Appearance** dialog box where you can modify the appearance properties for the menu items.



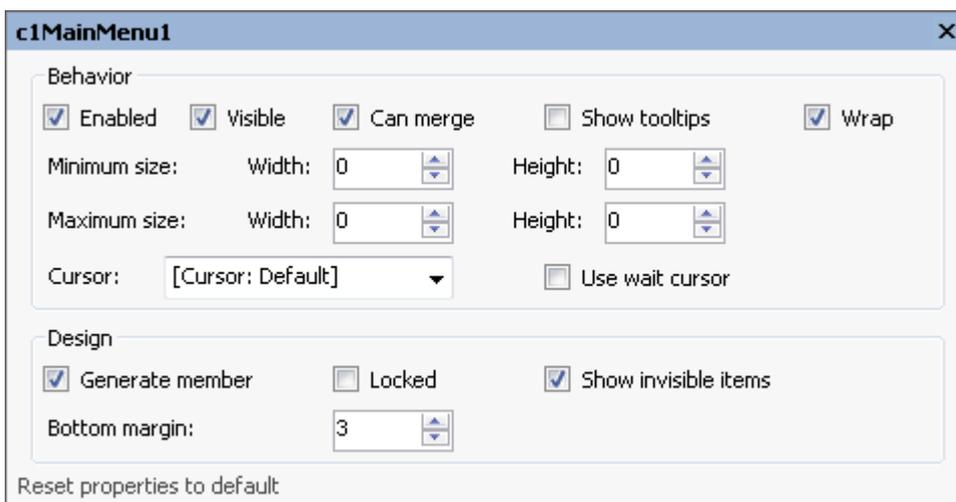
The following table defines the items included in the **Appearance** dialog box:

Item	Description
General	
Text	The Text textbox displays the text name that is associated with the C1MainMenu control. To rename the text name for the command, select the text in the Text textbox and type the desired text name.
Font	The Font drop-down box opens the Font dialog box where you can modify the Font style properties for the C1MainMenu control.
Colors	
Foreground	The Foreground drop-down list box contains the Custom, System, and Web colors for you to select from to set the Foreground color for the C1MainMenu control.
Highlight (ForeHiColor)	The Highlight drop-down list box contains the Custom, System, and Web colors for you to select from to set the text color of the highlighted item in the C1MainMenu control.
Background	The Background drop-down list box contains the Custom, System, and Web colors for you to select from to set the back color of the highlighted item in the C1MainMenu control.
Highlight (BackHiColor)	The Highlight drop-down list box contains the Custom, System, and Web colors for you to select from to set the back color of the highlighted item in the C1MainMenu control.
Background Image	
Image	The Image drop-down box opens an Open dialog box where you can apply the background image used for the C1MainMenu control.
Layout	The Layout drop-down box opens a list of layout items (None, Tile, Center, Stretch, and Zoom) for you to select from that gets the background image layout for the C1MainMenu control.

Show in image bar	The Show in image bar check box determines whether to show background image in image bar when the toolbar's style is Drop-downMenu.
Reset properties to default	Selecting the Reset properties to default item resets the modified C1MainMenu appearance properties back to their default values.

Edit Miscellaneous Properties

Clicking on the **Edit Miscellaneous Properties** button opens the **C1MainMenu Miscellaneous** dialog box where you can edit miscellaneous properties for the [C1MainMenu](#) control.



The following table defines the items included in the **c1MainMenu Miscellaneous** properties dialog box:

Items	Description
Behavior	
Enabled	The Enabled check box indicates whether the selected command will be enabled at run time. By default, the C1Command.Enabled property is set to True .
Visible	The Visible check box indicates whether the selected command will be shown at run time. By default, the C1Command.Visible property is set to True .
Minimum Size	The Minimum size field includes a Width and Height NumericUpDown controls for specifying the minimum width and height size for the C1MainMenu control.
Maximum Size	The Maximum size field includes a Width and Height NumericUpDown controls for specifying the maximum width and height size for the C1MainMenu control.
Can Merge	The Can merge check box indicates whether to merge MDI child menu with MDI parent menu. Selecting the check box enables this property and deselecting the check box disables this property.
Show tooltips	The Show tooltips check box indicates whether to show ToolTip texts when the mouse cursor is over the menu item. Selecting the check box enables this property and deselecting the check box disables this property.
Wrap	The Wrap check box gets the value (True , if selected; False , if deselected) to wrap the menu or show a "More..." button if not all items fit on a single line.

Cursor	The Cursor drop-down box opens a list of different cursor items (None, Tile, Center, Stretch, and Zoom) for you to select from that appears when the pointer moves over the C1MainMenu control.
Use wait cursor	The Use wait cursor check box indicates whether to use wait cursor.
Design	
Generate member	The Generate member check box indicates whether to generate the member for the C1MainMenu control. (True , if selected; False , if deselected)
Locked	The Locked check box indicates whether the C1MainMenu is locked. (True , if selected; False , if deselected)
Show invisible items	The Show invisible items check box indicates whether to show invisible items in the C1MainMenu control.
Bottom margin	The Bottom margin gets the space for the bottom margin.
Reset properties to default	Selecting the Reset properties to default item resets the modified C1MainMenu properties back to their default values.

C1CommandLink Toolbar

The [C1CommandLink](#) or [C1Command](#) Toolbar appears for the [C1CommandLinks/C1Commands](#) in the [C1MainMenu](#) and [C1ToolBar](#) controls.

Exposing the C1CommandLink Toolbar

To expose the [C1CommandLink](#) toolbar select the [C1Command](#) item in the [C1ToolBar](#) or [C1MainMenu](#) control. The [C1CommandLink](#) toolbar appears below the menu item or toolbar item like the following image:



C1CommandLink Toolbar Buttons

The [C1CommandLink](#) toolbar consists of the following command buttons:

Toolbar Button	Description
	Insert command link: Adds a new command before the selected command.
	Delete command link: Deletes the selected command.
	Edit command link properties: Apply style, layout, and tab layout to the C1DockingTab control.

	Edit command properties: Set the text and font styles and apply images to the commands.
	Change linked command: Opens the Link to Command designer where you can change the link to command in the link to existing command list box.

Insert command link

Clicking on the **Insert command link** button adds a new command before the selected command.

Delete command link

Clicking on the **Delete command link** button deletes the selected command.

CommandLink properties

Clicking on the **CommandLink properties** button opens the **CommandLink properties** dialog box where you can apply settings to the selected command.

The following table defines the items included in the [C1CommandLink](#) **CommandLink** properties dialog box:

Item	Description
Command	
Command	The Command drop-down list box displays the list of command types that pertain to the selected C1CommandMenu .
Text	The Text textbox displays the text name that appears on the selected command. To rename the text name for the command, select the text in the Text textbox and type the desired text name.
Tooltip	The Tooltip textbox displays ToolTip that appears on the selected command. If there is no ToolTip defined, then the ToolTip textbox is empty. To modify or create a ToolTip, enter text in the ToolTip textbox.
Appearance	
Button look	The Button look drop-down list box contains the following values for you to select from: Default, Text, Image, and TextAndImage to set the button look for the command link
Owner draw	Selecting the Owner draw check box enables the C1CommandLink.OwnerDraw property.
Default item	Selecting the Default item check box gets the default appearance for the selected command link.
Behavior	
Merge type	The Merge type drop-down list box contains the following merge type values for you to select from: Add, Replace, MergeItems, and Remove to get the merge type for the command link. The default value is Add.
Merge order	The Merge order NumericUpDown box gets the value indicating the relative position of the menu item when its merged with another.
Delimiter	Selecting the Delimiter check box draws a delimiter before the selected command link.

New column	Selecting the New column check box enables the C1CommandLink.NewColumn property.
Always recent	Selecting the Always recent check box enables the C1CommandLink.AlwaysRecent property.
Reset properties to default	Selecting the Reset properties to default item resets the modified C1Command properties back to their default values.

C1Command properties

Clicking on the **C1Command properties** button opens the **C1CommandMenu Command** properties dialog box where you can apply settings to the selected command.



The following table defines the items included in the **c1CommandMenu Command** properties dialog box:

Fields	Description
General	
Text	The Text textbox displays the text name that appears on the selected command. To rename the text name for the command, select the text in the Text textbox and type the desired text name.
Show text As Tooltip	The Show text as Tooltip check box displays the value of the C1Command.Text property as the Tooltip when the check box is selected.
Tooltip	The Tooltip textbox displays a Tooltip that appears on the selected command. If there is no Tooltip defined, then the Tooltip textbox is empty. To modify or create a Tooltip, enter text in the Tooltip textbox. Note, that the C1CommandLink.TooltipText property takes precedence over the C1Command.TooltipText property.
Shortcut	The Shortcut drop-down box displays a list of keys for you to select from to associate a keyboard shortcut with the selected command.
Show shortcut	The Show shortcut check box indicates whether the shortcut for the selected command will be shown. The C1Command.ShowShortCut is enabled by default.
Category	The Category drop-down list box displays the category for the selected command.
C1ContextMenu	The C1ContextMenu drop-down list box displays the name(s) of the C1ContextMenu(s) so you can easily select the C1ContextMenu you want to associate with the selected command.
Image	
Icon	The Icon drop-down list box opens an Open dialog box for you to locate the icon to attach to the selected command.
Image	The Image drop-down list box displays the current image attached to the selected command. Clicking on the drop-down arrow opens the Open dialog box where you can locate the image you would like to associate with the selected command.

Image index	The Image index drop-down list box displays the index value of the command image.
Behavior	
Visible	The Visible check box indicates whether the selected command will be shown at run time. By default, the <code>C1Command.Visible</code> property is set to True .
Enabled	The Enabled check box indicates whether the selected command will be enabled at run time. By default, the <code>C1Command.Enabled</code> property is set to True .
Pressed	The Pressed check box indicates whether the selected command is pressed. By default, the <code>C1Command.Pressed</code> property is set to True .
Show drop down arrow	The Show drop down arrow check box indicates whether or not to show the drop down arrow when the selected command is in a toolbar.
Checked	The Checked check box indicates whether or not the command is checked.
Auto-toggle checked state	The Auto-toggle checked state check box indicates whether the <code>C1Command.Checked</code> property is toggled automatically when this command is invoked.
Reset properties to default	Selecting the Reset properties to default item resets the modified <code>C1Command</code> properties back to their default values.

Change linked command

Clicking on the **Change linked command** button opens the **Link to Command** designer where you can change the link to command in the link to existing command list box.

C1CommandMenu Toolbar

The `C1CommandMenu` toolbar appears for the submenu items in the `C1CommandMenu` of the `C1MainMenu` and `C1ToolBar` controls.

Exposing the C1CommandMenu Toolbar

To expose the `C1CommandMenu` toolbar, select the `C1CommandMenu` item and its submenu items in the `C1ToolBar` or `C1MainMenu` control, and click on the open button, , to open the floating `C1CommandMenu` toolbar like the following image:



C1CommandMenu Toolbar Buttons

The `C1CommandMenu` toolbar consists of the following command buttons:

Toolbar Button	Description

	Add new command link/command: Adds an empty command link and opens the Link to Command dialog box for you to create a new command/command link for the empty command.
	Edit command links: Opens the C1CommandLink Collection Editor with the current command link in the Members list box for you to edit.
	Edit command properties: Opens the C1CommandMenu Command properties dialog box where you can modify the selected command's properties.
	Edit menu appearance: Opens the C1CommandMenu Appearance dialog box where you can modify the appearance properties for the selected command.
	Edit side caption properties: Opens the C1CommandMenu FormCaption properties dialog box where you can modify the properties for the side caption in the selected menu and its submenus.

Insert command link

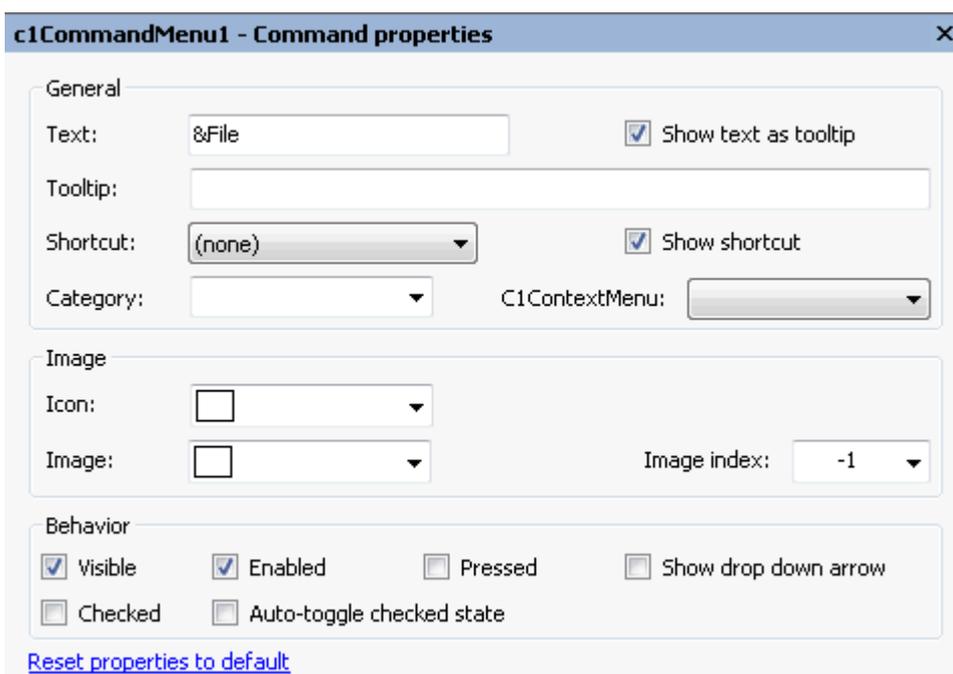
Clicking on the **Add new command link/command** button adds an empty command link and opens the **Link to Command** dialog box for you to create a new command/command link for the empty command.

Edit command links

Clicking on the **Edit command links** button opens the **C1CommandLink Collection Editor** with the current command link in the Members list box for you to edit.

Edit command properties

Clicking on the **Edit command properties** button opens the **C1CommandMenu Command** properties dialog box where you can modify the selected command's properties.



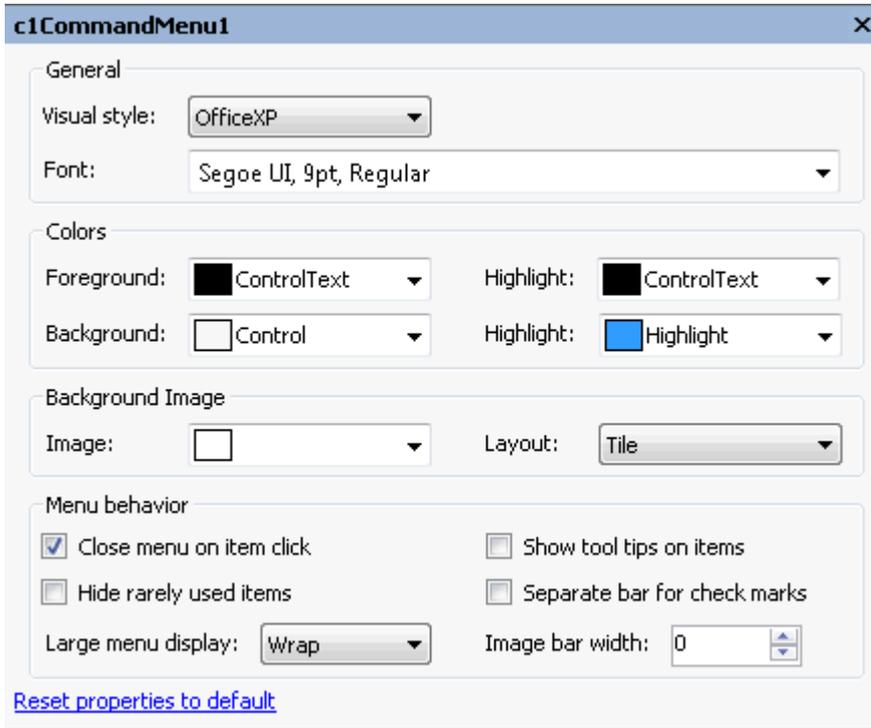
The following table defines the items included in the **C1CommandMenu Command** properties dialog box:

Item	Description
General	
Text	The Text textbox displays the text name that appears on the selected command. To rename the text name for the command, select the text in the Text textbox and type the desired text name.
Show text as tooltip	The Show text as Tooltip check box displays the value of the C1Command.Text property as the ToolTip when the check box is selected.
Tooltip	The Tooltip textbox displays ToolTip that appears on the selected command. If there is no ToolTip defined, then the ToolTip textbox is empty. To modify or create a ToolTip, enter text in the ToolTip textbox.
Shortcut	The Shortcut drop-down box displays a list of keys for you to select from to associate a keyboard shortcut with the selected command.
Show shortcut	The Show shortcut check box indicates whether the shortcut for the selected command will be shown. The C1Command.ShowShortcut is enabled by default.
Category	The Category drop-down list box displays the category for the selected command.
C1ContextMenu	The C1ContextMenu drop-down list box displays the name(s) of the C1ContextMenu(s) so you can easily select the C1ContextMenu you want to associate with the selected command.
Image	
Icon	The Icon drop-down list box opens an Open dialog box for you to locate the icon to attach to the selected command.
Image	The Image drop-down list box displays the current image attached to the selected command. Clicking on the drop-down arrow opens the Open dialog box where you can locate the image you would like to associate with the selected command.
Image Index	The Image index drop-down list box displays the index value of the command image.
Behavior	
Visible	The Visible check box indicates whether the selected command will be shown at run time. By default, the C1Command.Visible property is set to True .
Enabled	The Enabled check box indicates whether the selected command will be enabled at run time. By default, the C1Command.Enabled property is set to True .
Pressed	The Pressed check box indicates whether the selected command is pressed. By default, the C1Command.Pressed property is set to True .
Show drop down arrow	The Show drop down arrow check box indicates whether or not to show the drop down arrow when the selected command is in a toolbar.
Checked	The Checked check box indicates whether or not the command is checked.
Auto-toggle checked state	The Auto-toggle checked state check box indicates whether the C1Command.Checked property is toggled automatically when this command is invoked.
Reset properties	Selecting the Reset properties to default item resets the modified

to default | C1CommandMenu Command properties back to their default values.

Edit menu appearance

Clicking on the **Edit menu appearance** button opens the **C1CommandMenu Appearance** dialog box where you can modify the appearance properties for the selected command.



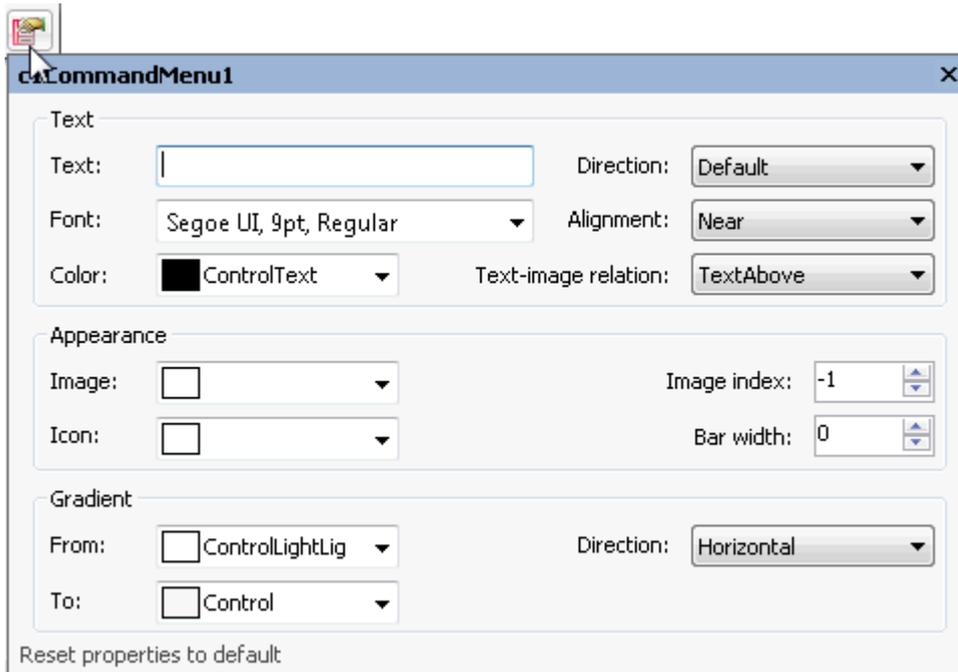
The following table defines the items included in the **C1CommandMenu Appearance** properties dialog box:

Item	Description
General	
Visual style	The Visual style drop-down box contains the following items for you to select from to change the style of the menu control: Custom, System, Office2007Blue, Office2007Black, Office2007Silver, Office2003Blue, Office2003Olive, Office2003Silver, OfficeXP, Classic, and WindowsXP.
Font	The Font drop-down box opens the Font dialog box where you can modify the Font style properties for the menu.
Colors	
Foreground	The Foreground drop-down list box contains the Custom, System, and Web colors for you to select from to set the Foreground color for the menu control.
Highlight (Foreground)	The Highlight drop-down box contains the Custom, System, and Web colors for you to select from to set the text color of the highlighted item.
Background	The Background drop-down box contains the Custom, System, and Web colors for you to select from to set the base background color of the menu.

Highlight (Background)	The Highlight drop-down list box contains the Custom, System, and Web colors for you to select from to set the base back color of the highlighted item in the menu.
Background Image	
Image	The Image drop-down box opens an Open dialog box where you can apply the background image used for the command.
Layout	The Layout drop-down box opens a list of layout items (None, Tile, Center, Stretch, and Zoom) for you to select from that gets the background image layout for the menu.
Menu behavior	
Close menu on item click	When the Close menu on item click check box is selected, it closes the submenu item after it's clicked at run time.
Show tooltips on items	When the Show tooltips on items check box is selected, it displays the ToolTips on the menu items at run time.
Hide rarely used items	When the Hide rarely used items check box is selected, it hides the rarely used menu items at run time.
Separate bar for check marks	When the Separate bar for check marks check box is selected, it shows check marks instead of images in a separate bar.
Large menu display	When the Large menu display check box is selected, it determines the way the large menus (when all items cannot fit in one column) are displayed.
Image bar width	When the Image bar width check box is selected, it gets or sets the width of the image/check box bar in the menu. If set to 0, the width is calculated automatically.
Reset properties to default	Selecting the Reset properties to default item resets the modified C1CommandMenu appearance properties back to their default values.

Edit side caption properties

Clicking on the **Edit side caption properties** button opens the **C1CommandMenu FormCaption** properties dialog box where you can modify the properties for the side caption in the selected menu and its submenus.



The following table defines the items included in the **C1CommandMenu FormCaption** dialog box:

Item	Description
Text	
Text	The Text textbox displays the text name that appears in the side caption of the selected C1CommandMenu. To rename the text name for the side caption, select the text in the Text textbox and type the desired text name.
Direction	The Direction drop-down box displays a list box that contains different values (Default, Horizontal, VerticalLeft, VerticalRight) for the text direction in the SideCaption for you to select from.
Font	The Font drop-down box opens the Font dialog box where you can modify the Font style properties for the SideCaption.
Alignment	The Alignment drop-down box displays a list box that contains the text alignment options (near, far, or center) for you to select from to specify the text alignment in the SideCaption.
Color	The Color drop-down box displays custom, web, and system colors for you to select from to set the text color of the SideCaption.
Text-image relation	The Text-image relation drop-down box displays a list box that contains different values (TextAbove, TextBelow, TextOnLeft, TextOnRight) for the text layout in the SideCaption for you to select from.
Appearance	
Image	The Image gets or sets the image for the SideCaption.
Image index	The Image index gets or sets the index of the image for the SideCaption in the C1CommandHolder.ImageList .
Icon	The Icon drop-down box opens the Open dialog box where you can select the icon for the SideCaption.

Bar width	The Bar width gets or sets the width of the image/check box bar in the menu. If set to 0, the width is calculated automatically.
Gradient	
From	The From drop-down box opens the custom, web, and system colors for you to choose from to set the beginning color of the gradient for the SideCaption.
To	The To drop-down box opens the custom, web, and system colors for you to choose from to set the ending color of the gradient for the SideCaption.
Direction	The Direction drop-down box includes different gradient directions (horizontal, vertical, forward diagonal, or backward diagonal) for you to set the type of gradient direction in the SideCaption.
Reset properties to default	Selecting the Reset properties to default item resets the modified C1CommandMenu FormCaption properties back to their default values.

C1DockingTab Toolbars

The [C1DockingTab](#) toolbar and the [C1DockingTabPage](#) toolbars appear for the [C1DockingTab](#) control. To expose the [C1DockingTab](#) toolbar, select the [C1DockingTab](#) control and slide your cursor on the [C1DockingTab](#) control. To expose the [C1DockingTabPage](#) toolbar slide you cursor inside the [C1DockingTabPage](#) area of the [C1DockingTab](#) control.

Opening and Closing the C1DockingTab Toolbar

To open the [C1DockingTab](#) toolbar, click on the  button. To close the [C1DockingTab](#) toolbar, click on the  button.

The **C1DockingTab** toolbar consists of the following command buttons:

Toolbar Button	Description
	Add tab page: Add tab pages to C1DockingTab .
	Edit the collection of tab pages: Add or remove tab pages or modify the properties for each tab page through the C1DockingTabPage Collection Editor .
	Edit tabs area properties: Apply style, layout, and tab layout to the C1DockingTab control.
	Edit docking tab appearance: Set the color and font styles and apply images to the C1DockingTab control.
	Edit docking tab behavior: Apply behavior settings to the C1DockingTab control.

Add New Page

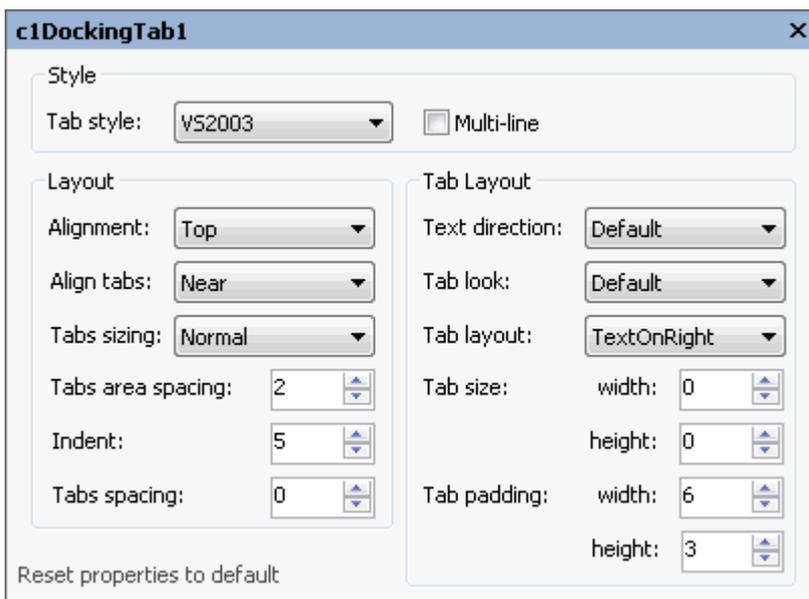
Clicking on the **Add tab page** button adds a tab after the existing tab.

Edit the collection of tab pages

Clicking on the **Edit the collection of tab pages** button opens the **C1DockingTabPage Collection Editor** where you can modify the property settings for each tab page as well as add or remove tab pages. For more information on the **C1DockingTabPage Collection Editor**, see [C1DockingTabPage Collection Editor](#).

Edit tabs area properties

Clicking on the **Edit tabs area properties** button opens the Tab Area dialog box where you can modify the settings for the tab area in the [C1DockingTab](#) control.



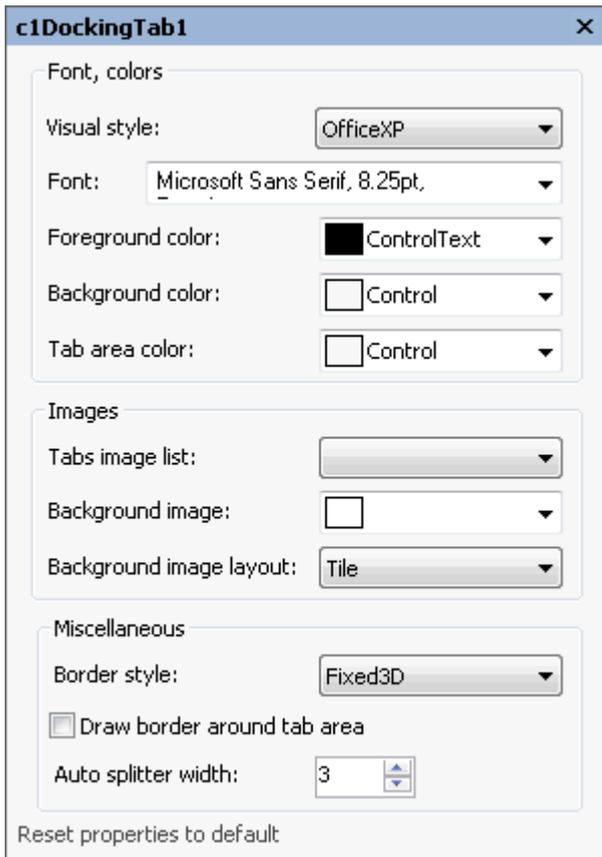
The following table defines the items included in the **Tab Area** properties dialog box:

Fields	Description
Style	
Tab style	The TabStyle drop-down box displays a list of tab style items (Default , WindowsXP , Classic , Sloping , Rounded , Office2003 , and Office2007) for you to select from to set the tab style for the C1DockingTab control.
MultiLine	The MultiLine check box gets or sets a value, (True if selected; False , if deselected) indicating whether more than one row of tabs can be displayed.
Layout	
Alignment	The Alignment drop-down box displays a list box that contains the area of the control (top, bottom, left or right) for you to select from to specify where the tabs are aligned.
Align tabs	The Alignment tabs drop-down box contains the items (Near, Center, or Far) for you to select from to specify how tabs are aligned along side of the page content area.
Tabs sizing	The Tab sizing drop-down box contains the items (Normal, Fit, FillToEnd, and User) to select from to set how the tabs are sized.
Tabs area spacing	The TabsArea spacing gets or sets the spacing between the edge of the tab area and the tabs.

Indent	The Indent property gets or sets the indentation of the first tab from the side of the control.
Tabs spacing	The Tabs spacing gets or sets the distance between the tabs (the distance may be negative to overlap the tabs).
Tab Layout	
Text direction	The Text direction item gets or sets the direction of the text drawn on the tabs.
Tab look	The Tab look drop-down box contains the items (Default, Text, Image, and TextAndImage) for you to select from to set the look of the tabs.
Tab layout	The Tab layout drop-down box contains the items (TextOnRight, TextOnLeft, TextBelow, TextAbove) for you to select from to set the layout of the text and images on the tabs.
Tab size	The Tab size gets the width and height of the tabs.
Tab padding	The Tab padding gets the width and height of the padding for the tabs.
Reset properties to default	Selecting the Reset properties to default item resets the modified C1DockingTab properties back to their default values.

Edit docking tab appearance

Clicking on the **Edit docking tab appearance** button opens the **Appearance** dialog box where you can modify the appearance properties for the [C1DockingTab](#).



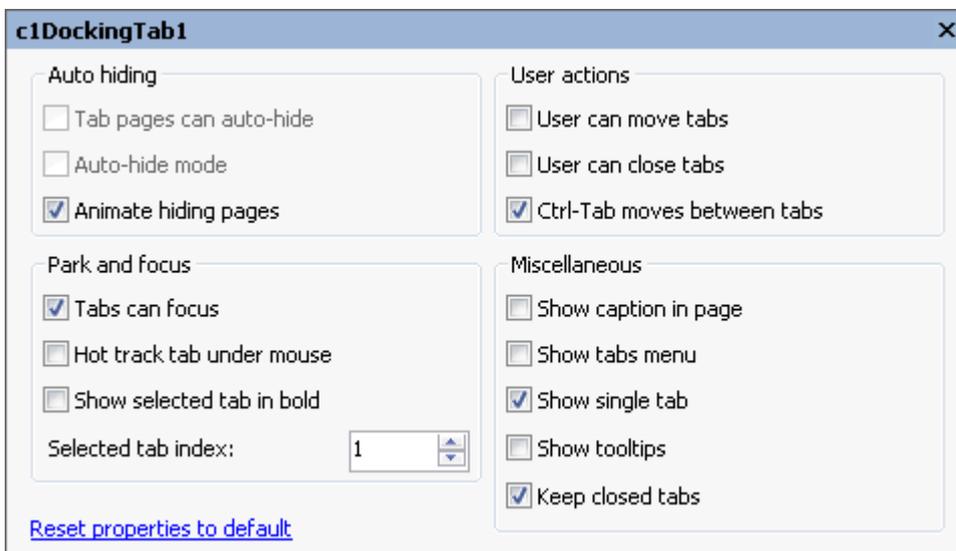
The following table defines the items included in the **Appearance** properties dialog box:

Fields	Description
Font, colors	
Visual style	The Visual style drop-down box contains the following items for you to select from to change the style of the C1DockingTab control: Custom, System, Office2007Blue, Office2007Black, Office2007Silver, Office2003Blue, Office2003Olive, Office2003Silver, OfficeXP, Classic, and WindowsXP.
Font	The Font drop-down box opens the Font dialog box where you can modify the Font style properties for the C1DockingTab control.
Foreground color	The Foreground drop-down list box contains the Custom, System, and Web colors for you to select from to set the Foreground color for the C1DockingTab control.
Background color	The Background drop-down list box contains the Custom, System, and Web colors for you to select from to set the back color of the C1DockingTab control.
Tab area color	The Tab area color drop-down list box contains the Custom, System, and Web colors for you to select from to set the tab area color for the C1DockingTab control.
Images	
Tabs image list	Manages the list of collection images for the tabs in the C1DockingTab control.
Background image	The Background image drop-down box opens an Open dialog box where you can apply the background image used for the C1DockingTab control.
Background	The Background image layout drop-down box opens a list of layout items (None, Tile,

image layout	Center, Stretch, and Zoom) for you to select from that gets the background image layout for the C1DockingTab control.
Miscellaneous	
Border style	The Border style drop-down box displays a list box that contains the different types of border styles (None, FixedSingle, Fixed3D) for you to select from to specify the border style for the C1DockingTab control.
Draw border around tab area	The Draw border around tab area check box indicates whether to draw a border around the tab area. (True , if selected; False , if deselected)
Auto splitter width	The Auto splitter width gets or sets the width of automatic splitters drawn between pages of the control when the page docking is enabled.
Reset properties to default	Selecting the Reset properties to default item resets the modified C1DockingTab properties back to their default values.

Edit docking tab behavior

Clicking on the **Edit docking tab behavior** button opens the **Behavior** dialog box where you can enable specific behavior properties for the **C1DockingTab**.



The following table defines the items included in the **Behavior** properties dialog box:

Item	Description
Auto hiding	
Tab pages can auto-hide	The Tab pages can auto hide check box indicates whether the tab pages can auto hide. (True if selected; False , if deselected)

Auto-hide mode	The Auto-hide mode check box indicates whether the <code>C1DockingTab</code> is in auto-hide mode. (True if selected; False , if deselected)
Animate hiding pages	The Animate hiding pages check box indicates whether to animate the hiding docking tab pages. (True if selected; False , if deselected)
Park and focus	
Tabs can focus	The Tabs can focus check box indicates whether tabs can receive focus on mouse click. (True if selected; False , if deselected)
Hot track tab under mouse	The Hot track tab under mouse check box indicates whether the control's tabs change in appearance when the mouse passes over them. (True if selected; False , if deselected)
Show selected tab in bold	The Show selected tab in bold check box indicates whether to bold the text in the selected tab. (True if selected; False , if deselected)
Selected tab index	The Selected tab index gets or sets the index of the currently selected page.
User actions	
User can move tabs	The User can move tabs check box indicates whether the end user can rearrange tabs by dragging them around at run time. (True if selected; False , if deselected)
User can close tabs	The User can close tabs check box indicates whether individual tab pages can be closed by the end user. If <code>C1DockingTab.CanCloseTabs</code> is True , a close icon appears either in the caption area (if <code>ShowCaption</code> is True), or in the tabs otherwise. (True if selected; False , if deselected)
Ctrl-Tab moves between tabs	The Ctrl-Tab moves between tabs check box indicates whether the <code>C1DockingTab</code> control handles <code>Ctrl-Tab</code> and <code>Ctrl-Shift-Tab</code> keys. (True if selected; False , if deselected)
Miscellaneous	
Show caption in a page	The Show caption in a page check box indicates whether the caption is shown on the pages. (True if selected; False , if deselected)
Show tabs menu	The Show tabs menu check box indicates whether to show a button with a drop-down list of all tabs. (True if selected; False , if deselected). This property is ignored in multiline mode.
Show single tab	The Show single tab check box indicates whether a tab will be shown when there is only one page in the control. (True if selected; False , if deselected)
Show tooltips	The Show tooltips check box indicates whether the <code>ToolTip</code> is shown when the mouse passes over the tab. (True if selected; False , if deselected)
Keep closed tabs	The Keep closed tabs check box indicates whether to keep a closed tab. (True if selected; False , if deselected)

Reset properties to default	Selecting the Reset properties to default item resets the modified C1DockingTab properties back to their default values.
------------------------------------	---

C1DockingTabPage toolbar

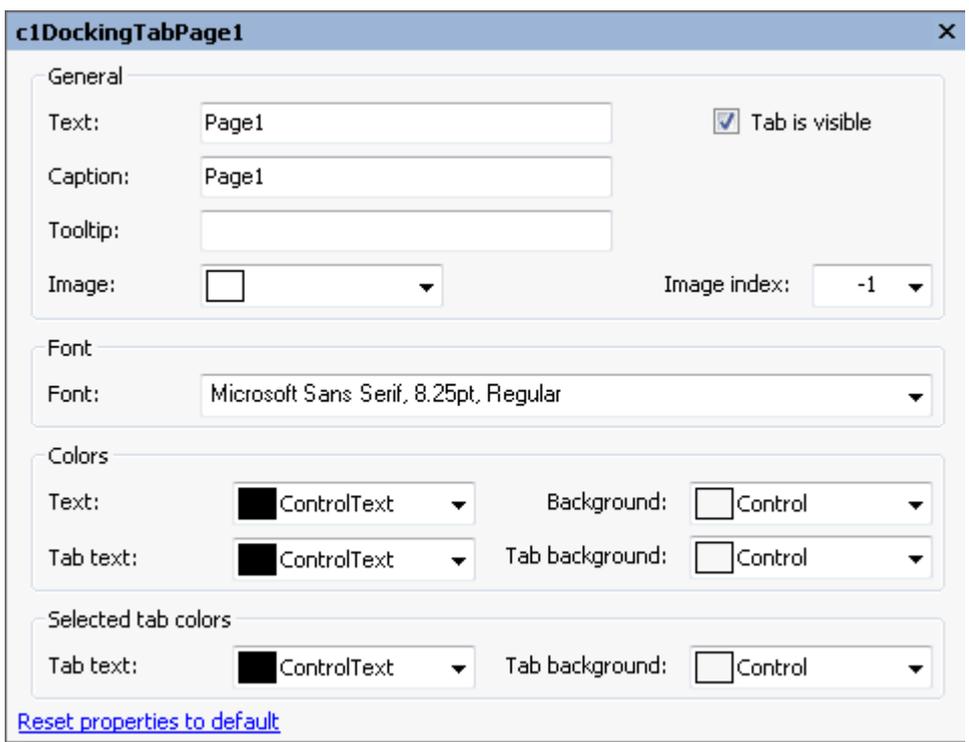
The **C1DockingTabPage** toolbar appears for the **C1DockingTabPage** control. To expose the **C1DockingTabPage** toolbar, select the **C1DockingTab** control and slide your cursor inside the **C1DockingTabPage** area of the **C1DockingTab** control.

The **C1DockingTabPage** toolbar consists of one command button:

Toolbar Button	Description
	Edit tab page properties: Set the color and font styles for the tab pages.

Edit tab page properties

Clicking on the **Edit tab page properties** button opens the **Appearance** dialog box where you can modify **C1DockingTabPage** appearance properties.



The following table defines the items included in the **Tab Page Properties** dialog box:

Item	Description
General	
Text	The Text textbox displays the text name that appears on the selected tab page. To rename the text name for the tab page, select the text in the Text textbox and type the desired text name.

Caption	The Caption text textbox displays the text name that appears on the caption page. To rename the text name for the tab page, select the text in the Text textbox and type the desired text name.
Tooltip	The ToolTip text textbox sets the ToolTip text for the tab page.
Image	The Image drop-down list box displays the current image attached to the selected tab page. Clicking on the drop-down arrow opens the Open dialog box where you can locate the image you would like to associate with the selected tab page.
Image index	The Image index drop-down list box displays the index value of the selected tab page image.
Font	
Font	The Font drop-down box opens the Font dialog box where you can modify the Font style properties for the selected tab page in the C1DockingTab control.
Colors	
Text	The Text drop-down list box contains the Custom, System, and Web colors for you to select from to set the text color on the tab and tab page.
Background	The Background drop-down list box contains the Custom, System, and Web colors for you to select from to set the back color of the tab and tab page.
Tab background	The Tab background drop-down list box contains the Custom, System, and Web colors for you to select from to set the back color of the tab.
Selected tab colors	
Text	The Text drop-down list box contains the Custom, System, and Web colors for you to select from to set the text color on the selected tab.
Background	The Background drop-down list box contains the Custom, System, and Web colors for you to select from to set the back color of the selected tab.
Reset properties to default	Selecting the Reset properties to default item resets the modified C1DockingTabPage properties back to their default values.

C1NavBar Toolbars

The [C1NavBar](#) toolbar appears for the [C1NavBar](#) control. To expose the [C1NavBar](#) toolbar, select the [C1NavBar](#) control and slide your cursor on the [C1NavBar](#) control.

Opening and Closing the C1NavBar Toolbar

To open the [C1NavBar](#) toolbar, click on the  button. To close the [C1NavBar](#) toolbar, click on the  button.

The [C1NavBar](#) toolbar consists of the following command buttons:

Toolbar Button	Description
	Add button and corresponding panel: Add buttons to C1NavBar .

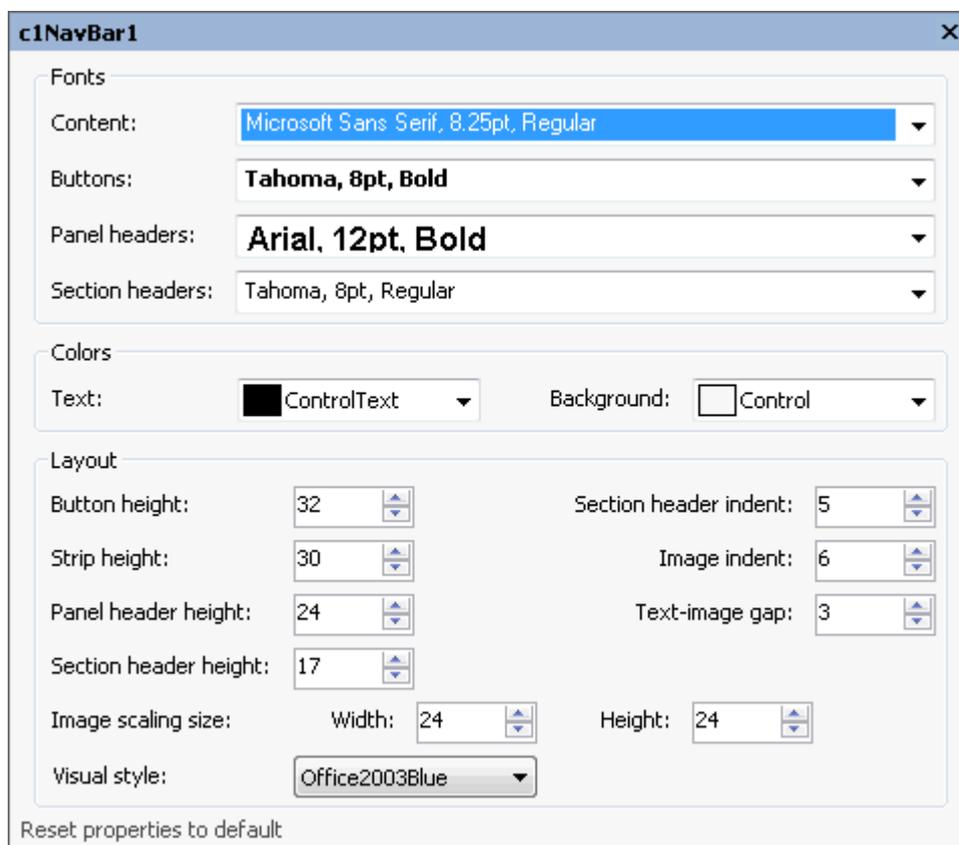
	<p>Edit navigation bar appearance and layout: Set the color and font styles and apply images to the C1NavBar control.</p>
	<p>Edit miscellaneous navigation bar properties: Opens the C1NavBar dialog box for the C1NavBar control where you can apply behavior properties to the C1NavBar control.</p>

Add button and corresponding panel

Clicking on the **Add button and corresponding panel** button adds a tab after the existing tab.

Edit navigation bar appearance and layout

Clicking on the **Edit navigation bar appearance and layout** button opens the **C1NavBar properties** dialog box where you can modify the appearance properties for the **C1NavBar**.



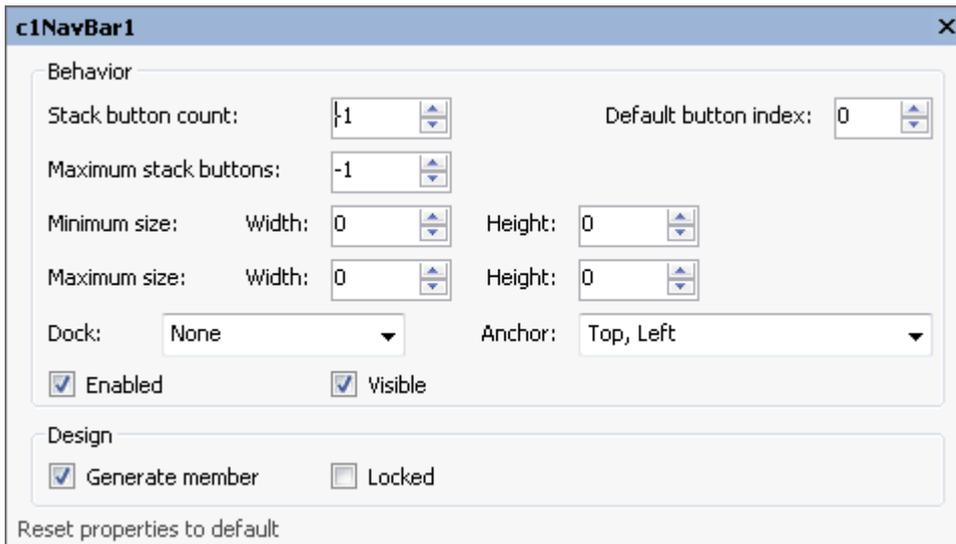
The following table defines the items included in the **C1NavBar** properties dialog box:

Item	Description
Fonts	
Content	The Contents drop-down box opens the Font dialog box where you can modify the Font style properties for the content in the C1NavBar control.
Buttons	The Buttons drop-down box opens the Font dialog box where you can modify the Font style properties for the buttons in the C1NavBar control.
Panel headers	The Panel headers drop-down box opens the Font dialog box where you

	can modify the Font style properties for the panel header in the C1NavBar control.
Section headers	The Section headers drop-down box opens the Font dialog box where you can modify the Font style properties for the section header in the C1NavBar control.
Colors	
Text	The Text drop-down list box contains the Custom, System, and Web colors for you to select from to set the Foreground color for the menu control.
Background	The Background drop-down list box contains the Custom, System, and Web colors for you to select from to set the back color of the C1NavBar control.
Layout	
ButtonHeight	The ButtonHeight sets the integer value for the height of the button.
Section header indent	The Section header indent determines the space before the text caption in a section header.
StripHeight	The StripHeight sets the integer value for the strip height of the button.
ImageIndent	The ImageIndent drop-down box sets the integer value for the height of the panel header.
PanelHeaderHeight	The PanelHeaderHeight sets the integer value for the height of the panel header.
Text-image gap	The Text-image gap gets the space between the image and the text on a stack button.
SectionHeaderHeight	The SectionHeaderHeight sets the integer value for the height of the section header.
Image scaling size	The Image scaling size sets the size, in pixels, of an image displayed on a stack button. The default is 24 x 24 pixels. To set the width for the image, use the Width numericupdown box and to set the height for the image, use the Height numericupdown box.
Visual style	The Visual style drop-down box contains the following items for you to select from to change the style of the C1NavBar control: Custom , System , Office2007Blue , Office2007Black , Office2007Silver , Office2003Blue , Office2003Olive , Office2003Silver , OfficeXP , Classic , and WindowsXP .
Reset properties to default	Selecting the Reset properties to default item resets the modified C1NavBar properties back to their default values.

Edit miscellaneous navigation bar properties

Clicking on the **Edit miscellaneous navigation bar properties** button opens the **C1NavBar** dialog box where you can modify **C1NavBar**s miscellaneous properties.



The following table defines the items included in the [C1NavBar](#) properties dialog box:

Item	Description
Behavior	
StackButtonCount	The StackButtonCount gets or sets the number of buttons displayed in the stack.
DefaultButtonIndex	The DefaultButtonIndex specifies the index of the button to be selected when the form opens.
Maximum stack buttons	The Maximum stack buttons gets or sets the maximum number of buttons in the stack. If set to -1 the maximum number is unlimited.
Minimum size	The Minimum size field includes a Width and Height NumericUpDown controls for specifying the minimum width and height size for the C1NavBar control.
Maximum size	The Maximum size field includes a Width and Height NumericUpDown controls for specifying the maximum width and height size for the C1NavBar control.
Dock	The Dock drop-down box contains items for you to select from to define which border of the C1NavBar control is bound to the container.
Anchor	The Anchor drop-down box defines the edges of the container to which the C1NavBar control is bound. When it's anchored to an edge, the distance between the C1NavBar 's closest edge and the specified edge will remain constant.
Enabled	The Enabled check box indicates whether the C1NavBar control will be enabled at run time.
Visible	The Visible check box indicates whether the C1NavBar will be shown at run time.
Design	
Generate member	The Generate member check box indicates whether to generate the member for the C1NavBar control. (True , if selected; False , if deselected)

Locked	The Locked check box indicates whether the C1NavBar is locked. (True , if selected; False , if deselected)
Reset properties to default	Selecting the Reset properties to default item resets the modified C1NavBar miscellaneous properties back to their default values.

C1OutBar Toolbars

The **C1OutBar** toolbar appears for the **C1OutBar** control. To expose the **C1OutBar** toolbar, select the **C1OutBar** control and position your cursor on the **C1OutBar** control.

Opening and Closing the C1OutBar Toolbar

To open the **C1OutBar** toolbar, click on the  button. To close the **C1OutBar** toolbar, click on the  button.

The **C1OutBar** toolbar consists of the following command buttons:

Toolbar Button	Description
	Add blank page: Add pages with a toolbar contained in each page.
	Add empty page: Add empty pages without a toolbar to the C1OutBar .
	Edit outbar appearance and layout: Set the color and font styles and apply images to the C1OutBar control.
	Edit miscellaneous outbar properties: Opens the Miscellaneous dialog box for the C1OutBar control where you can apply miscellaneous properties to the C1OutBar control.

Add blank page

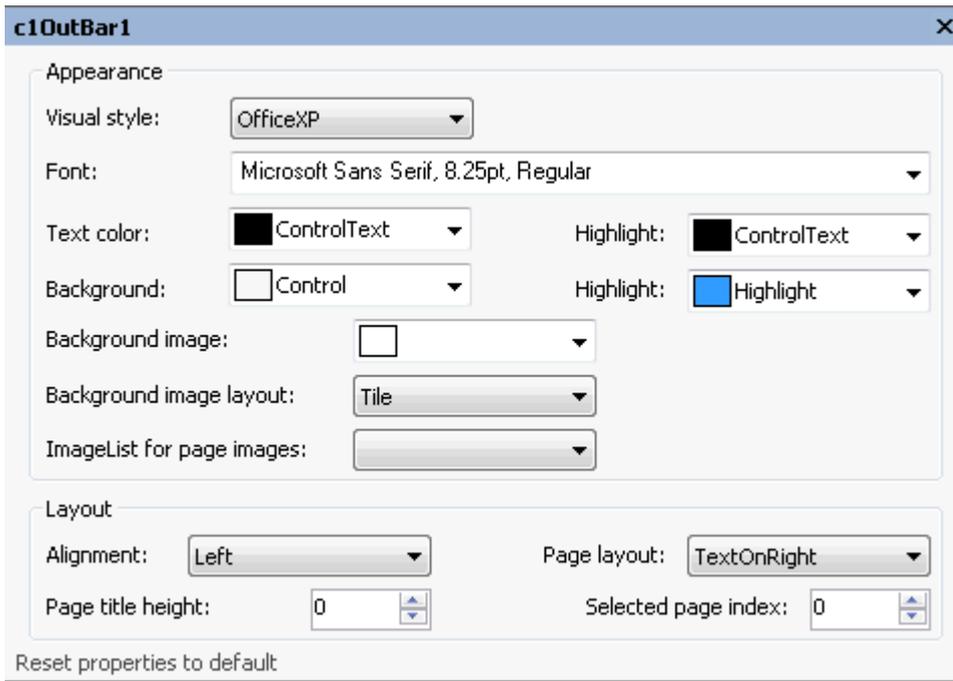
Clicking on the **Add blank page** button adds a new page with a toolbar to the **C1OutBar** control.

Add empty page

Clicking on the **Add empty page** button adds an empty page to the **C1OutBar** control.

Edit outbar appearance and layout

Clicking on the **Edit outbar appearance and layout** button opens the **Properties** dialog box where you can modify various properties for the **C1OutBar** control.



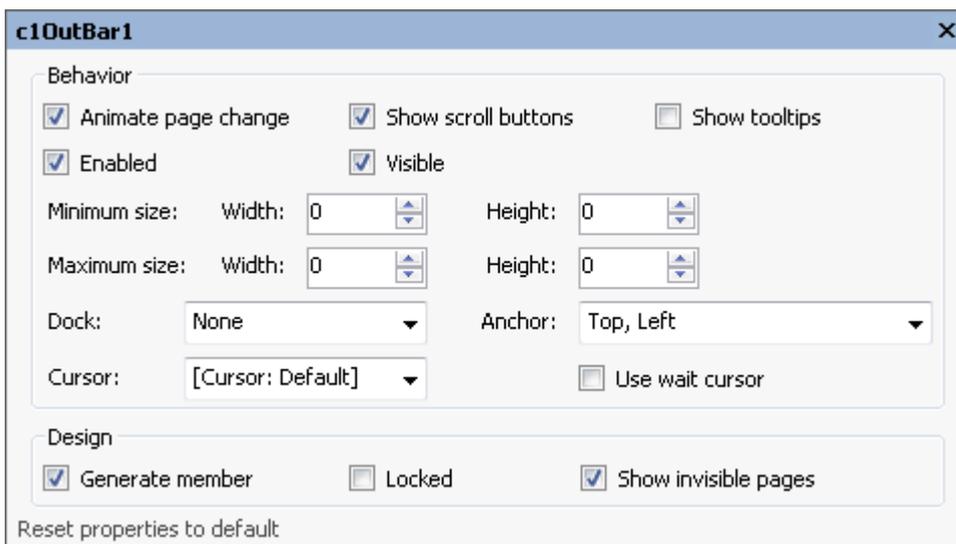
The following table defines the items included in the **C1OutBar Properties** dialog box:

Item	Description
Appearance	
Visual style	The Visual style drop-down box contains the following items for you to select from to change the style of the C1OutBar control: Custom, System, Office2007Blue, Office2007Black, Office2007Silver, Office2003Blue, Office2003Olive, Office2003Silver, OfficeXP, Classic, and WindowsXP.
Font	The Font drop-down box opens the Font dialog box where you can modify the Font style properties for the C1OutBar control.
Text color	The Foreground drop-down list box contains the Custom, System, and Web colors for you to select from to set the Foreground color for the highlighted text on C1OutBar control.
Highlight (ForeHiColor)	The Highlight drop-down list box contains the Custom, System, and Web colors for you to select from to set the fore color for the highlighted item on C1OutBar control.
Background	The Background drop-down list box contains the Custom, System, and Web colors for you to select from to set the back color in the C1OutBar control.
Highlight (BackHiColor)	The Highlight drop-down list box contains the Custom, System, and Web colors for you to select from to set the back color of the highlighted item in the C1OutBar control.
Background image	The Background image drop-down box opens an Open dialog box where you can apply the background image used for the C1OutBar control.
Background image layout	The Layout drop-down box opens a list of layout items (None, Tile, Center, Stretch, and Zoom) for you to select from that gets the background image layout for the C1OutBar control.
ImageList for page images	The ImageList for page images drop-down box gets the image list used to provide the images shown on the pages' title bars.

Layout	
Alignment	The Alignment drop-down box gets the alignment (Left, Right, Center) of text and image on the pages' title bars.
Page layout	The Page layout drop-down box contains items for you to set the text above, below, to the left or to the right of the image in the page title. The default value for the C1OutBar.PageLayout property is TextOnRight.
Page title height	The Page title height numericupdown box sets the height of each page title.
Selected page index	The Selected page index sets the index of the selected page.
Reset properties to default	Selecting the Reset properties to default item resets the modified C1OutBar properties back to their default values.

Edit miscellaneous properties

Clicking on the **Edit miscellaneous properties** button opens the **Miscellaneous** dialog box where you can modify **C1OutBar**'s **Miscellaneous** properties.



The following table defines the items included in the **C1OutBar Miscellaneous** dialog box:

Item	Description
Behavior	
Animate page change	The Animate page change gets or sets a value (True , if selected; False , if deselected) to indicate whether to use animation when changing the selected page.
Show scroll buttons	The Show scroll buttons gets or sets a value (True , if selected; False , if deselected) to indicate whether scroll buttons for scrolling toolbar links are visible.
Show Tooltips	The Show tooltips check box indicates whether to show ToolTip texts when the mouse cursor is over the page item. Selecting the check box enables this property and deselecting

	the check box disables this property.
Minimum size	The Minimum size field includes a Width and Height NumericUpDown controls for specifying the minimum width and height size for the C1OutBar control.
Maximum size	The Maximum size field includes a Width and Height NumericUpDown controls for specifying the maximum width and height size for the C1OutBar control.
Dock	The Dock drop-down box contains items for you to select from to define which border of the C1OutBar control is bound to the container.
Cursor	The Cursor drop-down box opens a list of different cursor items for you to set the type of cursor that appears when the pointer moves over the C1OutBar control.
Use Wait cursor	The Use wait cursor check box indicates whether to use wait cursor.
Anchor	The Anchor drop-down box defines the edges of the container to which the C1OutBar control is bound. When it's anchored to an edge, the distance between the C1OutBar's closest edge and the specified edge will remain constant.
Design	
Generate member	The Generate member check box indicates whether to generate the member for the C1OutBar control. (True , if selected; False , if deselected)
Locked	The Locked check box indicates whether the C1OutBar is locked. (True , if selected; False , if deselected)
Reset properties to default	Selecting the Reset properties to default item resets the modified C1OutBar's miscellaneous properties back to their default values.

C1TopicBar Toolbars

The Smart Designer for the [C1TopicBar](#) control provides the following toolbars for the topic bar, topic page, and topic link:

- C1TopicBar Toolbar
- C1TopicPage Toolbar
- C1TopicLink Toolbar

The **TopicBar**, [C1TopicPage](#), and [C1TopicLink](#) toolbars appear for the [C1TopicBar](#) control.

C1TopicBar Toolbar

The [C1TopicBar](#) toolbar appears for the [C1TopicBar](#) control. To expose the [C1TopicBar](#) toolbar, select the [C1TopicBar](#) control and position your cursor anywhere on the [C1TopicBar](#) control.

Opening and Closing the C1TopicBar Toolbar

To open the [C1TopicBar](#) toolbar, click on the  button. To close the [C1TopicBar](#) toolbar, click on the  button.

The [C1TopicBar](#) toolbar consists of the following command buttons:

Toolbar Button	Description
----------------	-------------

	<p>Add topic page: Adds a new topic page.</p>
	<p>Edit Pages: Opens the C1TopicPage Collection Editor where you can add or remove C1TopicPages or modify their property settings.</p>
	<p>Edit topicbar appearance and layout: Opens the C1TopicBar Appearance dialog box where you can modify the appearance, behavior, and layout styles for the C1TopicBar control.</p>
	<p>Edit miscellaneous properties: Opens the Miscellaneous dialog box for the C1TopicBar control where you can apply miscellaneous properties to the C1TopicBar control.</p>

Add new page

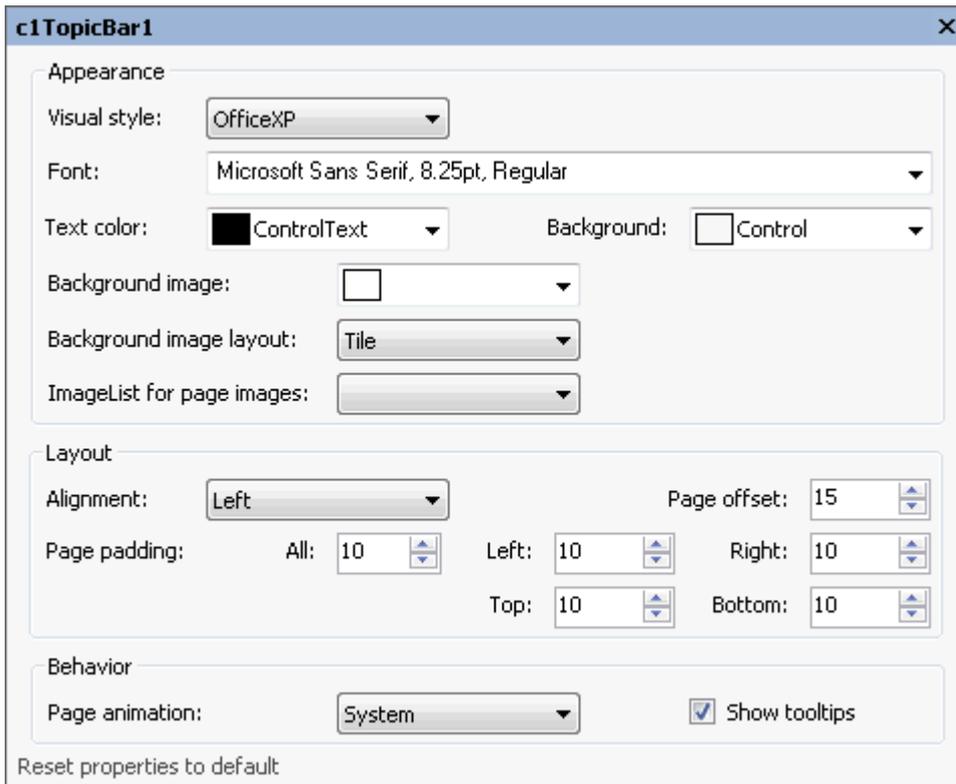
Clicking on the **Add New Page** button adds a new topic page after the existing topic page.

Edit Pages

Clicking on the **Edit Pages** button opens the **C1TopicPage Collection Editor** where you can add or remove [C1TopicPages](#) or modify their property settings.

Edit topicbar appearance and layout

Clicking on the **Edit topicbar appearance and layout** opens the **C1TopicBar properties** dialog box where you can modify the appearance, behavior, and layout styles for the [C1TopicBar](#) control.



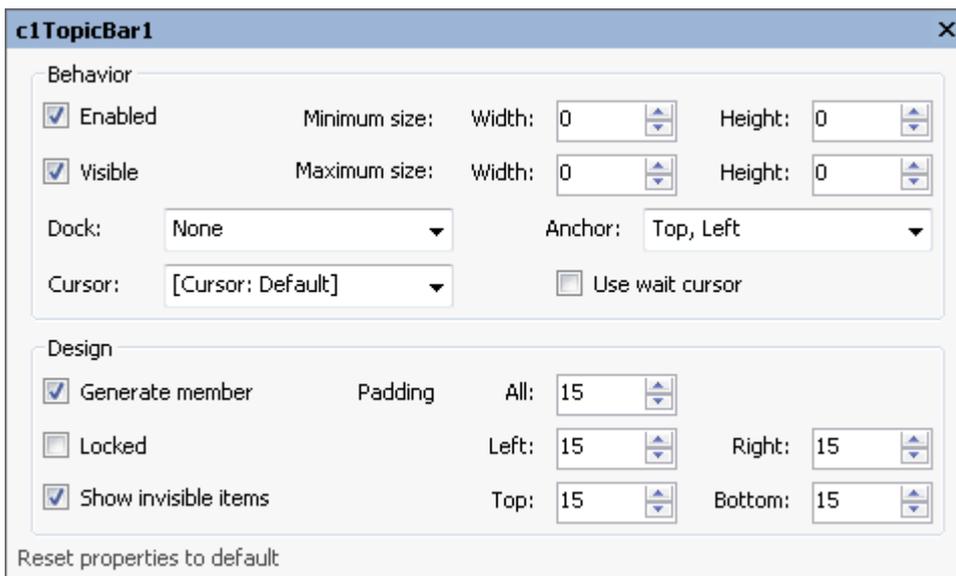
The following table defines the items included in the **C1TopicBar properties** dialog box:

Item	Description
Appearance	
Visual style	The Visual style drop-down box contains the following items for you to select from to change the style of the C1TopicBar control: Custom, System, Office2007Blue, Office2007Black, Office2007Silver, Office2003Blue, Office2003Olive, Office2003Silver, OfficeXP, Classic, and WindowsXP.
Font	The Font drop-down box opens the Font dialog box where you can modify the Font style properties for the C1TopicBar control.
Text color	The Foreground drop-down list box contains the Custom, System, and Web colors for you to select from to set the Foreground color for the C1TopicBar control.
Background	The Background drop-down list box contains the Custom, System, and Web colors for you to select from to set the back color of the highlighted item in the C1TopicBar control.
Background image	The Background image drop-down box opens an Open dialog box where you can apply the background image used for the C1TopicBar control.
Background image layout	The Background image layout drop-down box opens a list of layout items (None, Tile, Center, Stretch, and Zoom) for you to select from that gets the background image layout for the C1TopicBar control.
ImageList for page images	The ImageList for page images drop-down box gets the image list used to provide the images shown on the pages' title bars.
Layout	
Alignment	The Alignment drop-down box contains Left, Right, and Center items for you to set the

	alignment of the topicbar pages caption.
Page offset	The Page offset sets the integer value for the space between the pages.
Page padding	The Page padding sets the space between a page border and a link.
Behavior	
Page animation	The Page animation drop-down box contains the available values for you to select from to indicate whether to use animation on page collapsing/expanding.
ShowToolTips	The ShowToolTips check box indicates whether the ToolTip is shown when the mouse is over the page caption.
Reset properties to default	Selecting the Reset properties to default item resets the modified C1TopicBar properties back to their default values.

Edit miscellaneous topicbar properties

Clicking on the **Edit miscellaneous topicbar properties** button opens the **C1TopicBar** dialog box where you can modify **C1TopicBar**s miscellaneous properties.



The following table defines the items included in the **C1TopicBar** dialog box:

Item	Description
Behavior	
Enabled	The Enabled check box indicates whether the C1TopicBar will be enabled at run time.
Visible	The Visible check box indicates whether the C1TopicBar control will be shown at run time.
Minimum size	The Minimum size field includes a Width and Height NumericUpDown controls for specifying the minimum width and height size for the C1TopicBar control.
Maximum size	The Maximum size field includes a Width and Height NumericUpDown controls for specifying the maximum width and height size for the C1TopicBar control.

Dock	The Dock drop-down box contains items for you to select from to define which border of the C1TopicBar control is bound to the container.
Cursor	The Cursor drop-down box opens a list of different cursor items for you to set the type of cursor that appears when the pointer moves over the C1TopicBar control.
Use wait cursor	The Use wait cursor check box indicates whether to use wait cursor.
Anchor	The Anchor drop-down box defines the edges of the container to which the C1TopicBar control is bound. When it's anchored to an edge, the distance between the C1TopicBar 's closest edge and the specified edge will remain constant.
Design	
Generate member	The Generate member check box indicates whether to generate the member for the C1TopicBar control. (True , if selected; False , if deselected)
Locked	The Locked check box indicates whether the C1TopicBar is locked. (True , if selected; False , if deselected)
Padding	The Padding specifies the interior spacing of the C1TopicBar control.
Reset properties to default	Selecting the Reset properties to default item resets the modified C1TopicBar properties back to their default values.

C1TopicPage Toolbar

The [C1TopicPage](#) toolbar appears for the [C1TopicBar](#) control. To expose the [C1TopicPage](#) toolbar, select the [C1TopicBar](#) control, then select the topic page on the [C1TopicBar](#) and slide your cursor anywhere on the [C1TopicPage](#).

The [C1TopicPage](#) toolbar consists of one command button:

Toolbar Button	Description
	Add topic link: Adds a new topic link in the active topic page.
	Edit the topic page appearance: Opens the properties dialog box for the selected C1TopicPage .
	Delete topic page: Removes the selected C1TopicPage from the C1TopicBar .

Add new topic link

Clicking on the **Add topic link** button adds a new topic link to the topic page in the [C1TopicBar](#) control.

Edit topic page appearance

Clicking on the **Edit topic page appearance** button opens the **properties** dialog box for the selected [C1TopicPage](#).



The following table defines the items included in the **C1TopicBar properties** dialog box:

Item	Description
Page	
Text	The Text textbox displays the text name that appears on the caption. To rename the text name for the caption select the text in the Text textbox and type the desired text name.
Tooltip	The Tooltip textbox sets the ToolTip text for the selected topic page. To add a ToolTip, enter text in the Tooltip textbox.
Image index	The Image index drop-down list box displays the index value of the command image.
Special style	The Special style check box indicates whether the page has a special dark title. (True , if selected; False , if deselected).
Collapsed	The Collapsed check box indicates whether the page is collapsed or not. (True , if selected; False , if deselected).
Page is visible	The Page is visible check box indicates whether the page is visible or not. (True , if selected; False , if deselected).
Links	
Alignment	The Alignment drop-down box contains Left, Right, and Center alignment options to set the alignment for the page's links.
Wrap text in links	The Wrap text in links check box indicates whether the page's link text wraps if its length exceeds the page width. (True , if selected; False , if deselected).
Image list	The Image list sets the image list used to provide images shown in page links.
Reset properties to default	Selecting the Reset properties to default item resets the modified C1TopicPage properties back to their default values.

Delete topic page

Clicking on the **Delete topic page** button removes the selected topic page from the **C1TopicBar**.

C1ToolBar Toolbar

The **C1ToolBar** toolbar appears for the **C1ToolBar** control. To expose the **C1ToolBar** toolbar slide your cursor in the top left area of the **C1ToolBar** control. The **Open** button appears for the **C1ToolBar** toolbar.

Opening and Closing the C1ToolBar Toolbar

To open the **C1ToolBar** toolbar, click on the  button. To close the **C1ToolBar** toolbar, click on the  button.

The **C1ToolBar** toolbar consists of the following command buttons:

Toolbar Button	Description
	Add new command link/command: Adds a new command after the current command to the toolbar.
	Edit command links: Opens the C1CommandLink Collection Editor for you to edit the command links.
	Edit toolbar appearance: Opens the C1ToolBar appearance dialog box where you can set the general appearance properties for the C1ToolBar control.
	Edit toolbar layout: Opens the Layout dialog box where you can set the layout properties for the C1ToolBar control.
	Edit miscellaneous toolbar properties: Opens the Miscellaneous dialog box for the C1ToolBar control where you can apply behavior settings to the C1ToolBar control.

Add Command Links

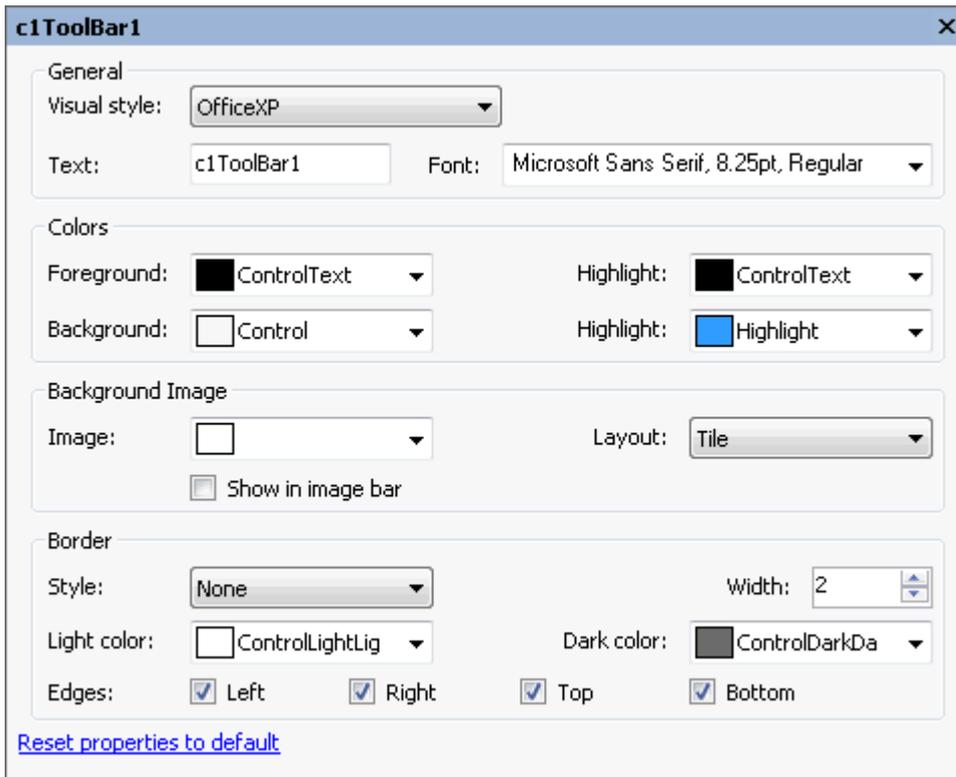
Clicking the **Add Command Links** button adds a new command after the current command. It displays the **Link to Command** designer below the new command so you can easily edit the new command without leaving the design surface.

Edit Command Links

Clicking on the **Edit Command Links** button opens the **C1CommandLink Collection Editor** where you can add or remove command links and edit the commandlink's properties.

Edit Toolbar Appearance

Clicking on the **Edit toolbar appearance** button opens the **Appearance** dialog box where you can modify the appearance properties for the **C1ToolBar** control.



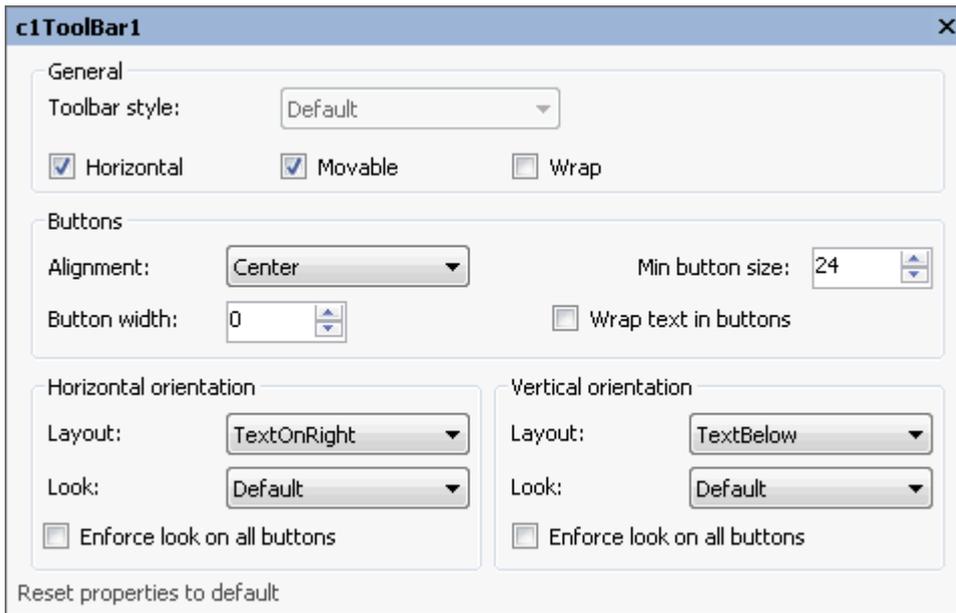
The following table defines the items included in the **Appearance** dialog box for the **C1ToolBar** control:

Item	Description
General	
Visual style	The Visual style drop-down box contains the following items for you to select from to change the style of the C1ToolBar control: Custom, System, Office2007Blue, Office2007Black, Office2007Silver, Office2003Blue, Office2003Olive, Office2003Silver, OfficeXP, Classic, and WindowsXP.
Text	The Text textbox displays the text name that appears on the selected toolbar button. To rename the text name for the toolbar button, select the text in the Text textbox and type the desired text name.
Font	The Font drop-down box opens the Font dialog box where you can modify the Font style properties for the C1ToolBar control.
Colors	
Foreground	The Foreground drop-down list box contains the Custom, System, and Web colors for you to select from to set the Foreground color for the C1ToolBar control.
Highlight (ForeHiColor)	The Highlight drop-down list box contains the Custom, System, and Web colors for you to select from to set the forecolor of the highlighted button for the C1ToolBar control.
Background	The Background drop-down list box contains the Custom, System, and Web colors for you to select from to set the back color of the highlighted button in the C1ToolBar control.

Highlight (BackHiColor)	The Background drop-down list box contains the Custom, System, and Web colors for you to select from to set the back color of the highlighted item in the C1TopicBar control.
Background image	
Image	The Image drop-down box opens an Open dialog box where you can apply the background image used for the C1ToolBar control.
Layout	The Layout drop-down box opens a list of layout items (None, Tile, Center, Stretch, and Zoom) for you to select from that gets the background image layout for the C1ToolBar control.
Show in image bar	The Show in image bar check box indicates whether to show background image in image bar when the toolbar's style is Drop-downMenu. (True , if selected; False , if deselected)
Border	
Style	The Style drop-down box includes a list of border style items for you to select from to set the border style for the C1ToolBar .
Width	The Width NumericUpDown control sets the width for the border around the C1ToolBar .
Light color	The Light color drop down box contains the Custom, System, and Web colors for you to select from to set the light color for the border around the C1ToolBar control.
Dark color	The Dark color drop down box contains the Custom, System, and Web colors for you to select from to set the dark color for the border around the C1ToolBar control.
Edges	The Edges check boxes (Left, Right, Top, and Bottom) let you specify which edge of the border to apply the border style to.
Reset properties to default	Selecting the Reset properties to default item resets the modified C1ToolBar properties back to their default values.

Edit Toolbar Layout

Clicking on the **Edit Toolbar Layout** button opens the **Layout** dialog box where you can modify the layout properties for the [C1ToolBar](#) control.



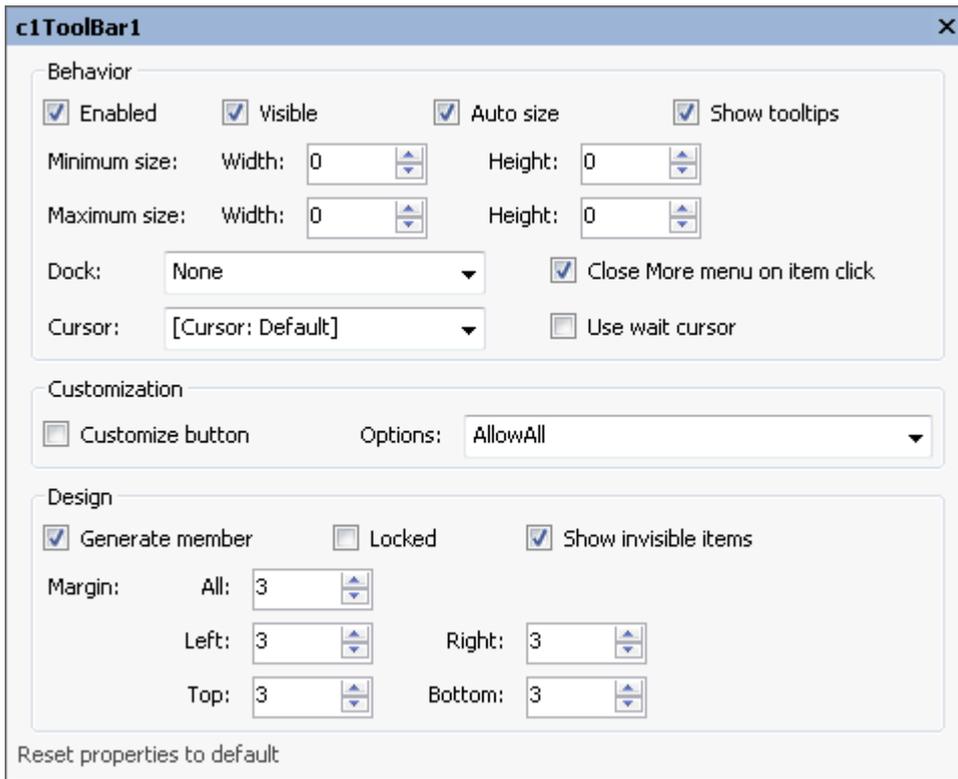
The following table defines the items included in the **Layout** dialog box for the **C1ToolBar** control:

Item	Description
General	
Toolbar style	The Toolbar style drop-down box contains the two values (Default and Drop-downMenu) of the <code>ToolBarStyle</code> property for you to select from to set the toolbar's style.
Horizontal	The Horizontal check box indicates whether to set the orientation of the C1ToolBar to horizontal. (True , if selected; False , if deselected).
Movable	The Moveable check box indicates whether the toolbar can be moved by the user. (True , if selected; False , if deselected)
Wrap	The Wrap check box indicates whether to wrap the toolbar or show a \"More...\" button if not all items fit on a single line. (True , if selected; False , if deselected)
Buttons	
Alignment	The Alignment drop-down box gets the Left, Right, or Center button alignment for vertical toolbars.
Minimum button size	The Minimum button size gets or sets the minimum size (width and height) of button in the toolbar.
Button width	The Button width gets or sets the width for all buttons (applies to horizontal toolbars only; if 0, buttons are individually sized).
Wrap text in buttons	The Wrap text in buttons check box gets or sets the value indicating whether to wrap text in links when the <code>C1ToolBar.ButtonWidth</code> is greater than zero and the text doesn't fit.
Horizontal orientation	
Layout	The Layout drop-down box contains the following members to set the <code>C1ToolBar.ButtonLayoutHorz</code> property of the C1ToolBar class: Default, Image, Text, and

	TextAndImage.
Look	The Look drop-down box contains the following members to set the C1ToolBar.ButtonLookHorz property of the C1ToolBar class: TextAbove, TextBelow, TextOnLeft, and TextOnRight.
Enforce look on all buttons	The Enforce look on all buttons check box gets the specified look for all of the buttons in horizontal orientation.
Vertical orientation	
Layout	The Layout drop-down box contains the following members to set the C1ToolBar.ButtonLayoutVert property of the C1ToolBar class: Default, Image, Text, and TextAndImage.
Look	The Look drop-down box contains the following members to set the C1ToolBar.ButtonLookVert property of the C1ToolBar class: TextAbove, TextBelow, TextOnLeft, and TextOnRight.
Enforce look on all buttons	The Enforce look on all buttons check box gets the specified look for all of the buttons in vertical orientation.
Reset properties to default	Selecting the Reset properties to default item resets the modified C1ToolBar 's layout properties back to their default values.

Edit Miscellaneous Properties

Clicking on the **Edit Miscellaneous Properties** button opens the [C1ToolBar](#) dialog box where you can edit miscellaneous properties for the [C1ToolBar](#) control.



The following table defines the fields included in the [C1ToolBar](#) dialog box for the C1ToolBar control:

Item	Description
Behavior	
Enabled	The Enabled check box indicates whether the C1ToolBar will be enabled at run time. (True , if selected; False , if deselected)
Visible	The Visible check box indicates whether the C1ToolBar control will be shown at run time. (True , if selected; False , if deselected)
AutoSize	The AutoSize check box indicates whether the toolbar automatically adjusts its size to fit all items. (True , if selected; False , if deselected)
Show tooltips	The Show tooltips check box gets or sets the value indicating whether to show ToolTip texts when the mouse cursor is over the toolbar button.
Minimum size	The Minimum size field includes a Width and Height NumericUpDown controls for specifying the minimum width and height size for the C1ToolBar control.
Maximum size	The Maximum size field includes a Width and Height NumericUpDown controls for specifying the maximum width and height size for the C1ToolBar control.
Dock	The Dock drop-down box contains items for you to select from to define which border of the C1ToolBar control is bound to the container.

Close More menu on item click	The Close More menu on item click check box gets or sets the value indicating whether the menu that is automatically created when some items do not fit on the toolbar closes when an item on that menu is clicked.
Cursor	The Cursor drop-down box opens a list of different cursor items for you to set the type of cursor that appears when the pointer moves over the C1ToolBar control.
Use wait cursor	The Use wait cursor check box indicates whether to use wait cursor.
Customization	
customize button	The customize button check box indicates whether to show the customize button. (True , if selected; False , if deselected)
Options	The Options drop-down box contains the following members to set the value of the C1ToolBar.CustomizeOptions property of the C1ToolBar class: AllowAddItem, AllowAll, AllowDelete, AllowNone, AllowRemoveItem, and AllowToggleCustomizeButton.
Design	
Generate member	The Generate member check box indicates whether to generate the member for the C1ToolBar control. (True , if selected; False , if deselected)
Locked	The Locked check box indicates whether the C1ToolBar is locked. (True , if selected; False , if deselected)
Show invisible items	The Show invisible items check box indicates whether to show invisible buttons in the C1ToolBar control. (True , if selected; False , if deselected)
Margin	The Margin specifies the space between the C1ToolBar control and another control's margin.
Reset properties to default	Selecting the Reset properties to default item resets the modified C1ToolBar 's miscellaneous properties back to their default values.

Menus and Toolbars Overview

The [C1Command](#) suite integrates menus and toolbars into a single system, allowing you to reuse the same objects and code for menu items and toolbar buttons.

The five main types of objects for building menu systems with [C1Command](#) are [C1MainMenu](#), [C1CommandMenu](#), [C1CommandControl](#), [C1CommandMdiList](#), and [C1ContextMenu](#). Whereas, the main types of objects for building toolbar systems are [C1ToolBar](#), [C1CommandMenu](#), [C1CommandControl](#), [C1CommandMdiList](#), and [C1ContextMenu](#). The only difference between the two is the two primary controls: [C1MainMenu](#) and [C1ToolBar](#).

C1MainMenu

[C1MainMenu](#) is a control that displays the main menu in a Windows forms. When you place this object on your form, it will show across the whole form at the top, as regular Windows main menus do. In addition to the main menu at the top of the form, a [C1CommandHolder](#) will automatically appear in the component tray. The [C1CommandHolder](#) stores all of the menu's commands as a single collection. For more information on how to use the [C1CommandHolder](#), please see [C1CommandHolder Component](#).

Command links of type [C1CommandLink](#) are used to represent the commands in menus.

C1RadialMenu

[C1RadialMenu](#) is a component that represents a circular context menu made up of pie slices where the hole is drawn in the center. Radial menus are common in touch screen devices so you can simply tap any of the icons that appear on the radial menu. The icon represents a command button. You can also tap an arrow to see more related options. The arrows around the outer edge of a radial menu indicates that you have more commands available.

C1ToolBar

[C1ToolBar](#) is a control which represents a toolbar. Like the [C1MainMenu](#) it contains a collection of command links stored in the [C1CommandHolder](#) component. The command links represent menu items on the main menu bar whereas the command links for [C1ToolBar](#) represent buttons on the toolbar.

The following topics provide further detail about the functionality of menus and toolbars and the common and unique objects used to create the menus and toolbar systems.

Menus and Toolbars Functionality

The functionality of a menu item and a toolbar button is very similar in [C1Command](#). The menu item or toolbar button is split between two components: a **command** and a **command link**.

Functionality of a C1Command Component

The **command** (an object of type [C1Command](#) or derived types, see [Class Hierarchy](#) for a list) is used to hold properties and event handlers related to the actual action which the command represents. Commands themselves are **not** contained in [C1Commands](#) menus and toolbars. Instead, all commands on a form are stored as a single collection in a component of type [C1CommandHolder](#), a single instance of which is automatically created on the form when you add the first [C1Commands](#) menu or toolbar to it.

For more information on how to use [C1CommandHolder](#), please see [C1CommandHolder Component](#). To represent commands in menus and toolbars, **command links** (components of type [C1CommandLink](#)) are used.

Functionality of a C1CommandLink Component

A **command link** is a small and quite simple component. Its most important property is `Command`, which points to the actual command object associated with this command link. Aside from this, a command link allows you to override some of the properties of the linked command, such as text. The visual representation of a command link depends on two factors: the command it links to and whether the link is contained in a main menu, a popup menu, or a toolbar.

Properties used to show the command link are taken from the command, for example text or image, whereas the way they are shown is determined by the container. In a main menu, only the command's text is shown and in a popup menu the image and the shortcut are also shown, and so on. Multiple command links can point to the same command, which is one of the main reasons why commands and command links are separate items.

Relationships Among Commands, Command Links, Menus and Toolbars, and Command Holder

To sum it up, the following relationships exist between commands, command links, menus and toolbars, and the command holder on a form:

- Commands (class `C1Command` and derived classes) are automatically stored in the form's command holder (object of type `C1CommandHolder`).
- Menus and toolbars (objects of type `C1MainMenu`, `C1CommandMenu`, `C1ContextMenu`, `C1ToolBar`) contain command links (type `C1CommandLink`) which represent menu items or toolbar buttons. Each command link points to the actual command in the command holder. Command links are stored in the **CommandLinks** collection of a menu or a toolbar. Command links can be edited via this collection or using the designer.
- Multiple command links can point to the same command. And command links pointing to the same command can be inside different containers. For example, a link from the File menu and another from the File Operations toolbar can point to the same file open command.
- Most properties of a command link visible to the user (text, image, and so on.) are normally taken from the linked command. The shown state of the command link (enabled/disabled, checked/unchecked and so on.) is also determined by the corresponding state of the linked command (command links do not have state properties).
- Most importantly, event handlers that actually perform user-defined actions (for example, opening a file or copying to the clipboard) are associated only with **commands** and never with **command links**. When a menu item is selected or a toolbar button is clicked by the user, the click event handler of the linked command is invoked.
- To enumerate all commands defined on a form, use the **Commands** collection of the command holder (which shows up in the component tray of the form). You can also use the collection editor to add or remove commands (although an easier approach is probably to use the menu or toolbar designer, accessing commands via their links).

Common Objects Used to Create Menus and Toolbars

C1Command's Menus and ToolBars use the following objects to create menu or toolbar systems:

- `C1CommandHolder` component
- `C1CommandMenu` command
- `C1ContextMenu` control
- `C1CommandControl` command
- `C1CommandMdiList` command

The following section introduces each command or component used in creating menus and toolbars.

CommandHolder Component

[C1CommandHolder](#) is a container for commands. It may also contain an image list for commands' images, and a few other general settings. Only one [C1CommandHolder](#) can be placed on a form. Whenever a [C1Command](#) is created, it is always added to the form's command holder. If no commands exist, it will be automatically created by the command's designer.

Command holder provides the following features:

- It is also an [IExtenderProvider](#) providing a [C1ContextMenu](#) property to all controls on the form.
- Provides idle-time automatic update of commands' status such as visible, enabled, checked and so on

CommandMenu Command

The **C1CommandMenu** component is a command (derives from the [C1Command](#) base class) that is a menu. In addition to other [C1Command](#) properties, it contains a collection of command links which are the menu items of this menu. [C1CommandMenu](#) can be included in another menu as a sub-menu.

When a new [C1CommandMenu](#) is created, an empty command link is automatically added to it in the same way as an empty command link was automatically added to the new main menu.

For more information about using the **C1CommandMenu** command, please see [Menu Tasks](#).

ContextMenu Control

The **C1ContextMenu** component is a menu (it derives from the [C1CommandMenu](#) base class) that can be attached to an arbitrary control as a context menu. To facilitate this, the [C1CommandHolder](#) (which always exists on a form using C1 menus) is an [IExtenderProvider](#) providing a **C1ContextMenu** property of the type **C1ContextMenu** to all controls on the form.

Note that a **C1ContextMenu** can be used in other menus in exactly the same way as its base class **C1CommandMenu**. Thus if you want to use the same menu as a submenu in the main menu system and as a context menu, just link [C1CommandLink](#) to the same **C1ContextMenu** in both places.

For more information about using the **C1ContextMenu**, please see [Context Menu Tasks](#).

CommandControl Command

[C1CommandControl](#) is a command which can be associated with an arbitrary control. This functionality is provided by the class **C1CommandControl**, derived from [C1Command](#). Controls can be dragged from the Visual Studio Toolbox and dropped onto a [C1MainMenu](#) or [C1ToolBar](#). This automatically creates a **C1CommandControl**, connects it to the dropped control, and adds a link to the new command to the toolbar. This command allows at most one command link to be connected to it.



Note: A small control can be added inside it, but it does not handle large controls such as containers.

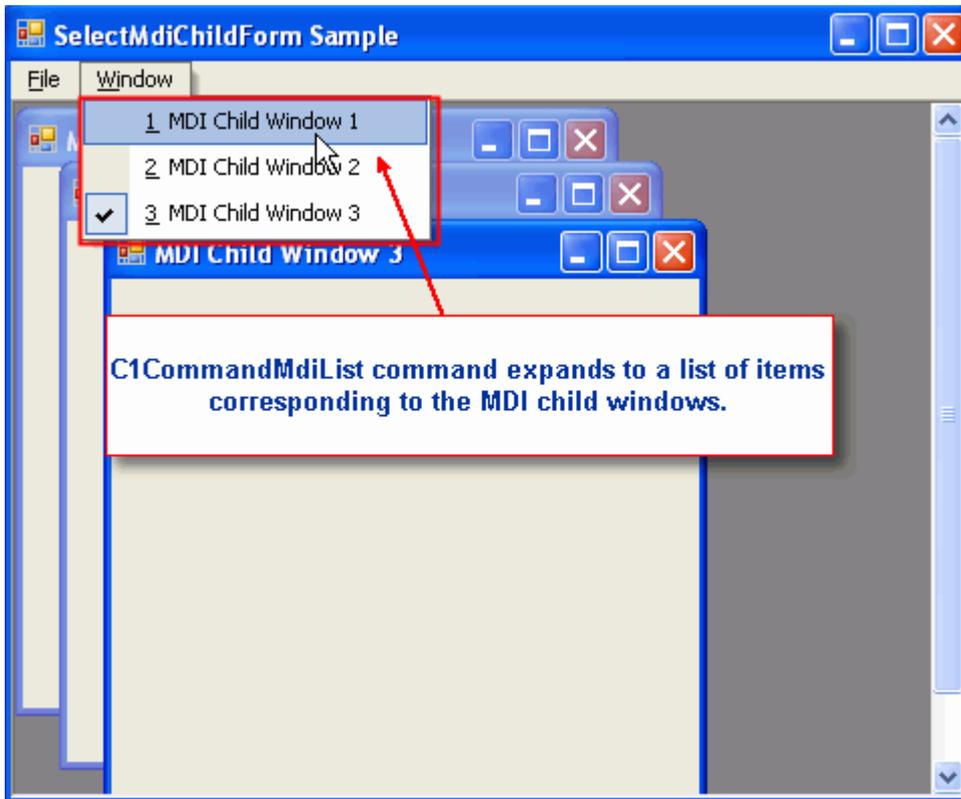
For more information about using [C1CommandControl](#), see [Adding an Arbitrary Control to the Toolbar](#).

CommandMdiList Command

[C1CommandMdiList](#) component is a command which, at run time, expands to a list of items corresponding to the MDI child windows of the current window. Note that this is not a submenu. You can either put this command in a submenu all by itself, or add other menu items before or after it.

The following image shows how the [C1CommandMdiList](#) displays a list of items corresponding to the MDI child

windows.



You can restrict the amount of items the [C1CommandMdiList](#) command displays in its Menu's list by setting the [MaxItems](#) to the desired amount of items you would like to show. The default value for this property is 10.

You can also show hidden MDI windows in the menu's list by setting [ListHidden](#) to **True**.

For more information on how to accomplish creating a Window list for MDI child windows, see [Creating a Window List for an MDI Form](#).

Unique Objects Among Menus and Toolbars

Menus and ToolBars share many objects, however, there are two distinct components among them. The menus have a [C1MainMenu](#) control which is the main menu and the toolbars have a [C1ToolBar](#) control which represents the toolbar.

The following section introduces the [C1MainMenu](#) and [C1ToolBar](#) controls and provides further information about their appearance and behavior properties.

MainMenu Control

[C1MainMenu](#) is a control that displays the main menu in a Windows form. When you place this object on your form, it will show across the whole form at the top, as regular Windows main menus do. In addition to the main menu at the top of the form, a [C1CommandHolder](#) will automatically appear in the component tray. The [C1CommandHolder](#) stores all of the menu's commands as a single collection. Only one [C1MainMenu](#) control can be added to a form.

To add the C1MainMenu control at design-time:

In the Visual Studio Toolbox, double-click on the [C1MainMenu](#) component or drag and drop it onto the form.

To add the C1MainMenu control programmatically:

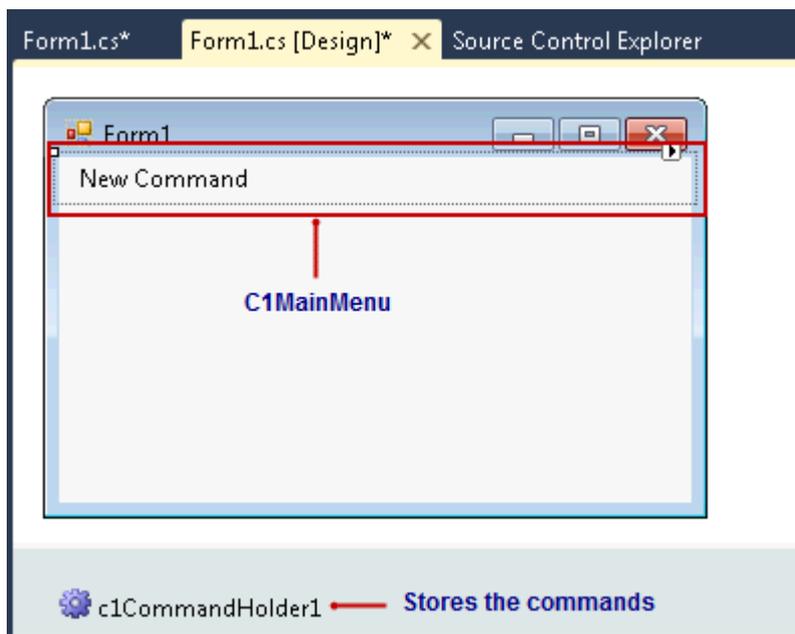
To write code in Visual Basic

```
Visual Basic
Imports Cl.Win.C1Command
Dim ch As C1CommandHolder (Me)
Dim mm As New C1MainMenu
Me.Controls.Add (mm)
```

To write code in C#

```
C#
using Cl.Win.C1Command
C1CommandHolder.CreateCommandHolder (this);
C1MainMenu mm = new C1MainMenu ();
this.Controls.Add (mm);
```

The following screen shot depicts a [C1MainMenu](#) control once it's been added to the form:



The [C1MainMenu](#) control includes a **Link to Command** designer that conveniently allows you to visually configure the menus.

Note: This editor is available for all [C1CommandLinks](#); therefore, you can easily edit all command links for any of the objects: [C1ContextMenu](#), [C1ToolBar](#), and [C1OutBar](#).

For more information about the elements in the **Link to Command** designer see [Link to Command Designer](#).

For more information that shows how to use the **C1MainMenu** control for specific tasks, see [Menu Tasks](#).

ToolBar Control

The **C1ToolBar** control is used on forms as a toolbar. When you place this object on your form, like the **C1MainMenu**, a **C1CommandHolder** will automatically appear in the component tray. The **C1CommandHolder** stores all of the command links as a single collection. The command links represent menu items on the main menu bar whereas the command links for **C1ToolBar** represent buttons on the toolbar.

Once the component **C1ToolBar** is added to the form, the **Link to Command** designer allows you to set up the toolbar system. **C1ToolBar** and **C1MainMenu** both use the same **Link to Command** designer. For more information about the interface for the **Link to Command** designer, see [Link to Command Designer](#).

The **C1ToolBar** provides two different types of toolbars: a default toolbar and a drop-down style toolbar. The toolbar buttons provide drop-down buttons for a drop-down menu. The buttons can be arranged vertically or horizontally on the toolbar depending on the orientation of the toolbar.

For more information on how to use the **C1ToolBar** control to do specific tasks such as wrapping text in the toolbar button, see [ToolBar Tasks](#).

Menus Appearance and Behavior

Menus provide a number of useful properties to control the behavior and appearance of the main menu and menu items.

C1Command's menus include a variety of appearance properties to visually enhance and customize the control. The menu's style, size, and layout can easily be customized by using the **C1MainMenu's** appearance properties. These properties can be set at design time through the Properties window or programmatically.

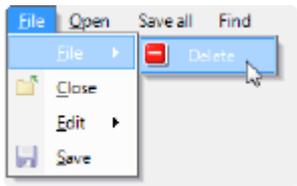
C1Command's menus also include several useful behavioral properties for wrapping, merging, and showing ToolTips in menu items.

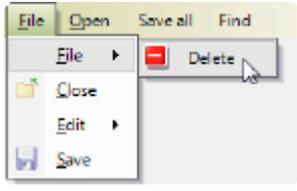
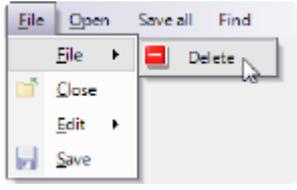
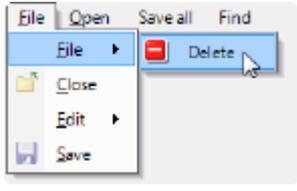
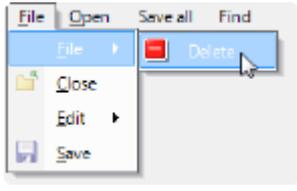
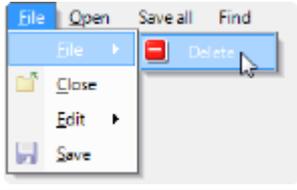
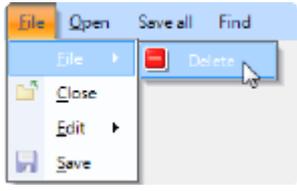
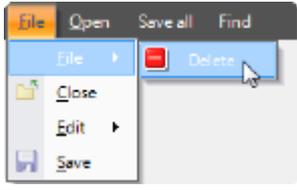
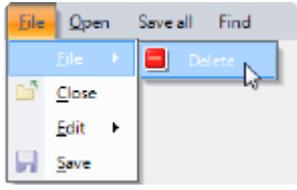
The following section introduces some of the common appearance and behavior properties used for the **C1MainMenu** control.

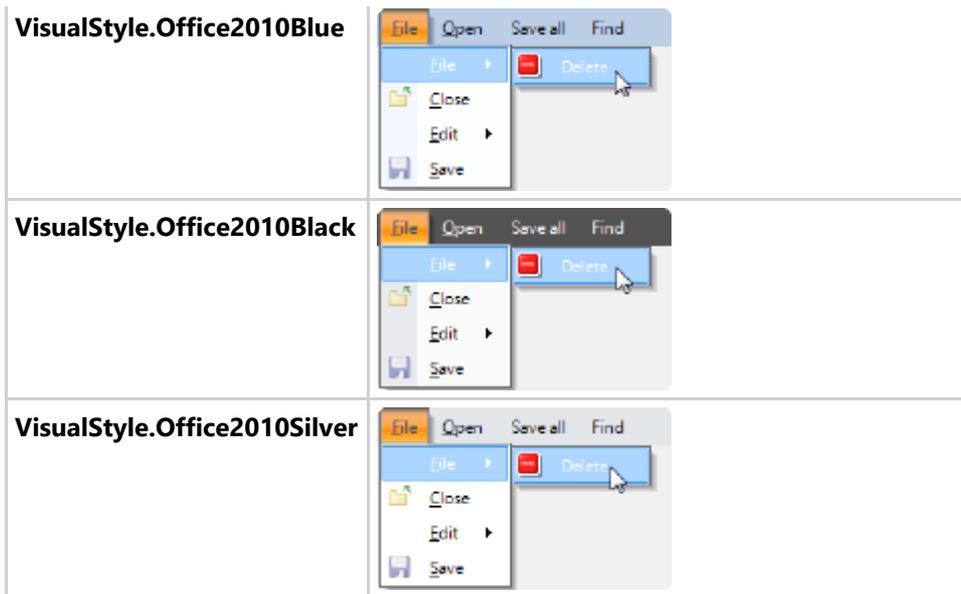
Menus Visual Styles

The **C1MainMenu** and **C1ContextMenu** controls provide several built-in styles, such as **Custom**, **System**, **Office2010Blue**, **Office2010Black**, **Office2010Silver**, **Office2007Blue**, **Office2007Black**, **Office2007Silver**, **Office2003Blue**, **Office2003Olive**, **Office2003Silver**, **OfficeXP**, **Classic**, and **WindowsXP** that can be easily applied using the controls' **VisualStyle** properties.

The following table illustrates each style of the **C1MainMenu** control. The **C1ContextMenu** control's visual styles are identical to the **C1MainMenu** control's, only the **C1ContextMenu** control doesn't contain the menu bar.

Property Setting	Image
VisualStyle.Custom	[Custom allows you to customize the control's visual style.]
VisualStyle.System	

VisualStyle.Office2003Blue	
VisualStyle.Office2003Olive	
VisualStyle.Office2003Silver	
VisualStyle.OfficeXP	
VisualStyle.Classic	
VisualStyle.WindowsXP	
VisualStyle.Office2007Blue	
VisualStyle.Office2007Black	
VisualStyle.Office2007Silver	



Look and Feel of Menu Items

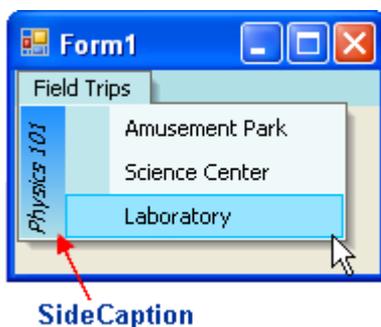
C1Command supports various settings for font style, border style, back color, and mouse-over styles for the menu items.

You can set the text name for the toolbar button/command either in the command object as well as in its command link. However, the **C1CommandLink.Text** property overrides the **C1Command.Text** property.

Special Side Caption Styles in Menus

C1Command has a special **SideCaption** property for its **C1CommandMenu** command type. With this property, you can display a side caption along the sub menu items for a particular **C1CommandMenu**. You can display text or an image inside the caption. In addition to having text or an image in the side caption, you can also customize the appearance and layout of the caption.

The following image shows a vertical side caption for the **Field Trips** menu.



For information on how to apply a side caption to your menu, see [Creating a Side Caption for a Command Menu](#).

Mouse-Over Styles in Menu Items

You can apply mouse over techniques to the menu items to improve your menu interaction with users.

The [C1MainMenu](#) component has two special properties for applying mouse-over techniques. The [BackHiColor](#) property gets the back color of the menu item when you hover your mouse over it and the [ForeHiColor](#) gets the fore color of the menu item when you hover your mouse over it.

For more information about how to use these properties, see [Modifying the Appearance of the Menus](#).

Merging Menus

In some cases, when you need to merge a MDI child window with a MDI parent menu you can enable the [CanMerge](#) property. You can also specify the type of behavior for the merging menu with its [MergeType](#) property. You can determine whether to add, replace, remove, or merge menu items with the [MergeType](#) property. The [MergeItems](#) causes the command links on the menus to be merged.

The command links for both menu items and toolbar buttons have a [MergeOrder](#) property which can be used to determine the order of the merged menu items or toolbar buttons.

For more information about how to accomplish merging menus see, [Merging Menu Items](#).

Layout and Text Wrapping in Menus

[C1MainMenu](#) has an automatic layout. The menu items are automatically sized. [C1MainMenu](#)'s [Wrap](#) property enables line-wrapping in the main menu bar. If there are too many items on the main menu bar to fit onto one line it will be wrapped.

ToolTips in Menus

A ToolTip is used to display text when the mouse hovers over the control. [C1MainMenu](#) provides a [ShowToolTips](#) property that displays the value of the [Text](#) property as a ToolTip for each menu item. This property is enabled by default.

If you would like to enter custom text for the ToolTip of each menu item you can through setting the [ShowTextAsToolTip](#) to **False**, and then setting custom text for the [ToolTipText](#) property.

For more information how to use the ToolTips, see [Displaying ToolTips for Menus and Toolbars](#).

Toolbars Appearance and Behavior

[C1ToolBar](#) provides a number of useful properties to control the behavior and appearance of the toolbars and toolbar buttons.

[C1ToolBar](#) includes a variety of appearance properties to visually enhance and customize the control. The toolbar's style, size, and layout can easily be customized by using the [C1ToolBar](#)'s appearance properties. These properties can be set at design time through the Properties window or programmatically.

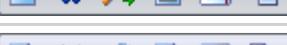
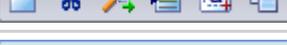
In addition to properties for setting the toolbar's appearance, [C1ToolBar](#) has several useful behavioral properties for docking and floating toolbars, moving toolbar buttons, embedding arbitrary controls to toolbars, customizing toolbars at run time, setting button layout for horizontal or vertical toolbars, showing ToolTips on the toolbar and/or its command buttons, and wrapping text in the toolbar buttons.

The following section introduces some of the common appearance and behavior properties used for the [C1ToolBar](#) control.

Toolbar Visual Styles

The [C1ToolBar](#) control provides several built-in styles, such as **Custom**, **System**, **Office2010Blue**, **Office2010Black**, **Office2010Silver**, **Office2007Blue**, **Office2007Black**, **Office2007Silver**, **Office2003Blue**, **Office2003Olive**, **Office2003Silver**, **OfficeXP**, **Classic**, and **WindowsXP** that can be easily applied using the [VisualStyle](#) property.

The following table illustrates each of the [C1ToolBar](#) control's visual styles.

Property Setting	Image
VisualStyle.Custom	[Custom allows you to customize the control's visual style.]
VisualStyle.System	
VisualStyle.Office2003Blue	
VisualStyle.Office2003Olive	
VisualStyle.Office2003Silver	
VisualStyle.OfficeXP	
VisualStyle.Classic	
VisualStyle.WindowsXP	
VisualStyle.Office2007Blue	
VisualStyle.Office2007Black	
VisualStyle.Office2007Silver	
VisualStyle.Office2010Blue	
VisualStyle.Office2010Black	
VisualStyle.Office2010Silver	

Look and Feel of Toolbars

C1Command supports various settings for font style, border style, back color, and mouse-over styles for the toolbar and its buttons.

You can set the text name for the toolbar button/command either in the command object as well as in its command link. However, the **C1CommandLink**. [Text](#) property overrides the **C1Command**. [Text](#) property.

For more information about using the general appearance properties, see [Modifying the Appearance of the Toolbar](#).

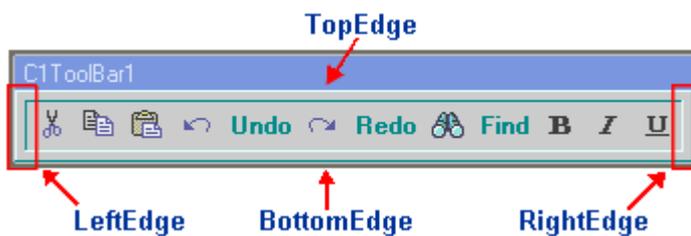
Special Border Styles in Toolbars

C1ToolBar has a special class, [C1Border](#), that allows you to add various border styles to the toolbars. C1Border contains the following members:

Name	Description
BottomEdge	Determines whether the border has a bottom edge.

DarkColor	Gets or sets the color of the group. In the Flat setting for the Style this color applies to the top, bottom, left, and right edges of the toolbar.
LeftEdge	Determines whether the border has a left edge.
LightColor	Gets or sets the color of the border. This color is not used in the Flat setting for the C1Border
RightEdge	Determines whether the border has a right edge.
Style	Gets or sets the border style.
TopEdge	Determines whether the border has a top edge.
Width	Gets or sets the border width in pixels.

The following image illustrates the TopEdge, LeftEdge, BottomEdge, and RightEdge properties for the **C1Border** class.



The TopEdge, LeftEdge, BottomEdge, and RightEdge properties are useful for applying borders to specific areas such as the top, bottom, left, or right edge of the **C1ToolBar**. These properties are set to **True** by default.

The following table illustrates each of the property settings for [Style](#) property. In addition to the various border styles shown below, the table also illustrates the **Width**, **DarkColor**, **LeftEdge**, **RightEdge**, **BottomEdge**, and **TopEdge** properties. The [Width](#) property is set to 5 pixels, [DarkColor](#) property is set to **DarkTurquoise**, [LeftEdge](#) property is set to PaleTurquoise, and the [LeftEdge](#), [RightEdge](#), [BottomEdge](#), and [TopEdge](#) are all set to **True**.

Property Setting	Image
Style.None	
Style.Flat	
Style.Groove	
Style.Ridge	
Style.Inset	



The following table illustrates the effect of the `LeftEdge`, `RightEdge`, `BottomEdge`, and `TopEdge` when each one is disabled:

Property Setting	Image
BottomEdge.False	
LeftEdge.False	
RightEdge.False	
TopEdge.False	

For more information about using these properties, see [Modifying the Appearance of the Toolbar](#).

Mouse-Over Styles in Toolbar Buttons

You can apply mouse over techniques to the toolbar buttons to improve your toolbar interaction with users.

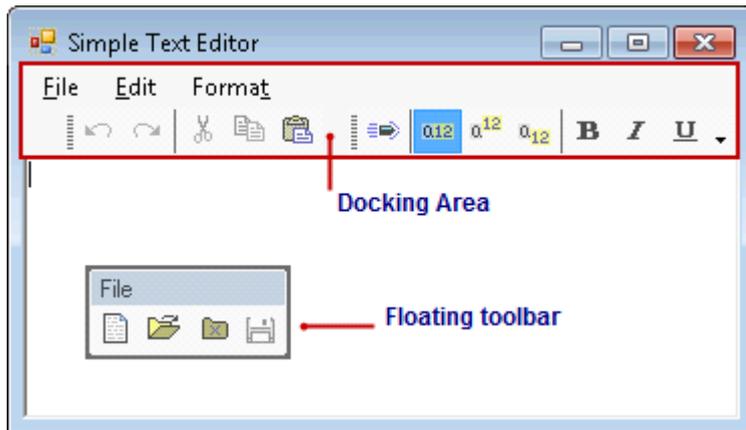
The `C1ToolBar` component has two special properties for applying mouse-over techniques. The `BackHiColor` property gets the back color of the toolbar button when you hover your mouse over it and the `ForeHiColor` gets the fore color of the toolbar button when you hover your mouse over it.

For more information about using these properties, see [Modifying the Appearance of the Toolbar](#).

Docking and Floating Toolbars

Toolbars can be docked to the top, left, right or bottom on the container that the `C1CommandDock` has been assigned to.

Each `C1ToolBar` resides inside the docking area when it is docked. Toolbars can be moved to different docking areas by using a drag-and-drop operation, and they can also be resized.



If you are creating a [C1ToolBar](#) programmatically and would like to use the [C1CommandDock](#) to enable docking and floating behavior you would add the toolbar to the [C1CommandDock](#) like the following:

To write code in Visual Basic

```
Visual Basic
Me.C1CommandDock = New C1.Win.C1Command.C1CommandDock()
Me.C1CommandDock.Controls.Add(Me.C1ToolBar1)
Me.Controls.Add(Me.C1CommandDock)
```

To write code in C#

```
C#
this.c1CommandDock = new C1.Win.C1Command.C1CommandDock();
this.c1CommandDock.Controls.Add(this.c1ToolBar1);
this.Controls.Add(this.c1CommandDock);
```

Embedded Controls in Toolbars

The [C1CommandControl](#) lets you embed arbitrary controls to the toolbar.

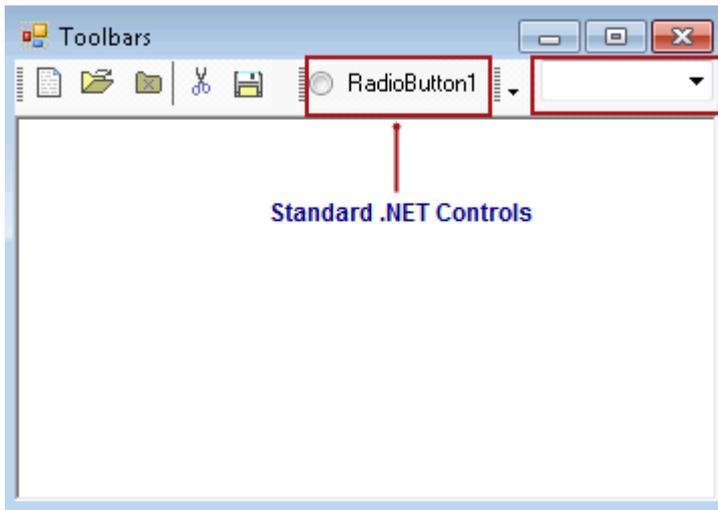
Arbitrary controls such as a textbox can be embedded in a [C1ToolBar](#) through the use of the [C1CommandControl](#).

This can simply be done by dragging an arbitrary control on to the toolbar, adding a [C1CommandControl](#) command type through the designer, or adding a [C1CommandControl](#) command type programmatically.

For more information about how to embed an arbitrary control into the [C1ToolBar](#) object, see [Adding an Arbitrary Control to the Toolbar](#).

When an arbitrary control is dragged to the toolbar it automatically creates a new command type called [C1CommandControl](#). The [C1CommandControl](#) includes a [Control](#) property which gets the arbitrary control attached to the command.

The following image shows a **RadioButton**, **CheckBox**, and a **ComboBox** control embedded into the [C1ToolBar](#).

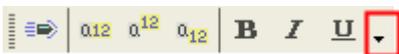


Run-Time Customization for Toolbars

C1ToolBars can be customizable at run time by setting the [CustomizeButton](#) property to **True** at design time.

Note: The toolbar needs to be placed inside a **C1CommandDock** before you set its [CustomizeButton](#) property to **True** at design-time.

When the customization is enabled a drop-down arrow appears on the toolbar at design time.



The pop-up menu appears at run time when you click on the drop-down arrow.



The Customize toolbars menu operates as follows:

Add or Remove Buttons

Clicking on a command item from the menu removes the command button from the toolbar.

Reset

Clicking on the Reset menu item resets the toolbar back to its original setting.

Customize

Clicking on Customize menu item opens the **Customize toolbars** dialog box.

The Customize Dialog contains three tabs for modifying the **C1ToolBar** component:

- **Toolbars** – This tab contains options for creating, renaming, deleting, and modifying the **C1ToolBar** component.
- **Commands** – This tab contains options for adding existing commands to the toolbars.

- **Options** – This tab contains options for modifying the **C1ToolBars** general appearance properties such as its look and feel and its font and color.

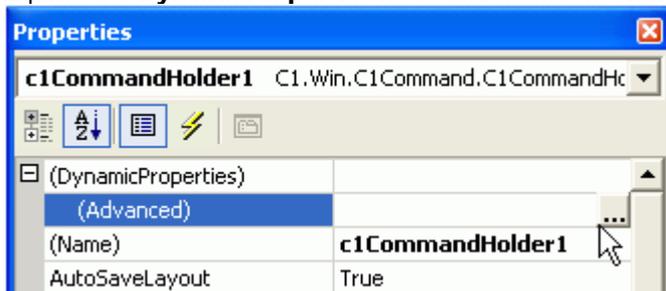
On the bottom of each tab in the **Customize toolbars** dialog box, there is a **Save, Restore, Reset, OK,** and **Cancel** command button which can be used to save the updated settings of the toolbar, restore the update settings, reset the default settings, accept the new settings, and cancel the Customize toolbars respectively.

For the end-user customizations to be persisted in the application config file, command holder's Environment property must be added to dynamic properties.

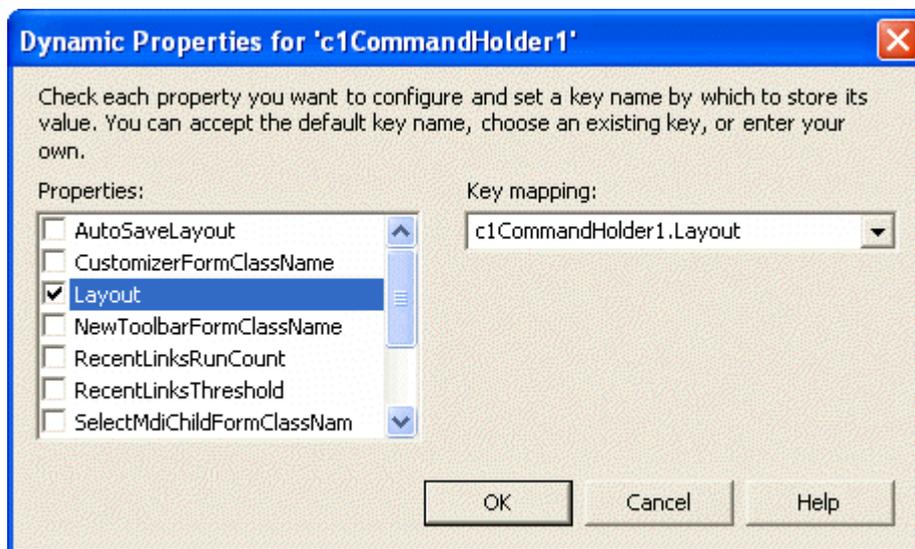
 **Note:** The user interface for dynamic properties has been removed from Visual Studio 2005. It still supports the dynamic properties. For more information about using the dynamic properties, please see the following topic in Microsoft Visual Studio 2005 documentation: [Introduction to Dynamic Properties \(Visual Studio\)](#).

To Save the Layout in the Application's .Config File:

1. Click on the **C1CommandHolder** in the form's Component Tray.
2. Expand the **DynamicProperties** node and then click on the **ellipsis** button next to the **Advanced** property.



The **Dynamic Properties** dialog box appears.



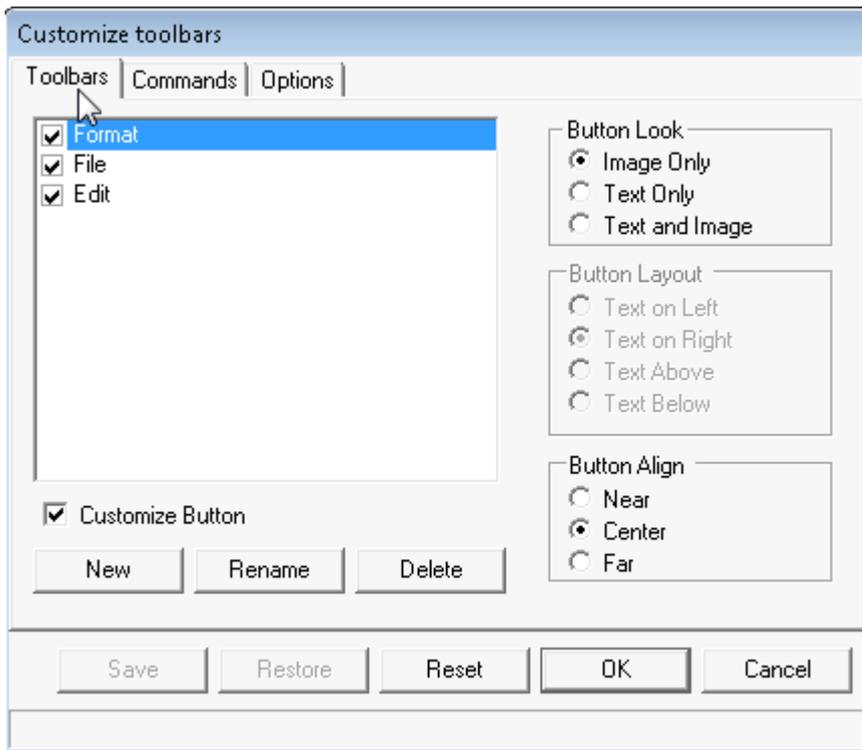
3. Click the **Layout** check box and then click **OK**. This will make Layout saved in the application's .config file instead of in the form's code.

 **Note:** When you run your program from the Visual Studio's designer, Visual Studio creates an app.config file in the project directory, and then on each run replaces the actual application's .config file (located in the bin directory) with that app.config file's contents. As a result, if you run the project in Visual Studio, change the toolbars layout, close it and then run it again, you won't see the last layout restored. This is not a bug, everything works fine when the application is not run from Visual Studio.

In addition to saving your toolbar layout using the form's dynamic properties you can also use your own scheme for saving and restoring the toolbars layout. For finer control, save and set the value of the [Layout](#) property in your code instead.

Toolbars

The **Toolbars** tab contains options for creating and manipulating toolbars.



By default, the [ButtonLook](#) and the [CustomizeButton](#) properties are disabled.

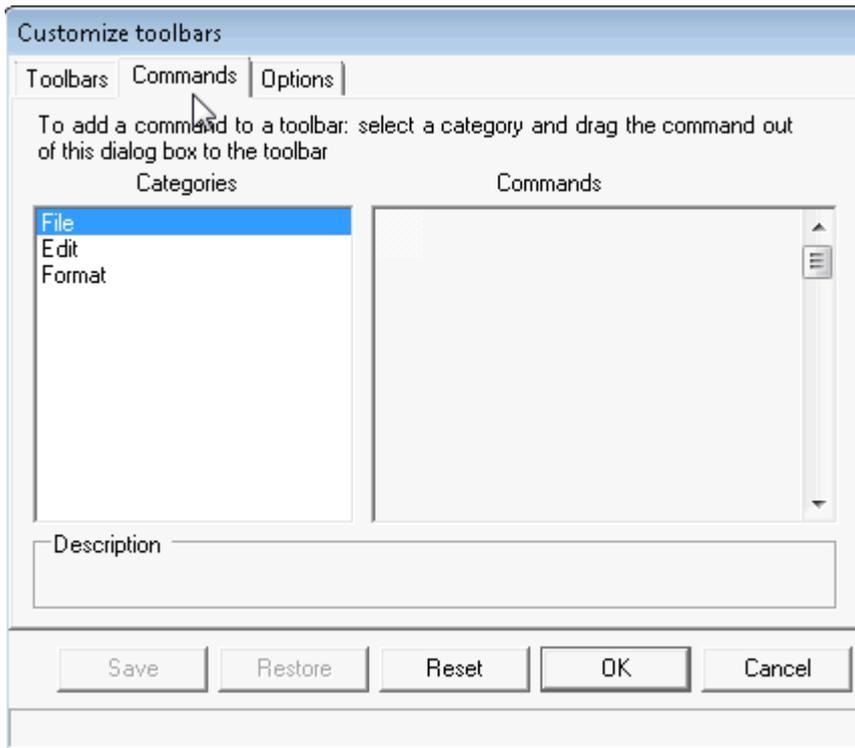
The Button Layout properties are enabled when the [ButtonLook](#) property is set to **Text and Image**(the **Text and Image** radio button is selected). This is because the [ButtonLayoutHorz](#) property determines how the text is placed by the image (above, below, to the left, or to the right of the image).

The [CustomizeButton](#) is enabled when a new [C1ToolBar](#) is added to the dialog box.

Commands

The **Commands** tab contains two list boxes: **Categories** and **Commands**. The **Categories** list box contains the categories for all of the commands. The **Commands** list box contains all of the commands for each category.

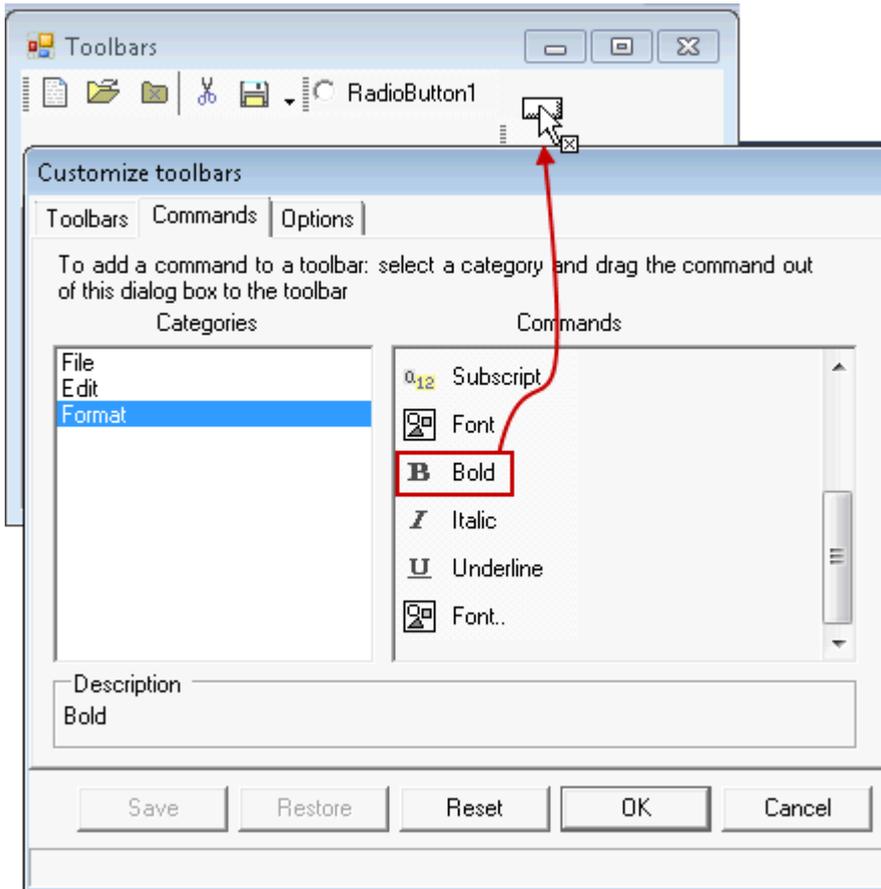
 **Note:** The Categories list box appears empty if the [Category](#) property is not set for the commands.



Commands can be easily added to the toolbars by doing either of the following:

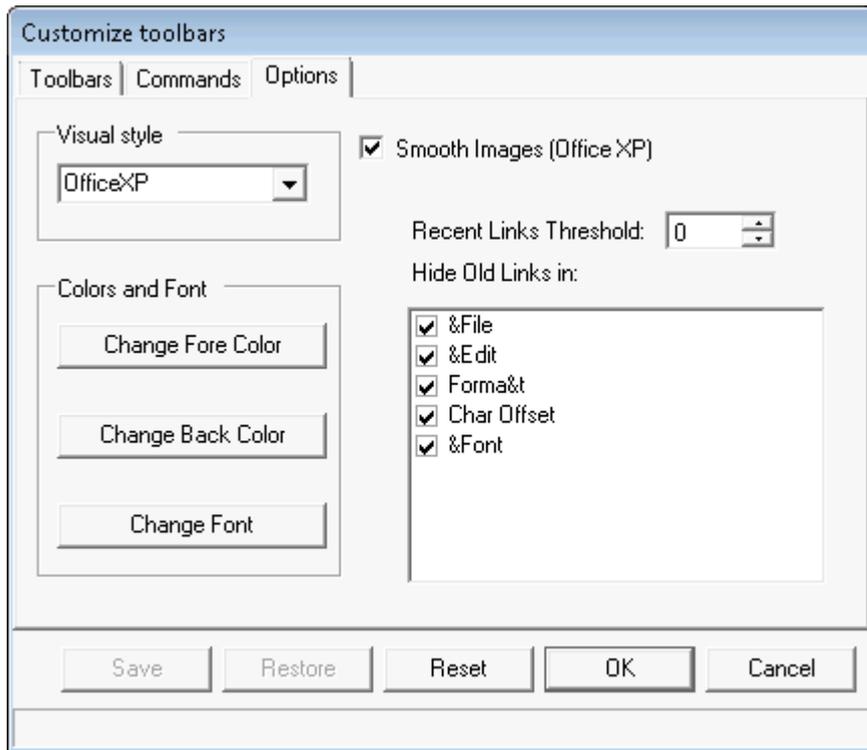
- Selecting a category from the **Categories** list box.
- Selecting a command from the **Commands** list box and then dragging it to the desired toolbar.

The following image illustrates a command being dragged from the Commands list to the Format toolbar on the form at run time.



Options

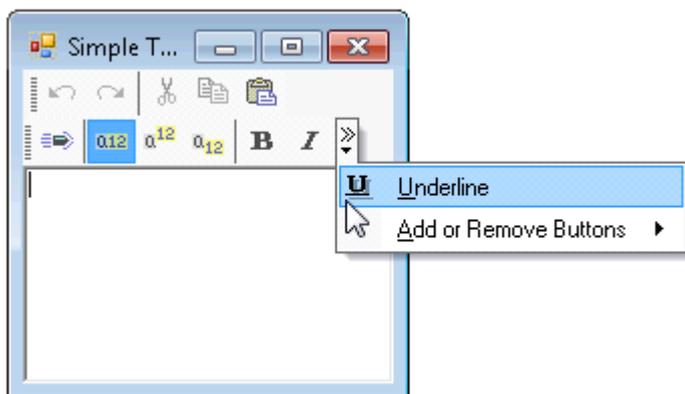
The **Options** tab contains options for modifying **C1ToolBar**'s general appearance properties such as its look and feel and its colors and font.



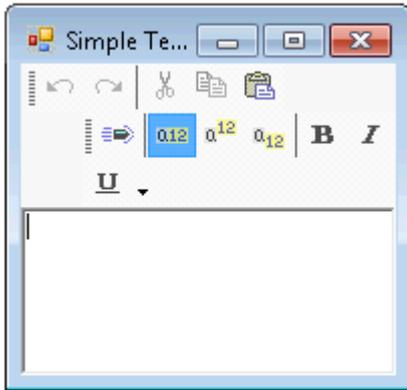
Wrapping Toolbar Buttons and Text

[C1ToolBar](#) provides wrapping ability for toolbar buttons as well as wrapping text in the toolbar buttons. The [Wrap](#) property wraps the toolbar to another line so all of its toolbar buttons appear. By default, this property is enabled.

The following image shows how the toolbar buttons appear when its [Wrap](#) property set to **False**.

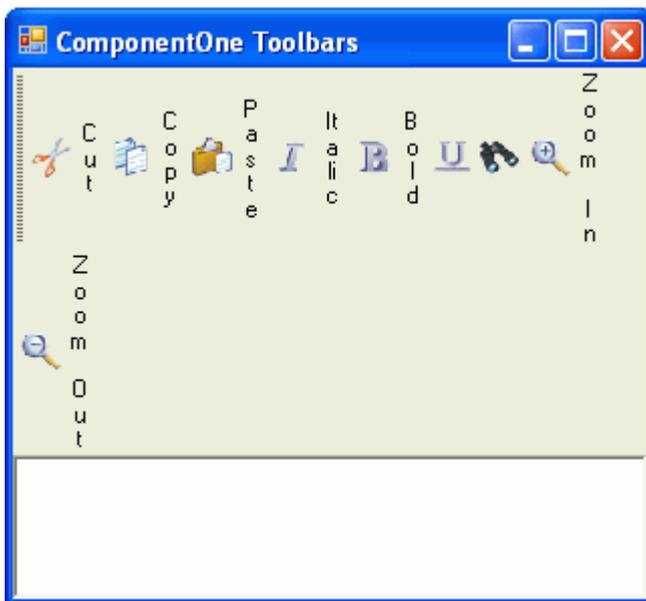


The following image shows how the toolbar buttons appear when its [Wrap](#) property set to **True**.



The following image shows how the toolbar buttons appear when its [Wrap](#) is set to **True** and its [WrapText](#) properties is set to **False**.

The following image shows how the toolbar buttons appear when their [Wrap](#) and [WrapText](#) properties are set to **True**.



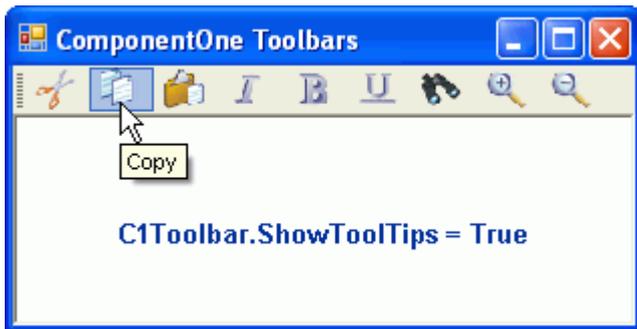
ToolTips in Toolbars

A ToolTip is used to display text when the mouse hovers over the control. [C1ToolBar](#) provides a [ShowToolTips](#) property that displays the value of the [Text](#) property as a ToolTip for each toolbar button. This property is enabled by default.

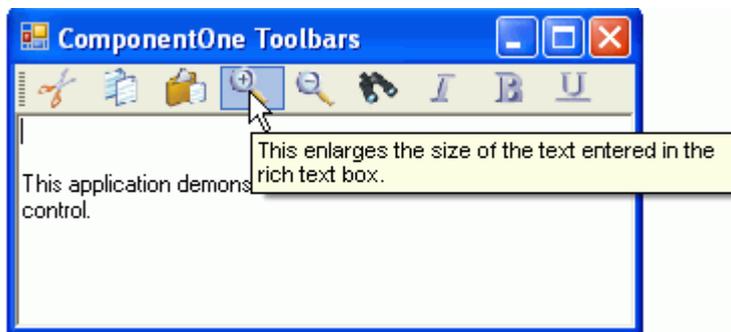
Note: If you have the [C1CommandLink.Text](#) property set for the button, but not its [C1Command.Text](#) property the ToolTip will get its default [C1Command.Text](#) name. For example, if it's the first button the tooltip and [C1Command.Text](#) name would be Button1.

If you would like to enter custom text for the ToolTip of each toolbar button you can through the [ToolTipText](#) property.

The following image depicts a toolbar that has its [ShowToolTips](#) property set to **True**.



The following image shows a toolbar that has its `ShowToolTips` property set to **True** and custom text entered for fourth `C1CommandLink`'s `ToolTipText` property.



For more information about using ToolTips, see [Displaying ToolTips for Menus and Toolbars](#).

Toolbar and Button Layout Behavior

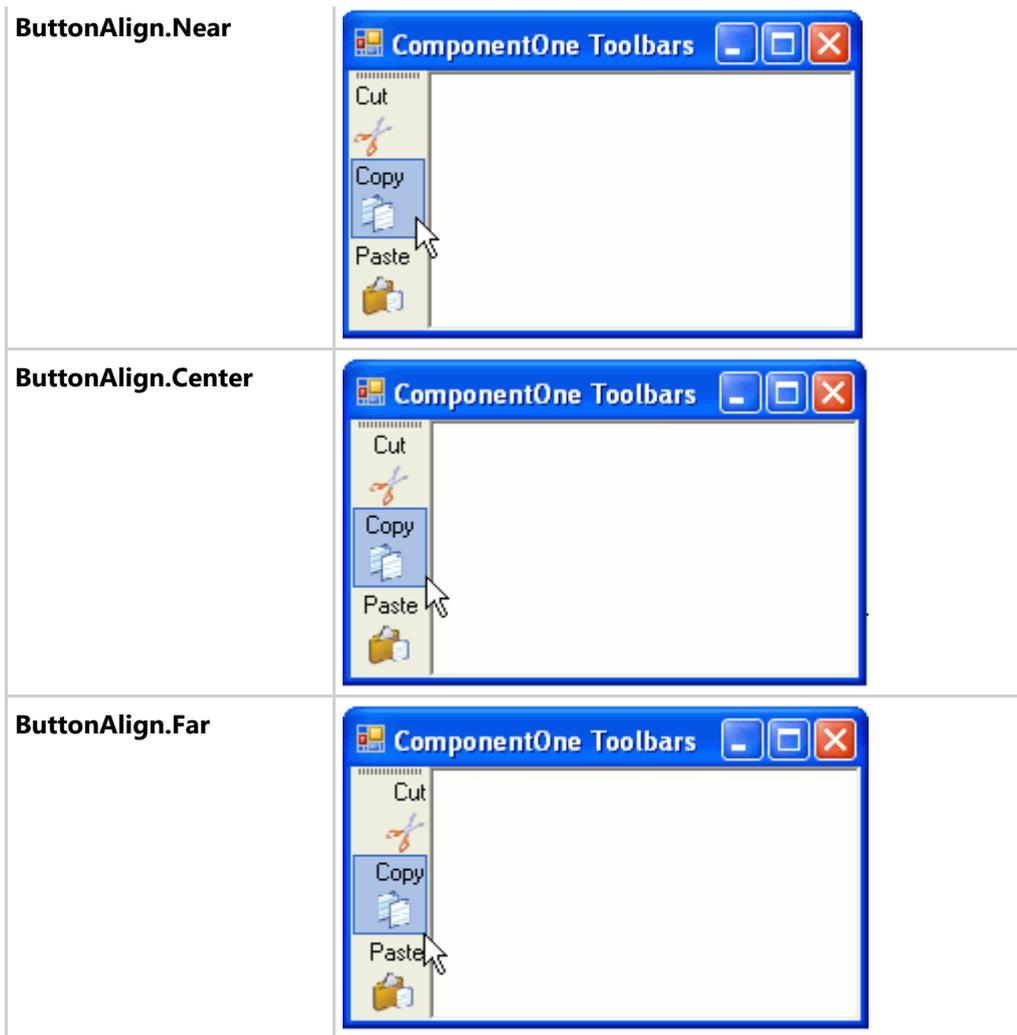
`C1ToolBar` layout is very flexible. They can be horizontal or vertical as well as docked to specific areas of the form. The toolbar's `Movable` property is enabled by default. This allows the user to move the toolbar anywhere on the form. The default layout for a toolbar is horizontal. You can change the toolbar layout to vertical by setting the `Horizontal` to **False**.

Note: When you set the `ToolBarStyle` property to `DropDownMenu`, the menu behaves like a drop-down so the toolbar becomes stationary.

In addition to toolbar orientation, `C1ToolBar` also provides button alignment for vertical toolbars. You can align the image or text near, center, or far from the button through the `ButtonAlign` property.

The following table shows the values for the `ButtonAlign` property:

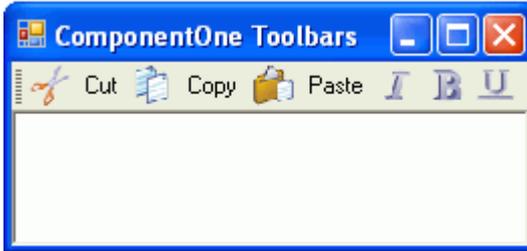
Property Setting	Image
------------------	-------



You can determine the relative position of text and images for toolbar buttons in horizontal and vertical toolbars using the [ButtonLayoutHorz](#) and [ButtonLayoutVert](#) properties. The [ButtonLayoutHorz](#) property gets the layout of the buttons when the toolbar is horizontal. This is the default orientation of the toolbar. The [ButtonLayoutVert](#) property gets the layout of the buttons when the toolbar is vertical. Setting the [Horizontal](#) property to **False** gets the vertical orientation for the toolbar.

Note: The default value for the [ButtonLayoutHorz](#) property is `TextOnRight`.

[C1ToolBar](#) provides several options for customizing the toolbar buttons for vertical and horizontal toolbars.

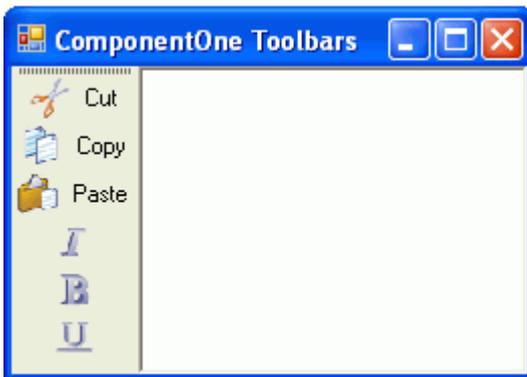
Property Setting	Image
<p>ButtonLayoutHorz.TextOnRight (default)</p>	

ButtonLayoutHorz.TextOnLeft	
ButtonLayoutHorz.TextAbove	
ButtonLayoutHorz.TextBelow	

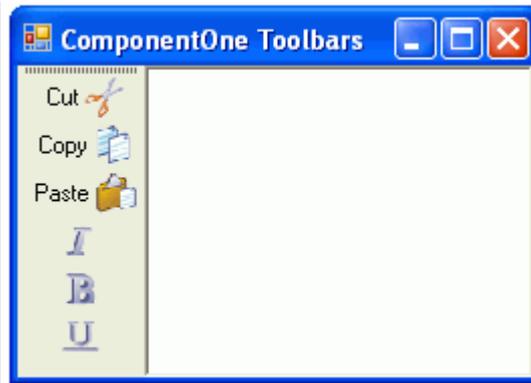
In addition to controlling the relative position of text and images for toolbar buttons you can also set the [ButtonLookHorz](#) property to display text, images, or both for the horizontal toolbar and the [ButtonLookVert](#) property to display text, images, or both for the vertical toolbar.

Note: The Text, Image, and TextAndImage values for the [ButtonLook](#) property overrides the values for the [ButtonLookHorz](#) and [ButtonLookVert](#) properties. The [ButtonLook](#) property should be set to default if you plan on setting values for the [ButtonLookHorz](#) or [ButtonLookVert](#) property.

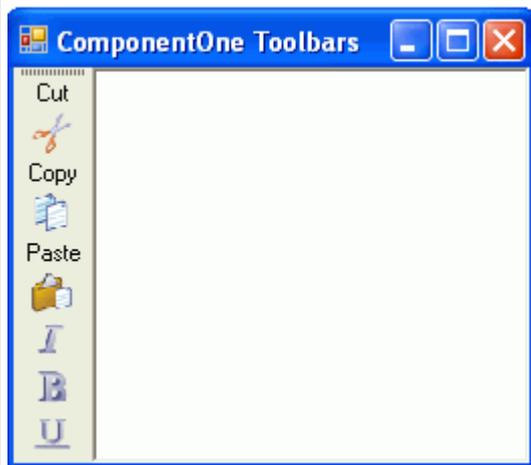
The following table shows the values for the [ButtonLayoutVert](#) property:

Property Setting	Image
ButtonLayoutVert.TextOnRight	

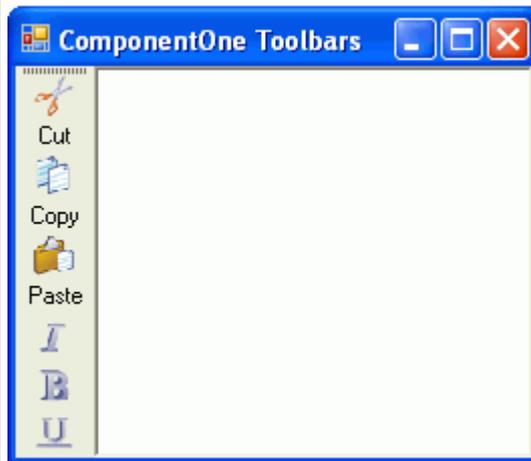
ButtonLayoutVert.TextOnLeft



ButtonLayoutVert.TextAbove



**ButtonLayoutVert.TextBelow
(default)**



DockingTab Overview

The [C1DockingTab](#) control implements the familiar tab control interface, where several overlaying pages (each of which contains arbitrary controls) are accessible via tabs at the side of the control.

In addition to this standard behavior, [C1DockingTab](#) also provides docking and floating behavior, where the whole control, or individual pages (tabs) can be torn off and automatically docked to one of the other sides of the form, to another [C1DockingTab](#) control, or floated in a separate tool window. To enable the docking and floating behavior of the [C1DockingTab](#) control, it must be placed inside a [C1CommandDock](#). For more information, see [Enabling DockingTab Docking and Floating](#).

DockingTab Appearance and Behavior Properties

[C1DockingTab](#) provides a number of useful properties to control the behavior and appearance of the tab items.

[C1DockingTab](#) includes a variety of appearance properties to visually enhance and customize the control. The control's tab style, size, and layout can easily be customized through the **C1DockingTab**'s appearance properties. These properties can be set at design time through the Properties window or through code.

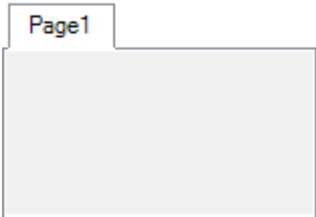
In addition to the appearance properties, [C1DockingTab](#) has several useful behavioral properties for closing tab pages, rearranging tabs, and mouse over effects for tab pages.

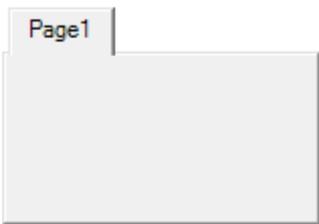
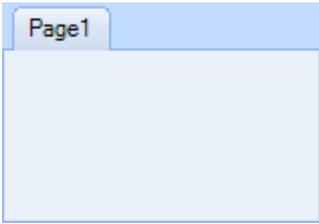
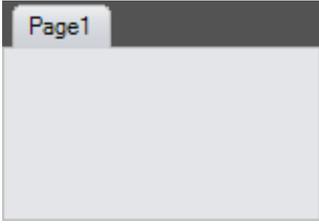
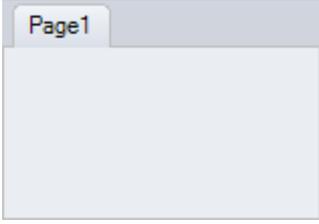
The following section introduces some of the common appearance and behavior properties used for the **C1DockingTab** control.

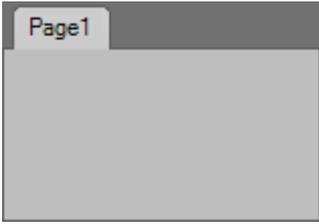
DockingTab Visual Styles

The [C1DockingTab](#) control provides several built-in styles, such as **Custom**, **System**, **Office2010Blue**, **Office2010Black**, **Office2010Silver**, **Office2007Blue**, **Office2007Black**, **Office2007Silver**, and **Classic** that can be easily applied using the [VisualStyle](#) property.

The following table illustrates each of the [C1DockingTab](#) control's visual styles.

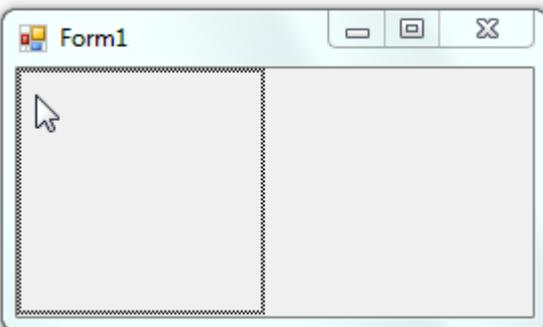
Property Setting	Image
VisualStyle.Custom	[Custom allows you to customize the control's visual style]
VisualStyle.System	

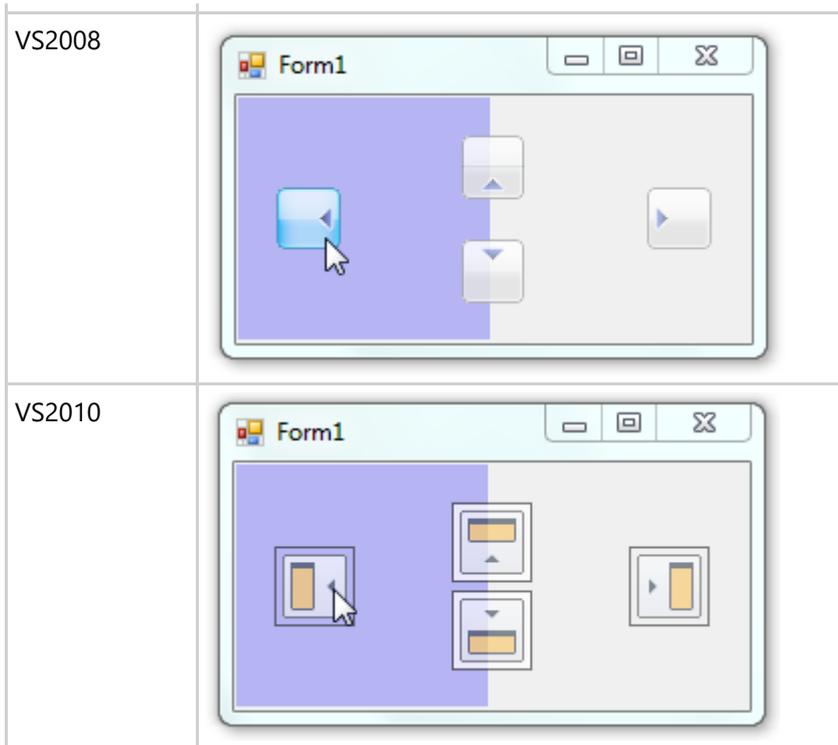
VisualStyle.Classic	
VisualStyle.Office2007Blue	
VisualStyle.Office2007Black	
VisualStyle.Office2007Silver	
VisualStyle.Office2010Blue	

VisualStyle.Office2010Black	
VisualStyle.Office2010Silver	

Docking Styles

When the [C1DockingTab](#) control is placed in the [C1CommandDock](#) component (see [Enabling DockingTab Docking and Floating](#)), it becomes a dockable control. You can select a specific style for docking by setting the [C1CommandDock](#) component's [DockingStyle](#) property to **Default**, **VS2005**, **VS2008**, or **VS2010**. The **Default** setting shows a gray shaded outline to illustrate where the control will be docked when you release the cursor, whereas **VS2005**, **VS2008**, and **VS2010** mimic the docking style of Visual Studio 2005, Visual Studio 2008, and Visual Studio 2010 respectively. The table below provides an example of each docking style.

Docking Style	Example
Default	
VS2005	



Tab Styles

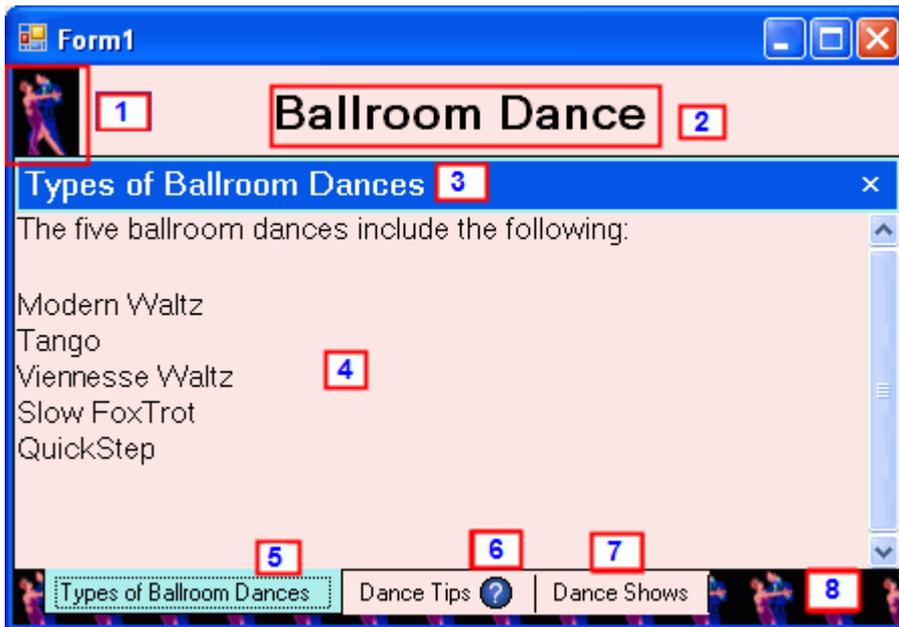
C1DockingTab provides several built-in styles such as **Custom**, **System**, **Office2007Blue**, **Office2010Blue**, **Office2010Black**, **Office2010Silver**, **Office2007Black**, **Office2007Silver**, **Office2003Blue**, **Office2003Olive**, **Office2003Silver**, and **Classic** that can be easily applied using the [VisualStyle](#) property.

C1DockingTab provides several built-in styles such as **Classic**, **Default**, **Office2003**, **Office2007**, **Office2010**, **Rounded**, **Sloping** that can be easily applied using the [TabStyle](#) property. You could also choose what type of look you would like for your tab pages. For example, you may want the tab pages to display and image, text, or a combination of both. You can do this by setting the [TabLook](#) property to **Default**, **Text**, **Image** or **TextandImage**.

You could also customize your tab's style using **C1DockingTab**'s general appearance properties such as the **C1DockingTab.BackgroundImage**, **C1DockingTab.BackgroundImageLayout**, [TabAreaBackColor](#), [TabBackColor](#), [TabBackColorSelected](#), and [TabForeColorSelected](#).

The following figure illustrates a few of **C1DockingTab**'s general appearance properties:

Figure 1



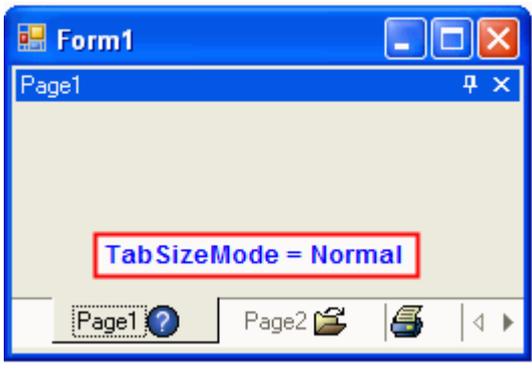
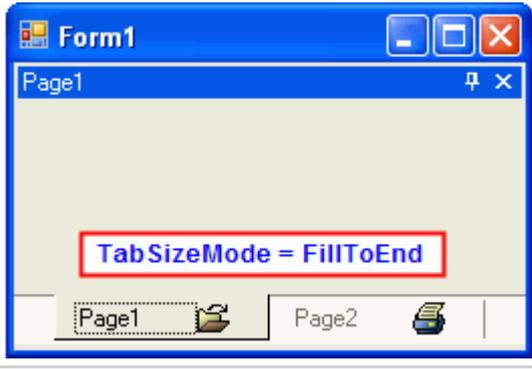
Labels for Figure 1	Description
1	PictureBox control
2	Label control
3	C1DockingTabPage.CaptionText
4	RichTextBox control
5	C1DockingTabPage.TabBackColorSelected = PaleTurquoise
6	C1DockingTabPage.Image
7	C1DockingTabPage.TabBackColor = MistyRose
8	C1DockingTab.BackgroundImage

Tab Sizing

You can customize the width and height of the tab pages by setting its [ItemSize](#) to an appropriate value. This will make all the tab pages the same size no matter what the content is inside each tab. Its default settings for its size are based on the size of its text and image. In addition to customizing the height and width of each tab you can also set a mode for the tabs. The [TabSizeMode](#) property contains four types of members: **FillToEnd**, **Fit**, **Normal**, and **User**. The **FillToEnd** stretches the tabs so they take the whole width of the tab control. The **Fit** squeezes all tabs so they fit into the width of the tab control. The **Normal** member gets the default sizing mode. The **User** member allows the user to specify the tab size in the MeasureTab event. The following table shows the effect of each [TabSizeMode](#) member.

Note: The third image in the following table contains only two tab pages to better illustrate the effect of the [FillToEnd](#) member.

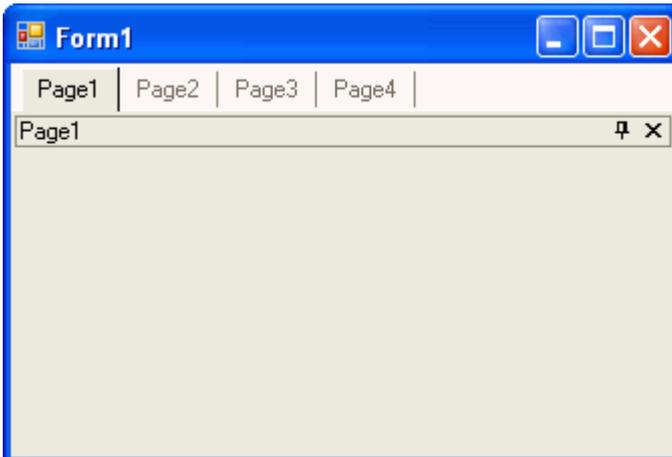
Property Setting	Image
------------------	-------

TabSizeMode.Normal	
TabSizeMode.Fit	
TabSizeMode.FillToEnd	
TabSizeMode.User	

Tab Orientation

C1DockingTab provides various options for the tab orientation. The tabs can be aligned to the top, bottom, left, or right of the **C1DockingTab** control by using the [Alignment](#) property.

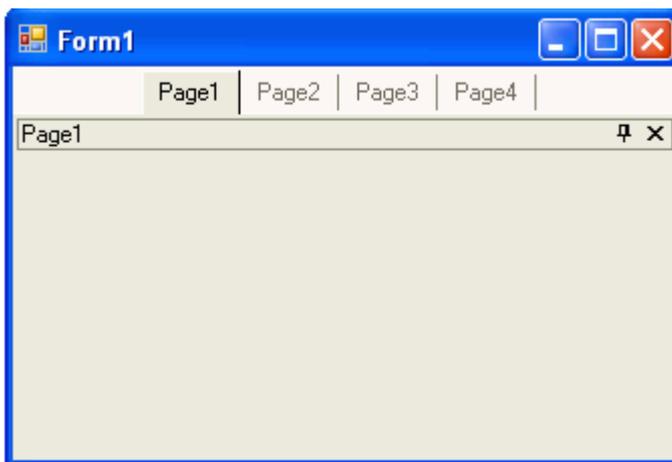
The following image displays the tabs aligned at the top of the [C1DockingTab](#).



In addition to choosing the tab layout you can also specify how the tabs are aligned along the side of the page content area by setting the `AlignTabs` property to **Near**, **Far**, or **Center**.

Note: The `AlignTabs` will not take effect if the `TabSizeMode` property is set to `Fit`.

The following image displays the tabs aligned at the Top and Center of the `C1DockingTab` control:

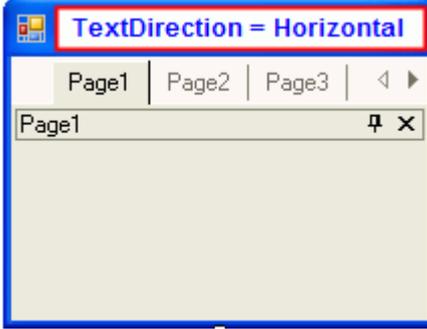
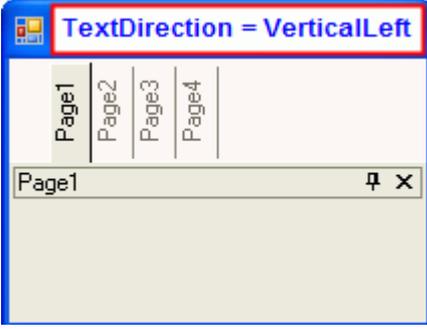
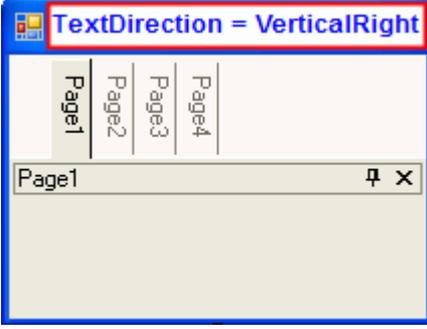


If you would like your tabs located at a specific location rather than **Near**, **Far**, or **Center** of the control you can specify exactly where by setting the `Indent` property to a specific number in pixels. For example the following image depicts the tabs indented to 30 pixels.



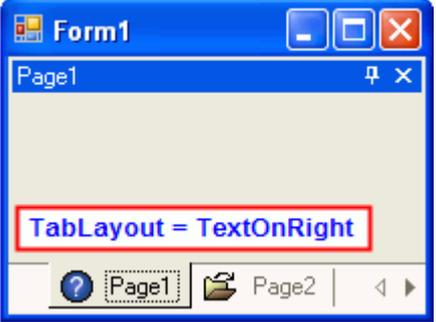
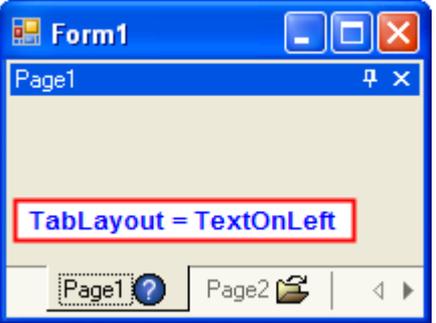
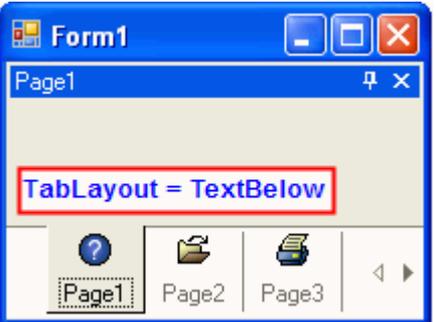
Text Orientation

In addition to determining the layout of the tabs you can also customize the layout of the text in the tabs. You can use the [TextDirection](#) property to display the text horizontally, vertically, vertically from right to left, or vertically from left to right. The following table illustrates how the tabs appear when the [TextDirection](#) is set to **Horizontal**, **VerticalLeft**, or **VerticalRight**:

Property Setting	Image
TextDirection.Horizontal	
TextDirection.VerticalLeft	
TextDirection.VerticalRight	

When you have text and an image displayed in the tab you may want to control whether the text is displayed above, below, to the left or to the right of the image. You can use [TabLayout](#) property to set the text above, below, to the left or to the right of the image in the tab.

Property Setting	Image
------------------	-------

TabLayout.TextOnRight	
TabLayout.TextonLeft	
TabLayout.TextAbove	
TabLayout.TextBelow	

Text Editing

The [C1DockingTab](#) control enables users to edit the text appearing on it by simply double-clicking tab. With added keyboard support, you can also edit the text appearing on the selected tab by pressing the **F2** key.

Hiding, Closing, and Moving Individual Tab Pages

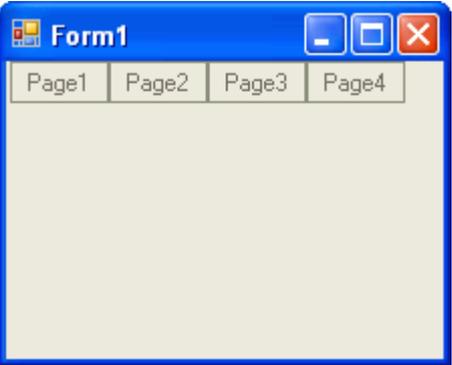
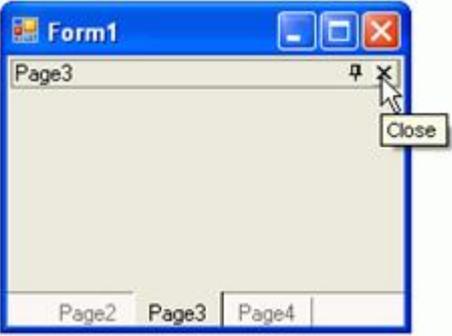
The [C1DockingTab](#) control includes the following behavioral properties for hiding, closing, and moving tab pages at run time:

- [CanAutoHide](#)

- [CanCloseTabs](#)
- [CanMoveTabs](#)

For your convenience these properties are set to **True** by default. If you don't want to enable these properties you can set them to **False**. The following table depicts how the [C1DockingTab](#) appears at run time when the [CanAutoHide](#), [CanCloseTabs](#), and [CanMoveTabs](#) are set to **True**.

 The AutoHidingChanging event allows you to recognize the change of the AutoHide property, before the C1DockingTab control is resized. The event is fired before the "SizeChanged" and the "AutoHidingChanged" events are fired.

Action	Result
AutoHide icon clicked	
Page1 Close button clicked	
Page1 tab moved	

Mouse-Over Styles in Tab Pages

[C1DockingTab](#) provides a simple property for applying mouse-over effects to the tabs. When the [HotTrack](#) property is set to **True** at design time or programmatically it changes the tab's appearance when the mouse hovers over it.

NavBar Overview

The **C1NavBar** is used for grouping information into distinct categories to help organize and navigate information quickly. It consists of a number of categories represented by preset buttons. Each button has a header which contains both text and an image as well as a panel for adding information to the button category. Only one category can be displayed at one time, with the remaining buttons visible, but their associated panels are hidden.

NavBar Appearance and Behavior Properties

C1NavBar provides a number of useful properties to control the behavior and appearance of the **C1NavBar**.

C1NavBar includes a variety of appearance properties to visually enhance and customize the control. The control's style, size, and layout can easily be customized through the **C1NavBar**'s appearance properties. These properties can be set at design time through the Properties window, or through the **C1NavBar Tasks**. Additionally, the NavBar's appearance can be configured programmatically through the **C1NavBar** class.

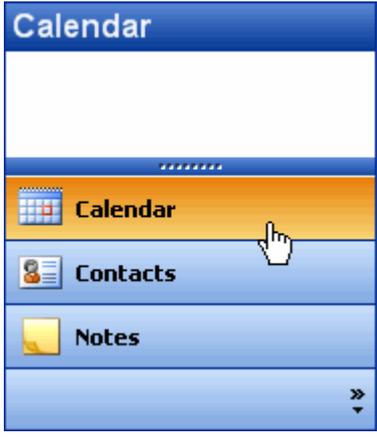
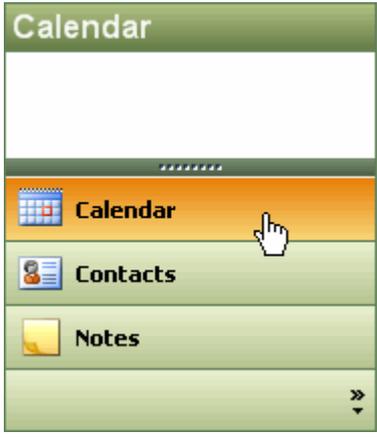
The following section introduces some of the common appearance and behavior properties used for the **C1NavBar** control.

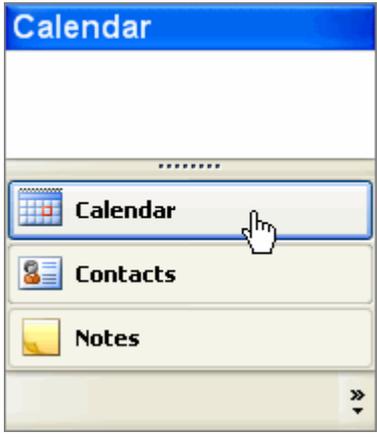
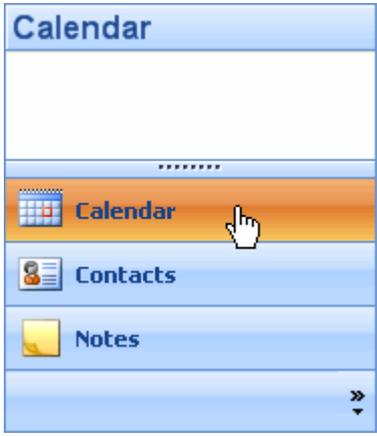
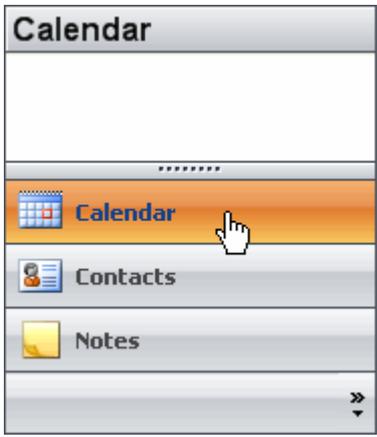
NavBar Visual Styles

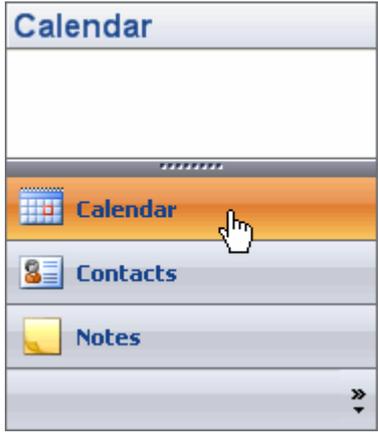
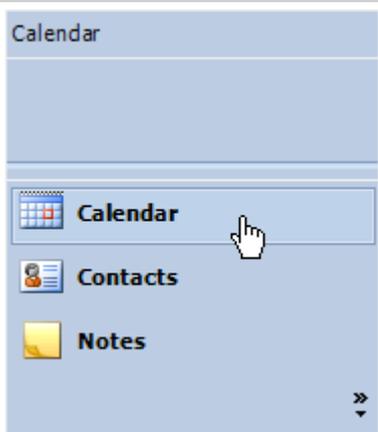
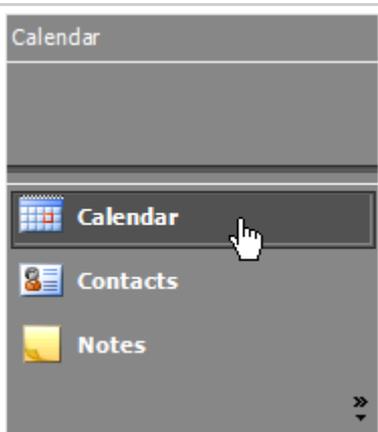
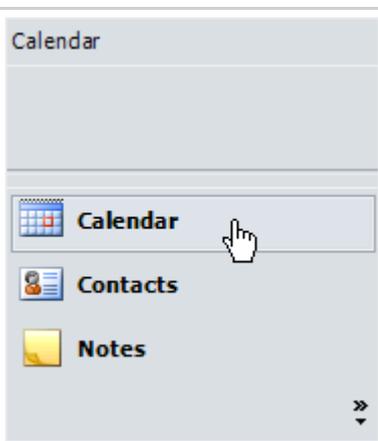
The **C1NavBar** control provides several built-in styles, such as **Custom**, **System**, **Office2010Blue**, **Office2010Black**, **Office2010Silver**, **Office2007Blue**, **Office2007Black**, **Office2007Silver**, **Office2003Blue**, **Office2003Olive**, **Office2003Silver**, **OfficeXP**, **Classic**, and **WindowsXP** that can be easily applied using the **VisualStyle** property.

The following table illustrates each of the **C1NavBar** control's visual styles.

Property Setting	Image
VisualStyle.Custom	[Custom allows you to customize the visual style.]
VisualStyle.System	 A screenshot of the C1NavBar control in the 'System' visual style. The control has a blue header bar with the text 'Calendar'. Below the header, there are three buttons: 'Calendar' (with a calendar icon), 'Contacts' (with a person icon), and 'Notes' (with a notepad icon). A mouse cursor is hovering over the 'Calendar' button. At the bottom right of the control, there are two small icons: a right-pointing arrow and a downward-pointing arrow.

VisualStyle.Office2003Blue	 A screenshot of a WinForms application using the Office 2003 Blue visual style. The window title bar is blue with the text "Calendar". Below the title bar is a menu bar with a blue background and white text. The menu items are "Calendar", "Contacts", and "Notes". The "Calendar" menu item is highlighted with a blue background and white text, and a mouse cursor is pointing at it. The "Contacts" and "Notes" menu items have a lighter blue background and black text. The "Calendar" menu item has a small calendar icon to its left. The "Contacts" menu item has a small person icon to its left. The "Notes" menu item has a small notepad icon to its left. At the bottom right of the menu bar, there are two right-pointing arrows and a downward-pointing arrow.
VisualStyle.Office2003Olive	 A screenshot of a WinForms application using the Office 2003 Olive visual style. The window title bar is olive green with the text "Calendar". Below the title bar is a menu bar with an olive green background and white text. The menu items are "Calendar", "Contacts", and "Notes". The "Calendar" menu item is highlighted with an orange background and white text, and a mouse cursor is pointing at it. The "Contacts" and "Notes" menu items have a lighter olive green background and black text. The "Calendar" menu item has a small calendar icon to its left. The "Contacts" menu item has a small person icon to its left. The "Notes" menu item has a small notepad icon to its left. At the bottom right of the menu bar, there are two right-pointing arrows and a downward-pointing arrow.
VisualStyle.Office2003Silver	 A screenshot of a WinForms application using the Office 2003 Silver visual style. The window title bar is silver with the text "Calendar". Below the title bar is a menu bar with a silver background and white text. The menu items are "Calendar", "Contacts", and "Notes". The "Calendar" menu item is highlighted with an orange background and white text, and a mouse cursor is pointing at it. The "Contacts" and "Notes" menu items have a lighter silver background and black text. The "Calendar" menu item has a small calendar icon to its left. The "Contacts" menu item has a small person icon to its left. The "Notes" menu item has a small notepad icon to its left. At the bottom right of the menu bar, there are two right-pointing arrows and a downward-pointing arrow.
VisualStyle.OfficeXP	 A screenshot of a WinForms application using the Office XP visual style. The window title bar is blue with the text "Calendar". Below the title bar is a menu bar with a light beige background and black text. The menu items are "Calendar", "Contacts", and "Notes". The "Calendar" menu item is highlighted with a blue background and white text, and a mouse cursor is pointing at it. The "Contacts" and "Notes" menu items have a light beige background and black text. The "Calendar" menu item has a small calendar icon to its left. The "Contacts" menu item has a small person icon to its left. The "Notes" menu item has a small notepad icon to its left. At the bottom right of the menu bar, there are two right-pointing arrows and a downward-pointing arrow.

VisualStyle.Classic	 <p>The image shows a Windows Forms calendar control with the Classic visual style. The title bar is blue with the text "Calendar". Below the title bar is a horizontal separator line. The main area contains three buttons: "Calendar" (with a calendar icon), "Contacts" (with a person icon), and "Notes" (with a notepad icon). The "Calendar" button is highlighted with a blue border and a mouse cursor is pointing at it. At the bottom right, there are two right-pointing arrows and a downward-pointing arrow.</p>
VisualStyle.WindowsXP	 <p>The image shows a Windows Forms calendar control with the WindowsXP visual style. The title bar is blue with the text "Calendar". Below the title bar is a horizontal separator line. The main area contains three buttons: "Calendar" (with a calendar icon), "Contacts" (with a person icon), and "Notes" (with a notepad icon). The "Calendar" button is highlighted with a blue border and a mouse cursor is pointing at it. At the bottom right, there are two right-pointing arrows and a downward-pointing arrow.</p>
VisualStyle.Office2007Blue	 <p>The image shows a Windows Forms calendar control with the Office2007Blue visual style. The title bar is light blue with the text "Calendar". Below the title bar is a horizontal separator line. The main area contains three buttons: "Calendar" (with a calendar icon), "Contacts" (with a person icon), and "Notes" (with a notepad icon). The "Calendar" button is highlighted with an orange border and a mouse cursor is pointing at it. At the bottom right, there are two right-pointing arrows and a downward-pointing arrow.</p>
VisualStyle.Office2007Black	 <p>The image shows a Windows Forms calendar control with the Office2007Black visual style. The title bar is light gray with the text "Calendar". Below the title bar is a horizontal separator line. The main area contains three buttons: "Calendar" (with a calendar icon), "Contacts" (with a person icon), and "Notes" (with a notepad icon). The "Calendar" button is highlighted with an orange border and a mouse cursor is pointing at it. At the bottom right, there are two right-pointing arrows and a downward-pointing arrow.</p>

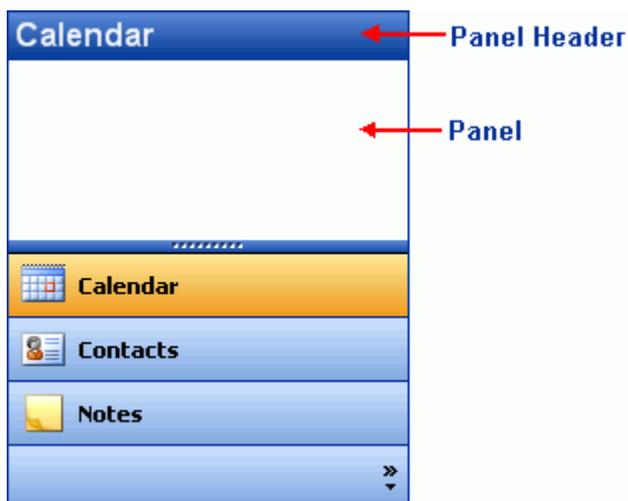
VisualStyle.Office2007Silver	 <p>The image shows a menu bar with a light blue header labeled "Calendar". Below it, a menu is open with a dark orange background. The menu items are "Calendar" (with a calendar icon), "Contacts" (with a person icon), and "Notes" (with a yellow notepad icon). A mouse cursor is pointing at the "Calendar" menu item. At the bottom right of the menu, there are two right-pointing arrows and a downward-pointing arrow.</p>
VisualStyle.Office2010Blue	 <p>The image shows a menu bar with a light blue header labeled "Calendar". Below it, a menu is open with a light blue background. The menu items are "Calendar" (with a calendar icon), "Contacts" (with a person icon), and "Notes" (with a yellow notepad icon). A mouse cursor is pointing at the "Calendar" menu item. At the bottom right of the menu, there are two right-pointing arrows and a downward-pointing arrow.</p>
VisualStyle.Office2010Black	 <p>The image shows a menu bar with a dark grey header labeled "Calendar". Below it, a menu is open with a dark grey background. The menu items are "Calendar" (with a calendar icon), "Contacts" (with a person icon), and "Notes" (with a yellow notepad icon). A mouse cursor is pointing at the "Calendar" menu item. At the bottom right of the menu, there are two right-pointing arrows and a downward-pointing arrow.</p>
VisualStyle.Office2010Silver	 <p>The image shows a menu bar with a light grey header labeled "Calendar". Below it, a menu is open with a light grey background. The menu items are "Calendar" (with a calendar icon), "Contacts" (with a person icon), and "Notes" (with a yellow notepad icon). A mouse cursor is pointing at the "Calendar" menu item. At the bottom right of the menu, there are two right-pointing arrows and a downward-pointing arrow.</p>

NavBar Panel Styles

NavBar Panel Header

You can use the [PanelHeader](#) property in the **C1NavBarButton Collection Editor** to change the default text that displays in the header for the preset buttons. To customize the font style of the header, you can use the [PanelHeaderFont](#) property.

The height of the panel header can be changed through the [PanelHeaderHeight](#) property.



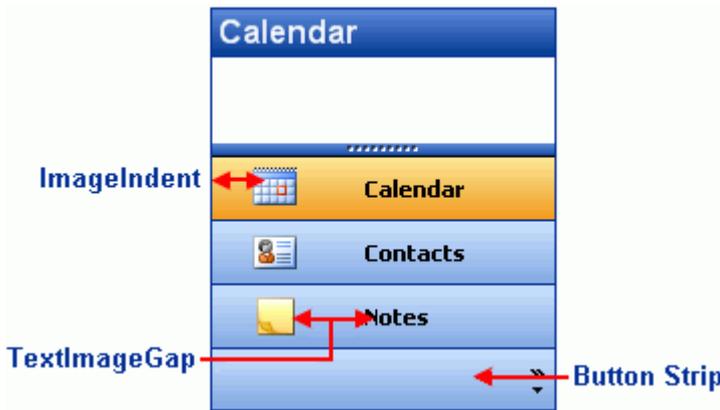
NavBar Panel

You can customize the **C1NavBar** panels' appearance properties such as its **BackColor** and **BackgroundImage** through the [C1NavBarPanel](#) class.

NavBar Button Styles

The **C1NavBar** consists of vertically stacked pre-set buttons or custom buttons. The preset buttons include the following: Custom, Mail, Calendar, Contacts, Tasks, Notes, Folder, Shortcut, and Journal. Each preset button has an image and text. By default the image appears to the left of the text. You can display images of buttons horizontally in the button strip at the bottom of the **C1NavBar**. If you plan to display many buttons on the button strip it's a good idea to increase the height of it. You can use the [StripHeight](#) property to increase the height of the button strip. The default height for the button strip is 30 pixels.

C1NavBar provides several appearance properties to customize the style of the buttons. The following image illustrates the [ImageIndent](#) and [TextImageGap](#) properties



Button Text and Image Layout

You can configure the layout of the text and image for each button to get the desired appearance through the [ImageIndent](#) and [TextImageGap](#) properties. If you would like to increase the indent of the image you can use the [ImageIndent](#) property. If you would like more space between the image and the text then you can use the [TextImageGap](#) property to increase the pixel width. By default, the [ImageIndent](#) property is 6 pixels and the [TextImageGap](#) is 3 pixels.

NavBar Vertical Text

Text can be displayed vertically upon the collapse of the [C1NavBar](#) control if its [VisualStyle](#) property is set to **Office2010Black**, **Office2010Blue**, or **Office2010Silver**. The following image displays an example of vertical text on the [C1NavBar](#) control.

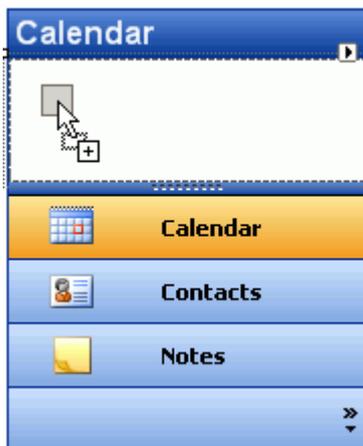


To achieve this effect, set the [UIStrings CollapsedBarText](#) user interface string enumeration to the text you want to appear when the user collapses the control. Please note that you must also set the control's [AllowCollapse](#) property to **True** if you want to make the control collapsible at run time.

Embedded Controls in NavBar Panels

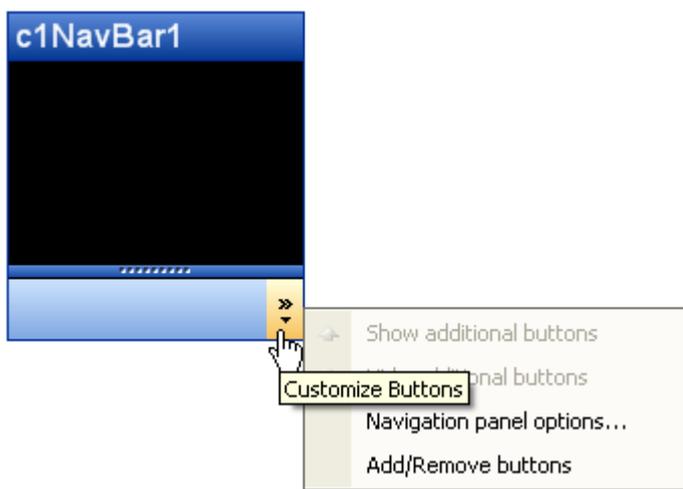
Arbitrary controls such as a label can be embedded in a [C1NavBarPanel](#).

This can simply be done by dragging an arbitrary control onto the panel. The following image illustrates a label control being dragged into the panel:



Run-Time Customization for NavBar

The buttons for the **C1NavBar** can be customizable at run time by clicking on the drop-down arrow button. The following pop-up menu appears at run time when you click on the drop-down arrow:



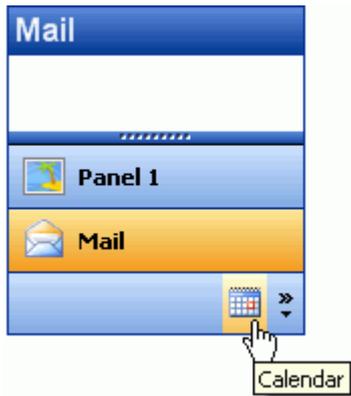
The Customize Buttons menu operates as follows:

Show additional buttons

Clicking on the **Show additional buttons** command item from the menu removes the button from the task bar area and places it in the panel area of the **C1NavBar**.

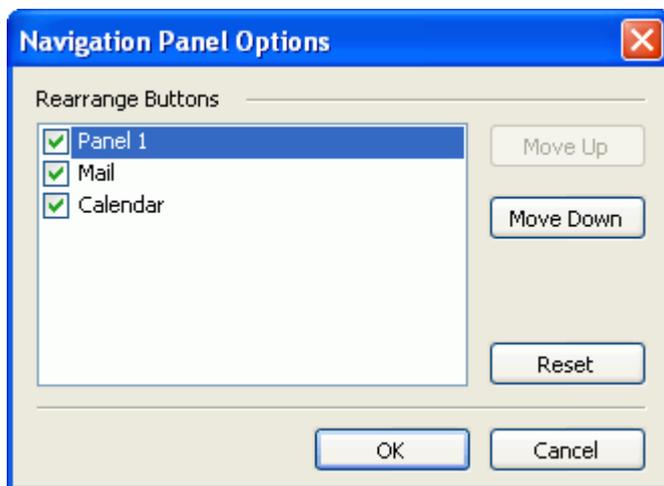
Hide additional buttons

Clicking on the hide additional items removes the button from the panel and places in the task bar area.



Navigation panels options

Clicking on the **Navigation panel options** item opens the **Navigation Panel Options** dialog box.



The Navigation panel options dialog box consists of the following command buttons:

Name	Description
Move Up	Moves the selected button up one position.
Move Down	Moves the selected button down one position.
Reset	Resets the buttons to their default position.
OK	Saves the changes and closes the Navigation Panel Options dialog box.
Cancel	Cancel the changes to the buttons and closes the Navigation Panel Options dialog box.

Add/Remove buttons

Clicking on the **Add/Remove** buttons item opens a menu with the names of the present buttons shown in the **C1NavBar** along with a check box next to the names of the present buttons shown in the **C1NavBar**. To remove a specific button click on the check box next to the name of the button you wish to remove.



OutBar Overview

C1OutBar is an outlook-style container/tab control. It provides a collection of pages (of the type **C1OutPage**). Each page includes a title bar and an empty page or a page with commands such as toolbar buttons. Clicking on a page title or title bar displays that page and hides the remaining pages. Each page is organized in a vertical stack. Pages may be used to store arbitrary controls, but special processing (such as smart scrolling and so on) is provided when storing a single **C1ToolBar** per page.

The following topics provide further detail about the appearance and behavior of the **C1OutBar** control.

OutBar Appearance and Behavior Properties

C1OutBar provides a number of useful properties to control the behavior and appearance of the tab items.

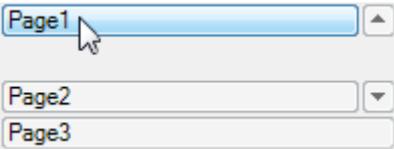
C1OutBar includes a variety of appearance properties to visually enhance and customize the control. The control's page style, size, and layout can easily be customized through the using **C1OutBar**'s appearance properties. These properties can be set at design time through the Properties window or programmatically.

The following section introduces some of the common appearance and behavior properties used for the **C1OutBar** control.

OutBar Visual Styles

The **C1OutBar** control provides several built-in styles, such as **Custom**, **System**, **Office2010Blue**, **Office2010Black**, **Office2010Silver**, **Office2007Blue**, **Office2007Black**, **Office2007Silver**, **Office2003Blue**, **Office2003Olive**, **Office2003Silver**, **OfficeXP**, **Classic**, and **WindowsXP** that can be easily applied using the **VisualStyle** property.

The following table illustrates each of the **C1OutBar** control's visual styles.

Property Setting	Image
VisualStyle.Custom	[Custom allows you to customize the visual style.]
VisualStyle.System	
VisualStyle.Office2003Blue	
VisualStyle.Office2003Olive	
VisualStyle.Office2003Silver	

VisualStyle.OfficeXP	
VisualStyle.Classic	
VisualStyle.WindowsXP	
VisualStyle.Office2007Blue	
VisualStyle.Office2007Black	
VisualStyle.Office2007Silver	
VisualStyle.Office2010Blue	
VisualStyle.Office2010Black	
VisualStyle.Office2010Silver	

Page Styles

C1OutBar provides several built-in styles such as **Custom**, **System**, **Office2007Blue**, **Office2007Black**, **Office2010Blue**, **Office2010Black**, **Office2010Silver**, **Office2007Silver**, **Office2003Blue**, **Office2003Olive**, **Office2003Silver**, **OfficeXP**, **Classic**, and **WindowsXP** that can be easily applied using the **VisualStyle** property. Note that the **VisualStyle** property and **VisualStyle** enumeration supersede the **LookAndFeel** property and **LookAndFeelEnum** enumeration, which are now obsolete.

You could also choose what type of look you would like for your tab pages. For example, you may want the buttons to display an image, text, or a combination of both. You can do this by setting the **ButtonLook** property to **Default**, **Text**,

Image or TextandImage.

You could also customize your out bar's style using **C1OutBar**'s general appearance properties such as the [BackColor](#), **C1OutBar.BackgroundImage**, **C1OutBar.BackgroundImageLayout**, and [BackHiColor](#).

You can set a different background color for each c1outpage using its **C1OutPage.BackColor** property.

For more information about using these properties to modify page styles for the C1OutBar, see [Modifying the Appearance of the C1OutBar](#).

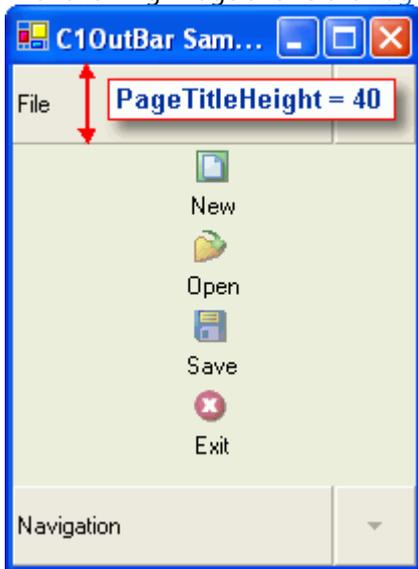
Embedded Controls in Pages

Arbitrary controls such as a TextBox can be embedded in an empty page (a page without a toolbar) or a page with a toolbar.

Page Sizing

You can customize the height of the **C1OutPages** by setting its [PageTitleHeight](#) to an appropriate value. This will make all the page titles the same height. Its default value is zero.

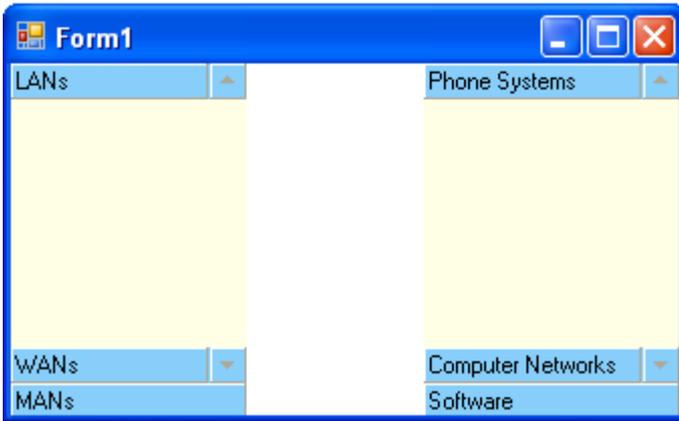
The following image shows the [PageTitleHeight](#) set to 40 pixels.



Page Layout and Alignment

C1OutBar can be docked to the top, left, right or bottom on the container that the **C1CommandDock** has been assigned to.

The following image shows two **C1OutBars** docked to the left and right side of the form. Each **C1OutBar** is docked inside a **C1CommandDock**.



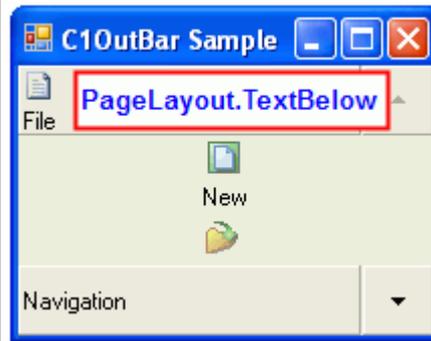
C1OutBar provides various options for the text and image alignment in the pages. The image or text in the pages can be aligned to the left, right, or center of the title pages/outpages in the **C1OutBar** control by using the **Align** property.

When you have text and an image displayed in the page title you may want to control whether the text is displayed above, below, to the left or to the right of the image. You can use **PageLayout** property to set the text above, below, to the left or to the right of the image in the page title. The default value for the **PageLayout** property is **TextOnRight**.

The following table shows the values for the **PageLayout** property.

Property Setting	Image
PageLayout.TextOnRight	
PageLayout.TextonLeft	
PageLayout.TextAbove	

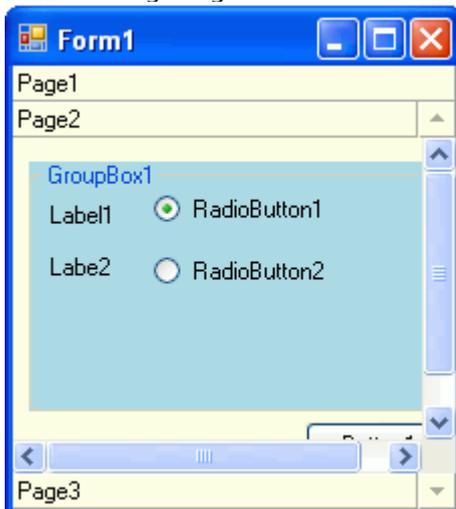
PageLayout.TextBelow



Scroll Bars

C1OutBar provides a **C1OutPage.AutoScroll** property to enable scroll bars for specific pages in the **C1OutBar**. By default, the **C1OutPage.AutoScroll** property is disabled. To enable the scroll bars, you can set the **C1OutPage.AutoScroll** property to **True**.

The following image shows scroll bars in the second page of the **C1OutBar**.



In addition to scroll bars, **C1OutBar** has a **ShowScrollButtons** property that enables scroll buttons along side of the **C1OutBar**. By default, this property is enabled. To disable this property, set the **ShowScrollButtons** to **False**.

TopicBar Overview

C1TopicBar represents a topic bar. In **C1OutBar** the control provides a collection of single pages organized into one group whereas **C1TopicBar** contains a collection of pages organized into various groups. The collection of pages in **C1TopicBar** are of the type, **C1TopicPage**. Each page includes a title bar with a collapsible/expandable image and the content area consists of links. Each page/group can have one or more links. Clicking once on the group collapses the link items in the selected group and clicking again expands the link items in the selected group. Each topic page is organized in a vertical stack.

The following topics provide further detail about the appearance and behavior of the **C1TopicBar** control.

TopicBar Appearance and Behavior Properties

C1TopicBar provides a number of useful properties to control the behavior and appearance of the tab items.

C1TopicBar includes a variety of appearance properties to visually enhance and customize the control. The control's page style, size, and layout can easily be customized through the using **C1TopicBar**'s appearance properties. These properties can be set at design time through the Properties window or programmatically.

C1TopicBar includes several useful behavioral properties for closing tab pages, rearranging tabs, and mouse over effects for tab pages, and so on.

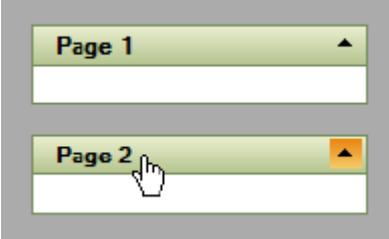
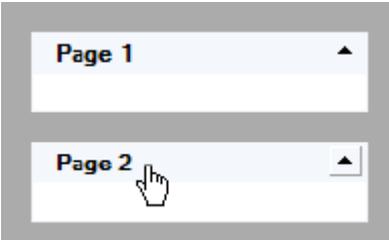
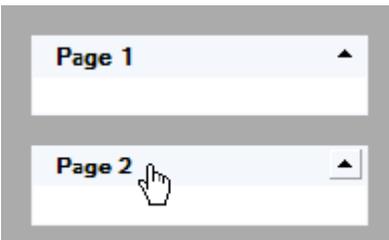
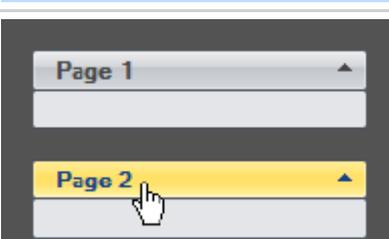
The following section introduces some of the common appearance and behavior properties used for the **C1TopicBar** control.

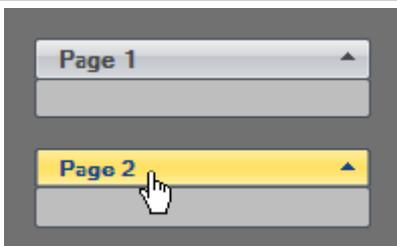
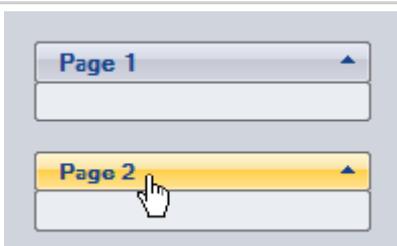
TopicBar Visual Styles

The **C1TopicBar** control provides several built-in styles, such as **Custom**, **System**, **Office2010Blue**, **Office2010Black**, **Office2010Silver**, **Office2007Blue**, **Office2007Black**, **Office2007Silver**, **Office2003Blue**, **Office2003Olive**, **Office2003Silver**, **OfficeXP**, **Classic**, and **WindowsXP** that can be easily applied using the **VisualStyle** property.

The following table illustrates each of the **C1TopicBar** control's visual styles.

Property Setting	Image
VisualStyle.Custom	[Custom allows you to customize the visual style.]
VisualStyle.System	
VisualStyle.Office2003Blue	

VisualStyle.Office2003Olive	 The image shows two page tabs. The top tab is labeled 'Page 1' and has a light green background. The bottom tab is labeled 'Page 2' and has a darker green background. A mouse cursor is pointing at the 'Page 2' tab.
VisualStyle.Office2003Silver	 The image shows two page tabs. The top tab is labeled 'Page 1' and has a light purple background. The bottom tab is labeled 'Page 2' and has a darker purple background. A mouse cursor is pointing at the 'Page 2' tab.
VisualStyle.OfficeXP	 The image shows two page tabs. The top tab is labeled 'Page 1' and has a light blue background. The bottom tab is labeled 'Page 2' and has a slightly darker blue background. A mouse cursor is pointing at the 'Page 2' tab.
VisualStyle.Classic	 The image shows two page tabs. The top tab is labeled 'Page 1' and has a light blue background. The bottom tab is labeled 'Page 2' and has a slightly darker blue background. A mouse cursor is pointing at the 'Page 2' tab.
VisualStyle.WindowsXP	 The image shows two page tabs. The top tab is labeled 'Page 1' and has a light blue background. The bottom tab is labeled 'Page 2' and has a slightly darker blue background. A mouse cursor is pointing at the 'Page 2' tab.
VisualStyle.Office2007Blue	 The image shows two page tabs. The top tab is labeled 'Page 1' and has a light blue background. The bottom tab is labeled 'Page 2' and has a yellow background. A mouse cursor is pointing at the 'Page 2' tab.
VisualStyle.Office2007Black	 The image shows two page tabs. The top tab is labeled 'Page 1' and has a light gray background. The bottom tab is labeled 'Page 2' and has a yellow background. A mouse cursor is pointing at the 'Page 2' tab.

VisualStyle.Office2007Silver	
VisualStyle.Office2010Blue	
VisualStyle.Office2010Black	
VisualStyle.Office2010Silver	

Collapsible and Expandable Topic Pages

You can use the [Collapsed](#) property to specify a collapsed or expanded page. Specific pages can be collapsed or expanded as well as all of the pages can be collapsed or expanded.

To Collapse a Specific Page at Design Time

To collapse a specific page at design time, select the desired page on the **C1TopicBar** control, and then open its **C1TopicBar Tasks** menu to set its [Collapsed](#) property to **True**.

To Collapse or Expand All Pages Programmatically

To collapse all pages programmatically, use the [CollapseAll](#) method or the [ExpandAll](#) method to expand all pages.

TopicBar Styles

C1TopicBar provides several built-in styles such as **Custom**, **System**, **Office2010Blue**, **Office2010Black**, **Office2010Silver**, **Office2007Blue**, **Office2007Black**, **Office2007Silver**, **Office2003Blue**, **Office2003Olive**, **Office2003Silver**, **OfficeXP**, **Classic**, and **WindowsXP** that can be easily applied using the [VisualStyle](#) property. Note

that the **VisualStyle** property and **VisualStyle** enumeration supersede the **LookAndFeel** property and **LookAndFeelEnum** enumeration, which are now obsolete.

You could also apply a special style to the topic pages to darken the back color of the page header. To apply the special style to the topic page use **SpecialStyle** property. You could also customize the **C1TopicBar**'s style using **C1TopicBar**'s general appearance properties such as the **BackColor**, **BackgroundImage**, **BackgroundImageLayout**, and **ForeColor**.

TopicBar Animation

The topic pages can have animation effects when it is being collapsed or expanded. To add animation effects to the topic pages, use the **Animation** property.

TopicPage Layout and Alignment

C1TopicBar can be docked to the top, left, right or bottom on the container that the **C1CommandDock** has been assigned to.

C1TopicBar provides various options for the text alignment in the pages. The text in the pages can be aligned to the left, right, or center of the title bars in the **C1TopicBar** control by using the **Align** property. When you display an image in the topic page, the text appears after the image by default.

ToolTips in TopicBar

The topic bar has a **TooltipText** property for you to use to create a user-friendly application. For example, you can add ToolTips to each topic page to provide more information to the user about the topic page. A ToolTip is used to display text when the mouse hovers over the control.

RadialMenu Overview

[C1RadialMenu](#) is a component that is similar to the [C1ContextMenu](#) except that its commands or buttons are arranged in a circle and are accessed via a button that pops up whenever you click on the form or in a touch enabled device, tap an item. The [C1RadialMenu](#) supports interactions via mouse, keyboard, and touch with touch-enabled monitor.

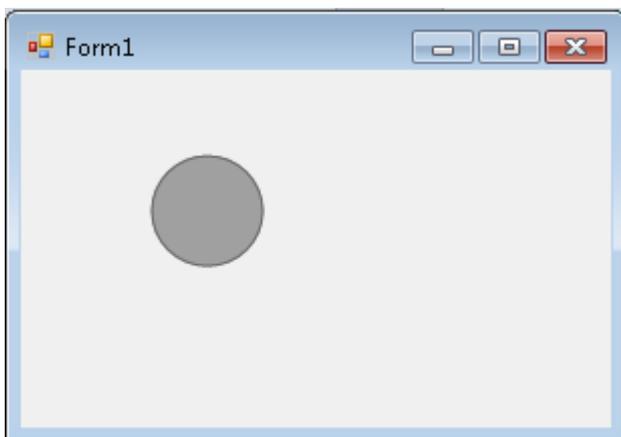
[C1RadialMenu](#) appears as a circular menu with several pie slices. In the center of the radial menu is a central button known as the inner radius button. Each slice in the radial menu can lead to another radial menu. An arrow button appears on the border edge of the pie slice if there are more submenu items. Clicking on the arrow will reveal another radial menu with several more pie slices that represent additional submenu items.

Since radial menus just like any context menu or popup menu are shown only when requested you will need to use the [C1RadialMenu.ShowMenu](#) method to determine how to display the [C1RadialMenu](#) on the form. By showing the [C1RadialMenu](#) only when needed will prevent visual distraction and memory overload.

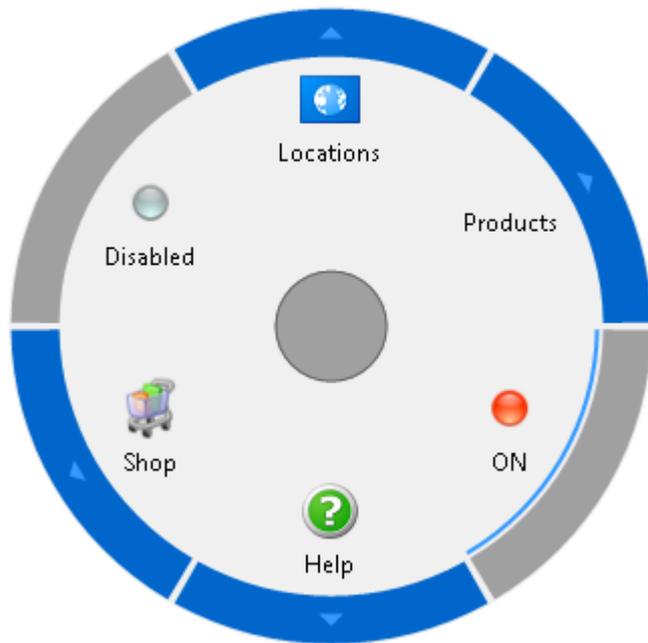
The following code example shows one simple way how to display the [C1RadialMenu](#) after adding the component to the form. You could also call arguments to show the [C1RadialMenu](#) as expanded or you could create a method to calculate the center.

C#
<pre>c1RadialMenu1.ShowMenu(this, new Point(350, 350));</pre>
Visual Basic
<pre>c1RadialMenu1.ShowMenu(Me, New Point(350, 350))</pre>

When you run your application the [C1RadialMenu](#) appears as one center button since there were no [C1RadialMenuItems](#) or [C1RadialCommandMenuItems](#) added yet:



The following image illustrates a [C1RadialMenu](#) with several [RadialMenuItems](#) and sub [RadialMenuItems](#). Each [RadialMenuItem](#) added to [C1RadialMenu](#) appears as a pie slice and clicking or tapping on the pie slice opens up a new circular [C1RadialMenu](#) with sub [RadialMenuItems](#).



RadialMenu Appearance and Behavior

The following section introduces some of the common appearance and behavior properties used for the `C1RadialMenu` control.

RadialMenu Animation

The Radial menu can have animation effects when it is opening or closing. To add animation effects to the topic pages, use the `UseAnimation` property.

ToolTips in Radial Menu

The Radial menu items and command items have `RadialMenuItem.ToolTip` and `RadialMenuCommandItem.ToolTip` properties for you to use to create a user-friendly application. For example, you can add ToolTips to each menu item page to provide more information to the user about the item or command. A Tooltip is used to display text when the mouse hovers over the item.

A `C1RadialMenu.ToolTipPosition` property is provided for you to specify the position of the tooltip text relative to the Radial menu. You can choose from Left, Top, Right, Bottom, or None (if you wish to specify no position).

Radius and Inner Radius Properties

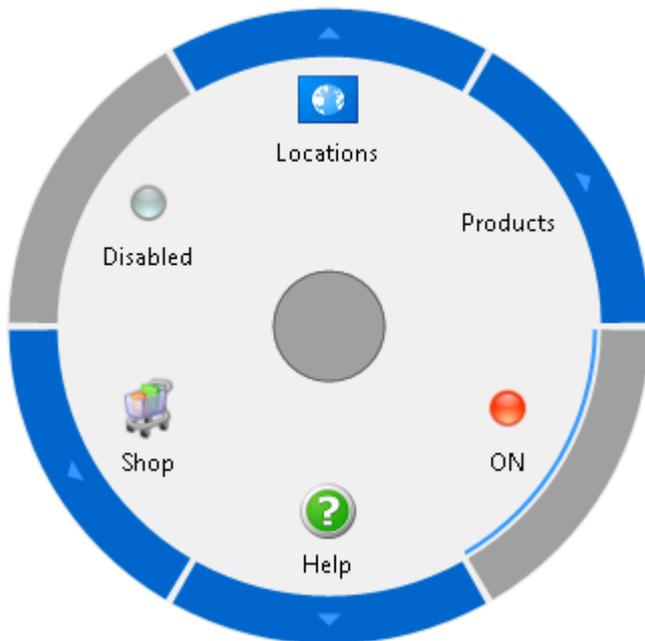
A Radial menu is like a doughnut chart with an inner radius. The inner radius of the Radial menu specifies the radius separating the area filled with the items' background color on the outside from the inner area filled with radial menu's own background. The size of the inner radius can be adjusted using the `C1RadialMenu.InnerRadius` property. The circle in the middle of the Radial menu is the central button. The central button's default size is 28 and can be adjusted using the `C1RadialMenu.ButtonRadius` property.

Hiding the Radial Menu

C1RadialMenu is set to automatically hide whenever it loses focus. To make the C1RadialMenu remain present at all times set the `C1RadialMenu.AutoHide` property to **False**.

Decreasing the Size of the Center Button

The default size of the central button is 28 and appears similar to the following:



To decrease the size of the center button from its default size of 28 to 10, use the following code:

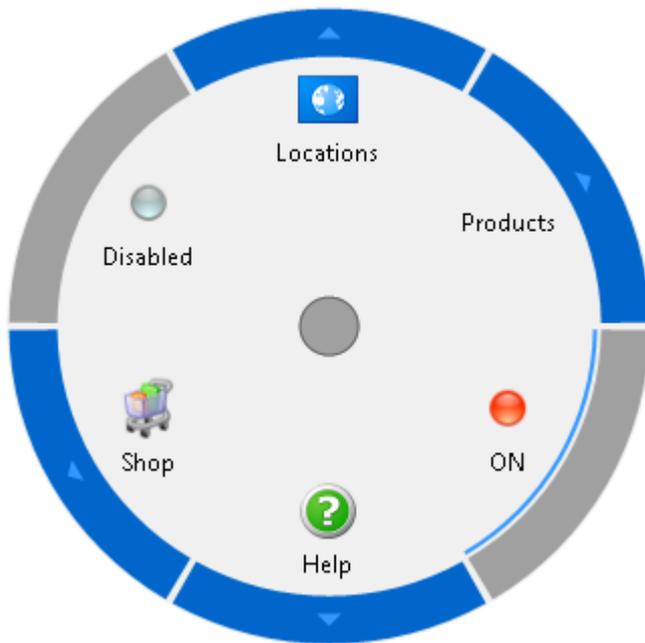
C#

```
c1RadialMenu1.ButtonRadius = 15;
```

Visual Basic

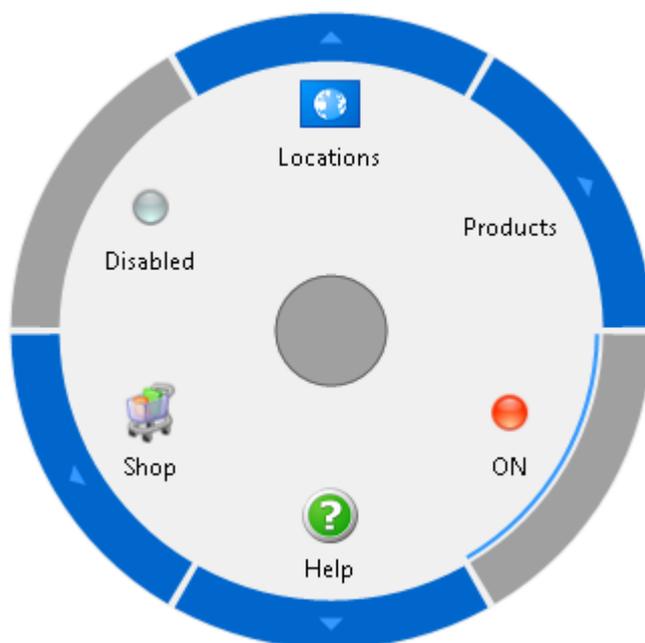
```
c1RadialMenu1.ButtonRadius = 15
```

After the `ButtonRadius` size is decreased from 28 to 15, it will appear similar to the following:



RadialMenu Tutorial

This section provides step-by-step instructions for creating a basic Radial chart. In this tutorial you will learn how to use the Show method to show the Radial menu on the form and how to add RadialMenuItems and sub RadialMenuItems. Once you complete the following steps your Radial menu will appear like the following:



RadialMenu Tutorial Step 1 of 3: Adding the First Menu and Submenu Items

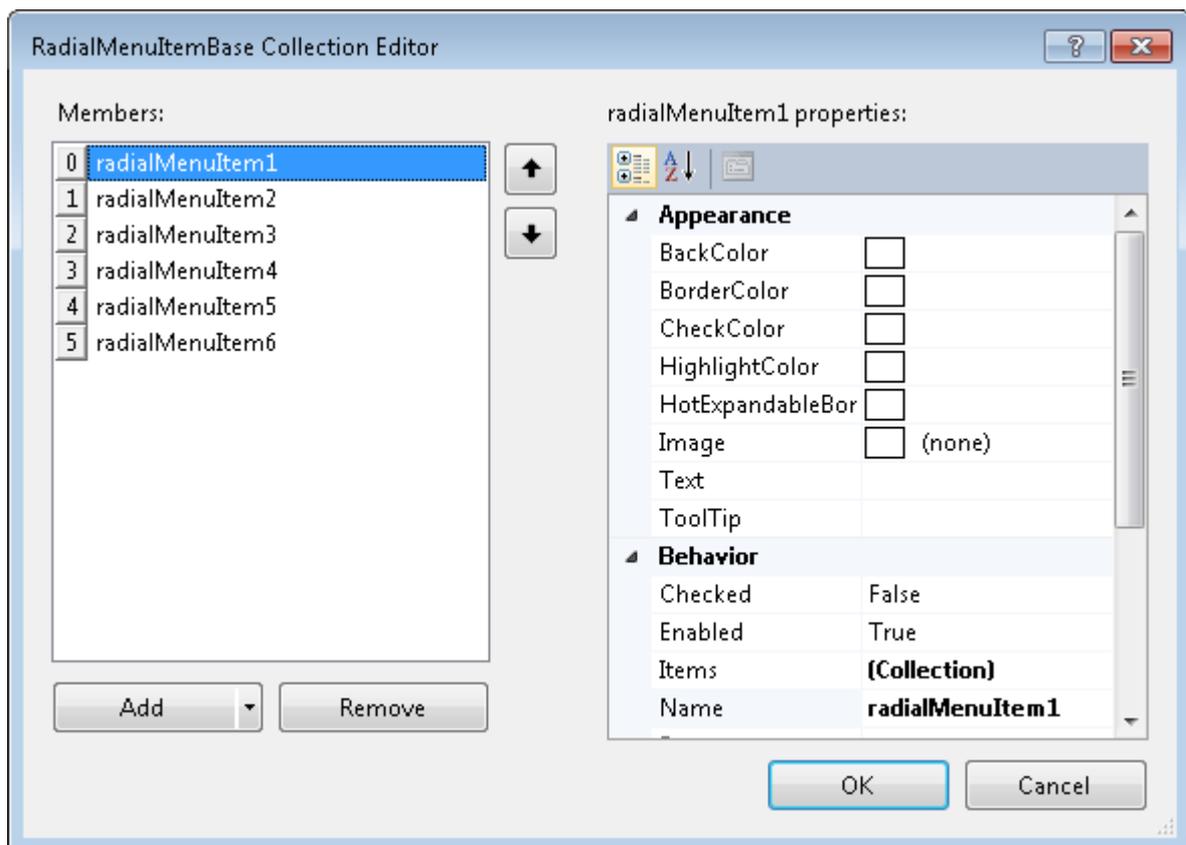
The C1RadialMenu component can be added to forms using the form's designer or programmatically. The Show method will need to be called when using either method, design-time or programmatically, to make C1RadialMenu appear when running the application.

To add RadialMenuItems to C1Radial, complete the following:

1. Create a new windows forms project.
2. Add the **C1RadialMenu** component from the toolbox onto the form.
3. Add the following code to call the **C1Radial.ShowMenu** method to make the C1RadialMenu when you run your application:

C#
<code>c1RadialMenu1.ShowMenu(this, new Point(350, 350));</code>
Visual Basic
<code>c1RadialMenu1.ShowMenu(Me, New Point(350, 350))</code>

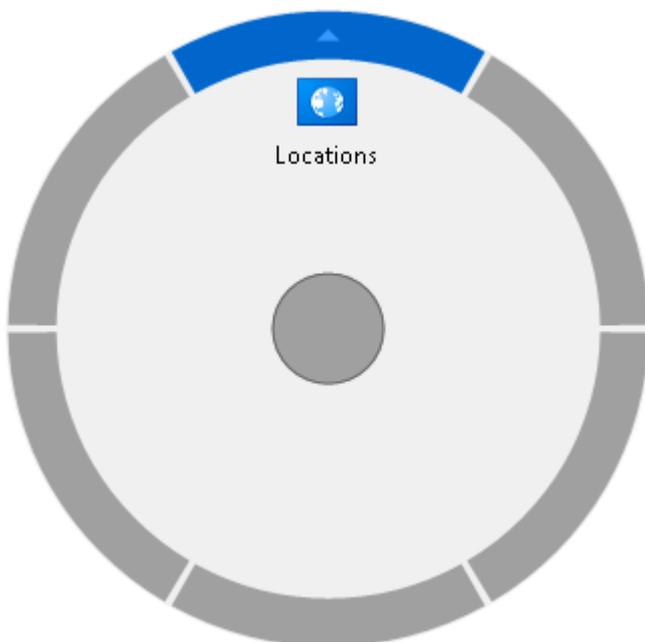
4. Right-click the C1RadialMenu component and select **Properties**.
5. Click on the ellipsis button next to the **Items** property. The **RadialMenuItemBase Collection Editor** appears.
6. Click on the **Add** dropdown button and select **RadialMenuItem**. Repeat this 5 more times so there are 6 RadialMenuItems.



7. Select radialMenuItem1 and set the **Text** property to **Locations**, **ToolTip** to **International and local C1 sites**, and **Name** to **rmiLocale**.
8. Select the ellipsis button next to the **Image** property. The **Select Resource** dialog box appears.
9. Click on the **Import** button to locate the image files you wish to add to the Project resource file.
10. In the **Project resource file:** dropdown listbox select your image for example, **flag_generic**, and click **OK**.
11. Select **rmiLocale** and click on the ellipsis button next to the **Items** property. The **RadialMenuItemBase**

Collection Editor appears.

12. Click on the **Add** dropdown button and RadialMenuItem. Repeat this 6 times. These menu items will be submenu items for radialMenuItem1.
13. Select radialMenuItem7 and set its ToolTip to USA and International and its Name to rmiUS.
14. Select the ellipsis button next to the **Image** property. The **Select Resource** dialog box appears.
15. In the **Project resource file:** dropdown listbox select your image, **flag_usa**, for example, and click **OK**.
16. Select **RadialMenuItem1** and set its **ToolTip** property to **Disabled item** and **Enabled** property to **False**.
17. Select the third menu item, **radialMenuItem8**, and set its **ToolTip** property to **China**, **Name** to **rmiChina**, and **UserData** to **locale**.
18. Select the ellipsis button next to the **Image** property. The **Select Resource** dialog box appears.
19. In the **Project resource file:** dropdown listbox select your image, **flag_china**, for example, and click **OK**.
20. Select **radialMenuItem9** and set its **ToolTip** property to **Japan**, **Name** property to **rmiJapan**, and **UserData** property to **locale**.
21. Select the ellipsis button next to the **Image** property. The **Select Resource** dialog box appears.
22. In the **Project resource file:** dropdown listbox select your image, **flag_japan**, for example, and click **OK**.
23. Select **radialMenuItem10** and set its **ToolTip** property to **India**, **Name** property to **rmiIndia**, and **UserData** to **locale**.
24. Select the ellipsis button next to the Image property. The Select Resource dialog box appears.
25. In the **Project resource file:** dropdown listbox select your image, **flag_india**, for example, and click **OK**.
26. Select **radialMenuItem11** and set its **ToolTip** property to **South Korea**, **Name** property to **rmiSouthKorea**, and **UserData** property to **locale**.
27. Select the ellipsis button next to the **Image** property. The **Select Resource** dialog box appears.
28. In the Project resource file: dropdown listbox select your image, **flag_south_korea**, for example, and click **OK**.
29. Select **radialMenuItem12** and set its **ToolTip** property to **Disabled item** and **Enabled** property to **False**.
30. Click **OK** to save and close the **RadialMenuItemBase Collection Editor**.
31. Run your project and observe the following:



Click or tap on the arrow and another Radial menu will appear with sub RadialMenuItems for the **Locations** RadialMenuItem.

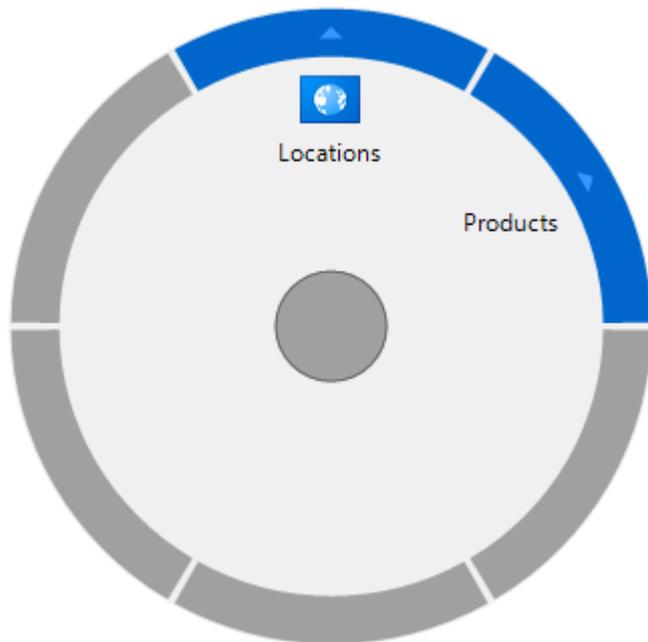


You can navigate back to the previous **Locations** RadialMenuItem by clicking the central back button arrow.

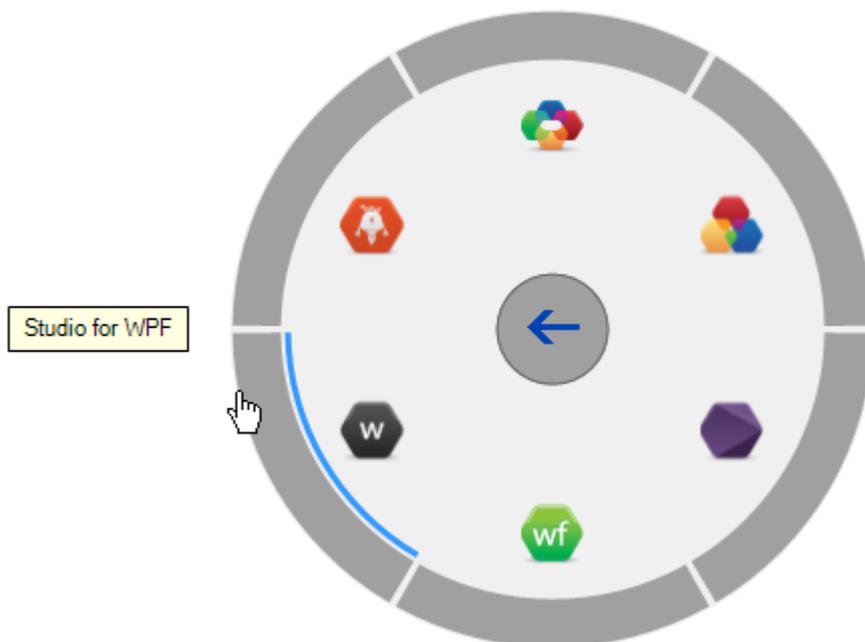
RadialMenu Tutorial Step 2 of 3: Adding the Second Menu and Submenu Items

In this step, you will add submenu items to the existing Products menu.

1. In the RadialMenu properties window click on the ellipsis button next to the **Items** property. The **RadialMenuBaseItem Collection Editor** appears.
2. Select **radialMenuItem2** from the Members list and set its Name property to **rmiProducts** and Text property to **Products**.
3. Click on ellipsis button next to the **Items** property. The **RadialMenuBaseItem Collection Editor** appears.
4. Click on the **Add** dropdown button and select **RadialMenuItem**. Repeat this so there are 6 RadialMenuItems.
5. Select the first member, **radialMenuItem2**, and set its **Image** property to **ultLogo_32**, **ToolTip** property to **Studio Ultimate**, and **Name** property to **rmiStuUlt**.
6. Select the second member, **radialMenuItem7**, and set its **Image** property to **seLogo_32**, **ToolTip** property to **Studio Enterprise**, and **Name** property to **rmiStuEnt**.
7. Select the third member, **radialMenuItem8**, and set its **Image** property to **c1powersuite_logo_32**, **ToolTip** property to **Enterprise-ready reporting and spreadsheets controls for .NET applications**, and its **Name** property to **rmiStuPower**.
8. Select the fourth member, **radialMenuItem9**, and set its **Image** property to **winformsLogo_321**, **ToolTip** property to **Studio for WinForms**, and its **Name** property to **rmiStuWinForms**.
9. Select the fifth member, **radialMenuItem10**, and set its **Image** property to **wpfLogo_32**, **ToolTip** property to **Studio for WPF**, and its **Name** property to **rmiStuWPF**.
10. Select the sixth member, **radialMenuItem11**, and set its **Image** property to **aspLogo_32**, **ToolTip** property to **Studio for ASP.NET Wijmo**, and **Name** property to **rmiStuWijmo**.
11. Click **OK** to save and close the **RadialMenuItemBase Collection Editor**.
12. Click **OK** to save and close the **RadialMenuItemBase Collection Editor**.
13. Run your project and observe the following:



Notice the second RadialMenuItem, Products, is added. Click on the **Products** menu and its submenu items appear like the following:



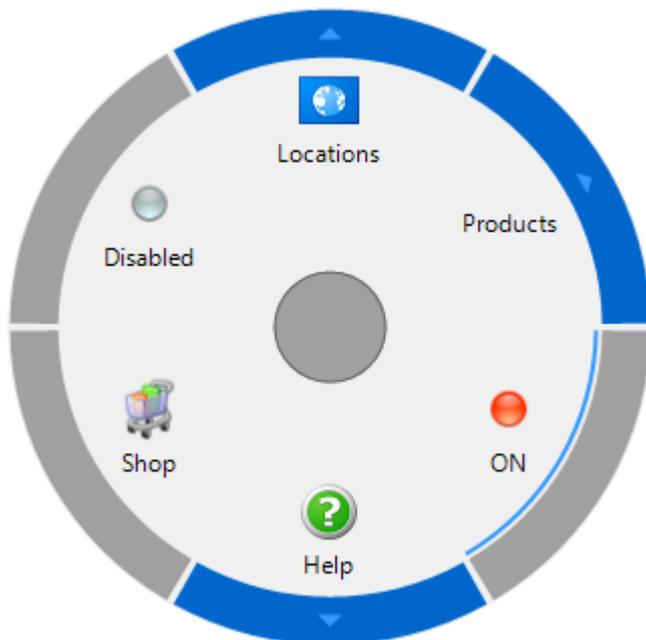
Notice when you hover over a menu item its tooltip appears. To navigate back to the main Radial menu, click on the back button.

RadialMenu Tutorial Step 3 of 3: Adding the Remaining RadialMenuItems

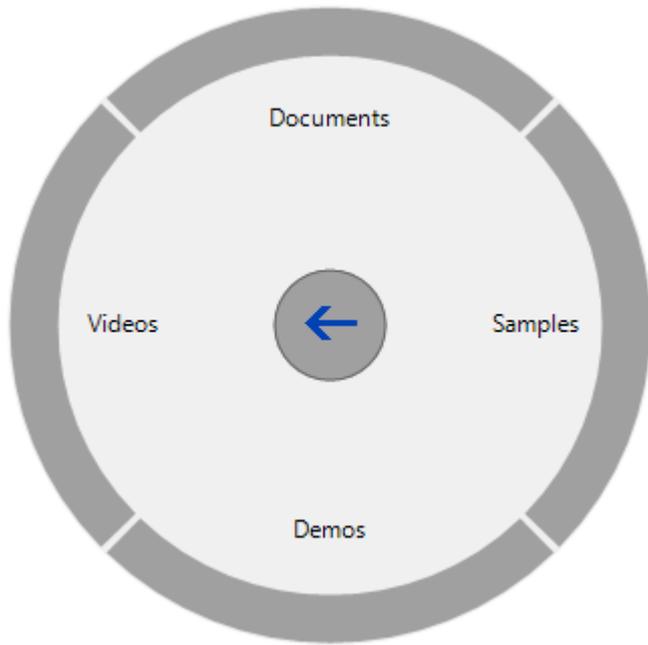
In this step, you will add submenu items to the existing Products menu.

1. In the C1RadialMenu properties window click on the ellipsis button next to the **Items** property. The **RadialMenuBaselItem Collection Editor** appears.
2. Select **radialMenuItem3** from the **Members list** and set its **Image** property to **bullet_ball_glass_red**, **Text**

- property to **ON**, **Checked** property to **True**, **Name** property to **rmiCheck**, and **UserData** property to **Check**.
3. Select **radialMenuItem4** from the **Members list** and set its **Image** property to **Help**, **Text** property to **Help**, **ToolTip** property to **Get help**, and **Name** property to **rmiHelp**.
 4. Click on ellipsis button next to the **Items** property. The **RadialMenuBaseItem Collection Editor** appears.
 5. Click on the **Add** dropdown button and select **RadialMenuItem**. Repeat this so there are 4 RadialMenuItems.
 6. Select **radialMenuItem2** from the **Members list** and set its **Text** property to **Documents** and **Name** property to **rmiDocs**.
 7. Select **radialMenuItem3** from the **Members list** and set its **Text** property to **Samples** and **Name** to **rmiSamples**.
 8. Select **radialMenuItem4** from the **Members list** and set its **Text** property to **Demo** and its **Name** property to **rmiDemos**.
 9. Select **radialMenuItem5** from the **Members list** and set its **Text** property to **Demos** and its **Name** property to **rmiDemos**.
 10. Select **radialMenuItem6** from the **Members list** and set its **Text** property to **Videos** and **Name** property to **rmiVideos**.
 11. Click **OK** to save and close the **RadialMenuItemBase Collection Editor**.
 12. In the **RadialMenuItemBase Collection Editor** select **radialMenuItem5** from the **Members list** and set its **Image** property to **shoppingcart_full1**, **Text** property to **Shop**, **ToolTip** property **Shop C1**, **Enabled** property to **False**, and **Name** property to **rmiStore**.
 13. Select **radialMenuItem6** from the **Members list** and set its **Image** property to **bullet_ball_glass_grey**, **Text** property to **Disabled**, **ToolTip** property to **Disabled Item**, and **Enabled** property to **False**.
 14. Click **OK** to save and close the **RadialMenuItemBase Collection Editor**.
 15. Run your project and observe the following:



The disabled menus appear with a grey border. The checked menu, ON, appears with a thin blue border around the inner edge of grey border. Clicking the **Help** button opens a new Radial menu with submenu items for the Help menu.



Menus and Toolbars 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

Click one of the following links to view a list of **C1Command** samples:

Visual Basic Samples

Sample	Description
CreateMenusInCode	Shows how to create and wire up context menus in code. Creates a C1ContextMenu for a text box completely in code, on a form that does not have any other C1Command elements. Wires up click event handlers for the menu items. Also sets up state query handlers for the commands so that their state (enabled and so on.) is kept up to date automatically by C1Command. This sample calls the C1.Win.C1Command namespace.
ProgramOutBar	This sample starts with a form containing a C1OutBar with two empty pages (created at design time). The form also contains an empty command holder. In the form load handler, several commands are created on the command holder. Two toolbars are also created and filled with links to the commands. The toolbars are placed on the outbar pages.
SimpleMenusInCode	This sample shows how to create menus and toolbars by customizing their appearance and attaching event handlers to them in code. The sample app is a form with a rich text editor control. The form load event handler creates C1Command commands to load a text file, create a new form, or exit the application. It also creates a main menu and a toolbar that allow the user to invoke those commands.
SimpleTextEditor	Shows the basic usage of C1Command menus and toolbars. Builds a rich text box-based text editor, with C1Command menus and floating toolbars. The toolbars are end user-customizable. Their layout and customizations are automatically saved by C1Command in the application config file. This sample uses C1ToolBar , C1ContextMenu , C1CommandMenu , C1Command , C1MainMenu , C1CommandLink , C1CommandDock , and C1CommandHolder controls.
SimpleTextEditor2	Shows how to add an arbitrary control to a toolbar item. This sample is similar to SimpleTextEditor, but additionally includes a combo box nested in a movable and dockable C1ToolBar , implementing a simple text search. This sample uses C1ToolBar , C1CommandMenu , C1CommandControl , C1CommandDock , C1CommandHolder , C1CommandLink , C1MainMenu , C1Command , and C1ContextMenu controls.

C# Samples

Sample	Description
CreateMenusInCode	Shows how to create and wire up context menus in code. Creates a C1ContextMenu for a text box completely in code, on a form that does not have

	any other C1Command elements. Wires up click event handlers for the menu items. Also sets up state query handlers for the commands so that their state (enabled and so on.) is kept up to date automatically by C1Command . This sample calls the C1.Win.C1Command namespace.
MdiTabs	This sample shows how to use C1DockingTab to create a Visual Studio-like multiple window environments.
NonMdiMenuMerge	Shows how to merge menus using methods provided by C1Command . This sample contains two forms, both with C1MainMenu 's on them. One form is created at application startup, and it is the main form of the app. It has a command to create the other form. When that form is created, its main menu is merged programmatically into the main menu of the first form, in the same way as MDI child menus are merged in Windows. This sample uses C1CommandLink , C1MainMenu , C1Command , C1CommandMenu , and C1CommandHolder controls.
SelectMdiChildForm	This sample shows how to provide a customized MDI child form selector window. The sample project contains an inherited form, which is based on C1SelectMdiChildForm . The custom selector form overrides the default colors, and in addition to the two default buttons (ok and cancel), provides a new button close, which allows the selected window to close. This sample also shows how to invoke the selector form from code.
SimpleMenusInCode	This sample shows how to create menus and toolbars by customizing their appearance and attaching event handlers to them in code. The sample app is a form with a rich text editor control. The form load event handler creates C1Command commands to load a text file, create a new form, or exit the application. It also creates a main menu and a toolbar that allow the user to invoke those commands.

Menus and Toolbars for WinForms Task-Based Help

The task-based help section assumes that you are familiar with programming in the Visual Studio environment, and know how to use the controls in general.

Each topic provides a solution for specific tasks using the **Menus and Toolbars for WinForms** product.

Each task-based help topic also assumes that you have created a new .NET project.

Menu Tasks

This section shows how to perform specific menu tasks.

Adding a Menu Item to MainMenu

You can add a menu item to [C1MainMenu](#) at design time by adding the [C1MainMenu](#) component and then appending menu items to it using the Edit designer (Menu designer). Menus can also be added programmatically by adding [C1CommandHolder](#) object to the Windows form to hold the menus, add the [C1MainMenu](#) object, and then create the command type for the menu. This topic shows how to create a menu item, called **menu 1** that can hold future submenu items.

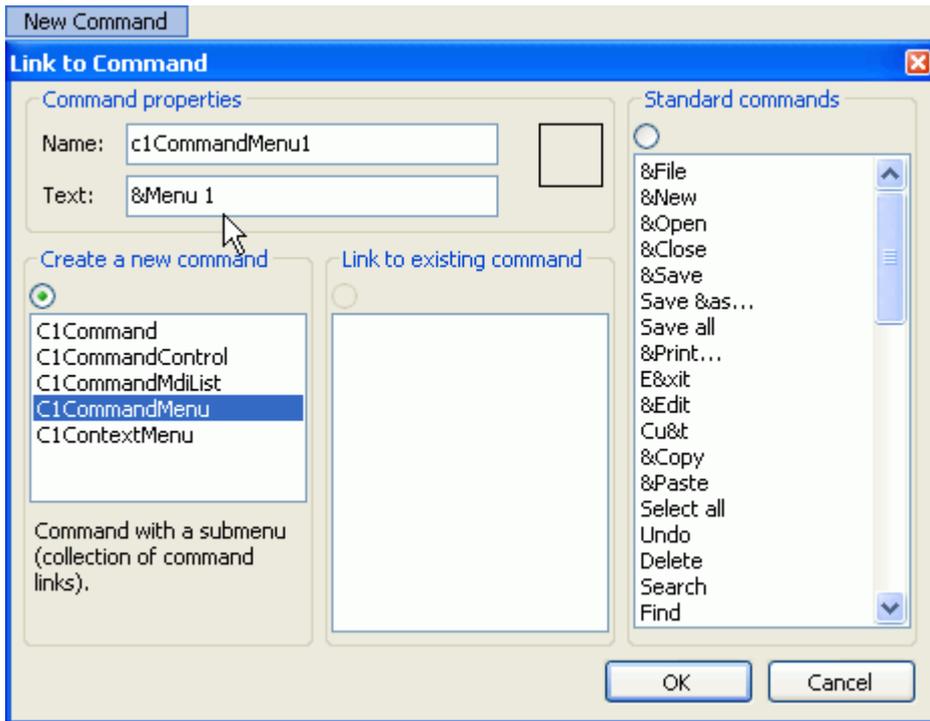


Note: **C1MainMenu** is a control that represents the main menu. It contains a collection of command links that represent items of the menu. Only one main menu can be added to the form. **C1CommandMenu** is a command that is a menu.

To add a menu item to C1MainMenu at design time

To add a menu item to [C1MainMenu](#) using the **Link to Command** designer, complete the following:

1. Place a [C1MainMenu](#) on the form by performing a drag-and-drop operation. A [C1CommandHolder](#) automatically appears on the component tray below the form.
2. Right-click the text, **New Command**, and select **Edit** from its context menu. The **Link to Command** designer appears.
3. In the **Link to Command** designer, select the **Text** field and enter **&Menu 1**.



4. Select **OK**. The new menu (Menu 1) appears. Here is how the menu looks on your form at design time:



To add a menu item to C1MainMenu programmatically

To add the menu item to `C1MainMenu` programmatically, complete the following steps:

1. Add the **C1.Win.C1Command** namespace to your references in your project.
2. Double-click the form to create a **Form_Load** event, declare the namespace in your source file, and then add a **C1CommandHolder** to hold the menu.

To write code in Visual Basic

Visual Basic

```
Imports C1.Win.C1Command
Dim ch As C1CommandHolder = C1CommandHolder.CreateCommandHolder(Me)
```

To write code in C#

C#

```
using C1.Win.C1Command;
C1CommandHolder ch = C1CommandHolder.CreateCommandHolder(this);
```

3. Create a new main menu, then add the main menu control to your form.

To write code in Visual Basic

Visual Basic

```
Dim mm As New C1MainMenu
Me.Controls.Add(mm)
```

To write code in C#

```
C#
C1MainMenu mm = new C1MainMenu();
this.Controls.Add(mm);
```

4. Create a submenu to hold commands, then set the text property for the new menu.

To write code in Visual Basic

```
Visual Basic
Dim mmenu As C1CommandMenu = CType(ch.CreateCommand(GetType(C1CommandMenu)),
C1CommandMenu)
mmenu.Text = "&menu1"
```

To write code in C#

```
C#
C1CommandMenu mmenu = ch.CreateCommand(typeof(C1CommandMenu)) as C1CommandMenu;
mmenu.Text = "&menu1";
```

5. Add the command link to the new submenu.

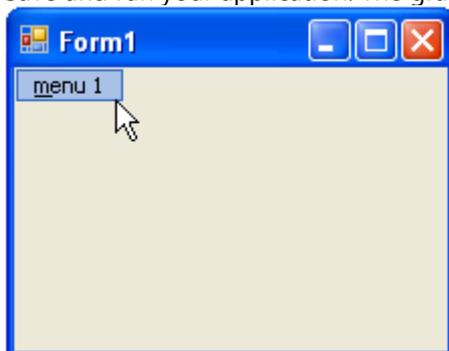
To write code in Visual Basic

```
Visual Basic
mm.CommandLinks.Add(New C1CommandLink(mmenu))
```

To write code in C#

```
C#
mm.CommandLinks.Add(new C1CommandLink(mmenu));
```

6. Save and run your application. The graphic shows your new menu item on the form at run time:



Adding an Icon to a Menu Item

You can add an icon to a menu item at design time or through code.

To add an icon to a menu item programmatically

Use the following code to add an icon to a menu:

To write code in Visual Basic

Visual Basic

```
'Create a new instance of a Bitmap and assign it to the Image property of the Command Object  
c1Command1.Image = new System.Drawing.Bitmap(@"D:\componentOne\Images\App.ico')
```

To write code in C#

C#

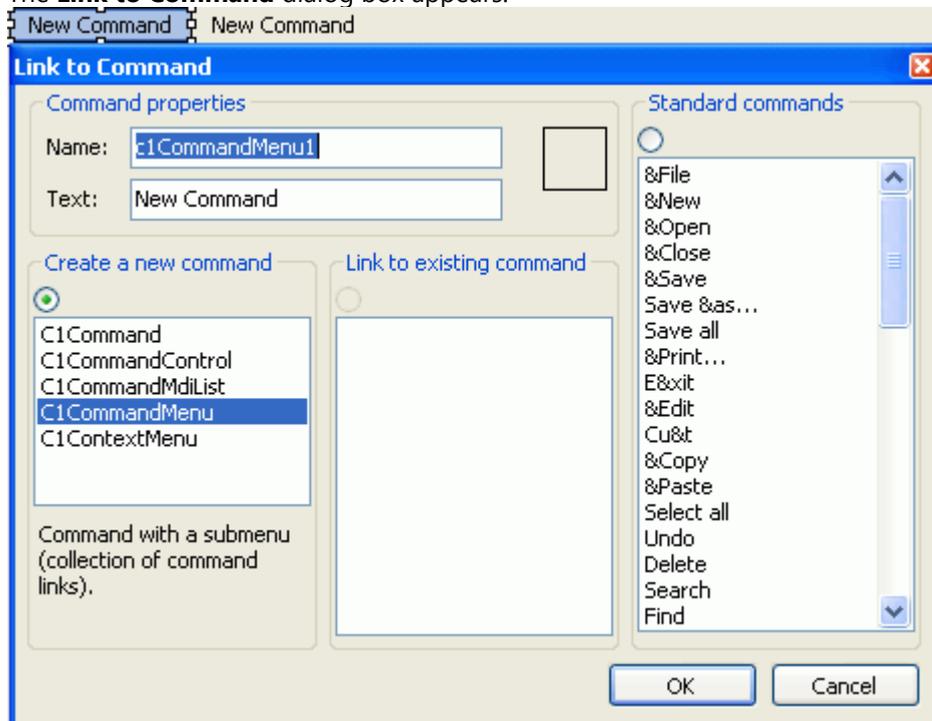
```
//Create a new instance of a Bitmap and assign it to the Image property of the Command Object  
c1Command1.Image = new System.Drawing.Bitmap(@"D:\componentOne\Images\App.ico");
```

Adding a Menu Item Before the Current Menu Item

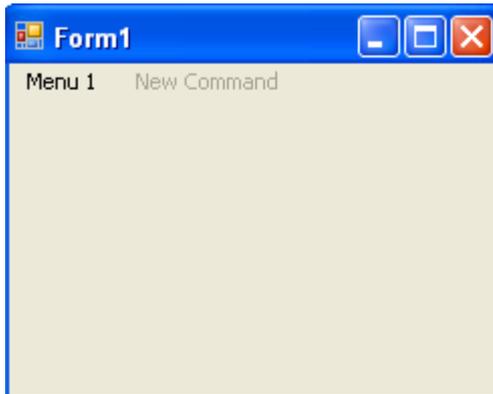
To insert a menu item at design time, select the **Insert Item** option from **C1MainMenu**'s context menu. The **Insert Item** command inserts a new menu item before the current one. To insert a menu item at design time, complete the following steps:

1. Place the **C1MainMenu** control on your form using a drag-and-drop operation.
2. Right-click on **C1MainMenu** control and select, **Insert Item** from its context menu.

The **Link to Command** dialog box appears:



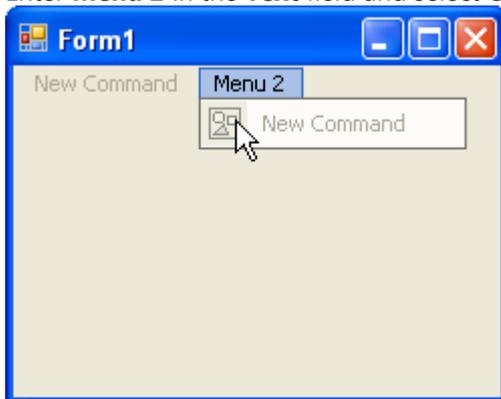
3. Enter **Menu 1** in the **Text** field and select **OK**. The graphic shows the menu item on your form at design time:



Adding a Menu Item After the Current Menu Item

To add a new menu item at design time after the **last one** in the current menu, complete the following steps:

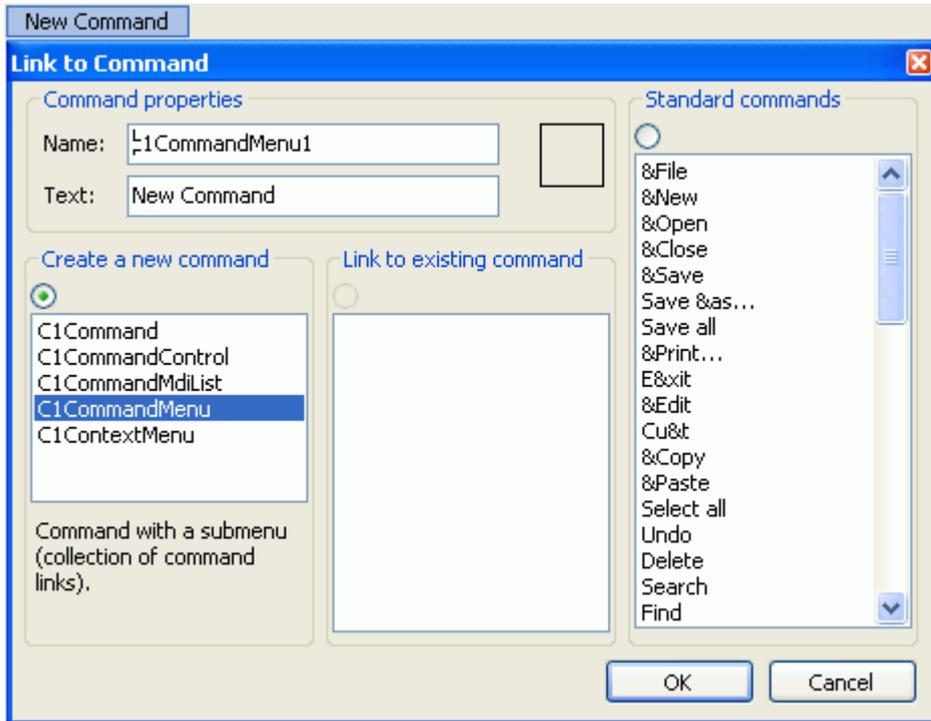
1. Place a [C1MainMenu](#) component on your form using a drag-and-drop operation.
2. Right-click on [C1MainMenu](#) and select **Append Item** from the context menu. The **Link to Command** designer appears:
3. Enter **Menu 2** in the **Text** field and select **OK**. The graphic shows the menu item on your form at design time:



Adding a Standard Menu Item from the Link to Command Designer

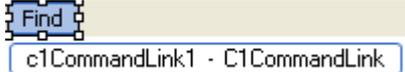
To add a standard command item with a built-in image from the **Link to Command** designer, complete the following steps:

1. Click the empty command item twice from the [C1MainMenu](#) control. The **Link to Command** designer appears.



2. Select one of the standard commands from the listbox, for example, **Find**.
3. Click **OK**.

The new command item and its built-in image appears in the `C1MainMenu` control on the form.



4. Select the menu, **Find**, and select the **C1CommandLink Properties** button from its toolbar. The dialog box for the `C1CommandLink` properties to appear.

Note: The toolbar for the command item will not appear if the Smart Designer is not enabled.

5. Click on the drop-down arrow for the **Button look** drop-down box and select **TextAndImage**. The built-in image appears next to the text, Find.



Note: The same procedure can be applied when adding a standard command item to the `C1ToolBar` control.

Adding a Submenu

You can add a submenu through the designer or through code. Click on either of the following links to expand the steps for the designer or for the code.

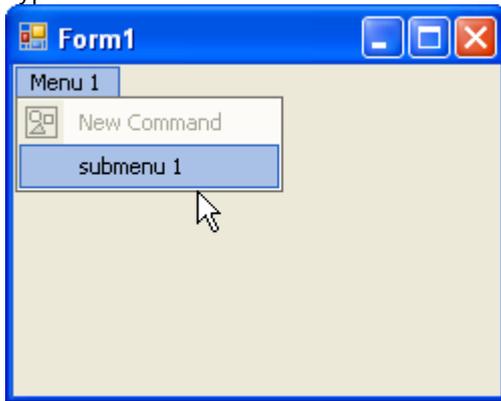
To add a submenu item to `C1MainMenu` at design time

To add a new command link to the menu (`C1CommandMenu` or `C1ContextMenu`) linked by the currently selected command link, complete the following steps:

Note: This item is disabled unless the current command link is linked to a `C1CommandMenu` or `C1ContextMenu` type command.

1. Right-click on an existing menu item in the `C1MainMenu` and select **Add Child Item** from the context menu. The **Link to Command** designer appears.

2. Type **submenu 1** in the **Text** textbox field and select **OK**. The menu appears like the following menu:



To add a submenu item to C1MainMenu programmatically

To programmatically add a submenu item, complete the following steps:

1. Add the C1.Win.C1Command namespace to your references in your project.
2. Declare the namespace in your source file.

To write code in Visual Basic

```
Visual Basic
Imports C1.Win.C1Command
```

To write code in C#

```
C#
using C1.Win.C1Command;
```

3. Double-click the form to create a Form_Load event handler, then insert the following code snippets from the remaining steps into the Form_Load event handler.
4. Add a C1CommandHolder to hold the menu, then create a new main menu.

To write code in Visual Basic

```
Visual Basic
Dim ch As C1CommandHolder = C1CommandHolder.CreateCommandHolder(Me)
Dim mm As New C1MainMenu
```

To write code in C#

```
C#
C1CommandHolder ch = C1CommandHolder.CreateCommandHolder(this);
C1MainMenu mm = new C1MainMenu();
```

5. Add the main menu control to your form, create the main menu to hold the commands, and then set the text property for the new menu.

To write code in Visual Basic

```
Visual Basic
Me.Controls.Add(mm)
```

```
Dim mmenu As C1CommandMenu = CType(ch.CreateCommand(GetType(C1CommandMenu)),
C1CommandMenu)
mmenu.Text = "Menu 1"
```

To write code in C#

C#

```
this.Controls.Add(mm);
C1CommandMenu mmenu = ch.CreateCommand(typeof(C1CommandMenu)) as C1CommandMenu;
mmenu.Text = "Menu 1";
```

6. Add the commandlink to the new main menu, then create and set up a menu item under menu 1. Fill the menu with a command.

To write code in Visual Basic

Visual Basic

```
mm.CommandLinks.Add(New C1CommandLink(mmenu))
Dim submenu As C1Command = ch.CreateCommand()
```

To write code in C#

C#

```
mm.CommandLinks.Add(new C1CommandLink(mmenu));
C1Command submenu = ch.CreateCommand();
```

7. Add text to the new submenu item and add a new c1commandlink to the submenu item.

To write code in Visual Basic

Visual Basic

```
submenu.Text = "submenu 1"
'add a new c1commandlink to the submenu item
mmenu.CommandLinks.Add(New C1CommandLink(submenu))
```

To write code in C#

C#

```
submenu.Text = "submenu 1";
//add a new c1commandlink to the submenu item
mmenu.CommandLinks.Add(new C1CommandLink(submenu));
```

Your menu should appear like the following menu:



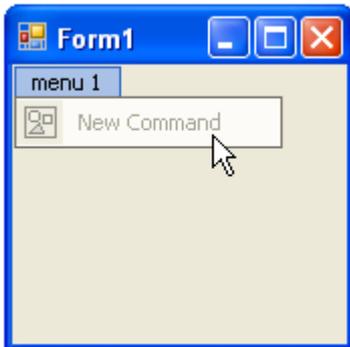
Adding Multiple SubMenus

You can add a submenu and another menu within the submenu through the designer or through code. Click on either of the following links to expand the steps for the designer or for the code.

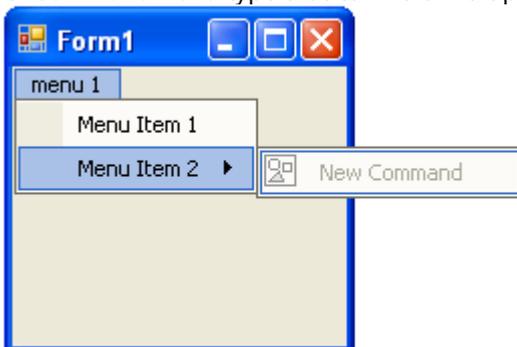
To add multiple submenu items at design time

To add a submenu and another menu within the submenu at design time, complete the following steps:

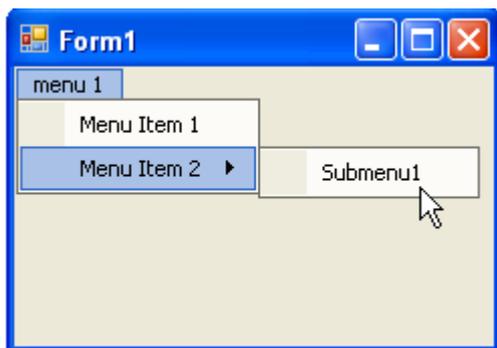
1. Place the [C1MainMenu](#) control on your form using a drag-and-drop operation.
2. Right-click on the [C1MainMenu](#) and select **Edit** from its context menu. The **Link to Command** designer appears.
3. In the **Text** textbox enter **menu 1** and click **OK**. The menu appears like the following:



4. Click on the **New Command** and select **Edit** from its context menu. In the **Text** textbox enter **Menu Item 1** and select **OK**.
5. Click on **menu 1** on your form.
6. Click on **Menu Item 1** on your form and select **Append Item** from its context menu. The **Link to Command** designer appears.
7. In the **Command Text** box enter **Menu Item 2**, and select **OK**. The menu appears like the following:
8. Click on **Menu Item 2** on your form and select **Edit** from its context menu. The **Link to Command** designer appears.
9. In the **Command Type** box select **C1CommandMenu**, and then select **OK**. The **Menu Item 2** is a [C1CommandMenu](#) type that can hold multiple menus inside it.



10. Select the **New Command** menu item and select **Edit** from its context menu.
11. In the **Text** textbox enter **Submenu1** and click **OK**. The menu appears like the following image:



To add multiple submenus programmatically

To programmatically add a submenu and another menu within the submenu, complete the following steps:

1. Add the **C1.Win.C1Command** namespace to your references in your project, and then declare the namespace in your source file.

To write code in Visual Basic

Visual Basic

```
Imports C1.Win.C1Command
```

To write code in C#

C#

```
using C1.Win.C1Command;
```

2. Double-click the form to create a **Form_Load** event handler, then insert the following code snippets from the remaining steps into the **Form_Load** event handler.
3. Add a **C1CommandHolder** to hold the menu, then create a new **C1MainMenu** object.

To write code in Visual Basic

Visual Basic

```
Dim ch As C1CommandHolder = C1CommandHolder.CreateCommandHolder(Me)
Dim mm As New C1MainMenu
```

To write code in C#

C#

```
C1CommandHolder ch = C1CommandHolder.CreateCommandHolder(this);
C1MainMenu mm = new C1MainMenu();
```

4. Add the main menu control to your form, then create the main menu to hold commands.

To write code in Visual Basic

Visual Basic

```
Me.Controls.Add(mm)
Dim mmenu As C1CommandMenu = CType(ch.CreateCommand(GetType(C1CommandMenu)),
C1CommandMenu)
```

To write code in C#

C#

```
this.Controls.Add(mm);
C1CommandMenu mmenu = ch.CreateCommand(typeof(C1CommandMenu)) as C1CommandMenu;
```

- Set the text property for the new menu, then add the commandlink to the new main menu.

To write code in Visual Basic

Visual Basic

```
mmenu.Text = "&menu 1"
mm.CommandLinks.Add(New C1CommandLink(mmenu))
```

To write code in C#

C#

```
mmenu.Text = "&menu 1";
mm.CommandLinks.Add(new C1CommandLink(mmenu));
```

- Create and set up a menu item under the menu (**menu 1**), then fill the menu with a command.

To write code in Visual Basic

Visual Basic

```
Dim menuitem1 As C1Command = ch.CreateCommand()
```

To write code in C#

C#

```
C1Command menuitem1 = ch.CreateCommand();
```

- Add text to the new menu item and add a new c1commandlink to menuitem1.

To write code in Visual Basic

Visual Basic

```
menuitem1.Text = "Menu Item 1"
'add a new c1commandlink to the menuitem1
mmenu.CommandLinks.Add(New C1CommandLink(menuitem1))
```

To write code in C#

C#

```
menuitem1.Text = "Menu Item 1";
//add a new c1commandlink to the menuitem1
mmenu.CommandLinks.Add(new C1CommandLink(menuitem1));
```

- Create a new command menu for the second menu item below the menu (**menu1**), and then add a commandlink for the new menuitem2 menu. **Menuitem2** is going to be a menu that contains submenus.

To write code in Visual Basic

Visual Basic

```
Dim menuItem2 As C1CommandMenu = New C1CommandMenu()
menuItem2.Text = "Menu Item 2"
menu.CommandLinks.Add(New C1CommandLink(menuItem2))
```

To write code in C#

C#

```
C1CommandMenu menuItem2 = new C1CommandMenu();
menuItem2.Text = "Menu Item 2";
mmenu.CommandLinks.Add(new C1CommandLink(menuItem2));
```

9. Create a submenu for the new **menuItem2** menu and call it **Submenu 1**, then add a commandlink for submenu1.

To write code in Visual Basic

Visual Basic

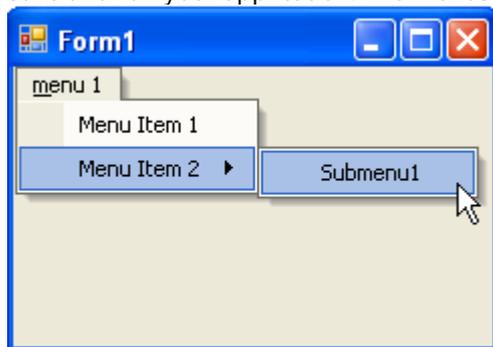
```
Dim submenu1 As C1Command = ch.CreateCommand()
submenu1.Text = "Submenu1"
menuItem2.CommandLinks.Add(New C1CommandLink(submenu1))
```

To write code in C#

C#

```
C1Command submenu1 = ch.CreateCommand();
submenu1.Text = "Submenu1";
menuItem2.CommandLinks.Add(new C1CommandLink(submenu1));
```

10. Save and run your application. The menus appear like the following at run time:



Applying ShortCut Keys to Menus

This topic demonstrates how to use shortcut and mnemonic keys to access a menu by the keyboard instead of the mouse. You can apply shortcut and mnemonic keys to a menu item at design time or through code.

To add a shortcut key to a menu item at design time

1. Add a [C1MainMenu](#) control to your form.
2. Right-click the [C1MainMenu](#) control, select **Edit** from its context menu, and click **Ok**. The default name for the menu is **New Command**.

- Right-click the **New Command** item and select **Properties** from its context menu. In the Properties window for the **C1CommandLink**, expand the **Command** node, select the **Shortcut** property, and then select **F6** from the Shortcut list box.

 **Note:** The remaining steps are listed below for you to try and see how the short cut key functions.

- Add a **PictureBox** control to your form.
- Double-click **New Command** to create an event handler for the click event for **New Command**.

Within the event handler insert code to change the back color of the **PictureBox** to violet:

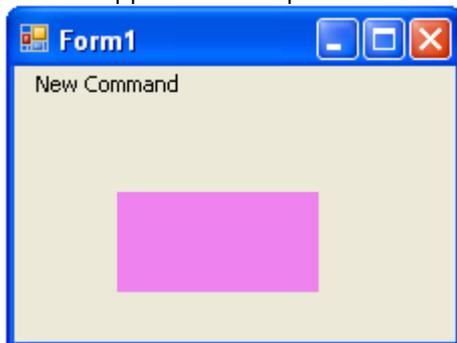
To write code in Visual Basic

```
Visual Basic
If sender Is c1CommandMenu1 Then
    Me.pictureBox1.BackColor = Color.Violet
End If
```

To write code in C#

```
C#
if (sender == c1CommandMenu1)
{
    this.pictureBox1.BackColor = Color.Violet;
}
```

- Run the application and press the F6 key when the form appears. The picture box appears on the form.



To add a shortcut key to a menu item programmatically

- In your source file locate the item in the menu where you want the shortcut key to exist.
- Create a shortcut key for the **C1CommandMenu1** by entering the following code:

To write code in Visual Basic

```
Visual Basic
Me.C1CommandMenu1.Shortcut = System.Windows.Forms.Shortcut.F6
```

To write code in C#

```
C#
this.C1CommandMenu1.Shortcut = System.Windows.Forms.Shortcut.F6;
```

Localizing the Text for the Shortcut Key

To localize the text for the Shortcut Key, complete the following steps:

1. Add a C1MainMenu control to your form.
2. Right-click on the control and select **Edit** from the context menu.
3. Click **OK** in the **Link to Command** dialog box.
4. Select **C1CommandLink2** from the **Properties** drop-down list.
5. Right-click on the **New Command** and select **Edit** from the context menu. Enter **File** in the text textbox and then click **OK**.
6. In the Properties window, locate and expand the **C1CommandLink2.Command** node.
7. Select the **C1CommandLink2.Shortcut** property and choose **F3** from the drop-down list.
8. Select the **C1CommandLink2.ShortcutText** property and set it to **F3**, for this topic.

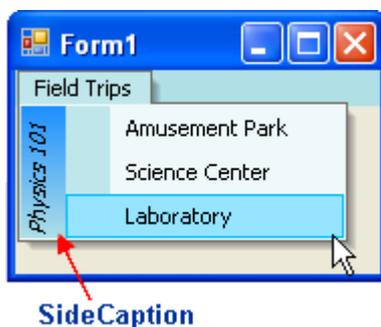
The text for the shortcut will appear as in the following image:



Creating a Side Caption for a Command Menu

To use the **SideCaption** property at design time to display a side caption along the commands, complete the following steps:

1. Select a **C1CommandMenu** from the **Properties** drop-down list.
2. Locate and expand the **C1CommandMenu.SideCaption** node.
3. Select the **BarGradientBegin** property and select **Dodger Blue** from its **Web** tab.
4. Select the **BarGradientEnd** property and select **PowderBlue** from its **Web** tab.
5. Set **Text** property to **Physics 101**.
6. The side caption appears to the left side of the menu items. The graphic illustrates the appearance of the side caption:



Note: The text displayed in the side caption represents the name of the class that participated in the field trips.

Creating a Separate Click Event for a Command Object

To assign an address of a method to the **Click** event of [C1Command](#), with the same signature as the **ClickEventHandler** delegate, use the **AddHandler** method. To do this, use the following code:

To write code in Visual Basic

Visual Basic

```
Dim ch As C1CommandHolder = C1CommandHolder.CreateCommandHolder(Me)
'create commands for file ops.
Dim cNew as C1Command = ch.CreateCommand()
'Use the AddHandler method to assign a delegate for the click event.
AddHandler cNew.Click, New ClickEventHandler(AddressOf cNew_Click)
```

To write code in C#

C#

```
C1CommandHolder ch = C1CommandHolder.CreateCommandHolder(this);
//create commands for file ops.
C1Command cNew = ch.CreateCommand();
//Use the AddHandler method to assign a delegate for the click event.
cNew.Click += new C1.Win.C1Command.ClickEventHandler(cNew_Click);
```

Creating a Window List for an MDI Form

[C1Command](#) has a [C1CommandMdiList](#) command type used to create a Window list for an MDI (Multiple Document Interface) form. A Window list is useful for keeping track of the various MDI child windows an application has open.

To create a Window list for an MDI form, complete the following steps:

1. In the Properties window, set the `Form1.IsMDIContainer` property to **True**.
2. In Solution Explorer, right-click the project and select **Add | Add New Item**.
3. From the dialog box, select **Windows Form**, name the form **MdiChild**, and then set its **Text** property to **MdiChild**.
4. Add the [C1MainMenu](#) component to your MDI parent form, **Form1**.
5. In the Properties window, set the **Name** property for the [C1MainMenu](#) to **C1MainMenu1**.
6. Create the **File** menu. Add a [C1CommandMenu](#) and a submenu item to the [C1MainMenu](#) component using the **Link to Command** designer.

Set the following properties:

Command Name	Command Type	Command Text
cmdFile	C1CommandMenu	&File
cmdFileNew	C1Command	&New

7. Create the **Window** menu to store the MDI child windows. In the **Link to Command** designer, add another [C1CommandMenu](#) to the [C1MainMenu](#) component, then add a submenu of the type [C1CommandMdiList](#) to it.

Set the following properties:

Command Name	Command Type	Command Text
cmdWindow	C1CommandMenu	&Window
c1CommandMdiList1	C1CommandMdiList	<MDI Windows List>

8. Create a procedure to display new instances of the **MdiChild** form as MDI children of **Form1**. Add the following code to your source file:

To write code in Visual Basic

```
Visual Basic

Private Sub createNewMdiChild()
    Dim mc As New MdiChild()
    mc.MdiParent = Me
    mc.Text = String.Format("MDI Child Window {0}", Me.MdiChildren.Length)
    mc.Show()
End Sub
```

To write code in C#

```
C#

private void createNewMdiChild()
{
    MdiChild mc = new MdiChild();
    mc.MdiParent = this;
    mc.Text = string.Format("MDI Child Window {0}",
this.MdiChildren.Length);
    mc.Show();
}
```

9. In Design view, double-click the **cmdFileNew** menu item to create a click event handler to call the **createNewMdiChild** procedure. Add the following code within the event handler:

To write code in Visual Basic

```
Visual Basic

createNewMdiChild()
```

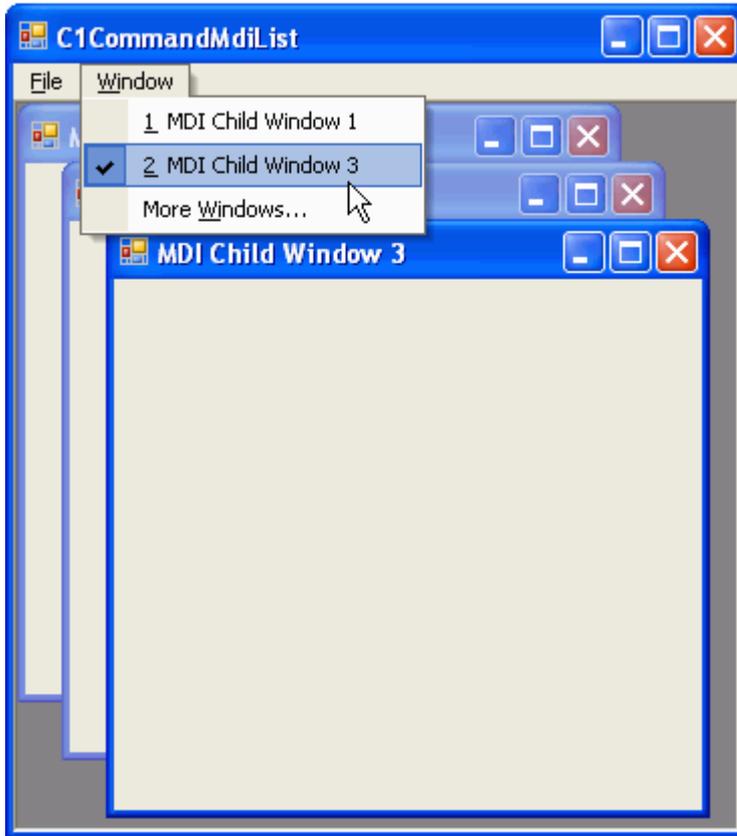
To write code in C#

```
C#

CreateNewMdiChild();
```

10. Press F5 to run the application.
11. From the **File** menu, click **New** to create a new MDI child form.

The [C1CommandMdiList](#) expands to the list of available MDI child windows.



 **Note:** The Window menu always displays a list of the MDI child forms open within the application with a check mark next to the MDI child that has the focus.

Deleting Menu Items

When a menu item is highlighted, and you press the DEL key or select **delete** from the context menu, either the command link or the command (simple or menu) linked by that link can be deleted, depending on which one is currently selected. (To check what is currently selected, look at the Properties window of the Visual Studio designer.)

To delete a menu item at design time

To delete the command link (this will leave the command, so that you can link to it from another command link):

1. Click once on the menu item you want to delete. Make sure that the Properties window shows the properties of the **command link** and not the command.
2. Right-click on the item to open the context menu and select **delete** or simply press the DEL key. To delete the command itself (this will remove the command but leave the command link so that you'll be able to link it to another command or menu):
3. Click on the menu item you want to delete. Then click on it again. This action will select the command rather than the command link. Make sure that the Properties window shows the properties of the **command** and not the command link.
4. Right-click on the item to open the context menu and select **delete**, or simply press the DEL key. The command will be deleted and the now empty command link will be left. You can re-link it to another command by right-clicking on it and selecting **edit** from the context menu.

To delete a menu item programmatically

To delete the menu, use the **Dispose** method. Use the following code to delete the menu or menu item:

To write code in Visual Basic

Visual Basic

```
'The variable name for the menu is called menu
'This deletes the menu
menu.Dispose()

'The variable name for the menu item is called menuitem1
'This deletes the menu item
menuitem1.Dispose()
```

To write code in C#

C#

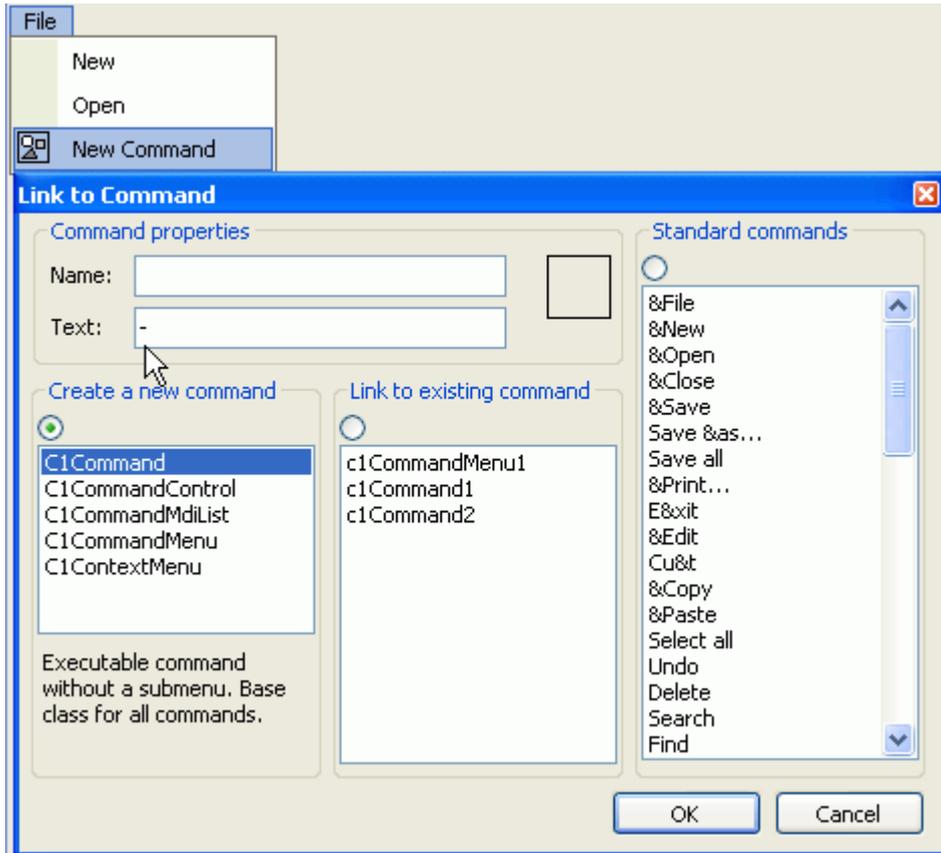
```
//The variable name for the menu is called menu
//This deletes the menu
menu.Dispose();

//The variable name for the menu item is called menuitem1
//This deletes the menu item
menuitem1.Dispose();
```

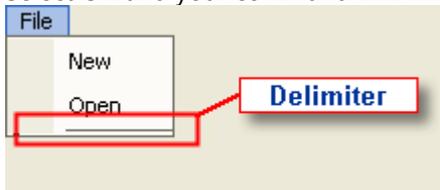
Displaying the Delimiter in Drop-Down Menu

To display a delimiter in the drop-down menu, complete the following steps:

1. Right-click a command from an existing drop-down menu and select **Append Item** from its context menu.
2. A new command will appear. In the New Command's **Link to Command to Command** designer, delete the name in the **Name** textbox. In the **Text** textbox change the name to a hyphen. The input value for the **Text** textbox and **Name** textbox fields should appear like the **Link to Command** designer shown below.



3. Select **OK** and your command link will appear as a delimiter like the one shown below.



Displaying ToolTips for Menus and Toolbars

This topic demonstrates how to customize the ToolTips by creating your own ToolTips for menus and toolbars. Both the menus and toolbars have the property called [ToolTipText](#) that provides users the ability to display ToolTips when the user mouses over a menu or a toolbar.

Displaying ToolTips for Menu Items

1. Right-click on the top level menu ([C1CommandMenu](#)) in the main menu and select Properties from its context menu.
2. Select **C1CommandMenu1** from the **Properties** drop-down list and expand the Command node.
3. Set the [ShowToolTips](#) property to **True** to enable the text to appear when the user mouses over the menus.
4. On the form, select a **C1Command** menu item that you want to display ToolTips. Set the [ShowTextAsToolTip](#) to **False** so that the command's text will not appear for the selected menu item in the [C1CommandMenu](#) when the user mouses over the menus.
5. Select the [ToolTipText](#) property and enter the information in the textbox that you want to appear when the user mouses over the menu item.
6. Save and run your application. Below is what the [ToolTipText](#) will look like at run time:

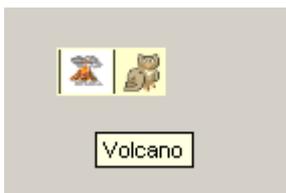


Displaying ToolTips for Buttons on the Toolbar

1. Select the toolbar button from the [C1ToolBar](#) control on your form.
2. The command link's properties for the [C1ToolBar](#) appear in the Properties Toolbox on the right-pane. Locate the **C1CommandLinks** properties and expand the **Command** properties.
3. Set the [ShowTextAsToolTip](#) to **False**.
4. Select the [ToolTipText](#) property and enter the information in the textbox that you want to appear when the user mouses over the toolbar button.
5. Save and run your application. Here is what the ToolTip text will look like for the toolbar button when you mouse over the volcano:



Note: If you leave the [ShowTextAsToolTip](#) property to its default value, **True**, then the toolbar button's text will be displayed when the user mouses over the toolbar button rather than the ToolTip's text. Also, delete the information that you entered for the [ToolTipText](#) property in the toolbox. The toolbar appears like the following image:



Displaying ToolTips Text for Buttons on the Outbar

You can follow the same procedure listed above to display ToolTips for buttons on the outbar.

Hiding Rarely Used Menu Items

To hide menu items that have not been used for a long time, use the [RecentLinksThreshold](#) property of the [C1CommandHolder](#) object to enable this feature.

To turn on the feature, set this property to a value between 0 and 100. A good starting point is 50.

Merging Menu Items

To merge the links from the current menu with the links in the child form's menu, use the [MergeCommandLinks](#)

method:

To write code in Visual Basic

Visual Basic

```
Me.c1CommandHolder1.MergeCommandLinks (Me.c1MainMenu1.CommandLinks,  
Me.c1MainMenu1.CommandLinks,  
_form2.c1MainMenu1.CommandLinks)
```

To write code in C#

C#

```
this.c1CommandHolder1.MergeCommandLinks (this.c1MainMenu1.CommandLinks,  
this.c1MainMenu1.CommandLinks,  
_form2.c1MainMenu1.CommandLinks);
```

Modifying the Appearance of the Menu

To modify [C1MainMenu](#) properties, use the property editor at design time.

To customize each menu item, use the associated properties as demonstrated in the following steps:

1. Right-click on the [C1MainMenu](#) control. Select **Menu Properties** in the context menu.

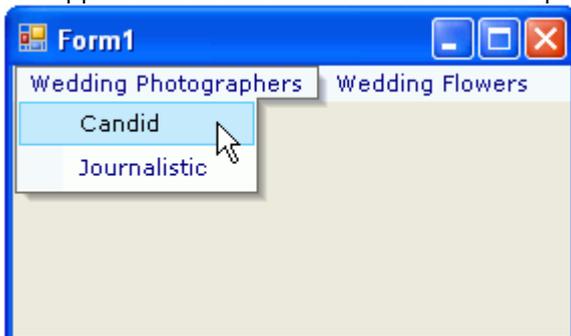
 **Note:** You can also view **C1MainMenu**'s properties in the Properties window.

2. Select **C1MainMenu1** from the Properties drop-down box and set the related **Font** properties:
 - o **Font** = **Verdana**
 - o **Font Size** = **11pt**
3. Select **C1MainMenu1** control from the Properties drop-down box and set the following properties:
 - o **BackColor** = Select the **Web** tab and select **Alice Blue**.
 - o **BackHiColor** = **#66CCFF**
 - o **ForeColor** = Select the **Web** tab and select **Navy**.
 - o **ForeHiColor** = **Black**
4. Build and run the Web application.

This topic illustrates the following:

- Unless otherwise modified, the submenu items inherit formatting from the parent Menu.
- You can modify the individual items as necessary.

The appearance of the [C1MainMenu](#) looks the picture:



 **Note:** This graphic illustrates the appearance of the C1MainMenu with content that was not added in this topic.

Setting the Width of the Image/Checkmarks Bar

To set the image/checkmark's bar in a `C1CommandMenu`, use the `ImageBarWidth` property.

Set the `ImageBarWidth` property of `C1CommandMenu` to a value greater than zero.

Showing a Dialog Form when a Message Filter is not Installed

`C1Command`'s Menus and Toolbars rely on installing a message filter (implementing the `IMessageFilter` interface) to implement main menu and some other functionality. In some cases (for example, when `C1Command` is used in a component's designer, and is run in Visual Studio's design time), installing a message filter does not work. In such cases, `C1Command` can still be used.

The following code fragment demonstrates how to show a dialog box when a message filter could not be installed. You can use this approach, for example, if you want to show a dialog box in your component's designer.

To write code in Visual Basic

Visual Basic

```
Imports Cl.Win.C1Command;
    C1CommandHolder.UninstallMessageFilter()
    'Create the C1CommandMsgHook
    Dim hook As New C1CommandMsgHook()
    hook.Install()
    Try
        result = dialog.ShowDialog()
    Finally
        Hook.Uninstall();
    End Try
```

To write code in C#

C#

```
using Cl.Win.C1Command;
    ...
    C1CommandHolder.UninstallMessageFilter();
    C1CommandMsgHook hook = new C1CommandMsgHook();
    hook.Install();
    try
    {
        result = dialog.ShowDialog();
    }
    finally
    {
        hook.Uninstall();
    }
```

Wrapping Items at the End of the Menu

To wrap the items at the end of the menu, use the [Wrap](#) property of [C1MainMenu](#).

If **True** (default), the menu is wrapped when the window is not wide enough. If **False**, the menu is not wrapped, and the extension menu is added at the end instead.

ToolBar Tasks

This section shows how to perform specific toolbar tasks.

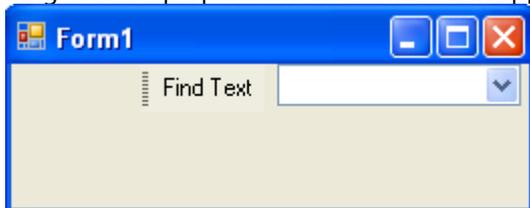
Adding an Arbitrary Control to the Toolbar

To add an arbitrary control to your toolbar, complete the following steps:

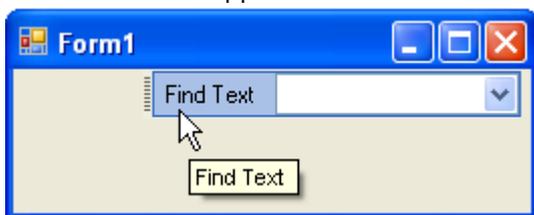
1. Add the [C1CommandDock](#) control to your form.
2. Place the [C1ToolBar](#) control onto the [C1CommandDock](#) using a drag-and-drop operation.
3. Right-click on the [C1ToolBar](#) located on your form. Select **Edit** from the context menu. The **Link to Command** designer appears. Set the following fields in the **Link to Command** designer:
 - o **Text** textbox to **Find text:**
 - o **Name** to **c1CommandControl1**
 - o Command Type to **C1CommandControl**

Press the **OK** button in the **Edit** dialog box.

4. Select the **C1CommandControl** from the Create a new command list box.
5. Select **c1CommandLink1** from the Properties drop-down list, then set the [ButtonLook](#) property to **Text**. The toolbar button appears as text.
6. Select the Windows form tab and place the **ComboBox** control to the right side of the **C1ToolBar** using a drag-and-drop operation. The combo box appears like the following:



7. Select **c1CommandControl1** from the Properties drop-down list, then select the [Control](#) property, and select **comboBox1** from its drop-down list.
8. Build and run the application. Your toolbar will appear like this at run time:



Adding an Image to the Toolbar Button

This topic assumes you have a toolbar button created. To add an image to your toolbar button in code, complete the following steps:

1. Locate the command in your source file that you want to add an image to.

2. Enter the following code to add an image to your toolbar button:

To write code in Visual Basic

```
Visual Basic
cNew.Image = System.Drawing.Image.FromFile("C:\bitmap\New.bmp")
```

To write code in C#

```
C#
cNew.Image = System.Drawing.Image.FromFile("C:\\bitmap\\New.bmp");
```

Adding Separators Between the Buttons

To add separators between the buttons, complete the following steps:

1. Right-click the second toolbar button and select **Properties** from its context menu.
2. Locate the appearance properties and set the **Delimiter** property to **True**.
3. Save and run your application. The button separator appears like the following:



Changing the Position of the Toolbar from Horizontal to Vertical

To change the position of the toolbars from horizontal to vertical, use the **Horizontal** property. To do this, complete the following steps:

1. Create a toolbar.
2. Use the following code to make the toolbar appear vertical:

To write code in Visual Basic

```
Visual Basic
toolbar.Horizontal = False
```

To write code in C#

```
C#
toolbar.Horizontal = False;
```

Creating a Toolbar

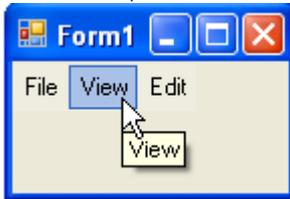
You can create a toolbar at design time or through code. Click on either of the following links to expand the steps for the designer or for the code.

To create a toolbar at design time

To create a [C1ToolBar](#) using its **Link to Command** designer, complete the following steps;

1. Place a **C1ToolBar** control on the form, then right-click the **C1ToolBar** control and select **Edit** from its context menu. The **Link to Command** designer for the selected toolbar item appears.
2. Enter **File** in the **Text** textbox, and then click **OK**.
3. Select **c1CommandLink1** from the Properties drop-down list. Set the CommandLink's **ButtonLook** property field for the File toolbar button to **Text**. The File toolbar button appears as text.
4. Right-click on the [C1ToolBar](#) control and select **Append Item** from the context menu. The new command link will be added after the current one. The **Link to Command** designer appears.
5. Enter **Edit** in the **Text** textbox and press the **OK** button.
6. Select **c1CommandLink2** from the **Properties** drop-down list box. Set the CommandLink's **ButtonLook** property field for the **Edit** toolbar button to **Text**.
7. Right-click on the Edit toolbar button and select **Insert Item** from its context menu. The new command link will be added before the current one. The **Link to Command** designer.
8. Enter **View** in the **Text** textbox then click **OK**.
9. Select **c1CommandLink3** from the Properties drop-down list. Set the CommandLink's **ButtonLook** property field for the View toolbar button to **Text**.
10. Build and run the Windows application.

At run time, the toolbar appears like the following image:



To create a toolbar programmatically

To programmatically create a [C1ToolBar](#) with text buttons, complete the following steps:

1. Add the `C1.Win.C1Command` namespace to your references in your project.
2. Declare the namespace in your source file, then add a **C1CommandHolder** to hold the toolbar.

To write code in Visual Basic

Visual Basic

```
Imports C1.Win.C1Command
Dim ch As C1CommandHolder = C1CommandHolder.CreateCommandHolder(Me)
```

To write code in C#

C#

```
using C1.Win.C1Command;
C1CommandHolder ch = C1CommandHolder.CreateCommandHolder(this);
```

3. Create a new [C1ToolBar](#), then add the [C1ToolBar](#) on your form.

To write code in Visual Basic

Visual Basic

```
Dim tb As New C1ToolBar()
Me.Controls.Add(tb)
```

To write code in C#

C#

```
C1ToolBar tb = new C1ToolBar();
this.Controls.Add(tb);
```

- Assign the [C1CommandHolder](#) to the [C1ToolBar](#), then create a new command for the toolbar. The name for the new command will be called **File**.

To write code in Visual Basic

Visual Basic

```
tb.CommandHolder = ch
Dim cFile As New C1Command()
cFile.Text = "File"
```

To write code in C#

C#

```
tb.CommandHolder = ch;
C1Command cFile = new C1Command();
cFile.Text = "File";
```

- Create a new commandlink for the new command in the toolbar, then add the new commandlink to the toolbar.

To write code in Visual Basic

Visual Basic

```
Dim cl As C1CommandLink
cl = New C1CommandLink(cFile)
tb.CommandLinks.Add(cl)
```

To write code in C#

C#

```
C1CommandLink cl = new C1CommandLink(cFile);
tb.CommandLinks.Add(cl);
```

- Make the commandlink appear as text, then create another command for the toolbar and call it **View**.

To write code in Visual Basic

Visual Basic

```
cl.ButtonLook = ButtonLookFlags.Text
Dim cView As New C1Command()
```

```
cView.Text = "View"
```

To write code in C#

```
C#  
cl.ButtonLook = ButtonLookFlags.Text;  
C1Command cView = new C1Command();  
cView.Text = "View";
```

7. Create a new commandlink for the new command, **View**, then add it to the toolbar.

To write code in Visual Basic

```
Visual Basic  
cl = New C1CommandLink(cView)  
tb.CommandLinks.Add(cl)
```

To write code in C#

```
C#  
cl = new C1CommandLink(cView);  
tb.CommandLinks.Add(cl);
```

8. Make the commandlink for **View** appear as text, then create another command and call it **Edit**.

To write code in Visual Basic

```
Visual Basic  
cl.ButtonLook = ButtonLookFlags.Text  
Dim mEdit As New C1Command()  
mEdit.Text = "Edit"
```

To write code in C#

```
C#  
cl.ButtonLook = ButtonLookFlags.Text;  
C1Command mEdit = new C1Command();  
mEdit.Text = "Edit";
```

9. Create a new commandlink for the new command, **Edit**, then add the commandlink to the toolbar.

To write code in Visual Basic

```
Visual Basic  
cl = New C1CommandLink(mEdit)  
tb.CommandLinks.Add(cl)
```

To write code in C#

```
C#  
cl = new C1CommandLink(mEdit);  
tb.CommandLinks.Add(cl);
```

10. Make the command link for the edit toolbar button to appear as text.

To write code in Visual Basic

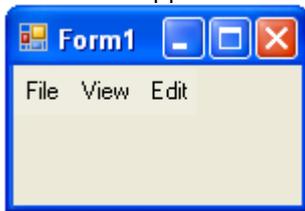
```
Visual Basic
cl.ButtonLook = ButtonLookFlags.Text
```

To write code in C#

```
C#
cl.ButtonLook = ButtonLookFlags.Text;
```

11. Save and run your application.

The toolbar appears like the following:



Docking a Toolbar Programmatically

To make a [C1ToolBar](#) dockable by using the [C1CommandDock](#) class, complete the following steps:

Note: This topic assumes you have a [C1ToolBar](#) created in code.

1. Create a new [C1CommandDock](#) and call it dock.

To write code in Visual Basic

```
Visual Basic
Dim dock As New C1CommandDock()
```

To write code in C#

```
C#
C1CommandDock dock = new C1CommandDock();
```

2. Add the [C1CommandDock](#) to your control so it will appear on the form.

To write code in Visual Basic

```
Visual Basic
Me.Controls.Add(dock)
```

To write code in C#

```
C#
this.Controls.Add(dock);
```

Increasing the Image Size in the Toolbar

To display larger icons/bitmaps you need to increase the size of the button in the Toolbar. This will in turn increase the size of the image. You can use the `MinButtonSize` property of the toolbar to achieve this behavior. The default size for the `MinButtonSize` property is 24 pixels.

To increase the size of the button in the toolbar at design time, complete the following task:

- Open the **C1ToolBar Tasks** menu and enter **50 pixels** in the `C1ToolBar.MinButtonSize` text box.

OR

- Set `C1ToolBar.MinButtonSize` property to **40 pixels** through its Properties window.

The following toolbar image had a minimum button size that has been increased to 40 pixels:



Modifying the Appearance of the Toolbar

To modify the appearance of the toolbar, complete the following steps:

1. Select the `C1ToolBar` drop-down arrow in the Properties window.
2. Expand the Font properties of the `C1ToolBar` component. Set the following font properties:
Font.Name = **Arial**
Font.Size = **9**
3. Set the following `C1ToolBar` properties:
`BackColor` = **#0066CC**
`BackHiColor` = **Pale Turquoise**
`ForeColor` = **White**
`ForeHiColor` = **#003399**
4. Expand the `C1ToolBar.Border` properties node. Set the following `C1ToolBar` properties:
`Border.Light Color` = **Pale Turquoise**
`Border.Style` = **Outset**
`Border.Width` = **3**

The toolbar appears like the following at run time:



Making the Toolbar Appear Like the Default Toolbar in Internet Explorer(IE)

This topic demonstrates how to make the `C1ToolBar` appear like the default toolbar in IE by using the `ButtonLookFlags` enumeration. The `ButtonLookFlags` enumeration contains the parameters: `Text`, `TextAndImage`, `Image`, and `Default`. These parameters give you the ability to modify your toolbar button to show text, text and an image, or just an image.

The following code demonstrates how you can make the `C1ToolBar` appear like the default toolbar in IE. The following example uses all of the parameters (`TextAndImage`, `Image`, `Default`, and `Text`) contained in the `ButtonLookFlags` enumeration to make the `C1ToolBar` appear similar to the IE toolbar:

1. Create the first toolbar button and make it appear as an image. The following code shows you how to use the **ButtonLookFlags** enumeration to make the toolbar button appear as an image:

To write code in Visual Basic

Visual Basic

```
Dim ch As C1CommandHolder = C1CommandHolder.CreateCommandHolder(Me)
Dim tb As New C1ToolBar()
Me.Controls.Add(tb)
tb.CommandHolder = ch
Dim cNew As New C1Command()
Dim cl As C1CommandLink
cNew.Text = "New"
cNew.Image = System.Drawing.Image.FromFile("C:\Images\New.bmp")
cl = New C1CommandLink(cNew)
tb.CommandLinks.Add(cl)
'Use the ButtonLookFlags enumeration to make the toolbar button appear as an
image
cl.ButtonLook = ButtonLookFlags.Image
```

To write code in C#

C#

```
C1CommandHolder ch = C1CommandHolder.CreateCommandHolder(this);
C1ToolBar tb = new C1ToolBar();
this.Controls.Add(tb);
tb.CommandHolder = ch;
C1Command cNew = new C1Command();
cNew.Text = " New ";
cl = new C1CommandLink(cNew);
tb.Commandlinks.Add(cl);
//Use the ButtonLookFlags enumeration to make the toolbar button appear as an
image
cl.ButtonLook = ButtonLookFlags.Image;
```

2. Create a second toolbar button and make it appear as text and image. The following code shows you how to use the **ButtonLookFlags** enumeration to make the toolbar button appear as text and an image:

To write code in Visual Basic

Visual Basic

```
Dim cOpen As New C1Command()
cOpen.Text = "Open"
cl = New C1CommandLink(cOpen)
cOpen.Image = System.Drawing.Image.FromFile("C:\Images\FileOpen.bmp")
tb.CommandLinks.Add(cl)
'Use the ButtonLookFlags enumeration to make the toolbar button appear as text
and an image
cl.ButtonLook = ButtonLookFlags.TextAndImage
```

To write code in C#

C#

```
C1Command cOpen = new C1Command();
cOpen.Text = "Open";
cl = new C1CommandLink(cOpen);
cOpen.Image = System.Drawing.Image.FromFile("C:\Images\FileOpen.bmp");
tb.CommandLinks.Add(cl);
//Use the ButtonLookFlags enumeration to make the toolbar button appear as text
and an image
cl.ButtonLook = ButtonLookFlags.TextAndImage;
```

3. Create a third toolbar button and make it appear as the default appearance. The following code shows you how to use the **ButtonLookFlags** enumeration to make the toolbar button appear as the default appearance:

To write code in Visual Basic

Visual Basic

```
Dim cSave As New C1Command()
cSave.Text = "Save"
cl = New C1CommandLink(cSave)
tb.CommandLinks.Add(cl)
cSave.Image = System.Drawing.Image.FromFile("C:\Images\FileSave.bmp")
'Use the ButtonLookFlags enumeration to make the toolbar button appear as
default
cl.ButtonLook = ButtonLookFlags.Default
```

To write code in C#

C#

```
C1Command cSave = new C1Command();
cSave.Text = "Save";
cl = new C1CommandLink(cSave);
tb.CommandLinks.Add(cl);
cSave.Image = System.Drawing.Image.FromFile("C:\Images\FileSave.bmp")
//Use the ButtonLookFlags enumeration to make the toolbar button appear as
default
cl.ButtonLook = ButtonLookFlags.Default;
```

4. Create a fourth toolbar button and make it appear as text. The following code shows you how to use the **ButtonLookFlags** enumeration to make the toolbar button appear as text:

To write code in Visual Basic

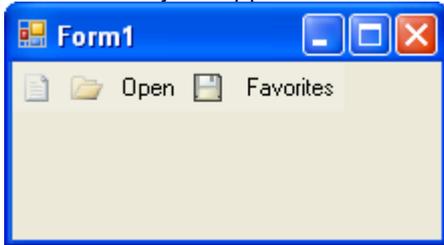
Visual Basic

```
Dim cFavorites As New C1Command()
cFavorites.Text = "Favorites"
cl = New C1CommandLink(cFavorites)
tb.CommandLinks.Add(cl)
'Use the ButtonLookFlags enumeration to make the toolbar button appear as text
cl.ButtonLook = ButtonLookFlags.Text
```

To write code in C#

```
C#
C1Command cFavorites = new C1Command();
cFavorites.Text = "Save";
c1 = new C1CommandLink(cFavorites);
tb.CommandLinks.Add(c1);
//Use the ButtonLookFlags enumeration to make the toolbar button appear as text
c1.ButtonLook = ButtonLookFlags.Text;
```

5. Save and run your application. Your **C1ToolBar** will appear similar to the following toolbar:



Note: Your toolbar images will appear different from the toolbar images in this image. The first, second, third, and fourth toolbar buttons are shown as an image, text and image, default, and text, respectively.

Making the Image in the Toolbar Button Appear More Vibrant

The **SmoothImages** property is enabled by default to draw smooth images for unselected items. You can set this property to **False** to make the unselected images appear more vibrant.

The following table illustrates the difference between the **SmoothImages** property enabled (**True**) and disabled (**False**):

Property	Toolbar Images
False	
True	

Specifying a Docking/Floating Position

This topic demonstrates how to specify a position for your **C1CommandDock** in code.

To dock your toolbar to the top, enter the following code:

To write code in Visual Basic

```
Visual Basic
dock.Dock = DockStyle.Top
```

To write code in C#

```
C#
dock.Dock = DockStyle.Top;
```

 **Note:** If you want to dock your toolbar to the bottom, left, or right, change the *Top* parameter to its desired position: *Bottom*, *Left* or *Right*.

Turning on the Customization Feature

Setting the `CustomizeButton` property to **True** can turn on the customization feature, which enables the user to customize the toolbars at run time. For more information about this feature, see [Run Time Customization for ToolBars](#).

Wrapping Text in a ToolBar

To wrap the text in a `C1ToolBar`, use the `WrapText` property. To do this, complete the following steps:

1. Set the `ButtonWidth` property to a value greater than zero
2. Set the `WrapText` property to **True**.

Context Menu Tasks

This section shows how to perform specific context menu tasks.

Adding a ContextMenu to a Control

You can add a `C1ContextMenu` 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

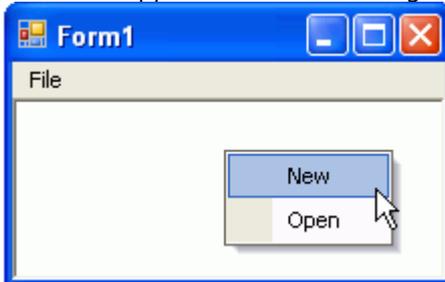
To create a context menu and associate it with a menu item, complete the following tasks:

 **Note:** In this example, a `C1ContextMenu` is associated with a menu item.

1. Place a `C1MainMenu` 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.
4. 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**
5. Select **C1Command** from the **Create a new command** listbox.
6. Click **OK**.
7. 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**
8. Select **C1Command** from the **Create a new command** listbox.
9. Click **OK**.
10. Select the **Windows Form** tab from the Toolbox and place the **RichTextBox** control onto your form using a

drag-and-drop operation.

11. Select the **richTextBox1** control from the **Properties** drop-down list box then set its **Dock** property to **Fill**.
12. Select the **MenuFile** from the Properties drop-down list, then select the **Category** property and enter **File** in its box.
13. Select **richTextBox1** from the Properties drop-down list, then select the **C1ContextMenu on C1CommandHolder** property, and then select **MenuFile**.
14. 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** 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.

To write code in Visual Basic

Visual Basic

```
Dim ch As C1.Win.C1Command.C1CommandHolder
ch = C1CommandHolder.CreateCommandHolder(Me)
```

To write code in C#

C#

```
C1CommandHolder ch = C1CommandHolder.CreateCommandHolder(this);
```

4. Create and set up a copy command, then use the **Click** 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

```
'Create and set up the Copy command
Dim cmdCopy As C1Command = ch.CreateCommand()
cmdCopy.Text = "&Copy"
AddHandler cmdCopy.Click, AddressOf clickCopy
AddHandler cmdCopy.CommandStateQuery, AddressOf queryCopy
```

To write code in C#

C#

```
// 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);
```

5. 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](#) and assign it to the context menu. Use the **C1CommandHolder.CreateCommand** and **C1CommandHolder.SetC1ContextMenu** methods from the [C1CommandHolder](#) class. Add the following code within the **Form_Load** event handler:

To write code in Visual Basic

Visual Basic

```
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#

```
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);
```

6. 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

```
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 command

Private Sub queryCopy(ByVal sender As Object, ByVal e As
C1.Win.C1Command.CommandStateQueryEventArgs)
    e.Enabled = Me.textBox1.SelectionLength > 0
End Sub
```

To write code in C#

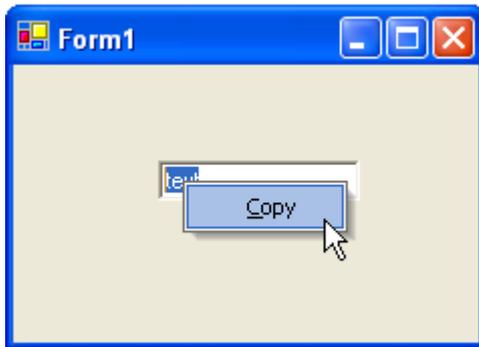
```
C#
private void clickCopy(object sender, C1.Win.C1Command.ClickEventArgs e)
{
    this.textBox1.Copy();
}

//provides the current state of the copy command

private void queryCopy(object sender,
C1.Win.C1Command.CommandStateQueryEventArgs e)
{
    e.Enabled = this.textBox1.SelectionLength > 0;
}

```

7. 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 ContextMenu Control Attached to the TextBox

Use the [GetC1ContextMenu](#) method of [C1CommandHolder](#) class to determine which [C1ContextMenu](#) is attached to a control. The [GetC1ContextMenu](#) 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
'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#
//retrieves the contextmenu attached to the C1TextBox control
C1.Win.C1Command.C1ContextMenu contextMenu;

```

```
contextMenu = C1CommandHolder1.GetC1ContextMenu(C1TextBox1);
MessageBox.Show(ContextMenu.Name);
```

Linking the Context Menu to a NotifyIcon Control

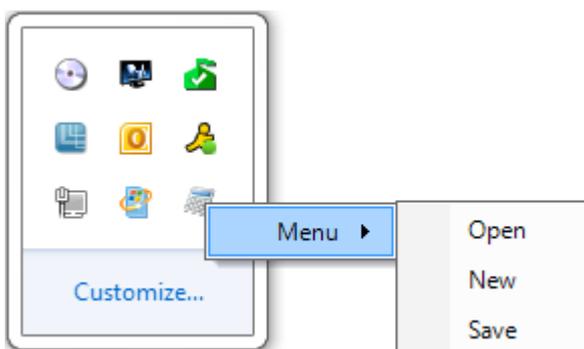
[C1ContextMenu](#) supports linking to the standard **NotifyIcon** Control.

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

1. Double-click on the [C1ContextMenu](#) 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](#) 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.
4. 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**.
5. Select [C1CommandMenu](#) from the Create a new command listbox.
6. Select **OK** in the **Link to Command** designer and the new menu will appear on the form.
7. Right-click the **New Command** item in the Menu and select **Edit** from its context menu to open the **Link to Command** designer.
8. Change the following properties in the **Link to Command** designer:
 - Change the **Name** textbox text to **OpenFile**.
 - Change the **Text** textbox text to **Open**.
9. Select **View | Properties** from the Visual Studio toolbar. Select **C1ContextMenu1** from the drop-down list at the top of the Properties window.
10. Set the [C1ContextMenu](#) property to **C1ContextMenu1**.
11. Double-click the **NotifyIcon** component in the Windows Forms Toolbox to add the component to the component tray.
12. Right-click on the component and select **Choose an icon** from its context menu. Choose an icon to represent the component at run time.
13. Select **View | Properties** from the Visual Studio toolbar. Select **notifyicon1** from the drop-down list at the top of the Properties window.
14. Set the [C1ContextMenu](#) property to **C1ContextMenu1** to link the two components.
15. 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 ContextMenu to a DockingTab

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

1. Locate the [C1ContextMenu](#) control in your Toolbox and double-click the control to add it to your Component Tray. A [C1CommandHolder](#) will also be added to the Component Tray.
2. Use the [C1ContextMenu](#) smart tag to open the [C1ContextMenu Tasks Menu](#). Select **Add Item** from the **Tasks Menu** to add an item to the [C1ContextMenu](#). Add two more items to the menu.
3. Place a [C1DockingTab](#) control on the form and open the [C1DockingTab](#) Context Menu. Select **Add Page** from the context menu. Add several pages to the [C1DockingTab](#).
Select the [C1DockingTab](#) control on your form to view the properties. Click the Events button and locate the **MouseDown** event. Insert **c1DockingTab1_MouseClick** to create the **MouseDown** event.
4. 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

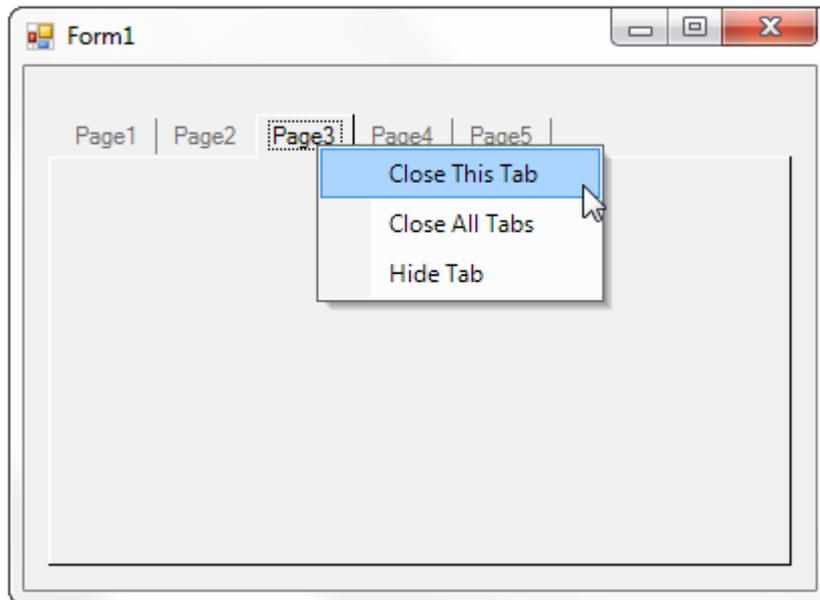
```
Private Sub c1DockingTab1_MouseClick(ByVal sender As Object, ByVal e As
MouseEventArgs)
    If (e.Button = System.Windows.Forms.MouseButtons.Left) Then
        If (e.X > Me.c1DockingTabPage1.Location.X) Then
            c1ContextMenu1.ShowContextMenu(Me.c1DockingTab1, New Point(e.X,
e.Y))
        Else
            Me.c1DockingTab1.ContextMenu = Nothing
        End If
    End If
End Sub
```

To write code in C#

C#

```
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));
        }
        else
            this.c1DockingTab1.ContextMenu = null;
    }
}
```

5. Press F5 to run your application. Note that when you click on a tab, the [C1ContextMenu](#) opens:



DockingTab Tasks

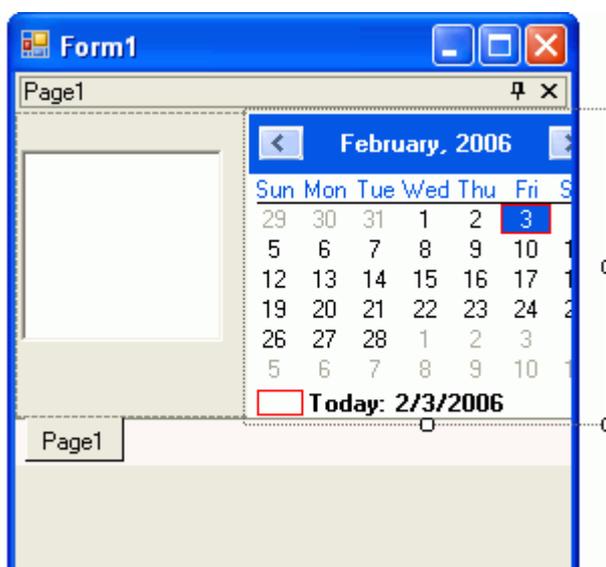
This section shows how to perform specific docking tab tasks.

Adding a Scrollbar to a DockingTab

To add a scrollbar to a `C1DockingTab`, complete the following steps:

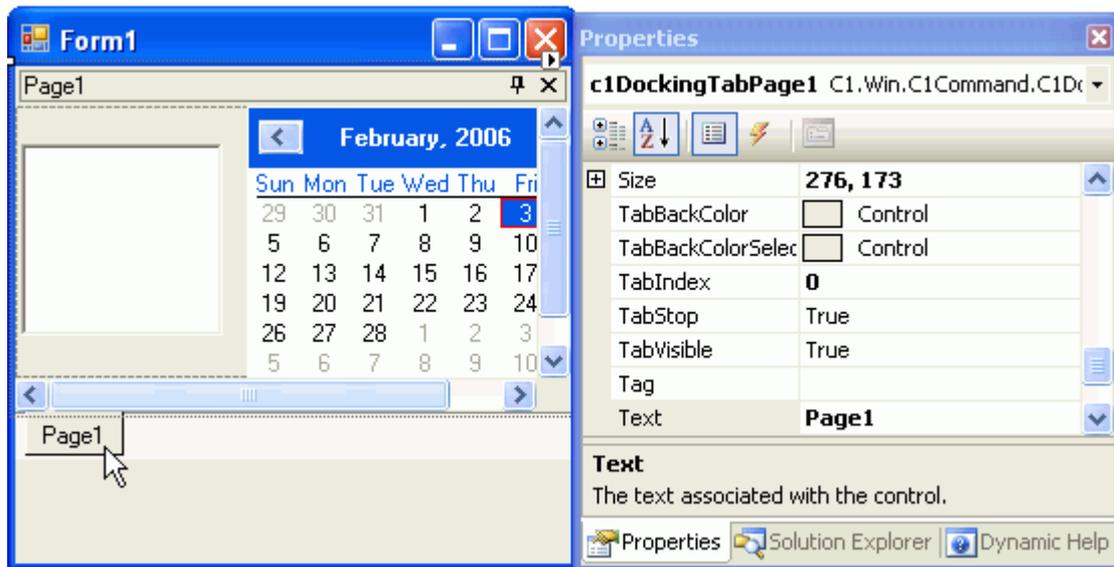
 **Note:** The `C1DockingTab` does not have its own scrollbar.

1. Place a `C1CommandDock` control on the form, and then set its **Dock** property to **Fill**.
2. Add a `C1DockingTab` inside the `C1CommandDock`.
3. Add a **Panel** control inside the `C1DockingTabPage`, and then set the Panel's **Dock** property to **Fill**.
4. Place a few Windows Form controls inside Panel1 of the `C1DockingTabPage`.



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

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



Closing a DockingTabPage

You can close the [C1DockingTabPage](#) in code by using the [TabVisible](#) property of the [C1DockingTabPage](#). The following code can be used for closing the first page:

To write code in Visual Basic

```
Visual Basic
Me.C1DockingTab1.TabPages(0).TabVisible = False
```

To write code in C#

```
C#
this.C1DockingTab1.TabPages(0).TabVisible = False;
```

Determining if the DockingTab is Floating

To find out if the docking tab is currently Floating, use the Floating (run time) only property of the [C1DockingTab](#). This syntax of the property is as following:

To write code in Visual Basic

```
Visual Basic
Public Floating As Boolean
```

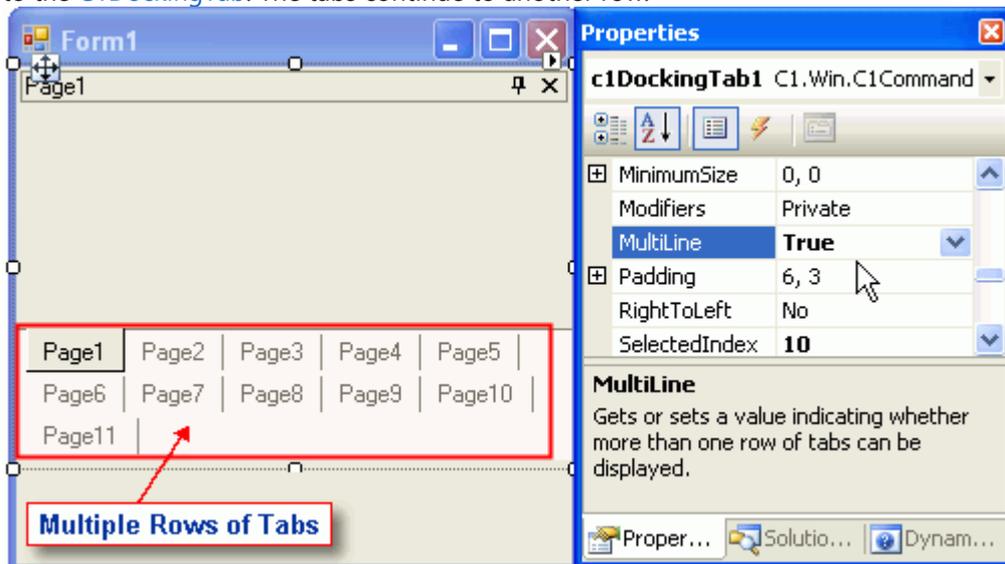
To write code in C#

```
C#
Public bool Floating {get;}
```

Displaying Multiple Tab Rows

To display more than one row of tab in a [C1DockingTab](#), set the [MultiLine](#) property of the [C1DockingTab](#) class to **True**. To do this, complete the following tasks:

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



Displaying the Same Set of Controls on each DockingTabPage

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

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

To write code in Visual Basic

Visual Basic

```
Me.dockingTab.SelectedTab.Controls.Add (myPanel)
```

To write code in C#

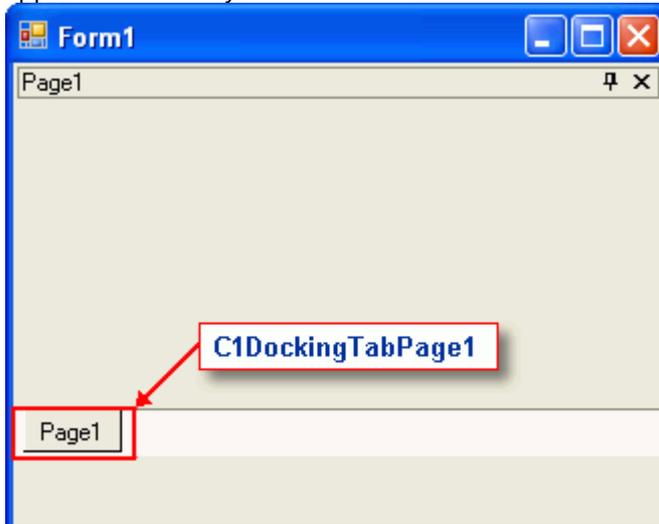
C#

```
this.dockingTab.SelectedTab.Controls.Add (myPanel);
```

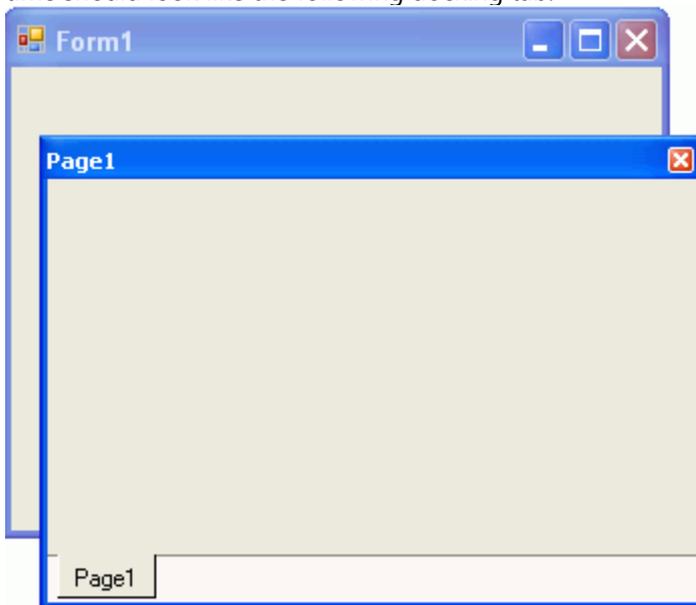
Enabling DockingTab Docking and Floating

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

1. Place the [C1CommandDock](#) control onto your form using a drag-and-drop operation.
2. The [C1CommandDock](#) 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](#) control inside the [C1CommandDock](#) 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](#) property [FloatHide](#) to control how the [C1DockingTabPage](#) 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](#): **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` to **false**.

In Code

To write code in Visual Basic

Visual Basic

```
c1DockingTab1.TabsShowFocusCues = false
```

To write code in C#

C#

```
c1DockingTab1.TabsShowFocusCues = false;
```

Loading and Saving the Layout of the DockingTab

You can save and load the layout of the `C1DockingTab` by using the `SaveLayout` and `RestoreLayout` methods. The syntax for the methods is as follows:

To write code in Visual Basic

Visual Basic

```
'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 file  
Sub RestoreLayout(form As Form, filename As String)
```

To write code in C#

C#

```
//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  
file  
Static 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

```
//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 file  
C1DockingTab.RestoreLayout (myForm, "myLayoutFile.xml")
```

To write code in C#

C#

```
//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  
file  
C1DockingTab.RestoreLayout (myForm, "myLayoutFile.xml");
```

Moving Tab Pages at Run Time

To move tabs in different positions, use the [CanMoveTabs](#) property.

In the properties toolbox set the [CanMoveTabs](#) to **True**.

Pinning the DockingTab

To pin the [C1DockingTab](#) programmatically, set the [AutoHiding](#) property of the [C1DockingTab](#) to **False**. Use the following code:

To write code in Visual Basic

Visual Basic

```
Me.C1DockingTab1.AutoHiding = False
```

To write code in C#

C#

```
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](#) 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 [SelectedIndexChanged](#) 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 [SelectedIndexChanged](#) event:

To write code in Visual Basic

Visual Basic

```
Private Sub c1DockingTab1_SelectedIndexChanging(sender As Object, e As
C1.Win.C1Command.SelectedIndexChangingEventArgs)
    If e.NewIndex = 1 And e.CanCancel Then
        e.Cancel = True
    End If
End Sub
```

To write code in C#

C#

```
private void c1DockingTab1_SelectedIndexChanging(object sender,
C1.Win.C1Command.SelectedIndexChangingEventArgs e)
{
    if(e.NewIndex == 1 && e.CanCancel)
        e.Cancel = true;
}
```

NavBar Tasks

This section shows how to perform specific navigation bar tasks.

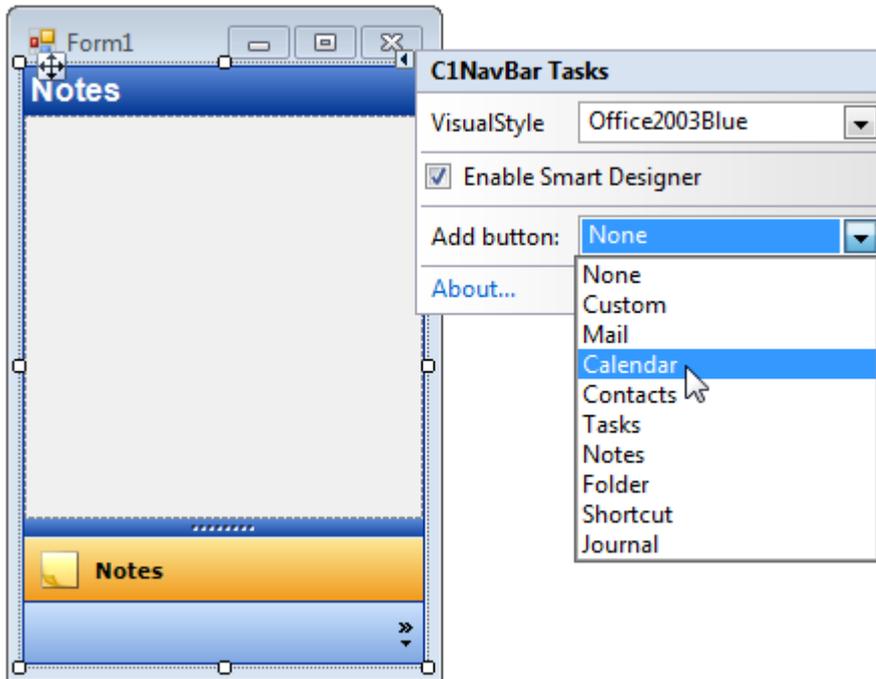
Adding a Panel

In this topic, you will add a panel to the [C1NavBar](#) control using the smart tag and the floating toolbar.

Using the Smart Tag

Complete the following steps:

1. Navigate to the Toolbox and double-click the [C1NavBar](#) icon. The [C1NavBar](#) control is added to the form. Observe that one panel, named **Notes**, appears on the control by default.
2. Click [C1NavBar](#)'s smart tag (📌) to open the **C1NavBar Tasks** menu.
3. On the **C1NavBar Tasks** menu, click the **Add button** drop-down button and select one of the panel types from the list. For this example, select **Calendar**.



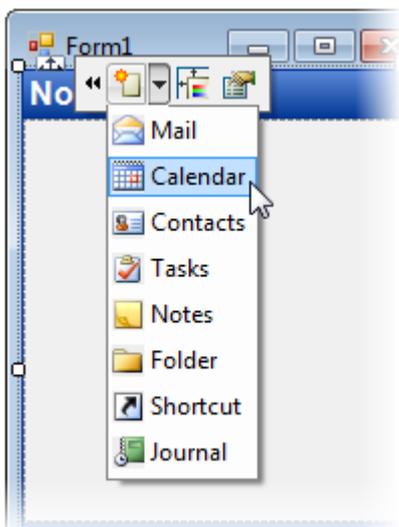
Using the Floating Toolbar

Complete the following steps:

1. Navigate to the Toolbox and double-click the **C1NavBar** icon. The **C1NavBar** control is added to the form. Observe that one page, named **Page 1**, appears on the control by default.
2. Hover your cursor over the control to activate the floating toolbar. The floating toolbar will appear on the page as follows:

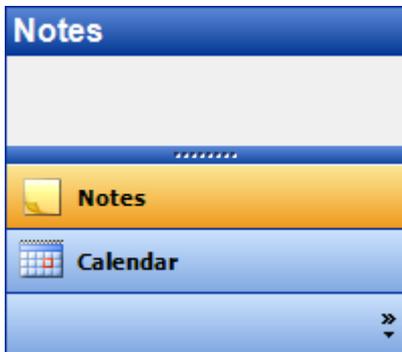


3. From the **Adding button and corresponding panel** drop-down list, select a panel type. For this example, select **Calendar**.



This topic illustrates the following:

The result of this topic resembles the following:



Creating a Panel Header

In this topic, you will learn how to add a header to a panel using the smart tag and the floating toolbar.

Using the Smart Tag

Complete the following steps:

1. Select the panel you would like to add the header to and click its smart tag (📌). The **C1NavBarPanel Tasks** menu opens.
2. From the **C1NavBarPanel Tasks** menu, select Add section header.
3. Select the header, **c1NavBarSectionHeader1**, and in the **Properties** window, set its **Text** property to a string. For this example, set it to "Hello World".

Using the Floating Toolbar

Complete the following steps:

1. Hover over the panel you wish to add the panel to until its floating toolbar appears. The floating toolbar will appear as follows:



The **C1NavBar** control is added to the form. Observe that one page, named **Page 1**, appears on the control by default.

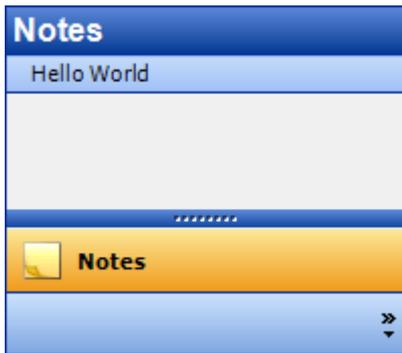
2. From the floating toolbar, select the **Add section header** button.



3. Select the header, **c1NavBarSectionHeader1** and, in the **Properties** window, set its **Text** property to a string. For this example, set it to "Hello World".

This topic illustrates the following:

The result of this topic resembles the following:



Using Vertical Text for Collapsed Panels

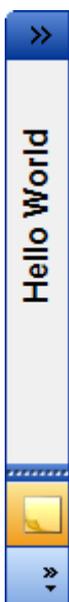
You can force the [C1NavBar](#) control to display vertical text when a panel is collapsed. The vertical text that appears is determined by the **NavBarCollapsedBarText** UIString. To utilize this feature, you will also need to set the [ShowVerticalTextOnCollapse](#) property to **True**.

Complete the following steps:

1. Right-click the [C1NavBar](#) control to open its context menu and then select Properties. The Properties window opens.
2. In the Properties window, complete the following:
 - o Set the [AllowCollapse](#) property to **True**. This will allow you to collapse the control at run time so you can see the vertical text on a collapsed panel.
 - o Set the [ShowVerticalTextOnCollapse](#) property to **True**.
 - o Set the **UIStrings.NavBarCollapsedBarText** enumeration to a string by expanding the [UIStrings](#) node and entering text into the **NavBarCollapsedBarText** text box.
3. Press F5 to run the project.

This topic illustrates the following:

While the project is running, click the expand/collapse button to collapse the control. The text on the collapsed panel will resemble the following:



OutBar Tasks

This section shows how to perform specific out bar tasks.

Customizing the Titles of OutPages

In this topic, you will learn how to customize the title area of a [C1OutBar](#) control's pages. You will create a [C1OutBar](#) with three [C1OutPages](#), set a few properties, and then add code to the project that will paint custom colors to each title.

Complete the following steps:

1. Add a [C1OutBar](#) control to your form.
2. Add three [C1OutPage](#) components to the [C1OutBar](#) control. (See [Adding a C1OutPage to the C1OutBar](#).)
3. Set the following properties:
 - Set **c1OutBar1**'s [VisualStyle](#) property to **Classic**. If you want, you can also choose **Custom**; the rest of the visual styles will not work for custom title drawing.
 - Set **c1OutPage1**'s [OwnerDraw](#) property to **True**.
 - Set **c1OutPage2**'s [OwnerDraw](#) property to **True**.
 - Set **c1OutPage3**'s [OwnerDraw](#) property to **True**.
4. In the **Properties** window, select **c1OutBar1** from the drop-down list, click the **Events** button, and then double-click the [DrawPage](#) event to add the [DrawPage](#) event handler to the project.
5. Import the following namespace to the project:

To write code in Visual Basic

```
Visual Basic
Imports Cl.Win.C1Command
```

To write code in C#

```
C#
using Cl.Win.C1Command;
```

6. Add the following code to the [DrawPage](#) event handler:

To write code in Visual Basic

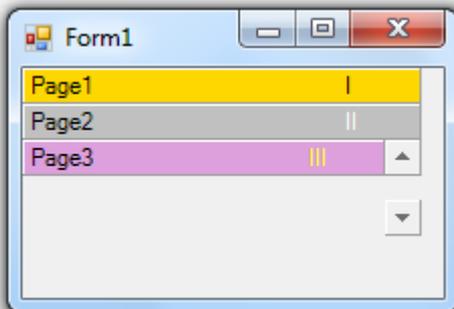
```
Visual Basic
'e.page parameter determines the page
If e.Page Is c1OutPage1 Then
    e.Graphics.FillRectangle(Brushes.Gold, e.Bounds)
    e.Graphics.DrawString("I", c1OutBar1.Font, Brushes.Black, New
PointF(e.Bounds.Right - 40, e.Bounds.Top + 2))
ElseIf e.Page Is c1OutPage2 Then
    e.Graphics.FillRectangle(Brushes.Silver, e.Bounds)
    e.Graphics.DrawString("II", c1OutBar1.Font, Brushes.White, New
PointF(e.Bounds.Right - 40, e.Bounds.Top + 2))
ElseIf e.Page Is c1OutPage3 Then
    e.Graphics.FillRectangle(Brushes.Plum, e.Bounds)
    e.Graphics.DrawString("III", c1OutBar1.Font, Brushes.Yellow, New
PointF(e.Bounds.Right - 40, e.Bounds.Top + 2))
```

End If

To write code in C#

```
C#
\\e.page determines the page
If (e.Page == c1OutPage1)
{
    e.Graphics.FillRectangle(Brushes.Gold, e.Bounds);
    e.Graphics.DrawString("I", c1OutBar1.Font, Brushes.Black, new
PointF(e.Bounds.Right - 40, e.Bounds.Top + 2));
}
else if (e.Page == c1OutPage2)
{
    e.Graphics.FillRectangle(Brushes.Silver, e.Bounds);
    e.Graphics.DrawString("II", c1OutBar1.Font, Brushes.White, new
PointF(e.Bounds.Right -40, e.Bounds.Top + 2));
}
else if (e.Page == c1OutPage3)
{
    e.Graphics.FillRectangle(Brushes.Plum, e.Bounds);
    e.Graphics.DrawString("III", c1OutBar1.Font, Brushes.Yellow, new
PointF(e.Bounds.Right - 40, e.Bounds.Top + 2));
}
}
```

7. Press F5 to run the project and observe that the titles are customized. The final product appears as follows:

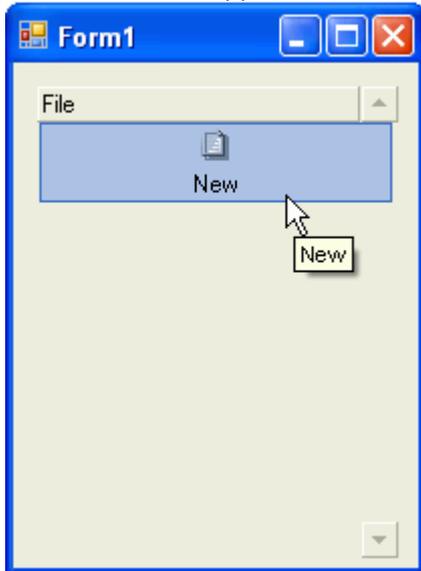


Creating and Configuring the OutBar Control

To create a [C1OutBar](#) at design time, complete the following steps:

1. Place the [C1OutBar](#) control to your form using a drag-and-drop operation. The newly created [C1OutBar](#) will initially contain a single page with a [C1ToolBar](#) control on it ([C1OutBar](#) pages can contain either [C1ToolBar](#) controls which are specially treated in that case, or arbitrary other controls, such as pages of a tab control). Also, a [C1CommandHolder](#) will be automatically created and placed on the component's tray.
2. Select **C1OutPage1** from the Properties window drop-down list. Change its **Text** property from Page1 to **File**.
3. Right-click on the (only) item in the [C1ToolBar](#), and select **Append Item** from the context menu. The **Link to Command** dialog box appears.
4. In the **Link to Command** designer set the following command properties:
 - o **Text** to New
 - o **Name** to cmdFileNew

5. Select **C1Command** from the **Create a new command** listbox.
6. Select **OK**.
7. Select the **C1CommandLink1** from the Properties window drop-down list and set the **ButtonLook** property to **TextAndImage**.
8. Select the **cmdFileNew** and locate its image property. Click on the ellipsis button  and locate the desired image.
9. Select the image and click **OK** in the **Select Resource** dialog box. The new image appears above **C1CommandLink1**'s text.
10. Build and run the application. It will look similar to the following image:



Adding a OutPage to the OutBar

To programmatically add a new **C1OutPage** to the **C1OutBar**, complete the following steps:

1. Add the **C1OutBar** control to the form.
2. Import the **C1.Win.C1Command** namespace. The code below imports the **C1.Win.C1Command** namespace.

To write code in Visual Basic

```
Visual Basic
Imports C1.Win.C1Command
```

To write code in C#

```
C#
using C1.Win.C1Command;
```

3. Add a new **C1OutPage** to the **C1OutBar**. The last line of code uses the **Add** method to add the new outpage1 to the **C1OutBar1**. Enter the following code within the **Form_Load** event:

To write code in Visual Basic

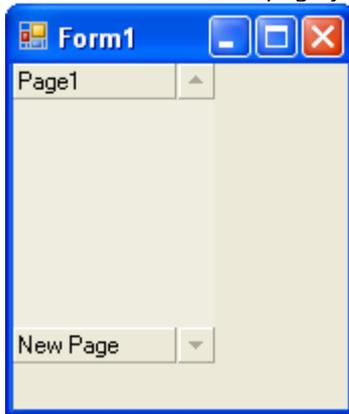
```
Visual Basic
Dim outpage1 As New C1.Win.C1Command.C1OutPage()
Outpage1.Text = "New Page"
```

```
Me.c1OutBar1.Pages.Add(outpage1)
```

To write code in C#

```
C#  
C1.Win.C1Command.C1OutPage outpage1 = new C1.Win.C1Command.C1OutPage();  
outpage1.Text = "New Page";  
this.c1OutBar1.Pages.Add(outpage1);
```

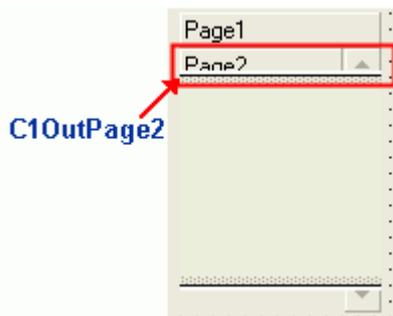
4. Build and run the new page you created for **C1OutBar**. The new out page appears like the following image.



Adding Multiple OutPages to OutBar

To add a new **C1OutPage** to a new **C1OutBar** at design time, complete the following steps:

1. Place the **C1OutBar** component from the toolbox to your form using a drag-and-drop operation. The **C1OutBar** consists of a **C1OutPage** and a **C1ToolBar**.
2. Right-click **C1OutPage1** and select **Add Page with Toolbar** from its context menu.
3. The new **C1OutPage2** appears similar to the picture below:



Note that you can create several **C1OutPages** by right-clicking on the **C1OutPage** and selecting **Add Page with Toolbar** from its context menu.

Modifying the Appearance of the OutBar

To modify the appearance of the **C1OutBar** at design time, complete the following steps:

1. Place the **C1OutBar** component from the toolbox to your form using a drag-and-drop operation. The default **C1OutBar** control contains a **C1OutPage** with a **C1ToolBar** control inside it.
2. In the Properties window for the **C1OutBar** control, select **C1OutPage1** from its Properties window drop-down list.

3. Select the **Text** property and change it from "Page1" to "Going Places".
4. Right-click the **C1ToolBar** control on the form and select **Append Item** from its context menu. A new command and the **Link to Command** designer appears.
5. In the **Link to Command** designer set the following properties:
 - o **Text** to "Air Tickets"
 - o **Name** to "cmdGPAirTicket"
6. Select **C1Command** from the **Create a new command** listbox and click OK to close the dialog box.

Adding a text and image to the command link

1. Select the **C1CommandLink1** from the Properties window drop-down list and set its **ButtonLook** property to **TextAndImage**.
2. In the **C1CommandLink1**'s Properties window, expand the Command properties node and click on the **ellipses** button in the image properties field. The **Open** dialog box appears.
3. Browse for an image to add to the Air tickets command link. The new image appears in the Air tickets command link.

Adding a new command link

1. Right-click the Air tickets commandlink on the toolbar and select **Append Item** from its context menu. A new commandlink appears in the toolbar.
2. In the **Link to Command** designer set the following properties:
 - o Text to **City Guides**
 - o Name to **cmdGPCityGuides**
 - o Command Type in the create a new command listbox to **C1Command**
3. Select **OK** to apply the changes and close the **Link to Command** designer.

Adding a text and image to the command link

1. Select the **C1CommandLink2, City Guides**, on the toolbar and set its **ButtonLook** property to **TextAndImage**.
2. Expand the Command properties node in **CommandLink2**'s Properties window and click on the **ellipses** button in the image properties field. Select the desired image and click on the **Open** button in the **Open** dialog box. The new image will appear above **C1CommandLink2**'s text.

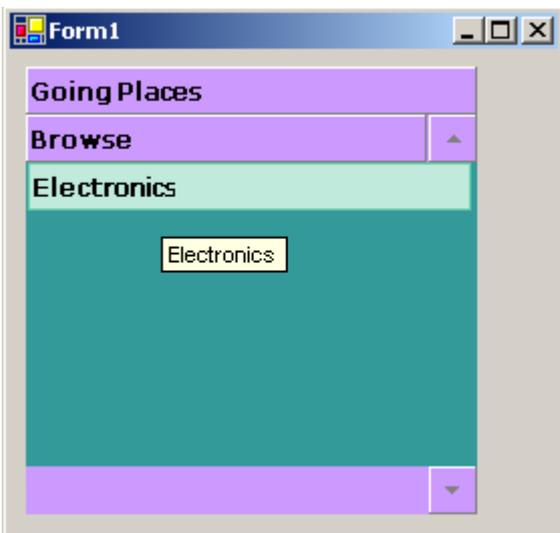
Modifying the Appearance properties for C1OutPage1 and C1ToolBar1

1. Select **C1OutPage1** from the Properties drop-down list and modify its properties to the following:
 - o Expand the **Font** property and set **Bold** to **True**.
 - o Select the **BorderStyle** property and select **FixedSingle** from its drop-down list.
2. Select the **C1ToolBar1** on the form and set its properties to the following:
 - o **BackColor** to **White**
 - o **BackHiColor** to **SkyBlue**
3. Select **C1OutBar1** from the Properties drop-down list and set its properties to the following:
 - o **BackColor** to **#CC99FF**
 - o **Font.Size** to **9**
 - o **Bold** to **True**

Adding a new C1OutPage

1. Right-click on **Going Places** and select **Add Page with Toolbar** from its context menu. A new **C1OutPage2** appears below the **C1OutPage1, Going Places**.
2. Select **C1OutPage2** from the drop-down list in the Properties window and set its **Text** property to **Browse**.
3. Right-click in the **C1ToolBar2** on the form and select **Append Item** from its context menu. The **Link to Command** designer appears.
4. In the **Link to Command** designer set the following properties:
 - o **Text** to Electronics
 - o **Name** to cmdBrowseElectronics
5. Select **C1Command** from the Create a new command link listbox and click **OK** to close the designer
6. Select the **C1CommandLink3**, Electronics, on the toolbar and set its **ButtonLook** property to **Text**.
7. Select **C1ToolBar2** from the Properties window drop-down list and set its properties to the following:
 - o **BackColor** to **Teal**
 - o **BackHiColor** to **MediumAquamarine**
 - o **ButtonAlign** to **Near**
8. Build and the run the application.

The **C1OutBar**'s Browse out page at run time appears like following image:



TopicBar Tasks

This section shows how to perform specific topic bar tasks.

Adding and Removing TopicBar Items

The following topics illustrate how to add and remove topic pages and topic links from the **C1TopicBar** control.

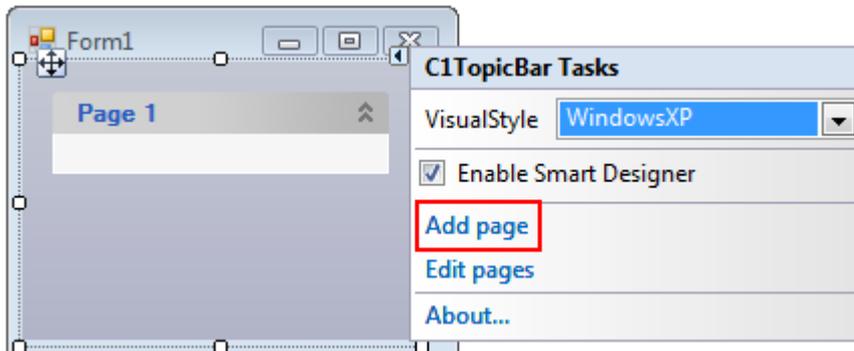
Adding Topic Pages to the TopicBar

There are five ways to add a page to the topic bar: you can use the smart tag, the floating toolbar, the context menu, the collection editor, or code. In this topic, you will learn how to add pages to the topic bar using each of these five methods.

Using the Smart Tag

Complete the following steps:

1. Navigate to the Toolbox and double-click the **C1TopicBar** icon. The **C1TopicBar** control is added to the form. Observe that one page, named **Page 1**, appears on the control by default.
2. Click **C1TopicBar**'s smart tag (☰) to open the **C1TopicBar Tasks** menu.
3. On the **C1TopicBar Tasks** menu, click **Add Page**.

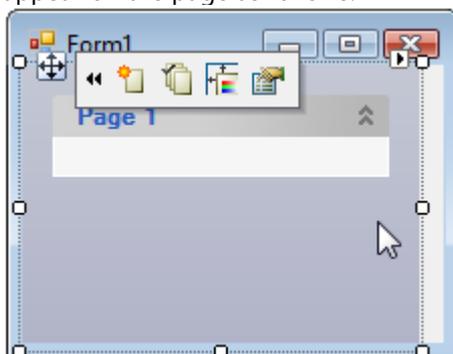


Page 2 is added to the **C1TopicBar** control.

Using the Floating Toolbar

Complete the following steps:

1. Navigate to the Toolbox and double-click the **C1TopicBar** icon. The **C1TopicBar** control is added to the form. Observe that one page, named **Page 1**, appears on the control by default.
2. Click the **C1TopicBar** control's chevron button (☰) to activate the floating toolbar. The floating toolbar will appear on the page as follows:



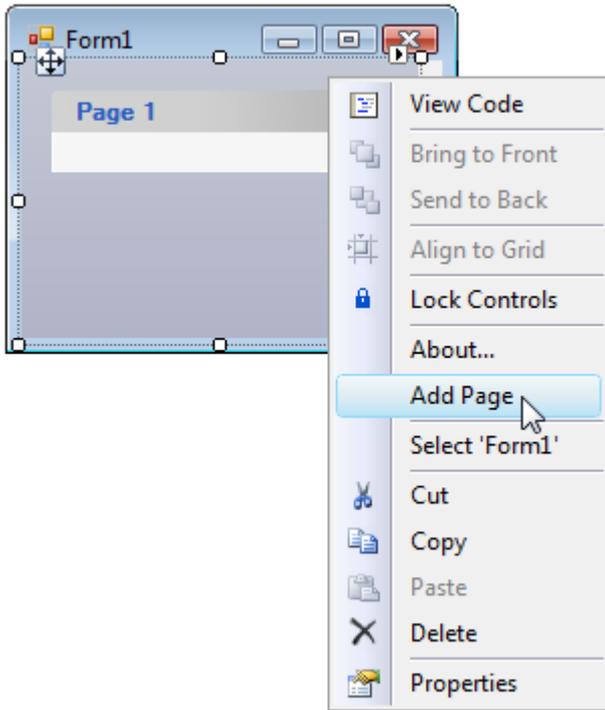
3. On the floating toolbar, select the **Add topic page** button .

Page 2 is added to the **C1TopicBar** control.

Using the Context Menu

Complete the following steps:

1. Navigate to the Toolbox and double-click the **C1TopicBar** icon. The **C1TopicBar** control is added to the form. Observe that one page, named **Page 1**, appears on the control by default.
2. Right-click on the **C1TopicBar** control to open its context menu.
3. From the context menu, select **Add Page**.

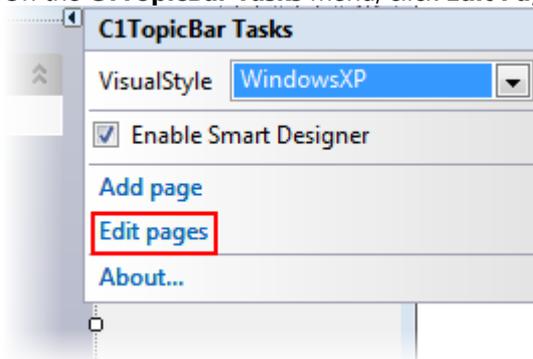


Page 2 is added to the [C1TopicBar](#) control.

Using the Collection Editor

Complete the following steps:

1. Navigate to the Toolbox and double-click the [C1TopicBar](#) icon. The [C1TopicBar](#) control is added to the form. Observe that one page, named **Page 1**, appears on the control by default.
2. Click [C1TopicBar](#)'s smart tag (Ⓜ) to open the **C1TopicBar Tasks** menu.
3. On the **C1TopicBar Tasks** menu, click **Edit Pages**.



The **C1TopicPage Collection Editor** opens.

4. Click **Add** to add a page to the collection.
5. Click **OK** to close the **C1TopicPage Collection Editor**.

Page 2 is added to the [C1TopicBar](#) control.

Using Code

Complete the following steps:

1. Navigate to the Toolbox and double-click the [C1TopicBar](#) icon. The [C1TopicBar](#) control is added to the form.

- Observe that one page, named **Page 1**, appears on the control by default.
2. Double-click the empty portion of the form to open Code view. Notice that a **Form_Load** event handler has been added to Code view.
 3. Import the following namespace into the project:

To write code in Visual Basic

```
Visual Basic
Imports Cl.Win.C1Command
```

To write code in C#

```
C#
using Cl.Win.C1Command;
```

4. Add the following code, which creates the new topic page and adds it to the topic bar, to the **Form_Load** event:

To write code in Visual Basic

```
Visual Basic
'Create new topic page object named "Page 2"
Dim c1TopicPage2 As New C1TopicPage("Page 2")
'Add new topic page to topic
c1TopicBar1.Pages.Add(c1TopicPage2)
```

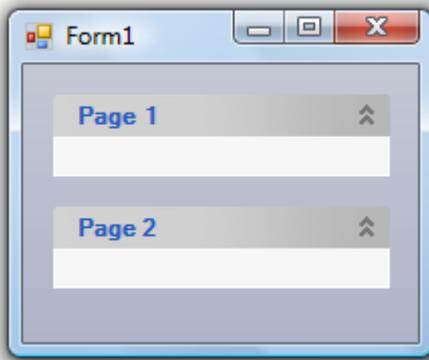
To write code in C#

```
C#
//Create new topic page object named "Page 2"
C1TopicPage c1TopicPage2 = new C1TopicPage("Page 2");
//Add new topic page to topic
c1TopicBar1.Pages.Add(c1TopicPage2);
```

5. Press F5 to build the project and observe that a new page, named **Page 2**, appears on the control.

This topic illustrates the following:

In this topic, you have learned to add a page to the [C1TopicBar](#) control using five separate methods. No matter which method you used, the final result of this topic will appear as follows:



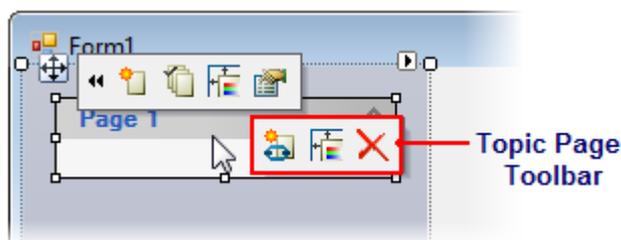
Removing a Topic Page from the TopicBar

There are three ways to remove a page from the topic bar: you can use the floating toolbar, the collection editor, or code. In this topic, you will learn how to remove pages from the topic bar using each of these three methods.

Using the Floating Toolbar

Complete the following steps:

1. Navigate to the Toolbox and double-click the [C1TopicBar](#) icon. The [C1TopicBar](#) control is added to the form. Observe that one page, named **Page 1**, appears on the control by default. If you'd like to add more pages to the control, see [Adding Topic Pages to the TopicBar](#).
2. Hover over **Page 1** with your cursor until the floating toolbar appears. The topic page's toolbar appears as follows:



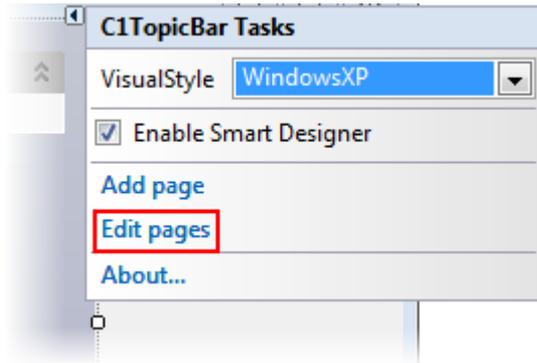
3. Click the **Delete topic page** button .

The topic page is removed from the control.

Using the Collection Editor

Complete the following steps:

1. Navigate to the Toolbox and double-click the [C1TopicBar](#) icon. The [C1TopicBar](#) control is added to the form. Observe that one page, named **Page 1**, appears on the control by default. If you'd like to add more pages to the control, see [Adding Topic Pages to the TopicBar](#).
2. Click [C1TopicBar](#)'s smart tag () to open the **C1TopicBar Tasks** menu.
3. On the **C1TopicBar Tasks** menu, click **Edit Pages**.



The **C1TopicPage Collection Editor** opens.

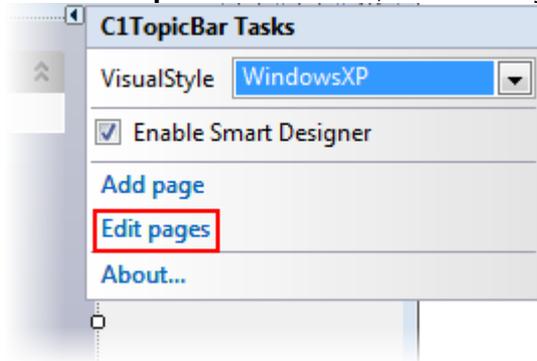
4. In the Members pane, select the page you wish to remove.
5. Click **Remove** to remove the page.
6. Click **OK** to close the **C1TopicPage Collection Editor**.

The page is removed from the control.

Using Code

Complete the following steps:

1. Navigate to the Toolbox and double-click the **C1TopicBar** icon. The **C1TopicBar** control is added to the form. Observe that one page, named **Page 1**, appears on the control by default. If you'd like to add more pages to the control, see [Adding Topic Pages to the TopicBar](#).
2. Click **C1TopicBar**'s smart tag (📌) to open the **C1TopicBar Tasks** menu.
3. On the **C1TopicBar Tasks** menu, click **Edit Pages**.



The **C1TopicPage Collection Editor** opens.

4. Select the page you wish to remove and then set its **Tag** property to "Tag1". Adding this tag gives the page a unique indicator that will allow you to find it later using the [FindPageByTag](#) method.
5. Click **OK** to close the **C1TopicPage Collection Editor**.
6. Double-click the empty portion of the form to open Code view. Notice that a **Form_Load** event handler has been added to Code view.
7. Add the following code, which finds the page and then removes it, to the **Form_Load** event:

To write code in Visual Basic

```

Visual Basic
'Find the page and assign it to a variable
Dim Page1 = c1TopicBar1.FindPageByTag("Tag1")
'Remove the specified page
c1TopicBar1.Pages.Remove(Page1)
    
```

To write code in C#

```
C#
//Find the page and assign it to a variable
var Page1 = c1TopicBar1.FindPageByTag("Tag1");
//Remove the specified page
c1TopicBar1.Pages.Remove(Page1);
```

8. Press F5 to build the project and observe that the page has been removed.

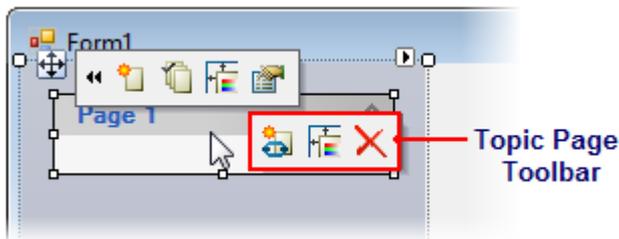
Adding Topic Links to Topic Pages

There are three ways to add topic links to the topic pages: you can use the floating toolbar, the collection editor, or code. In this topic, you will learn how to add links to topic pages using each of these three methods.

Using the Floating Toolbar

Complete the following steps:

1. Navigate to the Toolbox and double-click the **C1TopicBar** icon. The **C1TopicBar** control is added to the form. Observe that one page, named **Page 1**, appears on the control by default. If you'd like to add more pages to the control, see [Adding Topic Pages to the TopicBar](#).
2. Hover over **Page 1** with your cursor until the floating toolbar appears. The topic page's toolbar appears as follows:



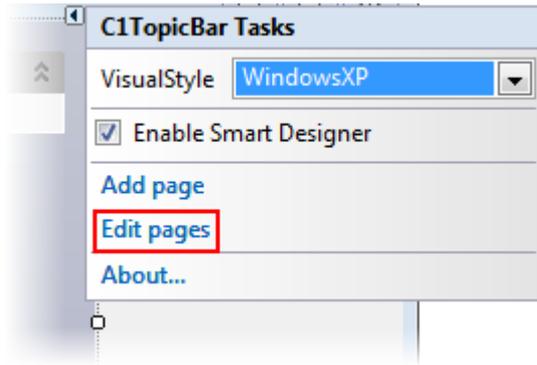
3. Click the **Add topic link** button .

The topic link, **Link 1**, is added to the topic page.

Using the Collection Editor

Complete the following steps:

1. Navigate to the Toolbox and double-click the **C1TopicBar** icon. The **C1TopicBar** control is added to the form. Observe that one page, named **Page 1**, appears on the control by default.
2. Click **C1TopicBar**'s smart tag (Ⓛ) to open the **C1TopicBar Tasks** menu.
3. On the **C1TopicBar Tasks** menu, click **Edit Pages**.



The **C1TopicPage Collection Editor** opens.

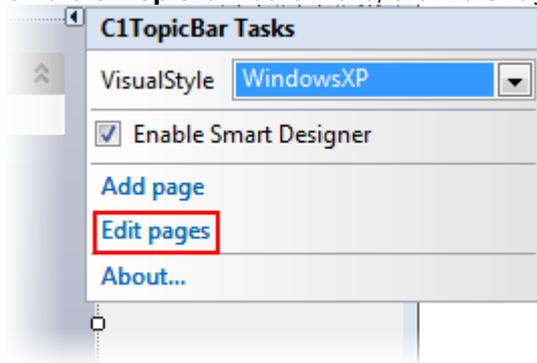
4. In the Members pane, select a page.
5. In the Properties grid, locate the **Links** property and click the ellipsis button . The **C1TopicLink Collection Editor** opens.
6. Click **Add** to add a topic link to the topic page.
7. Click **Close** to close the **C1TopicLink Collection Editor**.
8. Click **Close** to close the **C1TopicPage Collection Editor**.

The topic link, **Link 1**, is added to the topic page.

Using Code

Complete the following steps:

1. Navigate to the Toolbox and double-click the **C1TopicBar** icon. The **C1TopicBar** control is added to the form. Observe that one page, named **Page 1**, appears on the control by default. If you'd like to add more pages to the control, see [Adding Topic Pages to the TopicBar](#).
2. Click **C1TopicBar**'s smart tag  to open the **C1TopicBar Tasks** menu.
3. On the **C1TopicBar Tasks** menu, click **Edit Pages**.



The **C1TopicPage Collection Editor** opens.

4. Select the page you wish to remove and then set its **Tag** property to "Tag1". Adding this tag gives the page a unique indicator that will allow you to find it later using the [FindPageByTag](#) method.
5. Click **OK** to close the **C1TopicPage Collection Editor**.
6. Double-click the empty portion of the form to open Code view. Notice that a **Form_Load** event handler has been added to Code view.
7. Import the following namespace into the project:

To write code in Visual Basic

```
Visual Basic
Imports Cl.Win.C1Command
```

To write code in C#

C#

```
using C1.Win.C1Command;
```

8. Add the following code to the **Form_Load** event. This code will find the page and assign it to a variable, create a new topic link object, and then add the topic link object to the topic page.

To write code in Visual Basic

Visual Basic

```
'Find the topic page and assign it to a variable
Dim c1TopicPage1 = c1TopicBar1.FindPageByTag("Tag1")
'Create a new topic link and assign it a name
Dim c1TopicLink1As New C1TopicLink("Link 1")
'Add the new topic link to the topic page
c1TopicPage1.Links.Add(c1TopicLink1)
```

To write code in C#

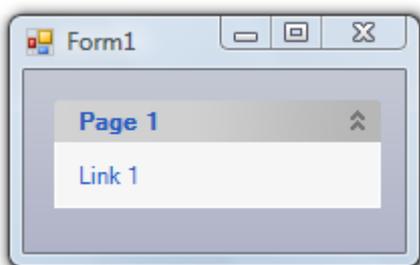
C#

```
//Find the topic page and assign it to a variable
var c1TopicPage1 = c1TopicBar1.FindPageByTag("Tag1");
//Create a new topic link and assign it a name
C1TopicLink c1TopicLink1 = new C1TopicLink("Link 1");
//Add the new topic link to the topic page
c1TopicPage1.Links.Add(c1TopicLink1);
```

9. Press F5 to build the project and observe that the link has been added to the page.

This topic illustrates the following:

In this topic, you learned how to add topic links to topic pages using three different methods. The result of this topic will look as follows:



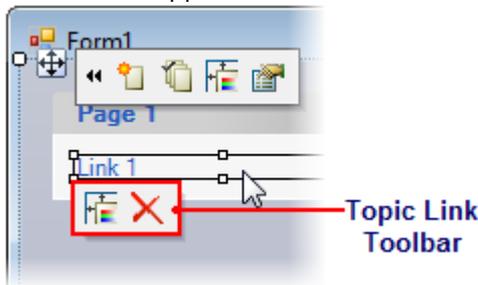
Removing Topic Links from Topic Pages

You can remove topic links from topic pages using the floating toolbar, the collection editor, and code. In this topic, you will learn how to remove topic links using each of the aforementioned methods. This topic assumes that you have a **C1TopicBar** control containing at least one topic link (see [Adding Topic Links to Topic Pages](#)) on your page.

Using the Floating Toolbar

Complete the following steps:

1. Using your cursor, hover over the topic link you wish to remove until its floating toolbar appears. The topic link's toolbar appears similar to the following:



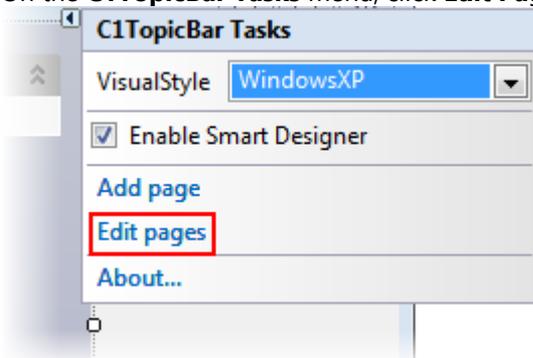
2. Click the **Delete topic link**  button.

The topic link is removed from the topic page.

Using the Collection Editor

Complete the following steps:

1. Click **C1TopicBar**'s smart tag  to open the **C1TopicBar Tasks** menu.
2. On the **C1TopicBar Tasks** menu, click **Edit Pages**.



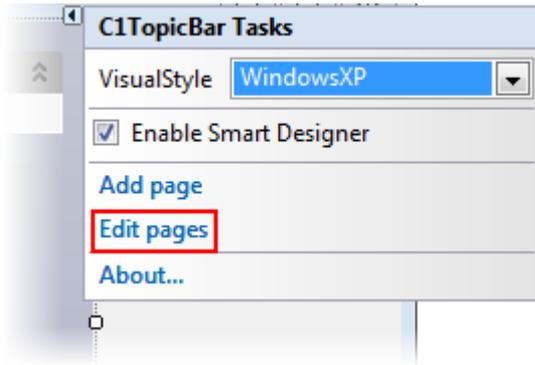
The **C1TopicPage Collection Editor** opens.

3. In the Members pane, select a page.
4. In the Properties grid, locate the **Links** property and click the ellipsis button . The **C1TopicLink Collection Editor** opens.
5. In the Members pane, select the link you wish to remove.
6. Click **Remove** to remove the link.
7. Click **OK** to close the **C1TopicPage Collection Editor**.
8. Click **OK** to close the **C1TopicLink Collection Editor**. The topic link is removed from the topic page.

Using Code

Complete the following steps:

1. Click **C1TopicBar**'s smart tag  to open the **C1TopicBar Tasks** menu.
2. On the **C1TopicBar Tasks** menu, click **Edit Pages**.



The **C1TopicPage Collection Editor** opens.

3. In the Members pane, select the page that holds the link you wish to remove.
4. In the Properties grid, set the page's **Tag** property to "PageWithLink". This will allow you to find the page in the code by using the [FindPageByTag](#) method.
5. In the Properties grid, locate the **Links** property and click the ellipsis button . The **C1TopicLink Collection Editor** opens.
6. Select the link you wish to remove in code and, in the Properties grid, set the link's **Tag** property to "LinkToRemove". This will give the link a unique indicator, which will allow you to find it in code using the [FindLinkByTag](#) method.
7. Click **OK** to close the **C1TopicPage Collection Editor**.
8. Click **OK** to close the **C1TopicLink Collection Editor**.
9. Double-click the empty portion of the form to open Code view. Notice that a **Form_Load** event handler has been added to Code view.
10. Add the following code to the **Form_Load** event. This code will find the page and link, assign them to variables, and then will remove the link from the page.

To write code in Visual Basic

Visual Basic

```
'Find page and assign to variable
Dim c1TopicPage1 = c1TopicBar1.FindPageByTag("PageWithLink")
'Find link and assign to variable
Dim c1TopicLink1 = c1TopicPage1.FindLinkByTag("LinkToRemove")
'Remove link from page
c1TopicPage1.Links.Remove(c1TopicLink1)
```

To write code in C#

C#

```
//Find page and assign to variable
var c1TopicPage1 = c1TopicBar1.FindPageByTag("PageWithLink");
//Find link and assign to variable
var c1TopicLink1 = c1TopicPage1.FindLinkByTag("LinkToRemove");
//Remove link from page
c1TopicPage1.Links.Remove(c1TopicLink1);
```

11. Press F5 to build the project.

Customizing the Appearance

The following topics illustrate how to modify the appearance of the [C1TopicBar](#) control.

Adding a Background Image

You can add a background image to the `C1TopicBar` control using the `BackgroundImage` property. In this topic, you will learn to set this property using the Properties window and code.

Using the Properties Window

Complete the following steps:

1. Navigate to the Toolbox and double-click the `C1TopicBar` icon to add the `C1TopicBar` control to your form.
2. Right-click the `C1TopicBar` control to open its context menu and then select **Properties**. The Properties window opens with the `C1TopicBar` control's properties in focus.
3. Locate the `BackgroundImage` property and click its ellipsis button  to open the **Select Resource** dialog box.
4. Select the **Local resource** radio button and then click **Import**. The **Open** dialog box opens.
5. Navigate to the folder holding your background image, select the image, and then click **Open** to import the image.
6. The **Open** dialog box closes, returning you to the **Select Resource** dialog box.
7. Click **OK** to close the **Select Resource** dialog box.

Your background image is added to the `C1TopicBar` control.

Using Code

Complete the following steps:

1. Navigate to the Toolbox and double-click the `C1TopicBar` icon to add the `C1TopicBar` control to your form.
2. Double-click the empty portion of the form to open Code view. Notice that a **Form_Load** event handler has been added to Code view.
3. Set the background image by adding the following code (replacing `C:\YourImage.jpg` with your own path and image name) to the **Form_Load** event:

To write code in Visual Basic

Visual Basic

```
c1TopicBar1.BackgroundImage = System.Drawing.Image.FromFile("C:\YourImage.jpg")
```

To write code in C#

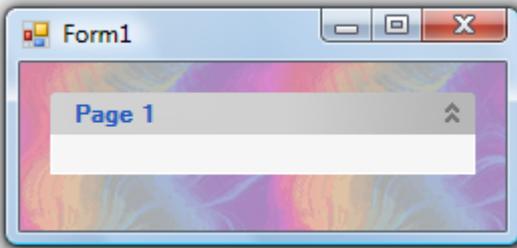
C#

```
c1TopicBar1.BackgroundImage =  
System.Drawing.Image.FromFile(@"C:\YourImage.jpg");
```

4. Press F5 to build the project.

This topic illustrates the following:

In this topic, you learned how to add a background image using the Properties window and code. The following image depicts a `C1TopicBar` control with a custom background image.

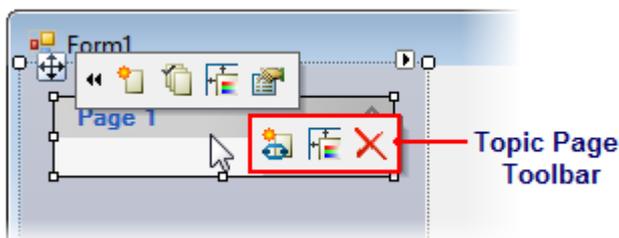


Adding an Icon to a Topic Page

In this topic, you will create an image list and then an image in that list to a topic page.

Complete the following steps:

1. In the Toolbox, double-click the **C1TopicBar** icon to add the **C1TopicBar** control to your project.
2. In the Toolbox, double click the **Image List** icon to add the **Image List** component to your project.
3. Click the **ImageList** control's smart tag (🔗) to open the **Image List Tasks** menu.
4. Select **Choose Images** to open the **Images Collection Editor**.
5. Click **Add** to open the **Open** dialog box and complete the following:
 1. Navigate to the image you wish to use as an icon.
 2. Select the image.
 3. Press **OK** to close the **Open** dialog box and to return to the **Images Collection Editor**. Observe that the image is numbered "0".
6. Press **OK** to close the **Images Collection Editor**.
7. Right-click the **C1TopicBar** control to open its context menu and then select **Properties** from the list. The Properties window opens with the **C1TopicBar** control's properties in focus.
8. Locate the **ImageList** property, click its drop-down arrow, and select **imageList1** from the list. This loads the image list into the control so that the control's topic pages can pull images from the list.
9. Hover over **Page 1** with your cursor until the floating toolbar appears. The topic page's toolbar appears as follows:

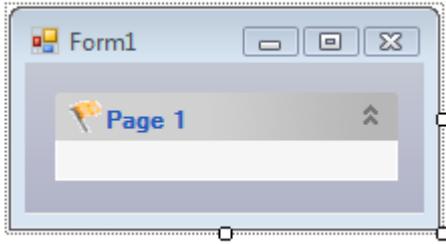


10. Click the **Edit topic page appearance** button  to open the **C1TopicPage Properties** editor.
11. Click the **Image Index** drop-down arrow and select **0** from the list.

The icon is added to **Page 1**.

This topic illustrates the following:

In this topic, you added an icon to a topic page. The image below depicts a page with an icon.

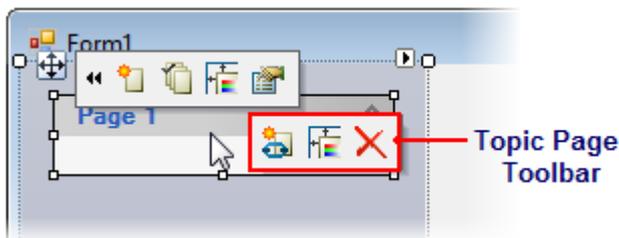


Adding an Icon to a Topic Link

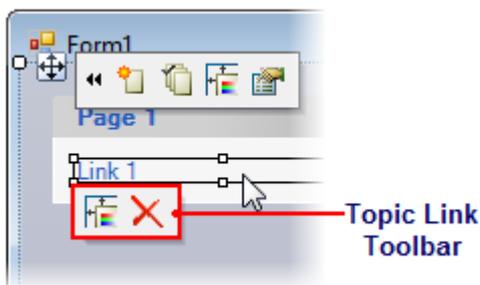
In this topic, you will create an image list and then add an image in that list to a topic link.

Complete the following steps:

1. In the Toolbox, double-click the **C1TopicBar** icon to add the **C1TopicBar** control to your page.
2. In the Toolbox, double click the **Image List** icon to add the **Image List** component to your page.
3. Click the **ImageList** control's smart tag (🔗) to open the **Image List Tasks** menu.
4. Select **Choose Images** to open the **Images Collection Editor**.
5. Click **Add** to open the **Open** dialog box and complete the following:
 1. Navigate to the image you wish to use as an icon.
 2. Select the image.
 3. Press **OK** to close the **Open** dialog box and to return to the **Images Collection Editor**. Observe that the image is numbered "0".
6. Press **OK** to close the **Images Collection Editor**.
7. Hover over **Page 1** with your cursor until the floating toolbar appears. The topic page's toolbar appears as follows:



8. Click the **Add topic link** button (🔗). **Link 1** is added underneath **Page 1**.
9. On the topic page toolbar, click the **Edit topic page appearance** button (🔧) to open the **C1TopicPage Properties** editor.
10. Click the **Image List** drop-down arrow and select **imageList1** from the list. This will load the image list so that topic page's links can pull images from the list.
11. Hover over **Link 1** with your cursor until the floating toolbar appears. The topic link's toolbar appears similar to the following:



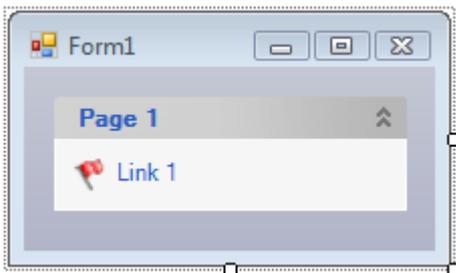
12. Click the **Edit topic link appearance** button (🔧) to open the **C1TopicLink Properties** editor.

13. Click the **Image Index** drop-down arrow and select **0** from the list.

The icon is added to **Link 1**.

This topic illustrates the following:

In this topic, you added an icon to a link page. The image below depicts a topic link with an icon.



Changing the Visual Style

The [C1TopicBar](#) control carries several visual styles that can be added to the control by setting the [VisualStyle](#) property. This topic illustrates how change the visual style using the smart tag, the Properties window, and code.

Using the Smart Tag

Complete the following steps:

1. Navigate to the Toolbox and double-click the [C1TopicBar](#) icon to add the control to your page.
2. Click [C1TopicBar](#)'s smart tag (🔗) to open the **C1TopicBar Tasks** menu.
3. Click the **VisualStyle** drop-down arrow and select a style from the list. For this example, select **Office2003Olive**.

The selected visual style is applied to the control.

Using the Properties Window

Complete the following steps:

1. Navigate to the Toolbox and double-click the [C1TopicBar](#) icon to add the control to your page.
2. Right-click the [C1TopicBar](#) control to open its context menu and then select **Properties**. The Properties window opens with the [C1TopicBar](#) control's properties in focus.
3. Locate the [VisualStyle](#) property, click its drop-down arrow, and select a visual style. For this example, select **Office2003Olive**.

The selected visual style is applied to the control.

Using Code

Complete the following steps:

1. Navigate to the Toolbox and double-click the [C1TopicBar](#) icon.
2. Double-click the empty portion of the form to open Code view. Notice that a **Form_Load** event handler has

been added to Code view.

3. Import the following namespace into the project:

To write code in Visual Basic

```
Visual Basic
Imports Cl.Win.C1Command
```

To write code in C#

```
C#
using Cl.Win.C1Command;
```

4. To set the visual style, add the following code, which sets the visual style to **Office2003Olive**, to the **Form_Load** event:

To write code in Visual Basic

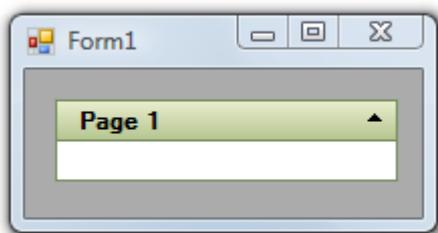
```
Visual Basic
c1TopicBar1.VisualStudio =VisualStyle.Office2003Olive
```

To write code in C#

```
C#
c1TopicBar1.VisualStudio =VisualStyle.Office2003Olive;
```

This topic illustrates the following:

In this topic, you learned how to change the visual style using three different methods. The result of this topic will look as follows:



Customizing the Expand/Collapse Behaviors

The topics in this section illustrate how to alter the expand/collapse behaviors for [C1TopicBar](#) pages.

Changing the Expand / Collapse Animation

By default, the animation for the expand/collapse behavior of the [C1TopicBar](#) control is set to **System**, meaning that the behavior follows the settings of a user's operating system. If you'd rather have control over this, you can turn the animation on by setting to [Animation](#) property to **On**, or you can turn the animation off by setting the [Animation](#) to **Off**.

Using the Properties Window

Complete the following steps:

1. Navigate to the Toolbox and double-click the `C1TopicBar` icon. The `C1TopicBar` control is added to the form. Observe that one page, named **Page 1**, appears on the control by default. If you'd like to add more pages to the control, see [Adding Topic Pages to the TopicBar](#).
2. Right-click the `C1TopicBar` control to open its context menu and then select **Properties**.
3. The Properties window opens with the `C1TopicBar` control's properties in focus.
4. Locate the [Animation](#) property, click its drop-down arrow, and select a setting from the list.

Using Code

Complete the following steps:

1. Navigate to the Toolbox and double-click the `C1TopicBar` icon. The `C1TopicBar` control is added to the form. Observe that one page, named **Page 1**, appears on the control by default. If you'd like to add more pages to the control, see [Adding Topic Pages to the TopicBar](#).
2. Double-click the empty portion of the form to open Code view. Notice that a **Form_Load** event handler has been added to Code view.
3. Import the following namespace into the project:

To write code in Visual Basic

```
Visual Basic
Imports Cl.Win.C1Command
```

To write code in C#

```
C#
using Cl.Win.C1Command;
```

4. Complete ONE of the following:
 - o To turn the animation on, add the following code to the **Form_Load** event

To write code in Visual Basic

```
Visual Basic
c1TopicBar1.Animation = C1AnimationEnum.On
```

To write code in C#

```
C#
c1TopicBar1.Animation = C1AnimationEnum.On
```

- o To turn the animation off, add the following code to the **Form_Load** event

To write code in Visual Basic

```
Visual Basic
c1TopicBar1.Animation = C1AnimationEnum.Off
```

To write code in C#

C#

```
c1TopicBar1.Animation = C1AnimationEnum.Off;
```

5. Press F5 to build the project.

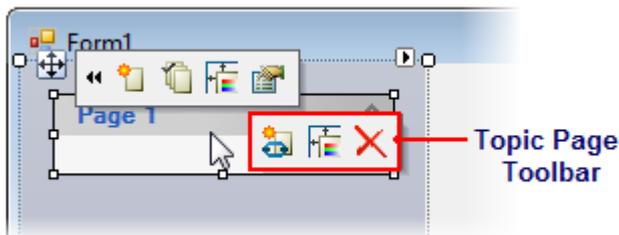
Creating a Collapsed Topic Page

By default, a page added to the [C1TopicBar](#) control will be expanded. You can create a collapsed page by setting the [Collapsed](#) property to **True**. In this topic, you will learn how to set the [Collapsed](#) property using the floating toolbar, the collection editor and code.

Using the Floating Toolbar

Complete the following steps:

1. Navigate to the Toolbox and double-click the [C1TopicBar](#) icon. The [C1TopicBar](#) control is added to the form. Observe that one page, named **Page 1**, appears on the control by default. If you'd like to add more pages to the control, see [Adding Topic Pages to the TopicBar](#).
2. Hover over **Page1** (or another page of your choice) with your cursor until the floating toolbar appears. The topic page's toolbar appears as follows:

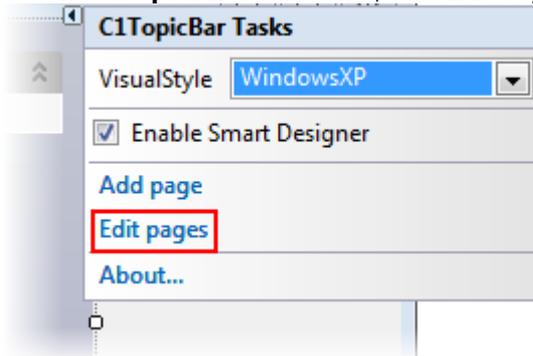


3. Click the **Edit topic page appearance** button  to open the **C1TopicPage Properties** editor.
4. Select the **Collapsed** check box and observe that the page is now collapsed.

Using the Collection Editor

Complete the following steps:

1. Navigate to the Toolbox and double-click the [C1TopicBar](#) icon. The [C1TopicBar](#) control is added to the form. Observe that one page, named **Page 1**, appears on the control by default. If you'd like to add more pages to the control, see [Adding Topic Pages to the TopicBar](#).
2. Click [C1TopicBars](#) smart tag (D) to open the **C1TopicBar Tasks** menu.
3. On the **C1TopicBar Tasks** menu, click **Edit Pages**.



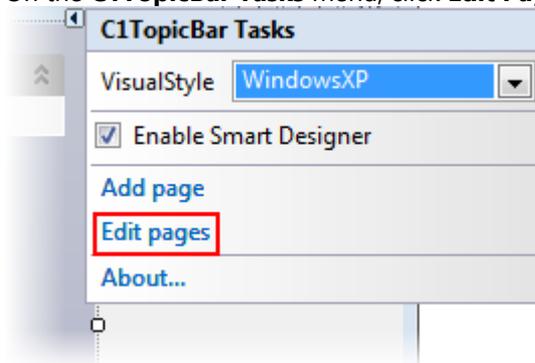
The **C1TopicPage Collection Editor** opens.

4. In the Members pane, select the page you wish to have collapsed.
5. In the Properties grid, set the **Collapsed** property to **True**.
6. Press **OK** to close the **C1TopicPage Collection Editor** and observe that the topic page is collapsed.

In Code

Complete the following steps:

1. Navigate to the Toolbox and double-click the **C1TopicBar** icon. The **C1TopicBar** control is added to the form. Observe that one page, named **Page 1**, appears on the control by default. If you'd like to add more pages to the control, see [Adding Topic Pages to the TopicBar](#).
2. Click **C1TopicBar**'s smart tag (📌) to open the **C1TopicBar Tasks** menu.
3. On the **C1TopicBar Tasks** menu, click **Edit Pages**.



The **C1TopicPage Collection Editor** opens.

4. In the Members pane, select the page you wish to have collapsed.
5. In the Properties grid, set the **Tag** property to "PageToCollapse". This tag is a unique indicator that will allow you to find the appropriate page in code using the [FindPageByTag](#) method.
6. Click **OK** to close the **C1TopicPage Collection Editor**.
7. Double-click the empty portion of the form to open Code view. Notice that a **Form_Load** event handler has been added to Code view.
8. Add the following code to the **Form_Load** event. This code will find the page, assign it to a variable, and then set the page's **Collapsed** property to **True**.

To write code in Visual Basic

Visual Basic

```
Dim c1TopicPage1 = c1TopicBar1.FindPageByTag("PageToCollapse")
c1TopicPage1.Collapsed = True
```

To write code in C#

C#

```
var c1TopicPage1 = c1TopicBar1.FindPageByTag("PageToCollapse");
c1TopicPage1.Collapsed = true;
```

9. Press F5 to build the project.

Using Topic Bar ToolTips

The topics in this section teach you how to use and manipulate **C1TopicBar** ToolTips. You will learn how to create

ToolTips for topic pages and topic links, as well as learn how to disable all ToolTips for the control.

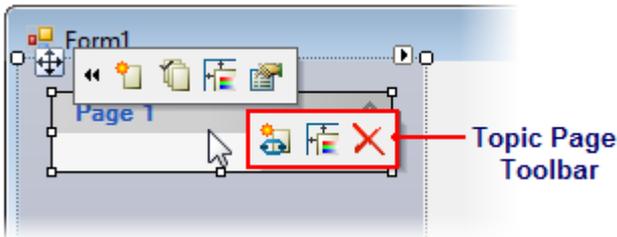
Adding ToolTips to Topic Pages

You can add ToolTips to topic pages using the floating toolbar, the collection editor, and code.

Using the Floating Toolbar

Complete the following steps:

1. Navigate to the Toolbox and double-click the **C1TopicBar** icon. The **C1TopicBar** control is added to the form. Observe that one page, named **Page 1**, appears on the control by default. If you'd like to add more pages to the control, see [Adding Topic Pages to the TopicBar](#).
2. Hover over **Page1** (or another page of your choice) with your cursor until the floating toolbar appears. The topic page's toolbar appears as follows:

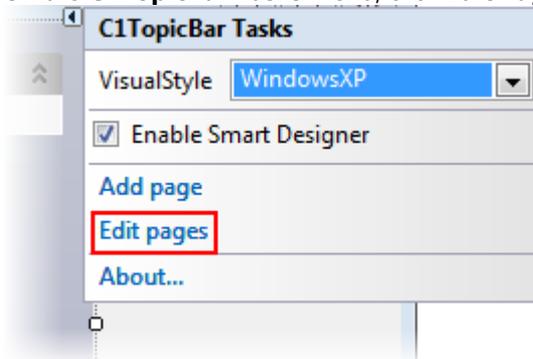


3. Click the Edit topic page appearance button  to open the **C1TopicPage Properties** editor.
4. In the **ToolTip** text box, enter "I am a topic page ToolTip!".
5. Press F5 to build the project.
6. Hover over the page you added the ToolTip to and observe that a ToolTip appears.

Using the Collection Editor

Complete the following steps:

1. Navigate to the Toolbox and double-click the **C1TopicBar** icon. The **C1TopicBar** control is added to the form. Observe that one page, named **Page 1**, appears on the control by default. If you'd like to add more pages to the control, see [Adding Topic Pages to the TopicBar](#).
2. Click **C1TopicBar**'s smart tag  to open the **C1TopicBar Tasks** menu.
3. On the **C1TopicBar Tasks** menu, click **Edit Pages**.



The **C1TopicPage Collection Editor** opens.

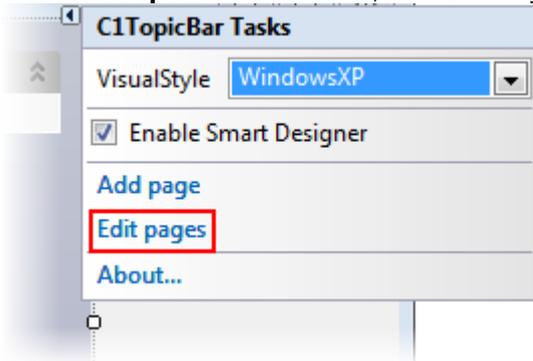
4. In the Members pane, select the page you wish to add the ToolTip to.
5. In the Properties grid, set the **ToolTipText** property to "I am a topic page ToolTip!".
6. Click **OK** to close the **C1TopicPage Collection Editor**.

7. Press F5 to build the project.
8. Hover over the page you added the ToolTip to and observe that a ToolTip appears.

Using Code

Complete the following steps:

1. Navigate to the Toolbox and double-click the **C1TopicBar** icon. The **C1TopicBar** control is added to the form. Observe that one page, named **Page 1**, appears on the control by default. If you'd like to add more pages to the control, see [Adding Topic Pages to the TopicBar](#).
2. Click **C1TopicBar**'s smart tag (⏏) to open the **C1TopicBar Tasks** menu.
3. On the **C1TopicBar Tasks** menu, click **Edit Pages**.



The **C1TopicPage Collection Editor** opens.

4. In the Members pane, select the page you wish to dynamically add the ToolTip to.
5. In the Properties grid, set the **Tag** property to "PageWithToolTip". In a later step, you will use this tag to find the topic page.
6. Click **OK** to close the **C1TopicPage Collection Editor**.
7. Double-click the empty portion of the form to open Code view. Notice that a **Form_Load** event handler has been added to Code view.
8. Import the following namespace into the project:

To write code in Visual Basic

```
Visual Basic
Imports Cl.Win.C1Command
```

To write code in C#

```
C#
using Cl.Win.C1Command;
```

9. Add the following code to the **Form_Load** event. This code will find the page, assign it to a variable, and then set the page's **ToolTipText** property.

To write code in Visual Basic

```
Visual Basic
Dim c1TopicPage1 = c1TopicBar1.FindPageByTag("PageWithToolTip")
c1TopicPage1.ToolTipText = "I am a topic page ToolTip!"
```

To write code in C#

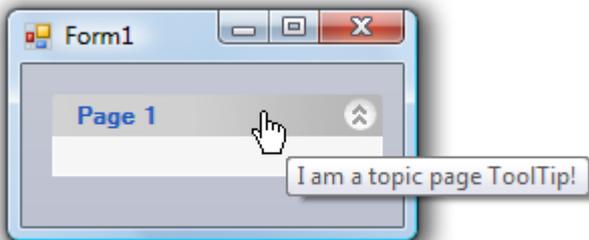
C#

```
var c1TopicPage1 = c1TopicBar1.FindPageByTag("PageWithToolTip");
c1TopicPage1.ToolTipText = "I am a topic page ToolTip!";
```

10. Press F5 to build the project.
11. Hover over the page you added the ToolTip to and observe that a ToolTip appears.

This topic illustrates the following:

In this topic, you learned how to add a ToolTip to a topic page using the floating toolbar, the collection editor, and code. No matter which method you used in this topic, the result will resemble the following:



Adding ToolTips to Link Pages

You can add ToolTips to topic links using the floating toolbar, the collection editor, and code.

Using the Floating Toolbar

Complete the following steps:

1. Navigate to the Toolbox and double-click the [C1TopicBar](#) icon. The [C1TopicBar](#) control is added to the form. Observe that one page, named **Page 1**, appears on the control by default.
2. Add a topic link to **Page 1** (see [Adding Topic Links to Topic Pages](#)).
3. Hover over the new link until its floating toolbar appears and then click the **Edit topic link appearance** button



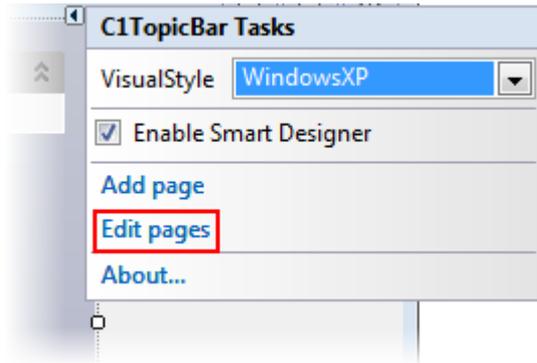
The **C1TopicLink Properties** editor opens.

4. In the Tooltip text box, enter "I am a topic link ToolTip!".
5. Press F5 to build the project.
6. Hover over the link you added the ToolTip to and observe that a ToolTip appears.

Using the Collection Editor

Complete the following steps:

1. Navigate to the Toolbox and double-click the [C1TopicBar](#) icon. The [C1TopicBar](#) control is added to the form. Observe that one page, named **Page 1**, appears on the control by default.
2. Click [C1TopicBar](#)'s smart tag (🔗) to open the **C1TopicBar Tasks** menu.
3. On the **C1TopicBar Tasks** menu, click **Edit Pages**.



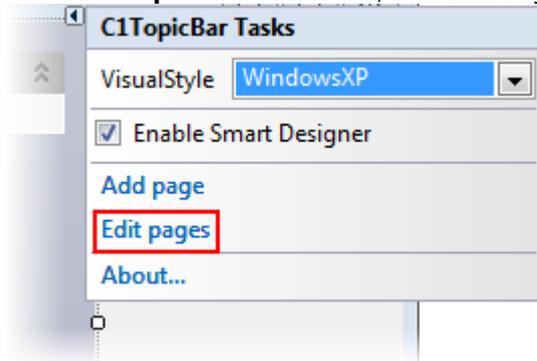
The **C1TopicPage Collection Editor** opens.

4. In the Properties grid, locate the **Links** property and click the ellipsis button . The **C1TopicLink Collection Editor** opens.
5. In the Members pane, select the link you wish to add the Tooltip to.
6. In the Properties grid, set the **ToolTipText** property to "I am a topic link Tooltip!".
7. Click **OK** to close the **C1TopicLink Collection Editor**.
8. Click **OK** to close the **C1TopicPage Collection Editor**.
9. Press F5 to build the project.
10. Hover over the link you added the Tooltip to and observe that a Tooltip appears.

Using Code

Complete the following steps:

1. Navigate to the Toolbox and double-click the **C1TopicBar** icon. The **C1TopicBar** control is added to the form. Observe that one page, named **Page 1**, appears on the control by default.
2. Click **C1TopicBar**'s smart tag  to open the **C1TopicBar Tasks** menu.
3. On the **C1TopicBar Tasks** menu, click **Edit Pages**.



The **C1TopicPage Collection Editor** opens.

4. In the Members pane, select a page.
5. In the Properties grid, locate the **Links** property and click the ellipsis button . The **C1TopicLink Collection Editor** opens.
6. In the Members pane, select the link you wish to add the Tooltip to.
7. In the Properties grid, set the **Tag** property to "LinkToolTip". In a later step, you will use this tag to find the topic link.
8. Click **OK** to close the **C1TopicLink Collection Editor**.
9. Click **OK** to close the **C1TopicPage Collection Editor**.
10. Double-click the empty part of the form to open Code view. Notice that a **Form_Load** event handler has been added to Code view.
11. Add the following code to the **Form_Load** event. This code will find the page, assign it to a variable, and then set the page's **ToolTipText** property.

To write code in Visual Basic

Visual Basic

```
Dim c1TopicLink1 = c1TopicBar1.FindLinkByTag("LinkToolTip")
c1TopicLink1.ToolTipText = "I am a topic link ToolTip!"
```

To write code in C#

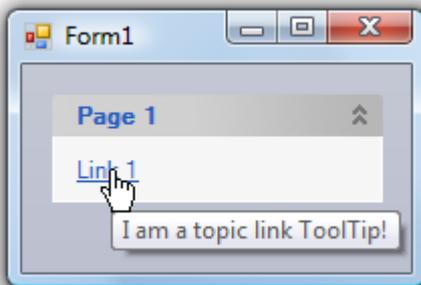
C#

```
var c1TopicLink1 = c1TopicBar1.FindLinkByTag("LinkToolTip");
c1TopicLink1.ToolTipText = "I am a topic link ToolTip!";
```

12. Press F5 to build the project.
13. Hover over the link you added the ToolTip to and observe that a ToolTip appears.

This topic illustrates the following:

In this topic, you learned how to add a ToolTip to a topic page using the floating toolbar, the collection editor, and code. No matter which method you used in this topic, the result will resemble the following:



Disabling ToolTips

You can disable all ToolTips by setting the `ShowToolTips` property to **False**.

Using the Properties Window

Complete the following steps:

1. Right-click the `C1TopicBar` control to open its context menu.
2. Select **Properties** to open the Properties window.
3. Locate the `ShowToolTips` property, click its drop-down arrow, and select **False**.

In Code

Complete the following steps:

1. Double-click in the empty part of the form to open Code view. Notice that a **Form_Load** event handler has been added to Code view.
2. To set the `ShowToolTips` property to **False**, add the following code to the **Form_Load** event:

To write code in Visual Basic

Visual Basic

```
C1TopicBar1.ShowToolTips = False
```

To write code in C#

C#

```
c1TopicBar1.ShowToolTips = false;
```

3. Press F5 to build the project.